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 de Génie Logistique et Transport

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

D'INGENIEUR d'État

-Filière-

Ingénierie des transports

-Spécialité -

Ingénierie des transports

- Thème -

Optimisation des tournées de distribution de fournitures agricoles

Cas de CPH Agriculture

Réalisé par

ELOUNEG Amine

KHERMANE Zakaria

Membres de Jury :

BOUZID Mouaouia Cherif	MCA ENSTA	Président
ANNAD Oussama	MCB ENSTA	Encadrant
MOULAI Ratiba	MAA ENSTA	Examinatrice
TALBI Tayeb	MAA ENSTA	Examinateur

Alger, le 01/07/2024

Année universitaire 2023/2024

Dédicaces

Je dédie ce travail :

À mon cher père, pour son amour, ses encouragements et son soutien indéfectible tout au long de mes études.
À ma tendre mère, pour ses sacrifices, sa tendresse et ses prières qui m'ont permis d'avancer.
À mon petit frère, pour sa joie de vivre contagieuse et sa capacité à me redonner le sourire dans les moments difficiles.
À mes deux petites sœurs, pour leur douceur et leur affection sincère.
Que ce modeste travail soit le témoignage de ma profonde gratitude et de mon amour éternel pour vous tous.

- Zakaria -

Dédicaces

À mes chers parents, Source intarissable d'amour, de conseils et de soutien Vous qui n'avez jamais cessé de croire en moi Je vous dédie ce modeste travail, témoignage de votre dévouement.

À mon frère, Complice de toujours, présent dans les bons comme les mauvais moments Merci pour tes encouragements et ta précieuse complicité Cette réussite, je la partage avec toi.

À ma sœur, Ma première fan, ma conseillère de chaque instant Ton réconfort et tes mots ont éclairé mon chemin
Que ce mémoire soit le reflet de ton soutien infaillible.

À toute ma famille, Qui n'a eu de cesse de croire en mes rêves Et de m'apporter la force d'avancer Je vous dédie les fruits de cette aventure studieuse.

En espérant avoir transcrit à travers ces dédicaces toute ma gratitude envers vous qui avez jalonné mon parcours et permis l'accomplissement de ce projet.

. - Amine -

Remerciement

Tout d'abord, nous adressons nos remerciements les plus sincères à Allah, le Tout-Puissant, Créateur de l'univers, pour nous avoir accordé la force, la persévérance et la sagesse nécessaires à l'accomplissement de ce mémoire.

Nous exprimons notre profonde gratitude à notre encadrant, M. ANNAD Oussama, pour ses précieux conseils, son accompagnement rigoureux et sa disponibilité tout au long de ce projet. Ses encouragements constants et son expertise ont été des atouts majeurs pour la réussite de cette entreprise.

Nous adressons également nos vifs remerciements à M. BOUZID Mouaouia Cherif, dont l'expertise en recherche opérationnelle et les conseils avisés ont été d'une aide précieuse pour approfondir nos connaissances et améliorer la qualité de notre travail.

Nos remerciements vont aussi à Monsieur BADJARA Mohamed El Amine, chef du département GLT, pour avoir mis à notre disposition les ressources informatiques nécessaires, facilitant ainsi le bon déroulement de nos recherches.

Nous tenons à exprimer notre gratitude envers M. ABI SMAIL HEDJOUDJA Redouan, notre encadrant au sein de l'entreprise CPH Agriculture, pour son accueil chaleureux et ses conseils judicieux durant notre stage.

Nous remercions également toute l'équipe de CPH Agriculture, notamment l'équipe IT, pour leur collaboration et leur aide précieuse.

Nos chaleureux remerciements vont au président du jury, M. BOUZID Mouaouia Cherif, ainsi qu'aux membres du jury, Mme. MOULAI Ratiba et M. TALBI Tayeb, pour avoir accepté d'évaluer notre travail et pour les commentaires constructifs qu'ils apporteront.

Ce mémoire est le fruit d'un travail acharné et d'une précieuse collaboration. Nous sommes reconnaissants envers tous ceux qui ont contribué, de près ou de loin, à son accomplissement.

Table des matières

Dédicace

Remerciements

Table des figures

Liste des tableaux

Liste des graphes

Liste des abréviations

Introduction générale : **11**

Chapitre 1 : Présentation de l'entreprise **13**

1.1 Introduction : 14

1.2 L'entreprise CPH Agriculture : 14

1.3 Structure organisationnelle : 15

1.3.1 L'organigramme de l'entreprise : 15

1.3.2 Les missions de chaque département 15

1.4 Politique de distribution de l'entreprise CPH Agriculture : 16

1.5 Disponibilité des ressources et des infrastructures 17

1.6 Conclusion : 21

Chapitre 2 : Problème des tournes de véhicules **22**

2.1 Fondements théoriques du Problème des tournées de véhicules : 23

2.1.1 Introduction au VRP 23

2.1.1.1 Historique du VRP 23

2.1.1.2 Définition du VRP 23

2.1.1.3 Importance du VRP 24

2.1.2 Classification de VRP 24

2.1.2.1 VRP Statique 24

2.1.2.2 VRP Dynamique 24

2.1.3 Variantes du VRP 25

2.1.4 Revue de littérature 27

2.1.5 Modélisation du Problème 28

2.1.5.1 Modélisation descriptive du VRP 28

2.1.5.2 Modélisation mathématique fondamentale du VRP 29
2.2 Optimisation Combinatoire 34
2.2.1 Introduction de l'Optimisation Combinatoire 34
2.2.1.1 Définition des problèmes d'optimisation combinatoire 34
2.2.1.2 Complexité des problèmes d'optimisation combinatoire 35
2.2.1.3 Définition des problèmes de décision 35
2.2.1.4 Planification sous contraintes 35
2.2.1.5 Conclusion 35
2.2.2 Complexité algorithmique et classes de problèmes 36
2.2.3 Méthodes de résolution 38
2.2.3.1 Méthodes exactes 38
2.2.3.2 Méthodes approchées 38
2.2.3.3 Choix de la méthode en fonction de la complexité du problème 43

Chapitre 3 : Approches de résolution 44
3.1 Introduction 45
3.2 Modélisation du problème pour CPH Agriculture 45
3.2.1 Description du système de distribution 45
3.2.2 Formulation mathématique 46
3.3 Méthode exacte avec IBM – CPLEX 49
3.3.1 Les éléments du logiciel IBM ILOG CPLEX Optimization Studio 50
3.3.1.1 IDE (Environnement de Développement Intégré) 50
3.3.1.2 Langage de modélisation OPL (Optimization Programming Language) 50
3.3.1.3 Le solveur IBM ILOG CPLEX 51
3.3.1.4 Le solveur CP Optimizer 51
3.3.1.5 IBM ILOG Script for OPL 52
3.3.2 L'algorithme de résolution 52
3.4 Conception de l'algorithme génétique 53
3.4.1 Structure des entrés 53
3.4.1.1 Paramètres d'entré généraux 53
3.4.1.2 Paramètres d'entré spéciaux 54
3.4.1.3 Paramètres d'entré spécifiques du problème 54
3.4.1.4 Données entrées 55
3.4.1.5 Données générés 56

3.4.2 Structure de l'Algorithme Génétique - Cas général 57
3.4.2.1 Phase Initiale 57
3.4.2.2 Boucle Évolutive 58
3.4.2.3 Finalisation : 60
3.4.3 Étapes de l'Algorithme Génétique – Cas du problème 61
3.4.3.1 Phase Initiale 61
3.4.3.2 Boucle Évolutive 63
3.4.3.3 Finalisation 69
3.5 Conclusion 70

Chapitre 4 : Implémentation et test 72
4.1 Méthode exacte avec IBM-CPLEX 73
4.1.1 Configuration du solveur 73
4.1.2 Modèle mathématique et de la fonction d'affichage 73
4.1.3 Saisi des données sur le fichier dat 76
4.1.4 Discussions des résultats obtenus 77
4.2 Méthode approchée : Méta-Heuristique - l'Algorithme génétique 78
4.2.1 Choix de langage de programmation 78
4.2.1.1 Bibliothèques et APIs Utilisées 79
4.2.1.2 Avantages de Python 79
4.2.1.3 Inconvénients de Python 79
4.2.2 Paramétrisation de l'algorithme 80
4.2.2.1 Processus de test 80
4.2.2.2 Détermination l'approche de paramètres 81
4.2.3 Tests et validation 86
4.2.3.1 Tests 86
4.2.3.2 Validation 86
4.3 Comparaison des performances des méthodes 87
4.3.1 Méthode exacte vs Méthode Approchée 87
4.3.2 Analyse de performance 88
4.3.3 Interprétation des Résultats 91
4.3.3.1 Facteur de Décision pour le Choix de la Méthode 91
4.3.4 Conclusion 94
4.4 Fonctionnement de VRS (Vehicule Routing Solver) 95
4.4.1 Architecture logicielle 95

4.4.2 Fonctionnalités et interaction utilisateur 96
4.4.3 Intégration des résultats 99
4.4.4 Évaluation l'efficacité de Solveur VRS 102
4.4.5 Impact sur CPH Agriculture 105

Conclusion 107

Annexe : 108

Bibliographie 118

Résumé : 121

Tables des figures

FIG 1.1 – Logo de l'entreprise CPH Agriculture 14
FIG 1.2 – Diagramme de l'entreprise CPH Agriculture 15
FIG 1.3 – Emplacements des entrepôts de l'entreprise 20

FIG 2.1 – Problème des tournées de véhicules 23
FIG 2.2 – Hiérarchisation des variantes de problème VRP 26
FIG 2.3 – Classes de complexités de problèmes 36
FIG 2.4 – Les différentes classes de complexité 37
FIG 2.5 – Classification des méthodes de résolution 42
FIG 2.6 – Optima locaux et optimum global 43

FIG 3.1 – Les principaux éléments du logiciel CPLEX [22] 50
FIG 3. 2 – L'interface du logiciel CPLEX [24] 51
FIG 3.3 – L'arbre de recherche de Branch-and-Cut 52
FIG 3.4 – Opération génétique - Sélection 58
FIG 3.5 – Opération génétique - Croisement #1 58
FIG 3.6 – Opération génétique - Croisement #2 59
FIG 3.7 – Opération génétique - Mutation 59
FIG 3. 8 – Les étapes de l'algorithme génétique 60
FIG 3.9 – Structure de la solution #1 61
FIG 3.10 – Structure de la solution #2 61
FIG 3.11 – Initialisation de la population 62
FIG 3.12 – Méthode BRBAX 64
FIG 3.13 – Méthode BCR 65
FIG 3.14 – Rupture de la capacité #1 66
FIG 3.15 – Rupture de la capacité #2 67
FIG 3.16 – Rupture de la capacité #3 67
FIG 3.17 – Rupture de la capacité #4 68
FIG 3.18 – Evolution de coût total en fonction de génération 70

FIG 4.1 – Résultat statistique de la régression linéaire sur les données de paramètres de l'algorithme génétique 83
FIG 4.2 – Résultat Statistique de la régression linéaire sur les données de la méthode approchée 92
FIG 4.3 – Résultat Statistique de la régression linéaire sur les données de la méthode exacte 92
FIG 4.4 – L'interface de Solveur VRS 95
FIG 4.5 – Les Onglets de VRS 96
FIG 4.6 – VRS - Onglet « Clients » 96
FIG 4.7 – VRS - Onglet « Entrepôts » 97
FIG 4.8 – VRS - Onglet « Commandes » 97

FIG 4.9 – VRS - Onglet « Véhicules »....98
FIG 4.10 – VRS - Onglet « Solveur »....99
FIG 4.11 – VRS - Onglet « Rapport »....100
FIG 4.12 – VRS - Onglet « Carte Géographique »....100
FIG 4.13 – VRS - La tournée optimal tracée sur la carte géographique....101

Liste des tableaux

TAB 1.1 – Les taches associées à chaque département....16
TAB 1.2 – Types de fournitures distribués par l'entreprise....17
TAB 1.3 – Types de véhicules utilisés par l'entreprise....19

TAB 3.1 – Exemple sur les données d'entrepôts....55
TAB 3.2 – Exemple sur les données de clients....56
TAB 3.3 – Exemple sur les données de commandes....56
TAB 3.4 – Exemple de la matrice de distances....57

TAB 4.1 – Meilleur paramètres sélectionnés pour chaque scénario....82
TAB 4.2 – Classification des paramètres....82
TAB 4.3 – Résultat de tests effectué sur différentes configurations....86
TAB 4.4 – Comparaison entre méthode exacte et approchée pour des données de petite taille....89
TAB 4.5 – Comparaison entre méthode exacte et approchée pour des données de moyenne taille....89
TAB 4.6 – Comparaison entre méthode exacte et approchée pour des données de grande taille....89
TAB 4.7 – Données de commandes de programme 01....103
TAB 4.8 – Données de véhicules disponibles pour le programme 01....103
TAB 4.9 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 01....103
TAB 4.10 – Données de commandes de programme 02....104
TAB 4.11 – Données de véhicules disponibles pour le programme 02....104
TAB 4.12 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 02....104
TAB 4.13 – Données de commandes de programme 03....105
TAB 4.14 – Données de véhicules disponibles pour le programme 03....105
TAB 4.15 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 03....105

Liste de l'abréviation

API	*Application Programme Interface*
BCR	*Best Cost Route Crossover*
BRBAX	*Best Route Better Adjustment Recombination*
CPU	*Central Processing Unit*
CVRP	*Capacitated Vehicle Routing Problem*
DCVRP	*Distance Constrained Vehicle Routing Problem*
ERP	*Enterprise Resource Planning*
FAO	*Food and Agriculture Organisation*
FSVRP	*Fleet Size Vehicle Routing Problem*
GIL	*Global Interpreter Lock*
HFVRP	*Heterogeneous Fleet Vehicle Routing Problem*
MDHFVRP	*Multi Depots Heterogeneous Fleet Vehicle Routing Problem*
MDVRP	*Multi Depots Vehicle Routing Problem*
mTSP	*Multiple Travel Salesman Problem*
NP	*Non deterministic Polynomial*
ONU	*Organisation des Nations Unies*
OPL	*Optimization Programming Language*
OS	*Operation System*
OSRM	*Open Source Routing Machine*
OVRP	*Open Vehicle Routing Problem*
PVC	*Problème de Voyage de Commerce*
RAM	*Random Access Memories*
SAT	*Satisfiability Problem*
SDVRP	*Split Delivery Vehicle Routing Problem*
TCVRP	*Time Constrained Vehicle Routing Problem*
TSP	*Travel Salesman Problem*
VRP	*Vehicle Routing Problem*
VRPPD	*Vehicle Routing Problem with Pickup and Delivery*
VRPTW	*Vehicle Routing Problem with Time Windows*
VRS	*Vehicle Routing Solver*

Introduction générale :

La sécurité alimentaire est une préoccupation mondiale essentielle, définie par l'Organisation des Nations Unies pour l'alimentation et l'agriculture (FAO) lors du Sommet mondial de l'alimentation en 1996 comme « l'accès de tous à tout moment à une nourriture suffisante, saine et nutritive permettant de satisfaire leurs besoins énergétiques et leurs préférences alimentaires pour mener une vie saine et active ». Dans un contexte de croissance démographique rapide, de changement climatique, et de pressions économiques, garantir la sécurité alimentaire des populations est l'un des objectifs de développement durable fixés par l'ONU et qui est devenu un défi complexe et multidimensionnel. [1]

Le transport efficace des intrants et matériels agricoles est un maillon essentiel de la chaîne de production alimentaire. En effet, des systèmes de transport performants pour l'acheminement des semences, engrais et pesticides jouent un rôle crucial pour assurer la productivité et la rentabilité du secteur agricole. Un réseau de transport bien conçu permet de livrer les intrants nécessaires aux exploitations agricoles au bon moment et au bon endroit, cela évite les pénuries et les retards qui peuvent gravement compromettre les récoltes et les rendements.

L'impact économique du transport agricole est indéniable, car il joue un rôle crucial dans la réduction des coûts d'acheminement et de stockage des intrants agricoles réduisant ainsi les prix finaux des produits agricoles pour les consommateurs. En optimisant les itinéraires et en réduisant les délais de livraison, le transport agricole permet ainsi d'améliorer la rentabilité des exploitations. [2]

De plus, en rationalisant les processus de transport, les acteurs du secteur peuvent mieux anticiper les fluctuations des prix des matériaux et des intrants, garantissant ainsi une meilleure gestion financière. [3]

Mais au-delà de l'enjeu de la sécurité alimentaire, le transport occupe une place centrale dans les réflexions sur la transition agro-écologique et de la durabilité des systèmes agricoles et alimentaires, car il représente une part importante des émissions de gaz à effet de serre du secteur agricole. D'où l'optimisation des itinéraires d'acheminement des matériaux et intrants agricoles et l'adoption de modes de transport plus écologiques s'avèrent essentielles pour réduire significativement l'empreinte carbone de la production agricole. Dans le but d'assurer la durabilité du transport agricole dans le contexte actuel de la sensibilisation à l'environnement. [4]

Des réseaux de transport agiles et flexibles sont un impératif pour l'agriculture de demain. Car derrière chaque récolte se cache un véritable défi de transport pour acheminer au bon moment et au bon endroit les semences, engrais, pesticides, équipements et autres intrants indispensables aux exploitations agricoles.

