Création d'une *Learning Heuristic* pour CVRP

Clément Legrand [1]

July 3, 2018

[1]Stage réalisé dans l'équipe ORKAD du centre CRISTAL à Lille, sous la direction de Laetitia Jourdan (laetitia.jourdan@univ-lille1.fr), Marie-Éléonore Kessaci (me.kessaci@univ-lille1.fr) et Diego Cattaruzza (diego.cattaruzza@ec-lille.fr)

Capacitated Vehicle Routing Problem

Instance

- $n-1$ clients (et leur demande d_i)
- k véhicules disponibles, de capacité C

Objectif

Déterminer *Sol* (ensemble des tournées) tel que:

$$Sol = argmin_{Sol} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{k} distance(i,j) x_{i,j}^{v} = argmin_{Sol} cost(Sol)$$

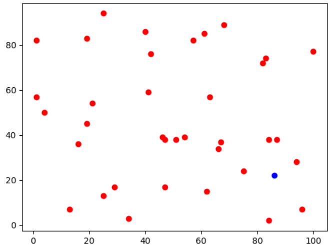
où $x_{i,j}^{v} = 1$ si j est desservi après i par le véhicule v (et 0 sinon).

Contraintes

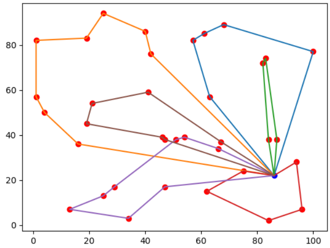
- Chaque client doit être desservie par un unique véhicule;
- Chaque tournée doit partir et s'arrêter au dépôt;
- La somme des demandes sur une tournée ne peut excéder la capacité du véhicule.

Illustration

Instance A-n37-k06 (donc 36 clients, et 6 véhicules disponibles de capacité 100):



Représentation instance



Meilleure solution connue, *cost* $= 950$

Objectif

Intégrer de la connaissance pour trouver une meilleure solution

Méthode

Réussir à prédire des arêtes qui appartiendront à la solution optimale, et les exploiter pour construire une nouvelle solution.

- Comparer à des solutions optimales pour des petites instances;
- Établir des règles qui caractérisent ces arêtes;
- Exploiter ces arêtes dans un algorithme d'optimisation.

Problèmes

- Comment construire une solution initiale de bonne qualité ?
- Quel algorithme d'optimisation utiliser ?
- Comment extraire la connaissance ?
- Comment intégrer la connaissance dans un algorithme d'optimisation ?

Algorithme Clarke & Wright (CW)

CW[2] → Algorithme glouton (chaque client est initialement desservi par un véhicule (contrainte de véhicules non respectée), puis la fusion des tournées est basée sur un calcul de saving).

Définition saving

Calcul du saving de i et j avec:

$$s(i,j) = c_{i0} + c_{0j} - \lambda c_{ij} + \mu |c_{i0} - c_{0j}| + \nu \frac{d_i + d_j}{\bar{d}}$$

(λ, μ, ν) sont des paramètres à déterminer

Fonctionnement

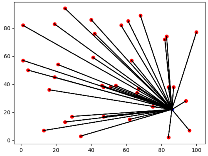
Tant que $max_{(i,j)} s(i,j) > 0$:

- $(i,j) \leftarrow argmax_{(i,j)} s(i,j)$;
- Les tournées qui contiennent i et j sont fusionnées (si possible);
- $s(i,j) \leftarrow 0$.

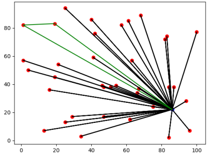
[2] IK. Altinel and T. Öncan, A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem (2005)

Exécution pour $(\lambda, \mu, \nu) = (1, 1, 1)$ sur A-n37-k06

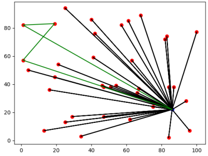
Initialisation



1^{ere} fusion

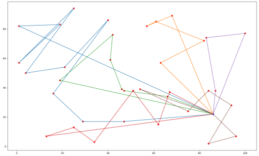


2^{eme} fusion



$\Downarrow$

Solution (31 fusions, *cost* $= 1297$)



Observation

Pour améliorer la solution obtenue, on pourrait réorganiser chaque tournée, pour diminuer leur coût.

Choix de (λ, μ, ν) ?

