Ronan Houée *10/02/2025*

Allocation en ligne de tâches

Sommaire :

- Travail réalisé

- Résultats

- Conclusion

Travail réalisé

Paramétrage :

La partie Une concerne principalement la modélisation du problème, de l'environnement, des tâche…

Au démarrage de l'application, une liste de commandes s'affiche, sélectionnables par l'input d'un numéro.

On peut choisir de modifier les paramètres de l'environnement avec l'option 0. Par défaut, les paramètres sont initialisés par la fonction :

```
parse_args_default()
```

On peut donc les modifier par input un à un, grâce à la fonction

```
parse_args()
```

Algorithme

Le déroulement de l'algorithme principal se trouve dans la fonction

```
main(ord, taxis_ct, env_size, task_freq, task_method, tasks_ct, total_time,
taxis_random_location = False, plots = True):
```

Elle simule l'exécution de l'environnement sur la durée totale, génère les tâches quand nécessaire et les alloue aux taxis selon l'ordonnancement ord en paramètre.

Il y a une option par type d'ordonnancement dans le menu.

Pour permettre les comparaisons entre ordonnancements, une autre fonction doit être utilisée :

```
compare_main(ord1, ord2, taxis_ct, env_size, task_freq, task_method, tasks_ct, total_time, taxi_random_location)
```

Celle-ci permet d'assurer que les tâches générées aléatoirement sont bien les mêmes pour les deux ordonnancements.

Pour obtenir des données sur un certain nombre de répétitions, on dispose de l'option 10, utilisant la fonction :

```
def multi_main(reps, ord, taxis_ct, env_size, task_freq, task_method, tasks_ct, total_time, taxis_random_location = False):
```

Il existe plusieurs types d'ordonnancement, qui héritent tous de la classe abstraite ordonnancement :

Pour la partie 1, un ordonnancement retournant l'optimal mais avec un seul taxi

Pour la partie 2, un ordonnancement DCOP simple

Pour la partie 3, des ordonnancements SSI, SSI avec regrets et PSI.

DCOP Simple

On modélise le DCOP comme un problème d'assignement : chaque tâche correspond à une variable, et peut prendre une valeur possible par taxi différent.

Le coût de chaque assignement de tâche est :

Pour chaque tache, le coût qu'aurait le taxi depuis sa position actuelle à se déplacer jusqu'au point initial de la tâche additionné au coût du déplacement du point initial vers le point d'arrivé.

(Voir le calcul du dictionnaire costs dans algo_assign de l'ordonnancement DCOP_Simple)

Cet ordonnancement ne prend donc pas en compte les synergies négatives ou positives entre les tâches.

L’ordonnancement écrit un fichier .yaml, utilisable sous pydcop pour trouver une allocation.

Pour les tests, ne possédant pas de moyen d’exécuter le résoudre automatiquement le DCOP une fois le fichier pydcop_file.yaml écrit, je n’ai réalisé que quelques tests à la main, pas assez pour en tirer des conclusions.

Partie 3

Pour cette partie, l’heuristique de l’insertion est utilisée.

Des options du menu permettent de comparer PSI avec SSI et SSI avec SSI regrets.

