Nouvelle heuristique pour VRP

Clément Legrand

31 Mai 2018 [1]

[1] D'après l'article *A simple, deterministic, and efficient knowledge-driven heuristic for the vehicle routing problem* de Florian Arnold et Kenneth Sorensen

Description

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

Etapes de l'algorithme:

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

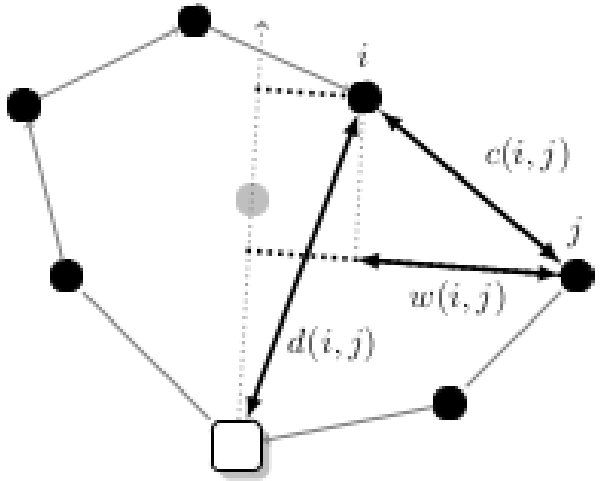
Comment caractériser une arête ?

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.

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

- p représente le nombre de fois où l'arête a été pénalisée (initialement 0)
- Les paramètres λ_w,λ_c et λ_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.

Cross-exchange

Objectif

Essaie d'échanger deux séquences de clients entre deux tournées.

Complexité

Complexité en $O(n^4)$: beaucoup trop...

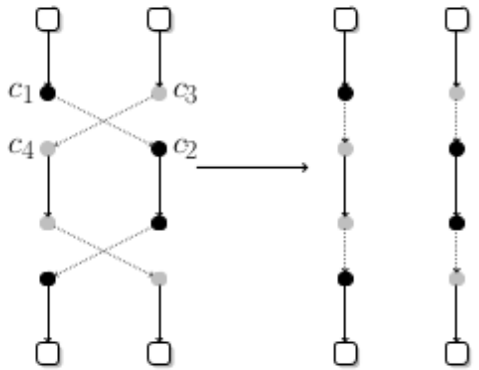


Figure 1: Illustration of the CROSS-exchange with sequences of two customers.

Amélioration locale

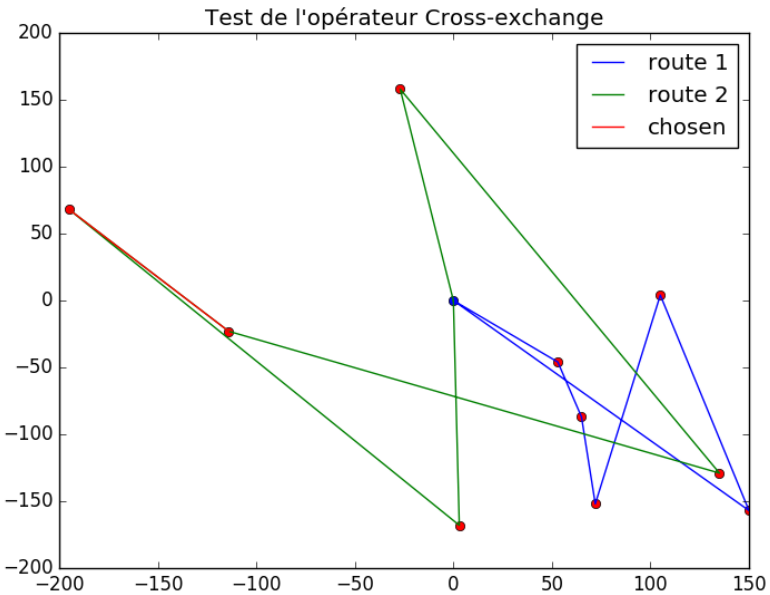
- Choisir une arête (c_1, c_2) à éliminer;
- Trouver une autre tournée grâce aux voisins de c_1; (e.g. premier voisin c_4 sur une tournée différente, sur arête (c_3, c_4));
- Premier échange: (c_1, c_4) et (c_3, c_2);
- On essaie d'échanger $n(c_4)$ avec $n(c_2)$, $n(n(c_2))$, ... jusqu'à trouver une amélioration ou avoir parcouru toute la tournée (dans ce cas on poursuit avec $n(n(c_4))$).

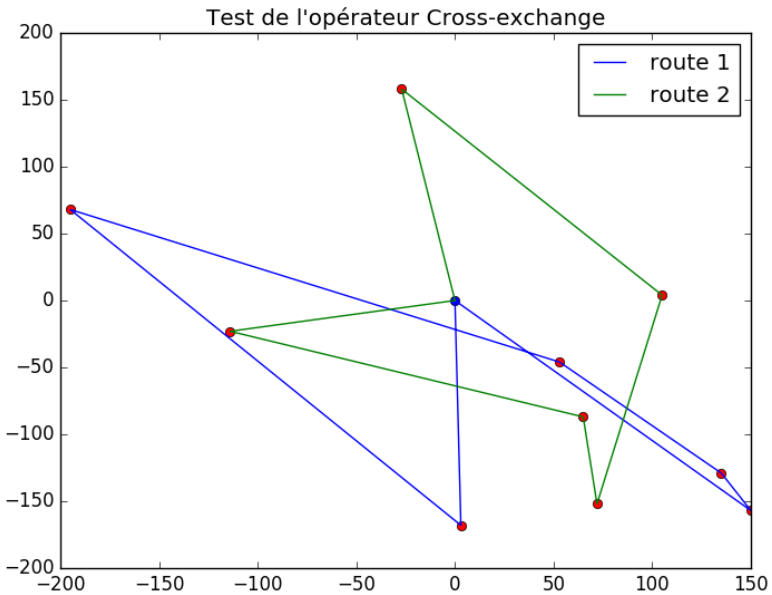
Nouvelle complexité

$O(n^2)$ dans le pire cas, mais proche de linéaire en général.