C'est dans ce contexte que s'inscrit ce mémoire, dont la partie pratique a été effectué au sein l'entreprise CPH Agriculture ou nous avons eu l'opportunité de voir de près les différentes méthodes de transport et d'acheminement des matériaux et intrants agricoles utilisés, et de constater l'existence de nombreux défis rencontrés lors de la planification des itinéraires de livraison pour les clients.

L'objectif de ce mémoire est d'étudier en profondeur ces défis dans le but de minimiser les couts de transports tout en assurant la disponibilité des intrants et d'équipements pour tous les clients afin d'améliorer l'efficacité du réseau de distribution de l'entreprise.

Pour une meilleure organisation de ce travail, nous avons choisi de le répartir en quartes chapitres dont le premier parle de : L'Historique de l'entreprise CPH Agriculture, ses activités et zones de distribution ainsi que sa politique de distribution et les spécificités et défis qui lui sont liés

Dans Le deuxième chapitre : nous nous pencherons sur l'étude théorique des problèmes de routage des véhicules (VRP) dans le contexte du transport agricole en mettant la lumière sur : leurs évolutions historiques, les modèles mathématiques qui leurs sont associés ainsi que les différentes méthodes d'optimisation combinatoire utilisés pour leurs résolution.

Quant au troisième chapitre, il explique en détail les méthodes que nous avons adoptées pour résoudre les problèmes de transport spécifiques à l'entreprise CPH Agriculture et expose les résultats obtenus et leur analyse, tout en montrant les avantages et les limitations de chaque approche.

Enfin dans quatrième chapitre, nous présentons le programme informatique et son interface graphique (GUI) que nous avons conçus. L'interface graphique permet à l'opérateur de CPH de planifier les itinéraires de transport que nous avons choisis et a pour but de faciliter la tâche au personnel chargé de planifier les expéditions des commandes aux clients demandeurs.

En s'appuyant sur une étude de cas concrète et des outils de résolution avancés, ce mémoire aspire à apporter une contribution significative à la littérature existante sur le transport et le routage de véhicules dans le secteur agricole et à offrir des solutions pragmatiques et efficaces pour l'amélioration du système de transport de CPH Agriculture.

Chapitre 1 :
Présentation de l'entreprise

1.1 Introduction :

La sécurité alimentaire est un enjeu crucial en Algérie et qui occupe une place centrale dans les politiques publiques du pays, a cet égard le gouvernement algérien a mis en place plusieurs programmes pour soutenir les distributeurs des intrants agricoles dont le leader sur la scène nationale est l'entreprise « CPH Agriculture », et cela dans le but de favoriser l'accès des agriculteurs notamment dans les zones rurales aux intrants et au matériels agricoles.

Dans ce chapitre nous allons présenter l'entreprise qui a fait l'objet de notre terrain de stage à savoir : « CPH Agriculture ».

1.2 L'entreprise CPH Agriculture :

CPH Agriculture est une filiale du groupe CPH, fondé en 1985, spécialisée dans la distribution des semences fourragères et des équipements destinés aux différentes activités agricoles, à savoir l'irrigation, la filtration, le pompage, l'outillage et le machinisme.

L'entreprise est dotée d'un réseau de distribution très actif et dynamique déployé sur les quatre coins de notre pays, elle est représentée par plusieurs points de vente qui assurent une disponibilité permanente des produits de grandes marques de renommé internationale qu'elle représente en Algérie.

CPH Agriculture se projette dans l'avenir en tant que leader dans son domaine d'activité, une ambition traduite par des actions engagées aujourd'hui dans l'objectif de promouvoir d'avantage son réseau de distribution et sa qualité de service.

FIG 1.1 – Logo de l'entreprise CPH Agriculture

Source : cph-agro.com

1.3 Structure organisationnelle :

1.3.1 L'organigramme de l'entreprise :

L'organisation de l'entreprise CPH Agriculture est structurée d'une manière hiérarchique comme suit :

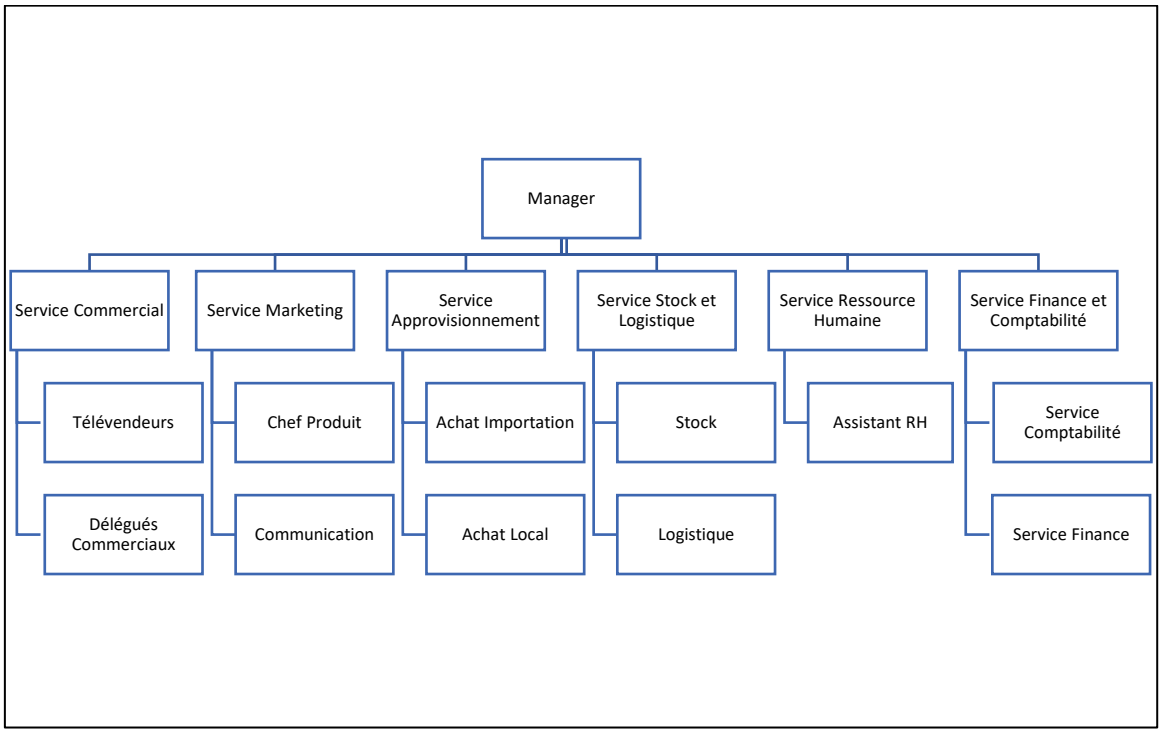


FIG 1.2 – Diagramme de l'entreprise CPH Agriculture

Source : Document interne de l'entreprise

1.3.2 Les missions de chaque département

Le tableau suivant récapitule les taches associées à chaque département de l'entreprise CPH Agriculture :

Département	Taches
Manager	- Assurer l'implantation et l'application des bonnes pratiques afin d'atteindre les objectifs - Élaborer une vision stratégique - Assurer la coordination entre les services
Service Commercial	- Mettre les objectifs commerciaux - Suivre la réalisation des objectifs - Augmenter la part de marché - Commercialiser les produits actuels - Suivre les créances

Service Marketing	- Faire des études de marché - Suivre les réclamations - Créer de nouveaux produits - Élaborer et réaliser un plan de communication - Organiser des événements - Participer aux foires
Service Approvisionnement	- Élaborer le plan d'approvisionnement - Réaliser le plan d'approvisionnement - Faire des achats - Trouver les bons fournisseurs - Suivre les procédures d'achat
Service Stock et Logistique	- Assurer la bonne gestion des stocks - Préparer et contrôler les commandes - Recevoir les achats - Suivre les états de stock - Expédier les commandes vers les bons clients et dans les brefs délais - Suivre la gestion avec les sous-traitants
Service RH	- Recruter les bons éléments - Former le personnel - Suivre la gestion des carrières - Suivre le personnel
Service Finance et Comptabilité	- Assurer la bonne gestion des flux monétaires - Suivre les créances des clients et les dettes fournisseurs - Élaborer les états financiers - Suivre la comptabilité générale de l'entreprise - Suivre la fiscalité - Élaborer les tableaux de bord

TAB 1.1 – Les taches associées à chaque département

1.4 Politique de distribution de l'entreprise CPH Agriculture :

La politique de distribution de l'entreprise CPH Agriculture est un élément clé de sa stratégie commerciale. CPH Agriculture possède trois dépôts de distribution situés stratégiquement à Alger, Oran, et Biskra, permettant ainsi de couvrir efficacement l'ensemble du territoire national et de répondre aux besoins de leurs clients répartis dans toutes les régions du pays.

Pour assurer la livraison des commandes de ses clients, CPH Agriculture dispose de plusieurs flottes de différentes tailles, et différents types de véhicules. Cependant, la planification actuelle des livraisons, gérée manuellement, est loin d'être optimale. Cette situation a mis en évidence une problématique liée aux inefficacités inhérentes à la gestion humaine de la distribution.

En effet, la planification manuelle des livraisons, bien qu'elle soit assistée par des logiciels comme l'ERP « Odoo », reste souvent lente, sujette à des erreurs humaines et incapable de s'adapter en temps réel aux aléas du trafic ou aux changements de dernière minute. Cela se traduit par des retards dans les livraisons, des coûts de transport plus élevés et une insatisfaction croissante des clients.

1.5 Disponibilité des ressources et des infrastructures

❖ Types de fournitures agricoles

CPH Agriculture propose une large gamme de fournitures agricoles, incluant :

- Engrais : divers types d'engrais chimiques et organiques adaptés à différentes cultures.
- Semences : une variété de semences pour les cultures maraîchères, céréalières, et autres.
- Produits phytosanitaires : pesticides, herbicides, et fongicides pour protéger les cultures.
- Matériel agricole : outils, équipements, et machines nécessaires aux différentes étapes de la production agricole.

Type de Produit	Produits
Semences	
Matériels	

TAB 1.2 – Types de fournitures distribués par l'entreprise

❖ Types de Véhicules disponibles

Pour la distribution, CPH Agriculture utilise différents types de véhicules afin de répondre aux besoins variés des livraisons :

- Camions légers et Véhicules Utilitaires : pour les livraisons rapides et les petits volumes.
- Camions moyens : pour les volumes moyens, souvent utilisés pour les livraisons régionales.
- Poids lourds : pour les grands volumes et les longues distances.

Type de Véhicule	Véhicule
Utilitaire	
Fourgon	
Camion 2.5T	

Camion 3.5T	
Camion 4T	
Camion 5T	
Camion 6T	
Camion 7T	
Camion 10T	
Camion 20T	

TAB 1.3 – Types de véhicules utilisés par l'entreprise

Note : Les véhicules de type Camion sont classifiés par leur poids (Tonnage).

❖ Emplacements des entrepôts

Les trois dépôts principaux de CPH Agriculture sont situés comme suit :

- Alger : Situé au centre du pays, ce dépôt dessert principalement les régions du nord et de l'est.
- Oran : Couvrant la région ouest, ce dépôt assure les livraisons dans les zones côtières et intérieures de l'ouest.
- Biskra : Situé au sud, ce dépôt dessert les régions désertiques et agricoles du sud du pays.

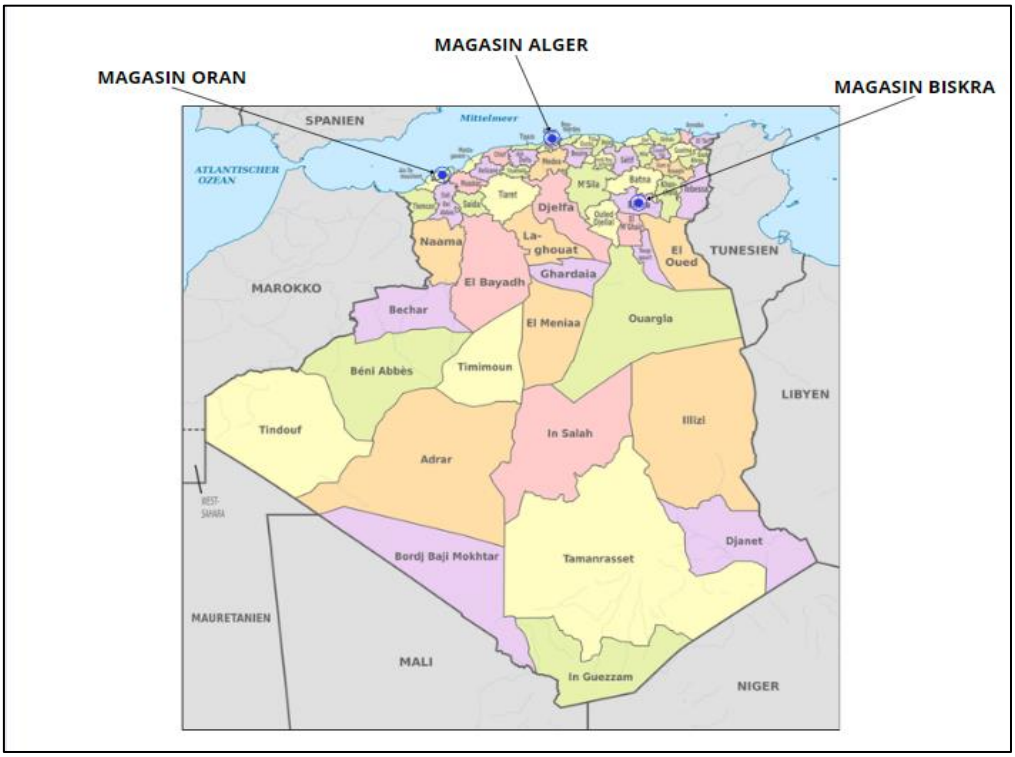


FIG 1.3 – Emplacements des entrepôts de l'entreprise

- **Problématique de Mémoire**

Comment pouvons-nous ajuster la méthode de planification des tournées de livraison de notre entreprise en intégrant de nouvelles méthodes plus pratiques pour maximiser leur efficacité ?

L'objectif de notre travail dans ce mémoire est de résoudre cette problématique qui consiste à optimiser les opérations de distribution en mettant en œuvre nos connaissances théoriques dans le but de réduire les coûts de transport tout en respectant les obligations de livraison et les contraintes opérationnelles.

1.6 Conclusion :

CPH Agriculture vise à améliorer l'efficacité de sa chaîne logistique dans le but de renforcer la satisfaction client et de maintenir sa compétitivité sur le marché national.

Nous avons fait dans ce chapitre la connaissance de l'entreprise CPH Agriculture, son domaine d'activité ainsi que sa structure hiérarchique, nous avons aussi peut constater la présence d'une problématique d'optimisation que nous allons explorer dans le chapitre suivant d'un point de vue théorique.

Chapitre 2 :
Problème des tournes de véhicules

2.1 Fondements théoriques du Problème des tournées de véhicules :

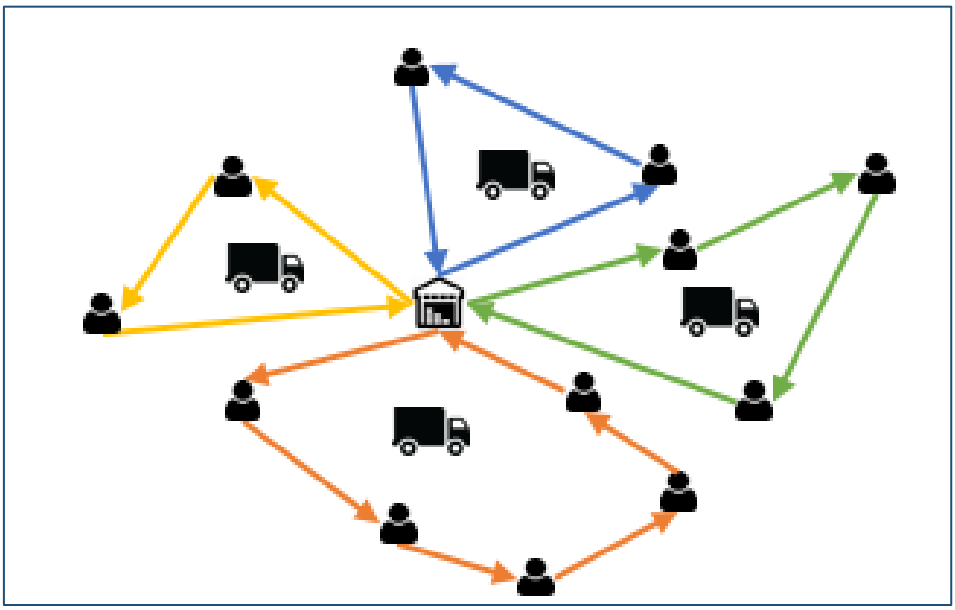


FIG 2.1 – Problème des tournées de véhicules

2.1.1 Introduction au VRP

2.1.1.1 Historique du VRP

Le problème des tournées de véhicules (Vehicle Routing Problem, VRP) a été introduit pour la première fois en 1959 par George Dantzig et John Ramser dans le cadre d'une application de livraison de carburant. Depuis lors, il est devenu l'un des problèmes d'optimisation combinatoire les plus étudiés dans la recherche opérationnelle [9].

