ÉQUIPE CENTRALE-OPAL

Rapport Optimisation et Logiciels : Études de Cas en transport

Élèves :
CARDINAL JULIEN
RONDIER LUCIEN
JAMAL FEDOUA
SOUILAH ADEL

Encadrants :
NAGIH ANASS
KOKSAL BURAK
Akbalik-Rapine AYSE

Année universitaire 2023-2024

22 février 2024

Table des matières

1) Presentation du Problème

a. Introduction au problème

Dans un monde où l'efficacité logistique est synonyme de succès concurrentiel, l'optimisation des itinéraires de livraison est un défi majeur pour les entreprises de distribution. Ce projet s'attaque à une composante critique de ce domaine : the capacitated vehicle routing problem (CVRP). Notre projet se penche sur la conception d'un planning de distribution optimisé pour répondre aux besoins des clients d'un cas industriel réel. Le CVRP ne se limite pas à la simple minimisation de la distance parcourue ou du coût total. Il s'agit de développer un modèle qui prend en compte la capacité des véhicules, et d'autes facteurs tels que la diversité des types de véhicules et les contraintes de coûts fixes et variables. Avec des données réalistes fournies par un acteur industriel anonyme, nous allons fournir un plan de distribution qui soit à la fois réalisable et optimal. Notre approche méthodologique est double : nous développons un modèle(PLNE) et explorons des heuristiques pour des solutions approchées. Ces méthodes seront implémentées et testées à l'aide d'IBM Cplex, fournissant une comparaison rigoureuse entre les solutions heuristiques et les solutions optimales. Ce rapport documente notre processus de modélisation, nos méthodes de résolution, et offre une évaluation critique de la performance de nos heuristiques face à la complexité et à la taille réelle des instances de CVRP.

b. Données du problème

Voici une version partiellement modifiée des données de notre problème :

	1	2	3
0	161.432	106.605	114.983
1	0	88.993	54.0271
2	88.993	0	38.1777
3	178.864	112.212	0

TABLE 1 – Distance Matrix (truncated)

Customer	Demand
0	0
1	10
2	7
3	5

TABLE 2 – Customer Demand (truncated)

Customer	Coordinates
0	(48.7435, 6.20295)
1	(48.541, 7.64312)
2	(48.9635, 6.08812)

TABLE 3 – Customer Coordinates (truncated)

Au cours de notre projet nos algorithmes ont tous été testés avec succès sur des tables de données de plus en plus complexes et larges. Par exemple, à partir du 16ème jeux de données le nombre de véhicules disponibles par type de véhicule dépasse 1.
Sur la table ci-dessous se trouve les données spécifiant chaque caractéristique des véhicules :

	Truck	Motorcycle	Bicycle	Short-Term Bicycle	Short-Term Truck
No. Vehicles	1	1	1	1	1
Capacity	40 kg	15 kg	10 kg	-	-
Avg Speed	40 km/h	50 km/h	10 km/h	-	-
Fixed Cost	180 €	100 €	60 €	250 €	350 €
Soft Time	6 h	6 h	6 h	-	-
Hard Time	8 h	8 h	8 h	-	-
Soft Dist.	300 km	300 km	150 km	-	-
Hard Dist.	-	-	-	75 km	-
Dist. Penalty	0.5 €/km	0.5 €/km	0.5 €/km	-	-
Time Penalty	50 €/h	40 €/h	30 €/h	-	-

TABLE 4 – Vehiclue Specifications

2) Modélisation Linéaire du Problème

a. Modélisation Mathématiques

Variables de décision :

- x_{ijk} : variable binaire, 1 si le véhicule k se déplace de i à j, 0 sinon.
- y_{ik} : variable binaire, 1 si le véhicule à court terme k visite le client i, 0 sinon.
- d_k : variable continue représentant la pénalité de distance pour le véhicule k.
- t_k : variable continue représentant la pénalité de temps pour le véhicule k.
- u_{ik} : variable continue pour l'élimination de sous-tour pour le véhicule k.

