



الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

المدرسة الوطنية العليا للتكنولوجيات المتقدمة

National Higher School of Advanced Technologies

Département : Génie logistique et Transports

Mémoire de fin d'étude en vue de l'obtention du diplôme

D'INGENIEUR d'État

-Filière-

INGENIERIE DES TRANSPORTS

-Spécialité -

INGENIERIE DE LA CHAINE LOGISTIQUE ET TRANSPORTS

Département de Génie logistique et Transport

-Thème -

La gestion des transports dans la logistique inverse

Cas de Condor Logistics

Realisé par :

CHEURFI Rihab & DANNINE Khadidja

Soutenu le 28/06/2025

Les membres de Jury :

MME. SELLIDJ Wassila	Présidente
M. AGGUINI Chafik	Promoteur
M. AMICHI Hocine	Examineur

Année universitaire 2024/2025

Dédicace

À ma maman, la plus précieuse des femmes,
Tu es celle qui a porté mes douleurs sans se plaindre, qui a prié pour moi en silence,
qui a séché mes larmes sans que je parle, et m'a offert une force que je ne soupçonnais
pas. Tu es mon refuge, mon pilier, mon repère dans le chaos.
Tout ce que je suis aujourd'hui, je te le dois.
Merci, maman... du plus profond de mon âme.

À mon père,
Ta confiance en moi a toujours été une force silencieuse qui m'a poussée à aller plus loin.
Par ta présence discrète, ton soutien constant et ta patience, tu as su me transmettre la
rigueur et la détermination.
Ce mémoire t'est dédié, avec respect, gratitude et tout l'amour d'une fille fière de ton
regard.

À mes petits frères et sœurs, Mohamed Islam, Reham, Remaas,
et à mes cousins Walid et Younes, vos sourires et votre innocence ont illuminé mes journées.
Ce mémoire est pour vous, en espérant qu'il vous inspire à croire en vos rêves.

À ma tante Khaira que j'aime et que je n'oublierai jamais (que Dieu ait son âme),
Tu n'es plus là pour voir ce jour, mais ta mémoire m'accompagne dans chaque réussite.
Je t'offre ce diplôme, en hommage à ton amour et à ta présence passée.

À la famille Azzouza : Soulef, Ahlem, Zohra, Karima, Salima, Khaled, Mohamed et mon
cher grand-père, Votre amour et votre soutien ont été mon refuge. Merci pour votre
présence constante et votre affection sans limites.

À la famille Cheurfi,
Merci pour votre soutien tout au long de ce parcours.

Rania,
Ma sœur de cœur, mon âme sœur. Tu as été là à chaque étape, dans les rires, les larmes
et les défis.
Merci pour ton amour inconditionnel, ta loyauté et ta présence qui rendent ma vie plus
belle.
Tu es bien plus qu'une amie, tu es ma famille.

À Amo Rachid et Tata Seham,

Vous m'avez accueillie comme votre propre fille, avec un amour et une tendresse qui ont comblé mon cœur.

Merci pour votre générosité sans limites, vos conseils précieux et cette chaleur familiale qui m'a tant manqué.

Vous êtes ma seconde famille.

À Nihed,

Ma camarade, mon soutien constant, celle qui a toujours été là avec ses mots réconfortants et ses encouragements sincères.

Merci d'avoir cru en moi, même quand j'en doutais.

À Amina, Amel, Nihed, Maria, Isra, Wessal, Manel et Aicha,

Un merci spécial à vous, pour les rires, les larmes et les souvenirs partagés.

Vous avez rendu ce parcours moins lourd.

À Ahmed,

Merci sincère pour ton aide précieuse et ta disponibilité.

À mon binôme, Khadija,

Ma sœur de cœur, ma partenaire d'aventure. On a commencé ensemble, on a lutté ensemble, et aujourd'hui, on triomphe ensemble.

Tu as été bien plus qu'une amie : mon miroir, ma motivation, et parfois ma raison de ne pas abandonner.

Ce mémoire est autant le tien que le mien.

Last but not least,

I wanna thank me... I wanna thank me for believing in me.

I wanna thank me for doing all this hard work.

I wanna thank me for having no days off.

I wanna thank me for never quitting.

I wanna thank me for always being a giver and try to give more than I receive.

I wanna thank me for trying to do more right than wrong.

I wanna thank me for just being me at all times.

Rihab

Dédicace

Je dédie ce travail :

À mes parents chéris, et Mamati

Votre amour, vos sacrifices silencieux et votre foi en moi ont été les piliers de mon parcours. Merci de m'avoir toujours soutenue, guidée et inspirée par votre sagesse, votre force et votre tendresse.

À mes frères Mohamed, Faïz et Alae, et à ma sœur Tasnime,

Votre présence m'a toujours réconfortée, même dans les moments les plus durs. Vos encouragements, vos sourires et votre humour ont été une véritable force pour moi. Merci d'avoir été là. Je vous porte dans mon cœur à chaque étape de ce parcours.

À ma cousine Nassima,

Ton soutien sincère, ta bienveillance et ta complicité ont fait de toi bien plus qu'une cousine : une sœur de cœur.

À toi, ma binôme, mon amie de toujours,

Franchement, je n'aurais pas pu rêver meilleure partenaire pour vivre cette aventure.

On a galéré, on a ri, on s'est soutenues... et on l'a fait, ensemble.

Merci pour ta patience, ta présence, et surtout pour être restée toi : sincère, drôle et toujours là. Je suis tellement chanceuse de t'avoir eue à mes côtés dans cette étape importante de ma vie.

À celui qui a toujours cru en moi,

Merci de m'avoir encouragée quand j'en avais le plus besoin, de m'avoir guidée avec patience et confiance, et de m'avoir rappelé que rien n'est hors de portée quand on y croit vraiment. Ta présence a été une lumière tout au long de ce chemin.

Et enfin, à moi-même,

Pour avoir persévéré malgré les doutes, cru en mes capacités et affronté chaque épreuve avec courage.

Ce mémoire est le fruit d'un long parcours, et aujourd'hui, je suis fière de moi.

Khadidja

Remerciements

En tout premier lieu, nous remercions **ALLAH**, tout puissant, de nous avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés, nous permettant ainsi de mener à bien ce travail.

Nous tenons à exprimer notre reconnaissance à **Monsieur AGGUINI**, pour son encadrement, ses orientations et sa disponibilité tout au long de ce projet.

Nos remerciements s'adressent également à **Madame KERRI**, dont le soutien, l'écoute attentive et les conseils pertinents ont été d'une grande aide. Son accompagnement a fortement contribué à l'avancement de notre travail.

Nous souhaitons aussi exprimer notre reconnaissance envers l'équipe de **CONDOR LOGISTIQUE**, l'entreprise qui nous a accueillis pour notre stage.

Nous adressons aussi nos remerciements à **Haroune BAKOUR** pour son aide et sa collaboration efficace tout au long de ce projet.

Un merci particulier à **Madame LINA BOUDA** pour son accompagnement.

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Enfin, nos pensées vont à toutes les personnes qui nous ont soutenus, encouragés ou aidés, de près ou de loin, dans la réalisation de ce projet.

Merci à toutes et à tous.

Résumé

Le problème de routage des véhicules avec ramassage et livraison simultanés (SPDVRP) est une extension du VRP classique, consistant à planifier les tournées de véhicules devant à la fois livrer et récupérer des marchandises chez les clients. Dans le cadre de notre projet de fin d'études chez *Condor Logistique*, nous avons étudié une version spécifique de ce problème en intégrant une contrainte de limite de temps sur les tournées, avec pour objectif la minimisation du *makespan*, c'est-à-dire la durée maximale de service parmi tous les véhicules.

Pour résoudre ce problème complexe, nous avons adopté deux approches complémentaires : une approche exacte via une modélisation mathématique avec *Pyomo* et le solveur *Gurobi*, et une approche approchée utilisant des algorithmes heuristiques et métaheuristiques, permettant de trouver rapidement des solutions satisfaisantes.

Mots-clés : Problème de routage avec ramassage et livraison simultanés, Logistique inverse, Limite de temps, Minimisation du makespan, Méthode exacte, Heuristique, Métaheuristique, Optimisation.

Abstract

The Simultaneous Pickup and Delivery Vehicle Routing Problem (SPDVRP) is an extension of the classical VRP, where vehicles must both deliver and collect goods from customers on the same route. As part of our final-year project with *Condor Logistique*, we studied a specific version of this problem by introducing a global time limit constraint and aiming to minimize the *makespan*, i.e., the maximum service duration among all vehicles.

To address this complex problem, we applied two complementary approaches : an exact mathematical formulation using *Pyomo* and the *Gurobi* solver, and an approximate approach based on heuristic and metaheuristic algorithms, which provide efficient solutions within a reasonable computational time.

Keywords : Vehicle Routing Problem, Simultaneous Pickup and Delivery, Reverse Logistics, Time Limit, Makespan Minimization, Exact Method, Heuristic, Metaheuristic, Optimization.

ملخص:

يُعدّ مشكل توجيه المركبات مع الالتقاط والتسليم المتزامنين امتدادًا للمشكل الكلاسيكي، حيث يتوجب على المركبات توصيل وجمع البضائع من الزبائن ضمن الجولة نفسها. في إطار مشروع نهاية الدراسة بشركة كوندور لوجيستيك ، تناولنا نسخة خاصة من هذا المشكل، وذلك بإدراج حدّ زمني أقصى لكل جولة، مع هدف يتمثل في تقليل المدة القصوى (مايكسبان) لجميع المركبات.

لمعالجة هذا الإشكال، اعتمدنا على نهجين متكاملين:

- نهج دقيق قائم على النمذجة الرياضية باستخدام مكتبة بايومو والمحلل قوروبي؛
- ونهج تقريبي يركز على الخوارزميات الاستدلالية (اوريستيكس) والخوارزميات الميتاستدلالية (ميتااوريستيكس) للحصول على حلول فعّالة ضمن وقت مقبول.

الكلمات المفتاحية: مشكل توجيه المركبات، الالتقاط والتسليم المتزامن، اللوجستيك العكسي، الحد الزمني، تقليل المدة القصوى، النمذجة الدقيقة، الخوارزميات التقريبية، التحسين.

Table des matières

Table des figures	I
Table des tableaux	II
Liste des abréviations	III
Introduction générale	1
1 Cadre conceptuel et théorique	4
I. La logistique :	4
1. Logistique directe :	4
2. La logistique inverse :	5
II. Logistique inverse dans le système logistique	6
III. Logistique inverse organisation et activite :	6
IV. Les principales étapes des flux de logistique inverse :	7
V. Les enjeux de la logistique inverse :	8
1. Les incitations réglementaires :	9
1.1 La problématique des emballages :	10
1.2 Piles et accumulateurs :	10
1.3 Les équipements électroniques :	10
1.4 Les véhicules hors d’usage (VHU) :	10
1.5 La pression concurrentielle :	10
2. L’intérêt économique : entre stratégie durable et rentabilité :	10
VI. Les différents types de flux dans la logistique inverse :	11
1. Produits neufs retournés ou invendus :	11
2. Produits défectueux à réparer ou à remettre en état :	11
3. Produits arrivés en fin de vie :	12
4. Déchets issus des processus industriels :	12
5. Déchets liés aux emballages :	12
6. Supports de manutention réutilisables :	13
7. Déchets d’usage domestique dispersés :	13
8. Déchets ménagers et assimilés :	13

VII. Défis et contraintes :	13
1. Incertitude sur les quantités et la fréquence des retours :	13
2. Problèmes de consolidation et d'optimisation des itinéraires :	14
3. Réglementations et contraintes environnementales :	14
2 Cadre methodologique et technique	15
I. Recherche Opérationnelle	15
II. Optimisation Combinatoire	16
III. Stratégies de Résolution en Optimisation Combinatoire	16
1. Méthodes exactes :	16
2. Méthodes approchées :	17
2..1 Les Heuristiques :	17
2..2 Méthodes métaheuristiques :	18
2..3 Les métaheuristiques basées sur la trajectoire :	18
2..4 Les métaheuristiques basées sur la population :	18
3. Évolution des méthodes de résolution des problèmes d'optimisation :	19
IV. Complexité	21
1. Complexité d'un algorithme	21
2. Complexité des problèmes	21
V. Problèmes d'optimisation	22
VI. Lien entre la complexité des problèmes et le choix des méthodes de résolution	23
VII. Vehicle Routing Problem (VRP) :	23
1. Les paramètres du problème de tournée de véhicules (VRP) :	24
2. Variante de VRP :	25
2..1 VRP avec profit (VRPP) :	25
2..2 CVRP (VRP avec contrainte de capacité)	25
2..3 VRP dynamique :	25
2..4 VRP avec fenêtre de temps "Time Window" (VRPTW) :	25
2..5 VRP avec contrainte LIFO :	25
2..6 VRP avec livraison et ramassage (VRPPD) :	25
2..7 Simultaneous VRP :	26
VIII. Les objectifs classiques en optimisation de tournées :	26
3 Formulation et Stratégie de Résolution du VRPSPD-TL	28
I. Enoncé du problème et modèle mathématique :	29
1. Description du problème :	29
2. Composants du problème :	29
3. Objectif :	29
II. Le modèle mathématique :	30
1. Formulation mathématique	31

1.1	Indices et paramètres	31
1.2	Variables de décision	31
2.	Explication des contraintes du modèle mathématique	32
3.	Remarque sur l'objectif	33
III.	Approche de résolution :	33
1.	Méthode exacte :	33
2.	Les heuristiques :	34
2.1	Construction Initiale : Clarke Wright Savings (CWS) : . .	34
3.	La recherche locale :	36
4.	Les métaheuristiques :	37
4.1	Optimisation avancée : Tabu Search	37
4	Implémentation et Analyse des Résultats	41
I.	Les outils Informatiques :	42
1.	Python :	42
1.1	Avantage de Python :	42
2.	Pyomo :	43
2.1	Avantage de Pyomo :	44
3.	Gurobi :	44
3.1	Avantage de Gurobi :	45
II.	Contexte Réel d'Application : Réseau Logistique de Condor :	46
III.	Méthodologie de Résolution :	46
1.	Résolution exacte d'une instance réduite	46
1.1	Transcription de la formulation mathématique en langage de programmation	47
2.	Déroulement de l'Algorithme	51
3.	Résultats Obtenus	52
3.1	Makespan Optimal	52
3.2	Tournées des Véhicules	52
3.3	Validation des Contraintes	52
3.4	Limitations rencontrées avec l'approche exacte :	53
IV.	Application à l'instance réelle (35 clients) :	54
1.	Méthode approchée :	54
1.1	Heuristique Constructive :	54
1.2	Méthode approchée : heuristique Clarke-Wright Savings : .	55
1.3	Utilité de la matrice des temps dans la modélisation : . . .	57
1.4	L'implémentation de l'Algorithme CWS :	57
1.5	Évaluation des tournées (calcul de la durée)	59
1.6	Résultats Obtenus :	61

1..7	Interprétation de la solution initiale (Clarke & Wright Savings) :	62
1..8	Limites de la solution initiale	62
2.	Heuristique amélioratives (Relocate) :	62
2..1	Résultats obtenus	65
2..2	Interprétation de la solution améliorée	66
3.	Métaheuristique :	67
3..1	Tabu- search :	67
3..2	Objectif général :	67
3..3	Schéma explicatif de Tabu-search :	67
3..4	Le déroulement de l'algorithme Tabu :	68
3..5	L'implémentation :	68
3..6	Résultats Obtenus :	71
3..7	Interprétation des résultats	72
V.	Évaluation comparative des approches d'optimisation :	72
Conclusion générale et perspectives		75
A Présentation de l'entreprise CONDOR LOGISTICS		77
I.	Historique :	77
II.	Le Domaine d'activité de Condor Logistique :	78
III.	Les missions de Condor logistique :	79

Table des figures

1.1	La différence entre la logistique inverse et la logistique directe. (réalisation personnelle)	5
1.2	Les principales étapes des flux de logistique inverse. (Pucheu, F. Beauvy, E., 2020)	9
2.1	Méthodes de résolution pour l'optimisation de tournées (source : réalisation personnelle)	20
2.2	Classification des complexités de problèmes.	21
2.3	Vehicle Routing Problem	24
3.1	Le recherche Locale par Déplacement de Solution (Douiri et al .cours des méthodes de résolution)	36
4.1	Logo du Python	42
4.2	Logo du Pymo	43
4.3	Logo du Gurobi	45
4.4	Visualisation des tournées optimisées autour du dépôt	53
4.5	Schéma de fonctionnement de l'algorithme Tabu Search(Youle Song, Ding Nie, Litao . Journal of robotics)	68
A.1	Logo de l'entreprise CONDOR LOGISTICS	77

Liste des tableaux

1.1	Principales différences entre logistique traditionnelle et logistique inverse (Adapté de Rogers et al., 2001)	7
2.1	Les objectifs classiques en optimisation de tournées.	26
4.1	Données initiales du problème	55
4.2	Synthèse des résultats avant et après Tabu-search	72
4.3	Résumé des résultats obtenus selon la méthode appliquée	73

Liste des abréviations

VHU : Véhicule Hors d'Usage

DIB : Déchets Industriels Banals

REP : Responsabilité Élargie du Producteur

DEEE : Déchets d'Équipements Électriques et Électroniques

DIS : Déchets Industriels Spéciaux

VRP : Vehicle Routing Problem (Problème de Tournée de Véhicules)

RO : Recherche Opérationnelle

PLNE : Programmation Linéaire en Nombres Entiers

TSP : Traveling Salesman Problem (Problème du Voyageur de Commerce)

VNS : Variable Neighborhood Search

NP : Problème Non Déterministe Polynomial

P : Problème Polynomial

PVC : Problème du Voyageur de Commerce (autre sigle pour TSP)

CVRP : Capacitated Vehicle Routing Problem (VRP avec contraintes de capacité)

VRPP : Vehicle Routing Problem with Profit

VRPTW : Vehicle Routing Problem with Time Windows (fenêtres de temps)

LIFO : Last In, First Out (dernier entré, premier sorti)

VRPPD : Vehicle Routing Problem with Pickup and Delivery

VRPSPD : Vehicle Routing Problem with Simultaneous Pickup and Delivery

VRPSPD-TL : Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Limit

MILP : Mixed Integer Linear Programming (Programmation Linéaire en Nombres Entiers Mixtes)

CWS : Clarke and Wright Savings

LP : Linear Programming (Programmation Linéaire)

API : Application Programming Interface

CPU : Central Processing Unit (Unité Centrale de Traitement)

QHSE : Qualité, Hygiène, Sécurité, Environnement

TEU : Twenty-foot Equivalent Unit (Unité Équivalente Vingt Pieds – mesure de conteneurs)

Introduction générale

Dans un contexte logistique en constante évolution, les entreprises sont contraintes de revoir leurs processus pour rester performantes, compétitives et répondre aux attentes des clients en termes de satisfaction. La gestion optimale des flux logistiques, qu'ils soient directs ou inverses, constitue un enjeu stratégique, particulièrement dans les secteurs fortement dépendants de la distribution et du transport. Cet enjeu est amplifié par la raréfaction des ressources, les impératifs de développement durable visant la préservation de l'énergie, ainsi que les réglementations exigeant le recyclage des déchets.

C'est dans ce cadre que s'inscrit notre travail, mené au sein de l'entreprise Condor Logistique, spécialisée dans la distribution de produits électroniques et électroménagers à travers le territoire national.

Sur le terrain, l'entreprise est confrontée à des difficultés liées à la planification des tournées de véhicules avec ramassage et livraison simultanés. Les méthodes actuelles de gestion, principalement manuelles ou basées sur des règles empiriques, entraînent un déséquilibre dans la répartition des missions entre les véhicules. Ce déséquilibre se traduit par une surcharge de certaines tournées, une sous-utilisation d'autres, et un allongement significatif du temps total d'exécution des livraisons, appelé makespan.

Ce constat soulève une question centrale :

Comment peut-on minimiser le makespan dans un problème de tournées de véhicules avec ramassage et livraison simultanés, tout en respectant les contraintes réelles de capacité, de faisabilité, de service, ainsi que les impératifs de durabilité et de conformité réglementaire propres à l'activité de Condor Logistique ?

Afin de mieux comprendre les fondements de cette problématique, le premier chapitre de ce mémoire est consacré au cadre conceptuel et théorique. Il débute par une présentation de la logistique directe et de la logistique inverse, avant d'analyser la place de cette dernière dans le système logistique global. L'organisation des activités de logistique inverse, ses étapes clés, ses enjeux économiques, environnementaux (notamment la préservation de l'énergie et le recyclage des déchets) et ses contraintes réglementaires y sont détaillés. Différents types de flux inverses sont présentés, incluant les produits retournés, défectueux ou en fin de vie, ainsi que les déchets industriels et domestiques. Enfin, les défis liés à l'incertitude des retours, à la consolidation des itinéraires, et aux

contraintes environnementales seront abordés, posant ainsi les bases théoriques nécessaires à l'étude pratique développée dans les chapitres suivants.

Pour répondre à la problématique, ce mémoire propose une modélisation mathématique du problème, suivie de l'application de différentes techniques de résolution. Une approche exacte est d'abord testée pour valider la formulation sur des cas de petite taille.

