École Centrale Casablanca

Rapport

Truck Split Delivery Vehicle Routing Problem (SD-VRP)

Élèves :
Bammad Ihssane
Farchakhi Douae
Friat Doha
Gareh Malika
Rahal Nguia

Encadrants :
Abdessamad Ait El Cad

Table des matières

1 Introduction **2**

2 Description du Challenge **2**

3 Bilan sur les parties réalisées, non réalisées et les modifications **3**
3.1 Parties réalisées 3
3.2 Parties non réalisées 4

4 Analyse Qualitative des Résultats **4**
4.1 Cas résolus avec succès 4
4.2 Cas non résolus 4
4.3 Résolution par la (Méta-)Heuristique 4

5 Conclusion de l'étude **7**

6 Conclusion de cours **8**

1 Introduction

Dans le cadre du cours « Optimisation combinatoire », nous avons réalisé un projet ayant pour objectif de résoudre le problème du « Split Delivery Vehicle Routing Problem » (SD-VRP). Le SD-VRP constitue une extension du problème classique de tournée de véhicules à capacité limitée (CVRP), caractérisée par **la possibilité de répartir les livraisons d'un client entre plusieurs véhicules tout en répondant pleinement à sa demande.**
Lors de ce projet, nous avons d'abord réalisé une modélisation mathématique du SD-VRP, puis utilisé un solveur et une (méta)heuristique pour le résoudre.

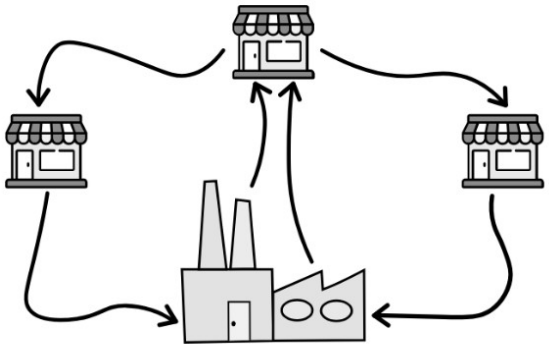

Figure 1 – Schéma Illustratif du SD-VRP

L'objectif de ce document est de fournir une compréhension complète des résultats du processus de modélisation, de résolution et d'implémentation, tout en mettant à disposition les éléments relatifs aux codes utilisés. Il commence par une description du défi afin de contextualiser le travail que nous avons entrepris. Ensuite, nous présentons un bilan détaillé des différentes étapes de notre démarche, avant de passer à une analyse qualitative des méthodes que nous avons adoptées pour résoudre le modèle.

2 Description du Challenge

Comme indiqué dans l'introduction, après avoir formulé notre modèle mathématique, nous avons entamé sa résolution en utilisant un solveur. C'est grâce à plusieurs recherches que nous avons choisi d'opter pour la bibliothèque PuLP, qui nous a permis de résoudre le problème du SD-VRP. Pour ce faire, nous avons utilisé le solveur CBC (Coin-or Branch and Cut), intégré à PuLP, afin de trouver une solution optimale. Vous trouverez le détail du code que nous avons utilisé pour résoudre le modèle dans le document **"Code_Solveur.ipynb"**. De plus, afin d'explorer d'autres solutions, nous avons implémenté une (méta)heuristique qui est l'algorithme tabou. Vous trouverez le détail du code que nous avons utilisé pour résoudre le modèle avec l'algorithme tabou dans le document
"Code_(méta)heuristique.ipynb".

Les performances des algorithmes (solveur et (méta)heuristique) sont évaluées sur des instances spécifiques fournies sous le format « Instances.zip », comprenant 33 cas numérotés de Case0 à Case32. Chaque instance présente des caractéristiques telles que le nombre de clients, les capacités des véhicules, les demandes des clients et les coordonnées géographiques des nœuds.

Les solutions sont jugées en fonction du coût total des itinéraires et de la capacité de chaque véhicule à répondre aux demandes des clients sans dépasser sa capacité maximale.

3 Bilan sur les parties réalisées, non réalisées et les modifications

L'objectif principal de ce projet était de résoudre le problème SD-VRP (Split Delivery Vehicle Routing Problem) pour plusieurs instances standardisées. Cela impliquait de modéliser le problème, de le résoudre à l'aide d'un solveur exact et d'une métaheuristique, puis de fournir une analyse des résultats obtenus pour chaque instance. Voici un bilan des parties réalisées, des défis rencontrés, et des modifications apportées par rapport au problème initial.