Fonction objectif

Minimiser le coût total incluant les coûts fixes pour l'utilisation des véhicules et des véhicules à court terme, les pénalités de distance et de temps :

$$\min \left(\sum_{k=1}^{K_{ST}} \sum_{i=2}^{N} c_{ST,k} \cdot y_{ik} + \sum_{k=1}^{K} \left(p_{d,k} \cdot d_k + p_{t,k} \cdot t_k + \sum_{j=2}^{N} c_{V,k} \cdot x_{1jk} \right) \right)$$

Contraintes

Contraintes de temps

$$\sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{dist_{ij}}{s_k} \cdot x_{ijk} \leq HTL_k, \forall k$$
$$\sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{dist_{ij}}{s_k} \cdot x_{ijk} - STL_k \leq t_k, \forall k.$$

Contraintes de distance

$$\sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} dist_{ij} \cdot x_{ijk} - SDL_k \leq d_k, \forall k$$
$$\sum_{i=2}^{N} dist_{1i} \cdot y_{ik} \leq HDL_{ST,k}, \forall k \text{ pour short-term vehicles.}$$

Contraintes de flux

$$\sum_{j=1, j\neq i}^{N} x_{ijk} - \sum_{j=1, j\neq i}^{N} x_{jik} = 0, \forall i, k.$$

Contrainte de visite

$$\sum_{k=1}^{K} \sum_{i=1, i\neq j}^{N} x_{ijk} + \sum_{k=1}^{K_{ST}} y_{jk} = 1, \forall j.$$

Contrainte d'utilisation	$\sum_{j=2}^{N} y_{jk} \leq 1, \forall k$ pour short-term vehicles(excepted Truck) $\sum_{j=2}^{N} x_{1jk} \leq 1, \forall k$ pour long-term vehicles
Elimination sous-tours	$u_{ik} - u_{jk} + Q_k \cdot x_{ijk} \leq Q_k$ - $q_j, \forall i \neq j, k$ $q_i \leq u_{ik} \leq Q_k - q_i, \forall i > 1, k$ $u_{0k} = 0, \forall k.$

Notation and Variable Descriptions

— N : Nombre total de sommets ou de lieux, y compris le dépôt.
— K, K_{ST} : Nombre total de véhicules long terme (resp. court terme)
— $c_{V,k}$, $c_{ST,k}$: Coût fixe pour l'utilisation d'un véhicule long terme (resp. court terme)
— $p_{d,k}$, $p_{t,k}$: Taux de pénalité pour la distance (resp. le temps) pour le véhicule k.
— HTL_k : Limite de temps maximale pour le véhicule à long terme k.
— $HDL_{ST,k}$: Limite de distance maximale pour les véhicules à court terme k.
— STL_k, SDL_k : Limite de temps (resp. distance) souple pour le véhicule k, au-delà de laquelle une pénalité de temps t_k (resp. d_k) est encourue.
— $dist_{ij}$: Distance de l'emplacement i à l'emplacement j.
— s_k, Q_k : Vitesse (resp. Capacité) du véhicule k.
— q_i : Demande à l'emplacement i.

b. Résolution à l'aide du solveur CPlex

Performances et Limitations

Nos observations montrent que CPLEX, tout en étant performant sur les petites instances avec une capacité à résoudre à l'optimum, présente des limitations en termes de temps de calcul lorsque confronté à des instances de taille moyenne ou grande. Cette situation est attribuée à la complexité du solveur, qui, en raison d'une sur-ingénierie, effectue un grand nombre de vérifications internes, réduisant ainsi son efficacité sur des problèmes de grande envergure. Par exemple dès le 2ème tableau de données le temps de résolution du sovleur dépasse les 10 minutes.

Approche Hybride

Afin de surmonter ces limitations, l'idée d'adopter une approche hybride CPLEX/-heuristique a été envisagée. Cette stratégie consisterait à utiliser CPLEX pour les composantes du problème bien adaptées à l'optimisation exacte, tout en recourant à des méthodes heuristiques pour les aspects du problème où CPLEX se montre moins efficace. L'utilisation de l'API de C++ pour cette approche hybride est justifiée par la nécessité d'uniformiser nos données et nos résultats. Finalement nous n'avons pas fait le choix d'une approche hybride CPLEX/heuristique car nous trouvé une autre résolution exacte du problème, que nous appelerons "méthode constructive".