2.1.1.2 Définition du VRP

Le VRP consiste à concevoir des tournées optimales pour une flotte de véhicules, partant d'un ou plusieurs dépôts, afin de desservir un ensemble de clients dispersés géographiquement, tout en minimisant les coûts totaux de transport. Les coûts peuvent inclure la distance totale parcourue, le temps de trajet, le nombre de véhicules utilisés, etc [6].

2.1.1.3 Importance du VRP

Le VRP a une importance cruciale dans de nombreux domaines, notamment la logistique, la distribution, les services à domicile et la collecte des déchets. Il permet d'optimiser l'utilisation des ressources (véhicules, carburant, main-d'œuvre) et de réduire les coûts opérationnels, tout en améliorant le niveau de service aux clients. De plus, En minimisant les distances parcourues, le problème des tournées de véhicules (VRP) permet non seulement de réduire les émissions de gaz à effet de serre et la congestion routière, mais aussi de limiter la dégradation des infrastructures. Cette dernière est particulièrement importante dans le cas des poids lourds, dont les arrêts fréquents en cas d'embouteillages accélèrent l'usure des routes. [6]

2.1.2 Classification de VRP

La classification des problèmes de tournées de véhicules (VRP) varie selon les auteurs. Selon Ghiani et al. (2003), les VRP peuvent être classés en fonction des caractéristiques des données d'entrée. Un VRP est considéré comme statique si toutes les données nécessaires à la planification ne dépendent pas du temps, sinon il est qualifié de dynamique. Un VRP est déterministe si toutes les données sont connues avec précision, sans incertitude, et stochastique dans le cas contraire. [7]

On distingue ainsi quatre types de VRP :

2.1.2.1 VRP Statique

- VRP statique déterministe : toutes les données sont connues à l'avance et le facteur temps n'est pas pris en compte explicitement [7].
- VRP statique stochastique : les tournées des véhicules sont planifiées au début de l'horizon de planification, avant que les données incertaines (demandes, temps de service, temps de trajet) ne soient connues. L'incertitude sur ces données rend généralement impossible de satisfaire toutes les contraintes pour toutes les réalisations des variables aléatoires. Si l'incertitude affecte les contraintes mais que la fonction objective est déterministe, on peut exiger que les contraintes soient satisfaites avec une certaine probabilité [7].

2.1.2.2 VRP Dynamique

- VRP dynamique déterministe : les données confirmées sont connues à l'avance, seules quelques informations dépendent du temps [7].

❖ VRP dynamique stochastique : les données incertaines sont représentées par des variables aléatoires. Par exemple, les demandes des clients peuvent suivre une loi de distribution de type Poisson [7].

Les divers événements peuvent conduire à modifier la planification initiale : arrivée d'une nouvelle demande, panne de véhicule, retard dû aux embouteillages. Chaque événement doit être traité conformément aux plans du système de planification des tournées.

2.1.3 Variantes du VRP

Avant d'aborder les différentes variantes du problème des tournées de véhicules (VRP), il est essentiel de comprendre le problème du voyageur de commerce (Traveling Salesman Problem, TSP), qui constitue la base du VRP.

Le TSP consiste à trouver le chemin le plus court pour visiter un ensemble de villes en passant une seule fois par chacune et en revenant au point de départ, avec un seul véhicule. Il peut être vu comme un cas particulier du VRP lorsqu'un seul véhicule sans contrainte de capacité est considéré.

Le VRP généralise le TSP en considérant une flotte de véhicules basée à un ou plusieurs dépôts, devant desservir un ensemble de clients. De nombreuses variantes du VRP ont ensuite été développées pour prendre en compte diverses contraintes supplémentaires rencontrées dans les applications réelles :

❖ Variantes liées aux véhicules :

- Problème des tournées de véhicules avec capacité limitée (Capacitated VRP, CVRP) : Chaque véhicule a une capacité maximale limitée.
- Problème des tournées de véhicules avec flotte hétérogène (Heterogeneous Fleet VRP, HFVRP) : Les véhicules ont des capacités et des coûts d'utilisation différents.
- Problème des tournées de véhicules avec flotte limitée (Fleet Size VRP, FSVRP) : Le nombre de véhicules est limité

❖ Variantes liées aux dépôts :

- Problème des tournées de véhicules avec dépôts multiples (Multi-Depot VRP, MDVRP) : Plusieurs dépôts à partir desquels les véhicules partent et reviennent.
- Problème des tournées de véhicules avec flotte ouverte (Open VRP, OVRP) : Les véhicules ne sont pas tenus de revenir au dépôt.

❖ Variantes liées aux contraintes temporelles :

- Problème des tournées de véhicules avec fenêtres de temps (VRP with Time Windows, VRPTW) : Chaque client doit être desservi dans une fenêtre de temps spécifique.

- Problème des tournées de véhicules avec contraintes de temps (Time Constrained VRP, TCVRP) : Une limite maximale de temps est imposée pour chaque tournée [8].

❖ Variantes liées au type de service :

- Problème des tournées de véhicules avec ramassage et livraison (VRP with Pickup and Delivery, VRPPD) : Certains clients doivent être livrés, d'autres ramassés.

- Problème des tournées de véhicules avec livraison fractionnée (Split Delivery VRP, SDVRP) : Les demandes des clients peuvent être satisfaites par plusieurs véhicules.

❖ Variantes liées aux contraintes de distance :

- Problème des tournées de véhicules avec contraintes de distance (Distance Constrained VRP, DCVRP) : Une limite maximale de distance est imposée pour chaque tournée [9].

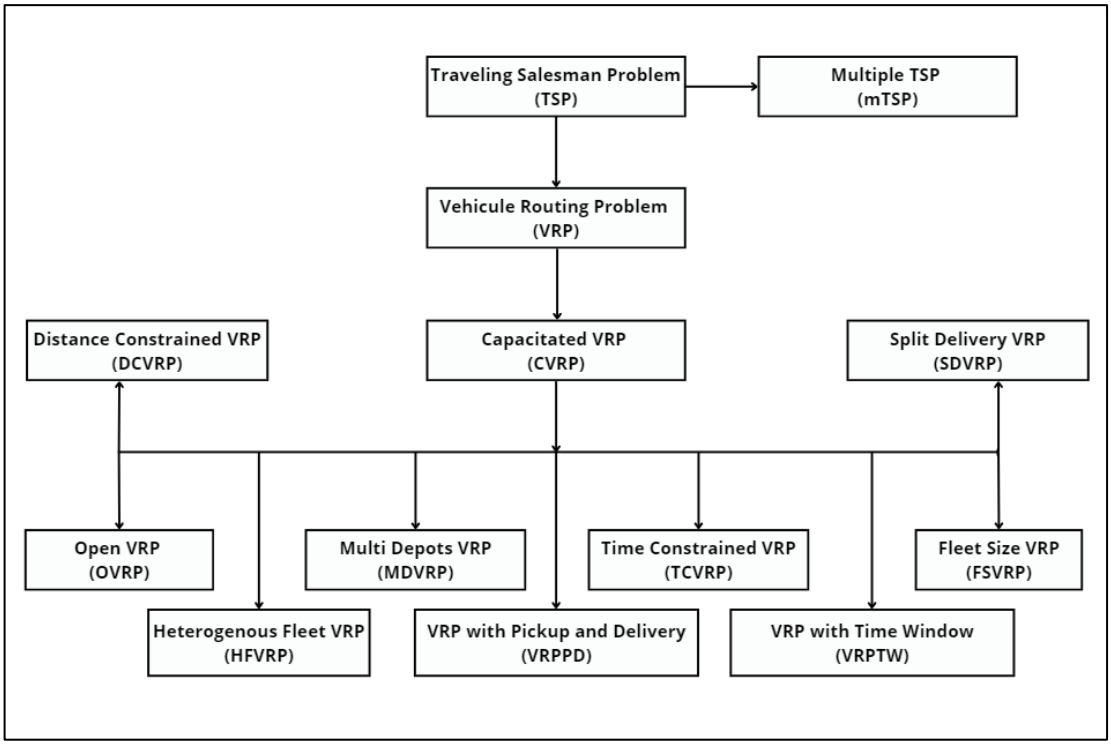


FIG 2.2 – Hiérarchisation des variantes de problème VRP

2.1.4 Revue de littérature

La revue de littérature sur le problème des tournées de véhicules (VRP) et ses variantes révèle une riche diversité d'approches de résolution proposées par les chercheurs au fil des années. Face à la complexité de ces problèmes d'optimisation combinatoire, de nombreuses méthodes ont été explorées, des plus classiques aux plus avancées, dans le but de trouver des solutions efficaces pour différents contextes applicatifs.

Les méthodes exactes, telles que la programmation linéaire en nombres entiers ou les algorithmes de type branch-and-bound, branch-and-cut, garantissent l'optimalité des solutions mais sont limitées par la taille des instances traitables. Par conséquent, des méthodes approchées, comprenant les heuristiques constructives et d'amélioration ainsi que les méta-heuristiques comme les algorithmes génétiques, le recuit simulé ou la recherche tabou, ont été largement étudiées pour obtenir des solutions de bonne qualité dans un temps raisonnable.

Les problèmes réels étant souvent contraints par des spécificités opérationnelles, de nombreuses variantes du VRP ont été considérées, telles que les problèmes avec flotte hétérogène, fenêtres de temps, contraintes de distance/durée, ou encore les problèmes de tournées avec ramassage et livraison. Pour chacune de ces variantes, différentes approches ont été explorées, comme illustré par les exemples suivants :

Quelques exemples de problèmes similaires au VRP et les méthodes qui ont été utilisées pour les résoudre :

❖ Problèmes de tournées avec collecte et livraison (Vehicle Routing Problems with Pickup and Delivery - VRPPD) :

- Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008) ont développé une recherche de voisinage étendue adaptative sophistiquée pour le problème de ramassage et de livraison avec plusieurs types de contraintes, incluant les capacités de manutention et les fenêtres de temps de service [10].

❖ Problèmes de tournées de véhicules avec flotte hétérogène (Heterogeneous Vehicle Routing Problems - HFVRP) :

- Baldacci, R., Bartolini, E., & Mingozzi, A. (2011) ont introduit une méthode exacte basée sur la relaxation de routes pour résoudre le HFVRP, démontrant des économies significatives dans le déploiement des véhicules [11].

Problèmes de tournées de véhicules avec fenêtres de temps (Vehicle Routing Problems with Time Windows - VRPTW) :

- Bräysy, O. et Gendreau, M. (2005) ont appliqué une heuristique de recherche tabou pour le VRP avec fenêtres de temps, en se concentrant spécifiquement sur la minimisation de la durée totale des tournées pour des services de livraison urbains [12].

❖ Problèmes de tournées de véhicules avec contraintes de distance/durée (Distance Constrained VRP) :

- Laporte, G., & Nobert, Y. (1987) ont abordé le problème de routage de véhicules avec contraintes de distance, où les itinéraires sont optimisés pour adhérer strictement aux limites de distance, essentiel pour les entreprises de logistique gérant des volumes de livraison quotidiens sur de grandes zones géographiques [13].

Ces exemples illustrent la variété des méthodes utilisées, allant des heuristiques classiques aux méta-heuristiques plus avancées, en fonction des spécificités du problème traité.

Cette riche littérature témoigne des efforts continus des chercheurs pour développer des techniques toujours plus performantes et adaptées aux défis posés par les applications réelles du VRP.

2.1.5 Modélisation du Problème

La modélisation du problème des tournées de véhicules comprend généralement une description détaillée du système de distribution, incluant les informations sur les clients, les dépôts, les véhicules et les contraintes spécifiques. Une formulation mathématique du problème est ensuite proposée, utilisant des variables de décision et des contraintes pour représenter les différents aspects du problème, tels que les coûts, les capacités, les fenêtres de temps, etc.

2.1.5.1 Modélisation descriptive du VRP

Le problème des tournées de véhicules (VRP) peut être décrit comme suit :

Étant donné un ensemble de véhicules basés à un ou plusieurs dépôts et un ensemble de clients géographiquement dispersés avec des demandes connues, l'objectif est de concevoir des tournées optimales pour la flotte de véhicules afin de desservir tous les clients au moindre coût total. Chaque client doit être visité exactement une fois par un seul véhicule, et chaque tournée doit commencer et se terminer au même dépôt.

Les données d'entrée comprennent généralement :

- La localisation des dépôts et des clients
- La demande de chaque client
- La capacité de chaque véhicule
- La matrice des coûts/distances entre les différents points

L'objectif typique est de minimiser le coût ou la distance totale parcourue par la flotte, bien que d'autres objectifs comme le nombre de véhicules utilisés ou le temps total de transport puissent également être considérés.

2.1.5.2 Modélisation mathématique fondamentale du VRP

Soit :

- N : l'ensemble des nœuds à visiter, indicé de 0 à n
- V : l'ensemble des véhicules, indicé de 1 à v
- C : la capacité de chaque véhicule
- c_{ij} : le coût/distance entre les nœuds i et j

Variables de décision :

$$x_{ij}^{k} = \begin{cases} 1 \; si \; véhicule \; k \; emprunte \; l'arc \; ij \\ 0 \; sinon \end{cases} \qquad \forall i, j = 0 \; \dots \; n \; ; \; k = 1 \; \dots \; v \qquad (2.1)$$

Fonction objectif :

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} c_{ij} \times x_{ij}^{k} \qquad (2.2)$$

Sous contraintes :

$$\sum_{k=1}^{v} \sum_{j=0}^{n} x_{ij}^{k} = 1 \qquad \forall i = 0 \dots n \qquad (2.3)$$

$$\sum_{k=1}^{v}\sum_{i=0}^{n} x_{ij}^{k} = 1 \qquad \forall j = 0 \dots n \qquad (2.4)$$

$$\sum_{i=0}^{n} x_{ij}^{k} = \sum_{i=0}^{n} x_{ji}^{k} \qquad \forall k = 1 \dots v, j = 1 \dots n \qquad (2.5)$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ij}^{k} \leq |S| - 1 \qquad \forall S \subset N;\ k = 1 \dots v \qquad (2.6)$$

La fonction-objectif (2.2) cherche à minimiser le coût total de la tournée parcourue. Les contraintes (2.3) et (2.4) imposent que le voyageur passe une et une seule fois par chaque nœud. La contrainte (2.5) assure la continuité de la tournée. La contrainte (2.6) élimine les sous-tours, c'est-à-dire des trajets passant par un sous-ensemble de nœuds sans retour au dépôt.

Pour les variantes comme le CVRP, HFVRP, MDVRP, des contraintes supplémentaires ou des variables de décision peuvent être ajoutées au modèle de base pour représenter les spécificités de chaque problème.

Exemple :

❖ CVRP : Contrainte de capacité sur chaque véhicule

- d_i : la demande du client i

Contraintes ajoutées :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_i x_{ij}^{k} \leq C \qquad \forall k = 1 \dots v \qquad (2.7)$$

La contrainte (2.7) limite la capacité du véhicule.

❖ HFVRP : Capacités et coûts différents pour chaque véhicule [31]

- C^k : Capacité du véhicule k
- f^k : Coût fixe d'utilisation du véhicule k
- y^k : 1 si le véhicule k est utilisé, 0 sinon

Fonction objectif :

$$min \sum_{k=1}^{v} f^k y^k + \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{v} c_{ij} \times x_{ij}^k \quad (2.8)$$

Contraintes ajoutées :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_i x_{ij}^k \leq C^k \qquad \forall k = 1 \dots v \quad (2.9)$$

La fonction-objectif (2.8) cherche à minimiser le coût total de la tournée parcourue additionné par le coût fixe des véhicules utilisés. La contrainte (2.9) limite la capacité du véhicule utilisé.

❖ MDVRP : Contraintes sur l'affectation des véhicules aux dépôts [21]

- M : Ensemble des dépôts 0 à m
- N : Ensemble des clients m+1 à n
- $m_i^k = 1$ si le véhicule k est affecté au dépôt i, 0 sinon

Contraintes ajoutées :

$$\sum_{i=0}^{m} m_i^k = 1 \qquad \forall k = 1 \dots v \quad (2.10)$$

$$\sum_{k=1}^{v} m_i^k \times x_{ij}^k \geq x_{ij}^k \qquad \forall i = 0 \dots m;\ j = m+1 \dots n \quad (2.11)$$

La contrainte (2.10) impose que chaque véhicule utilisé démarre par un seul entrepôt. La contrainte (2.11) assure le démarrage du véhicule de l'entrepôt

❖ VRPTW : Contraintes sur les fenêtres de temps de chaque client [32]

- e_i : Temps de début de service au plus tôt pour le client i
- l_i : Temps de fin de service au plus tard pour le client i
- t_{ij} : Temps de trajet entre i et j

- w_i^k : Temps d'arrivée du véhicule k au client i

Contraintes ajoutées :

$$e_i \leq w_i^k \qquad \forall i = 0 \dots n;\ k = 1 \dots v \qquad (2.12)$$

$$w_i^k \leq l_i \qquad \forall i = 0 \dots n;\ k = 1 \dots v \qquad (2.13)$$

$$w_i^k + t_{ij} - M\left(1 - x_{ij}^k\right) \leq w_i^k \qquad \forall i, j = 0 \dots n;\ k = 1 \dots v \qquad (2.14)$$

La Contrainte (2.12) garantit que les véhicules ne peuvent pas commencer le service chez un client avant le début de la fenêtre de temps spécifiée. La Contrainte (2.13) garantit que les véhicules doivent arriver chez les clients avant la fin de la fenêtre de temps spécifiée. La Contrainte (2.14) lie les temps d'arrivée des véhicules aux clients successifs.

