Qi QIU

Rapport du projet MADMC

M2 Informatique - Spécialité ANDROIDE

Sorbonne Université

Année universitaire 2019-2020

1 Organisation du code

1. Class Solveur
   (a) plne_MTZ():
      i. Résolution de l'instance par la PLNE avec la formulation MTZ
   (b) Branch-and-Cut():
      i. Résolution de l'instance par la PLNE avec la méthode Branch and Cut
   (c) bin_plne_heuristique():
      i. Résolution de l'instance par la PLNE pour obtenir une solution exacte de Bin Packing
      ii. Produire une solution avec une méhode gloutonne à partir de la solution de la pahse précédente
   (d) bin_glouton_heuristique():
      i. Résolution de l'instance par l'algorithme First fit decreasing pour obtenir une solution de Bin Packing
      ii. Produire une solution avec une méhode gloutonne à partir de la solution de la pahse précédente
   (e) setInstance():
      i. Lier une instance avec le solveur
   (f) Instance*
   (g) Solution*
2. class Solution
   (a) write_SVG_tour()
      i. Generer une fichier "CVRP_solution.svg" qui permet de visualiser la solution via un avigateur
   (b) visualisation()
      i. Generer une fichier "cvrp_solution.dot" et un fichier "cvrp_solution.png" qui permet de visualiser la solution
   (c) affichage():
      i. afficher la solution dans std::cout
3. class Instance
   (a) read-File()
   (b) Clients[ ], distance[ ] [ ]
   (c) Q, n, m
4. Client
   (a) x, y, i, demande

2 Méthode Heuristique

2.1 Bin Packing PLNE + Glouton

1. Générer une solution exacte de Bin Packing avec la methode First Fit Decreasing si possible, sinon avec la PLNE
2. Améliorer la solution avec une méthode gloutonne

2.2 Experiences

L'ensemble des instances: Vrp-Set-P

instance	temps d'éxecution	borne superieure trouvée	valeur optimale	gap
$n = 16, k = 8$	0.00s	526	450	16.9%
$n = 19, k = 2$	0.00s	300	212	41.5%
$n = 22, k = 8$	0.00s	921	603	52.7%
$n = 40, k = 5$	0.00s	927	458	102.4%

Table 1: BinPacking plne + glouton

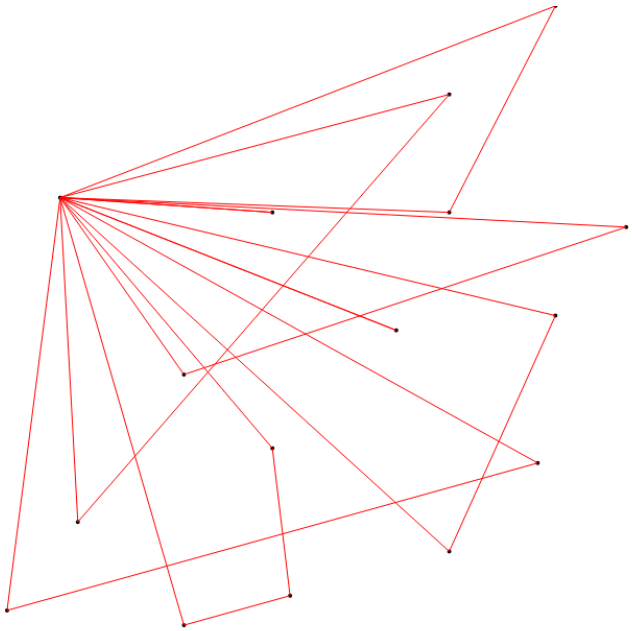

Figure 1: $P, n = 16, k = 8$

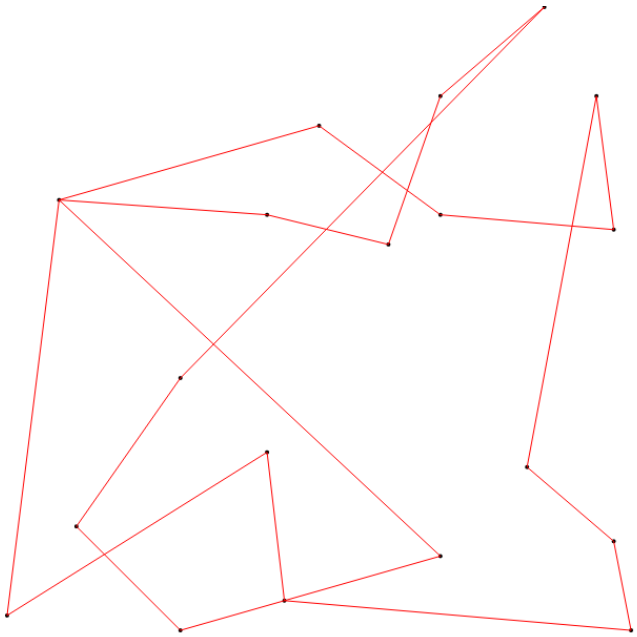

Figure 2: P,$n = 19$,$k = 2$.

