Rapport de projet – Master

Multi-Product Vehicle Routing Problem with Changeover Cost

Présenté par :

LOKOSSOU Iréné A. S. F.	10 %
TOSSOU Juste D.	10 %
HOUESSOU Méschac R.	10 %
AKABASSI Samuel J. G. O. W.	10 %
PATINVOH Mavic K. S. V.	10 %
DADO Omella I. A.	10 %

Encadré par : Dr Ratheil V. HOUNDJI

IFRI - UAC

Année académique : 2025–2026

Table des matières

Résumé

Cet rapport ...

Table des figures

Liste des tableaux

Introduction

Le Vehicle Routing Problem (VRP) est un problème central en logistique...

Chapitre 1

Présentation du problème

1.1 Contexte général

La logistique de distribution de produits pétroliers constitue un enjeu majeur pour les entreprises de transport et de distribution d'énergie. Ces opérations impliquent la livraison de plusieurs types de produits (essence, gasoil, kérosène, etc.) depuis des dépôts vers un ensemble de stations de service géographiquement dispersées, à l'aide d'une flotte de camions-citernes.

Dans ce contexte, l'optimisation des tournées de livraison est essentielle afin de réduire les coûts opérationnels tout en garantissant la satisfaction complète des demandes. Cette problématique s'inscrit dans la famille des *Vehicle Routing Problems* (VRP), qui sont largement étudiés en recherche opérationnelle. Toutefois, la présence de plusieurs produits et de contraintes spécifiques liées au transport en citerne rend le problème significativement plus complexe.

1.2 Définition du MPVRP-CC

Le *Multi-Product Vehicle Routing Problem with Changeover Cost* (MPVRP-CC) est une extension du problème classique de tournées de véhicules. Il vise à organiser la distribution efficace de plusieurs types de produits à partir de dépôts vers des stations de service, en tenant compte des coûts induits par le changement de produit transporté.

Chaque station de service exprime une demande spécifique pour chaque type de produit. Pour satisfaire ces demandes, une flotte hétérogène de camions-citernes est déployée. Chaque véhicule possède une capacité de chargement limitée et est rattaché à un garage donné, qui constitue son point de départ et de retour.

La particularité majeure du MPVRP-CC réside dans la gestion des produits : un camion ne peut transporter qu'un seul type de produit à la fois. Lorsqu'un véhicule souhaite changer de produit, une opération de nettoyage de la citerne est nécessaire, ce qui engendre un coût supplémentaire appelé *coût de changeover*.

1.3 Réseau logistique et acteurs

Le problème est défini sur un réseau logistique composé de plusieurs types de sites :

— les **garages**, qui sont les points de départ et de retour des véhicules ;
— les **dépôts**, où les véhicules chargent les produits ;
— les **stations de service**, qui représentent les clients à desservir.

Une matrice de distances est définie entre l'ensemble de ces sites, permettant d'évaluer les coûts de transport. Les dépôts sont supposés disposer de stocks suffisants pour satisfaire l'ensemble des demandes, et tous les sites sont accessibles sans contrainte temporelle.

1.4 Organisation des tournées

Les opérations de chaque camion suivent une structure rigoureuse. Une tournée complète commence et se termine obligatoirement au garage auquel le véhicule est affecté. Entre ces deux points, la tournée est constituée d'une succession de *mini-tournées*.

Une mini-tournée correspond à un cycle de livraison composé de trois étapes :

— le chargement du véhicule dans un dépôt pour un produit donné ;

— la livraison de ce produit à une ou plusieurs stations de service ;

— le retour vers un dépôt pour un éventuel rechargement ou vers le garage en fin de tournée.

Lors d'une mini-tournée, un véhicule ne peut transporter qu'un seul type de produit et ne peut pas desservir plusieurs fois une même station pour ce produit.

1.5 Gestion multi-produits et coût de changeover

Chaque véhicule est initialement configuré pour transporter un produit donné. Toutefois, il est possible de modifier ce produit lors d'un passage dans un dépôt. Cette opération nécessite un nettoyage de la citerne afin d'éviter toute contamination entre produits, ce qui entraîne un coût spécifique.

Le modèle doit donc arbitrer entre deux stratégies :

— conserver le même produit et effectuer éventuellement des détours supplémentaires ;

— changer de produit au dépôt en payant le coût de changeover.

Cette décision influence directement le coût total et la structure des tournées.

1.6 Objectifs du problème

L'objectif principal du MPVRP-CC est de déterminer l'ensemble des tournées des véhicules de manière à minimiser le coût total du système logistique. Ce coût est composé de :

— le coût de transport, proportionnel à la distance totale parcourue ;

— le coût total des changements de produit, lié aux opérations de nettoyage des citernes.