❖ DCVRP : Contrainte sur la distance totale maximale de chaque tournée

- D_{max} : Distance maximale autorisée par tournée
- c_{ij} : Distance entre les nœuds i et j

Contraintes ajoutées :

$$\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} \times x_{ij}^k \leq D_{max} \qquad \forall k = 1 \dots v \qquad (2.15)$$

La Contrainte (2.15) limite la distance totale de la tournée

❖ TCVRP : Contrainte sur la durée totale maximale de chaque tournée

- T_{max} : Distance maximale autorisée par tournée
- t_{ij} : Temps de trajet entre i et j

Contraintes ajoutées :

$$\sum_{i=0}^{n}\sum_{j=0}^{n} t_{ij} \times x_{ij}^k \leq T_{max} \qquad \forall k = 1 \dots v \qquad (2.16)$$

La Contrainte (2.16) limite le temps total de la tournée

❖ VRPPD : Contrainte de livraison et de ramassage [33]

- D ⊆ N : Ensemble des nœuds de livraison
- P ⊆ N : Ensemble des nœuds de ramassage
- $d_{ij}^k = 1$ si le véhicule k dessert l'arc (i, j) pour une livraison, 0 sinon
- $p_{ij}^k = 1$ si le véhicule k dessert l'arc (i, j) pour un ramassage, 0 sinon

Contraintes ajoutées :

$$\sum_{k=1}^{v}\sum_{j=0}^{n} d_{ij}^k = 1 \qquad \forall i \in \mathrm{D} \qquad (2.17)$$

$$\sum_{k=1}^{v}\sum_{j=0}^{n} p_{ij}^k = 1 \qquad \forall i \in \mathrm{P} \qquad (2.18)$$

La contrainte (2.17) garantit que chaque nœud de livraison est visité par exactement un véhicule. La contrainte (2.18) garantit que chaque nœud de ramassage est visité par exactement un véhicule.

❖ SDVRP : Contrainte de livraison fractionnée

Contraintes remplacées :

$$\sum_{k=1}^{v}\sum_{j=0}^{n} x_{ij}^k \geq 1 \qquad \forall i = 0 \dots n \qquad (2.19)$$

La contrainte (2.19) garantit que chaque client peut être visité plusieurs fois

❖ FSVRP : Contrainte de flotte de taille limitée

- y^k : 1 si le véhicule k est utilisé, 0 sinon

Contraintes ajoutées :

$$\sum_{k=1}^{v} y^k \leq |V| \tag{2.20}$$

Contraintes remplacées :

$$\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ij}^k \leq |S| - y^k \qquad \forall S \subset N, k = 1 \dots v \tag{2.21}$$

La contrainte (2.20) assure que le nombre de véhicules utilisés ne dépasse pas le nombre total de véhicules disponibles. La contrainte (2.21) empêche la formation de sous-tours

2.2 Optimisation Combinatoire

2.2.1 Introduction de l'Optimisation Combinatoire

Optimisation combinatoire, un domaine complexe de l'informatique qui traite de la recherche de la meilleure solution parmi un ensemble de solutions possibles. Ces problèmes sont présents dans de nombreux domaines, tels que la biologie, les mathématiques, l'électronique, la logistique et transports. En informatique, des exemples célèbres incluent le problème du PVC, le problème SAT, la planification de tâches et la coloration de graphes. [14]

2.2.1.1 Définition des problèmes d'optimisation combinatoire

Un problème d'optimisation combinatoire est défini par :

- Un ensemble de variables : Notées X = $\{x_1, x_1, x_1, \dots, x_n\}$, chacune avec un domaine de définition respectif D_i.
- L'espace de recherche : Représenté par ϒ, l'ensemble des valeurs possibles $(v_1, v_1, v_1, \dots, v_n)$ pour chaque variable, où $v_i \in D_i$.
- Des contraintes : Des règles qui limitent les valeurs que les variables peuvent prendre simultanément.
- Une fonction objectif : Notée F, qu'il faut minimiser ou maximiser.

L'objectif est de trouver une solution optimale, c'est-à-dire un ensemble de valeurs pour les variables qui minimise (ou maximise) la fonction objectif tout en respectant les contraintes.

2.2.1.2 Complexité des problèmes d'optimisation combinatoire

La difficulté de ces problèmes réside dans le nombre colossal de solutions possibles, qui augmente exponentiellement avec le nombre de variables et de contraintes. Cela rend l'analyse exhaustive du problème par un humain pratiquement impossible [15].

2.2.1.3 Définition des problèmes de décision

Un problème de décision est un cas particulier d'optimisation combinatoire où la réponse est binaire (vrai ou faux). Par exemple, déterminer si un graphe peut être colorié avec un nombre donné de couleurs est un problème de décision.

❖ Exemple de problème de décision : coloration de graphe

Soit un graphe G = (V, E) avec V un ensemble de sommets et E un ensemble d'arêtes. Le problème consiste à déterminer s'il est possible d'attribuer une couleur à chaque sommet de V de sorte qu'aucune arête ne relie deux sommets de la même couleur.

❖ Exemple de problème d'optimisation : planification de tâches

Soit un ensemble d'activités A à planifier sur un horizon temporel H. L'objectif est de trouver un ordre d'exécution des activités qui minimise leur temps total d'exécution ou l'utilisation de certaines ressources.

2.2.1.4 Planification sous contraintes

La planification sous contraintes est un type de problème d'optimisation combinatoire où les contraintes sont particulièrement importantes et complexes. Ces problèmes sont souvent utilisés pour modéliser des situations réelles où il faut prendre en compte de nombreuses restrictions et exigences.

2.2.1.5 Conclusion

Les problèmes d'optimisation combinatoire et la planification sous contraintes sont des domaines fascinants et complexes de l'informatique avec des applications dans une multitude de domaines. La recherche dans ce domaine vise à développer des algorithmes plus efficaces pour résoudre des problèmes de plus en plus complexes, et à mieux comprendre les propriétés fondamentales de ces problèmes.

2.2.2 Complexité algorithmique et classes de problèmes

La complexité algorithmique est une mesure théorique permettant de classer les problèmes d'optimisation en fonction de leur difficulté intrinsèque à être résolus [15]. On distingue principalement les classes de problèmes P, NP et NP-complet :

❖ Classe P :

Problèmes que l'on peut résoudre en un temps polynomial, c'est-à-dire dont la complexité est bornée par un polynôme en fonction de la taille de l'instance. Ces problèmes sont considérés comme "faciles" à résoudre. [15]

❖ Classe NP :

Problèmes pour lesquels on peut vérifier une solution proposée en un temps polynomial, mais pour lesquels on ne sait pas encore s'il existe un algorithme polynomial pour les résoudre. [15]

❖ Classe NP-complet :

Sous-ensemble des problèmes NP les plus difficiles. Un problème est NP-complet si tout problème NP peut se ramener à lui en un temps polynomial. Ces problèmes sont considérés comme intrinsèquement difficiles et on ne connaît pas à ce jour d'algorithme polynomial pour les résoudre de manière générale. [15]

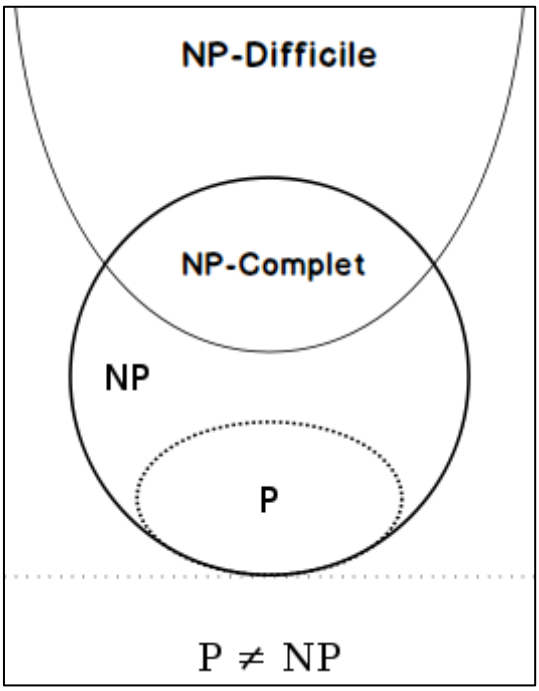


FIG 2.3 – Classes de complexités de problèmes

Source : https://en.wikipedia.org/wiki/NP_%28complexity%29

Note : pour le moment, aucune étude ne prouve que P = NP

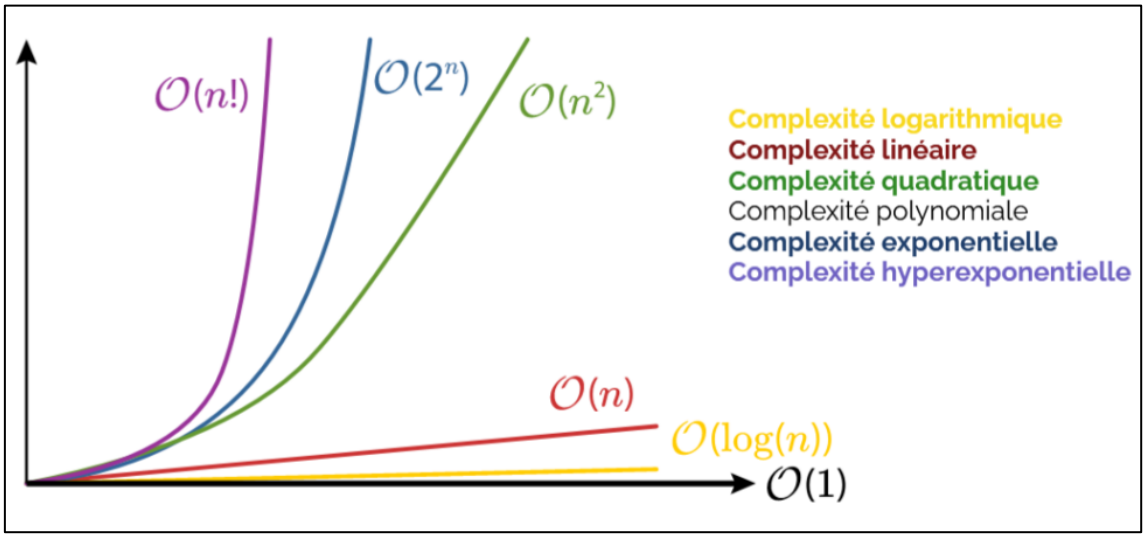


FIG 2.4 – Les différentes classes de complexité

Source : mathweb.fr/euclide/complexite-algorithmique/

Le problème des tournées de véhicules (VRP) fait partie de la classe des problèmes NP-complets. Sa complexité est de l'ordre de O(n!) [16], où n est le nombre de clients à desservir. Cela signifie que le temps de calcul pour trouver la solution optimale exacte augmente de manière exponentielle avec la taille du problème.

Cependant, certaines variantes peuvent avoir une complexité légèrement différente en raison des contraintes supplémentaires ajoutées. Par exemple :

- Le CVRP (Capacitated VRP) avec uniquement des contraintes de capacité reste NP-complet avec une complexité en O(n!).
- Le VRPTW (VRP with Time Windows) avec des fenêtres de temps est NP-complet au sens fort, ce qui le rend encore plus difficile que le VRP classique.
- Le HFVRP (Heterogeneous Fleet VRP) avec une flotte hétérogène reste NP-complet, mais pourrait avoir une complexité légèrement supérieure en fonction des contraintes supplémentaires liées à la flotte.

2.2.3 Méthodes de résolution

2.2.3.1 Méthodes exactes

Les méthodes exactes ont pour objectif de trouver la solution optimale du problème d'optimisation combinatoire en explorant de manière intelligente et complète l'espace des solutions réalisables. Elles reposent sur des techniques d'énumération implicite ou explicite couplées à des stratégies de séparation et d'évaluation pour élaguer les parties non prometteuses de l'espace de recherche.

❖ Programmation dynamique :

Cette méthode décompose le problème en sous-problèmes de taille réduite. Elle résout ces sous-problèmes de manière récursive, en mémorisant leurs solutions afin d'éviter les calculs redondants lors des appels récursifs ultérieurs [7].

❖ Branch-and-Bound/Branch-and-Cut :

Ces algorithmes subdivisent récursivement l'espace complet des solutions en sous-ensembles disjoints (branch). Pour chaque sous-ensemble, ils calculent des bornes inférieures et supérieures sur la valeur de la fonction objective (bound/cut). Les sous-ensembles dont les bornes prouvent qu'ils ne peuvent pas contenir une meilleure solution que la meilleure solution trouvée jusqu'à présent sont alors élagués de l'espace de recherche. [19]

2.2.3.2 Méthodes approchées

Contrairement aux méthodes exactes, les méthodes approchées ne garantissent pas l'optimalité de la solution obtenue, mais visent à trouver rapidement une bonne solution réalisable, en particulier pour les problèmes NP-complets de grande taille. Elles offrent un compromis entre la qualité de la solution et le temps de calcul nécessaire.

➢ Heuristiques :

Les heuristiques sont des algorithmes simples et rapides, généralement conçus pour un problème spécifique. Elles exploitent certaines caractéristiques du problème pour construire ou améliorer des solutions réalisables de manière efficace, sans garantir une solution optimale. Ces méthodes sont souvent utilisées lorsqu'une solution approximative mais acceptable est préférée à une solution exacte trop coûteuse en termes de calcul ou de ressources.

❖ Heuristiques constructives : Construisent une solution réalisable étape par étape en suivant une règle spécifique [15]. Par exemple :

- Plus proche voisin (Nearest Neighbor)

C'est une heuristique gloutonne simple qui construit une solution pas à pas en choisissant à chaque étape le nœud non visité le plus proche du dernier nœud ajouté. Elle commence par un nœud initial et ajoute les nœuds un par un jusqu'à ce que tous soient parcourus, formant un cycle hamiltonien dans le graphe. Cette méthode est rapide mais ne garantit pas une solution optimale. [18]

- Insertion du plus proche voisin (Nearest Insertion)

Cette heuristique constructive commence par une solution triviale contenant un seul nœud. Ensuite, elle insère un à un les nœuds restants dans la solution à l'emplacement qui minimise l'augmentation de la distance totale. Cet emplacement est déterminé en évaluant la distance la plus courte entre le nœud non visité le plus proche et l'ensemble des nœuds déjà insérés dans la tournée. L'ordre d'insertion des nœuds peut avoir un impact sur la qualité de la solution finale [19].

- Clarke & Wright

L'algorithme des économies de Clarke et Wright est une heuristique constructive élaborée spécifiquement pour le VRP. Elle commence avec une tournée par nœud, chaque tournée partant du dépôt, visitant un seul nœud et revenant au dépôt. Puis, elle fusionne de manière itérative les tournées permettant de réaliser les plus grandes économies, c'est-à-dire de diminuer le plus la distance totale parcourue. Ce processus se poursuit jusqu'à ce qu'aucune fusion ne soit plus possible [28].

❖ Heuristiques d'amélioration : Partent d'une solution réalisable existante (souvent construite par une heuristique constructive) et tentent de l'améliorer par des modifications locales dans son voisinage [19]. Par exemple :

- 2-opt

L'heuristique 2-opt est une méthode d'amélioration locale utilisée pour résoudre les problèmes de parcours optimaux, comme le voyageur de commerce. Elle part d'une solution réalisable initiale (un cycle hamiltonien) et tente de l'améliorer en retirant deux arcs non consécutifs du cycle et en reconnectant les deux segments restants de la meilleure façon possible, c'est-à-dire celle qui minimise la distance totale parcourue [18].

- λ-opt

L'heuristique λ-opt est une généralisation du 2-opt où λ arcs (λ >= 2) sont retirés du cycle plutôt que seulement deux. Les segments résultants sont ensuite ré-optimisés, c'est-à-dire reconnectés de la meilleure façon possible pour minimiser la distance totale. Plus λ est grand, plus l'espace de recherche est large mais aussi plus le temps de calcul est important [18].

- Or-opt

L'heuristique Or-opt (ou relocation) est une méthode d'amélioration spécifique aux problèmes impliquant plusieurs cycles (tournées), comme le VRP. Elle consiste à transférer un ou plusieurs nœuds d'une tournée à une autre, en insérant ces nœuds à l'emplacement qui minimise l'augmentation du coût total [18].

➢ Méta-heuristiques :

Les méta-heuristiques sont des algorithmes plus généraux et plus élaborés, s'appuyant souvent sur des processus naturels ou des techniques issues de l'intelligence artificielle. Elles guident la recherche de solutions de manière plus intelligente que les heuristiques simples, en explorant l'espace des solutions de manière plus globale et en évitant de rester bloquées dans des optimums locaux.

