Modélisation du Problème de Routage de Véhicules Électriques avec Multi-Technologies

Ouassim Behlil

September 23, 2024

0.1 Introduction

Soit $G = (N \cup R, E)$ un graphe de sommets représentant la réunion de $\mathbf{N}$, l'ensemble de clients, et $\mathbf{R}$, les stations de recharge. Un cas particulier de $\mathbf{R}$ est le dépôt numéroté 0.

Tous les sommets de clients $\mathbf{N}$ doivent être visités une seule fois par une seule voiture. Diviser la demande d'un client sur plusieurs voitures n'est pas autorisé. Chaque client $i \in \mathbf{N}$ a une demande q_i.

Les stations, i.e., les sommets de $\mathbf{R}$, peuvent être visitées plusieurs fois par plusieurs voitures et simultanément.

0.2 Technologies de Recharge

On considère un ensemble $\mathbf{H}$ de technologies de recharge de batteries. Pour chaque technologie $h \in \mathbf{H}$, on a ρ_h, le taux de recharge en unité de temps, et γ_h, l'efficacité de charge de la voiture. Dans chaque visite à une station de recharge, la voiture peut être rechargée par une seule technologie disponible dans la station. On dénote par $\mathbf{H}_j$ l'ensemble des technologies disponibles dans la station $j \in \mathbf{R}$.

Tous les sommets $\mathbf{i} \in \mathbf{R} \cup \mathbf{N}$ sont caractérisés par un temps de service s_i. Dans le cas de clients, s_i est le temps de service pour livrer la demande du client i, et dans le cas des stations, s_i est le temps de service pour recharger la voiture.

0.3 Caractéristiques du Problème

Les coefficients non négatifs t_a et e_a associés à chaque arc $a \in \mathbf{E}$ représentent respectivement le temps de trajet et la consommation d'énergie pour parcourir l'arc a dans les deux sens.

Nous considérons une flotte construite par un ensemble $\mathbf{K}$ de k identiques voitures électriques. Chaque voiture a une capacité de batterie $\mathbf{B}$ et une capacité $\mathbf{Q}$. La durée de chaque route est limitée par une durée maximale T qui est la durée de travail d'un chauffeur.

On considère que chaque voiture commence son itinéraire à partir du dépôt 0 avec un niveau de batterie $b_k = B$.

0.4 Contraintes d'un Itinéraire Réalisable

Un itinéraire réalisable est un cercle fermé respectant les contraintes suivantes :

- La route doit inclure le dépôt.
- La somme des demandes des clients visités par une voiture ne doit pas dépasser la capacité de la voiture.

- La durée totale de l'itinéraire ne doit pas dépasser la durée maximale T, incluant :
  - La durée de trajet totale : somme des termes t_a pour chaque arc a parcouru.
  - La durée de service totale : somme des termes s_i pour chaque sommet i visité.
  - Le temps de recharge total dans les stations de recharge.
- Le niveau de batterie de chaque voiture doit être entre 0 et la capacité de la batterie B à chaque instant, en considérant que :
  - L'énergie consommée le long de l'itinéraire est égale à la somme des termes e_a pour chaque arc a parcouru.
  - La quantité d'énergie rechargée dans chaque station de recharge est égale à B.

Un ensemble de routes réalisables est une solution faisable si tous les clients sont visités une fois et qu'aucun plus de K véhicules n'est utilisé.