1.7 Contraintes du modèle

Toute solution admissible doit respecter un ensemble strict de contraintes opérationnelles :

— **Satisfaction de la demande** : toutes les demandes des stations, pour tous les produits, doivent être entièrement satisfaites ;

— **Capacité des véhicules** : la quantité transportée ne doit jamais dépasser la capacité maximale du camion ;

— **Contraintes de flux** : chaque véhicule doit terminer sa tournée à son garage d'origine ;

— **Unicité des livraisons** : un véhicule ne peut pas desservir plusieurs fois une même station pour un même produit au cours d'une mini-tournée.

1.8 Positionnement du travail

Le MPVRP-CC constitue un problème combinatoire complexe, particulièrement adapté à l'étude de méthodes d'optimisation exactes et hybrides. Dans ce travail, une modélisation basée sur la programmation par contraintes est proposée, et le solveur OR-Tools CP-SAT est utilisé afin de résoudre efficacement différentes catégories d'instances du problème.

Chapitre 2

Modélisation du problème

2.1 Ensembles et indices

— $\mathcal{K}$: ensemble des véhicules, indice k.

— $\mathcal{P}$: ensemble des produits, indice p.

— $\mathcal{G}$: ensemble des garages, indice g.

— $\mathcal{D}$: ensemble des dépôts, indice d.

— $\mathcal{S}$: ensemble des stations de service (clients), indice s.

— $\mathcal{V}$: ensemble des nœuds du réseau (images des garages, dépôts, stations), indices i, j.

— $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$: arcs orientés, indice (i, j).

— $\mathcal{R}$: ensemble des trajets possibles par véhicule, indice r (borne supérieure finie).

Nous notons $\phi^G(\cdot), \phi^D(\cdot), \phi^S(\cdot)$ les applications injectives des garages/dépôts/stations vers les nœuds de $\mathcal{V}$. Par hypothèse, $\phi^G(\mathcal{G}) \cap \phi^D(\mathcal{D}) = \varnothing$ et les garages sont distincts des stations.

2.2 Paramètres

— c_{ij} : coût de déplacement (ou distance) sur l'arc $(i, j) \in \mathcal{A}$.

— Q_k : capacité (volume) du véhicule k.

— $d_s^p \geq 0$: demande du produit p à la station s.

— $\alpha_{kg} \in \{0, 1\}$: vaut 1 si le véhicule k est affecté au garage g (garage d'origine), avec $\sum_g \alpha_{kg} = 1$.

— $InitProd_k \in \mathcal{P}$: produit de configuration initiale du véhicule k (résidu/état uniquement, pas une quantité transportée).

— $Stock_d^p \geq 0$: stock disponible du produit p au dépôt d.

— C_{pq} : coût de changement pour passer du produit p au produit q (donné par la matrice de l'instance). En général $C_{pp} = 0$.

2.3 Variables de décision

Routage / utilisation

— $x_{ij}^{kr} \in \{0, 1\}$: vaut 1 si le véhicule k parcourt l'arc (i, j) durant le trajet r, 0 sinon.

— $y_{kr} \in \{0, 1\}$: vaut 1 si le trajet r du véhicule k est utilisé, 0 sinon.

Produit transporté sur un arc et flux de quantité

— $y_{ij}^{kpr} \in \{0,1\}$: vaut 1 si le véhicule k, lors du trajet r, parcourt l'arc (i,j) en transportant le produit p sur cet arc.

— $f_{ij}^{kpr} \geq 0$: quantité (volume) du produit p transportée par le véhicule k sur le trajet r le long de l'arc (i,j).

Chargement / quantités livrées

— $w_d^{kpr} \geq 0$: quantité du produit p chargée au dépôt d par le véhicule k lors du trajet r.

— $q_s^{kpr} \geq 0$: quantité du produit p livrée à la station s par le véhicule k lors du trajet r.

Indicateurs de changement de produit

— $t_{d,k,r}^{p,q} \in \{0,1\}$: vaut 1 si le véhicule k, durant le trajet r, visite le dépôt d et passe du produit p (entrant) au produit q (sortant).