Résultats

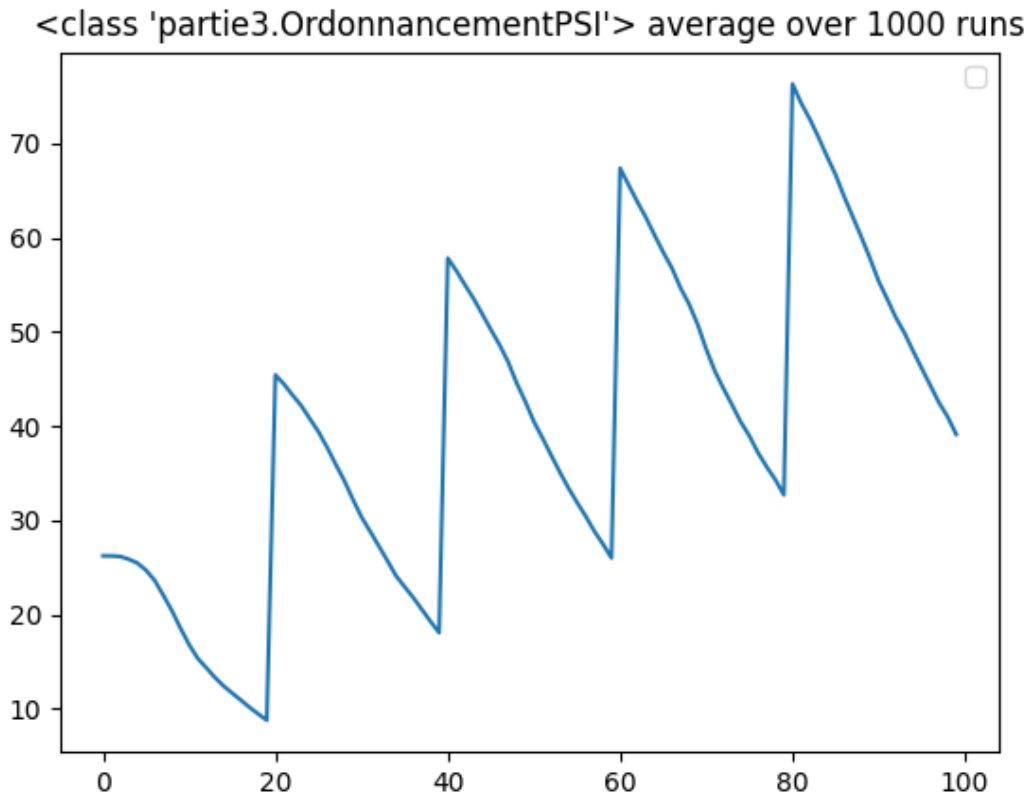
Pour comparer les différents algorithmes, on propose plusieurs métriques :

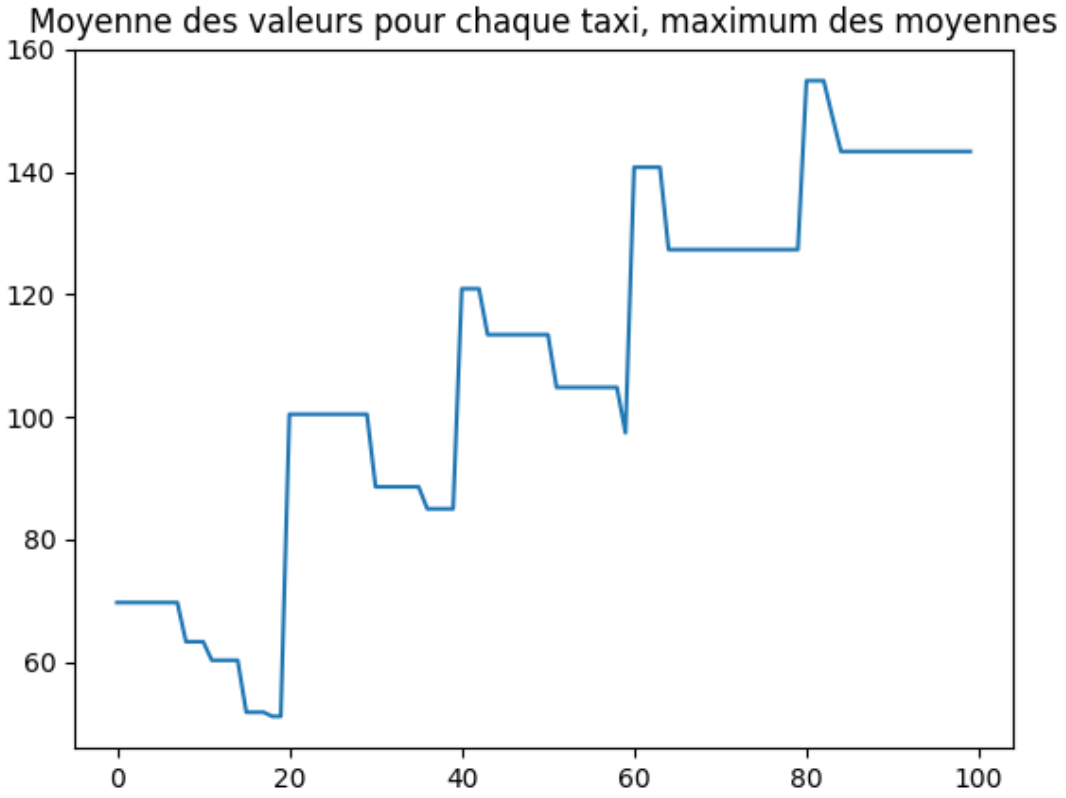
Sur 1000 exécutions :