Ensuite, face à la complexité croissante des instances, des méthodes approchées sont mises en œuvre, notamment des heuristiques constructives, des techniques de recherche locale, ainsi qu'une méta-heuristique Tabu Search, afin d'obtenir des solutions de qualité dans des temps de calcul raisonnables.

Chapitre 1

Cadre conceptuel et théorique

Introduction :

Ce chapitre introduit les concepts fondamentaux de la logistique, en distinguant la logistique directe, qui concerne l'acheminement des produits du producteur vers le client, et la logistique inverse, qui gère les flux de retour, qu'il s'agisse de produits défectueux, en fin de vie ou à revaloriser.

Après avoir défini ces deux formes de logistique, l'accent est mis sur l'importance croissante de la logistique inverse dans le fonctionnement global des chaînes logistiques, notamment pour répondre aux exigences environnementales, aux réglementations en vigueur et à la nécessité de maîtriser les coûts liés aux retours.

I. La logistique :

La logistique est l'une des fonctions essentielles pour l'entreprise, car elle regroupe l'ensemble des activités en œuvre pour assurer la disponibilité d'un bien ou d'un service, dans les lieux où les besoins existent et garantir une gestion optimale de la combinaison (quantités, délais et coûts). Cette dernière ne peut se limiter seulement à l'organisation des transports, des matières premières et de marchandises, mais elle regroupe l'ensemble des techniques de contrôle et de gestion des flux des matières premières et des produits depuis leurs sources d'approvisionnement jusqu'au point de consommation.

Le développement de la logistique a conduit à l'apparition de deux formes distinctes : la logistique directe et la logistique inverse incluant notamment le modèle de boucle fermée (closed-loop supply chain), où les produits usagés réintègrent le cycle de production

1. Logistique directe :

Elle concerne le processus classique d'acheminement des produits depuis le fabricant jusqu'au consommateur final. Ce flux est linéaire et vise à optimiser la distribution des

biens en minimisant les coûts et les délais tout en garantissant la satisfaction du client.

2. La logistique inverse :

La logistique inversée se distingue clairement de la logistique traditionnelle. Alors que cette dernière suit un chemin classique allant du producteur vers le consommateur, la logistique inversée fait le trajet en sens inverse, en partant du client pour remonter jusqu'au fabricant. (figure 1.1)

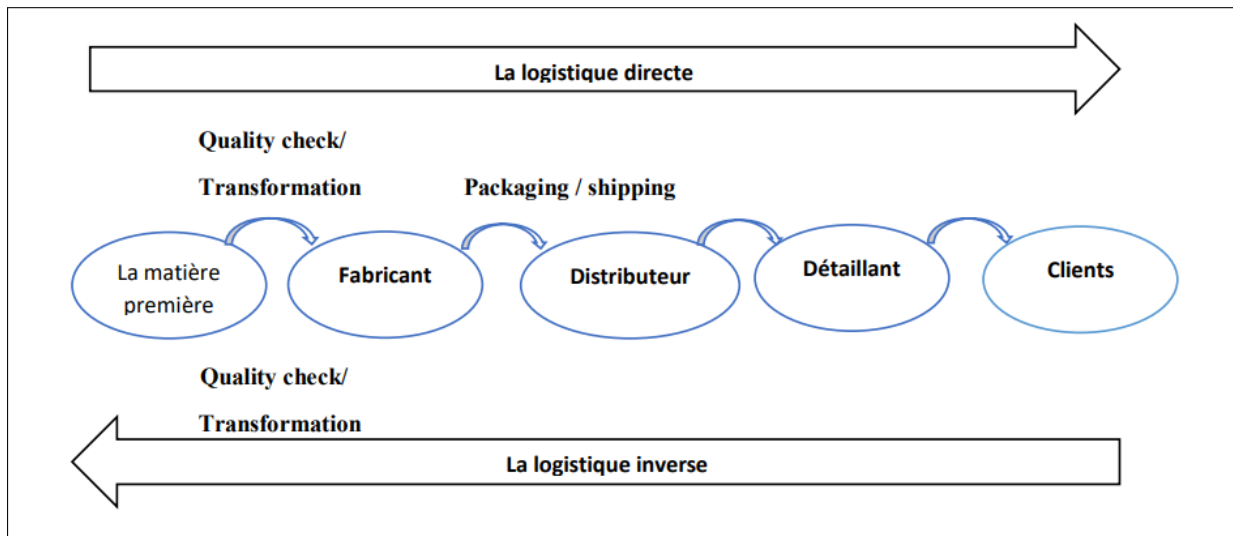


FIGURE 1.1 – La différence entre la logistique inverse et la logistique directe. (réalisation personnelle)

La logistique inversée regroupe les opérations liées au retour des produits, matériaux ou emballages depuis leur point d'utilisation ou de consommation vers un lieu de collecte, de traitement ou de revalorisation. Elle est apparue dès les années 1990, avec une première définition proposée par Stock (1992), qui l'associe à la gestion des déchets, à la récupération des matériaux et à la réduction de l'impact environnemental. Cette vision environnementale a ensuite été complétée par d'autres approches. Certains auteurs, comme Thierry et al. (1995), se concentrent sur les retours de produits sans inclure les déchets, tandis que d'autres les considèrent essentiels, notamment pour la récupération d'emballages réutilisables (palettes, bacs, etc.).

Une définition de référence a été proposée par Rogers et Tibben-Lembke (1998), qui présentent la logistique inversée comme un processus de planification, d'exécution et de contrôle des flux physiques et informationnels, depuis le consommateur jusqu'à un point d'origine, dans le but de récupérer de la valeur ou d'éliminer les produits de manière appropriée. Cette définition a l'avantage de mettre clairement en avant le rôle central du consommateur et les objectifs économiques et environnementaux.

Cependant, elle reste centrée sur le retour vers le point d'origine. C'est pourquoi De Brito et Dekker (2003) proposent une approche plus ouverte, en soulignant que les flux

inversés peuvent partir de différents points de la chaîne (consommation, distribution, production) vers des centres de traitement adaptés.

Enfin, la logistique inversée suit un processus similaire à celui de la logistique classique, avec des étapes telles que la collecte, le tri, le traitement (réparation, remanufacturing, recyclage) et la redistribution des produits revalorisés, un cycle pouvant se répéter plusieurs fois durant la vie du produit

II. Logistique inverse dans le système logistique

D'après Lu et al. (2001), l'émergence de la logistique inverse s'explique par la nécessité croissante pour les entreprises de mieux organiser la gestion des retours. En effet, les systèmes logistiques classiques ne sont généralement pas conçus pour prendre en charge ces flux inverses. Dawe (1995) souligne d'ailleurs que la logistique inverse reste souvent reléguée au second plan dans les priorités stratégiques des entreprises.

Par ailleurs, Witt (1995) observe que l'essor des ventes à distance (notamment en ligne) entraînera une augmentation significative du volume de retours, avec environ 15 concernés. Il est également essentiel de noter que le taux de retour varie considérablement selon les secteurs. À ce propos, Rogers et Tibben-Lembke (1998) indiquent, à travers une enquête, que les taux de retour oscillent entre 2 à 3 % pour les produits chimiques ménagers et peuvent atteindre jusqu'à 50 % pour certaines publications périodiques.

III. Logistique inverse organisation et activite :

La logistique inverse constitue un processus multidimensionnel qui s'articule autour de trois axes fondamentaux. Premièrement, elle englobe un ensemble d'activités de valorisation incluant la récupération, le tri, la remise en état et le recyclage des produits retournés, tout en intégrant des considérations environnementales telles que l'éco-conception et la réduction des déchets. Deuxièmement, son organisation se distingue radicalement de la logistique traditionnelle par sa nature complexe et imprévisible, avec des flux convergents (multiples points vers un centre) et une gestion rendue difficile par la variabilité des produits et l'absence de standardisation (Rogers et al., 2001). Enfin, elle présente des défis opérationnels spécifiques liés à la réduction des cycles de vie des produits, à l'opacité des coûts et à la nécessité d'une réactivité permanente face aux retours. Cette triple caractéristique - activités de valorisation, organisation complexe et défis spécifiques - fait de la logistique inverse un élément stratégique mais difficile à maîtriser dans la chaîne logistique actuelle.

TABLE 1.1 – Principales différences entre logistique traditionnelle et logistique inverse (Adapté de Rogers et al., 2001)

Aspect	Logistique traditionnelle	Logistique inverse
Prévision	Relativement simple	Difficile en raison de l'aléa
Points de distribution	Un vers plusieurs	Plusieurs vers un
Qualité des produits	Homogène	Hétérogène
Emballage	Standardisé	Varié et non uniforme
Destination / route	Bien définie	Souvent incertaine
Options de disposition	Clares et planifiées	Mal définies
Politique de prix	Stable	Dépend de plusieurs critères
Priorité de la rapidité	Haute	Peu prioritaire
Coûts de distribution	Facilement mesurables	Difficiles à estimer
Gestion des stocks	Maîtrisée	Souvent incohérente
Cycle de vie produit	Prévisible	Plus contraignant
Négociation	Directe	Plus complexe
Marketing	Méthodes éprouvées	Moins claires, plus délicates
Visibilité du processus	Transparente	Réduite

IV. Les principales étapes des flux de logistique inverse :

Comme l'ont démontré De Brito et Dekker (2003) et Lambert et al. (2011), la logistique inverse s'organise selon un processus structuré en cinq étapes clés. La première étape, le contrôle des retours (gatekeeping), joue un rôle crucial en filtrant les produits selon leur état, leur conformité et les politiques de retour, permettant ainsi d'optimiser les coûts en amont (Lambert et al., 2011).

La collecte constitue la deuxième étape, organisant le transport des articles depuis les points de retour vers les centres de traitement. Vient ensuite l'inspection et le tri, où chaque produit est systématiquement évalué pour déterminer son potentiel de

valorisation (De Brito Dekker, 2003). Les options incluent la réutilisation directe, la revente, la réparation ou le recyclage.

Le retraitement, quatrième étape, peut prendre différentes formes : remise à neuf, reconditionnement, ou récupération de pièces. Enfin, la redistribution réintègre les produits traités dans divers circuits, complétant ainsi la boucle de l'économie circulaire.

Ce processus complet, bien que complexe, transforme les retours en véritables opportunités de création de valeur.

Plusieurs options s'offrent alors à l'entreprise :

1. Réparation :

L'entreprise intervient uniquement sur les composants défectueux, permettant une remise en circulation rapide et peu coûteuse du produit.

2. Remise à neuf :

Un contrôle approfondi des composants critiques et le remplacement des pièces usagées redonnent au produit des caractéristiques proches de son état d'origine, sans atteindre le niveau du neuf.

3. Remanufacturation :

Ce processus industriel complet, courant dans l'automobile et l'électronique, démonte entièrement le produit pour le reconstruire selon des normes strictes, équivalentes à celles du neuf.

4. Cannibalisation :

Les pièces encore fonctionnelles sont extraites sélectivement pour alimenter les stocks de pièces détachées ou servir à d'autres réparations, maximisant ainsi la valeur résiduelle.

5. Recyclage :

Quand toute réutilisation est impossible, les matières premières sont récupérées et réintroduites dans les circuits de production, minimisant l'impact environnemental.

Ces différentes stratégies de récupération permettent de limiter le gaspillage, de valoriser

V. Les enjeux de la logistique inverse :

Avec l'essor du commerce électronique, des préoccupations environnementales et de la compétitivité industrielle, la logistique inverse est devenue une fonction stratégique.

Trois grandes motivations expliquent son développement dans les entreprises : les contraintes réglementaires, la pression de la concurrence, et les opportunités économiques qu'elle offre.



FIGURE 1.2 – Les principales étapes des flux de logistique inverse. (Pucheu, F. Beauvy, E., 2020)

1. Les incitations réglementaires :

Au fil des années, les autorités publiques ont pris conscience des impacts environnementaux des activités industrielles et logistiques. Suite à des événements internationaux majeurs, tels que le Sommet de la Terre à Rio en 1992 et le Protocole de Kyoto en 2005, les États ont renforcé leur engagement en faveur d'un développement durable. Ils ont mis en place des politiques visant à responsabiliser les entreprises concernant leurs produits, même après leur mise sur le marché.

Les gouvernements ont adopté diverses approches, allant de réglementations strictes à des incitations, telles que des lois antipollution, des normes environnementales et des politiques de taxation des déchets. Ces mesures encouragent les entreprises à intégrer la gestion de la fin de vie de leurs produits dans leur stratégie logistique, tout en préservant leur développement économique. Certaines entreprises pionnières ont anticipé ces exigences en établissant des dispositifs de récupération et de traitement avant l'entrée en vigueur de la législation, ce qui leur a permis de mieux gérer les contraintes réglementaires futures et d'améliorer leur image de marque.

La notion de responsabilité élargie du producteur (REP) est de plus en plus intégrée dans les textes législatifs, obligeant les entreprises à prendre en charge le cycle de vie de leurs produits, y compris leur collecte et leur élimination. Ce principe s'applique particulièrement aux secteurs à fort impact environnemental, tels que l'industrie chimique, les hôpitaux et le domaine nucléaire. Les réglementations récentes se concentrent sur des catégories de produits spécifiques, comme les emballages, les piles, les équipements électriques et électroniques (DEEE) et les véhicules hors d'usage, imposant des systèmes de reprise et de traitement qui favorisent le développement des réseaux de logistique inverse.

1..1 La problématique des emballages :

Les emballages représentent une part importante des déchets générés, constituant plus d'un tiers des déchets ménagers et industriels en Europe. Ils se distinguent en deux catégories aux enjeux distincts : le conditionnement primaire, qui enveloppe directement le produit pour le protéger et le valoriser auprès du consommateur, souvent caractérisé par un usage excessif de plastiques rigides, et l'emballage secondaire, destiné à faciliter le stockage, la manutention et le transport des produits. Ce dernier, bien que moins visible, constitue une part conséquente des coûts logistiques supportés par les entreprises.

Face à leur impact environnemental grandissant, de nombreux pays ont instauré des réglementations strictes, notamment via la responsabilité élargie du producteur, qui contraint les fabricants à financer la collecte, le tri et le recyclage des emballages. La logistique inverse s'impose ainsi comme un élément clé pour organiser efficacement le retour et la valorisation des matériaux d'emballage, participant activement à l'économie circulaire et à la réduction des déchets.

1..2 Piles et accumulateurs :

Ces déchets dangereux nécessitent une prise en charge spécifique. La logistique inverse permet leur retour sécurisé vers des filières de recyclage.

1..3 Les équipements électroniques :

Face à leur croissance rapide, ces déchets doivent être récupérés et valorisés. La logistique inverse en assure le tri et le traitement.

1..4 Les véhicules hors d'usage (VHU) :

Leur recyclage implique plusieurs étapes techniques. La logistique inverse organise leur acheminement vers les centres agréés.

1..5 La pression concurrentielle :

Les retours clients se multiplient avec le e-commerce et les garanties. La logistique inverse devient un atout stratégique pour les entreprises.

2. L'intérêt économique : entre stratégie durable et rentabilité :

La logistique inverse ne se limite pas à une exigence réglementaire ; elle représente également une opportunité économique réelle. En intégrant des pratiques telles que l'écoconception, le recyclage, le réemploi ou le reconditionnement, les entreprises peuvent réduire leurs coûts, notamment dans les domaines de la maintenance, de la

location ou du service après-vente. Dans un contexte de raréfaction des ressources, la récupération et la valorisation de composants fiables deviennent des leviers stratégiques.

Par ailleurs, l'engagement dans une démarche écoresponsable améliore l'image de marque et permet de se différencier sur des marchés de plus en plus sensibles aux enjeux environnementaux.

VI. Les différents types de flux dans la logistique inverse :

La logistique inverse englobe de multiples flux générés à différentes étapes du cycle de vie d'un produit, depuis sa fabrication jusqu'à son élimination finale. Ces flux, variés tant dans leur nature que dans leurs causes, peuvent être classés en huit grandes catégories selon leur origine et les traitements qu'ils nécessitent.

1. Produits neufs retournés ou invendus :

Cette première catégorie regroupe des articles ne présentant pas de défaut technique, mais qui sont réintroduits dans la chaîne logistique pour différentes raisons :

- Des clients peuvent renvoyer des produits simplement parce qu'ils ne leur conviennent pas après un premier essai ou un changement d'avis,
- Certains articles sont refusés à la réception pour non-conformité aux exigences qualité du client (par exemple, suite à un échantillonnage révélant un taux de défauts supérieur à la tolérance),
- Des produits peuvent également être repris chez les distributeurs car ils ne respectent plus certaines conditions commerciales (comme une date limite trop proche),
- Des invendus ou articles promotionnels retournés à cause d'une anticipation erronée de la demande,
- Des erreurs dans la livraison (mauvais étiquetage, documents manquants, colis manquant sur une palette, etc.) peuvent entraîner le retour complet de la marchandise,

Enfin, certains produits sont retournés ou retirés du marché car leur étiquetage ou conditionnement n'est plus conforme à la réglementation (par exemple, dans le secteur pharmaceutique lorsqu'un changement de remboursement impose une nouvelle vignette).

2. Produits défectueux à réparer ou à remettre en état :

Ce type de flux concerne les produits qui présentent un dysfonctionnement ou un dommage après leur livraison :

Cela inclut les retours effectués sous garantie, suite à une panne ou un défaut constaté par le client ou le revendeur, il peut également s'agir de produits endommagés pendant le transport, rendant l'ensemble de la livraison inacceptable, dans certains cas, des rappels sont organisés par les fabricants lorsqu'un défaut ou un risque a été identifié après la mise sur le marché (ex. : rappel de denrées alimentaires ou de véhicules présentant un vice caché).

3. Produits arrivés en fin de vie :

Certains produits arrivent naturellement en fin de cycle, et leur sort dépend souvent du type d'usage et de la réglementation :

Une partie est simplement jetée dans les ordures ménagères ou déposée dans des déchetteries, ce qui implique une prise en charge par les collectivités, et une autre partie est reprise volontairement par les professionnels (ex. : reprise d'un ancien appareil lors de la livraison d'un neuf) ou dans le cadre d'une obligation réglementaire (ex. : reprise des véhicules hors d'usage).

4. Déchets issus des processus industriels :

Les activités industrielles génèrent différents types de déchets à traiter en logistique inverse.

- Les déchets inertes, comme les gravats, sont peu polluants mais leur enfouissement est interdit depuis 2002, ce qui impose leur réemploi.
- Les déchets industriels banals (DIB), proches des déchets ménagers, sont généralement recyclables ou valorisables.
- Les déchets industriels spéciaux (DIS), plus dangereux, nécessitent un traitement encadré et un suivi réglementaire rigoureux.

5. Déchets liés aux emballages :

Cette catégorie comprend les matériaux d'emballage et de conditionnement : cartons, plastiques, fûts, etc.

Leur mise en décharge est interdite et leur valorisation est obligatoire, que ce soit par réutilisation, recyclage ou récupération énergétique.

Les emballages souillés (ayant contenu des substances dangereuses) sont traités comme des déchets dangereux. Certains peuvent être nettoyés et remis sur le marché, les autres sont broyés ou pressés après traitement.

6. Supports de manutention réutilisables :

Palettes, conteneurs, rolls ou housses sont des éléments qui assurent le transport et la manutention des marchandises . Ces supports sont conçus pour être réutilisés, sous réserve d’inspection ou de réparations, leur gestion est bien établie, notamment dans la grande distribution et les entrepôts logistiques.

7. Déchets d’usage domestique dispersés :

Certains déchets sont générés par l’usage courant des produits par les consommateurs : piles usagées, huiles de vidange, emballages souillés, solvants, bombes aérosols ou produits chimiques, en raison de leur toxicité et de leur faible concentration géographique, leur collecte repose souvent sur des dispositifs incitatifs ou la bonne volonté des usagers.

8. Déchets ménagers et assimilés :

Les déchets ménagers et ceux produits par de petites structures économiques (commerçants, artisans, administrations) sont généralement pris en charge par les services municipaux, sauf en cas de risques particuliers.

Cette diversité de flux illustre l’hétérogénéité de la logistique inverse. Si certains retours relèvent des entreprises, d’autres impliquent les consommateurs finaux. Toutefois, les évolutions législatives renforcent la responsabilité des producteurs, notamment à travers le principe de responsabilité élargie, comme dans le secteur automobile.

VII. Défis et contraintes :

Le transport dans la logistique inverse présente plusieurs contraintes et défis majeurs, liés à la nature même des flux de retours, qui sont souvent imprévisibles et complexes à gérer.

1. Incertitude sur les quantités et la fréquence des retours :

Contrairement à la logistique traditionnelle où les flux sont planifiés et relativement stables, la logistique inverse doit faire face à une forte incertitude concernant les volumes et la fréquence des produits retournés. Ces retours peuvent varier en fonction de nombreux facteurs comme la saisonnalité, la satisfaction client, ou encore des campagnes promotionnelles. Cette imprévisibilité complique la planification des ressources de transport et des capacités de stockage, ce qui peut entraîner des surcoûts et des retards dans le traitement des retours.