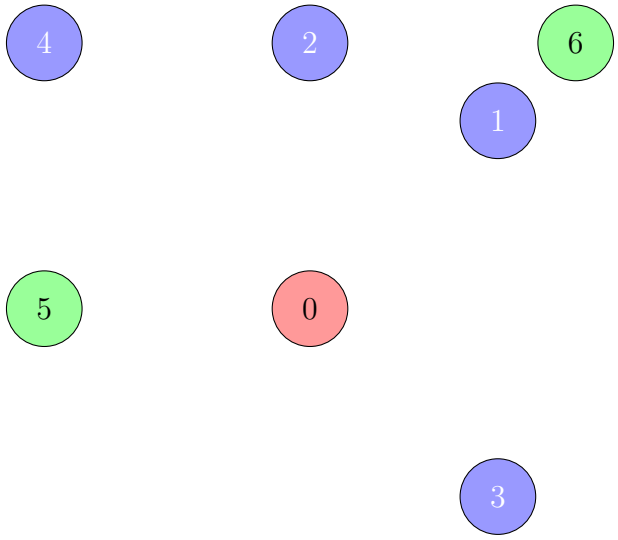

Figure 1: Exemple du problème de routage de véhicules électriques

0.5 Modélisation Mathématique

0.5.1 Variables de Décision

- $x_{ik} = 1$ si le client i est visité par la voiture k, 0 sinon.
- $y_{ijk} = 1$ si l'arc ij est parcouru par la voiture k, 0 sinon.
- b_{ik} le niveau de batterie de la voiture k après avoir visité le client i.

- t_{ij} le temps de trajet entre les sommets i et j.
- h_{ik} la technologie de recharge utilisée par la voiture k dans la station i.
- e_{ijk} l'énergie consommée par la voiture k pour parcourir l'arc ij.
- q_i la demande du client i.
- s_i le temps de service pour le client i.

0.5.2 Contraintes

$$\sum_{k \in \mathbf{K}} x_{ik} = 1, \forall i \in \mathbf{N} \quad \text{(chaque client est visité une fois)} \tag{1}$$

$$\sum_{i \in \mathbf{N}} x_{ik} \times q_i \leq Q, \forall k \in \mathbf{K} \quad \text{(capacité du véhicule)} \tag{2}$$

$$\sum_{i \in \mathbf{N} \cup \mathbf{R}} x_{ik} \times s_i \leq T, \forall k \in \mathbf{K} \quad \text{(durée maximale de service)} \tag{3}$$

$$\sum_{i,j \in \mathbf{N} \cup \mathbf{R}} e_{ijk} \times y_{ijk} \leq B, \forall k \in \mathbf{K} \quad \text{(limite de consommation d'énergie)} \tag{4}$$

$$y_{0ik} = 1, \quad \forall i \in \mathbf{N} \cup \mathbf{R}, \forall k \in \mathbf{K} \quad \text{(départ du dépôt)} \tag{5}$$

$$\sum_{i \in \mathbf{N} \cup \mathbf{R}} y_{i0k} = 1, \quad \forall k \in \mathbf{K} \quad \text{(retour au dépôt)} \tag{6}$$

0.5.3 Fonction Objectif

$$\text{Min } Z = \sum_{k \in \mathbf{K}} \sum_{i,j \in \mathbf{N} \cup \mathbf{R}} e_{ijk} \times y_{ijk} \quad \text{(minimiser la consommation d'énergie totale)} \tag{7}$$

0.6 Méthodes de Résolution

0.6.1 Méthode Exacte

Branch and Bound

La méthode branch and bound est une technique d'optimisation utilisée pour résoudre des problèmes de programmation linéaire en nombres entiers (PLNE) et d'autres problèmes d'optimisation combinatoire.

- **Branching**: Diviser le problème en sous-problèmes plus petits. Chaque sous-problème est une version restreinte du problème principal.
- **Bounding**: Évaluer les bornes inférieures et supérieures des sous-problèmes. Si la borne inférieure d'un sous-problème est supérieure à la borne supérieure du problème principal, on peut éliminer le sous-problème.

L'algorithme s'écrit comme suit :

Algorithm 1 Branch and Bound

```
1: procedure BranchAndBound
2:     Initialize the best solution found as best_sol ← ∞ (for minimization) or −∞ (for maximization)
3:     Initialize the list of nodes to explore, Q ← {initial node}
4:     while Q is not empty do
5:         Select a node N from Q
6:         Solve the relaxed problem associated with N to get a solution sol_N and bound b_N
7:         if sol_N is better than best_sol then
8:             if sol_N is integer feasible then
9:                 Update best_sol ← sol_N
10:            else
11:                Create two child nodes N_1 and N_2 by branching on a variable
12:                Add N_1 and N_2 to Q
13:            end if
14:        end if
15:        if b_N ≥ best_sol then
16:            Eliminate node N from Q (prune)
17:        end if
18:    end while
19:    Return best_sol
20: end procedure
```

Exemple:

Nous allons résoudre un problème classique de sac à dos. Nous avons un sac à dos avec une capacité de 10 unités et les objets suivants :

Objet	Poids	Valeur
1	2	10
2	3	15
3	5	25
4	7	40

Pour commencer, nous initialisons la capacité du sac à dos à 10 unités, la liste des nœuds à explorer, $Q = \{\text{nœud initial}\}$, et la meilleure solution trouvée, best_sol $= 0$.