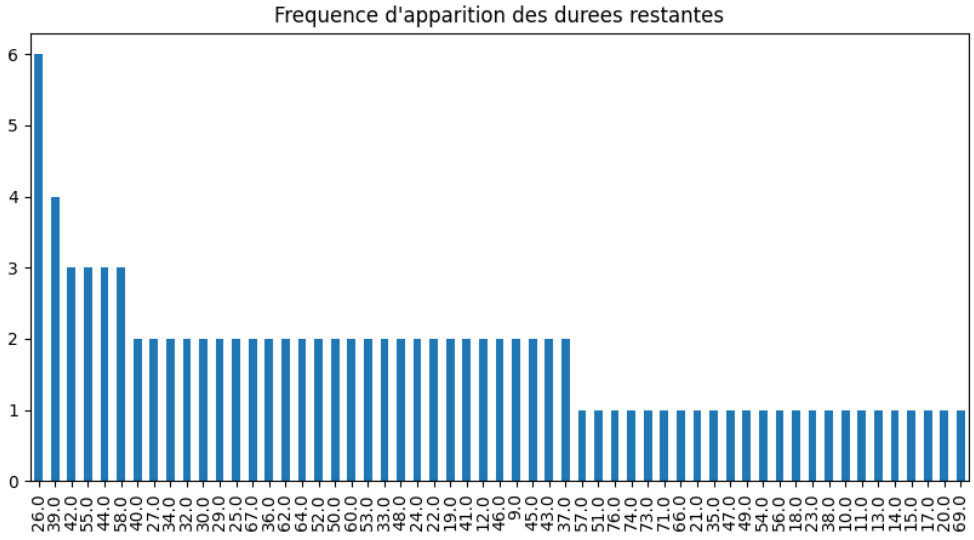
- Faire la moyenne des durées à chaque instant pour chaque taxi, et récupérer le maximum des moyennes (pire cas)
- Moyenne du temps restant cumulé à chaque instant (cas moyen)
- Fréquence d’apparition des durées restantes

On obtient trois graphes par ordonnancement :