Pour limiter cet impact, les entreprises analysent les données historiques des retours pour anticiper au mieux les flux, mais cette démarche reste partielle car la variabilité demeure élevée.

2. Problèmes de consolidation et d'optimisation des itinéraires :

Les retours proviennent souvent de multiples points géographiques dispersés (clients individuels, magasins, points relais), ce qui rend la consolidation des flux difficile. Regrouper efficacement ces retours est essentiel pour éviter des trajets à vide et réduire les coûts de transport. Cependant, la diversité des produits retournés (différentes tailles, états, catégories) complique leur regroupement.

L'optimisation des itinéraires de collecte nécessite l'utilisation de logiciels spécialisés pour planifier des tournées efficaces, prenant en compte la dispersion des points de collecte et les contraintes de temps. Par ailleurs, le manque d'espaces dédiés au stockage des retours dans les entrepôts classiques peut retarder leur traitement, impactant la fluidité du transport inverse.

3. Réglementations et contraintes environnementales :

Le transport dans la logistique inverse doit aussi se conformer à des réglementations strictes, notamment en matière de gestion des déchets, de recyclage et de protection de l'environnement. Par exemple, certains produits comme les équipements électroniques ou les emballages doivent être acheminés vers des centres spécifiques pour être recyclés ou valorisés.

Ces obligations légales imposent souvent des circuits de transport dédiés, augmentant la complexité et le coût des opérations. Par ailleurs, les entreprises sont de plus en plus incitées à réduire leur empreinte écologique en adoptant des modes de transport moins polluants et en favorisant les circuits courts. La gestion responsable des emballages et la sensibilisation des consommateurs aux retours durables font également partie des contraintes à intégrer dans la stratégie logistique inverse.

Conclusion :

Ce chapitre a permis de différencier les concepts fondamentaux de la logistique directe et inverse, tout en soulignant leur complémentarité. La logistique directe, structurée autour de flux prévisibles (fournisseur $\rightarrow$ client), vise l'efficacité des livraisons, tandis que la logistique inverse, plus réactive, gère des retours complexes (client $\rightarrow$ fournisseur) pour en maximiser la valorisation (réemploi, recyclage, etc.). Leurs différences - linéarité vs. incertitude des flux, standardisation vs. adaptation des processus - révèlent aussi leur interdépendance dans une supply chain durable.

Chapitre 2

Cadre methodologique et technique

Introduction

Ce chapitre traite de l'optimisation pratique de la logistique inverse, indispensable pour assurer une gestion efficace des flux de retour. Il met en lumière le rôle central de la recherche opérationnelle, qui permet de modéliser, analyser et optimiser les opérations logistiques à travers des outils mathématiques.

L'approche se concentre particulièrement sur la planification des tournées de récupération, en intégrant contraintes réelles et objectifs économiques. À ce titre, le Problème de Tournée de Véhicules (VRP) et ses variantes sont présentés comme des instruments puissants pour optimiser les itinéraires de collecte dans un contexte de logistique inverse, en contribuant à la réduction des coûts, du temps et de l'impact environnemental.

I. Recherche

Opérationnelle

La Recherche Opérationnelle (RO) se positionne comme une discipline clé au sein des mathématiques appliquées, se consacrant à l'étude et à la résolution des problématiques liées à l'allocation et à l'utilisation les plus efficaces des ressources disponibles.

Initialement développée pour des applications militaires, son champ d'action s'est considérablement élargi au fil du temps, touchant désormais des secteurs variés tels que l'industrie, les services publics, mais aussi l'économie, la finance, les stratégies marketing et la planification globale d'entreprise. Fondamentalement, la RO peut être perçue comme un ensemble structuré de méthodes et de techniques logiques et quantitatives. Son objectif principal est d'aider à la prise de décision en identifiant le meilleur choix possible parmi un éventail d'options pour opérer, afin d'atteindre un objectif prédéfini, d'obtenir le meilleur résultat réalisable dans les circonstances données, ou de parvenir à la solution strictement optimale.

II. Optimisation

Combinatoire

L'Optimisation Combinatoire représente un domaine spécifique au sein de la programmation mathématique, focalisé sur l'identification de la solution la plus avantageuse au sein d'un ensemble discret de possibilités. Bien que le nombre de solutions potentielles soit fini, il peut être extrêmement vaste, rendant l'énumération exhaustive impraticable. Cette discipline englobe un ensemble de techniques visant à déterminer la valeur optimale (maximum ou minimum) d'une fonction objectif, tout en respectant un ensemble de contraintes définies. L'enjeu réside souvent dans la manipulation d'ensembles dont la structure mathématique complexe ne se prête pas aisément à une caractérisation simple, nécessitant des approches algorithmiques spécifiques pour naviguer efficacement dans l'espace des solutions.

III. Stratégies de Résolution en Optimisation Combinatoire

Pour aborder les défis posés par l'optimisation combinatoire, la recherche opérationnelle propose un arsenal de techniques de résolution. Celles-ci se regroupent principalement en deux catégories distinctes :

- **Les méthodes exactes** : visent à garantir l'identification de la solution rigoureusement optimale. Bien que leur coût computationnel puisse être élevé, elles assurent la complétude de la recherche.
- **Les méthodes approchées** : cherchent des solutions de très bonne qualité en un temps raisonnable, sans garantir l'optimalité absolue.

1. Méthodes exactes :

Les méthodes exactes sont utilisées pour trouver la solution optimale en modélisant le problème sous forme de **programme linéaire en nombres entiers (PLNE)**. Elles prennent en compte des contraintes comme la capacité des véhicules, les distances et les fenêtres de temps.

Les techniques classiques incluent :

- **Branch and Bound** : explore l'espace des solutions de façon arborescente, en éliminant des branches non prometteuses.
- **Branch and Cut** : amélioration du précédent, ajoutant des contraintes (coupes) pour réduire l'espace de recherche.
- **Programmation dynamique** : utilisée pour certains sous-problèmes de type séquentiel ou de taille réduite.

2. Méthodes approchées :

Les méthodes approchées sont des techniques qui visent à fournir des solutions satisfaisantes à des problèmes complexes en un temps de calcul raisonnable.

Dans les problèmes d'optimisation combinatoire tels que le VRP, où le nombre de solutions possibles croît de manière exponentielle, les méthodes approchées représentent une alternative précieuse, surtout en contexte industriel.

Elles sont souvent divisées en deux grandes familles :

2.1 Les Heuristiques :

Les heuristiques sont des techniques rapides qui donnent de bonnes solutions (mais pas forcément optimales).

On distingue deux types principaux :

Heuristiques constructives :

Les heuristiques constructives **construisent une solution à partir de zéro**, en ajoutant progressivement des éléments de manière intelligente et structurée. Elles sont souvent utilisées comme point de départ pour des méthodes d'amélioration ultérieures.

Quelques heuristiques constructives notables :

- **Clarke and Wright Savings** : commence par attribuer un véhicule à chaque client, puis fusionne les tournées en fonction d'un critère d'économie de distance ou de coût.
- **Nearest Neighbor (plus proche voisin)** : commence par un dépôt ou un client aléatoire, puis ajoute à chaque étape le client le plus proche non encore visité.
- **Farthest Insertion** : démarre avec une tournée partielle, puis ajoute à chaque étape le client le plus éloigné de la tournée actuelle.
- **Sweep Algorithm** : trie les clients par angle polaire autour du dépôt et les regroupe dans des tournées en respectant les contraintes de capacité.
- **Insertion heuristics** (cheapest insertion, nearest insertion, etc.) : insèrent les clients un par un dans les positions minimisant un certain critère (coût, distance...).

Heuristiques d'amélioration :

Partent d'une solution existante et la modifient pour l'améliorer.

- **Relocate** : consiste à déplacer un client d'une tournée à une autre pour réduire le coût total.
- **2-opt, 3-opt** : techniques de permutations locales utilisées notamment dans le TSP.

Avantages des heuristiques :

- **Utilisables comme point de départ :** Les solutions générées peuvent servir de base à des méthodes d'amélioration ou à des métaheuristiques.
- **Rapidité de calcul :** Les heuristiques sont généralement très rapides et peuvent fournir une solution en quelques secondes, même pour des instances de taille moyenne à grande.
- **Simplicité d'implémentation :** Elles sont faciles à programmer et à comprendre, avec peu de paramètres à ajuster.

Inconvénients des heuristiques :

- **Pas de garantie d'optimalité :** Les solutions obtenues ne sont pas garanties comme étant optimales, et leur écart par rapport à l'optimum est souvent inconnu.
- **Risque de solutions médiocres :** En se basant sur des décisions locales (ex. plus proche voisin), les heuristiques peuvent facilement rester bloquées dans des solutions de mauvaise qualité.
- **Manque de vision globale :** Les heuristiques n'explorent qu'une petite partie de l'espace des solutions, ce qui limite leur capacité à découvrir des solutions innovantes.

2..2 Méthodes métaheuristiques :

Les métaheuristiques sont des stratégies de haut niveau qui guident intelligemment la recherche dans des espaces de solution vastes et complexes. Elles permettent d'équilibrer l'exploration de nouvelles solutions (diversification) et l'exploitation des meilleures solutions déjà trouvées (intensification).

On distingue généralement deux grandes familles :

2..3 Les métaheuristiques basées sur la trajectoire :

solutions en suivant un chemin unique à chaque itération, en modifiant progressivement une solution de départ. Parmi elles :

- **Tabu Search** (Glover, 1977), qui utilise une liste tabou et un critère d'aspiration,
- **Simulated Annealing**, inspirée du processus de recuit métallurgique,
- **Variable Neighborhood Search (VNS)**, qui explore dynamiquement différents voisinages.

2..4 Les métaheuristiques basées sur la population :

elles manipulent simultanément plusieurs solutions (une population) et s'inspirent souvent de phénomènes naturels ou biologiques. Exemples :

- **Algorithme génétique**, fondé sur les principes de la sélection naturelle, du croisement et de la mutation,
- **Algorithme à colonies de fourmis**, basé sur la coopération indirecte d'agents artificiels.

Avantages des métaheuristiques

- Capacité à **explorer un grand espace de recherche**, sans être limité à une région locale.
- Aptitude à **échapper aux minima locaux**.
- Adaptabilité à des **problèmes complexes et multi-contraints** comme le VRP, même à grande échelle.

Inconvénients :

- **Paramétrage délicat** : la performance dépend fortement du bon choix des paramètres (taille de population, température initiale, durée de l'interdiction tabou, etc.).
- **Temps de convergence** parfois long si mal calibrées.
- **Implémentation plus complexe** que les heuristiques simples

3. Évolution des méthodes de résolution des problèmes d'optimisation :

Depuis l'apparition du problème de tournées de véhicules (VRP) dans les années 1950, les stratégies de résolution ont connu une évolution progressive, en lien avec les avancées théoriques et technologiques.

— Années 1950-1970 :

Les premières approches reposaient exclusivement sur des **méthodes exactes**, telles que la **programmation linéaire en nombres entiers** et les méthodes d'énumération (Branch and Bound, Branch and Cut). Ces méthodes garantissent l'optimalité mais sont rapidement devenues **inapplicables pour des instances de taille moyenne à grande**, en raison de la complexité combinatoire du VRP.

— Années 1980-1990 :

Face aux limites des méthodes exactes, les chercheurs ont développé des **heuristiques simples**, comme l'algorithme de **Clarke and Wright (1964)**, le **Sweep Algorithm**, ou encore les méthodes de **plus proche voisin**. Ces heuristiques, rapides à mettre en œuvre, ont permis d'obtenir des solutions acceptables pour des problèmes industriels, bien que non optimales.

— **Années 1990-2000 :**

Cette période marque l'**essor des métaheuristiques**, avec l'introduction de méthodes inspirées de la nature ou de principes physiques : **Recuit simulé, algorithmes génétiques, colonies de fourmis**, et **recherche tabou**. Ces approches ont offert un meilleur compromis entre qualité de solution et temps de calcul, et ont été largement adoptées pour résoudre des VRP complexes.

- **Depuis 2010 :** La tendance actuelle est à l'utilisation d'**approches hybrides**, combinant plusieurs stratégies pour tirer parti de leurs forces respectives. Par exemple, on peut intégrer une **heuristique constructive dans une métaheuristique**, ou encore **associer la recherche tabou à un algorithme génétique**. On observe également l'émergence de techniques issues de l'intelligence artificielle, telles que le machine learning, pour guider ou accélérer la recherche de solutions.

Le schéma ci-dessous illustre les principales méthodes de résolution utilisées pour le VRP, classés selon leur nature :

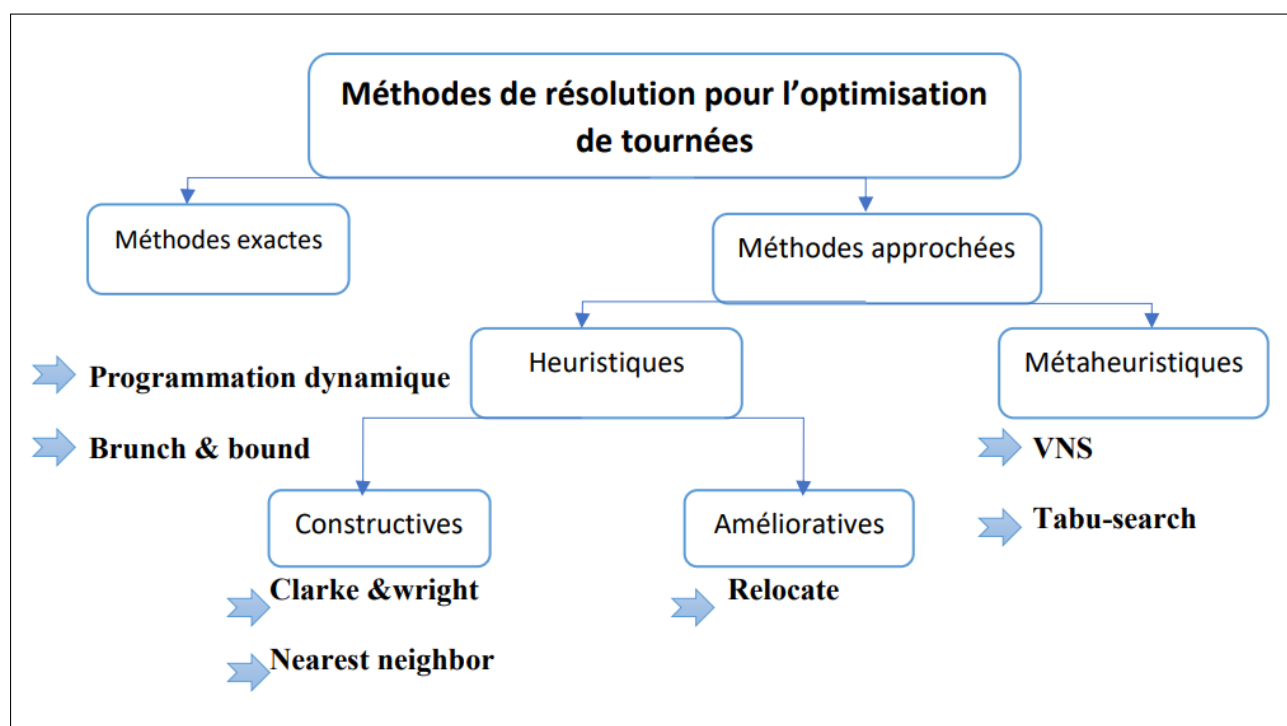


FIGURE 2.1 – Méthodes de résolution pour l'optimisation de tournées (source : réalisation personnelle)

IV. Complexité

La théorie de la complexité est l'étude formelle de la difficulté des problèmes décidables.

Elle désigne la quantité de ressources informatiques pour résoudre une tâche donnée .

Un problème est décidable s'il existe un algorithme, généralement une procédure qui se termine en un nombre fini d'étapes, qui répond par oui ou par non à la question posée par ce problème .

1. Complexité d'un algorithme

L'objectif de la théorie de la complexité est d'examiner les coûts des résolutions, notamment en termes de temps de calcul. Il cherche également à diviser les tâches en différents niveaux de difficulté.

En général, deux paramètres sont utilisés pour évaluer la complexité algorithmique : Le temps alloué à l'exécution de l'algorithme est proportionnel au nombre d'instructions exécutées et à la quantité de données traitées.

L'espace mémoire requis est lié à la taille de l'instance d'un problème.

2. Complexité des problèmes

la théorie de la complexité consiste de classer les problèmes de décision en deux classes importantes P (problème polynomial) et NP (Problème Non-déterministe Polynomial).

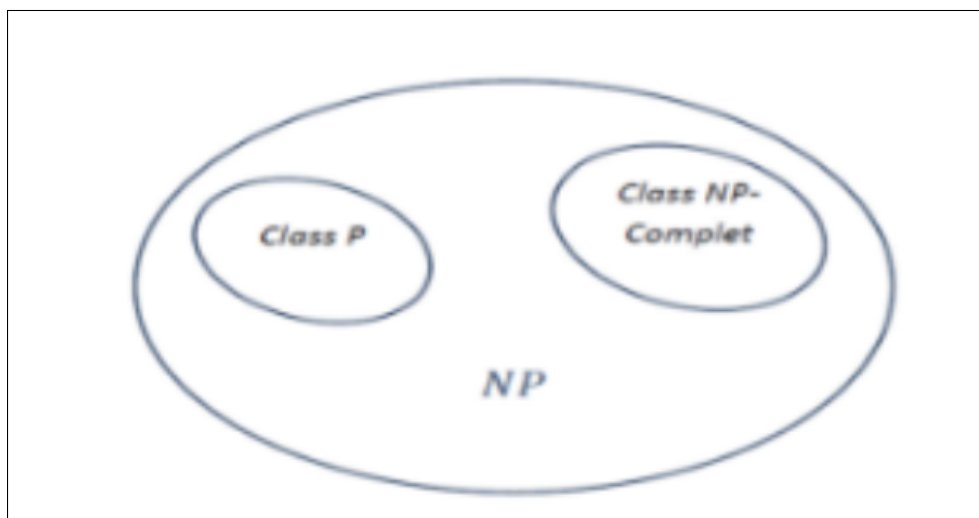


FIGURE 2.2 – Classification des complexités de problèmes.

- La classe des problèmes P est la classe de tous les problèmes qui peuvent être résolus par un algorithme déterministe en temps polynomial.
- La classe des problèmes NP est la classe de tous les problèmes qui peuvent être résolus par un algorithme non déterministe dans un temps polynomial. Il existe d'autres

classes de problèmes NP-complet et NP-difficile.

- Un problème est NP-Complet est résolu en un temps polynomial si et seulement si tous les problèmes NP-complet pouvant être résolu dans un temps polynomial .
- Un problème est NP-difficile s'il est plus difficile qu'un problème NP-complet, c'est à dire s'il existe un problème NP-complet se réduisant à ce problème. Les problèmes NP-difficile sont des problèmes d'optimisation .

V. Problèmes d'optimisation

L'optimisation occupe une place centrale en recherche opérationnelle, car elle permet d'améliorer la performance des systèmes en prenant des décisions éclairées. Elle se déroule généralement en deux grandes étapes. La première, souvent appelée "perception" en économie, comprend l'analyse de la situation ainsi que la modélisation du problème. La seconde étape vise à trouver la solution optimale parmi celles qui respectent les contraintes identifiées.

Les domaines d'application sont très variés, allant de la gestion des chaînes logistiques, à la répartition des ressources (comme dans les réseaux informatiques), en passant par la planification des emplois du temps, l'ordonnancement des tâches, ou encore le calcul de trajectoires. Cependant, dans bien des cas, le système est trop complexe pour être modélisé dans son ensemble. Le décideur doit alors se concentrer sur une partie du problème, ce qui conduit à une sous-optimisation susceptible d'impacter les autres sous-systèmes liés.

La construction d'un modèle mathématique est une étape clé pour accompagner le décideur dans son choix. Ce modèle repose sur la définition de variables de décision, la fixation des paramètres, et la formulation d'une fonction objectif à optimiser (minimiser ou maximiser), tout en respectant un ensemble de contraintes. Le rôle du spécialiste en optimisation ou du conseiller en recherche opérationnelle est d'élaborer ce modèle, puis d'aider à en interpréter les résultats et à en extraire des solutions pertinentes.

La qualité des solutions obtenues dépend en grande partie de la pertinence du modèle choisi et de l'efficacité des algorithmes utilisés pour résoudre le problème.

La formulation mathématique de n'importe quel problème d'optimisation P peut être décrite comme suit :

$$(P) \left\{ \begin{array}{ll} \text{s.c.} & (\min / \max) f(S) \quad (1) \\ & S \in \Omega \quad (2) \\ & S \subseteq \mathbb{R}^p / \mathbb{N}^p / E \quad (3) \end{array} \right. \quad (2.1)$$

L'équation (1) représente la fonction objectif f , à optimiser (minimiser ou maximiser)

selon le type de problème. L'ensemble défini par (2) correspond aux contraintes assurant la faisabilité des solutions. Enfin, (3) précise le domaine des variables de décision S , qui peuvent être continues, entières ou appartenir à un ensemble de valeurs discrètes E .

