Ferreira Maadini Badaoui

Rapport projet problème VRP : Modèle heuristique

1. Méthodes heuristiques de construction

Explication

Il y a eu trois implémentations de construction.
La première constructionNextFit consiste à placer les clients dans l'ordre de la liste a la fin du cycle, donc soit dans le véhicule en cours soit si la demande du client ne peut être respecté par ledit véhicule ajouter un nouveau véhicule a la route et y ajouter le client.

La deuxième constructionBestDistance consiste à placer à la fin du cycle le client qui possède la plus petite distance avec le dernier client du cycle. Si le client ne peut aller dans ce véhicule à cause de la demande trop grande on ajoute un nouveau véhicule et on ajoute à ce nouveau véhicule le client qui possède la distance avec le fournisseur la plus petite.

La troisième constructionAllVehicle consiste à mettre chaque client dans un véhicule différent la route sera ainsi constituer du même nombre de véhicules que de client.

Pseudo-code

```
constructionNextFit (data, sol)
    Pour chaque client :
        Si la demande du client est réalisable par le véhicule actuel :
            on ajoute le client au véhicule actuel
        sinon :
            on ajoute le client à un nouveau véhicule
        Fin Si
    Fin pour
Fin fonction

constructionBestDistance (data, sol)
    Pour i de 1 jusqu'au nombre de clients avec un pas de 1 :
        on retrouve le client le plus proche du dernier client parcouru
        Si la demande du client est réalisable par le véhicule actuel :
            on ajoute le client au véhicule actuel
        sinon :
            on ajoute le client le plus proche du fournisseur dans un nouveau véhicule
        Fin Si
    Fin pour
Fin fonction

constructionAllVehicle (data, sol)
    Pour chaque client :
        ajouter le client a un nouveau véhicule
    Fin pour
Fin fonction
```

2. Méthodes heuristiques d'amélioration

```
echange(data, sol, firstvéhicule, firstclient, limitValue)
    pour chaque vehicule a partir de firstvéhicule :
        pour chaque client qui se situe après firstclient dans la route :
            si le changement de demande lors de l'echange des deux clients est possible :
                si la distance value < limitValue:
                    effectuer l'échange entre les deux clients
                Fin si
            Fin si
        Fin pour
    Fin pour
Fin fonction
```

Échange prend en paramètre un client et recherche parmi tous les clients qui le suivent une possibilité d'échanger les positions entre les deux clients. Si l'échange est possible alors on calcule la distance value si la value est inférieur à limitValue alors on effectue l'échange.
La distance value correspond au changement de distance avec l'échange.
limitValue a une valeur qui varie avec le temps entre 0 et limitValueMax.

```
deplace(data, sol, firstvéhicule, firstclient ,limitValue )
    pour chaque vehicule :
        pour chaque client :
            si le déplacement du premier client est possible avec la demande :
                si la distance value < limitValue :
                    effectuer le déplacement du premier client a la place du deuxième
                    si plus de client dans ce segment de route :
                        supprimer le segment
                    Fin si
                Fin si
            Fin si
        Fin pour
    Fin pour
    si distance value avec un déplacement sur un nouveau segment de route :
        déplacer le client sur un nouveau segment de route
    Fin si
Fin fonction
```

Déplace prend en paramètre un client et recherche parmi tous les clients une possibilité pour déplacer le premier client a la position du deuxième et décalé les clients à partir du deuxième client. Si l'échange est possible alors on calcule la distance value si la value est inférieur à limitValue alors on effectue le déplacement.
La distance value correspond au changement de distance avec l'échange.
limitValue a une valeur qui varie avec le temps entre 0 et limitValueMax.

```
oneImprove (data, sol, limitValue)
    Pour chaque vehicule :
        Pour chaque client dans le véhicule actuel :
            echange(data, sol, véhicule, client, limitValue)
            deplace(data, sol, véhicule, client, limitValue)
        Fin pour
    Fin pour
Fin fonction
```

oneImprove exécute pour tous les clients échange et déplace.

```
makeSolMin(data, constructor, limitValueMax, secondMax, distanceFixed)
  Solution sol
  Solution solMin
  lance une contstruction dependant de constructor
  double cost = cost de sol
  int limitValue = 0
  tant que le temps d'execution ne depasse pas secondMax :
    oneImprove (data, sol, limitValue)
    si cost <= que cost de sol +0.01
      si limitValue > limitValueMax
        limitValue = 0
      sinon
        limitValue ++
      fin si
    fin si
    cost = cost de sol
    si cost < cost de solMin
      solMin = sol
      si non distanceFixed
        limitValueMax = cost de solMin / 50
      fin si
      limitValue = 0
    fin si
  fin tant que
  return solMin
fin fonction
```

makeSolMin lance une construction puis lance en boucle oneImprove en changeant la valeur de limitValue si la nouvelle solution n'est pas meilleur que la précédente. Cela permet de ne pas rester bloqué dans une solution qui ne peut pas être amélioré, car echange et move ne change rien, parce que les seules possibilités de changement ne sont pas une meilleure solution. En augmentant limitValue on accepte d'effectuer un changement qui donne une moins bonne solution pour avoir une meilleure solution plus tard.

3. Choix de développement

Il y a beaucoup de paramètre qui influence le résultat final. Les trois constructions donne des résultats différents ainsi que les valeurs de limitValueMax. Les résultats sont différents en fonction des instances. C'est alors impossible d'avoir tout le temps le minimum avec une même exécution pour différentes instances.

