Présentation stage

Clément Legrand

July 2, 2018

Capacitated Vehicle Routing Problem

Notations

Instance I : n clients et 1 dépôt
Solution Sol : k tournées
La demande d_i du client i, une capacité C

Règles

- $\forall i > 0 \in I, \exists! R_j \in Sol, i \in R_j$;
- Chaque tournée doit partir et s'arrêter au dépôt;
- $\forall R_j \in Sol, \sum_{i \in R_j} d_i \leq C$.

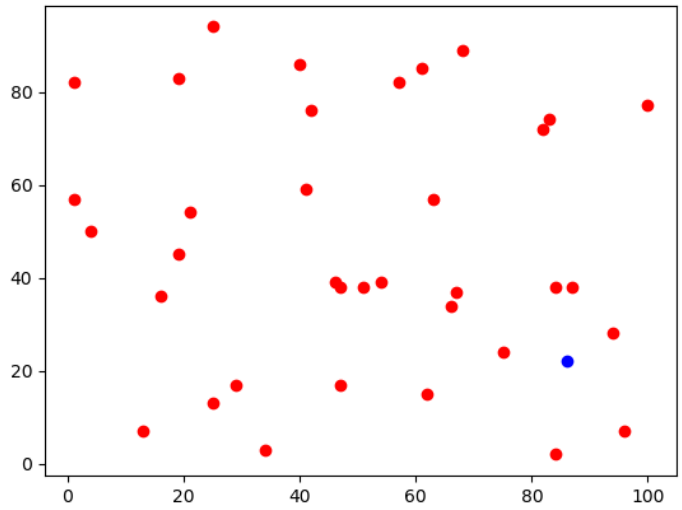
Objectif

Déterminer Sol tel que:

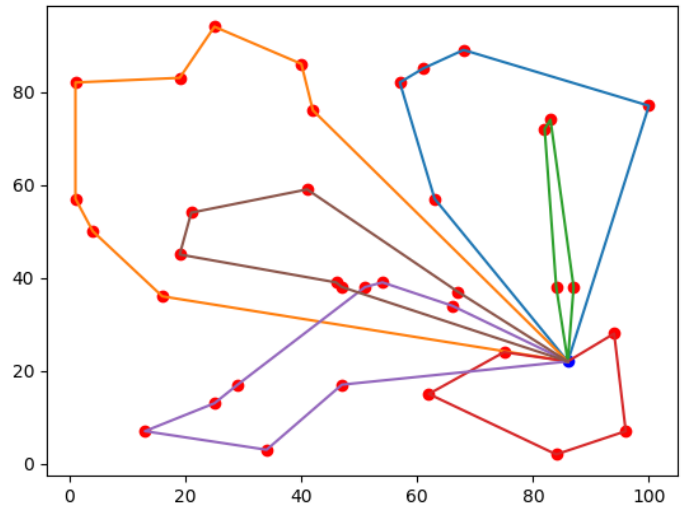
$$Sol = argmin_{Sol} \sum_{R_j \in Sol} \sum_{i=0}^{|R_j|-1} dist(R_j[i], R_j[i+1]) = argmin_{Sol} cost(Sol)$$

Exemple

Instance A-n37-k06:



Représentation instance



Meilleure solution connue

Objectif

Intégrer de la connaissance lors du calcul d'une solution

Idée

Prédire les arêtes optimales en apprenant à partir de solutions initiales de bonne qualité

Problèmes

- Comment construire une solution initiale de bonne qualité ?
- Quelle heuristique utiliser ?
- Comment extraire la connaissance ?
- Comment intégrer la connaissance dans l'heuristique ?

Algorithme Clarke & Wright (CW)

CW[1] $\rightarrow$ Algorithme glouton.