VI. Lien entre la complexité des problèmes et le choix des méthodes de résolution

Les problèmes d'optimisation combinatoire, tels que le problème de tournées de véhicules (VRP), sont bien souvent classés comme NP-difficiles, ce qui signifie qu'aucun algorithme connu ne permet de les résoudre de manière exacte dans un temps polynomial pour toutes les instances. En effet, à mesure que la taille du problème augmente (nombre de clients, de contraintes ou de véhicules), le nombre de combinaisons possibles explose de manière exponentielle, rendant les méthodes exactes impraticables au-delà d'un certain seuil.

Dans ce contexte, les méthodes exactes (comme la programmation linéaire en nombres entiers, le Branch and Bound, etc.) sont réservées aux petites instances ou utilisées comme référence pour évaluer des solutions approchées. En revanche, les méthodes approchées, notamment les heuristiques et métaheuristiques, sont largement utilisées dans la pratique pour leur capacité à fournir des solutions de bonne qualité en un temps raisonnable, même pour des instances de grande taille.

Ces méthodes ne garantissent pas toujours une solution optimale, mais elles offrent un compromis acceptable entre temps de calcul et qualité de la solution, ce qui est particulièrement important dans un cadre industriel ou opérationnel où le temps de réponse est limité. Par exemple, une entreprise de logistique ne peut pas se permettre d'attendre plusieurs heures pour obtenir un itinéraire optimal lorsque les livraisons doivent être planifiées en quelques minutes.

Ce constat justifie pleinement le recours aux approches heuristiques et métaheuristiques dans le cadre de notre étude, appliquée à un cas réel en Algérie impliquant un nombre significatif de clients et de contraintes logistiques.

VII. Vehicle Routing Problem (VRP) :

Le problème de tournée de véhicules (ou **VRP** pour Vehicle Routing Problem) est un problème d'optimisation complexe, dans lequel une flotte de véhicules part d'un dépôt central pour livrer un ensemble de clients situés à différents emplacements. Chaque client a des besoins spécifiques en matière de livraison, et l'objectif principal est d'organiser les itinéraires de manière optimale afin de minimiser les coûts totaux. Plusieurs contraintes doivent être prises en compte dans cette planification, notamment

la capacité des véhicules, la consommation de carburant, ou encore les conditions de circulation. Tous ces facteurs influencent directement l'efficacité des trajets et, par conséquent, les coûts globaux.

Un exemple simplifié du Problème de Tournée de Véhicules (VRP) est le Problème du Voyageur de Commerce (PVC), également connu sous son appellation anglaise Traveling Salesman Problem (TSP). Dans ce cas, on considère un seul véhicule disposant d'une capacité illimitée. l'objectif du PVC est de déterminer le chemin le plus court possible entre les destinations, sans avoir à passer par le même chemin deux fois.

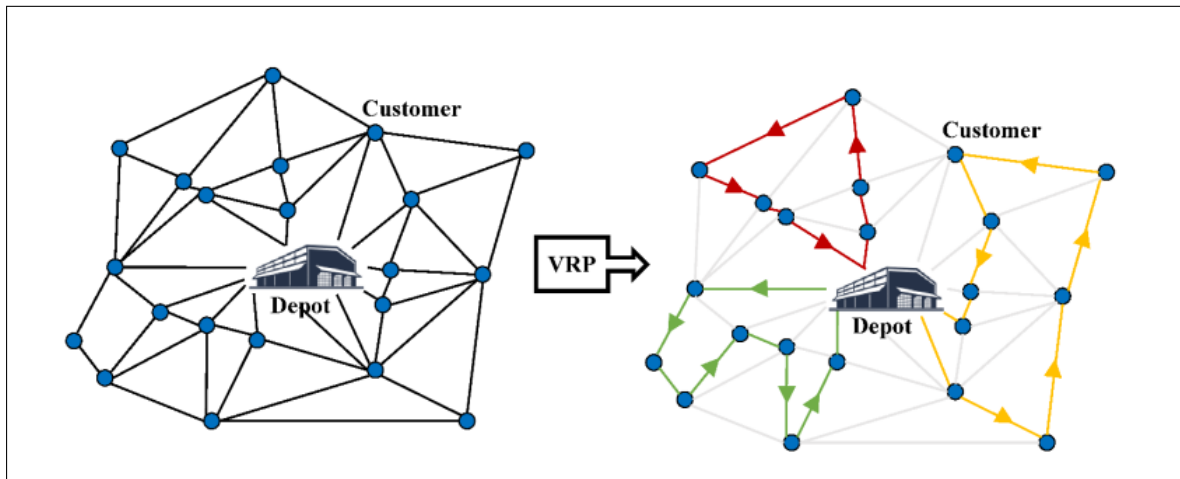


FIGURE 2.3 – Vehicle Routing Problem

1. Les paramètres du problème de tournée de véhicules (VRP) :

Le problème de tournée de véhicules (VRP) prend en compte plusieurs contraintes, comme la capacité des véhicules, les temps de livraison, ou encore les coûts de déplacement. En raison de sa complexité, il est souvent résolu à l'aide d'algorithmes d'optimisation. Les principaux paramètres du VRP sont :

- * **Nombre de véhicules disponibles** : il s'agit du nombre total de véhicules pouvant être utilisés pour effectuer les tournées de livraison.
- * **Capacité des véhicules** : chaque véhicule a une limite en termes de volume ou de poids de marchandises qu'il peut transporter. Cette contrainte est essentielle dans les variantes comme le Capacitated VRP (CVRP).
- * **Coûts de déplacement** : les coûts associés aux trajets entre les clients (ou points de livraison) dépendent généralement de la distance, du temps de parcours, ou encore de la consommation énergétique.
- * **Temps de service (livraison ou ramassage)** : chaque point de livraison peut nécessiter un temps spécifique pour charger ou décharger les marchandises, ce qui doit être pris en compte dans l'optimisation.

- * **Contraintes de temps et de distance** : certaines variantes du VRP incluent des fenêtres temporelles pendant lesquelles les clients doivent être servis, ou des limites maximales sur la distance ou le temps total de chaque tournée.

2. Variante de VRP :

Nous distinguons plusieurs variants du problème, nous citons :

2..1 VRP avec profit (VRPP) :

Dans le VRP avec profit, chaque client génère un gain. L'objectif est de maximiser le profit total tout en respectant une contrainte de temps pour chaque véhicule.

2..2 CVRP (VRP avec contrainte de capacité)

Chaque véhicule a une capacité limitée. L'objectif est d'optimiser les tournées tout en respectant cette contrainte de chargement.

2..3 VRP dynamique :

La demande des clients change en temps réel. Elle n'est pas totalement connue au départ, ce qui rend la planification plus complexe et nécessite des ajustements durant la distribution.

2..4 VRP avec fenêtre de temps "Time Window" (VRPTW) :

Dans le VRPTW, chaque client a une plage horaire précise pour être livré. Le véhicule doit organiser sa tournée en respectant ces contraintes de temps.

2..5 VRP avec contrainte LIFO :

Cette variante impose une contrainte de chargement basée sur le principe du « Last In, First Out » (LIFO), ce qui signifie que les articles ramassés en dernier doivent être les premiers à être livrés ou déchargés. Cette règle est essentielle dans les contextes où les véhicules ont un espace de stockage limité, et où l'ordre des chargements affecte directement l'efficacité des opérations de collecte et de livraison, notamment en logistique inverse.

2..6 VRP avec livraison et ramassage (VRPPD) :

Ce problème consiste à concevoir des itinéraires optimaux permettant à des véhicules de desservir un ensemble de clients pour effectuer à la fois des livraisons et des collectes.

Cette variante est particulièrement adaptée à la logistique inverse, où il s'agit de

recupérer des produits ou matériaux usagés tout en assurant la livraison simultanée de marchandises.

2..7 Simultaneous VRP :

Le problème de tournées de véhicules consiste à planifier des itinéraires pour plusieurs véhicules partant d'un dépôt, afin de desservir un ensemble de clients, en minimisant un critère (coût, distance, temps, etc.). Le Simultaneous VRP ajoute la contrainte que les tournées doivent être réalisées simultanément (par exemple, tous les véhicules doivent démarrer et finir leur tournée à peu près en même temps).

VIII. Les objectifs classiques en optimisation de tournées :

Dans les problèmes de tournées de véhicule (VRP) ,plusieurs objectifs classiques sont considérés afin d'optimiser la performance logistique .Le tableau suivant résume les plus courants :

TABLE 2.1 – Les objectifs classiques en optimisation de tournées.

Objectif	Description	Intention principale
Minimiser la distance totale	Réduire la somme des distances parcourues par tous les véhicules	Faire des économies de carburant
Minimiser le coût total	Réduire la somme des coûts (distance, nombre de véhicule, temps)	Optimiser économiquement
Minimiser le nombre de véhicules	Utiliser le moins de véhicules possible	Réduire les ressources
Minimiser le makespan	Réduire la durée de la tournée la plus longue	Équilibrer la charge de travail

Conclusion :

Ce chapitre a exploré les défis d'optimisation spécifiques à la logistique inverse dans le transport. L'importance de la recherche opérationnelle et de la modélisation mathématique pour structurer ces opérations complexes a été soulignée. L'analyse des

tournées de récupération a montré la nécessité d'une planification efficace pour maîtriser coûts et impact environnemental. Les modèles VRP, CVRP et VRPSDP offrent un cadre robuste pour cela, essentiel pour améliorer la performance globale.

Chapitre 3

Formulation et Stratégie de Résolution du VRPSPD-TL

Introduction

:

Dans ce chapitre, nous abordons la modélisation mathématique et la résolution du problème de tournées de véhicules avec ramassage et livraison simultanés (Simultaneous Pickup and Delivery VRP - SPDVRP), intégrant des contraintes de temps ainsi que la minimisation du makespan (durée totale maximale des tournées), dans le cadre des activités logistiques de l'entreprise Condor Logistique.

Nous commençons par définir précisément les données du problème ainsi que sa formulation mathématique. Cette étape inclut la présentation des hypothèses de modélisation, suivie par la structuration du problème à travers les ensembles, paramètres, variables de décision, contraintes, et la fonction objective.

Concernant la résolution, nous avons adopté une démarche en plusieurs étapes :

- Obtention d'une solution initiale à l'aide de l'algorithme heuristique de Clarke & Wright Savings, qui permet une première construction efficace des tournées en regroupant les clients selon une logique d'économie de temps de trajet.
- Amélioration de la solution obtenue par une heuristique de recherche locale, qui explore le voisinage de la solution initiale en modifiant progressivement l'organisation des tournées afin de réduire davantage le **makespan** et équilibrer les charges.
- Optimisation avancée par l'application d'une méta-heuristique appelée recherche tabou, qui permet d'éviter les pièges des optima locaux en interdisant temporairement certaines solutions déjà explorées. Cette approche favorise une exploration plus large et intelligente de l'espace des solutions possibles.

Ainsi, ce chapitre expose à la fois la modélisation théorique du problème et les étapes pratiques de sa résolution, en s'appuyant sur des techniques d'optimisation combinatoire adaptées à la complexité des contraintes réelles rencontrées chez Condor Logistique.

I. Enoncé du problème et modèle mathématique :

1. Description du problème :

Le **VRPSPD-TL** (Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Limit) a été introduit en 1989 et son application aux situations réelles est aujourd'hui des plus répandues et des plus utiles. Il consiste à planifier des tournées pour un nombre limité de véhicules afin de desservir un ensemble de clients ayant à la fois des quantités à livrer et des quantités à collecter tout en respectant une limite maximale de temps par tournée.

Chaque véhicule doit partir et revenir au dépôt central, en assurant simultanément les opérations de livraison et de collecte pour chaque client visité. Les véhicules sont supposés homogènes, de capacité limitée, et peuvent être utilisés à condition que leur durée de tournée ne dépasse pas un seuil fixé T_{max} .

2. Composants du problème :

- **Clients** : Chaque client i a deux demandes : une livraison d_i et une collecte p_i ,
- **Véhicules** : Un nombre fixe m de véhicules identiques, chacun ayant une capacité maximale Q . La charge transportée à tout moment (livraisons effectuées + collectes embarquées) ne doit pas dépasser la capacité maximale Q
 - Chaque véhicule a une capacité maximale Q (prise en compte à chaque instant comme la somme cumulée des livraisons effectuées et des collectes embarquées).
 - Chaque tournée doit débuter et finir au dépôt
 - Chaque client doit être visité une seule fois, sans fractionnement de demande avec une demande de livraison d_i et demande de ramassage p_i
 - La durée totale de chaque tournée ne doit pas excéder une limite temporelle T_{max}
 - Le service est considéré immédiat à l'arrivée (aucun coût d'attente n'est pris en compte en cas d'arrivée anticipée).

3. Objectif :

L'objectif est de **minimiser le makespan**, c'est-à-dire la durée maximale parmi toutes les tournées effectuées par les véhicules. Cela permet d'assurer un meilleur équilibrage de la charge de travail entre les conducteurs, en évitant qu'un seul véhicule ne supporte une tournée beaucoup plus longue que les autres.

Ce problème reflète des situations courantes dans la logistique inverse, la gestion des retours, où les véhicules doivent livrer et collecter en même temps, sous des contraintes opérationnelles strictes.

II. Le modèle mathématique :

Le modèle mathématique du problème de tournées de véhicules avec ramassage et livraison simultanés (VRPSPD) étudié au sein de l'entreprise Condor Logistique est formulé dans le but d'optimiser le makespan, c'est-à-dire la durée maximale des tournées effectuées par les véhicules, tout en respectant plusieurs contraintes opérationnelles, notamment la collecte et la livraison simultanées chez chaque client ainsi que la limite de 10 heures imposée par tournée.

Le problème considéré est une extension du modèle classique du Vehicle Routing Problem (VRP), avec les spécificités suivantes :

- Chaque client a simultanément une demande de livraison et une demande de collecte (cas typique de la logistique inverse).
- Chaque tournée est limitée à 10 heures, ce qui correspond à une journée de travail.
- L'objectif est de minimiser le temps maximal utilisé par un véhicule (objectif min-max).

Ce modèle est particulièrement pertinent dans le contexte de la logistique inverse, où les véhicules doivent livrer des produits tout en récupérant des retours, des déchets ou des emballages vides à chaque point de passage. Il permet ainsi de représenter de façon réaliste le système logistique de Condor Logistique, en tenant compte de la capacité fixe des véhicules (20 tonnes), des contraintes temporelles, et de la structure simultanée des flux aller-retour.

Enfin, il convient de noter que ce problème est de nature combinatoire et NP-difficile, ce qui signifie qu'il devient rapidement complexe à résoudre exactement lorsque le nombre de clients augmente. La formulation mathématique proposée constitue donc un cadre rigoureux pour modéliser, analyser et optimiser les opérations logistiques quotidiennes de l'entreprise, tout en évaluant la performance des différentes solutions mises en œuvre.

Les caractéristiques spécifiques du problème étudié sont les suivantes :

- La flotte est composée de 17 camions homogènes, chacun ayant une capacité maximale de 20 tonnes.
- Chaque tournée doit être effectuée dans une limite de temps de 10 heures, correspondant à une journée de travail standard.
- Pour des raisons de simplification, les temps de chargement et de déchargement sont considérés comme négligeables devant les temps de transport.
- Conformément aux règles sociales internes de l'entreprise, chaque chauffeur bénéficie d'une journée de repos après avoir effectué une tournée complète.

1. Formulation

mathématique

Nous présentons d'abord un modèle pour le VRPSDP, puis montrons comment il peut être modifié pour modéliser les autres classes de problèmes. Introduisons la notation suivante.

1.1 Indices

et

paramètres

- $N = \{1, \dots, n\}$: clients
- $N_0 = \{0\} \cup N$: nœuds, avec 0 représentant le dépôt
- $K = \{1, \dots, 17\}$: véhicules
- $Q = 20$: capacité des véhicules
- d_i, p_i : demande de livraison et de collecte du client i
- t_{ij} : temps de trajet de i à j
- $T_{\max} = 10$: durée maximale autorisée

1.2 Variables

de

décision

- $x_{ijk} \in \{0, 1\}$: 1 si le véhicule k parcourt l'arc (i, j)
- $z_{ij} \geq 0$: quantité à livrer transportée de i à j
- $u_{ij} \geq 0$: quantité à collecter transportée de i à j
- $Z \geq 0$: temps maximal d'un véhicule.

Le VRPSDP peut être modélisé comme suit :

Objectif

Minimiser le temps maximal parmi les tournées effectuées :

$$\min Z$$

Contraintes

$$\sum_{j \in N} x_{0jk} = 1 \quad \forall k \in K \quad (1)$$

$$\sum_{j \in N} x_{i0k} = 1 \quad \forall k \in K \quad (2)$$

$$\sum_{i \in N_0} \sum_{j \in N_{+k}} x_{ijk} = 1 \quad \forall j \in N \quad (3)$$

$$\sum_{j \in N_0, j \neq i} x_{ijk} = \sum_{j \in N_0, j \neq i} x_{jik} \quad \forall i \in N, \forall k \in K \quad (4)$$

$$z_{ij} + u_{ij} \leq Q \cdot \sum_{k \in K} x_{ijk} \quad \forall i, j \in N_0, i \neq j \quad (5)$$

$$\sum_{j \in N_0, j \neq i} z_{ji} - \sum_{j \in N_0, j \neq i} z_{ij} = d_i \quad \forall i \in N \quad (6)$$

$$\sum_{j \in N_0, j \neq i} u_{ij} - \sum_{j \in N_0, j \neq i} u_{ji} = p_i \quad \forall i \in N \quad (7)$$

$$d_j \cdot \sum_k x_{ijk} \leq z_{ij} \leq (Q - d_i) \cdot \sum_k x_{ijk} \quad \forall (i, j) \in A \quad (8a)$$

$$p_i \cdot \sum_k x_{ijk} \leq u_{ij} \leq (Q - p_j) \cdot \sum_k x_{ijk} \quad \forall (i, j) \in A \quad (8b)$$

$$t_{0j} = 0 \quad \forall j \in N, \quad z_{i0} = 0 \quad \forall i \in N \quad (9)$$

$$\sum_{(i,j) \in A} t_{ij} \cdot x_{ijk} \leq T_{\max} \quad \forall k \in K \quad (10)$$

$$\sum_{(i,j) \in A} t_{ij} \cdot x_{ijk} \leq Z \quad \forall k \in K \quad (11)$$

2. Explication des contraintes du modèle mathématique

Le modèle mathématique du **VRPSPD avec contrainte de durée et objectif makespan** intègre plusieurs contraintes qui traduisent fidèlement les réalités opérationnelles du système logistique de Condor Logistique. Voici une explication détaillée de chaque contrainte :

- Contraintes (1) et (2) : Chaque véhicule doit partir du dépôt et y revenir exactement une fois. Cela garantit la fermeture des tournées et la traçabilité des véhicules.
- Contrainte (3) : Chaque client doit être visité exactement une fois, ce qui assure une planification complète des demandes de livraison et de collecte.
- Contrainte (4) : Si un véhicule entre chez un client, il doit également en sortir, garantissant ainsi la continuité du trajet sans interruption.
- Contrainte (5) : À tout moment, la somme des quantités transportées (livraisons à effectuer + collectes en cours) ne doit pas dépasser la capacité maximale du véhicule, fixée ici à 20 tonnes.

- Contrainte (6) : Les flux de livraison sont équilibrés entre les entrées et les sorties de chaque nœud (client), assurant une gestion cohérente des quantités livrées.
- Contrainte (7) : De même, les flux de collecte sont équilibrés à chaque nœud, garantissant que la quantité collectée correspond à ce qui est prévu.
- Contraintes (8a–8b) : Ces contraintes imposent des bornes sur les quantités transportées uniquement sur les arcs effectivement parcourus, pour éviter toute surévaluation de la charge sur des trajets non utilisés.
- Contrainte (9) : Lors du départ du dépôt, chaque véhicule est à vide, sans résidu de collecte ni de livraison, ce qui reflète une situation initiale réaliste.
- Contrainte (10) : La durée totale de chaque tournée ne doit pas dépasser 10 heures, ce qui correspond à une journée de travail réglementaire pour les chauffeurs.
- Contrainte (11) : Le temps utilisé par chaque tournée est borné par une variable Z , que l'on cherche à minimiser. Cette contrainte traduit l'objectif de makespan du modèle.

3. Remarque sur l'objectif

L'objectif principal du modèle est de **minimiser le temps maximal (makespan)** parmi toutes les tournées effectuées. Ce critère est particulièrement pertinent dans des contextes logistiques où l'on souhaite :

- Équilibrer la charge de travail entre les véhicules,
- Garantir des retours rapides au dépôt,
- Ou réduire l'impact opérationnel en logistique inverse, urbaine ou de maintenance/-recyclage.

Ce type d'optimisation permet une meilleure utilisation des ressources, une réduction du surmenage des conducteurs, et une planification plus robuste face aux aléas du terrain.