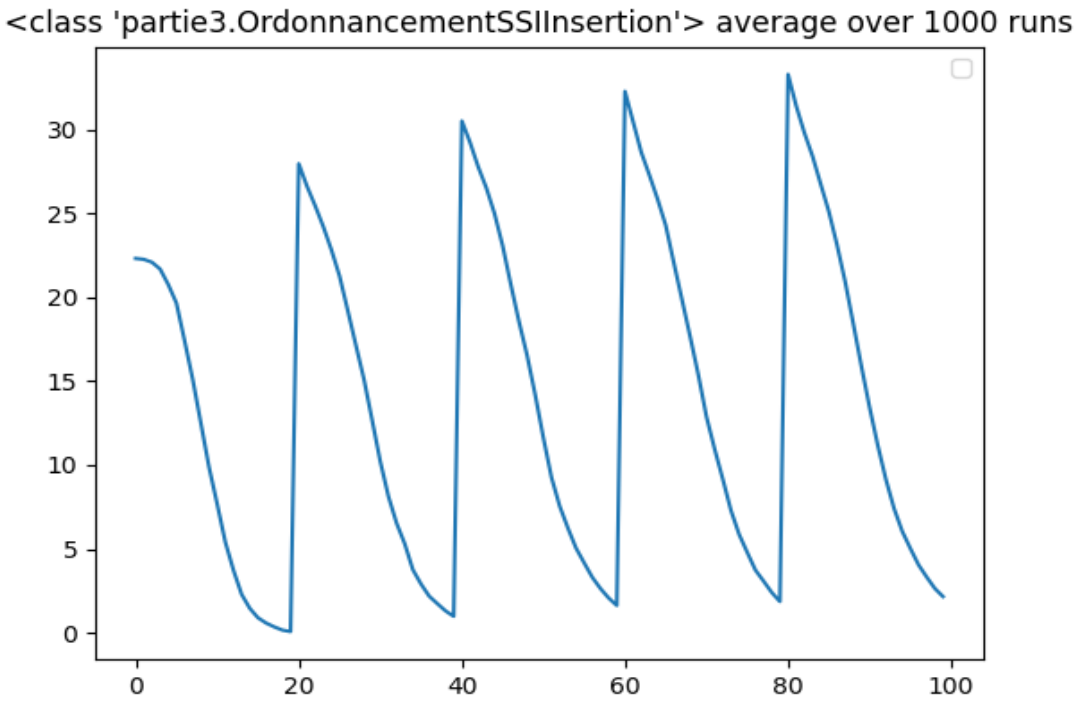
PSI

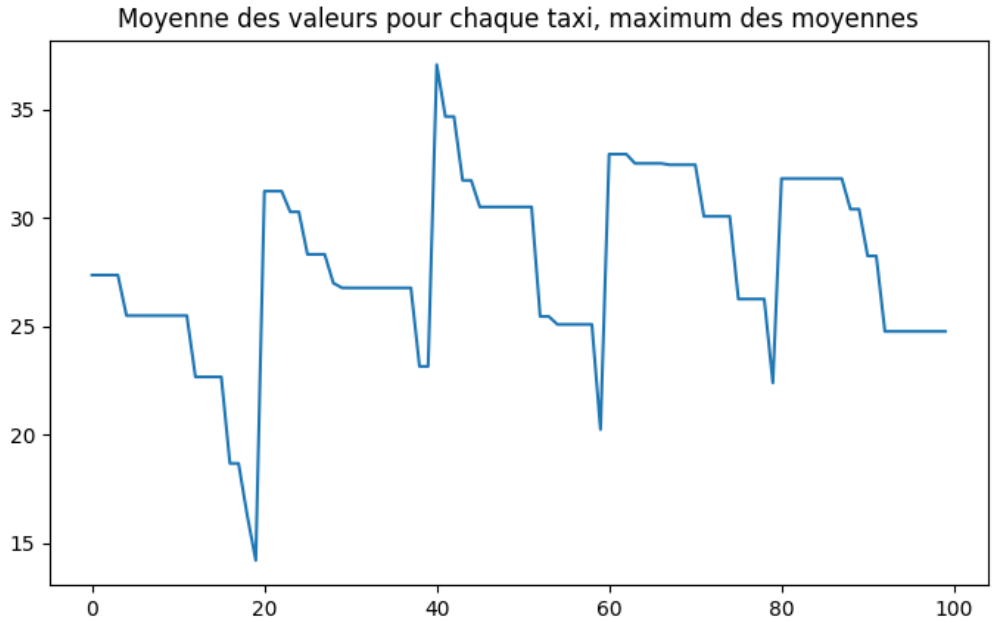

<class 'partie3.OrdonnancementPSI'> average over 1000 runs
70
60
50
40
30
20
10
0
20
40
60
80
100