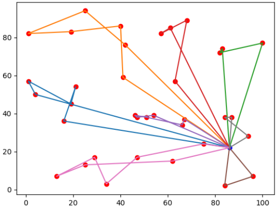
Dans la littérature plusieurs résultats sont disponibles :

- On peut se restreindre à l'intervalle $[0, 2]$ pour choisir $\lambda(\neq 0)$, μ etν.
- Il est inutile de regarder ce qui se passe au centième.

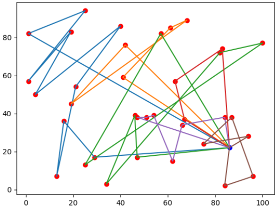
Bilan

Ainsi on se contentera pour la suite de prendre des valeurs de λ, μ etν arrondies au dixième, et comprises entre $[0, 2]$ (et $\lambda \neq 0$). Donc 8820 triplets possibles.

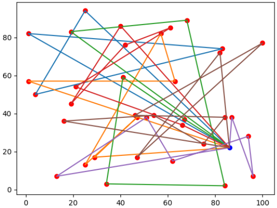
Choix de (λ, μ, ν) ?



$(1.9, 0.1, 1.5)$, *cost* $= 1106$



$(0.1, 0.1, 0.1)$, *cost* $= 1569$



$(0.0, 1.0, 1.5)$, *cost* $= 2191$

Bilan

Difficile de prévoir l'influence des paramètres (λ, μ, ν) (pas d'évolution linéaire...).
C'est aussi le cas pour toute instance : l'influence de (λ, μ, ν) dépend de l'instance.

Heuristique Arnold & Sörensen

Heuristique A & S [3]

$Sol \leftarrow CW(\lambda, \mu, \nu)$
$NewSol \leftarrow Sol$
while *Pas d'amélioration depuis 3 min* **do**
    Calcul de la pire arête
    $NewSol \leftarrow EjectionChain_{BI-O}$
    $NewSol \leftarrow LinKernighan_{BI-O}$
    $NewSol \leftarrow CrossExchange_{BI-O}$
    $NewSol \leftarrow LinKernighan_{BI-O}$
    if $cost(NewSol) < cost(Sol)$ **then**
        $Sol \leftarrow NewSol$
return *Sol*

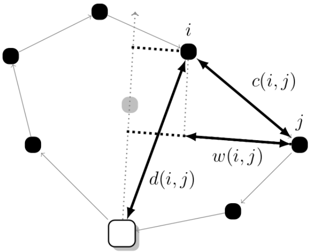
[3] Florian Arnold and Kenneth Sörensen, A simple, deterministic and efficient knowledge-driven heuristic for the vehicle routing problem (2017)

Pire arête

Pire arête

La pire arête du graphe est l'arête (i, j) qui maximise la fonction:

$$b(i,j) = \frac{[\gamma_w w(i,j) + \gamma_c c(i,j)][\frac{d(i,j)}{max_{k,l} d(k,l)}]^{\frac{\gamma_d}{2}}}{1 + p(i,j)}$$



Opérateurs de voisinage

Ejection-chain

Déplacer l clients sur des tournées. On fixe $l = 3$ d'après la littérature.

Cross-exchange

Échanger deux séquences de clients entre deux tournées.

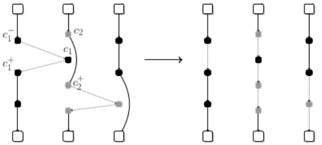


Figure 2: Illustration of the ejection chain with two relocations.

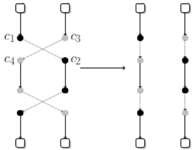
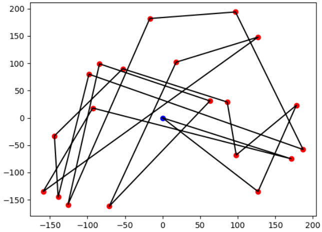


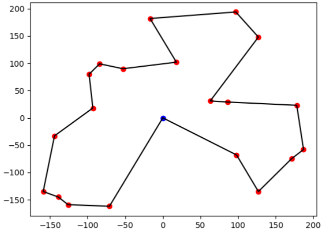
Figure 1: Illustration of the CROSS-exchange with sequences of two customers.

Opérateurs de voisinage

Lin-Kernighan

- Créé pour TSP;
- Optimisation intra-tournée (chaque tournée est améliorée indépendamment des autres).