III. Approche de résolution :

1. Méthode exacte :

nous avons d'abord adopté une approche exacte pour résoudre le problème VRPSPD en utilisant :

1. Modélisation mathématique :

- Formulation du problème comme un programme linéaire mixte en nombres entiers (MILP)
- Définition des variables de décision binaires (affectation des arcs aux véhicules)

- Ajout des variables continues pour gérer les flux de livraison et collecte
- Intégration des contraintes spécifiques au VRPSPD

2. Implémentation technique :

- Utilisation de Pyomo comme interface de modélisation
- Appel du solveur Gurobi pour la résolution
- Paramétrage des options du solveur (temps limite..)

3. Analyse des résultats :

- Vérification de la faisabilité des solutions obtenues
- Contrôle du respect de toutes les contraintes
- Calcul des indicateurs de performance (makespan, charge des véhicules).

Cette approche garantit théoriquement l'obtention de la solution optimale, mais s'est révélée impraticable pour les instances de taille réelle en raison du temps de calcul exponentiel requis. Cette limitation intrinsèque nous a conduits à développer des méthodes approchées plus adaptées aux problèmes de grande taille.

2. Les heuristiques :

2.1 Construction Initiale : Clarke Wright Savings (CWS) :

Cette méthode permet de construire une solution initiale réalisable à partir de tournées individuelles. Elle repose sur un principe simple : fusionner les routes afin de maximiser les économies de temps.

Principe : Chaque client est d'abord affecté à une tournée individuelle partant du dépôt. Ensuite, pour chaque paire de clients, on calcule un gain (**savings**) représentant la réduction de coût obtenue en fusionnant leurs tournées. Les tournées sont alors combinées de manière itérative en sélectionnant les fusions offrant les économies les plus importantes, tout en respectant les contraintes de capacité des véhicules et de temps.

Le « savings » entre deux clients i et j est calculé comme suit :

$$S_{ij} = t_{i0} + t_{0j} - t_{ij}$$

où t_{i0} et t_{0j} représentent les coûts de retour au dépôt depuis i et j , et t_{ij} est le coût de déplacement direct entre les deux clients.

Les clients sont ensuite regroupés progressivement en suivant l'ordre décroissant des « savings », formant ainsi des tournées optimisées. Chaque client tant initialement affecté une tournée individuelle, les regroupements se font uniquement par fusion successive de ces tournées, dans le respect des contraintes. L'objectif de cette approche est de minimiser le temps total parcouru tout en garantissant la faisabilité de la solution.

Algorithm 1 Clarke & Wright Savings Algorithm

```

1: function CLARKEWRIGHTSAVINGS(dpt, clients, capacit_vhicule)
2:   // Étape 1 : Initialisation - Chaque client a sa propre tournée
3:   for all client  $\in$  clients do
4:     tourne  $\leftarrow$  [dpt, client, dpt]
5:     Ajouter tourne à liste_tournes
6:   end for
7:   // Étape 2 : Calcul des économies (savings)
8:   liste_savings  $\leftarrow$  []
9:   for all paires (i, j) telles que  $i \neq j$  et  $i, j \in$  clients do
10:    saving  $\leftarrow$  cot(dpt, i) + cot(dpt, j) - cot(i, j)
11:    Ajouter (i, j, saving) à liste_savings
12:  end for
13:  // Étape 3 : Trier les savings par ordre décroissant
14:  Trier liste_savings selon saving décroissant
15:  // Étape 4 : Fusionner les tournées
16:  for all (i, j, saving)  $\in$  liste_savings do
17:    tournei  $\leftarrow$  TROUVERTOURNÉECONTENANT(i, liste_tournes)
18:    tournej  $\leftarrow$  TROUVERTOURNÉECONTENANT(j, liste_tournes)
19:    if tournei  $\neq$  tournej et demande(tournei) + demande(tournej)  $\leq$ 
       capacit_vhicule then
20:      nouvelle_tourne  $\leftarrow$  FUSIONNER(tournei, tournej, i, j)
21:      Supprimer tournei, tournej de liste_tournes
22:      Ajouter nouvelle_tourne à liste_tournes
23:    end if
24:  end for
25:  return liste_tournes
26: end function
27: function FUSIONNER(tourne1, tourne2, i, j)
28:  if i est le dernier client de tourne1 avant le dépôt et j est le premier client de
    tourne2 après le dépôt then
29:    nouvelle_tourne  $\leftarrow$  tourne1[0 : -1] + tourne2[1 :]
30:  else if j est le dernier client de tourne1 avant le dépôt et i est le premier client
    de tourne2 après le dépôt then
31:    nouvelle_tourne  $\leftarrow$  tourne2[0 : -1] + tourne1[1 :]
32:  end if
33:  return nouvelle_tourne
34: end function

```

3. La recherche locale :

La recherche locale est appliquée tant qu'il est possible d'améliorer la solution, chaque tournée étant scrupuleusement inspectée. Avant l'application d'un mouvement, l'algorithme vérifie sa faisabilité selon ces critères. Le **critère d'acceptation** est strict : une solution améliorée n'est retenue que si le makespan diminue à celui de la solution courante de référence. Lorsqu'un mouvement est accepté, la solution ainsi générée devient le nouveau point de départ pour l'itération suivante de la recherche.

Parmi les stratégies employées, une méthode particulière consiste à :

1. **Identifier la tournée la plus longue** (celle qui détermine le makespan),
2. **Sélectionner un client de cette tournée** et le déplacer vers une tournée moins chargée ou vide,
3. Accepter le déplacement uniquement s'il réduit le makespan et respecte toutes les contraintes (capacité, couverture de la demande).

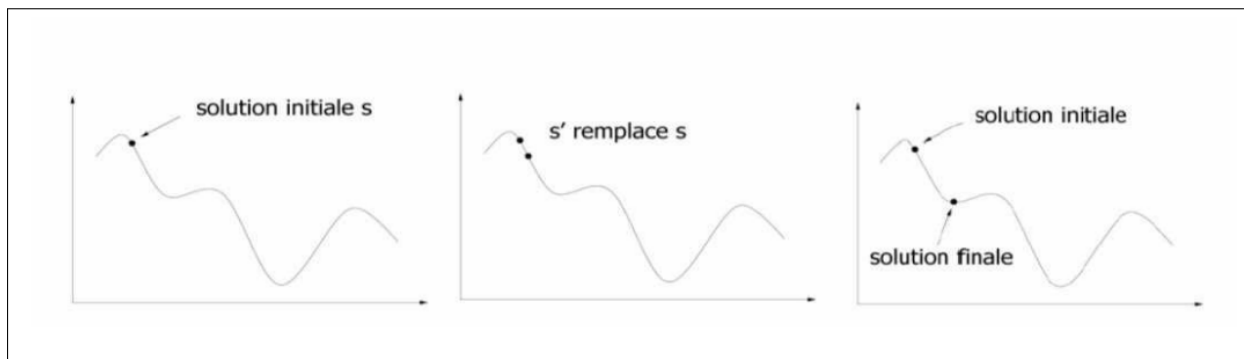


FIGURE 3.1 – Le recherche Locale par Déplacement de Solution
(Douiri et al .cours des méthodes de résolution) .

Cette approche contribue à équilibrer les charges entre véhicules et à réduire la durée maximale des tournées.

Algorithm 2 Recherche locale par déplacement (Relocate)

```

1: Complexité :  $O(mn^2)$ 
2: Input : Solution Routes, paramètres  $Q, T_{\max}, c$ 
3: Output : Solution améliorée Routes
4: while amélioration possible do
5:   Identifier la tournée  $R_{\max}$  ayant la durée maximale  $Z$ 
6:   for all client  $i \in R_{\max}$  do
7:     for all autre tournée  $R_k$  do
8:       for all position possible dans  $R_k$  do
9:         Simuler insertion de  $i$  dans  $R_k$  et retrait de  $R_{\max}$ 
10:        if capacitaire et temporellement faisable et makespan diminue then
11:          Appliquer le déplacement
12:          Recommencer la boucle
13:        end if
14:      end for
15:    end for
16:  end for
17: end while

```

4. Les métaheuristiques :

4.1 Optimisation avancée : Tabu Search

La Tabu Search est une métaheuristique développée par Fred W. Glover en 1977, conçue pour échapper aux optima locaux et améliorer progressivement une solution initiale à travers une recherche locale intelligente, tout en mémorisant les mouvements récents afin d'éviter les cycles. Contrairement à une simple recherche locale qui peut rapidement se bloquer, la Tabu Search permet, sous certaines conditions, d'accepter temporairement des solutions non améliorantes, dans le but d'explorer des régions inaccessibles autrement.

Dans le contexte du Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Limit (VRPSPD-TL), cette méthode est particulièrement pertinente, car elle permet de réduire le makespan (durée maximale des tournées) même lorsque des heuristiques classiques comme Relocate cessent d'apporter des améliorations. Elle s'avère ainsi plus flexible et robuste que les heuristiques purement locales.

Composants clés de la Tabu Search :

- **Solution courante** : point de départ pour générer les voisins.
- **Liste taboue** : mémoire à court terme interdisant certains mouvements récemment effectués.
- **Critère d'aspiration** : permet d'outrepasser l'interdiction taboue si le mouvement mène à une amélioration significative.

- **Critères d'arrêt** : peuvent être un nombre maximum d'itérations, un temps de calcul écoulé ou un nombre d'itérations sans amélioration.

Fonctionnement **général** :

La recherche taboue fonctionne par exploration itérative de l'espace des solutions. À chaque itération :

- Point de départ : une solution initiale réalisable est générée (la solution après une amélioration par relocate).
- Exploration du voisinage : à chaque itération, l'algorithme génère un ensemble de solutions voisines en appliquant un mouvement (dans notre cas : le déplacement d'un client entre deux tournées).
- Évaluation et choix : parmi les solutions voisines, on sélectionne la meilleure (celle qui donne le makespan le plus faible), même si elle est moins bonne que la solution actuelle.
- Liste taboue : pour éviter de revenir en arrière, les derniers mouvements effectués sont stockés dans une "liste taboue" pour quelques itérations.
- Critère d'aspiration : un mouvement tabou peut être accepté s'il conduit à une solution meilleure que toutes celles rencontrées jusqu'à présent.
- Mise à jour : si une meilleure solution est trouvée, elle est sauvegardée comme solution globale. Sinon, l'algorithme continue avec la nouvelle solution courante.
- Arrêt : l'algorithme s'arrête lorsqu'il atteint un nombre maximal d'itérations ou lorsqu'aucune amélioration n'est observée pendant un certain temps.

L'algorithme suivant décrit le fonctionnement de la Tabu Search appliquée au
VRPSPD-TL :

Algorithm 3 Algorithme Tabou pour l'amélioration de tournées

Require: Solution initiale *Routes*, données Q , T_{max} , c , paramètres *tabu_tenure*, *max_iter*

Ensure: Solution améliorée *Routes*

```
1: solution_courante  $\leftarrow$  relocate(Routes)
2: meilleure_solution  $\leftarrow$  solution_courante
3: Initialiser liste_tabou comme file de taille tabu_tenure
4: for  $it \leftarrow 1$  to max_iter do
5:   meilleur_voisin  $\leftarrow$  null
6:   for all tournée source  $R$  do
7:     for all client  $e \in R$  do
8:       for all tournée cible  $R' \neq R$  do
9:         for all position  $p$  dans  $R'$  do
10:          Simuler déplacement de  $e$  vers  $R'$  à la position  $p$ 
11:          if déplacement faisable (capacité et temps) et (non tabou ou critère
d'aspiration vérifié) then
12:            Calculer makespan de la solution modifiée
13:            if makespan amélioré ou meilleur voisin courant then
14:              Enregistrer comme meilleur voisin admissible
15:            end if
16:          end if
17:        end for
18:      end for
19:    end for
20:  end for
21:  Appliquer le meilleur mouvement admissible
22:  Mettre à jour liste_tabou avec le mouvement effectué
23:  if solution améliorée then
24:    Mettre à jour meilleure_solution
25:    Réinitialiser compteur de stagnation
26:  else
27:    Incrémenter compteur de stagnation
28:    if stagnation prolongée then
29:      break
30:    end if
31:  end if
32: end for
```

Conclusion

:

Ce chapitre a présenté une modélisation rigoureuse du problème de tournées de véhicules avec ramassage et livraison simultanés, intégrant des contraintes de capacité, de temps, et l'objectif de minimisation du makespan. Ce cadre théorique a été formulé dans un contexte réaliste inspiré des opérations logistiques de l'entreprise Condor Logistique. La formalisation mathématique a structuré le problème à travers la définition précise des ensembles, paramètres, variables de décision, contraintes et fonction objective, constituant ainsi la base nécessaire à son traitement algorithmique. Plusieurs approches de résolution ont ensuite été explorées. Si les méthodes exactes conviennent aux petites instances, des techniques plus adaptées aux cas réels ont été développées : l'heuristique de Clarke & Wright pour la construction initiale, une amélioration par recherche locale, et une optimisation avancée par la métaheuristique Tabu Search. L'ensemble de ces méthodes offre une démarche pragmatique, équilibrée entre rigueur mathématique et efficacité computationnelle, que nous évaluons dans le chapitre suivant à travers des expérimentations sur des données inspirées du terrain.

Chapitre 4

Implémentation et Analyse des Résultats

Introduction

:

Le Problème de Routage de Véhicules (VRP – Vehicle Routing Problem) constitue l'un des défis majeurs en recherche opérationnelle et en logistique, notamment en raison de sa complexité combinatoire croissante avec la taille du problème. Il s'agit d'optimiser l'ensemble des tournées effectuées par une flotte de véhicules, partant d'un dépôt central pour desservir un ensemble de clients tout en minimisant un ou plusieurs critères, tels que la distance totale, le coût, le temps de transport ou le makespan dans notre cas tout en respectant des contraintes opérationnelles.

Afin de résoudre efficacement ce problème, nous avons adopté une approche mixte, combinant à la fois des méthodes exactes et des méthodes approchées. Plus précisément :

- **Pour les petites instances**, nous avons opté pour une méthode exacte, garantissant l'obtention d'une solution optimale. Cette méthode a été modélisée via la bibliothèque Pyomo en langage Python, et résolue à l'aide du solveur **Gurobi**, reconnu pour sa puissance dans le traitement des problèmes d'optimisation linéaire et mixte.
- **Pour les grandes instances**, dont la résolution exacte devient impraticable en raison du temps de calcul élevé, nous avons mis en œuvre une approche approchée en trois étapes successives :
 1. **CWS (Clarke & Wright Savings)** : introduite par Clarke and wright en (1964), une heuristique constructive permettant de générer rapidement une solution initiale en regroupant intelligemment les clients.
 2. **Relocate** : une heuristique d'amélioration locale consistant à déplacer des clients d'une tournée à une autre dans le but de réduire le coût total.

3. **Tabu Search** : une métaheuristique avancée, introduite par Fred Glover en 1986, visant à explorer efficacement l'espace de solutions en évitant les cycles grâce à une mémoire taboue.

I. Les outils Informatiques :

1. Python :

Pour l'implémentation des algorithmes de résolution du problème de routage de véhicules avec livraisons et collectes simultanées (VRPSPD-TL), nous avons opté pour le langage de programmation Python. Ce choix s'explique par la souplesse et la facilité de manipulation des données qu'offre ce langage. Créé en 1991, Python s'est rapidement imposé comme un outil incontournable dans des domaines exigeants tels que le Machine Learning, le Big Data ou encore la Data Science. Sa syntaxe claire, sa large communauté ainsi que la richesse de ses bibliothèques spécialisées (comme NetworkX, Gurobi, NumPy ou Matplotlib) en font un environnement de développement idéal pour modéliser et résoudre des problèmes complexes de logistique comme le VRPSPD-TL.

Grâce à ces atouts, nous avons pu coder, visualiser et optimiser efficacement les différentes solutions proposées dans ce travail.



FIGURE 4.1 – Logo du Python

1.1.1 Avantage de Python :

Pour le développement et la résolution de notre problématique, nous avons choisi d'utiliser le langage Python en raison de ses nombreux atouts. Il s'agit d'un langage open source, accessible gratuitement, qui permet de concevoir, modifier et partager des applications de manière simple et efficace. Sa syntaxe claire et lisible facilite l'écriture du

code, même pour des tâches complexes, ce qui le rend particulièrement adapté aux projets d'optimisation.

Python prend également en charge la programmation orientée objet et propose une grande variété de types de données, permettant une structuration souple et puissante des informations. De plus, il dispose d'une documentation complète, bien organisée et facilement compréhensible, ce qui représente un véritable avantage pour les développeurs.

Enfin, Python est doté de nombreuses bibliothèques, telles que NumPy, Pandas et Pyomo qui jouent un rôle essentiel dans la manipulation et la représentation des données. Ces outils permettent de traiter un grand volume d'instances de manière flexible et efficace, ce qui est particulièrement précieux dans le cadre des problèmes d'optimisation combinatoire.

2. Pyomo

:

Pyomo est une bibliothèque open-source puissante qui permet de formuler et de résoudre des modèles d'optimisation directement en Python. Elle offre une interface intuitive et rigoureuse, combinant formalisme mathématique et syntaxe accessible pour les programmeurs Python. Pyomo permet de modéliser différents types de problèmes d'optimisation, y compris ceux relevant de la programmation linéaire, entière et non linéaire. Dans le cadre de notre étude sur le problème de routage de véhicules avec livraisons et collectes simultanées (VRPSPD-TL), cette bibliothèque s'est révélée particulièrement efficace.

En effet, Pyomo permet d'intégrer des solveurs performants comme Gurobi, ce qui facilite la résolution de modèles complexes en garantissant des temps de calcul raisonnables et des résultats de qualité. Ainsi, nous avons utilisé Pyomo en combinaison avec Gurobi pour modéliser et résoudre efficacement notre problème de VRP dans l'environnement Python.



FIGURE 4.2 – Logo du Pyomo

II. Contexte Réel d'Application : Réseau Logistique de Condor :

L'entreprise dispose de plusieurs clients répartis sur l'ensemble du territoire algérien, que l'on peut regrouper en quatre grandes régions : l'Est, le Centre, l'Ouest et le Sud. Chaque région présente des particularités géographiques et logistiques qui influencent directement la stratégie de distribution adoptée.

Dans notre cas d'étude, chaque client est concerné par une opération logistique bidirectionnelle, c'est-à-dire qu'il fait l'objet à la fois d'une livraison (produits à envoyer) et d'une collecte (retours ou produits à récupérer), ce qui rend la problématique plus complexe et justifie l'emploi d'un modèle VRP avec livraisons et ramassages simultanés (VRPSPD).

De plus, chaque tournée de véhicule doit impérativement respecter une contrainte de temps limite, imposée par les conditions opérationnelles et les exigences de service, ce qui ajoute une dimension temporelle cruciale à la planification des tournées.

III. Méthodologie de Résolution :

Afin de répondre à la complexité du problème logistique rencontré par Condor Logistique, nous avons adopté une approche progressive et structurée.

Nous avons dans un premier temps testé une formulation mathématique exacte sur de petites instances, afin de valider la cohérence du modèle et vérifier que les contraintes du problème étaient correctement modélisées.

Cependant, cette approche s'est révélée inefficace pour les grandes instances, en raison des temps de calcul élevés et de la complexité combinatoire. Pour cette raison, nous avons opté pour une méthode heuristique plus adaptée aux dimensions réelles du réseau de l'entreprise.

Cette démarche permet donc de s'appuyer sur un modèle exact comme base de validation, tout en déployant des techniques approchées pour résoudre efficacement les cas réels à grande échelle.

1. Résolution exacte d'une instance réduite

Nous avons dans un premier temps appliqué une méthode exacte de résolution à une instance test comprenant **8 clients**. Cette étape a pour objectif de :

- Vérifier la validité de la modélisation mathématique du problème VRPSPD-TL avec contraintes de capacité ;

- Évaluer la performance du solveur Gurobi en termes de temps de résolution et de qualité de solution ;
- Comprendre le comportement du modèle dans un contexte maîtrisé avant de l'appliquer à une instance de grande taille.

La formulation a été implémentée à l'aide de **Pyomo**, un langage de modélisation algébrique en Python, qui permet de définir rigoureusement les variables de décision, les contraintes (capacité, simultanéité, synchronisation) et l'objectif de **minimisation du makespan** (durée totale maximale des tournées).

Les résultats ont confirmé la validité de la modélisation, tant sur le plan théorique (adéquation aux spécifications) que pratique (interprétation correcte par le solveur).

Cette étape a été cruciale avant le passage à des instances plus grandes.

1.1.1 Transcription de la formulation mathématique en langage de programmation

Le code ci-dessous présente :

1. **Le code Pyomo** implémentant la formulation mathématique, mettant en évidence la structure des variables (binaires de routage, continues pour les temps) et des contraintes.
2. **La solution optimale** retournée par Gurobi pour l'instance test, incluant :
 - La séquence des clients visités par chaque véhicule,
 - Les temps d'exécution associés,
 - La valeur du Makespan obtenu.