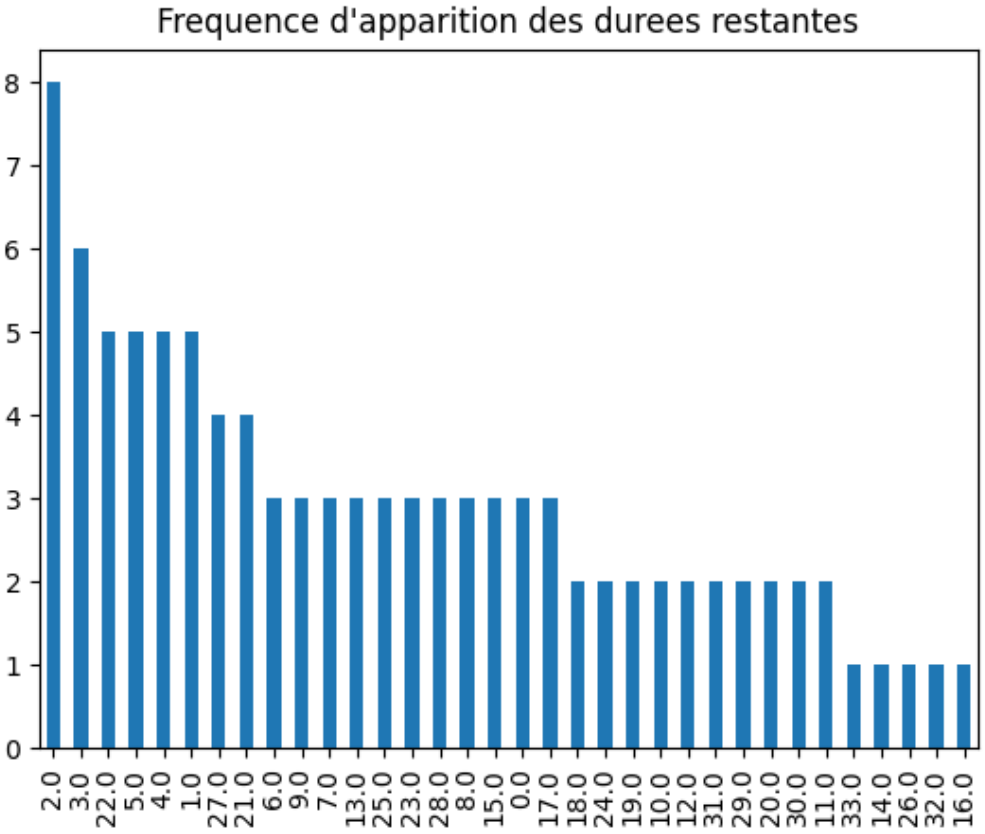

Moyenne des valeurs pour chaque taxi, maximum des moyennes
160
140
120
100
80
60
0
20
40
60
80
100


Frequence d'apparition des durees restantes
6
5
4
3
2
1
0
26.0
39.0
42.0
55.0
44.0
58.0
40.0
27.0
34.0
32.0
30.0
29.0
25.0
67.0
36.0
62.0
64.0
52.0
50.0
60.0
53.0
33.0
48.0
24.0
22.0
19.0
41.0
12.0
46.0
9.0
45.0
43.0
37.0
57.0
51.0
76.0
74.0
73.0
71.0
66.0
21.0
35.0
47.0
49.0
54.0
56.0
18.0
23.0
38.0
10.0
11.0
13.0
14.0
15.0
17.0
20.0
69.0