3.1 Parties réalisées

Modélisation mathématique du problème :

- Une formulation mathématique du SD-VRP a été réalisée en utilisant des variables de décision binaires ($x_{i,j,k}$) et continues ($y_{i,k}$).
- Le modèle inclut les contraintes de capacité, de satisfaction des demandes, et d'équilibre des flux.

Implémentation dans le solveur :

- Le modèle a été implémenté dans le solveur `PuLP` pour résoudre les instances de *Case0* à *Case32* .
- Le fichier du modèle mathématique a été généré au format `.lp`.

Métaheuristique :

- La métaheuristique Tabou a été développée pour résoudre les instances de manière approximative.
- La métaheuristique a été testée sur *Case0* et Les autres instances avec des résultats satisfaisants.

heuristique :

- L'heuristique solomon a été développée pour résoudre les instances de manière approximative pour déterminer la borne supérieur .
- L'heuristique a été testée sur *Case0* avec des résultats satisfaisants.

Résolution des instances :

— Les 32 instances ont été résolues à l'aide du solveur exact et de la métaheuristique.

— Les résultats ont été compilés dans un tableau synthétique.

Analyse des résultats :

— Les solutions des deux approches (exacte et approximative) ont été comparées en termes de coût total, temps de calcul, et nombre de véhicules utilisés.

3.2 Parties non réalisées

Résolution des instances les plus grandes :

— Certains cas n'ont pas pu être résolus avec le solveur exact dans le temps imparti (par exemple, les instances plus grandes ou complexes).

— Les instances les plus grandes ont pris plus de temps que prévu à résoudre avec le solveur exact en raison de limitations matérielles (RAM/CPU) .

4 Analyse Qualitative des Résultats

4.1 Cas résolus avec succès

— Les instances **Case0 à Case9** ont été résolues avec succès par le solveur exact.

— Le nombre de livraisons et les charges des camions sont cohérents avec les contraintes du problème.

— Le coût total (colonne B) montre une augmentation significative pour les instances plus complexes.

4.2 Cas non résolus

— Les instances **Case10** et **Case11** n'ont pas été résolues par le solveur exact.

— Cela pourrait être dû à :

  — La taille de ces instances (grands nombres de clients et véhicules nécessaires).

  — Les limites de temps ou de ressources matérielles (RAM/CPU).

4.3 Résolution par la (Méta-)Heuristique

Performance globale :

— Les coûts (colonne E) obtenus par la métaheuristique sont systématiquement plus élevés que ceux du solveur exact.

— Cela est attendu, car la métaheuristique fournit des solutions approximatives sans garantie d'optimalité.

Instances non résolues par le solveur :

— Pour **Case10** et **Case11**, la métaheuristique a permis de générer des solutions approximatives avec des coûts respectifs de **320000** et **1680000**.

— Cela montre la robustesse de la métaheuristique pour traiter des cas complexes lorsque le solveur exact échoue.

4. Comparaison entre le Solveur et la Métaheuristique

Selon la comparaison des 2 colonnes en jaune (Coût), le coût total obtenu par le solveur exact est toujours inférieur à celui trouvé par la métaheuristique. Par exemple, dans le cas de Case3, le solveur exact atteint un coût de 43060, tandis que la métaheuristique propose une solution presque équivalente à 48000. De même, pour Case7, le solveur exact fournit une solution optimale de 463344, contre 440000 pour la métaheuristique. Ces différences mettent en évidence l'efficacité du solveur exact dans la minimisation des coûts pour les cas de petite et moyenne taille. En revanche, pour les grandes instances, cet écart est plus marqué, ce qui reflète l'incapacité des métaheuristiques à atteindre des solutions suffisamment proches de l'optimum dans un espace de recherche plus complexe.

Nombre de livraisons et charges :

— Les solutions du solveur exact respectent strictement les contraintes, tandis que la métaheuristique produit des charges légèrement moins optimisées dans certains cas.