2.4 Objectif

Minimiser le coût total de déplacement plus le coût total de changement de produit :

$$\min \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} \sum_{(i,j)\in\mathcal{A}} c_{ij}\, x_{ij}^{kr} + \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} \sum_{d\in\mathcal{D}} \sum_{p\in\mathcal{P}} \sum_{q\in\mathcal{P}} C_{pq}\, t_{d,k,r}^{p,q}. \tag{2.1}$$

2.5 Contraintes

Affectation aux garages (paramètre)

$$\sum_{g\in\mathcal{G}} \alpha_{kg} = 1 \quad \forall k \in \mathcal{K}. \tag{2.2}$$

Utilisation des trajets et départs/retours au garage

$$\sum_{g\in\mathcal{G}} \alpha_{kg} \sum_{j\in\delta^+(\phi^G(g))} x_{\phi^G(g),j}^{kr} = y_{kr} \quad \forall k, r, \tag{2.3}$$

$$\sum_{g\in\mathcal{G}} \alpha_{kg} \sum_{i\in\delta^-(\phi^G(g))} x_{i,\phi^G(g)}^{kr} = y_{kr} \quad \forall k, r. \tag{2.4}$$

Conservation du flux de routage (arcs binaires)

$$\sum_{j\in\delta^-(i)} x_{ji}^{kr} = \sum_{j\in\delta^+(i)} x_{ij}^{kr} \quad \forall i \in \mathcal{V},\ \forall k, r. \tag{2.5}$$

Couplage arc–produit Chaque arc utilisé doit être associé à exactement un produit transporté (si l'arc est utilisé) :

$$\sum_{p\in\mathcal{P}} y_{ij}^{kpr} = x_{ij}^{kr} \quad \forall (i,j),\ k, r. \tag{2.6}$$

Lien entre l'affectation du produit et la quantité transportée sur le même arc (borne supérieure de capacité) :

$$0 \leq f_{ij}^{kpr} \leq Q_k\, y_{ij}^{kpr} \quad \forall (i,j),\ k, p, r. \tag{2.7}$$

Conservation des flux de quantité (par produit) — bilan aux nœuds Pour un nœud arbitraire $v = \phi^S(s)$ (station) : le flux entrant du produit p est égal à la quantité livrée plus le flux sortant :

$$\sum_{i \in \delta^-(\phi^S(s))} f^{kpr}_{i,\phi^S(s)} = q^{kpr}_s + \sum_{j \in \delta^+(\phi^S(s))} f^{kpr}_{\phi^S(s),j} \quad \forall s, k, p, r. \tag{2.8}$$

Pour les nœuds de dépôt $v = \phi^D(d)$: le flux sortant est égal au résidu entrant plus la quantité chargée à ce dépôt (le chargement au dépôt est autorisé) :

$$\sum_{j \in \delta^+(\phi^D(d))} f^{kpr}_{\phi^D(d),j} = \sum_{i \in \delta^-(\phi^D(d))} f^{kpr}_{i,\phi^D(d)} + w^{kpr}_d \quad \forall d, k, p, r. \tag{2.9}$$

Pour les nœuds garage $v = \phi^G(g)$, on suppose qu'il n'y a ni chargement ni livraison ; les flux entrants et sortants doivent s'équilibrer (géré par la conservation du routage) ; on peut également imposer une quantité transportée nulle au garage si nécessaire :

$$\sum_{j \in \delta^+(\phi^G(g))} f^{kpr}_{\phi^G(g),j} = \sum_{i \in \delta^-(\phi^G(g))} f^{kpr}_{i,\phi^G(g)} \quad \forall g, k, p, r. \tag{2.10}$$

Satisfaction de la demande (livraisons fractionnées autorisées) La quantité totale livrée à la station s sur l'ensemble des véhicules et trajets doit satisfaire la demande :

$$\sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} q^{kpr}_s = d^p_s \quad \forall s, p. \tag{2.11}$$

Lien entre livraisons et arrivées (évite les livraisons fantômes) Une livraison positive à la station s lors du trajet r nécessite que le véhicule arrive effectivement à s lors de ce trajet (c'est-à-dire qu'il existe un arc entrant associé à un produit) :

$$q^{kpr}_s \leq \sum_{i \in \delta^-(\phi^S(s))} f^{kpr}_{i,\phi^S(s)} \quad \forall s, k, p, r. \tag{2.12}$$

(Combinée à la non-négativité, cette contrainte garantit que les livraisons sont supportées par une quantité effectivement transportée.)

Capacité par trajet La quantité totale transportée à tout instant par un véhicule sur un trajet ne peut pas dépasser sa capacité ; une inégalité agrégée utilisant les chargements et les quantités transportées permet d'assurer le respect de la capacité. Une contrainte suffisante et sûre est :

$$\sum_{(i,j) \in \mathcal{A}} \sum_{p \in \mathcal{P}} f^{kpr}_{ij} \leq Q_k \, y_{kr} \quad \forall k, r. \tag{2.13}$$

(Cette formulation peut compter une même unité plusieurs fois le long des arcs ; une formulation alternative plus serrée peut suivre la quantité embarquée par position, mais celle-ci est plus simple et conservatrice si les flux sont cohérents.)