❖ Méta-heuristiques à solution unique :

Ces méta-heuristiques explorent l'espace de recherche en faisant évoluer une seule solution candidate à la fois, en utilisant généralement des techniques de voisinage et de recherche locale. Elles peuvent facilement converger vers un optimum local, mais sont souvent combinées avec des mécanismes pour s'en échapper [19], Par exemple :

- Recherche Locale (Local Search)

La recherche locale est une méta-heuristique qui explore le voisinage d'une solution en appliquant des modifications élémentaires définies par un opérateur de voisinage. À chaque itération, la solution courante est remplacée par la meilleure solution de son voisinage si elle est meilleure. Le processus se poursuit jusqu'à ce qu'un optimum local soit atteint, c'est-à-dire qu'aucune solution voisine n'améliore la solution courante [20].

- Recherche Tabou (Tabu Search)

La recherche tabou est une méta-heuristique qui guide une recherche locale améliorée en mémorisant les dernières solutions visitées dans une liste

tabou afin d'éviter les cycles. À chaque itération, le meilleur mouvement admissible (non présent dans la liste tabou) est appliqué, même s'il dégrade la solution. La liste tabou est mise à jour pour interdire les mouvements inverses pendant un certain nombre d'itérations [7].

- Recuit Simulé (Simulated Annealing)

Le recuit simulé s'inspire du processus de refroidissement utilisé en métallurgie. Il explore l'espace des solutions en acceptant des solutions de moindre qualité avec une probabilité décroissante au fil des itérations, contrôlée par un paramètre de "température". Cela permet d'éviter de rester bloqué dans des optimums locaux en autorisant des dégradations temporaires de la solution [7].

Optimum recherché : Optimum local (solution quasi-optimale) ou, au mieux, l'optimum global si le mécanisme permet d'échapper aux optimums locaux.

❖ Méta-heuristiques à population de solutions :

Ces méta-heuristiques font évoluer simultanément un ensemble de solutions candidates (une population) à chaque itération, plutôt que de se concentrer sur une seule solution. Elles s'inspirent généralement de phénomènes naturels comme l'évolution génétique, le comportement des colonies d'insectes, etc. Leur objectif est de trouver une solution optimale globale ou une bonne approximation de celle-ci, en évitant de rester bloquées dans des optimums locaux [19].

- Algorithme Génétique (Genetic Algorithm)

Est une méta-heuristique d'optimisation inspirée des processus génétiques naturels et de la théorie de l'évolution. Il fait évoluer une population de solutions candidates, appelées chromosomes ou individus, en appliquant des opérateurs stochastiques comme le croisement, la mutation et la sélection, afin de générer de meilleures solutions à chaque itération [7].

- Colonies de Fourmis (Ant Colony)

Les algorithmes de colonies de fourmis est une méta-heuristique d'optimisation inspirée du comportement de recherche de nourriture des fourmis réelles. Il simule le déplacement d'un ensemble d'agents (fourmis artificielles) qui construisent des solutions en déposant et en suivant des traces de phéromones virtuelles, représentant l'attractivité des composants de la solution [19].

- Essaim Particulaire (Particle Swarm)

L'optimisation par essaim particulaire est une méta-heuristique d'optimisation inspirée du comportement social des bancs d'oiseaux ou des nuées d'insectes. Elle modélise un essaim de particules qui se déplacent dans l'espace de recherche en ajustant leurs positions en fonction de leur meilleure position connue et de la meilleure position globale de l'essaim [20].

- Colonie d'Abeilles (Bee Colony)

Colonie d'abeilles est un algorithme d'optimisation bio-inspirée qui imite le comportement de butinage des abeilles réelles. Elle modélise une colonie d'abeilles artificielles qui explorent l'espace de recherche de manière aléatoire, échangent des informations sur les meilleures solutions trouvées et se déplacent vers ces bonnes régions pour une recherche plus approfondie.

Optimum recherché : Optimum global (solution optimale) ou une très bonne approximation de celui-ci.

Ces différentes méthodes approchées offrent une grande flexibilité et sont particulièrement adaptées aux problèmes d'optimisation combinatoire de grande taille et complexité, comme le problème des tournées de véhicules (VRP) et ses variantes.

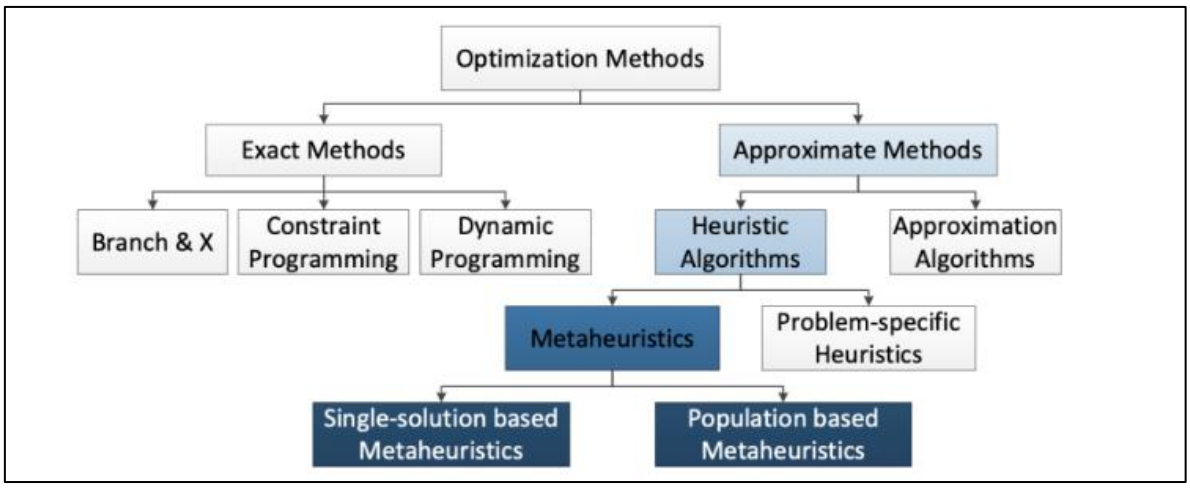


FIG 2.5 – Classification des méthodes de résolution

Source : ledatascientist.com/algorithme-genetique/

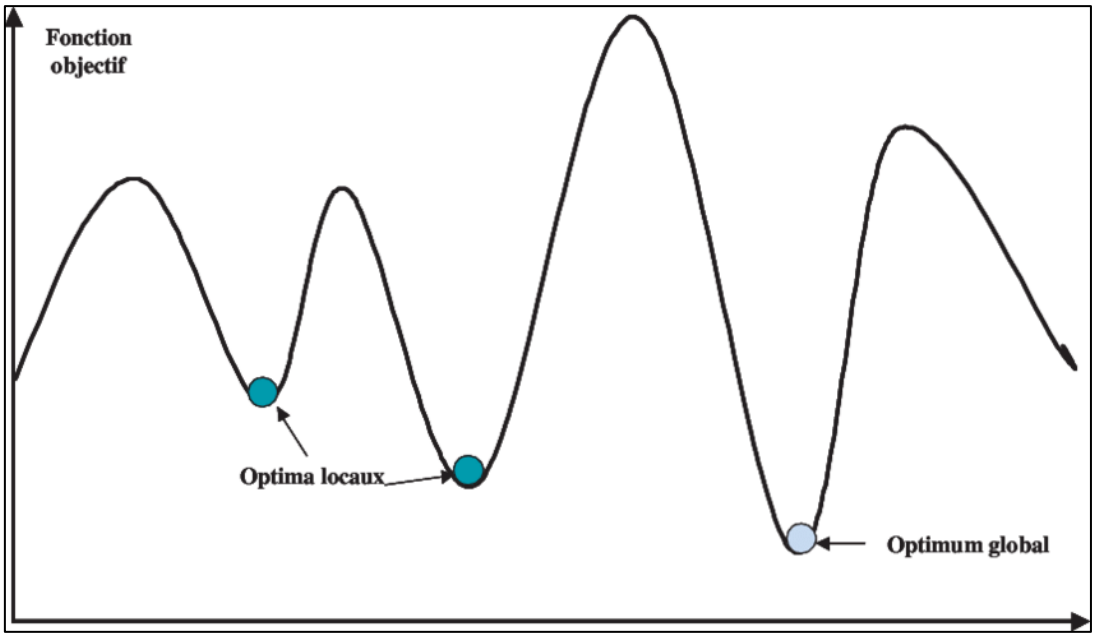


FIG 2.6 – Optima locaux et optimum global

Source : Logique floue et algorithmes génétiques pour le prétraitement de données de biopuces et la sélection de gènes - Scientific Figure on ResearchGate, researchgate.net/figure/Optima-locaux-et-optimum-global_fig11_41152980

2.2.3.3 Choix de la méthode en fonction de la complexité du problème

Le choix de la méthode d'optimisation la plus adaptée à un problème de VRP dépend de la complexité du problème et des ressources informatiques disponibles :

- Problèmes de petite taille

  Pour les problèmes de petite taille, il est possible d'utiliser des méthodes exactes afin de garantir la solution optimale. Cependant, si le temps de calcul est un facteur critique, des méthodes heuristiques ou des méta-heuristiques peuvent être une meilleure alternative.

- Problèmes de taille moyenne

  Pour les problèmes de taille moyenne, les méthodes heuristiques et les méta-heuristiques constituent généralement le meilleur choix. Elles offrent un bon compromis entre la qualité de la solution et le temps de calcul.

- Problèmes de grande taille

  Pour les problèmes de grande taille, les méthodes exactes ne sont généralement pas envisageables en raison de leur coût en temps de calcul prohibitif. Les méthodes heuristiques et les méta-heuristiques sont donc les seules options viables.

Chapitre 3 :
Approches de résolution

3.1 Introduction

L'entreprise CPH Agriculture est spécialisée dans la distribution de produits agricoles auprès de nombreux clients répartis sur un vaste territoire du pays. Pour assurer ses livraisons, elle dispose d'une flotte de véhicules hétérogènes, c'est-à-dire de capacités et de couts d'utilisations différentes, devant partir de plusieurs dépôts régionaux.

Le problème auquel CPH Agriculture est confrontée est une variante complexe du problème de tournées de véhicules (VRP), combinant à la fois les caractéristiques du MDVRP, l'entreprise doit construire des tournées de livraison optimales en tenant compte de la présence de plusieurs dépôts de départ pour les véhicules. Chaque client doit être affecté à un seul entrepôt et desservi par un seul véhicule parti de ce dernier, de plus, avec le HFVRP, l'entreprise doit composer avec une flotte hétérogène comportant différents types de véhicules, chacun ayant sa propre capacité de chargement. Cette contrainte ajoute une difficulté supplémentaire, car il faut assigner judicieusement les véhicules aux tournées en fonction de leurs capacités respectives et des demandes des clients.

L'objectif est donc de minimiser le coût total des tournées en déterminant un ensemble optimal permettant de livrer tous les clients à partir des dépôts disponibles, tout en respectant les capacités des véhicules et en assurant une affectation adéquate des clients aux dépôts. Résoudre ce problème combiné MDHFVRP est crucial pour CPH Agriculture afin d'optimiser ses opérations logistiques, réduire ses coûts de transport et améliorer son efficacité opérationnelle globale.

3.2 Modélisation du problème pour CPH Agriculture

3.2.1 Description du système de distribution

L'entreprise possède 3 dépôts situés à Alger, Oran et Biskra avec plusieurs flottes de véhicules aux caractéristiques différentes en termes de : capacité volumique, capacité en poids, coût fixe d'utilisation et coût variable unitaire. Des clients répartis sur le territoire national ont passé des commandes caractérisées pour chaqu'une par un poids et un volume. La politique de l'entreprise exige que chaque commande soit livrée au client avec un seul véhicule. Si les véhicules de l'entreprise ne suffisent pas, des véhicules supplémentaires peuvent être affrétés. De plus, si un dépôt a besoin de marchandises, celles-ci doivent lui être livrées à partir d'un autre dépôt (en le traitant alors comme un client).

L'objectif est de planifier les expéditions de manière à minimiser les coûts totaux de transport, tout en respectant les contraintes suivantes :

- Toutes les commandes des clients ou dépôts doivent être livrées
- Chaque commande doit être livrée au client par un seul véhicule
- Les capacités en poids et en volume des véhicules ne doivent pas être dépassées

Les décisions à prendre concernent l'affectation des commandes aux véhicules et la détermination des itinéraires de livraison. Les logisticiens de l'entreprise doivent donc trouver des réponses aux questions suivantes :

Quelles marchandises charger dans quel type de véhicules pour livrer quels clients et selon quel itinéraire ?

3.2.2 Formulation mathématique

Après une analyse détaillée du problème, nous avons conclu qu'il s'agit d'un problème de tournées de véhicules multi-dépôts avec une flotte hétérogène (MDHFVRP). Pour résoudre ce type de problème, Salhi et al. [21] ont proposé le modèle mathématique suivant :

Soit :

- $\boldsymbol{N}$: l'ensemble de tous les nœuds, respectivement les entrepôts et les clients, indicé de 1 à (m + n). Où :
  - les entrepôts sont indicés de 1 à m, où m est le nombre des entrepôts.
  - les clients sont indicés de (m + 1) à (m + n), où n est le nombre des clients.
- $\boldsymbol{P}$: ensemble des types des véhicules, indicé de 1 à v.
- $\boldsymbol{PO_i}$: poids de la commande pour le client i
- $\boldsymbol{VO_i}$: volume de la commande pour le client i
- $\boldsymbol{CP^k}$: capacité maximale de poids pour le type de véhicule k
- $\boldsymbol{CV^k}$: capacité maximale de volume pour le type de véhicule k
- $\boldsymbol{Cf^k}$: coût fixe d'utilisation pour le type de véhicule k
- $\boldsymbol{Cva^k}$: coût variable (kilométrique) pour le type de véhicule k
- $\boldsymbol{W^k}$: taille de la flotte pour le type de véhicule k.
- $\boldsymbol{D_{ij}}$: distance entre chaque paire de sommets i et j

Variables de décision :

$$X_{ij}^{kd} = \begin{cases} 1 \text{ Si type de vehicule k originaire d'entrepot d} \\ \qquad \text{parcout l'arc (i, j)} \\ 0 \text{ Sinon} \end{cases} \quad \begin{matrix} \forall i, j = 1 \dots n ; \\ \forall k = 1 \dots v \end{matrix} \tag{3.1}$$

Y_{ij} : Variable continu non négative qui donne le poids total restant dans le véhicule avant d'atteindre le nœud j en empruntant l'arc (i, j) (3.2)

Z_{ij} : Variable continu non négative qui donne le volume total restant dans le véhicule avant d'atteindre le nœud j en empruntant l'arc (i, j) (3.3)

Fonction objectif :

$$min \sum_{d=1}^{m} \sum_{k=1}^{K} \sum_{i=1}^{m} \sum_{j=m+1}^{m+n} Cf^k \times X_{ij}^{kd} + \sum_{d=1}^{m} \sum_{k=1}^{K} \sum_{i=1}^{m+n} \sum_{j=1}^{m+n} Cva^k \times D_{ij} \times X_{ij}^{kd} \tag{3.4}$$

Sous contraintes :

$$\sum_{d=1}^{m} \sum_{k=1}^{v} \sum_{i=1}^{m+1} X_{ij}^{kd} = 1 \qquad \forall j = m+1 \dots m+n \tag{3.5}$$

$$\sum_{d=1}^{m} \sum_{k=1}^{v} \sum_{j=1}^{m+1} X_{ij}^{kd} = 1 \qquad \forall i = m+1 \dots m+n \tag{3.6}$$

$$\sum_{i=1}^{m+1} X_{ij}^{kd} = \sum_{i=1}^{m+1} X_{ji}^{kd} \qquad \begin{matrix} \forall k = 1 \dots v; \forall j = 1 \dots m+n; \\ \forall d = 1 \dots m \end{matrix} \tag{3.7}$$

$$\sum_{i=1}^{m} \sum_{j=m+1}^{m+n} Y_{ij} = \sum_{j=m+1}^{m+n} PO_j \tag{3.8}$$

$$\sum_{i=1}^{m} \sum_{j=m+1}^{m+n} Z_{ij} = \sum_{j=m+1}^{m+n} VO_j \tag{3.9}$$

$$\sum_{i=1}^{m+n} Y_{ij} - \sum_{i=1}^{m+n} Y_{ji} = PO_j \qquad \forall j = m+1 \dots m+n \tag{3.10}$$

$$\sum_{i=1}^{m+n} Z_{ij} - \sum_{i=1}^{m+n} Z_{ji} = VO_j \qquad \forall j = m+1 \dots m+n \tag{3.11}$$

$$Y_{ij} \leq \sum_{d=1}^{m}\sum_{k=1}^{v} CPo^{k} \times X_{ij}^{kd} \qquad \forall i = 1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.12)$$

$$Z_{ij} \leq \sum_{d=1}^{m}\sum_{k=1}^{v} CV^{k} \times X_{ij}^{kd} \qquad \forall i = 1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.13)$$

$$X_{di}^{kb} = 0 \qquad \forall i = m+1 \dots m+n; \ \forall k = 1 \dots v; \ d \neq b = 1 \dots m \qquad (3.14)$$

$$X_{id}^{kb} = 0 \qquad \forall i = m+1 \dots m+n; \ \forall k = 1 \dots v; \ d \neq b = 1 \dots m \qquad (3.15)$$

$$Y_{ij} \leq \sum_{d=1}^{m}\sum_{k=1}^{v} (\ CPo^{k} - \ PO_{i}) \times X_{ij}^{kd} \qquad \forall i = 1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.16)$$

$$Z_{ij} \leq \sum_{d=1}^{m}\sum_{k=1}^{v} (\ CV^{k} - \ VO_{i}) \times X_{ij}^{kd} \qquad \forall i = 1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.17)$$

$$Y_{ij} \geq \sum_{d=1}^{m}\sum_{k=1}^{v} PO_{j} \times X_{ij}^{kd} \qquad \forall i = m+1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.18)$$

$$Z_{ij} \geq \sum_{d=1}^{m}\sum_{k=1}^{v} VO_{j} \times X_{ij}^{kd} \qquad \forall i = m+1 \dots m+n; \ \forall j = m+1 \dots m+n \qquad (3.19)$$

$$Y_{ij} = 0 \qquad \forall i = m+1 \dots m+n; \ \forall j = 1 \dots m \qquad (3.20)$$

$$Z_{ij} = 0 \qquad \forall i = m+1 \dots m+n; \ \forall j = 1 \dots m \qquad (3.21)$$

$$Y_{ij} = 0 \qquad \forall i = 1 \dots m; \forall j = 1 \dots m \qquad (3.22)$$

$$Z_{ij} = 0 \qquad \forall i = 1 \dots m; \forall j = 1 \dots m \qquad (3.23)$$

$$Y_{ii} = 0 \qquad \forall i = m+1 \dots m+n \qquad (3.24)$$

$$Z_{ii} = 0 \qquad \forall i = m+1 \dots m+n \qquad (3.25)$$

$$X_{ii}^{kd} = 0 \qquad \forall i = 1 \dots m; \ \forall j = 1 \dots m; \ \forall k = 1 \dots v; \ \forall d = 1 \dots m \qquad (3.26)$$

$$X_{ii}^{kd} = 0 \qquad \forall i = 1 \dots m+n;\ \forall k = 1 \dots v;\ \forall d = 1 \dots m \qquad (3.27)$$

$$\sum_{d=1}^{m} \sum_{i=1}^{m} \sum_{j=m+1}^{m+n} X_{ij}^{kd} \leq W^k \qquad \forall k = 1 \dots v \qquad (3.28)$$