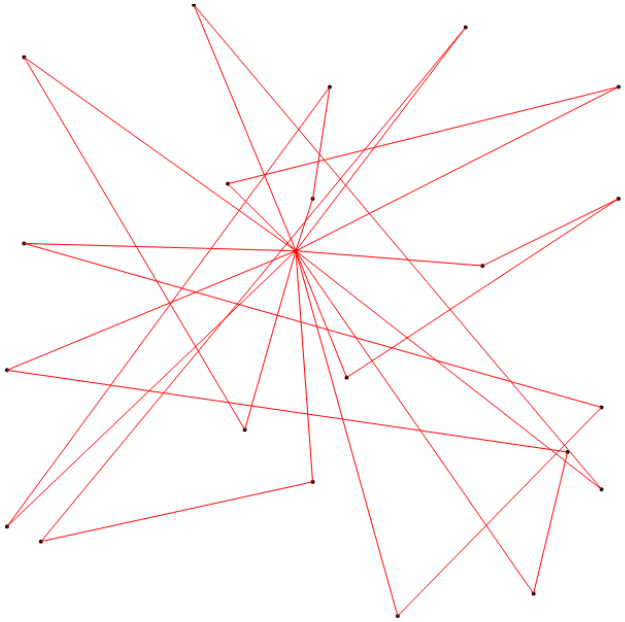

Figure 3: P,$n = 22$,$k = 8$.

3 MTZ

L'ensemble des instances: Vrp-Set-P

instance	temps d'éxecution	cout obtenu	cout indiqué dans le ficher
$n = 16,k = 8$	2.26	451.335	450
$n = 19,k = 2$	18.07	212.657	212
$n = 22,k = 8$	441.36	600.826	603