Limites de stock des dépôts La quantité totale chargée depuis le dépôt d pour le produit p ne peut pas dépasser le stock :

$$\sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} w^{kpr}_d \leq Stock^p_d \quad \forall d, p. \tag{2.14}$$

Lien entre les chargements et les flux sortants de produits depuis le dépôt (cohérence) :

$$\sum_{j \in \delta^+(\phi^D(d))} f^{kpr}_{\phi^D(d),j} \geq w^{kpr}_d \quad \forall d, k, p, r. \tag{2.15}$$

(Avec le bilan aux nœuds, cela impose que la quantité chargée apparaisse sur les arcs sortants.)

Continuité du produit : changement uniquement aux dépôts Sur les nœuds non dépôts (stations, garages), le produit transporté ne peut pas changer : le produit des arcs entrants doit être identique à celui des arcs sortants. Cela est imposé au nœud v en liant les indicateurs de produits entrants et sortants. Pour les nœuds station s :

$$\sum_{i \in \delta^-(\phi^S(s))} y^{kpr}_{i,\phi^S(s)} = \sum_{j \in \delta^+(\phi^S(s))} y^{kpr}_{\phi^S(s),j} \quad \forall s, k, p, r. \tag{2.16}$$

Pour les nœuds garage g, la même règle s'applique (aucune reconfiguration de produit n'est autorisée au garage) :

$$\sum_{i \in \delta^-(\phi^G(g))} y^{kpr}_{i,\phi^G(g)} = \sum_{j \in \delta^+(\phi^G(g))} y^{kpr}_{\phi^G(g),j} \quad \forall g, k, p, r. \tag{2.17}$$

Aux nœuds dépôt, le changement de produit est autorisé ; il est modélisé explicitement à l'aide des variables t ci-dessous.

Modélisation du changement de produit aux dépôts Pour chaque visite du véhicule k au trajet r au dépôt d, on détecte le produit entrant et le produit sortant du dépôt et on active l'indicateur correspondant $t^{p,q}_{d,k,r}$.

Soit $InProd^p_{d,k,r}$ une expression binaire valant 1 si le véhicule k, au trajet r, arrive au dépôt d en transportant le produit p sur un arc entrant ; de même, $OutProd^q_{d,k,r}$ vaut 1 si le véhicule quitte d sur un arc sortant en transportant le produit q. On linéarise ces expressions par les inégalités :

$$InProd^p_{d,k,r} \leq \sum_{i \in \delta^-(\phi^D(d))} y^{kpr}_{i,\phi^D(d)}, \quad InProd^p_{d,k,r} \in \{0,1\}, \tag{2.18}$$

$$OutProd^q_{d,k,r} \leq \sum_{j \in \delta^+(\phi^D(d))} y^{kqr}_{\phi^D(d),j}, \quad OutProd^q_{d,k,r} \in \{0,1\}. \tag{2.19}$$

Puis on lie les indicateurs t :

$$t^{p,q}_{d,k,r} \leq InProd^p_{d,k,r}, \tag{2.20}$$

$$t^{p,q}_{d,k,r} \leq OutProd^q_{d,k,r}, \tag{2.21}$$

$$t^{p,q}_{d,k,r} \geq InProd^p_{d,k,r} + OutProd^q_{d,k,r} - 1. \tag{2.22}$$

De plus, on impose qu'au plus un produit entrant et au plus un produit sortant soient sélectionnés pour une visite donnée (si un véhicule visite un dépôt plusieurs fois sur différents trajets, ceux-ci sont distingués par r) :

$$\sum_{p \in \mathcal{P}} InProd^p_{d,k,r} \leq y_{kr}, \tag{2.23}$$

$$\sum_{q \in \mathcal{P}} OutProd^q_{d,k,r} \leq y_{kr}. \tag{2.24}$$

(Si un véhicule ne visite pas le dépôt d au trajet r, les deux sommes valent 0.)

Gestion du produit initial Pour la première visite à un dépôt lors d'un trajet, le produit « entrant » doit être considéré comme le $InitProd_k$ du véhicule. Le modèle linéaire ci-dessus autorise $InProd$ à être nul pour tout p si le véhicule n'a pas de produit entrant (par exemple s'il démarre à vide). Pour garantir la prise en compte de l'état initial, une approche pratique consiste à définir un indicateur entrant artificiel pour le premier dépôt égal au produit initial ; dans notre

formulation, nous traitons le cas initial en autorisant un ajustement du paramètre d'instance $C_{p,q}$: si le modèle doit imposer l'absence de coût lorsque le véhicule passe de sa configuration initiale au premier produit chargé, on fixe $C_{InitProd_k,q} = 0$ pour tout q (prétraitement de l'instance). Alternativement, on peut introduire des contraintes supplémentaires forçant le $InProd$ du premier dépôt à correspondre à $InitProd_k$; nous laissons ce choix à l'implémentation et documentons les deux options.