3) Méthode constructive

a. Décomposition Hierarchique du problème

Présentation des sous-problèmes

L'idée est de séparer le problème de CVRP en deux problèmes unitaires différents :
- Un problème maitre de K-partition entre K véhicules avec respect des contraintes de capacité
- K sous-problèmes de voyageur de commerce pour cacluler la distance minimale de tournée de chaque véhicule, respectant les contraintes de distance et de temps.

Avec le problème de K-partition :

$$\min(\sum_k(\sum_i x_{i,k} \geq 1)c_k + \sum_k \max(\frac{\text{distTour}(k)}{s_k} - \text{STL}_k, 0)p_{t,k}) + \max(\text{distTour}(k) - \text{SDL}_k, 0)p_{t,k})$$

Contraintes de Capacité	$\sum_{i=1}^{N} x_{ik}q_i \leq Q_k \forall i, k.$
Contrainte de visite	$\sum_{k=1}^{K} x_{ik} = 1, \forall i.$
Contrainte de distance	$\text{distTour(k)} \leq \text{HDL}_{ST,k} \forall k$
Contrainte de temps	$\frac{\text{distTour(k)}}{s_k} \leq \text{HTL}_k \forall k$

Avec distTour(k) une fonction spéciale faisant appel à un solveur du problème de voyageur de commerce pour les clients associés au véhicule k. Pour réaliser une recherche exhaustive de solution, la complexité est en $O(K^N)$) en supposant que le TSP se résout en temps constant

Fonctionnement global

Le problème de K-partition fait donc appel à chaque itération à K solveur du Voyageur de commerce, un pour les clients de chaque véhicule. Il est possible d'appeler à chaque fois le solveur mais on risque d'appeler plusieurs fois le solveur pour la même instance, au cours de l'exécution. L'idée, pour les petites instances, est donc de stocker tous les résultats du TSP dans un tableau que l'on peut d'ailleurs calculer à l'avance.
La résolution du TSP se fait par brute force, pour une seule instance, la complexité de l'algorithme est de $O(n!)$ avec n le nombre de noeuds du graphe. Calculer les résultats possibles des TSP pour tous les sous-ensemble de ⟦1,N⟧ est donc en $O(2^N N!)$
Cette méthode est efficace et résout les problèmes à 10 clients intantannéments et ceux à 15 clients en une dizaine de minutes.

b. Algorithme de Held-Karp pour résoudre le TSP

Le calcul du tableau des résultats du TSP est très gourmand en temps de calcul et il existe une façon de faire beaucoup mieux. L'algorithme de Held-Karp résout le TSP en $O(n^2 2^n)$ grâce à la programmation dynamique. Il stoque dans un tableau de taille $n2^n$ les distances du meilleur chemin partant de la base, passant par un sous-ensemble

de client J et finissant au client i. À chaque fois, pour calculer cette valeur, il prend : $C(J,i) = \min_{j \in J}(C(J \backslash \{j\}, j) + \text{dist(j,i)})$
Le gros avantage pour nous de cet algorithme est que par construction on peut en extraire directement les résultats de tous les TSP des sous-ensembles de de ⟦1,N⟧.
Cette méthode est efficace pour résoudre tous les problèmes à 10, 15 et 20 clients. Au-delà la complexité exponentielle pose vraiment un problème.

c. Majorant pour le TSP

Un idée est donc de s'affranchir du calcul exact du TSP et de ne calculer plus qu'un majorant de la solution du TSP pour les sous-ensembles trop grands. Il y a plusieurs façon de calculer un majorant du TSP mais la manière qu'on a choisie ici est d'utiliser des arbres couvrant minimaux. Ils sont calculés grâce à l'algorithme de Kruksal qui est en $O(n^2 \log(n))$ il suffit de parcourir deux fois l'arbre couvrant pour avoir un majorant du TSP. Une amélioration est de prendre des raccourcis et de sauter des noeuds quand on revient en arrière.

d. Algorithme Heuristique pour le problème de partition

Il suffit maintenant d'appliquer une heuristique constructive pour faire une recherche intélligente de solution de K-partition. On a pensé à implémenter un algorithme de Branch and Bound ou un algorithme de MCTS, qui n'ont pas abouti faute de temps.