Algorithme d'optimisation (H_c)

$Sol \leftarrow CW(\lambda, \mu, \nu)$
$NewSol \leftarrow Sol$
while *La dernière amélioration date de moins de* $n/3$ *min* **do**
    Calcul de la pire arête
    $NewSol \leftarrow EjectionChain_{FI-RD}$
    $NewSol \leftarrow LinKernighan_{BI-O}$
    $NewSol \leftarrow CrossExchange_{FI-RD}$
    $NewSol \leftarrow LinKernighan_{BI-O}$
    if $cost(NewSol) < cost(Sol)$ **then**
        $Sol \leftarrow NewSol$
    if *Pas d'amélioration depuis* $n/2$ *itérations* **then**
        $NewSol \leftarrow Sol$ ▷ Restart
return Sol

Validation du *Restart*

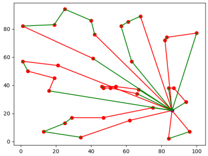
	A-n37-k06			A-n65-k09			P-n101-k04		
Restart	Best	$Mean_{10}$	Time	Best	$Mean_{10}$	Time	Best	$Mean_{10}$	Time
Sans	950	957	195	1197	1215	395	722	736	783
Avec	950	969	200	1200	1230	350	698	706	1500

Bilan

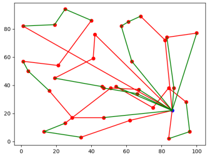
Diversification plus intéressante pour des grandes instances

Comment extraire de la connaissance ?

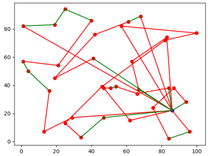
$CW(1.9, 0.1, 1.5) + LK$
$cost = 1041$, 19 arêtes opt



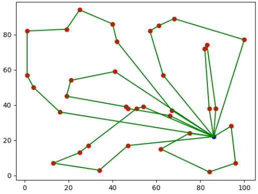
$CW(0.1, 0.1, 0.1) + LK$
$cost = 1170$, 19 arêtes opt



$CW(0.0, 1.0, 1.5) + LK$
$cost = 1600$, 11 arêtes opt



$\downarrow$



$cost = 950$, 42 arêtes

Observations

Les bonnes solutions semblent contenir davantage d'arêtes optimales. Le problème est donc de trouver les bons (λ, μ, ν)...

Protocole

Questions

- Combien de solutions dans l'échantillon ?
- Combien de solutions pour apprendre ?
- Comment choisir les arêtes à conserver ?

Protocole

Combien de solutions dans l'échantillon ?

- Considérer tous les (λ, μ, ν);
- Tirer N (λ, μ, ν) aléatoirement. $N \in [50, 100, 500]$;

Quelles solutions pour apprendre ?

- Tout l'échantillon (Tout);
- $x\%$ des meilleures solutions : quantité privilégiée (Quan$_x$);
- Solutions avec coût inférieur à $c_{min} + (c_{max} - c_{min})\frac{x}{100}$: qualité privilégiée (Qual$_x$);

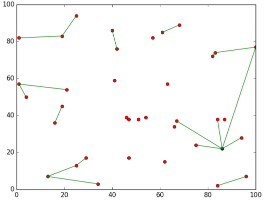
On choisit $x = 10$

Comment choisir les arêtes à conserver ?

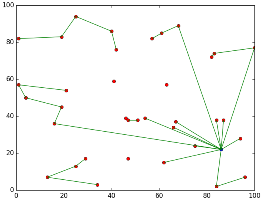
On initialise une matrice à 0. Pour chaque arête (i,j), on incrémente la valeur [i][j] de la matrice.

```
18, 21, 4, 9, 4, 0, 0, 19, 7, 9, 21, 0, 2, 10, 7
 1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 1, 2,
 0, 0, 5, 0, 0, 0, 0, 0, 0, 24, 0, 0, 0, 0, 3, 0
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
10, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 3,
0, 0, 0, 0, 0, 7, 4, 0, 0, 0, 0, 0, 0, 3, 0, 0,
0, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 0, 4, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 21, 0,
 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 10, 0, 0, 0, 0
 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 4, 0, 0, 0, 0,
0, 0, 0, 26, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 0, 0, 0, 1,
0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0, 4,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 16, 0, 22, 0, 0,
0, 0, 0, 0, 0, 6, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0,
6, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9,
```

Rang



Seuil



Comment choisir les arêtes à conserver ?