Définition saving

Calcul du saving de i et j avec:

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

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

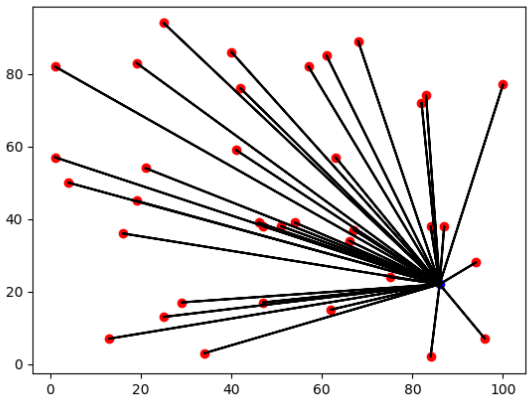
Fonctionnement

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

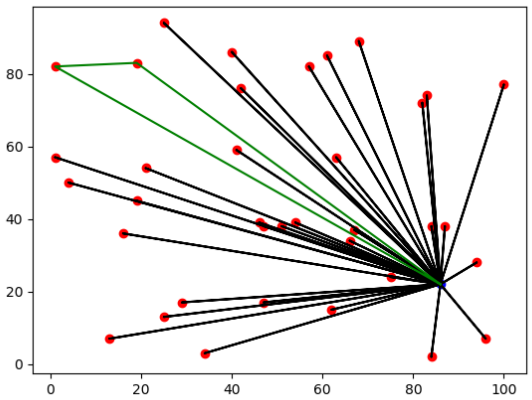
- $(i,j) \leftarrow argmax_{(i,j)} s(i,j)$;
- Les tournées qui contiennent i et j sont fusionnées (si possible);
- $s(i,j) \leftarrow 0$.

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

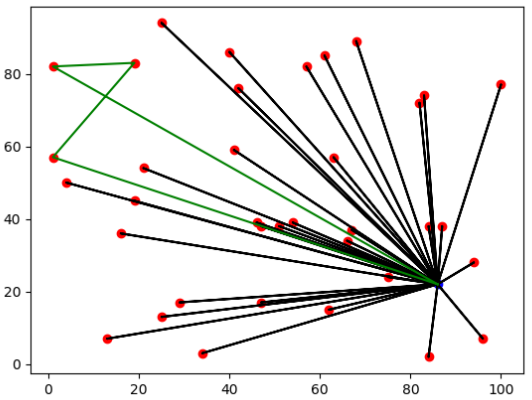
Exécution pour $(\lambda, \mu, \nu) = (1, 1, 1)$



