Prise de notes : Vehicle Routing Problem

Clément Legrand

Une heuristique simple, déterministe, et guidée par connaissance

Description VRP = NP difficile Nombreux algos au cours des dernières années : création d'heuristiques difficiles à ré-exploiter.

Principe général : opérateurs locaux → optima locaux. (parcours de toutes les solutions possibles).

Opérateurs Création d'une heuristique simple à mettre en place. 3 opérateurs retenus :

- CROSS-exchange (CE) : optimise le couple de routes passé en argument (pour une configuration déjà donnée). Nb de solutions trop important : ne s'intéresser qu'à une partie des routes. Pour savoir si on supprime une route, on considère les nœuds les plus proches des deux nœuds adjacents. Performance quadratique, mais pire cas rare.
- Ejection-chain (EC) : Essaie d'attribuer des usagers à d'autres routes que celle de départ. Il essaie de faire le max de mvmts en un seul parcours (il part d'une route, choisit un usager le fait passer par une autre route, puis poursuit sur cette nouvelle route tout en pouvant revenir sur la première). On suppose qu'on connaît les arêtes à supprimer. On garde en mémoire les changements possibles grâce à $s(c_1,c_2)$, si elle est positive on garde en mémoire le changement (on insère c_1 après c_2). Plusieurs chaînes possibles → structure d'arbre, on s'arrête à la profondeur l (ie l relocalisations). On exécute la branche qui maximise s.
- Lin-Kerninghan Heuristic (LK) : Sert à la résolution du pb du voyageur de commerce. Permet d'optimiser les routes indépendamment des autres (optimisation intra-route). Cette heuristique commence par faire une 2-opt, puis 3-opt, jusqu'à atteindre une k-opt (on exécute la meilleur k-opt). Puis recommence tant qu'on peut améliorer.

Détection de mauvaises arêtes Les opérateurs ne peuvent être appliqués que si l'on connaît les arêtes à supprimer. Utilisation de GLS : Guided Local Search. On pénalise les arêtes via une fonction.

L'objectif était de dégager des propriétés (caractéristiques) sur les solutions optimales pour les distinguer des solutions non-optimales. Métriques définies : basées sur des propriétés géométriques, largeur de la route (distance maximale entre toutes les paires d'usagers, mesurée le long de la perpendiculaire à la ligne joignant le dépôt au centre de gravité de la route (moyenne des coord de chaque points)),profondeur des routes (distance entre le dépôt et le plus lointain usager), et compacité (moyenne des distances de chaque usager au centre de gravité de la route), nb intersections sur les arêtes. Puis "Classification learner". Il utilise un certain nb de "datapoints" choisis aléatoirement, pour entraîner un modèle prédictif (distingue les solutions optimales des non-optimales). Si précision (nb de points classifiés correctement/nb total de pts). → précision de 93% (métriques permettent de distinguer).

3 métriques sont retenues pour pénaliser les arêtes (largeur, coût, profondeur). On définit une fonction décrivant l'état de l'arête : plus $b(i,j)$ est élevée et plus l'arête est "mauvaise". On pénalise ensuite l'arête qui a le plus grand b, en incrémentant $p(i,j)$.

Mise en place de l'heuristique Heuristique divisée en plusieurs étapes :

- Solution initiale avec heuristique Clarke-Wright.
- Identification de la "pire" arête via b.
- Optimisation locale avec CE et EC. Plus l'arête a été pénalisée et plus elle risque d'être supprimée.
- Utilisation de LK pour ré-optimiser les routes.
- On recommence l'étape 2.

Optimisation globale : ne pas se restreindre à l'arête à éliminer. On applique les opérateurs à tout le réseau. Très coûteux... Appliquer lorsque plus d'améliorations locales. Puis reprendre ensuite l'heuristique.

Reset et pénalisation adaptative : Accumulation de pénalités pas bonne → effacement de toutes les pénalités et retour à la dernière meilleure solution globale. Ces resets permettent aussi d'adapter la fonction de pénalité en jouant sur les paramètres de b. Pour cela on teste chaque fonction de pénalité dans l'ordre de la meilleure à la pire (théorique), en changeant lorsqu'il n'y a plus d'améliorations. Si on trouve un nouvel optimum global, la fonction de pénalité actuelle devient la meilleure.

Analyse et vérification de l'heuristique Heuristique à la fois minimale (aucune composante n'est inutile) et efficace (la performance justifie le calcul). Courbes → Impact de toutes composantes non négligeables. 2 min → même impact si composantes prises séparément. Heuristique plus efficace (de peu) que la meilleure heuristique classique (qui ne pénalise que la longueur, ie le coût). Heuristique complètement déterministe (beaucoup plus simple à évaluer).

Extension à d'autres problèmes VRP Heuristique développée : s'adapte facilement à d'autres problèmes plus généraux de VRP. Ajout de contraintes facile à mettre en place (opérateurs). La fonction de pénalisation est adaptative. Application au pb du multi-dépôt → L'aspect "depth" des arêtes est plus important ici, car un usager à tendance à se rapprocher du dépôt le plus proche de lui.

Résultats Test de l'heuristique sur des instances "Golden", "Uchoa" (VRP) et "Cordeau" (MDVRP).

Paramètres (N = customers) :

— 3 relocations dans EC
— calcul des 30 plus proches voisins
— global optimization apres N/10 non améliorations. (N/5 for MDVRP)
— penalization function changed après 20N non améliorations
— resets apres 100N non améliorations

Globalement calcul de solutions en temps linéaire.

Amélioration de l'heuristique de Clarke et Wright pour CVRP

Description Algo de Clarke et Wright (CW) très populaire (simple et plutôt efficace). Méta-heuristiques plus efficaces mais bcp de paramètres à trouver. Idée : ajout d'informations dans l'heuristique en plus de la distance.

Algo : au depart chaque client est visité par un véhicule différent. Puis on fusionne les routes si $s = c_{i0} + c_{j0} - c_{ij}$ est max.

Si i et j proches s est grand $\rightarrow$ fausse les résultats... Ajout de paramètres :

- λ prend en compte la circonférence des routes.
- μ témoigne de l'asymétrie entre i et j, en prenant en compte leur distance au dépôt.
- Nouvelle idée : prendre en compte la demande des clients dans le saving. Les usagers qui ont une demande cumulée proche de la capacité de stockage sont prioritaires. Paramètre ν.

Résultats : tps de calcul plus long, mais meilleure solution la plupart du temps. Choix des paramètres : calculs pour des valeurs $[0; 2]$ avec pas de 0.1, on garde le meilleur résultat.