Exemple

Test de l'opérateur sur une tournée de 10 clients. L'arête rouge est l'arête à éliminer.





Ejection-chain

Objectif

Déplacer au plus / clients sur des tournées plus adaptées.

Implémentation

- Déterminer une arête à éliminer; considérer un des deux clients;
- Regarder parmi les pp-voisins pour trouver une tournée;
- Déplacer le client sur cette tournée;
- Essayer de déplacer un client de cette tournée sur une autre tournée, tel que le saving soit positif;
- Recommencer l'étape précédente / fois.

Complexité

Complexité en $O(n^{l-1})$. En pratique $l = 3$.

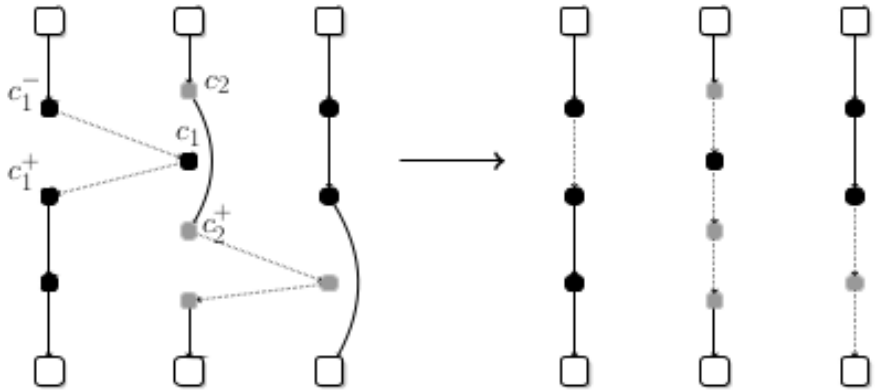


Figure 2: Illustration of the ejection chain with two relocations.

Lin-Kernighan Heuristic

Utilisation

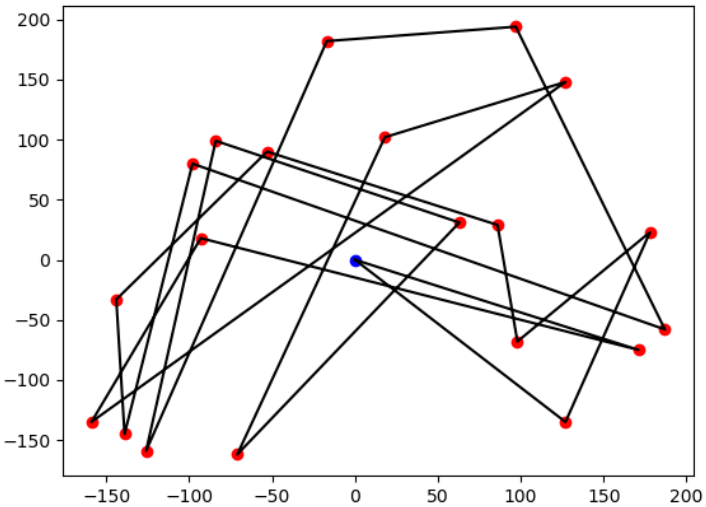
- Utilisé en général pour TSP;
- Optimisation intra-tournée (chaque tournée est améliorée indépendamment des autres).

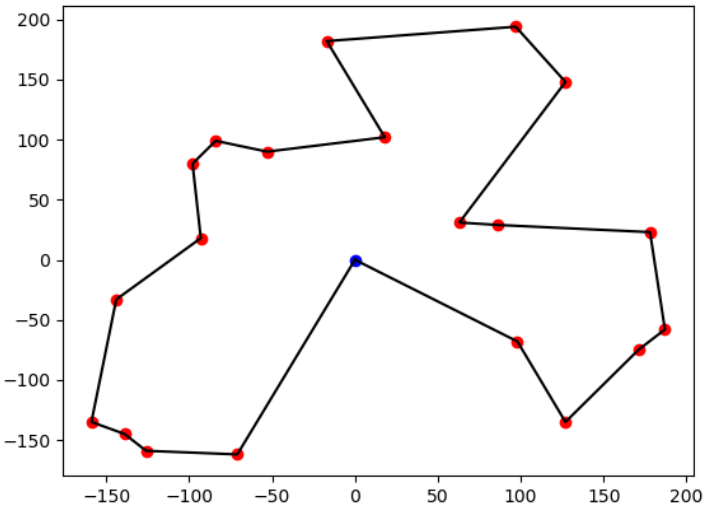
Implémentation

- Exécute i-opt (échange l'ordre de i clients sur la tournée; au départ $i = 2$).
- 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