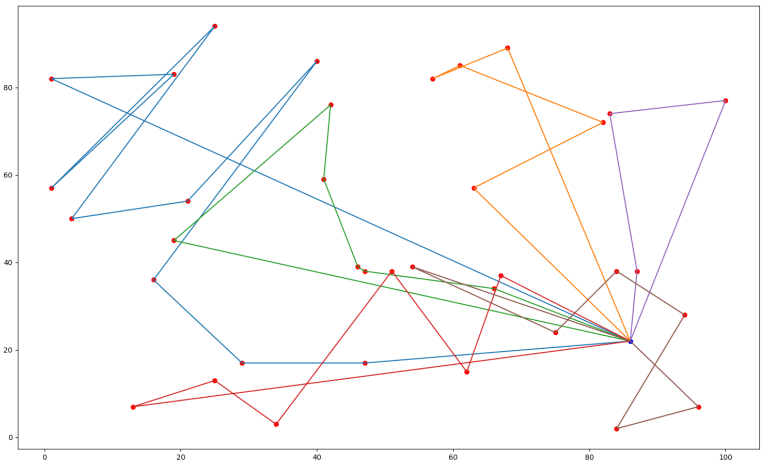
Initialisation



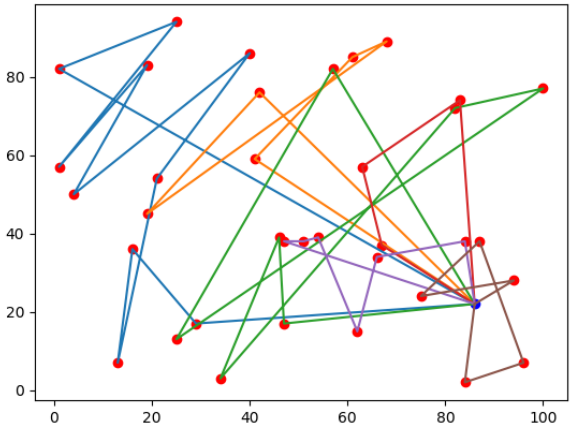
1^{ere} fusion



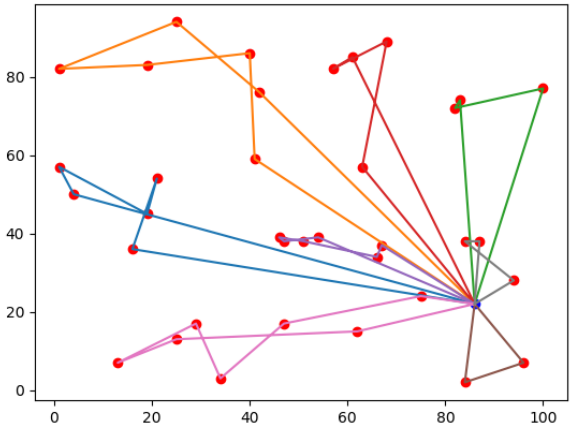
2^{eme} fusion



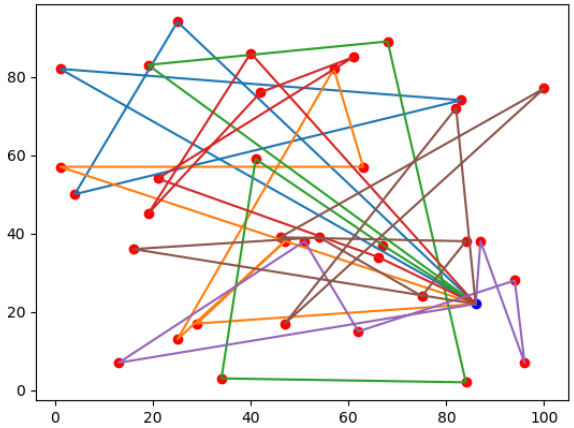
Choix de (λ, μ, ν) ?



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



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



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

Bilan

Difficile de prévoir l'influence des paramètres

Heuristique Arnold & Sörensen

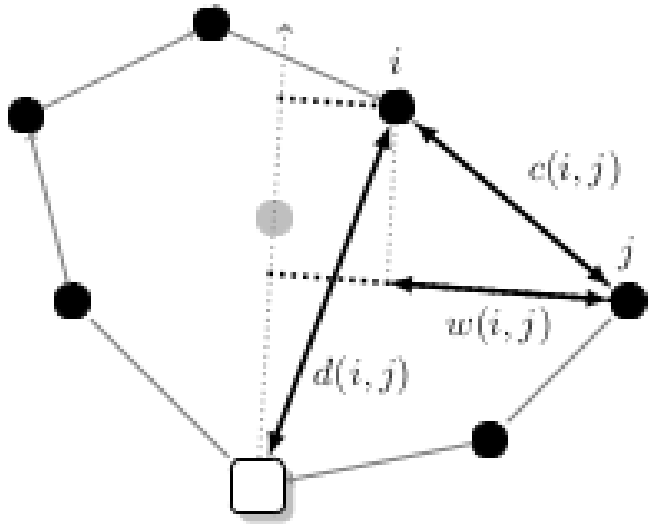
```
1  Sol ← CW(λ, μ, ν)
2  NewSol ← Sol
3  while Pas d'améliorations depuis 3 min do
4  |    Calcul de la pire arête
5  |    NewSol ← EjectionChain_{BI−O}
6  |    NewSol ← LinKernighan_{BI−O}
7  |    NewSol ← CrossExchange_{BI−O}
8  |    NewSol ← LinKernighan_{BI−O}
9  |    if cost(NewSol) < cost(Sol) then
10 |    |    Sol ← NewSol
11 return Sol
```

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 locaux

Ejection-chain

Déplacer / clients sur des tournées.

Cross-exchange

Échanger deux séquences de clients entre deux tournées.