La fonction objective (3.4) cherche à minimiser le coût total des tournées parcourues. Le contrainte (3.5) assure que chaque client n'est visité qu'une seule fois, La contrainte (3.6) garantis qu'au plus un type de véhicule provenant d'un entrepôt donné couvrira un arc i,j. La contrainte (3.7) assure la conservation des flux (un véhicule qui entre sur un site client doit en sortir). Les contraintes (3.8) et (3.9) assurent respectivement que le poids total et le volume total des marchandises quittant tous les entrepôts sont exactement égaux au poids total et au volume total des marchandises demandées par tous les clients. Les contraintes (3.10) et (3.11) garantissent respectivement que le poids restant et le volume restant après avoir visité le client j sont exactement le poids et le volume avant de visiter ce client moins respectivement le poids et le volume de sa commande. Les contraintes (3.12) et (3.13) garantissent respectivement que la capacité limite de poids et de celle de volume du véhicule de n'importe quel type ne sont pas violées. Les contraintes (3.14) et (3.15) garantissent respectivement qu'un véhicule quittant un dépôt (ou revenant à un dépôt) ne peut pas être relié à un dépôt différent. La contraintes (3.16) et (3.17) imposent respectivement que le poids et que le volume sur un arc ne peuvent pas être supérieure au poids et au volume sur ce véhicule après la livraison au client i sur l'arc (i, j) et en provenance de n'importe quel entrepôt. Les contraintes (3.18) et (3.19) garantissent respectivement qu'il reste suffisamment de poids et de volume pour servir le nœud j lorsqu'on se dirige vers le nœud j sur l'arc (i, j). Les contraintes (3.20) et (3.21) garantissent que tous les véhicules doivent voyager à vide (poids nul et volume nul respectivement) vers les entrepôts. Et les contraintes (3.22) et (3.23) imposent respectivement qu'aucun poids et aucun volume ne soit transportée entre les entrepôts. Les contraintes (3.24) et (3.25) imposent respectivement qu'aucun poids et qu'aucun volume ne soit transportée entre les entrepôts ou d'un client vers lui-même. Les contraintes (3.26) et (3.27) imposent respectivement qu'il n'y ait pas d'arc entre les entrepôts ou d'un client vers lui-même en utilisant n'importe quel type de véhicules. La contrainte (3.28) assure que le nombre total de véhicules de type k utilisés ne dépasse pas la taille de la flotte disponible pour ce type.

3.3 Méthode exacte avec IBM – CPLEX

IBM ILOG CPLEX Optimization Studio est un logiciel puissant, conçu spécialement pour modéliser et résoudre des problèmes d'optimisation complexes.

3.3.1 Les éléments du logiciel IBM ILOG CPLEX Optimization Studio

Le logiciel IBM ILOG CPLEX Optimization Studio comprend les outils suivants :

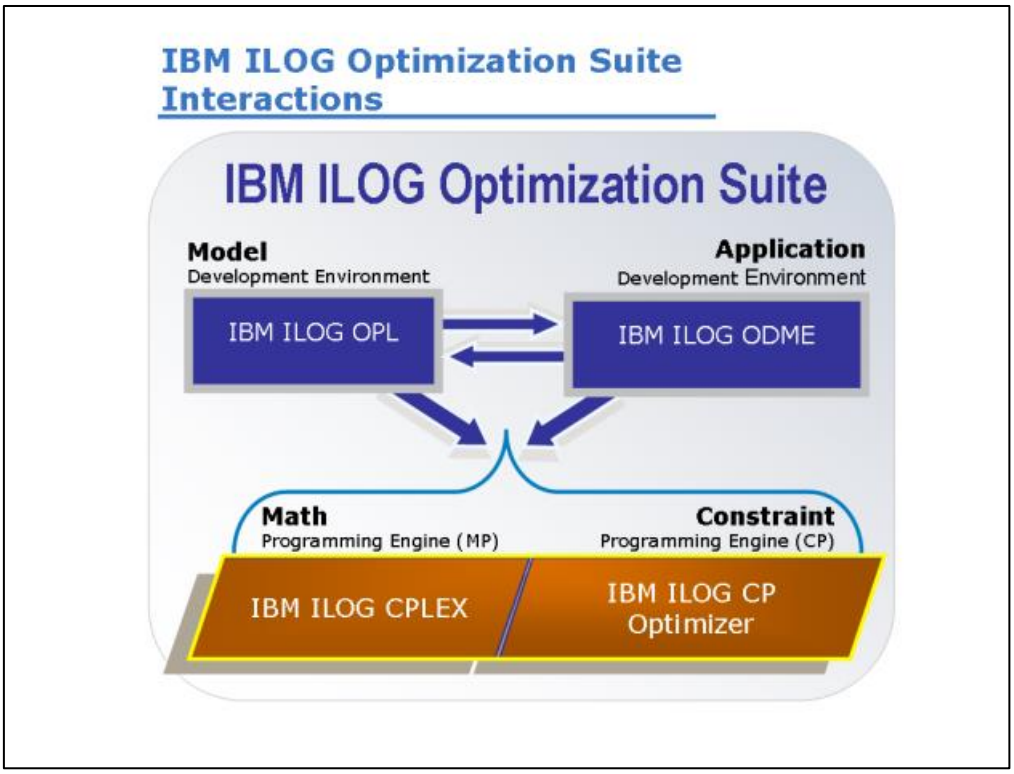


FIG 3.1 – Les principaux éléments du logiciel CPLEX [22]

3.3.1.1 IDE (Environnement de Développement Intégré)

Un environnement convivial pour le développement, le débogage et l'analyse des modèles, il comprend un débogueur, un éditeur de données et un visualiser de résultats.

3.3.1.2 Langage de modélisation OPL (Optimization Programming Language)

Le langage utilisé dans CPLEX Studio IDE est OPL (Optimization Programming Language). Il s'agit d'un langage de modélisation qui permet d'écrire facilement des programmes linéaires (ou quadratiques) grâce à une syntaxe proche de la formulation mathématique. Par ailleurs OPL offre à l'utilisateur la possibilité de séparer le modèle des données, de ce fait un même modèle peut être facilement testé avec différents jeux de données [23].

OPL fonctionne par projets : pour résoudre un modèle l'utilisateur doit créer un projet OPL dans CPLEX Studio IDE qui doit contenir au minimum un fichier "modèle" et un fichier de "configuration d'exécution". En effet chaque projet est

constitué de plusieurs types de fichiers [23] :

- Un fichier modèle (.mod) qui contient le modèle à résoudre.
- Un fichier de données (.dat) (facultatif) qui contient les données pour un modèle.
- Un fichier de paramètres (.ops) (facultatif) qui permet de paramétrer le solveur CPLEX.
- Un fichier de configuration d'exécution (.oplproject) qui indique à l'IDE ce qu'il doit faire quand l'utilisateur demande l'exécution du projet. C'est à dire quel est le modèle à résoudre et quels sont les paramètres et les données (s'il y en a) [23].

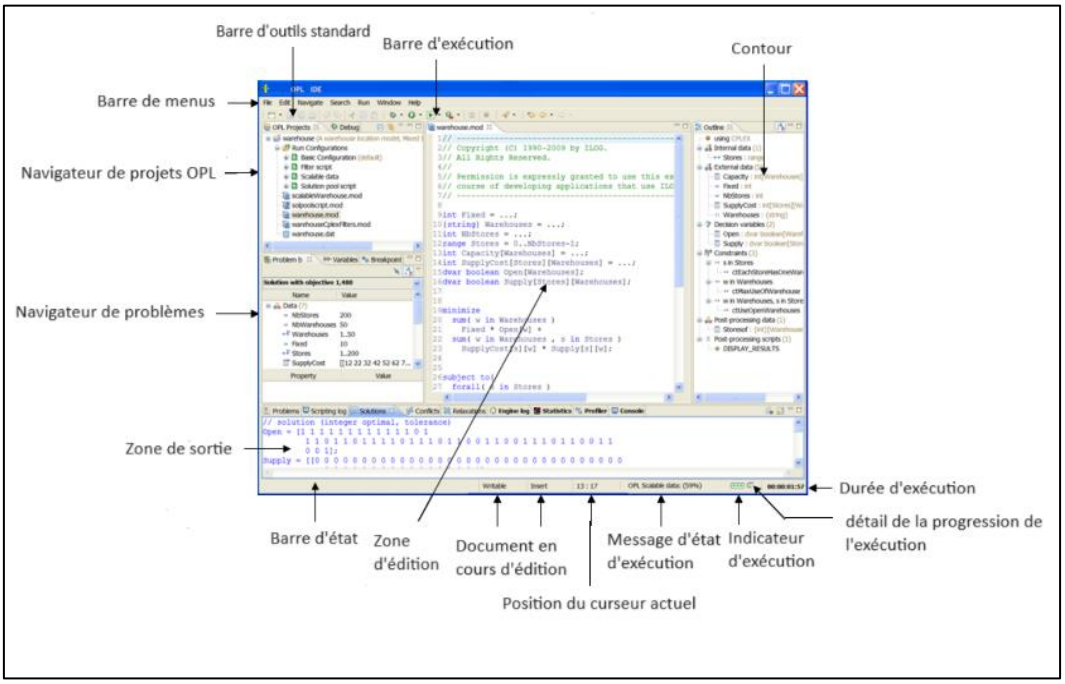


FIG 3. 2 – L'interface du logiciel CPLEX [24]

3.3.1.3 Le solveur IBM ILOG CPLEX

Un moteur d'optimisation pour la résolution des problèmes de programmation linéaire (PL), de programmation en nombres entiers mixtes (MIP) et de programmation quadratique (QP).

3.3.1.4 Le solveur CP Optimizer

Un solveur de programmation par contraintes utilisé pour résoudre des problèmes de planification, de gestion des ressources et de programmation, efficace pour les problèmes de nature combinatoire.

3.3.1.5 IBM ILOG Script for OPL

OPL sert uniquement à la modélisation, Cependant il existe un langage de script pour OPL appelé IBM ILOG Script for OPL (qui est une implémentation de JavaScript) et qui permet d'effectuer le prétraitement ou le post traitement des données dans le fichier modèle (initialisation via des calculs complexes, affichage de la solution…) à travers les trois types de blocs d'exécution suivants [23] :

- Blocs execute : Définis par « execute {…} » dans le fichier modèle, utilisés pour le pré ou post traitement, et éventuellement pour le paramétrage du solveur.
- Blocs main : définis par « main {…} » et uniques par fichier modèle, ils contrôlent le flot d'exécution.
- Blocs prepare : Définis par « prepare {…} » Dans le fichier de données, utilisés pour manipuler les données avant leur utilisation dans le fichier modèle.

3.3.2 L'algorithme de résolution

Pour les problèmes contenant des variables entières, CPLEX utilise une approche de Branch-and-Cut. L'algorithme d'optimisation maintient une hiérarchie de sous-problèmes de programmation linéaire liés, appelée arbre de recherche et généralement visualisée comme se ramifiant vers le bas, Il y a un sous-problème à chaque nœud de l'arbre, et chaque nœud est exploré en résolvant le sous-problème associé [25].

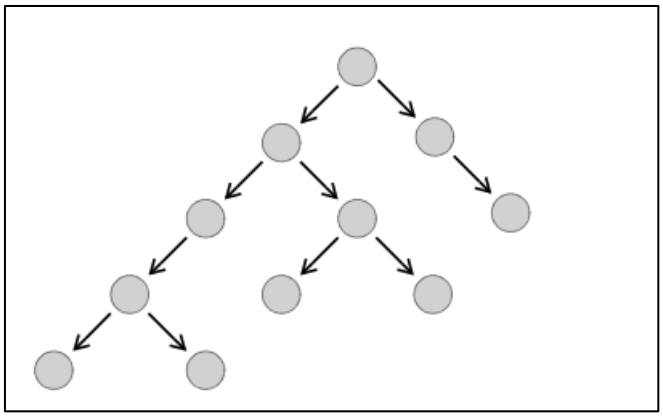


FIG 3.3 – L'arbre de recherche de Branch-and-Cut

L'algorithme commence avec un nœud racine dont le sous-problème est la relaxation du programme en nombres entiers, obtenue en supprimant les contraintes d'intégralité. Si cette relaxation donne une solution entière, elle est optimale. En général, l'optimum a des variables fractionnaires, nécessitant des plans de coupe et des algorithmes heuristiques pour resserrer la région faisable. Une variable fractionnaire est choisie pour le branchement, générant deux sous-problèmes avec des bornes plus restrictives, créant de nouveaux nœuds dans l'arbre de recherche. Le processus de branchement s'arrête si un sous-problème n'a pas de variables fractionnaires ou est infaisable. Lorsque tous les nœuds actifs sont explorés, CPLEX rapporte la solution finale, déclarée optimale si la valeur de coupure est définie à la valeur objective de la solution courante. La mémoire requise pour les sous-problèmes linéaires est similaire à celle des programmes linéaires, mais chaque nœud actif nécessite de la mémoire supplémentaire, rendant la mémoire totale nécessaire moins prévisible [25].

3.4 Conception de l'algorithme génétique

L'algorithme génétique utilise des techniques telles que la sélection, la mutation, et le croisement pour générer des solutions optimales ou quasi-optimales à des problèmes complexes. Avant de commencer le processus itératif, il est essentiel de définir clairement les paramètres qui régiront le fonctionnement de l'algorithme :

3.4.1 Structure des entrés

3.4.1.1 Paramètres d'entré généraux

- Taille de la Population : Nombre de solutions initiales générées aléatoirement.
- Valeur de Pénalité : Une valeur utilisée pour pénaliser les solutions qui violent les contraintes du problème.
- Taux de Croisement : Probabilité avec laquelle le croisement est appliqué pour créer de nouveaux individus.
- Taux de Mutation : Probabilité de mutation de chaque nouvel individu.
- Nombre de Générations : Nombre maximal d'itérations de l'algorithme.
- Elitisme : Option de conserver directement les meilleures solutions pour la génération suivante sans les modifier.

3.4.1.2 Paramètres d'entré spéciaux

❖ Critères de Sélection : Méthode utilisée pour sélectionner les individus pour la reproduction (par exemple, roulette, tournoi).

3.4.1.3 Paramètres d'entré spécifiques du problème

Pour le problème combiné de l'entreprise traitant du MDVRP et HFVRP, avec l'ajout de l'option OVRP (Open VRP), cette dernière est utilisée uniquement pour les véhicules affrétés, permettant des trajets ouverts sans retour obligatoire au dépôt. Par exemple :

❖ Type de Route :

- Fermée (Closed-VRP) : Chaque véhicule retourne à son point de départ après avoir complété sa route.
- Ouverte (Open-VRP) : Les véhicules ne sont pas requis de retourner au dépôt après la dernière livraison, optimisant ainsi les routes en fonction des dernières destinations.

❖ Nombre de type de véhicules : Nombre de différents types de véhicules disponibles, chacun avec des caractéristiques propres (capacité, coût, etc.).

❖ Capacité des véhicules : Capacité maximale de la charge pour chaque type de véhicule en termes de volume, de poids ou les deux en même temps.

❖ Cout fixe des véhicules : Coût fixe associé à l'utilisation de chaque type de véhicule, indépendamment des kilomètres parcourus ou du temps utilisé.