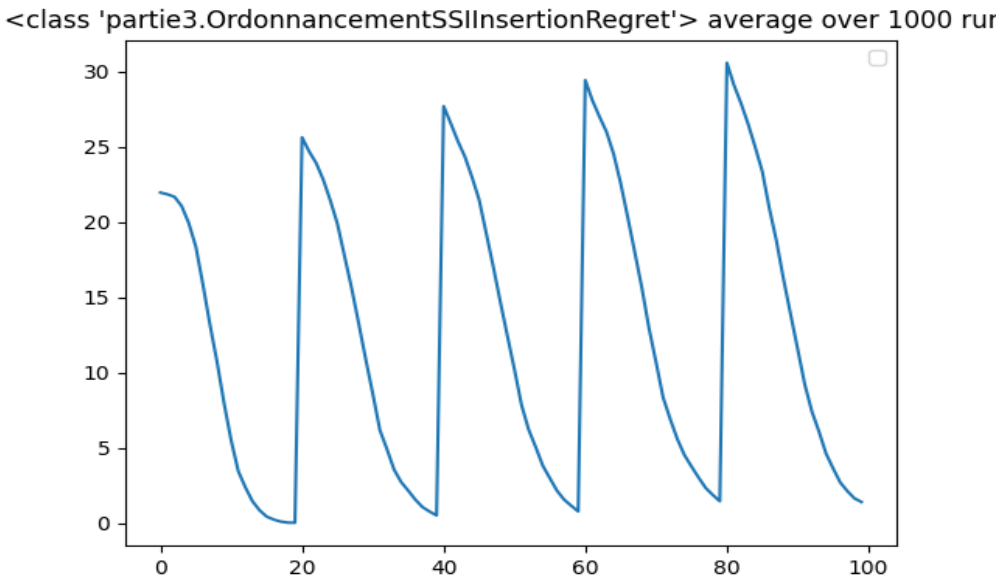
SSI

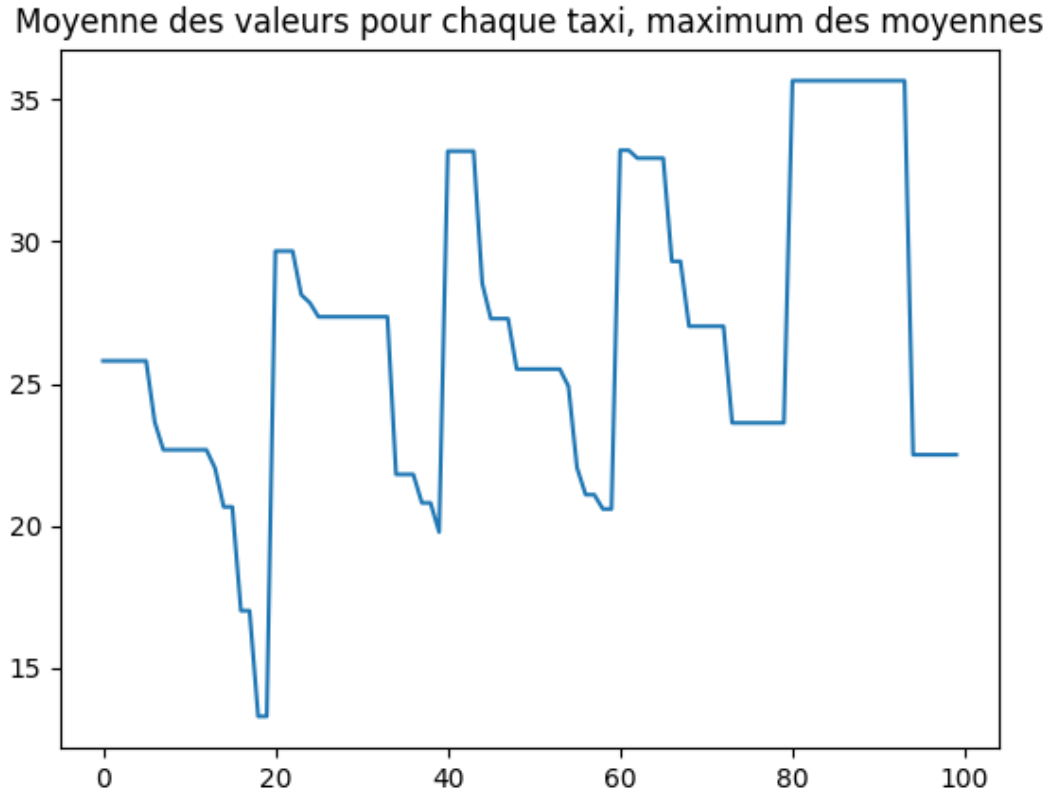

<class 'partie3.OrdonnancementSSIInsertion'> average over 1000 runs
0
5
10
15
20
25
30
0
20
40
60
80
100