Cohérence entre les arcs produits et $InProd/OutProd$ Pour garantir que les indicateurs binaires $InProd/OutProd$ reflètent bien les arcs entrants et sortants réels, on impose les contraintes (suffisantes) suivantes :

$$InProd^p_{d,k,r} \geq y^{kpr}_{i,\phi^D(d)} \quad \forall i \in \delta^-(\phi^D(d)), \tag{2.25}$$

$$OutProd^q_{d,k,r} \geq y^{kqr}_{\phi^D(d),j} \quad \forall j \in \delta^+(\phi^D(d)). \tag{2.26}$$

Combinées aux liaisons précédentes, ces contraintes rendent $t^{p,q}_{d,k,r}$ actif exactement lorsque le véhicule arrive avec p et repart avec q.

Comptabilisation des changements et cohérence de l'objectif Les variables $t^{p,q}_{d,k,r}$ représentent les transitions $p \to q$ lors des visites de dépôts ; l'objectif somme les termes $C_{pq}t^{p,q}_{d,k,r}$. Si le prétraitement de l'instance fixe $C_{InitProd_k,q} = 0$, on obtient le comportement souhaité où la configuration initiale n'entraîne pas de coût de changement pour le premier chargement lorsque cela est requis.

Bris de symétrie sur l'utilisation des trajets (optionnel)

$$y_{k,r+1} \leq y_{kr} \quad \forall k,\ r = 1, \ldots, |\mathcal{R}| - 1. \tag{2.27}$$

Domaines des variables

$$x^{kr}_{ij} \in \{0,1\},\ y^{kpr}_{ij} \in \{0,1\},\ t^{p,q}_{d,k,r} \in \{0,1\}, \tag{2.28}$$

$$f^{kpr}_{ij} \geq 0,\ w^{kpr}_{d} \geq 0,\ q^{kpr}_{s} \geq 0,\ y_{kr} \in \{0,1\}. \tag{2.29}$$

2.6 Remarques et notes pratiques

— Le modèle privilégie la clarté à la compacité : les variables binaires produit-sur-arc y^{kpr}_{ij} et les variables de flux f^{kpr}_{ij} facilitent l'imposition du fait que les changements de produit n'ont lieu qu'aux dépôts et le calcul des coûts de changement. Elles augmentent le nombre de variables mais rendent la logique de changement explicite.

— Les variables $t^{p,q}_{d,k,r}$ linéarisent la détection d'une transition de produit lors d'une visite de dépôt. Une alternative consiste à indexer les visites de dépôt (ordonnées) et à utiliser des liaisons entre visites consécutives ; cette approche est souvent plus serrée mais plus complexe.

— Cas particulier du produit initial : si la logique métier impose que le premier chargement depuis la configuration initiale d'un véhicule soit toujours gratuit, il suffit de fixer les $C_{InitProd_k,q} = 0$ correspondants dans l'instance avant résolution.

— La contrainte de capacité donnée est conservatrice ; une contrainte de capacité basée sur les flux, suivant la quantité embarquée le long des arcs, peut être introduite si nécessaire pour améliorer les performances du solveur.

Chapitre 3

Résolution

3.1 Présentation du solveur utilisé

3.1.1 OR-Tools CP-SAT : Un solveur de pointe

OR-Tools (Google Optimization Tools) est une suite open-source développée par Google pour résoudre des problèmes d'optimisation complexes.

Caractéristiques techniques :

— Type : Solveur par programmation par contraintes (CP-SAT)
— Algorithme : Recherche arborescente avec propagation de contraintes
— Parallélisation : Support multi-threads (4 workers)
— Heuristiques : Incorporation de solutions initiales (Hints)
— Garanties : Trouve des solutions optimales prouvées

3.2 Architecture du code

3.2.1 Structure modulaire en 5 parties principales

Chargement des données (`load_instance`)

La première étape consiste à **charger l'instance du problème** depuis un fichier `.dat`.
Les données extraites sont :

— le nombre de **produits**, **stations**, **véhicules**, **dépôts** et **garages**
— les **coûts de transition entre produits**
— les **capacités des véhicules**
— les **demandes par station et par produit**
— les **coordonnées géographiques** des entités

Analyse préalable de faisabilité

Avant toute résolution, une **analyse de faisabilité** est réalisée :

— calcul de la **demande totale**
— calcul de la **capacité totale disponible**

Cette étape permet :

— de détecter rapidement les cas **impossibles**
— d'interpréter correctement les résultats du solveur
— d'expliquer d'éventuelles situations d'infaisabilité dans le rapport

Construction d'une solution heuristique initiale

Afin d'améliorer les performances du solveur exact, une **solution heuristique initiale** est construite.