Cette visualisation concrète permet de valider l'adéquation entre le modèle théorique et son implémentation opérationnelle.

```
import pyomo.environ as pyo

# ===== PARAM TRES =====

# Donn es
Q = 20                # Capacit  des v hicules
T_max = 10            # Dur e max par tourn e
m = 3                 # Nombre de v hicules

# Demandes de livraison
d = {0: 0, 1: 3, 2: 2, 3: 4, 4: 1, 5: 3, 6: 2, 7: 4, 8: 1}

# Demandes de collecte
```

```
p = {0: 0, 1: 2, 2: 1, 3: 3, 4: 1, 5: 2, 6: 1, 7: 3, 8: 1}

# Temps de trajet (heures)
c = {
    (0,0): 0.0, (0,1): 1.2, (0,2): 1.1, (0,3): 0.4, (0,4): 0.8,
    (0,5): 1.5,
    (0,6): 0.9, (0,7): 1.3, (0,8): 1.0,
    (1,0): 1.2, (1,1): 0.0, (1,2): 0.4, (1,3): 1.1, (1,4): 0.5,
    (1,5): 0.6,
    (1,6): 0.3, (1,7): 0.8, (1,8): 0.4,
    (2,0): 1.1, (2,1): 0.4, (2,2): 0.0, (2,3): 0.8, (2,4): 0.5,
    (2,5): 0.6,
    (2,6): 1.3, (2,7): 1.2, (2,8): 1.4,
    (3,0): 0.4, (3,1): 1.1, (3,2): 0.8, (3,3): 0.0, (3,4): 0.3,
    (3,5): 1.0,
    (3,6): 0.5, (3,7): 1.7, (3,8): 1.3,
    (4,0): 0.8, (4,1): 0.5, (4,2): 0.5, (4,3): 0.3, (4,4): 0.0,
    (4,5): 0.8,
    (4,6): 0.7, (4,7): 1.0, (4,8): 1.7,
    (5,0): 1.5, (5,1): 0.6, (5,2): 0.6, (5,3): 1.0, (5,4): 0.8,
    (5,5): 0.0,
    (5,6): 0.9, (5,7): 0.6, (5,8): 1.0,
    (6,0): 0.9, (6,1): 0.3, (6,2): 1.3, (6,3): 0.5, (6,4): 0.7,
    (6,5): 0.9,
    (6,6): 0.0, (6,7): 1.1, (6,8): 0.8,
    (7,0): 1.3, (7,1): 0.8, (7,2): 1.2, (7,3): 1.7, (7,4): 1.0,
    (7,5): 0.6,
    (7,6): 1.1, (7,7): 0.0, (7,8): 0.3,
    (8,0): 1.0, (8,1): 0.4, (8,2): 1.4, (8,3): 1.3, (8,4): 1.7,
    (8,5): 1.0,
    (8,6): 0.8, (8,7): 0.3, (8,8): 0.0
}
```

Listing 4.1 – Instance test du VRPSPD-TL : paramètres des clients (demandes)

```
# ===== MOD LE PYOMO =====

model = pyo.ConcreteModel()

N = list(d.keys())
A = [(i, j) for i in N for j in N if i != j]
K = list(range(1, m+1))
```

```
model.I = pyo.Set(initialize=N)
model.A = pyo.Set(initialize=A, dimen=2)
model.K = pyo.Set(initialize=K)

# Variables
model.x = pyo.Var(model.A, model.K, domain=pyo.Binary)
model.z = pyo.Var(model.A, domain=pyo.NonNegativeReals) #
    livraison
model.v = pyo.Var(model.A, domain=pyo.NonNegativeReals) # collecte
model.Z = pyo.Var(domain=pyo.NonNegativeReals)

# Objectif : min Z
model.obj = pyo.Objective(expr=model.Z, sense=pyo.minimize)

# Contraintes

# (1) Chaque v hicule part une fois du d p t
model.c1 = pyo.ConstraintList()
for k in K:
    model.c1.add(sum(model.x[0, j, k] for j in N if j != 0) == 1)

# (2) Chaque v hicule retourne au d p t
model.c2 = pyo.ConstraintList()
for k in K:
    model.c2.add(sum(model.x[i, 0, k] for i in N if i != 0) == 1)

# (3) Chaque client est servi une seule fois
model.c3 = pyo.ConstraintList()
for j in N:
    if j != 0:
        model.c3.add(sum(model.x[i, j, k] for i in N if i != j for
            k in K) == 1)

# (4) Continuité du trajet
model.c4 = pyo.ConstraintList()
for i in N:
    if i != 0:
        for k in K:
            model.c4.add(sum(model.x[i, j, k] for j in N if j !=
                i) == sum(model.x[j, i, k] for j in N if j != i))
```

```
# (5) Capacit max
model.c5 = pyo.ConstraintList()
for i, j in A:
    model.c5.add(model.z[i, j] + model.v[i, j] <= Q *
        sum(model.x[i, j, k] for k in K))

# (6) Conservation livraison
model.c6 = pyo.ConstraintList()
for i in N:
    if i != 0:
        model.c6.add(sum(model.z[j, i] for j in N if j != i) -
            sum(model.z[i, j] for j in N if j != i) == d[i])

# (7) Conservation collecte
model.c7 = pyo.ConstraintList()
for i in N:
    if i != 0:
        model.c7.add(sum(model.v[i, j] for j in N if j != i) -
            sum(model.v[j, i] for j in N if j != i) == p[i])

# (8a) Borne inf/sup livraison
model.c8a = pyo.ConstraintList()
for i, j in A:
    model.c8a.add(d[j] * sum(model.x[i, j, k] for k in K) <=
        model.z[i, j])
    model.c8a.add(model.z[i, j] <= (Q - d[i]) * sum(model.x[i, j,
        k] for k in K))

# (8b) Borne inf/sup collecte
model.c8b = pyo.ConstraintList()
for i, j in A:
    model.c8b.add(p[i] * sum(model.x[i, j, k] for k in K) <=
        model.v[i, j])
    model.c8b.add(model.v[i, j] <= (Q - p[j]) * sum(model.x[i, j,
        k] for k in K))

# (9) Initialisation au d p t
for j in N:
    if j != 0:
        model.v[0, j].fix(0)
for i in N:
```

```
    if i != 0:
        model.z[i, 0].fix(0)

# (10) Temps max par tourn e
model.c10 = pyo.ConstraintList()
for k in K:
    model.c10.add(sum(c[i, j] * model.x[i, j, k] for (i, j) in A)
                  <= T_max)

# (11) D finition du makespan
model.c11 = pyo.ConstraintList()
for k in K:
    model.c11.add(sum(c[i, j] * model.x[i, j, k] for (i, j) in A)
                  <= model.Z)

# ===== SOLUTION =====

solver = pyo.SolverFactory('gurobi')
results = solver.solve(model, tee=False)

# ===== AFFICHAGE =====
print("=====AFFICHAGE =====")
print(f"Temps max utilis (Z) : {model.Z.value:.2f} heures")
for k in K:
    print(f"\nV hicule {k} :")
    for i, j in A:
        if model.x[i, j, k]() == 1:
            print(f" Arc ({i} {j})")
```

Listing 4.2 – Implémentation Pyomo du modèle MILP pour le VRPSPD-TL avec makespan (extrait du code)

2. Déroulement de l’Algorithme

1. Initialisation :

- Le modèle est construit sous Pyomo avec les sets N (nœuds), A (arcs), et K (véhicules).
- Les variables et contraintes sont générées automatiquement.

2. Résolution :

- **Solveur** : Gurobi (optimisation linéaire en nombres entiers).

3. Résultats Obtenus

3.1 Makespan Optimal

- **Z = 6.26 heures** (durée de la tournée la plus longue).

3.2 Tournées des Véhicules

- **Véhicule 1 :**
 $0 \rightarrow 1 \rightarrow 0$
Livraison/Collecte :
 - Livré : $d[1] = 3$ unités.
 - Collecté : $p[1] = 2$ unités.
- **Véhicule 2 :**
 $0 \rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 0$
Livraison/Collecte :
 - Livré : $d[3] + d[6] + d[5] = 4 + 2 + 3 = 9$ unités.
 - Collecté : $p[3] + p[6] + p[5] = 3 + 1 + 2 = 6$ unités.
- **Véhicule 3 :**
 $0 \rightarrow 4 \rightarrow 7 \rightarrow 2 \rightarrow 0$
Livraison/Collecte :
 - Livré : $d[4] + d[7] + d[2] = 1 + 4 + 2 = 7$ unités.
 - Collecté : $p[4] + p[7] + p[2] = 1 + 3 + 1 = 5$ unités.

Le graphe ci-dessous représente la solution obtenue pour l'instance test composée de 8 clients :

- Rouge : Véhicule 1 $\rightarrow [0, 1, 0]$
- Bleu : Véhicule 2 $\rightarrow [0, 3, 6, 5, 0]$
- Vert : Véhicule 3 $\rightarrow [0, 4, 7, 2, 0]$

3.3 Validation des Contraintes

- **Capacité :** Aucun véhicule ne dépasse $Q = 20$.
- **Temps :** Toutes les tournées respectent $T_{max} = 10h$.
- **Demandes :** Toutes les livraisons/collectes sont satisfaites.

Nous avons d'abord testé cette approche sur une petite instance composée de 8 clients, soit 16 nœuds pickup/delivery. Dans ce cas, le solveur a fonctionné correctement et a pu fournir une solution optimale dans un temps raisonnable. Cela nous a permis de valider la justesse de notre modélisation.

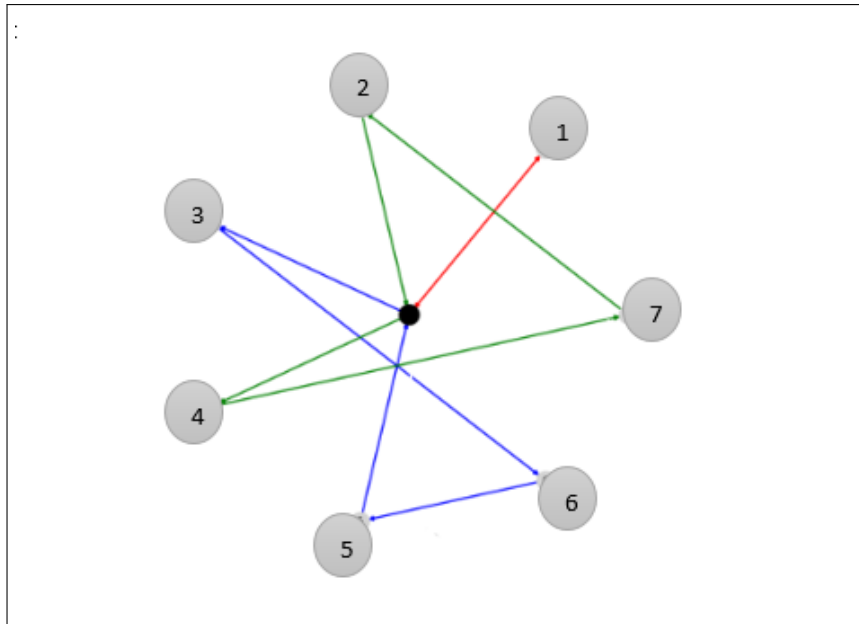


FIGURE 4.4 – Visualisation des tournées optimisées autour du dépôt

Cependant, lorsque nous avons appliqué la même approche à notre instance réelle, composée de 70 nœuds et nécessitant 17 véhicules, le solveur Gurobi n’a pas pu fournir de solution exploitable dans un temps raisonnable. Cette limitation s’explique par plusieurs facteurs clés :

- **Complexité combinatoire** : Le modèle mathématique génère un espace de recherche exponentiellement large pour une instance de cette taille, rendant la résolution exacte impraticable.
- **Contraintes logicielles** : La version de Gurobi utilisée était soumise à une licence académique restreinte, limitant sévèrement l’accès aux ressources de calcul (nombre de threads, mémoire vive disponible et temps CPU alloué).

Cette expérience démontre clairement les limites opérationnelles des méthodes exactes pour le VRPSPD-TL à moyenne échelle. Elle valide surtout le bien-fondé de notre orientation vers des approches heuristiques et métaheuristiques, qui offrent un meilleur compromis entre qualité des solutions et temps de calcul dans des contextes réalistes.

3.4 Limitations rencontrées avec l’approche exacte :

Plusieurs facteurs expliquent l’échec de la méthode exacte sur l’instance réelle :

- **Explosion combinatoire** : le nombre de variables et de contraintes dans un problème VRP croît de manière exponentielle avec le nombre de clients. Avec 70 nœuds, le modèle MILP devient très complexe, avec plusieurs milliers de variables binaires et de contraintes. Cette complexité rend l’exploration de l’espace de solutions extrêmement coûteuse en temps.

- **Capacités informatiques limitées** : malgré les performances du solveur Gurobi, les ressources matérielles disponibles (mémoire vive, puissance de calcul) ne permettaient pas de résoudre efficacement un modèle aussi vaste.
- **Difficultés de faisabilité** : La faisabilité du problème devient de plus en plus difficile à garantir à mesure que la taille de l'instance augmente. L'accumulation de contraintes strictes — comme la capacité maximale des véhicules, les limites de temps et l'obligation d'effectuer la livraison avant le ramassage — réduit fortement l'espace des solutions réalisables. Cela complique la tâche du solveur exact, surtout dans le cas d'instances réelles complexes.

Face à ces limitations, nous avons choisi de nous orienter vers des méthodes approchées, en particulier des heuristiques et métaheuristiques. Ces méthodes ne garantissent pas l'optimalité, mais permettent de :

- Obtenir des **solutions de bonne qualité** en un temps raisonnable,
- **Réduire fortement les temps de calcul**,
- Gérer efficacement des instances de grande taille,
- Adapter facilement les règles de construction ou d'amélioration des tournées aux contraintes de notre cas réel.

Ce choix s'est avéré plus adapté à notre problématique opérationnelle, où la **rapidité et la faisabilité** d'une solution sont souvent plus importantes que l'optimalité théorique.

IV. Application à l'instance réelle (35 clients) :

Après validation du modèle sur l'instance test, nous avons étendu notre étude à une instance plus représentative du réseau réel de Condor Logistique, comprenant 35 clients répartis à travers tout le territoire algérien. Cette instance correspond à une journée réelle d'opération, avec l'utilisation de 17 camions homogènes de 20 tonnes de capacité.

Les villes : Tebessa, Souk Ahras, Bordj BouArreridj, Setif, Constantine, Annaba, Skikda, Guelma, Jijel, Bejaia, Batna, Oran, Sidi bel abbes, Mostaganem, Tiaret, Tissemsilt, Saida , Naama, Relizane, El Bayedh, Alger, Blida, Boumerdes, Tipaza, Tizi Ouzou, Bouira, Medea, Ain defla , Chlef , Msila

- | | | |
|-------------------------|---------------------|----------|
| 1. Méthode | approchée | : |
| 1..1 Heuristique | Constructive | : |

Nous avons conçu un programme permettant de générer une solution initiale en nous appuyant sur une heuristique constructive. Le fonctionnement détaillé de cette heuristique sera présenté dans la section suivante :

Données initiales :

Pour résoudre le problème d'optimisation de tournées, un ensemble de données initiales est nécessaire. Ces données constituent la base de l'algorithme et permettent de modéliser le problème de manière réaliste. Elles sont les suivantes :

TABLE 4.1 – Données initiales du problème

Donnée	Description
Matrice des distances	Distances routières entre toutes les villes de l'instance.
Nombre de véhicules disponibles	17 camions identiques.
Dépôt	Centre de distribution de Condor (Bourdj Bou Arreridj, point 0).
Capacité des véhicules	20 tonnes par camion.
Demandes de livraison	Quantité à livrer par ville (en tonnes).
Demandes de collecte	Quantité à collecter à chaque point (en tonnes).
Vitesse moyenne	100 km/h.
Limite de temps par tournée	10 heures par tournée.

1.2 Méthode approchée : heuristique Clarke-Wright Savings :

Nous avons développé une version modifiée de l'heuristique Clarke Wright Savings, où l'objectif principal est de minimiser les temps de trajet plutôt que les distances. Pour cela, nous calculons le temps entre chaque paire de clients (i, j) comme le rapport entre

la distance et la vitesse moyenne du secteur concerné ($Temps(i, j) = Distance(i, j) / Vitesse_moyenne$). Cette adaptation permet de :

- Générer des tournées plus réalistes qui tiennent compte des conditions réelles de circulation
- Mieux respecter les contraintes temporelles comme les fenêtres de livraison
- Optimiser l'utilisation des véhicules en fonction des durées de trajet effectives.

Le principe de base de l'algorithme (calcul des "savings" et agrégation progressive des tournées) est conservé, mais appliqué sur cette nouvelle métrique temporelle. Cette approche montre sa pertinence particulièrement dans des contextes urbains où les vitesses varient significativement selon les zones

Implémentation Python et explication :

```
# ===== PARAM TRES =====

# Donn es
Q = 20          # Capacit des v hicules
```

```
T_max = 10      # Dur e max par tourn e
m = 17          # Nombre de v hicules

distance_matrix = [
    [0.000, 349.965, 321.979, 397.050, 287.701, 221.061, 36.672,
     89.914, 181.864, 325.854, ...],
    [349.965, 0.000, 655.875, 739.398, 598.776, 547.092, 322.902,
     435.220, 525.952, 668.288, ...],
    ...
]

# Demandes de livraison
d = {
    0: 0, 1: 1, 2: 3, 3: 3, 4: 2, 5: 3, 6: 3, 7: 2, 8: 2, 9: 2,
    10: 1, 11: 1, 12: 3, 13: 1, 14: 1, 15: 1, 16: 2, 17: 3, 18: 2,
    19: 3,
    20: 3, 21: 2, 22: 1, 23: 3, 24: 2, 25: 2, 26: 3, 27: 2, 28: 2,
    29: 2,
    30: 1, 31: 2, 32: 2, 33: 2, 34: 1
}

# Demandes de collecte
p = {
    0: 0, 1: 1, 2: 2, 3: 0, 4: 2, 5: 1, 6: 1, 7: 2, 8: 2, 9: 2,
    10: 1, 11: 1, 12: 1, 13: 1, 14: 2, 15: 0, 16: 1, 17: 0, 18: 2,
    19: 1,
    20: 2, 21: 2, 22: 2, 23: 1, 24: 2, 25: 2, 26: 2, 27: 1, 28: 2,
    29: 1,
    30: 2, 31: 2, 32: 2, 33: 1, 34: 1
}

# Vitesse moyenne en km/h
vitesse = 100

# G n ration du dictionnaire de temps
c = {
    (i, j): round(distance_matrix[i][j] / vitesse, 2)
    for i in range(len(distance_matrix))
    for j in range(len(distance_matrix[i]))
}
```

Listing 4.3 – Implémentation en Python

1.3 Utilité de la matrice des temps dans la modélisation :

Afin de respecter les contraintes liées à la limite de temps (time limit) imposées dans notre problème, il est nécessaire de disposer d'une matrice représentant les temps de trajet entre les différents points (clients, dépôt, etc.).

Étant donné que nous connaissons la vitesse moyenne de chaque véhicule, nous avons généré cette matrice des temps à partir de la matrice des distances, en divisant chaque distance par la vitesse correspondante. Cela nous a permis de calculer un temps estimé pour chaque trajet.

1.4 L'implémentation de l'Algorithme CWS :

L'algorithme a été implémenté en Python. Voici un aperçu des étapes principales du code :

Étape 1 : Calcul des économies

```
# Etape 1 : calcul des savings
savings = []
for i in N:
    for j in N:
        if i < j:
            s = c[(i, 0)] + c[(0, j)] - c[(i, j)]
            savings.append((i, j, s))
```

On calcule pour chaque paire de clients les économies réalisées.

Étape 2 : Tri décroissant des économies

```
# Tri d croissant
savings.sort(key=lambda x: x[2], reverse=True)
```

On trie les couples de clients du plus grand au plus petit gain.

Étape 3 : Construction des tourné

```
# Etape 2 : fusion des tournés si faisable
for i, j, s in savings:
    # Trouver les routes contenant i et j
    ri_key = rj_key = None
    for key, route in routes.items():
        if route[1] == i and route[-2] != j:
            ri_key = key
```

```
        if route[-2] == j and route[1] != i:
            rj_key = key

# Si i est en fin de tournée, j en début (et pas même route)
if ri_key is not None and rj_key is not None and ri_key != rj_key:
    ri = routes[ri_key]
    rj = routes[rj_key]
    new_route = ri[:-1] + rj[1:] # fusionner sans les
        d p ts internes

# Calcul de la charge
total_d = sum(d[k] for k in new_route if k != 0)
total_p = sum(p[k] for k in new_route if k != 0)
if total_d + total_p > Q:
    continue

# Calcul de la durée
time = 0
for u, v in zip(new_route[:-1], new_route[1:]):
    time += c[(u, v)]
if time > T_max:
    continue

# Fusion acceptée
del routes[ri_key]
del routes[rj_key]
routes[i] = new_route

# Résultat
return list(routes.values())
```

À partir des économies triées, l'algorithme construit les tournées en considérant trois cas suivants :

1. Initialisation :

— Chaque client est affecté à sa propre tournée individuelle :

$$[\text{Dépôt}] \rightarrow [\text{Client } i] \rightarrow [\text{Dépôt}]$$
$$[\text{Dépôt}] \rightarrow [\text{Client } j] \rightarrow [\text{Dépôt}]$$

2. Fusion des tournées :

Uniquement deux cas sont considérés :

Cas 1 - Fusion possible : .