Table 2: MTZ

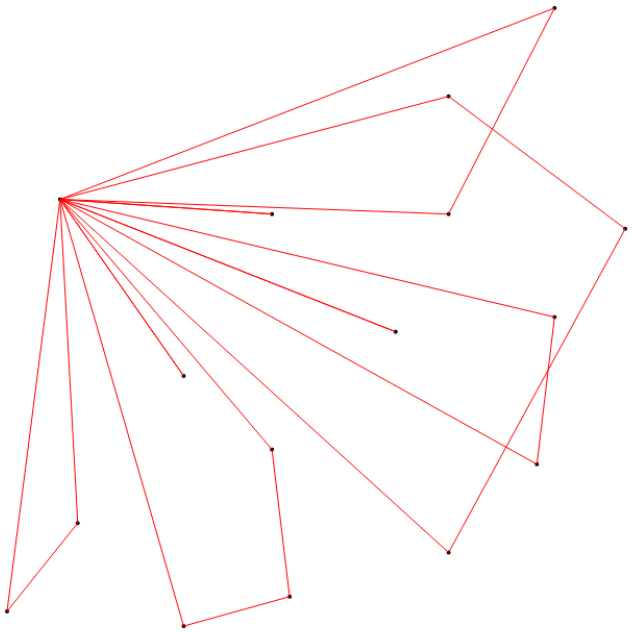

Figure 4: $P,n = 16,k = 8$

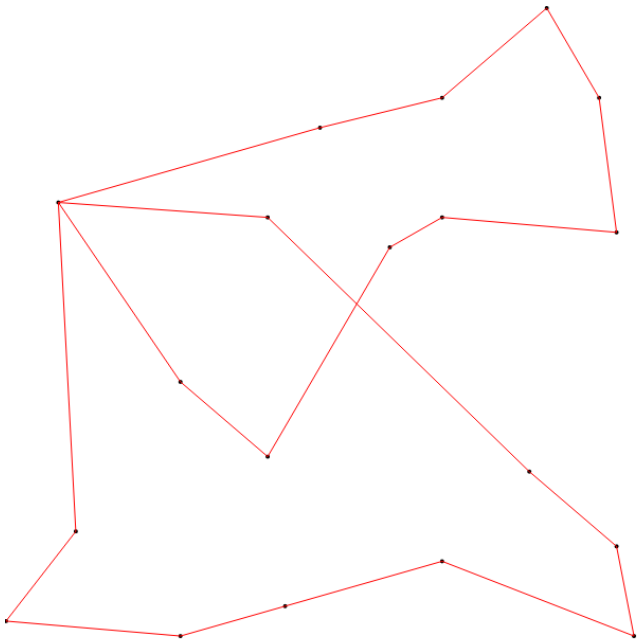

Figure 5: P,$n = 19$,$k = 2$.

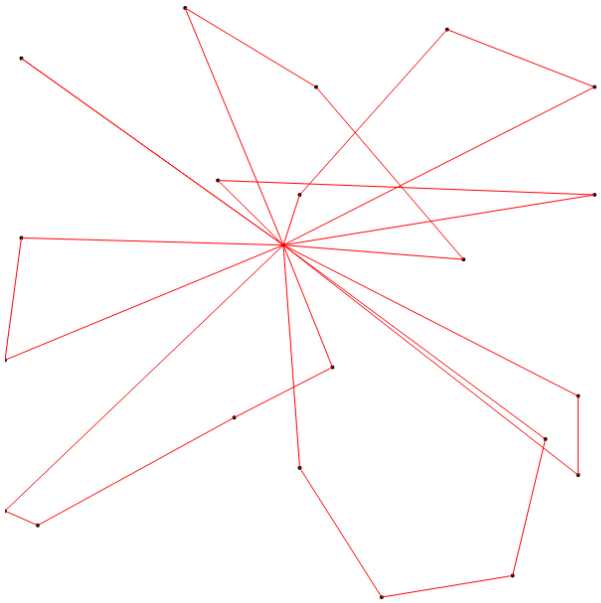

Figure 6: P,$n = 22$,$k = 8$.

4 Branch and Cut

4.1 Inégalités pour Couper des solutions entières

Pour une solution entière trouvée au cours de branchement, pour chaque tour S, on considère trois cas:

1. S'il est un sous tour, on ajoute la contrainte

$$\sum_{i \in S, j \in V \setminus S} X[i][j] >= ceil((\sum_{i \in S}) d[i])/Q)$$

2. S'il est un tour S passant par 0 avec $\sum_{i \in S} d[i] > Q$, on ajoute la contrainte

$$\sum_{i \in S \setminus \{0\}} X[i][j] * d[i] = ceil((\sum_{i \in S}) d[i])/Q)Q$$

3. sinon, c'est une solution admissible.

4.2 Experiences

Benchmark: Vrp-Set-P

instance	temps d'éxecution	borne inférieur	cout indiqué dans le ficher	nombre de coupes
$n = 16, k = 8$	1.77s	451.33	450	287
$n = 19, k = 2$	0.41s	212.65	212	97
$n = 22, k = 8$	3082s	600.826	603	3221