❖ Cout variable véhicules : Coût variable en fonction des kilomètres parcourus ou du carburant consommé.

❖ Taille de flotte : Nombre disponible pour chaque type de véhicule, limitant ainsi le nombre de véhicules qui peuvent être déployés simultanément dans le cas de FSVRP.

❖ Nombre d'entrepôts : Nombre total d'entrepôts ou de points de départ pour les véhicules.

❖ Nombre de clients : Nombre de clients a visités.

3.4.1.4 Données entrées

❖ Entrepôts : Ces données représentent les points de départ des véhicules pour les livraisons. Chaque entrepôt est identifié par un code et possède des coordonnées géographiques spécifiques (latitude et longitude).

- Code_Entrepot : Identifiant unique pour chaque entrepôt.
- Nom : Nom de l'entrepôt.
- Latitude : Position géographique nord-sud de l'entrepôt.
- Longitude : Position géographique est-ouest de l'entrepôt.

Exemple :

Code_Entrepot	Nom	Latitude	Longitude
E01	EntrepôtA	36,7525	3,042
E02	EntrepôtB	36,365	6,6147

TAB 3.1 – Exemple sur les données d'entrepôts

❖ Clients : Ces données représentent les points ou tous les clients sont situés. Chaque entrepôt est identifié par un code et possède des coordonnées géographiques spécifiques (latitude et longitude).

- Code_Client : Identifiant unique pour chaque entrepôt.
- Nom : Nom de Client.
- Latitude : Position géographique nord-sud du Client.
- Longitude : Position géographique est-ouest du Client.

Exemple :

Code_Client	Nom	Latitude	Longitude
C01	ClientA	35,6971	-0,6308
C02	ClientB	33,3404	6,0505
C03	ClientC	35,561	6,1739
C04	ClientD	34,8504	5,728
C05	ClientE	36,1911	5,4137
C06	ClientF	36,7372	3,0863
C07	ClientG	36,8796	7,7642

C08	ClientH	35,4142	8,1012
C09	ClientI	30,4267	9,4869
C10	ClientJ	34,6739	3,263
C11	ClientK	32,4902	3,6738
C12	ClientL	28,05	9,55
C13	ClientM	34,1558	4,1238
C14	ClientN	35,2122	0,6253
C15	ClientO	36,8796	3,0578

TAB 3.2 – Exemple sur les données de clients

❖ Commandes : Les informations sur les clients incluent la demande en volume et en poids, ainsi que l'emplacement géographique, ce qui est crucial pour planifier les itinéraires de livraison.

- Demande_Volume : Le volume total des articles commandés par le client, nécessaire pour calculer la capacité du véhicule requise.
- Demande_Poids : Le poids total des articles commandés.
- Code_Commande : Identifiant unique de la commande.
- Code_Client : Lien entre la commande et le client spécifique.
- Latitude : Position nord-sud du client.
- Longitude : Position est-ouest du client.

Exemple :

Code_Commande	Demande_Volume	Demande_Poids	Code_Client
CO01	3,2	675	C05
CO02	8,1	1598	C08
CO03	15	4025	C01
CO04	7,1	1388	C03
CO05	5,3	1620	C13
CO06	3,5	1154	C09
CO07	18,2	2512	C11
CO08	20,5	6285	C12

TAB 3.3 – Exemple sur les données de commandes

3.4.1.5 Données générés

❖ Matrice de distances/couts : cette matrice est essentielle pour le calcul des itinéraires les plus efficaces. Elle indique la distance ou le coût entre chaque paire d'entrepôts et de clients ayant passé commande, permettant de minimiser les distances parcourues.

Les valeurs de la matrice représentent les distances en kilomètres entre les points. Chaque ligne et chaque colonne correspond à un entrepôt ou un client commandeur, et l'intersection de chaque ligne et colonne donne la distance entre ces deux points.

Exemple :

Emplacement	E01	E02	C05	C08	C01	C03	C13	C09	C11	C12
E01	0.00	392.36	269.81	583.02	414.18	392.59	399.11	1212.11	598.94	1491.47
E02	392.36	0.00	126.95	198.25	789.14	110.01	407.15	1201.13	727.49	1343.83
C05	269.81	126.95	0.00	320.01	667.51	126.53	357.47	1045.04	606.40	1324.40
C08	583.02	198.25	320.01	0.00	980.71	216.70	481.30	1001.21	721.04	1162.28
C01	414.18	789.14	667.51	980.71	0.00	790.28	727.73	1423.52	699.94	1702.87
C03	392.59	110.01	126.53	216.70	790.28	0.00	298.67	955.99	536.93	1235.35
C13	399.11	407.15	357.47	481.30	727.73	298.67	0.00	936.87	331.60	1216.21
C09	1212.1	1201.13	1045.04	1001.21	1423.52	955.99	936.87	0.00	727.47	278.87
C11	598.94	727.49	606.40	721.04	699.94	536.93	331.60	727.47	0.00	1006.87
C12	1491.5	1343.83	1324.40	1162.28	1702.87	1235.35	1216.21	278.87	1006.87	0.00

TAB 3.4 – Exemple de la matrice de distances

Ces données et paramètres sont préparés pour être utilisées par l'algorithme de routage pour optimiser les itinéraires en fonction des besoins de livraison et des contraintes de capacité des véhicules.

3.4.2 Structure de l'Algorithme Génétique - Cas général

3.4.2.1 Phase Initiale

❖ Initialisation :

- Création des Chromosomes : La population initiale de chromosomes (solutions) est générée aléatoirement. Chaque chromosome représente une solution potentielle au problème.

❖ Évaluation :

- Calcul de Fitness : La fitness de chaque chromosome est calculée pour mesurer sa performance par rapport au problème à résoudre.

❖ Sélection :

- Sélection des Parents : Les chromosomes sont sélectionnés pour la reproduction selon leur fitness, en utilisant des méthodes comme la roulette ou le tournoi, en plus

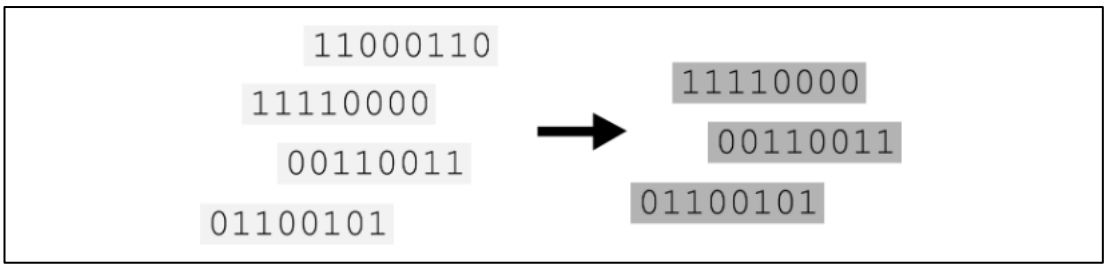


FIG 3.4 – Opération génétique - Sélection

Source : igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

3.4.2.2 Boucle Évolutive

❖ Croisement (Crossover) :

- Recombinaison Génétique : Des paires de chromosomes parents sont recombinées pour produire des descendants, en mélangeant leurs gènes (éléments de solution).

Exemple :

Le simple enjambement consiste à fusionner les particularités de deux individus à partir d'un pivot, afin d'obtenir un ou deux enfants :

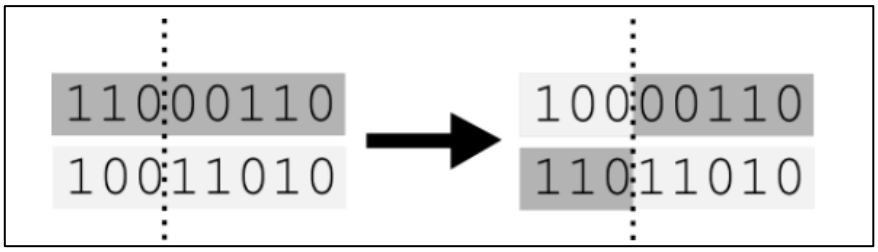


FIG 3.5 – Opération génétique - Croisement #1

Source : igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

Le double enjambement repose sur le même principe, sauf qu'il y a deux pivots :

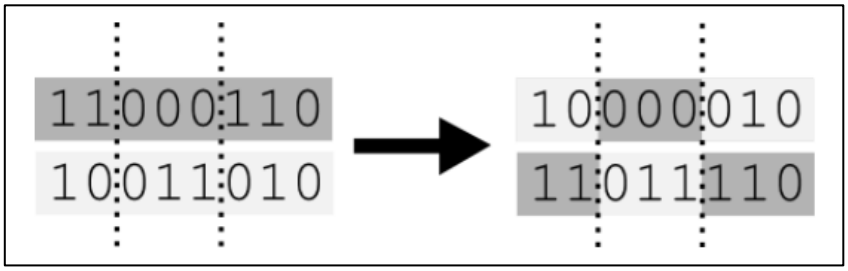


FIG 3.6 – Opération génétique - Croisement #2

Figure, croisement, source : igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

❖ Mutation :

- Modification Génétique : Les chromosomes descendants peuvent subir des mutations, introduisant des variations aléatoires pour explorer de nouvelles régions de l'espace de solution.

Exemple :

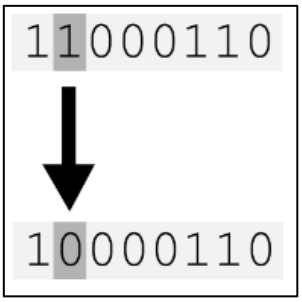


FIG 3.7 – Opération génétique - Mutation

Source : igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

❖ Evaluation :

- Calcul de Fitness pour les Descendants : La fitness des nouveaux chromosomes issus du croisement et de la mutation est évaluée.

❖ Sélection :

- Formation de la Nouvelle Population : La nouvelle génération de chromosomes est formée en sélectionnant les meilleurs parmi les anciens et les nouveaux chromosomes.

3.4.2.3 Finalisation :

❖ Solution Finale

- Critère d'Arrêt : Le processus se répète jusqu'à ce que le nombre maximal de générations soit atteint ou qu'une solution satisfaisante soit trouvée.

Les principes de bases étant expliqués, voici le fonctionnement des algorithmes génétiques :

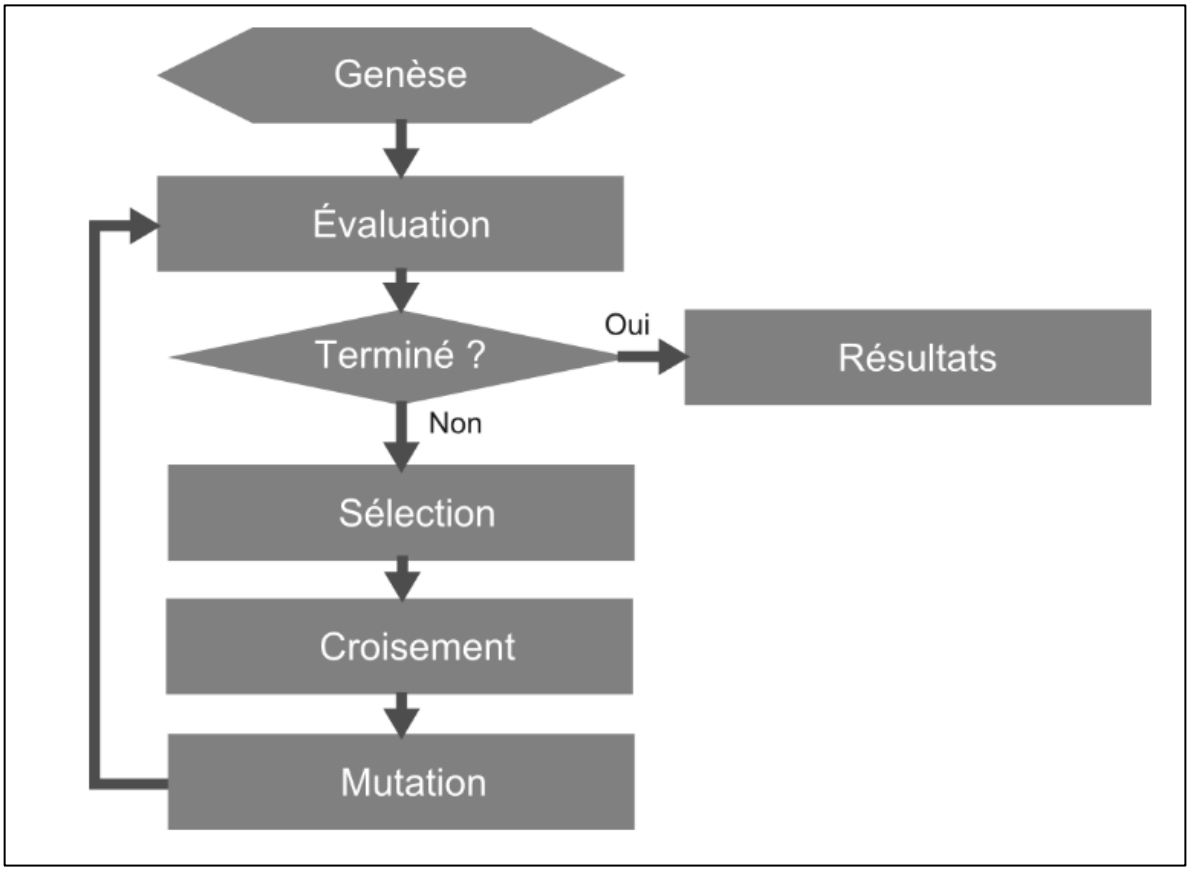


FIG 3. 8 – Les étapes de l'algorithme génétique

Source : igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

3.4.3 Étapes de l'Algorithme Génétique – Cas du problème

3.4.3.1 Phase Initiale

❖ Initialisation

- Structuration de Chromosome (individu) : La structure de la solution générée par l'algorithme génétique se compose de trois éléments principaux, l'indexation commence par les entrepôts suivis des clients, permettant ainsi une claire visualisation du départ des routes et de l'ordre de service comme illustré dans la figure suivante :

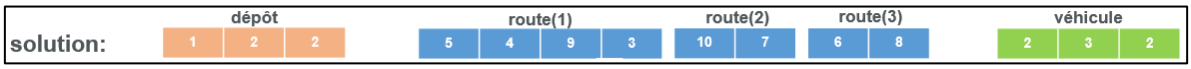


FIG 3.9 – Structure de la solution #1

Autrement, plus lisible :

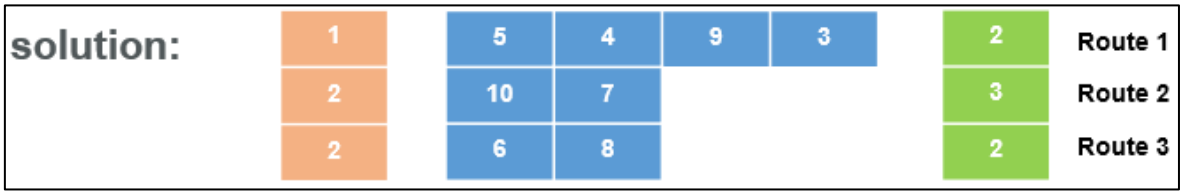


FIG 3.10 – Structure de la solution #2

Caractéristique du Chromosome (Solution de routage)

1- Entrepôt : Chaque solution commence par l'identification des entrepôts d'où les véhicules partent. Dans l'exemple donné, les véhicules partent de l'entrepôt 1 ou 2. Ce choix dépend de la proximité des entrepôts aux clients à desservir ou de leur capacité à gérer les volumes nécessaires.

2- Route : Chaque tourné à une ou plusieurs routes associées, chacune représentant un itinéraire spécifique qu'un véhicule suivra pour livrer aux clients respectant l'ordre. Dans l'exemple, il y a trois routes définies (Route (1), Route (2), et Route (3)), composées de différents clients identifiés par des numéros (comme 5, 4, 9, etc.). Ces numéros correspondent aux l'indices ou l'identifiants uniques des clients dans la base de données.

3- Véhicule : Chaque route est également associée à un type de véhicule, qui est choisi en fonction des besoins de capacité, du coût et de la disponibilité. Par exemple, les véhicules 2, 3, et 2 sont attribués aux différentes routes dans la solution présentée. Ces affectations dépendent des

caractéristiques de chaque véhicule, comme la capacité de charge, le coût d'opération et les contraintes de distance ou de temps.

- Création des Chromosomes : Les itinéraires initiaux sont générés aléatoirement, en veillant à ce que chaque client soit assigné une seule fois, évitant ainsi toute duplication dans les itinéraires. Cependant, à ce stade, d'autres contraintes comme la capacité des véhicules ne sont pas forcément respectées.