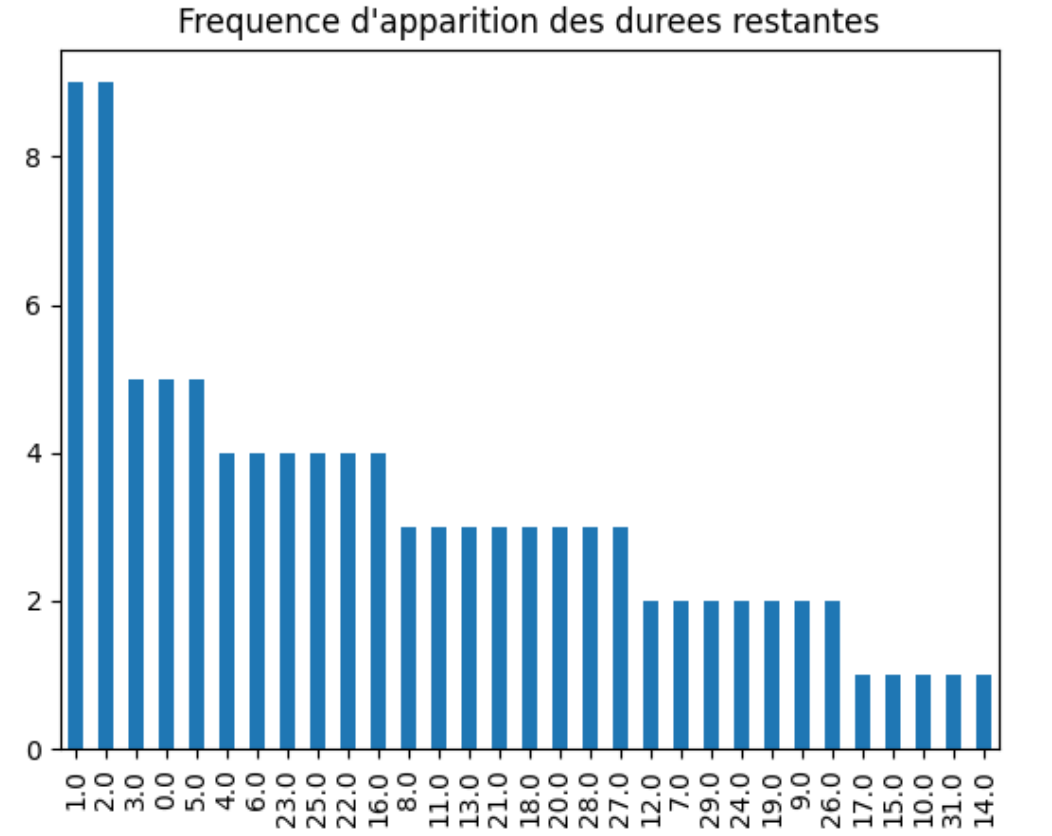

Moyenne des valeurs pour chaque taxi, maximum des moyennes
15
20
25
30
35
0
20
40
60
80
100


Frequence d'apparition des durees restantes
8
7
6
5
4
3
2
1
0
2.0
3.0
22.0
5.0
4.0
1.0
27.0
21.0
6.0
9.0
7.0
13.0
25.0
23.0
28.0
8.0
15.0
0.0
17.0
18.0
24.0
19.0
10.0
12.0
31.0
29.0
20.0
30.0
11.0
33.0
14.0
26.0
32.0
16.0

SSI Regrets


<class 'partie3.OrdonnancementSSIInsertionRegret'> average over 1000 rur
30
25
20
15
10
5
0
0
20
40
60
80
100


Moyenne des valeurs pour chaque taxi, maximum des moyennes
35
30
25
20
15
0
20
40
60
80
100


Frequence d'apparition des durees restantes
8
6
4
2
0
1.0
2.0
3.0
0.0
5.0
4.0
6.0
23.0
25.0
22.0
16.0
8.0
11.0
13.0
21.0
18.0
20.0
28.0
27.0
12.0
7.0
29.0
24.0
19.0
9.0
26.0
17.0
15.0
10.0
31.0
14.0

On peut observer assez clairement que l'ordonnancement PSI est moins performant que les méthodes SSI sur les trois métriques : les temps cumulés restants sont en moyenne plus grands, le maximum des moyennes des durées augmente à la différence des SSI, et les temps les plus fréquents sont élevés.

La version de SSI avec ou sans regrets on des résultats assez similaires. Une différence notable, qui tend à rendre la prise en compte des regrets meilleure est le graphe des fréquences : pour la version avec regrets, les durées les plus fréquentes sont basses (<= 6), alors que sans regrets les durées 22 et 27 apparaissent. Outre cela, la différence semble assez faible avec le jeu de paramètres testé, il faudrait ajouter des expériences.

Conclusion

On peut donc en conclure que les méthodes séquentielles sont meilleures que les enchères parallèles. Il semble que la prise en compte des regrets soit avantageuse, mais plus de tests sont nécessaires pour confirmer cette hypothèse.