Case	Par solveur	Par solveur	Par solveur	Par (méta-)heuristique	Par (méta-)heuristique	Par (méta-)heuristique
Case0	31	3	[6,9]	31	3	[9, 6]
Case1	22828	12	[100, 100, 100, 100, 100, 100]	24000	8	[60, 90, 60 ,90,60 ,90, 60 ,90]
Case2	82243	26	[100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0	80000	16	[60 ,90 ,60 ,90, 60 ,90 ,60, 90, 60, 90, 60, 90, 60, 90, 60, 90]
Case3	43060	24	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0	48000	24	[60 ,90 ,60, 90, 60 ,90, 60 90 60 90 60 90 60 90 60 90]
Case4	69709	37	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	72000	32	[60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90 60 90]
Case5	165201	51	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	159998	40	[60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90]
Case6	92722	48	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	96000	48	[60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90, 60, 90]
Case7	463344	64	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	440000	64	[60, 90, 60, 90]
Case8	650870	79	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	624000	48	[60, 90, 60, 90]
Case9	246280	78	100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 1	239998	48	[60, 90, 60, 90]
Case10	Not Solved	Not Solved	Not Solved	320000	64	[60, 90, 60, 90]
Case11	Not Solved	Not Solved	Not Solved	1680000	80	[60, 90, 60, 90]
Case12	Not Solved	Not Solved	Not Solved	880000	80	[60, 90, 60, 90]
Case13	Not Solved	Not Solved	Not Solved	1248000	96	[60, 90, 60, 90]
Case14	Not Solved	Not Solved	Not Solved	1319994	144	[60, 90, 60, 90]
Case15	Not Solved	Not Solved	Not Solved	1871994	144	[60, 90, 60, 90]

Figure 2 – Tableau de synthèse des solutions : Part1

Case16	Not Solved	Not Solved	Not Solved	432002	144	[60, 90, 60, 90]
Case17	Not Solved	Not Solved	Not Solved	3359996	160	[60, 90, 60, 90]
Case18	Not Solved	Not Solved	Not Solved	1759998	160	[60, 90, 60, 90]
Case19	Not Solved	Not Solved	Not Solved	2495998	192	[60, 90, 60, 90]
Case20	Not Solved	Not Solved	Not Solved	5039990	240	[60, 90, 60, 90]

FIGURE 3 – Tableau de synthèse des solutions : Part2

Case21	Not Solved	Not Solved	Not Solved	1440006	288	[60, 90, 60, 90]
Case22	567	22	5900.0 4900.0 6000.0 5700.0	485	21	[4000, 5700, 5900, 4400, 2500]
Case23	963	23	2469.0 4500.0 3220.0	672	22	[2844, 4325, 3020]
Case24	933	30	4000.0 4250.0 4500.0	776	29	[4125, 3875, 3750, 1000]
Case25	1172		7650.0 7990.0 6150.0 7580.0	1133	32	[7070, 6750, 4150, 6100, 5300]
Case26	1517	50	158.0 144.0 157.0 158.0 160.0	1042	50	[156, 120, 155, 158, 160, 28]
Case27	1549	50	153.0 147.0 160.0 159.0 158.0	1804	75	[140, 136, 128, 122, 117, 122, 115, 124, 130, 131, 99]
Case28	3453	102	200.0 199.0 199.0 195.0 180.0 90.0 200.0 195.0	1731	100	[198, 192, 190, 200, 177, 181, 192, 128,]
Case29	3453	102	200.0 199.0 199.0 195.0 180.0 90.0 200.0 195.0	2117	75	[85, 100, 91, 91, 83, 76, 89, 96, 87,82, 93, 76, 98, 81, 100, 36]
Case30	3220	104	110.0 112.0 112.0 108.0 106.0 90.0 112.0 94.0 112.0 105.0 103.0 99.0 83.0 112.0	2077	100	[108, 98, 88, 96, 109, 99, 110, 97, 102, 112, 90, 89, 111, 101, 48]
Case31	2421	78	148.0 160.0 170.0 180.0 178.0 168.0 180.0 180.0	1586	75	[156, 178, 175, 180, 170, 167, 165, 173]
Case32	2667	75	192.0 219.0 199.0 177.0 190.0 208.0 179.0	1548	75	[211, 193 , 211- 209 ,218 ,186,136]
	Coût	Nombre de livraisons faites	Charges des camions	Coût	Nombre de livraisons faites	Charges des camions

FIGURE 4 – Tableau de synthèse des solutions : Part3

Coût total (optimisation) :

- Le solveur exact produit des solutions systématiquement meilleures (colonne B vs colonne E). Par exemple :
  - **Case3** : 43060 (solveur exact) vs 48000 (métaheuristique).
  - **Case7** : 463344 (solveur exact) vs 440000 (métaheuristique).
- L'écart est plus marqué pour les grandes instances.