Table 3: Lazy cut

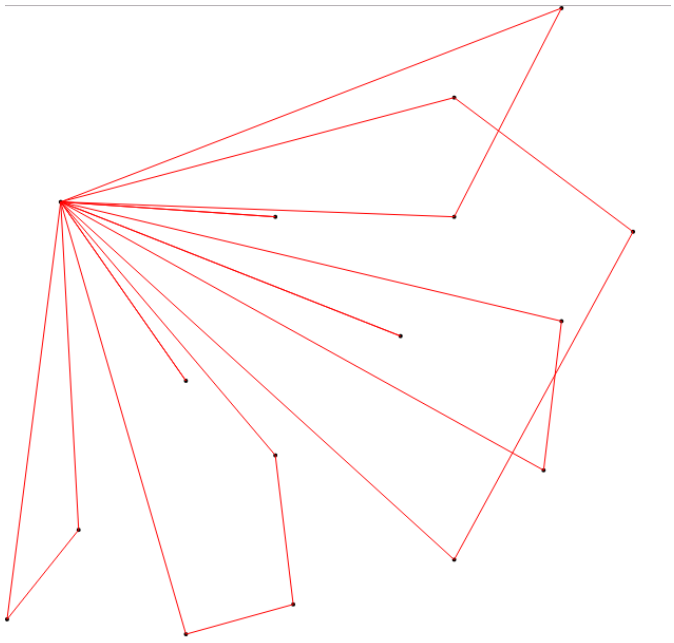

Figure 7: $P, n = 22, k = 8$

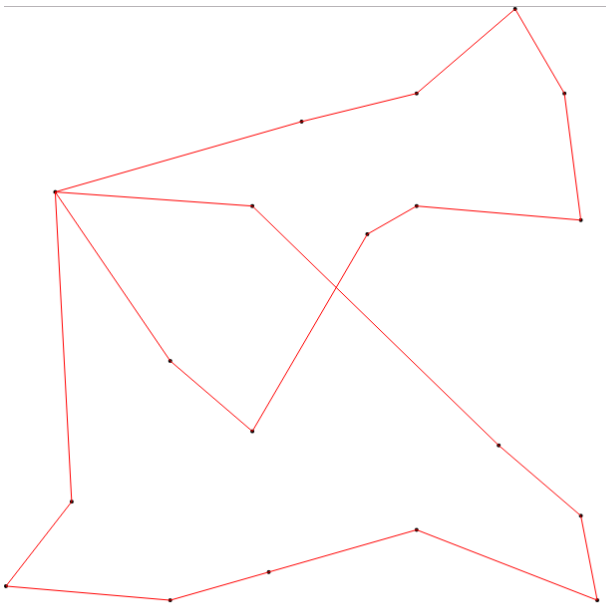

Figure 8: P,$n = 22$,$k = 8$

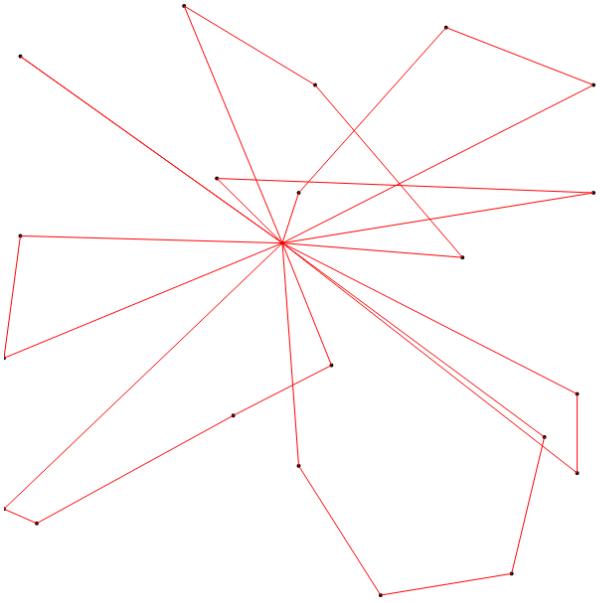

Figure 9: P,$n = 22$,$k = 8$.

4.3 Inégalités pour Couper de solutions fractionaires

Méthode utilisée: Greedy randomized algorithm

1. Posons S={i} où i est un élément arbitraire de V
2. À chaque étape, on cherche dans V un élément v qui maximise

$$\sum_{s \in S} solx[v][s]$$

3. Ajouter v dans S et vérifier l'inégalité $\sum_{i \in S, j \in V \setminus S} X[i][j] >= ceil((\sum_{i \in S}) d[i])/Q)$

4.4 Experiences

instance	temps d'éxecution	borne inférieur trouvée	valeur optimale	user cuts applied
$n = 16, k = 8$	3.81	451.335	450	335
$n = 19, k = 2$	0.59	212.657	212	97
$n = 22, k = 8$	4194	600.826	603	3221

Table 4: coupe de sous tours

4.5 Remarques

1. Les implémentations des coupes ne sont pas efficaces, parce que le graph est interprété par une matrice d'incidence.
2. Les coupes des solutions entières sont correctes.
3. Les coupes des solutions fractionnaires ne renforcent pas le programme. Par contre, elles fournissent de bonnes bornes inférieures:

instance	temps d'éxecution	borne inférieur trouvée	valeur optimale	gap
$n = 16, k = 8$	0.41	417	450	7.3%
$n = 19, k = 2$	0.34	196	212	7.5%
$n = 22, k = 8$	59	550	603	8.8%

Table 5: coupes des solutions fractionnaires

5 Comparaison

1. La méthode heuristique trouve une solution très rapide, mais la solution n'est pas optimale.
2. La méthode MTZ trouve la solution optimale, mais le temps augmente vite.
3. La méthode Branch and Cut qui utilise la liste d'incidence même avec seulement les coupes des solutions entières seraient beaucoup plus efficaces que MTZ.