Nouvelle heuristique pour VRP

Clément Legrand

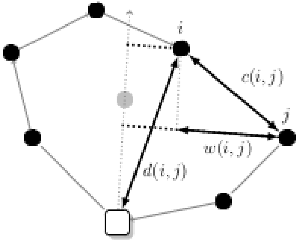
28 février 2018

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

Lien entre les solutions (presque) optimales ?
Modèle prédictif qui distingue les solutions en s'aidant de caractéristiques :

- Largeur de la route
- Profondeur de la route
- Compacité de la route : moyenne des distances de chaque client au centre de gravité de la route.



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.

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)

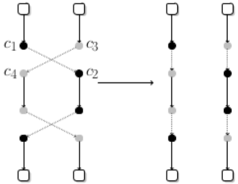


Figure 1: Illustration of the CROSS-exchange with sequences of two customers.

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*, jusque *l* déplacements.

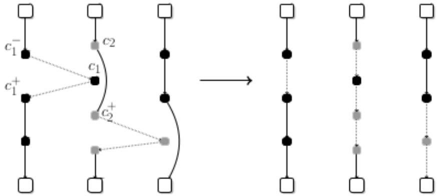


Figure 2: Illustration of the ejection chain with two relocations.

Utilisé en général sur TSP. Optimisation intra-tournée (chaque tournée est améliorée indépendamment ds autres). LK essaie de trouver un déplacement k-opt, pour améliorer la solution. Il commence par 2-opt, puis passe à 3-opt seulement si une meilleure solution a été trouvée. S'il atteint k, il exécute la meilleure i-opt et recommence à 2.
Complexité en $O(n^k)$. En général les changements sont mineurs et l'algo s'arrête après 2-opt.

Si on ne trouve plus d'améliorations

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