Deux possibilités pour choisir les arêtes à conserver :

Critères de choix

- Conserver $(i, j) \Leftrightarrow MAT[i][j] > seuil$ (Seuil). $seuil \in [S_{lb}/2, 3S_{lb}/4]$, S_{lb} (Size learning base) est la taille de la base d'apprentissage;
- Conserver les rg premières arêtes dans la matrice (Rang). $rg \in [10, 20, n/2]$.

Résultats A-n37-k06, critère Seuil

	Quan$_{10}$				Qual$_{10}$				Tout			
	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop
50	3	34	21	0.5	$S_{lb}/2$	33	21	0.50	25	23	15	0.35
	4	23	14	0.33	$3S_{lb}/4$	17	12	0.28	38	10	7	0.16
100	5	30	21	0.5	$S_{lb}/2$	31	23	0.55	50	24	17	0.40
	8	16	15	0.36	$3S_{lb}/4$	17	14	0.33	75	6	6	0.14
500	25	32	24	0.57	$S_{lb}/2$	31	22	0.52	250	22	15	0.36
	38	15	14	0.33	$3S_{lb}/4$	20	16	0.38	375	7	7	0.18
8000	400	33	24	0.57	$S_{lb}/2$	30	23	0.55	4000	25	16	0.38
	600	15	14	0.33	$3S_{lb}/4$	18	16	0.38	6000	9	6	0.14

- Taille de l'échantillon ne semble pas avoir d'influence sur les résultats (*prop* reste semblable quel que soit la taille de l'échantillon).
- Avec base *Tout* : valeurs de *prop* plus basses → pas la peine d'utiliser tout l'échantillon.

Remarque : Base Quan$_{10}$ trop petite avec échantillon 50 ou 100.

Résultats A-n37-k06, critère Rang

	$Quan_{10}$			$Qual_{10}$			Tout		
	Rang	Corr	Prop	Rang	Corr	Prop	Rang	Corr	Prop
50	10	6	0.14	10	6	0.14	10	7	0.16
	20	13	0.31	20	13	0.32	20	13	0.31
	18	12	0.28	18	13	0.3	18	12	0.28
100	10	9	0.21	10	9	0.21	10	10	0.24
	20	16	0.38	20	16	0.38	20	15	0.36
	18	13	0.3	18	13	0.3	18	12	0.29
500	10	9	0.21	10	10	0.24	10	9	0.21
	20	16	0.38	20	16	0.38	20	15	0.36
	18	13	0.3	18	13	0.3	18	12	0.28
8000	10	8	0.19	10	9	0.21	10	7	0.17
	20	14	0.33	20	14	0.33	20	14	0.33
	18	12	0.29	18	12	0.29	18	12	0.29

Les 3 bases fournissent des valeurs *prop* similaires.

Résultats A-n65-k09, critère Seuil

	$Quan_{10}$				$Qual_{10}$				Tout			
	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop
50	3	73	43	0.59	$L_{lb}/2$	64	44	0.60	25	40	31	0.43
	4	61	40	0.55	$3L_{lb}/4$	39	29	0.40	38	14	9	0.13
100	5	70	44	0.6	$L_{lb}/2$	58	42	0.58	50	43	33	0.45
	8	63	41	0.56	$3L_{lb}/4$	36	28	0.39	75	15	10	0.14
500	25	71	43	0.59	$L_{lb}/2$	56	41	0.56	250	45	35	0.48
	38	60	40	0.55	$3L_{lb}/4$	35	28	0.39	375	14	9	0.13
8000	400	62	41	0.56	$L_{lb}/2$	56	40	0.55	4000	45	35	0.48
	600	15	14	0.33	$3L_{lb}/4$	35	28	0.39	6000	13	9	0.12

Si trop d'arêtes renvoyées $\rightarrow$ Solutions infaisables ? (futurs tests)

Résultats A-n65-k09, critère Rang

	$Quan_{10}$			$Qual_{10}$			Tout		
	Rang	Corr	Prop	Rang	Corr	Prop	Rang	Corr	Prop
50	10	6	0.08	10	7	0.1	10	7	0.1
	20	14	0.2	20	15	0.21	20	14	0.19
	33	23	0.32	33	26	0.36	33	24	0.33
100	10	6	0.08	10	7	0.1	10	7	0.1
	20	16	0.22	20	16	0.22	20	14	0.19
	33	26	0.36	33	26	0.36	33	25	0.34
500	10	7	0.1	10	7	0.1	10	6	0.08
	20	17	0.23	20	15	0.21	20	13	0.18
	33	27	0.37	33	26	0.36	33	25	0.34
8000	10	7	0.1	10	7	0.1	10	6	0.08
	20	17	0.23	20	17	0.23	20	13	0.18
	33	27	0.37	33	27	0.37	33	25	0.34

De nouveau les 3 bases renvoient des résultats similaires.