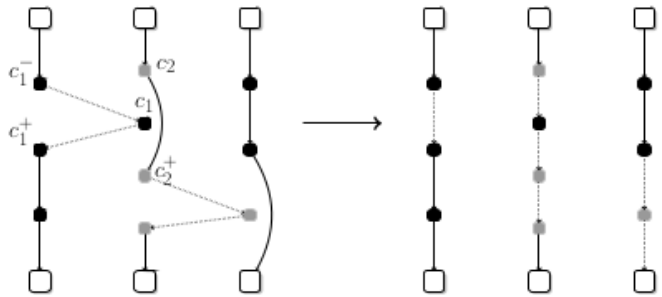


Figure 2: Illustration of the ejection chain with two relocations.

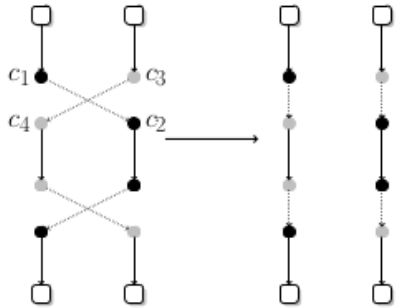
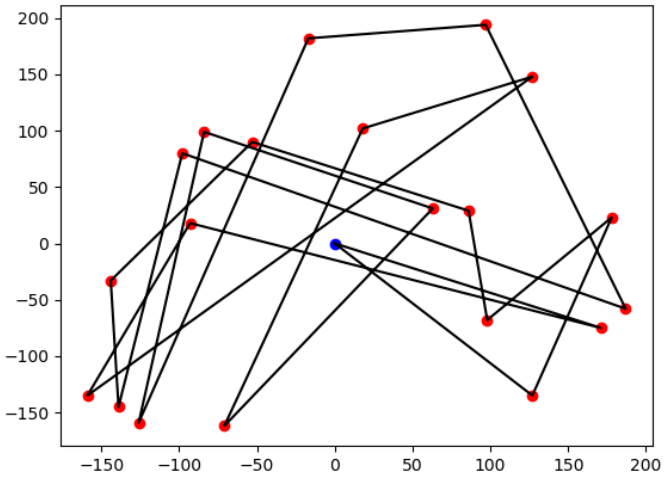


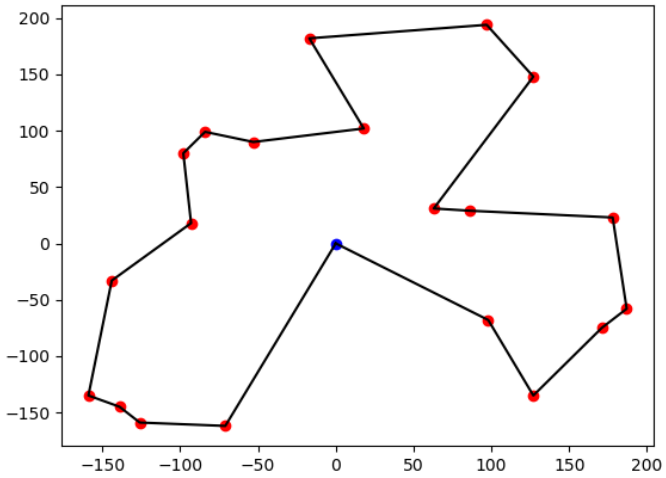
Figure 1: Illustration of the CROSS-exchange with sequences of two customers.

Opérateurs locaux

Lin-Kernighan

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





Heuristique utilisée (H_c)

```
1  Sol ← CW(λ, μ, ν)
2  NewSol ← Sol
3  while La dernière amélioration date de moins de n/3 min do
4      Calcul de la pire arête
5      NewSol ← EjectionChain_{FI−RD}
6      NewSol ← LinKernighan_{BI−O}
7      NewSol ← CrossExchange_{FI−RD}
8      NewSol ← LinKernighan_{BI−O}
9      if cost(NewSol) < cost(Sol) then
10         Sol ← NewSol
11     if Pas d'améliorations depuis n/2 itérations then
12         NewSol ← Sol
13 return Sol
```