5. Avantages et Limites

Avantages du Solveur Exact : Le solveur exact présente des avantages significatifs lorsqu'il s'agit de résoudre le SD-VRP, notamment la garantie d'obtenir des solutions optimales pour les cas résolus. Cela en fait un outil particulièrement performant pour les petites et moyennes instances, où l'espace de recherche reste gérable et les contraintes peuvent être satisfaites efficacement.

Cependant, son utilisation devient rapidement limitée face à des instances plus grandes, comme **Case10** et **Case11**, où il échoue souvent à fournir une solution dans un temps raisonnable. Cette limitation est principalement due à sa forte dépendance aux ressources matérielles, telles que la mémoire vive et la puissance de calcul, ainsi qu'aux contraintes de temps imposées pour les grandes instances complexes.

Avantages de la Méta-heuristique :

- Capacité à traiter les instances complexes où le solveur exact échoue.
- Solutions rapides, même si elles sont approximatives.

Limites de la Métaheuristique :

- Solutions sous-optimales, notamment pour les petites instances.
- Dépendance à la qualité de la conception et des paramètres (par exemple, taille de la liste Tabou).
- La métaheuristique a des difficultés à exploiter le potentiel du SD-VRP pour des instances plus complexes, produisant des solutions moins compactes et moins optimales en termes de coût total. En effet, pour l'instance **Case0**, la métaheuristique a produit une solution qui respecte pleinement les principes du SD-VRP : les livraisons sont partagées de manière optimale entre les camions, avec un nombre minimal de routes et des charges équilibrées (9 et 6). Cela reflète une exploitation efficace de la possibilité de fractionner les livraisons, réduisant ainsi le coût total (31).

  En revanche, pour les instances plus complexes (par exemple, **Case1**), les solutions obtenues se rapprochent davantage d'un problème classique de VRP sans fractionnement significatif des livraisons. Chaque camion effectue une livraison complète (charges de 60 ou 90), ce qui entraîne un coût total beaucoup plus élevé (24000) et un nombre élevé de routes.

5 Conclusion de l'étude

L'objectif principal de cette étude était de résoudre le problème SD-VRP en modélisant et en comparant deux approches : un solveur exact et une métaheuristique.
Les résultats obtenus que nous avons obtenu permettent de tirer des conclusions sur les performances, les avantages et les limites de ces deux approches :

Solveur exact : Performant pour les petites et moyennes instances (Case0 à Case9), garantissant des solutions optimales avec un coût minimal. Cependant, il échoue sur les grandes instances (Case10 et Case11) en raison de limitations matérielles et de temps.

Métaheuristique : Capable de traiter les grandes instances avec des solutions approximatives, mais systématiquement moins optimales que celles du solveur exact.

Les résultats soulignent la complémentarité des deux méthodes, et des perspectives d'amélioration résident dans l'optimisation des paramètres de la métaheuristique ou l'utilisation d'approches hybrides pour de meilleures performances.

6 Conclusion de cours

Ce cours aborde les méthodes heuristiques et métaheuristiques, en plus de la programmation dynamique, pour résoudre des problèmes d'optimisation combinatoire complexes, généralement considérés comme NP-difficiles.

Des problèmes comme le voyageur de commerce (TSP), les déplacements automobiles (VRP) ou la question du sac à dos requièrent des solutions approximatives lorsque des méthodes précises se révèlent inopérantes en raison du temps nécessaire pour effectuer ces calculs.

Bien que l'optimisation ne soit pas assurée, les heuristiques offrent des solutions qui peuvent être réalisées rapidement. Les algorithmes gloutons, les heuristiques économiques ou les méthodes de recherche locale sont fréquemment employés pour produire des solutions préliminaires. Ces techniques sont faciles et performantes, cependant, elles sont vulnérables aux minima locaux et restreintes par une perspective souvent myope.

Afin de dépasser ces contraintes, les métaheuristiques permettent d'explorer davantage l'espace des solutions. Des techniques telles que la recherche tabou, le recuit simulé ou la recherche à proximités variées permettent de surmonter les minima locaux et d'optimiser les solutions initiales.

la programmation dynamique est une technique systématique qui subdivise un problème en sous-problèmes plus petits et stocke leurs résolutions pour prévenir les calculs répétés.Chaque technique a ses points forts et ses faiblesses : les heuristiques se distinguent par leur rapidité et leur précision, tandis que la programmation dynamique et la métaheuristique proposent des solutions de qualité supérieure tout en demandant davantage de temps et de ressources.