Je me suis questionné sur un autre problème, si je prends une valeur fixe pour limitValueMax et que les distance sont à une autre échelle, pour une instance par exemple des distances cent fois plus grande ma valeur aura surement que très peu d'intérêt. J'ai donc réfléchi a utilisé une valeur qui dépend de l'instance et je l'ai calculé avec le cost de solMin en me disant que si la route était plus grande il fallait une plus grande valeur et de même si elle est plus petite.

Avec une analyse plus poussée on peut surement obtenir une meilleure formule qui utiliserais la moyenne des distances ou l'écart de distance moyen entre 2 clients ou encore bien d'autre possibilité.

J'ai à cause de tous ces paramètres effectuer beaucoup de test pour analyser les résultats. J'ai mis tout dans le dossier save du dossier test, chaque instance possède son fichier CSV avec les résultats obtenu. J'ai également un dossier solution où sont écrite les solutions obtenues.

4. Complexité et performance

rapport taille	taille	count	temps	rapport temps
	50	2 695 536,00	0,000111295	
1,5	75	1 650 441,00	0,00018177	1,633221666
1,333333333	100	906 496,00	0,000330945	1,820682055
1,5	150	516 515,00	0,000580816	1,755023571
1,326666667	199	353 288,00	0,000849166	1,462022486
0,603015075	120	566 478,00	0,000529588	0,623657053
0,833333333	100	773 676,00	0,000387759	0,732190219

Pendant les exécutions j'ai compte le nombre de oneImprove qui sont lancer en le comparant avec la taille de chaque instance on peut estimer que la complexité est linéaire. Pour limiter la perte de temps, j'ai pris la décision de ne pas faire de copie de solution lors des tentatives d'échange ou move et de calculer les nouvelles demandes ainsi que le changement de cost sans faire de copie.
Cette décision à demander à plus grand temps de développement que de simplement faire de copie, mais c'était une volonté dès le début du projet.

5. Résultat

vrpnc1	Next Fit	Best Distance	All Vehicle
0	599.8	659.311	637.579
1	582.329	666.23	629.705
2	583.784	648.582	624.779
3	597.201	622.463	616.61
4	596.43	648.582	570.843
5	546.296	567.187	581.81
6	573.62	625.846	571.91
7	563.806	540.124	563.543
8	530.583	537.472	533.947
9	536.384	532.284	532.284
10	531.08	533.068	532.028
11	558.688	529.474	529.474
unfixed	558.69	532.404	532.404

vrpnc2	Next Fit	Best Distance	All Vehicle
0	920.139	976.343	961.242
1	901.668	971.75	957.812
2	906.338	932.376	957.812
3	902.721	919.807	913.092
4	905.652	941.62	889.862
5	891.638	884.268	885.633
6	868.521	879.27	868.385
7	865.117	856.987	863.417
8	860.841	859.517	863.517
9	858.513	861.362	867.114
10	862.565	863.181	863.674
11	866.4	848.27	866.013
unfixed	851.478	865.978	849.873

vrpnc3	Next Fit	Best Distance	All Vehicle
0	964.438	1044.95	994.253
1	964.438	1054.88	987.649
2	909.515	918.296	928.601
3	899.214	889.21	870.857
4	851.868	849.937	849.128
5	855.323	850.01	853.431
6	854.528	855.983	854.917
7	845.58	861.388	856.605
8	853.138	847.675	851.934
9	855.72	853.591	859.342
10	858.656	853.961	859.11
11	856.299	860.84	864.416
unfixed	854.12	854.599	862.636

vrpnc4	Next Fit	Best Distance	All Vehicle
0	1279.44	1359.65	1372.48
1	1274.26	1282.15	1301.59
2	1183.97	1160.93	1259.38
3	1104.95	1125.38	1111.93
4	1104.58	1106.84	1092.44
5	1120.61	1107.5	1114.09
6	1102.58	1113.09	1114.96
7	1116.47	1107.15	1106.89
8	1109.78	1110.76	1123.81
9	1106.53	1119.21	1128.2
10	1120.02	1127.72	1133.24
11	1118.04	1114.08	1107.16
unfixed	1119.08	1115.2	1130.9

vrpnc11	Next Fit	Best Distance	All Vehicle
0	1098.02	1555.14	1483.04
1	1091.94	1467.52	1461.07
2	1061.75	1351.5	1151.68
3	1066.41	1062.97	1065.96
4	1062.3	1055.47	1058.52
5	1055.42	1054.94	1058.64
6	1056.89	1053.72	1064.06
7	1058.42	1054.87	1052.21
8	1050.92	1052.13	1048.86
9	1053.55	1055.18	1045.66
10	1059.38	1055.9	1048.96
11	1053.9	1056.45	1048.12
unfixed	1080.03	1093.55	1072.25

vrpnc12	Next Fit	Best Distance	All Vehicle
0	1002.35	1244.01	979.635
1	990.157	1092.76	853.858
2	979.678	1008.54	855.498
3	851.013	901.138	835.494
4	831.793	834.671	860.033
5	830.288	823.726	826.972
6	824.446	822.267	821.579
7	819.558	821.421	820.325
8	821.579	821.451	821.579
9	821.291	822.745	822.985
10	825.347	822.297	824.429
11	832.012	828.027	824.463
unfixed	844.719	830.71	826.892

Les valeurs minimum pour chaque méthode de construction a était affiché en jaune. On peut voir que la volonté d'avoir limitValueMax qui dépend de l'instance n'est pas très concluante, la valeur est minimum uniquement pour l'instance 2 et pour les autres instances il est assez mal placer. Cependant, on remarque qu'une valeur fixer vers 7 ou 8 obtient des résultats minimum sur beaucoup d'instances.

En ce qui concerne la méthode de construction les trois sont à peu près équivalente pour chaque ligne sauf pour des petites valeur fixée.