Le nœud initial représente le fait de ne prendre aucun objet, ce qui donne une solution relaxée de valeur totale 0 et de poids total 0, avec une borne de 0. Nous sélectionnons ce nœud N et résolvons le problème associé.

Nous considérons ensuite deux cas : prendre ou ne pas prendre l'objet 1. Pour illustrer ce **branching**, nous utilisons TikZ.

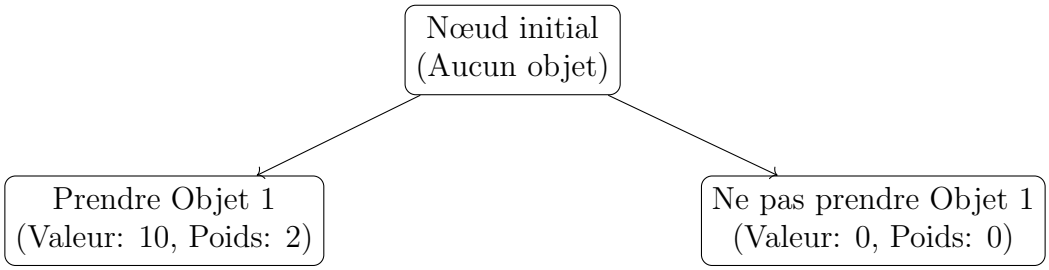

Figure 2: Branching sur l'objet 1

Si nous prenons l'objet 1, la valeur est de 10, le poids est de 2, et la borne calculée est de 10. Si nous ne prenons pas l'objet 1, la valeur est 0, le poids est 0, et la borne avec les objets restants est de 15.

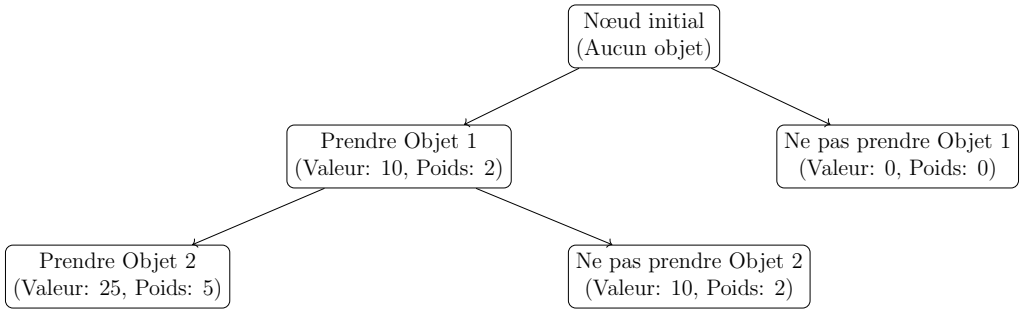

Figure 3: Branching sur l'objet 2

Nous mettons à jour notre meilleure solution trouvée, en définissant best_sol $= 10$ pour le nœud 1. Nous ajoutons alors les nœuds N_1 et N_2 à notre liste Q.

Nous continuons à évaluer les nœuds suivants. Si la borne d'un nœud est inférieure à best_sol, nous éliminons ce nœud de Q (pruning). Nous explorons davantage et découvrons, par exemple, un nœud N_3 qui représente la prise des objets 2 et 3, avec une valeur totale de 40, un poids total de 8 et une borne de 40. Nous mettons alors à jour best_sol $= 40$.

Après avoir exploré tous les nœuds, nous retournons best_sol, qui est 40. La meilleure solution trouvée consiste à prendre les objets 2 et 3, ce qui donne une valeur totale de 40 pour un poids total de 8, ce qui est optimal pour une capacité de sac de 10.

En conclusion, le processus de Branch and Bound permet d'explorer efficacement les combinaisons possibles tout en évitant d'explorer des solutions non prometteuses, réduisant ainsi le temps de calcul pour des problèmes d'optimisation comme le problème du sac à dos.

Branch and Cut

Branch and Price