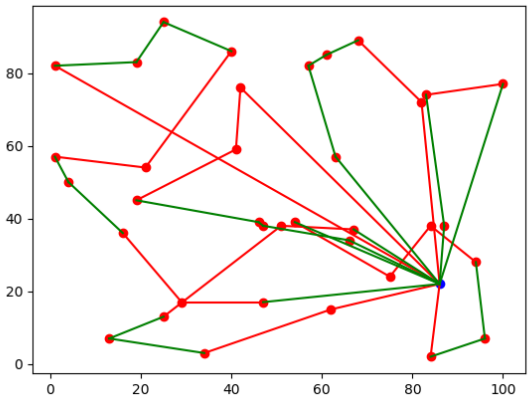
Validation

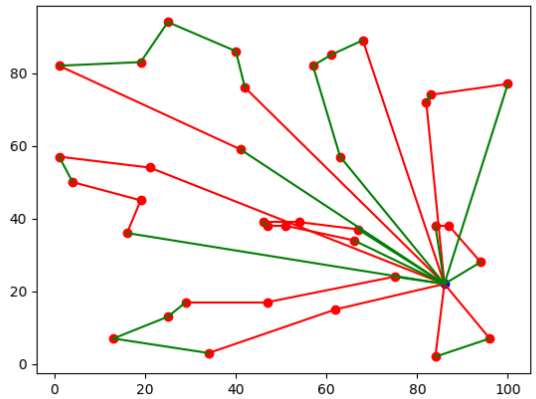
	A-n37-k06			A-n65-k09			P-n101-k04		
Ajout	Best	Mean	Time	Best	Mean	Time	Best	Mean	Time
Rien	950	957	195	1197	1215	395	722	736	783
Divers	950	969	200	1200	1230	350	698	706	1500

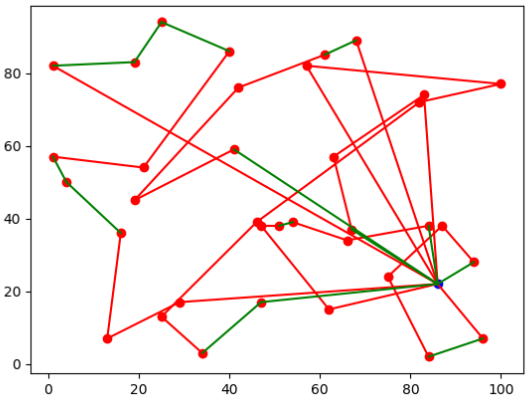
Conclusion

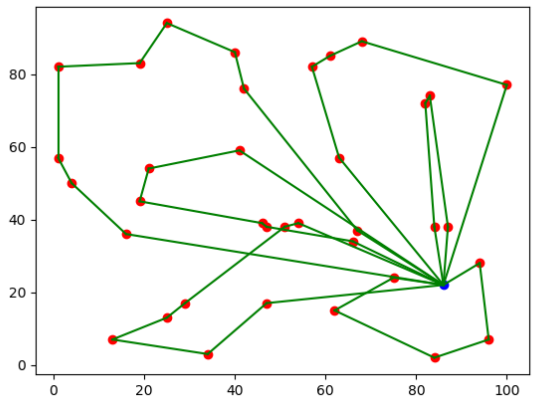
Diversification plus intéressante pour des grandes instances

Exemples









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;

Quelles solutions pour apprendre ?

- 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);

Comment choisir les arêtes à conserver ?

Pour chaque arête (i,j), on incrémente la valeur de MAT[i][j];

- Conserver $(i, j) \Leftrightarrow MAT[i][j] > seuil$ (Seuil);
- Conserver les rg premières arêtes dans la matrice (Rang).

Résultats

Instance A-n37-k06

	$Quan_{10}$				$Qual_{10}$		
	Seuil	Arêtes	Correctes	Proportion	Rang	Correctes	Proportion
50							
100							
500							
8000							

Résultats

Instance A-n65-k09

	$Quan_{10}$				$Qual_{10}$		
	Seuil	Arêtes	Correctes	Proportion	Rang	Correctes	Proportion
50							
100							
500							
8000							

Résultats

Instance P-n101-k04

	$Quan_{10}$				$Qual_{10}$		
	Seuil	Arêtes	Correctes	Proportion	Rang	Correctes	Proportion
50							
100							
500							
8000							

Description

Résultats

Conclusion

Ajouter de la diversification dans l'apprentissage