- Les clients i et j sont aux extrémités de leurs tournées respectives
- La capacité véhicule ≤ 20 (livraison + collecte)
- La durée totale ≤ 10 unités de temps
- Résultat :

[Dépôt] $\rightarrow \dots \rightarrow [i] - [j] \rightarrow \dots \rightarrow$ [Dépôt]

Cas 2 - Fusion impossible : .

- Si l'un des clients n'est pas en position d'extrémité
- Les tournées restent inchangées

3. Contraintes :

- Aucun client n'est jamais « non desservi » (tous ont une tournée initiale)
- Les fusions ne peuvent se faire qu'entre extrémités de tournées

Pour chaque opération, la charge totale est recalculée ainsi que le temps de parcours, afin de garantir la faisabilité de l'itinéraire.

1.5 Évaluation des tournées (calcul de la durée)

Après la construction des tournées via l'algorithme de Clarke and Wright, nous avons évalué chaque itinéraire en calculant sa durée totale, afin de vérifier le respect de la contrainte temporelle ($T_{max} = 10$).

Trois fonctions principales sont utilisées :

1. Calcule la durée d'une tournée en additionnant les temps de trajet entre chaque paire de clients

```
#===== FONCTION QUI CALCULE LA DUREE
DE CHAQUE TOURNNEE
=====
def get_route_duration(route, c):
    """
    Calcule la dur e totale d'une tourn e donn e.
    :param route: liste des n uds visit s dans l'ordre (ex:
        [0, 2, 4, 0])
    :param c: dictionnaire des temps de trajet {(i, j):
        dur e}
    :return: dur e totale de la tourn e
    """
```

```

duration = 0
for i in range(len(route) - 1):
    duration += c[(route[i], route[i+1])]
return duration

```

2. Application de la fonction précédente à l'ensemble des tournées

```

def get_all_durations(routes, c):
    """
    Calcule la durée totale de chaque tournée.
    :param routes: liste de tournées (ex: [[0,1,2,0],
        [0,3,4,0]])
    :param c: dictionnaire des temps de trajet
    :return: liste des durées [durée_1, durée_2, ...]
    """
    return [get_route_duration(route, c) for route in routes]

```

3. Extrait la durée maximale parmi toutes les tournées, ce qui permet d'évaluer le planning global et d'identifier la tournée la plus longue.

```

def get_makespan(routes, c):
    """
    Calcule le makespan, c'est-à-dire la durée maximale
    parmi toutes les tournées.
    :param routes: liste des tournées
    :param c: dictionnaire des temps
    :return: durée maximale
    """
    durations = get_all_durations(routes, c)
    return max(durations)

```

Cette étape permet de s'assurer que chaque tournée reste réalisable dans le délai imparti, tout en optimisant l'équilibrage entre véhicules.

Étape finale : Exécution du programme et analyse des résultats

Une fois les fonctions définies, nous appelons l'algorithme `clarke_wright_savings_VRPSPD_TL` avec les paramètres du problème : les demandes de livraison d , de collecte p , la matrice des temps c , la capacité Q et la durée maximale

$$T_{max}.$$

```

#=====APPEL DE LA
FONCTION=====

routes = clarke_wright_savings_VRPSPD_TL(d, p, c, Q, T_max)

```

```
print("Tournées construites :")
for r in routes:
    print(r)

durations = get_all_durations(routes, c)

for i, dur in enumerate(durations):
    print(f"Tournée {i+1} : durée = {dur:.2f} heures")

Z = get_makespan(routes, c)
print(f" Makespan = {Z:.2f} heures")
```

Le programme affiche alors :

- La **liste des tournées construites**, c'est-à-dire les itinéraires suivis par chaque véhicule.
- La durée de chaque tournée, calculée grâce à la fonction `get_all_durations()`.
- Enfin, le **makespan**, c'est-à-dire la durée maximale parmi toutes les tournées, obtenu via la fonction `get_makespan()`. Ce critère permet d'évaluer l'équilibrage du planning et la performance globale de la solution obtenue.

1.6 Résultats Obtenus :

Suite à l'exécution de l'algorithme de Clarke and Wright Savings, **7 tournées** ont été générées, couvrant l'ensemble des 35 clients. Voici les itinéraires obtenus :

Voici les tournées et leurs durées correspondantes :

$T_1 = [0, 23, 24, 25, 0]$	durée = 9,82 h
$T_2 = [0, 2, 19, 27, 4, 5, 0]$	durée = 9,66 h
$T_3 = [0, 10, 12, 1, 15, 32, 33, 29, 0]$	durée = 9,84 h
$T_4 = [0, 3, 13, 9, 14, 8, 11, 0]$	durée = 8,98 h
$T_5 = [0, 16, 26, 18, 20, 0]$	durée = 9,77 h
$T_6 = [0, 17, 34, 22, 28, 21, 30, 0]$	durée = 8,80 h
$T_7 = [0, 6, 7, 31, 0]$	durée = 2,39 h

Le **makespan**, c'est-à-dire la durée maximale parmi toutes les tournées, est :

$$\text{Makespan} = \max\{\text{durée}_i\} = 9,84 \text{ h}$$

1..7 Interprétation de la solution initiale (Clarke & Wright Savings) :

Dans la solution initiale obtenue par l'algorithme de Clarke & Wright Savings, seulement 7 véhicules ont été utilisés, bien que 17 soient disponibles. Cette réduction est une conséquence directe de la stratégie de l'algorithme, qui cherche à minimiser le temps total de trajet parcouru en fusionnant des tournées individuelles lorsque cela est possible.

Le but de l'heuristique est de maximiser le regroupement de clients dans des tournées faisables, tout en respectant les contraintes de capacité et de durée maximale.

Il est important de noter que cette méthode ne cherche pas à minimiser le makespan, mais bien le coût cumulé des trajets. Le makespan sera optimisé ultérieurement grâce à des phases d'équilibrage et de recherche locale.

1.8 Limites de la solution initiale

La solution obtenue à l'aide de l'heuristique de Clarke Wright Savings représente une **solution initiale faisable pas un optimum**, construit de manière rapide et efficace à partir des économies réalisées sur les distances.

Cependant, cette approche **gloutonne** (greedy) prend des décisions de manière séquentielle, en sélectionnant à chaque étape le **gain immédiat le plus élevé**, sans possibilité de retour en arrière. Cela peut conduire à une solution sous-optimale à l'échelle globale, en particulier si les premiers choix bloquent des regroupements meilleurs possibles par la suite.

2. Heuristique amélioratives (Relocate) :

Afin d'améliorer la solution initiale obtenue par l'algorithme de Clarke-Wright, nous avons mis en œuvre une heuristique améliorative de type Relocate. Celle-ci vise à réduire le makespan (la durée maximale parmi toutes les tournées) en déplaçant intelligemment certains clients d'une tournée vers une autre moins chargée ou vide.

Voici les étapes principales de l'implémentation :

Étape 1 : Extension à m tournées (ajout de tournées vides)

Avant de commencer l'amélioration, on vérifie si le nombre de tournées obtenues est inférieur au nombre total de camions disponibles m .

Dans ce cas, on ajoute des tournées vides sous la forme $[0, 0]$, représentant un trajet du dépôt vers le dépôt sans client. Ces tournées pourront être utilisées pour y insérer des clients ultérieurement.

Dans notre cas, l'algorithme initial n'a pas utilisé tous les camions. On complète donc la solution en ajoutant des tournées vides :

Si $|\mathcal{T}| < m$, alors on ajoute $m - |\mathcal{T}|$ tournées de la forme $[0, 0]$

Objectif : compléter les tournées afin d’obtenir un total égal au nombre de véhicules disponibles m .

```
# Etape 1 : Ajouter des tournées vides si  $m < |\mathcal{T}|$ 
while len(routes) < m:
    routes.append([0, 0])
```

Étape 2 : Fonctions auxiliaires :

L’objectif : calculer la durée et la charge d’une tournée pour vérifier les contraintes. Ces fonctions permettent de contrôler les **contraintes de durée** (T_{max}) et de **capacité** (Q).

```
# Fonction auxiliaire : durée d'une tournée
def route_duration(route):
    return sum(c[(route[i], route[i+1])] for i in
               range(len(route)-1))

# Fonction auxiliaire : charge (livraison + collecte)
def route_load(route):
    return sum(d[i] + p[i] for i in route if i != 0)
```

Étape 3 : Recherche d’amélioration :

L’objectif : tant qu’on peut améliorer le makespan, on cherche à déplacer des clients. On identifie la tournée la plus longue (r_{max}) car c’est elle qui détermine le makespan.

L’idée est de réduire sa charge en déplaçant un client vers une autre tournée, si cela reste faisable.

```
improved = True
while improved:
    improved = False

    # Identifier la tournée la plus longue
    durations = [route_duration(r) for r in routes]
    idx_max = durations.index(max(durations))
    r_max = routes[idx_max]
    Z_current = durations[idx_max]
```

Étape 4 : Tentatives de déplacement (Relocate) :

L'objectif : tester le déplacement de chaque client de `r_max` vers chaque autre tournée.

Technique utilisée : La recherche locale utilisée repose sur le déplacement (relocate) de chaque client vers toutes les positions possibles dans les autres tournées, à condition que le mouvement soit réalisable (capacité, temps) et améliore le makespan.

```
# Pour chaque client (hors d p t) de la tournée la plus longue
for i in range(1, len(r_max) - 1):
    client = r_max[i]

    for k in range(len(routes)):
        if k == idx_max:
            continue # pas la même tournée
        r_target = routes[k]

        for pos in range(1, len(r_target)): # positions
            possibles d'insertion
            # Créer copies temporaires
            new_r_max = r_max[:i] + r_max[i+1:]
            new_r_target = r_target[:pos] + [client] +
                r_target[pos:]

            # Vérifier la capacité
            if route_load(new_r_max) > Q or
                route_load(new_r_target) > Q:
                continue

            # Vérifier la durée
            if route_duration(new_r_max) > T_max or
                route_duration(new_r_target) > T_max:
                continue

            # Nouveau makespan
            test_routes = copy.deepcopy(routes)
            test_routes[idx_max] = new_r_max
            test_routes[k] = new_r_target
            test_durations = [route_duration(r) for r in
                test_routes]
            new_makespan = max(test_durations)

            # Si amélioration, appliquer
```

```
        if new_makespan < Z_current:
            routes[idx_max] = new_r_max
            routes[k] = new_r_target
            improved = True
            break # on relance une nouvelle recherche
    if improved:
        break
if improved:
    break
```

Étape 5 : Restitution des tournées finales

Objectif : fournir la nouvelle solution, une fois que plus aucun déplacement ne permet d'améliorer le makespan.

On retourne la structure finale des tournées optimisées.

```
return routes
```

Étape 6 : Évaluation finale de la solution :

Objectif : analyser les performances après l'application de l'heuristique.

Approche d'analyse : Chaque tournée est évaluée en termes de durée, et le nouveau makespan est recalculé.

```
# Etape 2 : améliorer la solution
balanced_routes = balance_makespan_with_relocate(routes, d, p, c,
    Q, T_max, m)

# Affichage
for i, r in enumerate(balanced_routes):
    duration = sum(c[(r[k], r[k+1])]) for k in range(len(r)-1))
    print(f"Tournée {i+1}: {r}      durée = {duration:.2f} h")

Z = max(sum(c[(r[k], r[k+1])]) for k in range(len(r)-1)) for r in
    balanced_routes)
print(f"\n Makespan après équilibrage : {Z:.2f} h")
```

2.1 Résultats obtenus

Suite à l'exécution de l'heuristique améliorative, une nouvelle solution a été générée, permettant une meilleure répartition des clients entre les tournées et une réduction du *makespan*.

Les tournées obtenues sont les suivantes :

- Tournée 1 : [0, 24, 25, 0] — durée = 7.32 h
- Tournée 2 : [0, 2, 27, 4, 5, 0] — durée = 9.24 h
- Tournée 3 : [0, 12, 15, 32, 33, 29, 0] — durée = 9.55 h
- Tournée 4 : [0, 3, 13, 9, 14, 8, 11, 0] — durée = 8.98 h
- Tournée 5 : [0, 18, 20, 0] — durée = 3.60 h
- Tournée 6 : [0, 17, 34, 22, 28, 21, 30, 0] — durée = 8.80 h
- Tournée 7 : [0, 1, 10, 6, 7, 31, 0] — durée = 8.75 h
- Tournée 8 : [0, 23, 0] — durée = 8.44 h
- Tournée 9 : [0, 16, 0] — durée = 6.18 h
- Tournée 10 : [0, 19, 0] — durée = 8.60 h
- Tournée 11 : [0, 26, 0] — durée = 9.58 h
- Tournée 12 : [0, 0] — durée = 0.00 h
- Tournée 13 : [0, 0] — durée = 0.00 h
- Tournée 14 : [0, 0] — durée = 0.00 h
- Tournée 15 : [0, 0] — durée = 0.00 h
- Tournée 16 : [0, 0] — durée = 0.00 h
- Tournée 17 : [0, 0] — durée = 0.00 h

Makespan après équilibrage : 9.58 h

2..2 Interprétation de la solution améliorée

Suite à l'application de l'heuristique améliorative, une **nouvelle configuration des tournées** a été obtenue. Sur les **17 tournées disponibles**, **11 seulement ont été effectivement utilisées** pour desservir l'ensemble des 35 clients. Le **makespan** de cette solution s'élève à **9.58 heures**, ce qui signifie que la tournée la plus longue ne dépasse pas les 10 heures. Ce résultat représente une **amélioration par rapport au makespan initial de 9.84 heures**, obtenu avec la solution constructive.

L'application de l'heuristique a permis de **réduire la durée maximale des tournées** en soulageant les itinéraires les plus chargés. Certains clients ont été réaffectés à d'autres tournées, ce qui a contribué à une **meilleure répartition de la charge de travail entre les véhicules**. Cette réorganisation favorise un équilibrage temporel, réduisant ainsi le risque de surcharge ou de retard sur un véhicule spécifique.

Bien que cette amélioration par l'opérateur Relocate ait permis une meilleure répartition des tournées et une réduction du makespan, la solution obtenue reste un optimum local. Afin d'explorer davantage l'espace des solutions et d'échapper à ces optima locaux, une métaheuristique plus puissante a été envisagée : **la Tabu Search**.

3. Métaheuristique :

3.1 Tabu-search :

Après l'obtention d'une solution initiale via l'algorithme de Clarke and Wright, puis son amélioration par une heuristique locale basée sur l'opérateur Relocate, il est apparu nécessaire d'explorer plus largement l'espace des solutions. En effet, les heuristiques locales peuvent conduire à des optima locaux sans possibilité d'en sortir. Dans cette optique, la méthode Tabu Search a été adoptée afin d'intensifier la recherche et de surmonter les blocages locaux. Cette métaheuristique repose sur un mécanisme de mémoire qui interdit temporairement certains mouvements déjà effectués, favorisant ainsi l'exploration de nouvelles solutions tout en évitant les cycles.

Ce type de problème (VRP avec contraintes de temps et ramassage/livraison simultanés) génère un espace de recherche extrêmement vaste et irrégulier. Les heuristiques classiques comme relocate ne permettent pas toujours d'explorer des configurations éloignées.

La Tabu Search permet justement de dépasser ces limites en autorisant temporairement des détériorations contrôlées de la solution pour mieux progresser ensuite.

Le problème de tournées avec ramassage et livraison simultanés et contrainte de durée maximale (VRPSPD-TL) est de nature combinatoire et très complexe à résoudre exactement.

Dans notre cas, la solution obtenue par les heuristiques précédentes (Clarke & Wright, relocate) reste sous-optimale, notamment à cause d'un déséquilibre entre les tournées.

La Tabu Search s'avère alors pertinente pour réorganiser dynamiquement les tournées afin de réduire le makespan tout en respectant les contraintes opérationnelles.

3.2 Objectif général :

Améliorer la solution obtenue précédemment (par Relocate ou Clarke-Wright) en réduisant le makespan, c'est-à-dire la durée maximale d'une tournée, en explorant des solutions voisines tout en évitant les répétitions grâce à une liste tabu.

3.3 Schéma explicatif de Tabu-search :

Le schéma ci-dessous illustre le fonctionnement global de cet algorithme : depuis une solution initiale jusqu'à la génération de solutions voisines, la gestion de la liste taboue et la mise à jour du meilleur résultat trouvé.

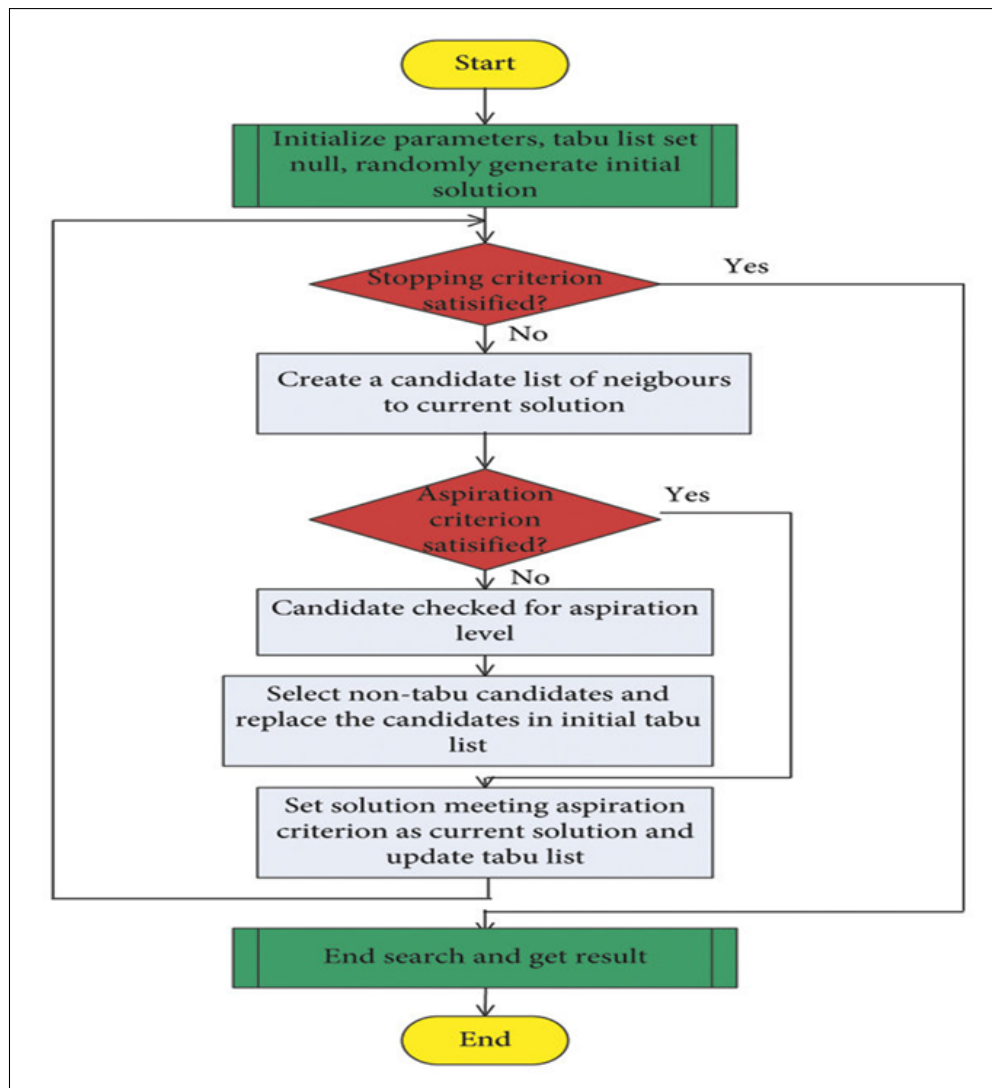


FIGURE 4.5 – Schéma de fonctionnement de l’algorithme Tabu Search (Youle Song, Ding Nie, Litao . Journal of robotics)

3.4 Le déroulement de l’algorithme Tabu :

3.5 L’implémentation :

— Déclaration de la fonction principale :

```
def tabu_search_vrpspd_tl(routes, d, p, c, Q, T_max,
    m, tabu_tenure=5, max_iter=10):
```

Cette fonction applique la **métaheuristique Tabu Search** pour améliorer une solution initiale (routes) d’un VRPSPD-TL.