4) Méthode de recherche locale

a. Présentation théorique de la recherche Tabu

La recherche tabou est une procédure itérative qui vise à construire un voisinage étendu, avec l'évitement des optima locaux. Cette méthode consiste à explorer l'espace de recherche en passant d'une solution à son meilleur voisin. Pour éviter les cycles, les solutions récemment examinées sont interdites ou déclarées taboues pendant un certain nombre d'itérations. La liste tabou est une file d'attente ordonnée contenant des mouvements interdits, et un critère d'aspiration peut permettre de passer outre le statut tabou d'un mouvement si cela s'avère prometteur. La stratégie est appliquée dans un cadre de séparation et évaluation pour optimiser les solutions de CVRP, montrant des résultats prometteurs sur des instances difficiles.

b. Implémentation d'une recherche tabu sur notre problème

Pour résoudre le problème de CVRP avec la méthode de recherche tabou, nous avons utilisé ces différentes étapes :

Initialisation : On génére une solution initiale de manière aléatoire, ce qui nous donne un point de départ pour l'exploration de l'espace des solutions. On améliorera ensuite cette solution de manière itérative.

Construction du voisinage : On explore le voisinage grâce à des opérations de permutation comme l'échange ou le réarrangement des différents clients dans ou entre les différentes tournées. En ajustant même légèrement la solution actuelle, on essaie de découvrir de nouvelles solutions potentielles.

Liste tabou : Pour éviter les cycles répétitifs et donner une direction à l'exploration, on utilise une liste tabou. Dans cette liste nous stockons pendant une durée paramétrable les mouvements récemment effectués pour empêcher leur répétition immédiate, ça force la recherche à explorer de nouvelles régions de l'espace des solutions.

Évaluation et sélection : À chaque itération, on évalue toutes les solutions voisines possibles et on sélectionne celle qui a le coût le plus bas.

5) Analyse des résultats

a. Comparaison des performances obtenues

Le tableau suivant indique les performances de nos méthodes selon 5 instances, chaque cellule contient le score obtenu, le temps de calcul nécessaire et le pourcentage d'erreur par rapport à la solution exacte :

Méthode	1 (n=10)	2 (n=15)	15 (n=20)	16 (n=50)	17 (n=100)
Tabou	700.4 / 0.04s / 0%	1109.36 / 0.18s / 25.4%	1071.62 / 0.45s / 37.1%	540 / 2.8s / 80%	1080 / 12.5s / 58.8%
Heur.	700.4 / 0.08s / 0%	884.81 / 0.01s / 0%	781.36 / 23.56s / 0%	300 / 61.01s / 0%	780 / 61.02s / 14.7%
Exact	700.4 / 0.002s / 0%	884.81 / 0.065s / 0%	781.36 / 5.62s / 0%	-	-
CPLEX	700.4 / 1.39s / 0%	884.81 / 284.77s / 0%	-	-	-

TABLE 5 – Résultats comparatifs des méthodes sur datasets (Un coeur sur CPU)

b. Pistes d'amélioration

Pour améliorer notre algorithme de recherche tabou,nous pouvons ajuster dynamiquement la taille de la liste tabou pour une meilleure flexibilité, pour l'ajout ou la suppression des solutions.L'implémentation d'une phase d'intensification ciblée, où l'algorithme se concentre sur l'exploration approfondie des régions prometteuses identifiées, pourrait accélérer la convergence vers des solutions de qualité supérieure. Enfin, l'adoption d'une approche hybride, combinant la recherche tabou avec d'autres méthodes d'optimisation, telles que les algorithmes génétiques ou le recuit simulé, pourrait offrir une synergie puissante, exploitant les forces de chaque méthode pour surmonter leurs limitations respectives et améliorer globalement la performance de recherche.

Références

— A Dynamic Programming Approach to Sequencing Problems, Michael Held and Richard M. Karp, https://doi.org/10.1137/0110015

— Recherche Tabou : http://www2.ift.ulaval.ca/~dupuis/Optimisation%20lineaire%20et%20applications/Divers%20Optimisation/Recherche_Tabou.pdf