Principe de l'heuristique :

— chaque véhicule conserve son **produit initial**

— il dessert les **stations les plus proches** demandant ce produit

— il livre jusqu'à **épuisement de sa capacité**

Rôle de cette étape :

— produire une **solution faisable rapidement**

— fournir une **borne supérieure initiale**

— guider le solveur CP-SAT grâce au mécanisme de **hint**

Cette solution n'est pas optimale, mais elle est réaliste et exploitable.

Modélisation mathématique avec CP-SAT

Variables de décision

— **Variables de livraison** : quantité livrée par un véhicule à une station pour un produit donné

— **Variables binaires de visite** : indiquent si un véhicule visite une station

Ces variables permettent de modéliser à la fois :

— les quantités transportées

— la structure des tournées

Contraintes du modèle Les principales contraintes sont :

1. **Satisfaction complète de la demande** : chaque demande (station, produit) doit être entièrement satisfaite par l'ensemble des véhicules
2. **Capacité des véhicules** : la somme des quantités livrées par un véhicule ne peut pas dépasser sa capacité
3. **Lien livraison–visite** : un véhicule est considéré comme visitant une station s'il y livre une quantité strictement positive

Ces contraintes garantissent la **faisabilité logistique** de la solution.

Fonction objectif

La fonction objectif vise à **minimiser le coût total**, composé de :

1. **Coût de transport** : distance entre le garage du véhicule et les stations visitées
2. **Coût de changement de produit (changeover)** : pénalité lorsqu'un véhicule livre un produit différent de son produit initial
3. **Coût de consolidation** : coût fixe par visite afin de limiter le nombre de stations visitées et favoriser des tournées plus compactes

Intégration de la solution heuristique (Hint)

La solution heuristique construite précédemment est injectée dans le modèle sous forme de **point de départ**.

Avantages :

— accélération de la recherche

— amélioration de la qualité des premières solutions

— réduction du temps de calcul

Le solveur reste libre d'améliorer ou de modifier cette solution.

Résolution avec OR-Tools CP-SAT

Le solveur est ensuite lancé avec :

— une **limite de temps**

— une **recherche multi-threads**

— un **critère d'écart relatif** pour accepter une solution quasi-optimale

À l'issue de la résolution, le solveur retourne :

— une solution **optimale**, **faisable**, ou

— un statut d'**infaisabilité** clairement identifié

Analyse et validation des résultats

La solution obtenue est analysée afin de :

— vérifier que **toutes les demandes sont satisfaites**

— mesurer le **taux d'utilisation des véhicules**

— évaluer le **coût total**

— comparer la solution finale à la solution heuristique initiale

Cette étape permet de justifier la qualité de la solution.

Visualisation des résultats

Une visualisation graphique est générée pour :

— représenter les **garages, dépôts et stations**

— afficher les **routes par véhicule**

— illustrer les **quantités livrées**

Sauvegarde et synthèse finale

Enfin, la solution est :

— sauvegardée dans un fichier de sortie

— accompagnée d'un **résumé chiffré** :

  — coût total

  — nombre de routes

  — taux de satisfaction de la demande

  — taux d'utilisation des capacités

3.3 Analyse des résultats par catégorie d'instances

3.3.1 Instance *small*

Ici le texte est effectué sur de petite instance (small instance). La commande de test est :

```
python solver.py instances/small.dat solution.dat
```

Présentation des résultats de l'instance

Description de l'instance L'instance catégorie *small* se caractérise par une taille réduite, permettant de valider le bon fonctionnement du modèle et du solveur.

- Nombre de stations : 5
- Nombre de produits : 2
- Nombre de véhicules : 2
- Nombre de dépôts : 1
- Nombre de garages : 1

Cette instance représente un cas simple mais représentatif, dans lequel chaque station peut demander un ou plusieurs produits.

Analyse de faisabilité a priori Avant la résolution, une analyse de faisabilité est effectuée afin de vérifier l'adéquation entre la demande globale et la capacité totale disponible.

- Demande totale : 6 900 unités
- Capacité totale des véhicules : 9 000 unités

La capacité totale étant supérieure à la demande, l'instance est théoriquement faisable.

- Marge de capacité : 2 100 unités
- Ratio demande / capacité : 0,77

Résolution avec OR-Tools CP-SAT La résolution est effectuée à l'aide du solveur OR-Tools CP-SAT, basé sur la programmation par contraintes et la recherche SAT.