Entrées principales

- **routes** : solution initiale (liste des tournées),
- **d, p** : vecteurs de livraisons et ramassages,
- **c** : matrice de distances ou durées,

- **Q** : capacité maximale des véhicules,
- **T_max** : durée maximale autorisée par tournée,
- **m** : nombre de véhicules,
- **tabu_tenure** : durée pendant laquelle un mouvement est tabou,
- **max_iter** : nombre maximal d'itérations.
- **Sous-fonctions utilitaires :**
 - **Calcule la durée totale d'une tournée :**

```
def route_duration(route):
    return sum(c[(route[i], route[i+1])] for i in range(len(route)-1))
```
 - **Calcule la charge totale (livraison + collecte) d'une tournée :**

```
def route_load(route):
    return sum(d[i] + p[i] for i in route if i != 0)
```
 - **Vérifie la faisabilité d'une tournée ($\text{charge} \leq Q$ et $\text{durée} \leq T_{max}$) :**

```
def is_feasible(route):
    return route_load(route) <= Q and route_duration(route) <= T_max
```
 - **Calcule le makespan : la durée de la tournée la plus longue (objectif à minimiser) :**

```
def get_makespan(routes):
    return max(route_duration(r) for r in routes)
```
- **Initialisation des variables**

```
best_solution = copy.deepcopy(routes)
best_makespan = get_makespan(best_solution)
current_solution = copy.deepcopy(routes)
current_makespan = best_makespan
```
- **Explication des variables**
 - **best_solution** : meilleure solution trouvée jusqu'à présent.
 - **current_solution** : solution courante de l'itération.
 - **tabu_list** : liste de mouvements interdits temporairement (mouvements tabous).

Boucle principale du Tabu Search

Pour répéter le processus d'exploration jusqu'à atteindre **max_iter** ou une stagnation, on utilise :

```
for iteration in range(max_iter):
```

Recherche des voisins par déplacement ("relocate") :

On tente de déplacer un client d'une tournée i vers une autre tournée j :

```
for i in range(len(current_solution)):
    for j in range(len(current_solution)):
```

Détails de la recherche de voisinage

1. Sélection d'un client à déplacer (hors dépôt) et insertion à une nouvelle position :

```
    for pos in range(1, len(route_from)-1):
        for insert_pos in range(1, len(route_to)):
```

2. Ignorer les solutions non faisables :

```
        if not is_feasible(new_from) or not is_feasible(new_to):
            continue
```

3. Ignorer les mouvements tabous, sauf si la solution améliore le best_makespan :

```
        move = (client, i, j)
        if move in tabu_list and candidate_makespan >= best_makespan:
            continue
```

4. Sauvegarder le meilleur voisin rencontré :

```
        if candidate_makespan < best_candidate_makespan:
            best_candidate = candidate
            best_candidate_makespan = candidate_makespan
            best_move = move
```

Mise à jour des solutions

1. Mise à jour de la solution courante et ajout du mouvement dans la liste taboue :

```
current_solution = best_candidate
current_makespan = best_candidate_makespan
tabu_list.append(best_move)
```

2. Mise à jour de la meilleure solution si amélioration, sinon incrémentation du compteur de stagnation :

```
if current_makespan < best_makespan:
    best_solution = copy.deepcopy(current_solution)
    best_makespan = current_makespan
    iter_no_improve = 0
else:
    iter_no_improve += 1
```

3. Arrêt anticipé en cas de stagnation (20 itérations sans amélioration) :

```
if iter_no_improve >= 20:
    break

return best_solution
```

Affichage des résultats :

```
for i, r in enumerate(final_routes):
    dur = sum(c[(r[k], r[k+1])]) for k in range(len(r)-1)
    print(f"Tournée {i+1}: {r}      durée = {dur:.2f} h")
Z = max(...)
print(f"\n Makespan final après Tabu: {Z:.2f} h")
```

3..6 Résultats **Obtenus** **:**

Tournée 1:	[0, 27, 0]	- durée = 8.98 h
Tournée 2:	[0, 4, 5, 0]	- durée = 5.81 h
Tournée 3:	[0, 33, 32, 29, 0]	- durée = 8.45 h
Tournée 4:	[0, 3, 13, 9, 14, 8, 11, 0]	- durée = 8.98 h
Tournée 5:	[0, 2, 18, 20, 0]	- durée = 7.05 h
Tournée 6:	[0, 17, 34, 22, 28, 21, 30, 0]	- durée = 8.80 h
Tournée 7:	[0, 12, 1, 10, 6, 7, 31, 0]	- durée = 8.75 h
Tournée 8:	[0, 24, 23, 0]	- durée = 8.62 h

Tournée 9: [0, 16, 0]	- durée = 6.18 h
Tournée 10: [0, 19, 0]	- durée = 8.60 h
Tournée 11: [0, 25, 26, 0]	- durée = 8.27 h
Tournée 12: [0, 15, 0]	- durée = 7.14 h
Tournée 13: [0, 0]	- durée = 0.00 h
Tournée 14: [0, 0]	- durée = 0.00 h
Tournée 15: [0, 0]	- durée = 0.00 h
Tournée 16: [0, 0]	- durée = 0.00 h
Tournée 17: [0, 0]	- durée = 0.00 h

Makespan final après Tabu: 8.98 h

3.7 Interprétation des résultats

TABLE 4.2 – Synthèse des résultats avant et après Tabu-search

Tournée	Avant Tabu-search	Après Tabu-search
T1	9.82 h	8.90 h
T7	2.39 h	7.85 h
T12	0.00 h	6.21 h

- La majorité des tournées ont des durées proches du seuil maximal ($T_{\max} = 9$ h), témoignant d'un bon équilibrage.
- Le makespan correspond désormais à 8,98 h, ce qui reflète une optimisation du temps global d'utilisation de la flotte.
- Les clients ont été redistribués pour alléger les tournées surchargées sans violer les contraintes.

V. Évaluation comparative des approches d'optimisation :

L'application des différentes méthodes à une instance réelle composée de **35 clients** et **17 véhicules de 20 tonnes**, dans le cadre d'un problème de type **VRPSPD-TL**, a permis d'évaluer leur efficacité en termes de réduction du **makespan**. Trois étapes ont été successivement mises en œuvre :

- La construction initiale des tournées à l'aide de l'algorithme des savings de Clarke-Wright
- Une heuristique améliorative basée sur des mouvements de type relocate

- Une phase d'amélioration via la métaheuristique Tabu Search, visant à optimiser la durée maximale des tournées.

Le tableau ci-dessous présente les résultats obtenus à chaque étape :

TABLE 4.3 – Résumé des résultats obtenus selon la méthode appliquée

Méthode appliquée	Nombre de tournées actives	Makespan (heures)
Heuristique constructive	7	9,84
Heuristique améliorative	11	9,58
Tabu-search	12	8,98

Nous avons constaté une amélioration significative du makespan, qui passe de 9.84 h à 8.98 h, soit une réduction de 0.86 heures. Cette diminution s'accompagne d'une augmentation du nombre de tournées actives, ce qui, dans notre cas, s'est révélé bénéfique en répartissant mieux les clients, nous avons pu éviter les surcharges et lisser les durées.

La méthode Tabu Search a notamment permis de réduire les déséquilibres entre les tournées avec des durées plus homogènes, tout en respectant les contraintes imposées par l'instance réelle. Même si le nombre de tournées a légèrement augmenté, le gain en qualité de service, fiabilité et flexibilité opérationnelle semble justifié.

Toutefois, cette approche reste dépendante de la qualité de la solution initiale et du voisinage exploré, ici limité au mouvement relocate. Ainsi, bien que non garantie optimale, la solution finale présente un bon compromis entre faisabilité, équilibre opérationnel et temps de calcul, démontrant l'intérêt concret de la Tabu Search pour résoudre le VRPSPD-TL dans un cadre applicatif réaliste.

Enfin, ces résultats confirment la pertinence de la combinaison entre heuristiques classiques et métaheuristiques pour résoudre des instances logistiques réalistes et complexes. Une telle démarche peut être envisagée comme un outil d'aide à la décision applicable en milieu industriel, notamment pour des structures logistiques confrontées à des contraintes similaires à celles du cas étudié.

Conclusion :

Ce chapitre a présenté une approche pragmatique et évolutive pour résoudre le problème de routage de véhicules avec ramassage et livraison simultanés avec limite de temps , appliqué à une instance inspirée d'un cas réel. Pour les petites tailles, la résolution exacte à l'aide de Pyomo et du solveur Gurobi a permis de valider la cohérence de la formulation mathématique. En revanche, pour les instances de taille plus importante, une combinaison d'heuristiques (Clarke & Wright, Relocate) et de métaheuristiques (Tabu Search) a été utilisée afin de construire puis améliorer les solutions de manière

efficace. L'intégration de la Tabu Search s'est révélée déterminante pour mieux équilibrer les tournées et réduire le makespan, assurant ainsi une qualité de service plus homogène, même au prix d'une légère augmentation du nombre de tournées actives. Les résultats obtenus confirment la pertinence d'une approche fondée sur des méthodes heuristiques robustes pour accompagner la prise de décision dans des contextes logistiques complexes.

Conclusion générale et perspectives

Dans un environnement logistique en constante mutation, marqué par des exigences accrues en termes de performance, de réactivité et de durabilité, l'optimisation des tournées de véhicules représente un levier stratégique pour les entreprises de transport. Ce mémoire s'est inscrit dans cette dynamique, en s'intéressant à un cas concret issu de l'entreprise Condor Logistique, confrontée à des problématiques réelles de déséquilibre dans la planification de tournées avec ramassage et livraison simultanés.

L'étude a d'abord permis de poser un cadre théorique rigoureux, en distinguant les spécificités de la logistique directe et inverse, tout en soulignant leur rôle complémentaire au sein d'une supply chain intégrée. Cette base a été essentielle pour comprendre la complexité du problème traité, notamment en termes de flux bidirectionnels, de contraintes opérationnelles et d'objectifs de performance. La formalisation mathématique du problème, intégrant les contraintes de capacité, de temps et la minimisation du makespan, a permis de structurer l'analyse et de définir un modèle adaptable à la réalité du terrain.

La mise en œuvre de différentes techniques de résolution a constitué un apport majeur de ce travail. Pour les petites instances, l'approche exacte basée sur Pyomo et Gurobi a permis d'atteindre des solutions optimales. En revanche, pour les cas plus complexes, l'intégration successive d'une heuristique constructive (Clarke & Wright), d'une technique d'amélioration locale (Relocate), puis d'une métaheuristique (Tabu Search) s'est révélée particulièrement pertinente. Cette approche hybride a permis de générer des solutions de bonne qualité dans des temps raisonnables, tout en assurant un meilleur équilibre entre les tournées, critère essentiel dans le cadre opérationnel de Condor Logistique.

En conclusion, les résultats obtenus confirment l'intérêt d'une combinaison méthodologique entre modélisation rigoureuse et techniques heuristiques pour répondre aux problématiques logistiques complexes. Cette démarche peut être envisagée comme un véritable outil d'aide à la décision, capable d'apporter des solutions concrètes aux défis rencontrés sur le terrain, et adaptable à d'autres contextes industriels confrontés à des contraintes similaires. Elle ouvre également la voie à des perspectives d'amélioration continue, notamment par l'intégration de données en temps réel, de contraintes environnementales ou de modèles d'apprentissage automatique pour renforcer encore la

qualité des décisions logistiques.

Annexe A

Presentation de l'entreprise CONDOR LOGISTICS

Condor Logistics est une entreprise affiliée au groupe Condor, spécialisée dans divers services liés au transport, notamment le fret, le transport de personnel, ainsi que la location et l'entretien de véhicules et d'équipements. Créée en 2014, elle s'est rapidement imposée comme un acteur clé du secteur logistique en Algérie. Avec un effectif d'environ 600 collaborateurs, la société assure le transport terrestre de marchandises sur l'ensemble du territoire national. Grâce à une flotte variée et adaptée, Condor Logistics est en mesure de répondre de manière optimale aux exigences logistiques de ses clients, en leur proposant des solutions efficaces et sur mesure.



FIGURE A.1 – Logo de l'entreprise CONDOR LOGISTICS

I. Historique

:

Depuis sa création en 2014, Condor Logistics s'est dotée d'un parc roulant conséquent, initialement destiné à desservir les différentes filiales du Groupe Condor. En 2016, dans une logique de diversification de ses activités, l'entreprise a élargi son champ d'intervention au-delà du transport de marchandises, en intégrant le transport de personnel, ainsi que des services de location et de maintenance de véhicules et d'engins.

En 2018, Condor Logistics a adopté une démarche QHSE (Qualité, Hygiène, Sécurité, Environnement) en alignant son système de management sur les standards internationaux, obtenant ainsi les certifications ISO 9001, ISO 14001 et ISO 45001. Cette orientation traduit une volonté d'agir en tant qu'entreprise responsable, engagée non seulement envers ses employés, clients et partenaires, mais aussi dans un esprit de concurrence saine. L'entreprise dispose par ailleurs d'un vaste réseau de distribution composé de 25 entrepôts répartis sur le territoire national, couvrant une surface globale de 260 000 m². En 2021, elle a franchi une nouvelle étape en inaugurant un centre logistique intégré de grande envergure à Bordj Bou Arréridj, comprenant une méga-plateforme logistique de 20 hectares, dont 65 000 m² de superficie couverte, une capacité de stockage de 150 000 palettes, un parc à conteneurs de 2 000 TEU, un atelier central de maintenance et une aire de stationnement pouvant accueillir jusqu'à 500 équipements.

II. Le Domaine d'activité de Condor Logistique :

Condor Logistics intervient dans plusieurs segments du secteur du transport et de la logistique, en proposant des solutions adaptées aux besoins variés de ses clients. En matière de transport de marchandises, l'entreprise opère à l'échelle nationale et internationale grâce à une flotte moderne et respectueuse de l'environnement, composée de semi-remorques, camions porteurs, porte-engins, camions de dépannage, porte-conteneurs, etc. Ces véhicules sont équipés de dispositifs de géolocalisation GPS et de chronotachygraphes, assurant un suivi précis et une gestion optimale des trajets.

Cette flotte est pilotée par une équipe de conducteurs expérimentés, formés aux exigences du métier. Concernant le transport de personnel, Condor Logistics propose un service fiable à travers la mise à disposition de bus et minibus de différentes capacités (allant de 12 à 100 places), répondant aux besoins des entreprises ou des collectivités. Enfin, dans le domaine de la location d'engins et de la manutention, l'entreprise dispose d'un parc d'équipements diversifié comprenant des nacelles télescopiques, des reach stackers, des chariots élévateurs allant de 2,5 à 35 tonnes, ainsi que des camions-nacelles et camions-grues, permettant de répondre efficacement aux opérations de levage, de chargement et de manutention lourde.

En complément de ses activités de transport, Condor Logistics propose également des services de location de véhicules, mettant à disposition une gamme variée de véhicules utilitaires et de tourisme, multimarques, disponibles avec ou sans chauffeur, selon les besoins du client. Cette flexibilité permet de répondre à des exigences ponctuelles ou régulières en matière de mobilité. Par ailleurs, l'entreprise met à profit son expertise technique à travers son service de maintenance de véhicules et d'engins, assurant l'entretien et la réparation de différents types d'équipements : camions, bus, minibus, chariots élévateurs, ainsi que des véhicules légers. Ces prestations sont réalisées par une

équipe qualifiée, dans des ateliers équipés pour garantir la fiabilité et la disponibilité des véhicules en exploitation.

III. Les missions de Condor logistique :

- Fournir un niveau de qualité de services à la hauteur des attentes
- Améliorer en continu les dynamiques de progrès et les synergies internes.
- Participer, aux côtés des différents opérateurs, à l'atteinte de l'excellence au sein de la chaîne logistique.

Bibliographie

- [1] Brach, F., Brusset, X. (s.d.). *La reverse logistique en tant qu'avantage compétitif*
- [2] Bennekrouf, M. (2013, 4 décembre). *Modélisation et simulation d'une chaîne logistique inverse en tenant compte de la robustesse*
- [3] De Brito M.P., Dekker R. (2003), *A framework for reverse logistics*, *ERIM Report Series Research in Management*, Erasmus University, Rotterdam
- [4] Dowlatshahi S. (2000), *Developing a theory of reverse logistics*, *Interfaces*, Vol. 30, n°3, pp. 143-155.
- [5] Fleischmann M. Bloemhof-Ruwaard J. M., Dekker R. (1997), *Quantitative models for reverse logistics : A review*, *European Journal of Operational Research*, Vol.103, pp.1-17.
- [6] Diane A. M. (2010), *Reverse Logistics - Effective Returns Management in an Integrated Supply Chain*
- [7] Mohammed B. (2013), *Modélisation et Simulation d'Une Chaîne Logistique Inverse en Tenant Compte de La Robustesse*, (n°4).
- [8] Rogers et Tibben-L. (1999), *Reverse logistics : stratégies et techniques*
- [9] Serge L, Diane R. (2003), *Logistique inverse : revue de littérature*, *Les Cahiers du GERAD*, (G-2003-61), 45 p
- [10] Errabi, G., Hamadi, C. (s.d.). *La logistique inversée et son impact sur la réduction de l'insatisfaction sur Internet*. École Nationale de Commerce et de Gestion, Université Cadi Ayyad, GREGO.
- [11] Giuntini, R., Andel, T. (1995). *Advance with reverse logistics. Transportation Distribution*
- [12] Philippe-Pierre Dornier, Michel Fender, *ALa Logistique Globale, Enjeux - Principes - Exemples »* (2001), Editions d'organisation.
- [13] Alexandre K. Samii, Dunod. *Stratégie logistique, Supply chain management »* (2001), (3ème édition, 2004).
- [14] Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I. (2016). *The vehicle routing problem : State of the art classification and review*. *Computers Industrial Engineering*, 99, 300–313.

- [15] Boggyrbayeva, A., Dauletbayev, B., Meraliyev, M. (2025). *Integrating Metaheuristics and Machine Learning for Enhanced Vehicle Routing : A Comparative Study of Hyperheuristic and VAE-Based Approaches*. World Electric Vehicle Journal, 16(5), 258.
- [16] Paché, G. (2015). *La logistique : Organisation, pilotage et performance*. Vuibert.
- [17] Developpez.com. (2019, décembre 9). *Sortie de Gurobi 9.0 avec un solveur quadratique non convexe, contraintes et objectifs, et des améliorations appréciables de performance*.
- [18] Radhoui, M. (2021). *Optimisation de tournées de véhicules avec ramassage et livraisons, prise en compte des aspects logistiques réels*. [Thèse de doctorat, Université de Lorraine].
- [19] DataCamp. (s.d.). *Optimisation linéaire avec Pyomo en Python*.
- [20] Trebilcock B. *The seven deadly sins of reverse logistics*. *Logistics Management* 2002 ; June
- [21] Fleischmann M, Bloemhof-Ruwaard JM, Dekker R, van der Laan E, van Nunen JAEE, van Wassenhove LN. *Quantitative models for reverse logistics : a review*. *European Journal of Operational Research* 1997
- [22] Dekker, R., Fleischmann, M., Inderfurth, K., Van Wassenhove, L. N. (2004). *Reverse logistics : Quantitative models for closed-loop supply chains*. Springer.
- [23] Gupta, S. M. (2013). *Reverse supply chains : Issues and analysis*. CRC Press.
- [24] Bourrières, J.-P. (2013). *La supply chain verte : Intégrer le développement durable dans la logistique*. Editions d'Organisation.
- [25] Michaud, G. (2022). *Circular economy and reverse logistics*. Routledge.
- [26] Rogers, D. S., Tibben-Lembke, R. S. (2002). *Differences between forward and reverse logistics in a retail environment*. *Supply Chain Management : An International Journal*
- [27] Pedram, A., Yusoff, N. B., Udoncy, O. E., Mahat, A. B., Pedram, P., Babalola, A. (2017). *Integrated forward and reverse supply chain : A tire case study*. *Procedia Manufacturing*,
- [28] Feito-Cespon, M., Sarache, W., Cespon-Castro, R. (2021). *Redesign of a sustainable reverse supply chain under uncertainty : A case study*. *Sustainable Production and Consumption*,
- [29] Trebilcock, B. (2002, June). *The seven deadly sins of reverse logistics*. *Logistics Management*.
- [30] Fleischmann, M., Bloemhof-Ruwaard, J. M., Dekker, R., Van der Laan, E., Van Nunen, J. A. E. E., Van Wassenhove, L. N. (1997). *Quantitative models for reverse logistics : A review*. *European Journal of Operational Research*
- [31] Toth, P., Vigo, D. (2002). *The vehicle routing problem*. SIAM.

- [32] Python Software Foundation. (n.d.). *Python documentation*.
<https://www.python.org/doc/> (Consulté le 01 juin 2022)
- [33] Formation Django. (n.d.). *Python : présentation d'ensemble du langage*.
<http://www.formation-django.fr/> (Consulté le 19 juin 2022)
- [34] Hart, W. E., Watson, J. P., Woodruff, D. L. (2011). *Pyomo : Modeling and solving mathematical programs in Python*. Springer and Mathematical Optimization Society.
- [35] Wikipedia. (n.d.). Pyomo. <https://en.wikipedia.org/wiki/Pyomo> (Consulté le 20 juin 2022)
- [36] Pyomo. (n.d.). About Pyomo. <http://www.pyomo.org/about>
- [37] Pyomo. (n.d.). Pyomo Documentation 6.4.1.
<https://pyomo.readthedocs.io/en/stable/>
- [38] Sandia National Laboratories. (2011). Pyomo. <https://www.pyomo.org/>
- [39] Gurobi Optimization, LLC. (n.d.). Gurobi documentation.
<https://www.gurobi.com/>
- [40] Gurobi Optimization, LLC. (n.d.). Gurobi Optimizer.
<https://www.gurobi.com/products/gurobi-optimizer/>