

# Nouvelle heuristique pour VRP

Clément Legrand

31 Mai 2018

# Description

Intérêt : heuristique simple à mettre en place, et performante.

## Etapas de l'algorithme :

- Recherche solution initiale : Algorithme Clarke and Wright
- Recherche de la "pire" arête
- Optimisations locales ensuite par 3 opérateurs

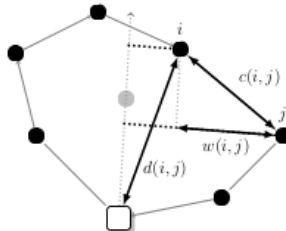
# Comment caractériser une arête ?

## Idée

Trouver des liens entre les solutions (quasi-) optimales

Entraînement d'un modèle prédictif pour distinguer les arêtes à supprimer des autres en s'aidant de caractéristiques :

- Largeur de la tournée
- Profondeur de la tournée
- Coût de la tournée



## Déterminer la pire arête

Description spatiale précise d'une arête avec ces caractéristiques.

$$b(i,j) = \frac{[\lambda_w w(i,j) + \lambda_c c(i,j)] \left[ \frac{d(i,j)}{\max_{k,l} d(k,l)} \right]^{\frac{\lambda_d}{2}}}{1 + p(i,j)}$$

- $p$  représente le nombre de fois où l'arête a été pénalisée (initialement 0)
- Les paramètres  $\lambda_w, \lambda_c$  et  $\lambda_d$  valent 0 ou 1, et sont choisis selon le type d'instance.

### Pire arête

La pire arête est celle qui maximise la fonction  $b$ .

Agit sur un couple de tournées, et essaie d'échanger toutes paires de séquence de clients consécutifs entre les deux routes. Complexité en  $O(n^4)$  : beaucoup trop...

Réduction : si on connaît une arête à éliminer, on choisit la tournée correspondante. Et on ne s'intéresse qu'aux tournées qui ont des nœuds dans le voisinage de l'arête.

Complexité en théorie quadratique, mais proche de linéaire en général (beaucoup de solutions violent les contraintes)

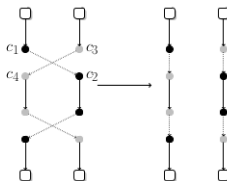


Figure 1: Illustration of the CROSS-exchange with sequences of two customers.

Agit (potentiellement) sur toutes les tournées. Commence par déplacer un client de la route  $a$  vers la route  $b$ , de même en partant de  $b$  vers  $c$ , jusqu'à  $l$  déplacements.

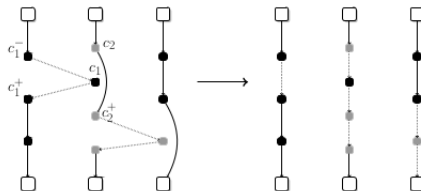


Figure 2: Illustration of the ejection chain with two relocations.

## Utilisation

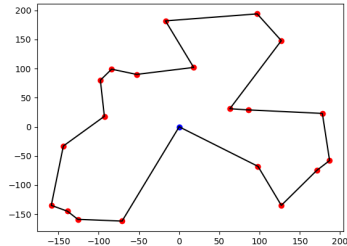
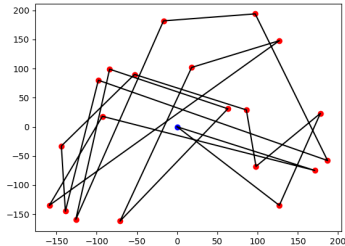
- Utilisé en général pour TSP ;
- Optimisation intra-tournée (chaque tournée est améliorée indépendamment des autres).

## Implémentation

- Exécute  $i$ -opt (échange l'ordre de  $i$  clients sur la tournée).
- Si  $i = k$  ou plus d'améliorations possibles  $\rightarrow$  réalise la meilleure  $i$ -opt ; recommence avec  $i = 2$
- Si amélioration possible  $\rightarrow$  recommence avec  $i + 1$
- Si  $i = 2$  et pas d'améliorations possibles  $\rightarrow$  on sort de la boucle.

## Complexité

Complexité en  $O(n^k)$ ,  $k$  choisi par l'utilisateur. En général exécution sur des morceaux de tournée.





# Que faire si on ne trouve plus d'améliorations ?

## Idées

- Optimisation globale (application des opérateurs sur l'ensemble du graphe) : très coûteux
- Remise à zéro des pénalités
- Changement de la fonction de pénalisation