- **Variables de décision** : 2 véhicules × 5 stations × 2 produits = 20 variables principales
- **Contraintes** :
  - 7 contraintes de satisfaction de la demande
  - 10 contraintes liant livraison et visite
  - 2 contraintes de capacité des véhicules
- **Fonction objectif** : minimisation du coût total (distance + coûts de changement de produit), composée de 30 termes

La phase de modélisation est réalisée en 0,003 seconde, ce qui montre une construction efficace du modèle.

Solution heuristique initiale Une solution heuristique est construite afin de fournir un point de départ au solveur CP-SAT.

- **Véhicule 1 (Produit initial 1, capacité 5000)** : dessert 3 stations, capacité utilisée : 2 500 / 5 000
- **Véhicule 2 (Produit initial 2, capacité 4000)** : dessert 4 stations, capacité utilisée : 4 000 / 4 000

Le coût estimé de cette solution initiale est de **18 937**, et **7 hints** sont fournis au solveur.

Résultats de la résolution Le solveur CP-SAT atteint une solution **OPTIMALE**.

— Statut : OPTIMAL
— Valeur de l'objectif : 13 418
— Temps de résolution : 0,058 s
— Temps total (modélisation + résolution) : 0,061 s
— Amélioration par rapport à la solution heuristique initiale : 5 519 unités soit 29,1 %

Toutes les demandes sont intégralement satisfaites.

3.3.2 Analyse argumentée des résultats de l'instance

Qualité de la solution obtenue

La solution trouvée est optimale et respecte l'ensemble des contraintes du problème. Le solveur parvient à réduire significativement le coût total par rapport à la solution heuristique initiale, ce qui démontre l'efficacité de l'exploration de l'espace des solutions par CP-SAT.

Utilisation des capacités des véhicules

Les capacités des véhicules sont exploitées de manière équilibrée :

— Véhicule 1 : 64 % de capacité utilisée
— Véhicule 2 : 92,5 % de capacité utilisée

Cette répartition permet de satisfaire l'ensemble des demandes tout en limitant les surcharges inutiles.

3.4 Instance *medium*

L'instance medium du problème MPVRP-CC se caractérise par une taille intermédiaire, permettant d'évaluer à la fois la qualité de la modélisation et les performances du solveur. Pour la tester, on utilise la commande :

```
python solver.py instances/medium.dat  solution.dat
```

3.4.1 Caractéristiques principales

— Stations : 12
— Produits : 3
— Véhicules : 4
— Dépôt : 1
— Garages : 2
— Demande totale : 19 100 unités
— Capacité totale des véhicules : 21 000 unités
— Ratio demande/capacité : 0,91

Cette configuration garantit la faisabilité théorique du problème tout en conservant une complexité suffisante pour tester l'efficacité du solveur.

3.4.2 Résolution avec OR-Tools CP-SAT

— Solveur utilisé : OR-Tools CP-SAT
— Temps limite : 30 s
— Temps effectif de résolution : 5,85 s
— Statut final : OPTIMAL
— Valeur fonction objectif : 27 396

Une solution heuristique initiale a été générée avant la résolution exacte :

— Coût heuristique initial : 103 951
— La solution optimale finale représente une amélioration de 73,6 %

3.4.3 Résultats opérationnels

— 100 % des demandes des stations sont satisfaites
— 4 routes générées, correspondant aux 4 véhicules
— Taux d'utilisation des capacités :
  — Véhicule 1 : 98,3 %
  — Véhicule 2 : 85,5 %
  — Véhicule 3 : 100 %
  — Véhicule 4 : 77,8 %

Une visualisation graphique des routes a été produite afin de faciliter l'interprétation spatiale de la solution.

3.4.4 Analyses argumentées des résultats obtenus

Qualité de la solution

La solution obtenue est **optimale**, ce qui démontre que la modélisation CP-SAT est **correcte et complète**. Toutes les contraintes (demande, capacité, compatibilité produits/véhicules) sont respectées sans exception. L'amélioration significative par rapport à la solution heuristique initiale montre que :

— l'heuristique joue efficacement son rôle de **point de départ**
— le solveur CP-SAT exploite cette base pour explorer rapidement des solutions de meilleure qualité

Performance du solveur OR-Tools CP-SAT

Le solveur atteint l'optimalité en moins de 6 secondes, bien en-dessous de la limite imposée (30 s). Cela met en évidence :

— la robustesse du moteur CP-SAT
— l'efficacité des stratégies internes (LNS, LP relaxation, local search, feasibility pump)

La diversité des sous-solveurs activés a permis une exploration rapide et équilibrée de l'espace de recherche.