Résultats P-n101-k04, critère Seuil

	Quan$_{10}$				Qual$_{10}$				Tout			
	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop	Seuil	Arêtes	Corr	Prop
50	3	93	65	0.62	$L_{lb}/2$	83	66	0.64	25	71	61	0.59
	4	54	44	0.42	$3L_{lb}/4$	42	37	0.36	38	24	21	0.20
100	5	80	66	0.64	$L_{lb}/2$	79	66	0.63	50	72	62	0.60
	8	45	41	0.40	$3L_{lb}/4$	42	39	0.38	75	24	22	0.21
500	25	83	69	0.67	$L_{lb}/2$	81	68	0.66	250	72	63	0.60
	38	43	39	0.38	$3L_{lb}/4$	39	36	0.35	375	22	20	0.19
8000	400	87	73	0.7	$L_{lb}/2$	85	71	0.68	4000	70	60	0.58
	600	42	39	0.38	$3L_{lb}/4$	41	38	0.37	6000	23	21	0.2

Plus la taille de l'instance augmente, et plus la proportion d'arêtes optimales renvoyées présentes dans la solution optimale est grande.

Résultats P-n101-k04, critère Rang

	Quan$_{10}$			Qual$_{10}$			Tout		
	Rang	Corr	Prop	Rang	Corr	Prop	Rang	Corr	Prop
50	10	8	0.08	10	8	0.08	10	8	0.08
	20	18	0.17	20	17	0.16	20	18	0.17
	50	43	0.41	50	44	0.43	50	44	0.43
100	10	8	0.08	10	8	0.08	10	8	0.08
	20	18	0.17	20	18	0.17	20	18	0.17
	50	46	0.44	50	46	0.44	50	46	0.44
500	10	8	0.08	10	8	0.08	10	8	0.08
	20	18	0.17	20	18	0.17	20	18	0.17
	50	46	0.44	50	46	0.44	50	46	0.44
8000	10	8	0.08	10	8	0.08	10	8	0.08
	20	18	0.17	20	18	0.17	20	18	0.17
	50	46	0.44	50	46	0.44	50	46	0.44

Il faut choisir un rang dépendant de la taille de l'instance (rangs fixés à 10 ou 20 ne revoient plus de bons résultats).

Où intégrer la connaissance

Intégration de connaissance lors de la construction de la solution initiale :

- On regroupe des clients grâce aux connaissances extraites;
- On applique CW sur cette initialisation. On choisit l,m,n qui donne le meilleur CW dans l'échantillon.

Learning Heuristic

LearnHeuristic (LH)

$(\lambda^*, \mu^*, \nu^*)$, *Init* $\leftarrow$ *Apprentissage* *newBase* $\leftarrow$ []
for $i \leftarrow 1$ ***to*** 10 **do**
  if $i = 1$ **then**
    $Sol \leftarrow H_c(Init, I, D, \lambda, \mu, \nu)$
    $newBase \leftarrow newBase \cup Sol$
  else
    Déterminer *Init* avec les connaissances de *newBase*
    $Sol \leftarrow H_c(Init, I, D, \lambda, \mu, \nu)$
    $newBase \leftarrow newBase \cup Sol$
return *La meilleure solution*

Résultats

Choix pour apprentissage

- Taille échantillon : 100
- Base : Qual_{10}
- Critère : Rang $= n/2$

	A-n37-k06			A-n65-k09			P-n101-k04		
Connaissances	Best	Mean_5	Time	Best	Mean_5	Time	Best	Mean_5	Time
Sans	963	974	805	1189	1236	776	696	708	1739
Avec	950	966	1073 (3)	1186	1193	911 (8)	694	704	1533 (78)

Bilan

L'intégration de connaissance semble apporter de meilleurs résultats

Limites

- Choix de meilleurs valeurs pour l'apprentissage (Taille échantillon, base, critère);
- Pas beaucoup de sol pour nouvel apprentissage;
- Toujours même triplet utilisé.

Prochain algorithme