Répartition des charges et routes

La charge est globalement bien répartie :

— Aucun véhicule n'est sous-utilisé de manière critique

— Un véhicule atteint même 100 % de capacité, indiquant une bonne exploitation des ressources

Les routes restent relativement courtes (2 à 4 stations par véhicule), ce qui contribue à :

— réduire les coûts

— simplifier l'exécution opérationnelle

3.5 Instance *large*

L'instance large du problème MPVRP-CC correspond à un cas de grande taille, destiné à évaluer la robustesse de la modélisation ainsi que les performances du solveur OR-Tools CP-SAT dans un contexte fortement combinatoire.

Pour le tester, on utilise la commande suivante :

```
python solver.py instances/large.dat solution.dat
```

3.5.1 Caractéristiques principales

— Stations : 20

— Produits : 2

— Véhicules : 6

— Dépôts : 2

— Garages : 3

— Demande totale : 24 210 unités

— Capacité totale des véhicules : 40 500 unités

— Ratio demande/capacité : 0,60

Cette configuration garantit une faisabilité confortable du problème, avec une marge de capacité importante, tout en introduisant une complexité élevée due au nombre de stations, de véhicules et aux livraisons multi-produits.

3.5.2 Résolution avec OR-Tools CP-SAT

— Solveur : OR-Tools CP-SAT

— Temps limite : 30 s

— Temps effectif de résolution : 2,83 s

— Temps total (modélisation + résolution) : 2,85 s

— Statut final : OPTIMAL

— Valeur de la fonction objectif : 45 772

Avant la résolution exacte, une solution heuristique initiale a été construite et injectée dans le solveur et son coût heuristique initial était : 123 836. La solution optimale finale représente une amélioration de 63,0 % par rapport à cette solution de départ.

3.5.3 Résultats opérationnels

— 100 % des demandes des stations sont satisfaites, pour l'ensemble des produits.

— 5 routes générées, correspondant à 5 véhicules effectivement utilisés sur les 6 disponibles.

— Taux d'utilisation des capacités :

  — Véhicule 1 : 66,9 %
  — Véhicule 2 : 38,6 %
  — Véhicule 3 : 92,9 %
  — Véhicule 5 : 99,8 %
  — Véhicule 6 : 58,2 %

Les routes desservent entre 1 et 6 stations, avec des livraisons mono-produit et multi-produits selon les stations. Une visualisation graphique des routes a été générée afin de faciliter l'analyse spatiale et la validation visuelle de la solution (*visualisation_large.png*).

3.5.4 Analyses argumentées des résultats obtenus

Qualité de la solution

La solution obtenue est optimale, ce qui confirme la validité et la complétude de la modélisation CP-SAT du MPVRP-CC. Toutes les contraintes sont strictement respectées :

— satisfaction exacte des demandes,

— respect des capacités des véhicules,

— cohérence entre visite d'une station et livraison effective,

— prise en compte des coûts de changement de produit

L'amélioration significative par rapport à la solution heuristique initiale montre que :

— l'heuristique joue efficacement son rôle de point de départ faisable,

— le solveur CP-SAT exploite cette base pour réduire rapidement l'espace de recherche et atteindre une solution optimale.

Performance du solveur OR-Tools CP-SAT

Le solveur atteint l'optimalité en moins de 3 secondes, largement en dessous de la limite imposée de 30 secondes. Cette performance met en évidence :

— la robustesse du moteur CP-SAT face à une instance de grande taille,

— l'efficacité des stratégies internes activées automatiquement, notamment :

  — Large Neighborhood Search (LNS),
  — relaxations linéaires (default_lp),
  — feasibility pump,
  — recherche locale et redémarrages contrôlés.

La diversité des sous-solveurs a permis une exploration rapide, équilibrée et progressive de l'espace de recherche, comme l'illustre la succession de 37 solutions améliorantes avant l'optimalité.

Répartition des charges et routes

La répartition des charges n'est pas parfaitement équilibrée, mais reste cohérente d'un point de vue économique :

— certains véhicules sont fortement exploités (jusqu'à **99,8 %**),

— d'autres sont moins sollicités lorsque cela permet de réduire le coût global.

Un véhicule n'est pas utilisé, ce qui indique que le solveur préfère minimiser les coûts plutôt que mobiliser inutilement toute la flotte disponible. Les routes restent de taille raisonnable (jusqu'à 6 stations par véhicule), ce qui contribue à :

— limiter les distances et coûts cumulés,

— conserver une exécution opérationnelle réaliste,

— gérer efficacement les livraisons multi-produits.

Conclusion et perspectives

Pour conclure ...

Bibliographie