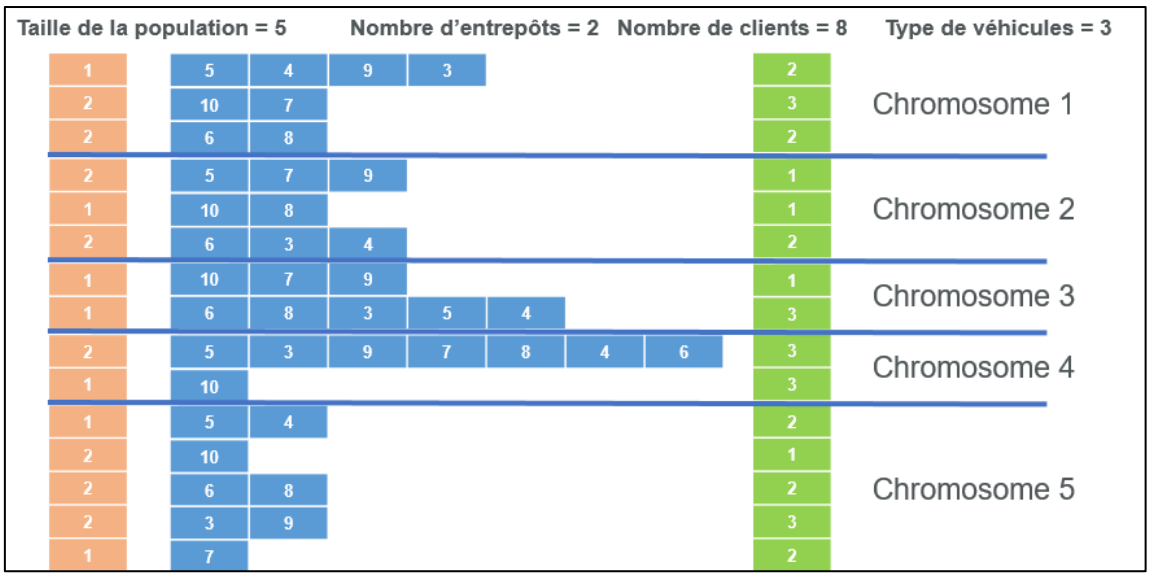


FIG 3.11 – Initialisation de la population

La figure montre une population initiale composée de cinq chromosomes, chacun représentant une solution candidate pour un problème de routage de véhicules. Ces chromosomes, qui intègrent des séquences d'entrepôts, de clients, et de types de véhicules, sont générés aléatoirement et serviront de points de départ pour l'évolution via les opérations génétiques.

❖ Évaluation

- Calcul de Fitness : Évalue la fitness de chaque chromosome en considérant non seulement le coût total des itinéraires, mais aussi en appliquant des pénalités pour non-respect des contraintes de capacité des véhicules, la taille maximale de la flotte et la présence de doublons dans même route.

❖ Sélection

- Sélection des Parents : Les chromosomes sont choisis pour la reproduction basée sur leur fitness, favorisant les solutions de meilleure qualité. Les méthodes telles que la roulette ou le tournoi sont employées, avec la pratique de l'élitisme pour conserver le meilleur chromosome de chaque génération.

3.4.3.2 Boucle Évolutive

❖ Croisement (Crossover)

- Recombinaison Génétique : Utilisation des méthodes spécifique pour le problème de routing comme BRBAX (Best Route Better Adjustment Recombination) et BCR (Best Cost Route Crossover) pour mélanger les gènes des parents de manière intelligente et créer des descendants. Ces méthodes assurent une combinaison efficace des traits des parents.

1- BRBAX (Best Route Better Adjustment Recombination) :

Est une méthode de croisement conçue spécifiquement pour les problèmes de routage. Elle fonctionne en échangeant des itinéraires entre deux solutions parentes pour créer un descendant. La méthode commence par sélectionner aléatoirement un itinéraire d'un parent. Cet itinéraire, qui consiste en une séquence de clients desservis par un véhicule, est ensuite intégré dans le second parent. Avant cette intégration, tous les clients présents dans l'itinéraire sélectionné sont retirés de leurs positions actuelles dans le second parent pour éviter la duplication. Enfin, l'itinéraire du second parent est réajusté pour accueillir le nouveau itinéraire tout en veillant à ce qu'il n'y ait pas d'itinéraires vides ou de violations des contraintes opérationnelles. Cette technique vise à exploiter les caractéristiques bénéfiques des deux parents, espérant ainsi améliorer la qualité de la solution résultante [26].

Exemple :

Apres la sélection des parents, pour deux différentes solutions qui possèdent plusieurs routes

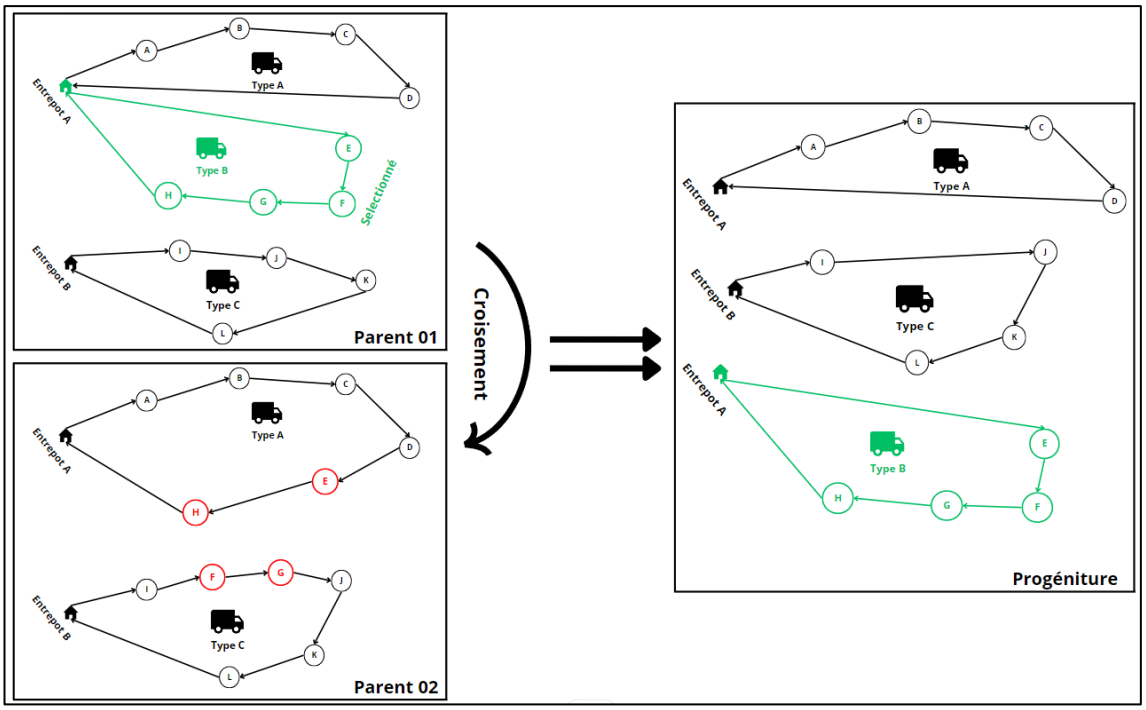


FIG 3.12 – Méthode BRBAX

Dans ce cas le programme sélectionne aléatoirement une route avec ces caractéristiques (entrepôt, type de véhicule, ordre) du premier parent, et l'insert complètement comme une nouvelle route pour le deuxième parent avec la suppression des nœuds identiques comme est illustré sur la figure.

2- BCR (Best Cost Route Crossover) :

Est une technique de croisement qui se concentre sur l'optimisation des coûts associés aux itinéraires dans les solutions de routage. Elle débute par la sélection aléatoire d'un ou plusieurs clients dans un parent, et évalue ensuite les coûts de leur insertion à différentes positions dans l'itinéraire du second parent. Pour chaque client sélectionné, l'algorithme examine plusieurs emplacements potentiels pour son insertion et calcule les implications en termes de distance, de coût et de conformité aux contraintes pour chaque configuration possible. L'insertion est effectuée à l'endroit qui minimise le plus le coût total, en tenant compte des coûts fixes et variables des véhicules ainsi que des pénalités pour les éventuelles violations des contraintes, comme le dépassement de la capacité des véhicules. Le BCR permet de peaufiner les itinéraires en mettant l'accent sur la réduction des coûts opérationnels tout en respectant les contraintes logistiques, offrant ainsi une solution potentiellement plus économique et efficace [27].

Exemple :

Pour deux différentes routes avec leurs caractéristiques

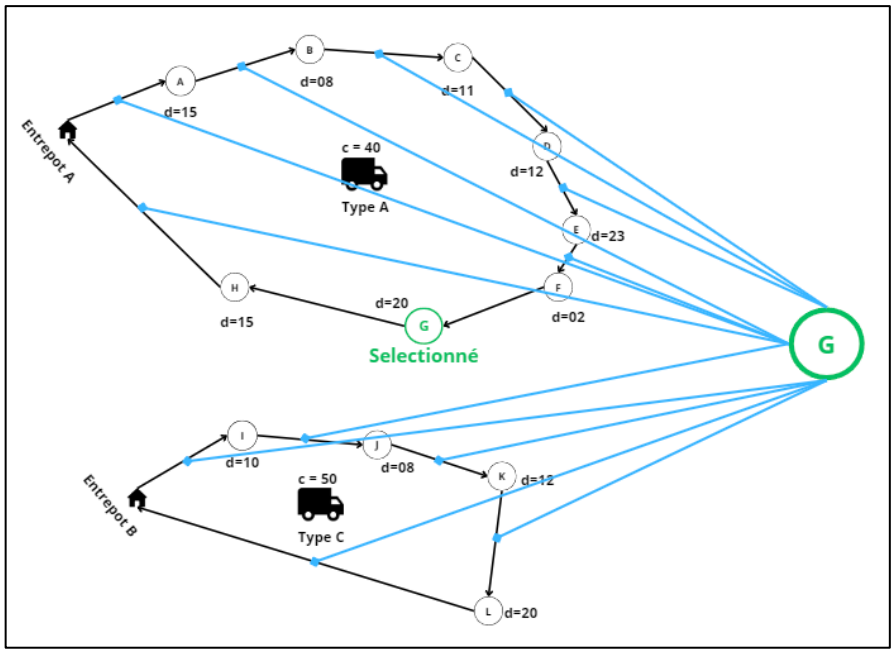


FIG 3.13 – Méthode BCR

Le programme sélectionne aléatoirement un nœud d'une route quelconque et essaie toutes les possibilités de le déplacer entre tous les arcs, comme illustré sur la figure. Pour chaque insertion, il calcule le coût total de l'algorithme, qui est composé du coût réel total et du coût des pénalités pour un problème de minimisation. Il doit sélectionner le coût minimum afin d'assurer la proposition la moins pénalisée pour se rapprocher de la solution réalisable.

- Corrections Post-Crossover :

Sont des ajustements nécessaires après la recombinaison génétique des chromosomes pour s'assurer que les solutions restent valides et optimisées en respectant les contraintes du problème. Ces corrections comprennent :

1- Correction des dépôts :

Après la recombinaison génétique, chaque itinéraire généré est évalué en fonction de la distance totale en considérant différents dépôts comme points de départ et d'arrivée (cas de route fermée). L'objectif est de sélectionner le dépôt qui minimise la distance totale de l'itinéraire.

2- Correction des véhicules :

Chaque itinéraire est également évalué en termes de coût total en envisageant l'utilisation de différents types de véhicules. Le véhicule qui

minimise le coût total, qui inclut le coût fixe, le coût variable par kilomètre et les éventuelles pénalités pour non-respect des capacités, est choisi. Cette étape assure que le véhicule assigné est le plus économique tout en étant capable de gérer la charge requise pour l'itinéraire.

3- Répartition des capacités (rupture de capacité)

Si la charge totale dans un itinéraire excède la capacité du véhicule assigné, l'itinéraire est divisé en plusieurs itinéraires. Cette division est réalisée de manière à ce que la charge de chaque itinéraire ne dépasse pas la capacité du véhicule. Cette étape peut entraîner la création de plusieurs itinéraires à partir d'un seul, chacun respectant les contraintes de capacité du véhicule.

Exemple :

Pour une solution qui possède dans ces tournées une route de 8 clients à visiter par véhicule de type « A » démarre de l'entrepôt « A ».

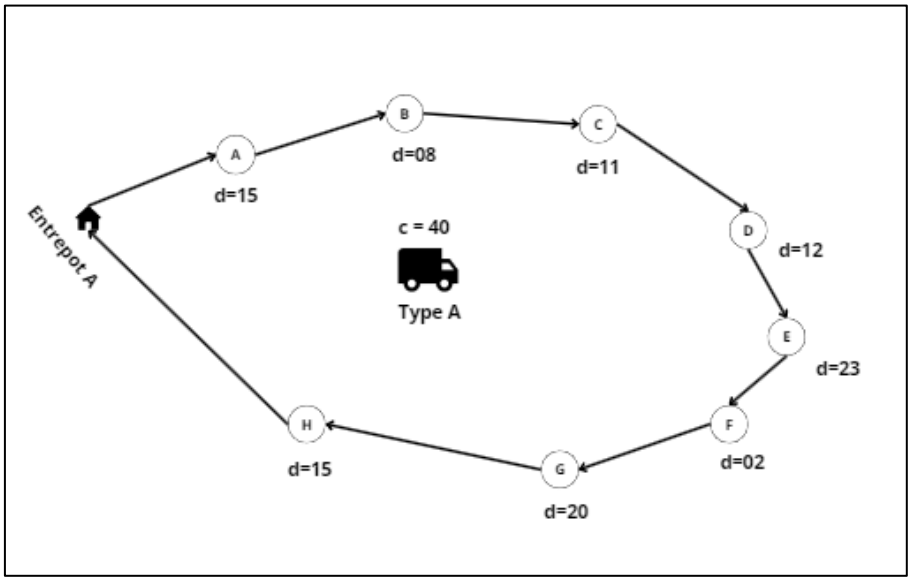


FIG 3.14 – Rupture de la capacité #1

d : demande

c : capacité

Etape 1 : calcule de la somme cumulée des demandes jusqu'à elle dépasse capacité

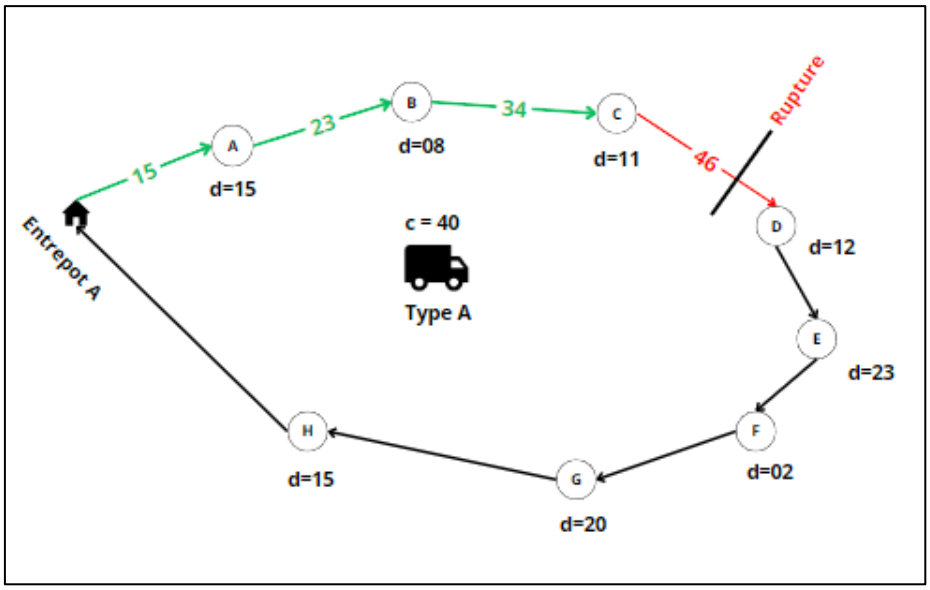


FIG 3.15 – Rupture de la capacité #2

La rupture sera sur l'arc (C,D) comme la somme des demandes selon l'ordre de la route dépasse la capacité de véhicule de type « A » associé

Etape 2 : rupture et décomposition de route

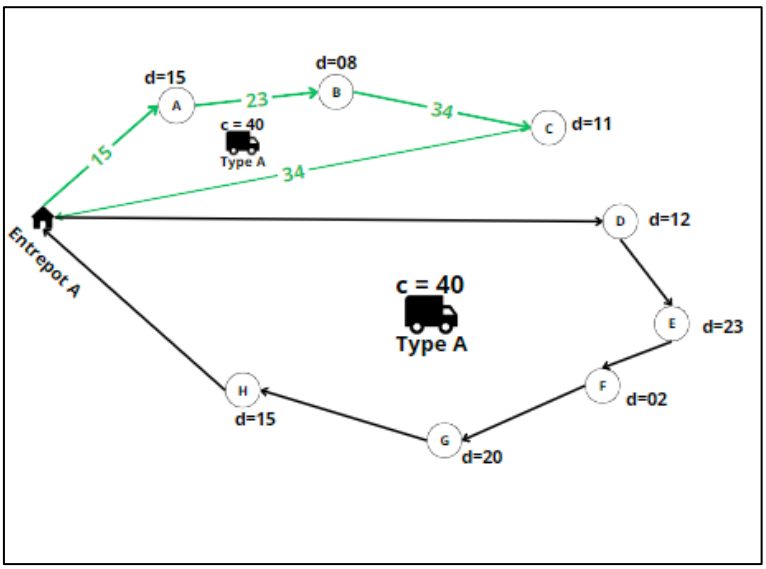


FIG 3.16 – Rupture de la capacité #3

Création nouvelle route porte les mêmes caractéristique de la route originaire (entrepôt, type de véhicule, ordre)

Etape 3 : calcule de la somme cumulée des demandes pour la nouvelle route créée

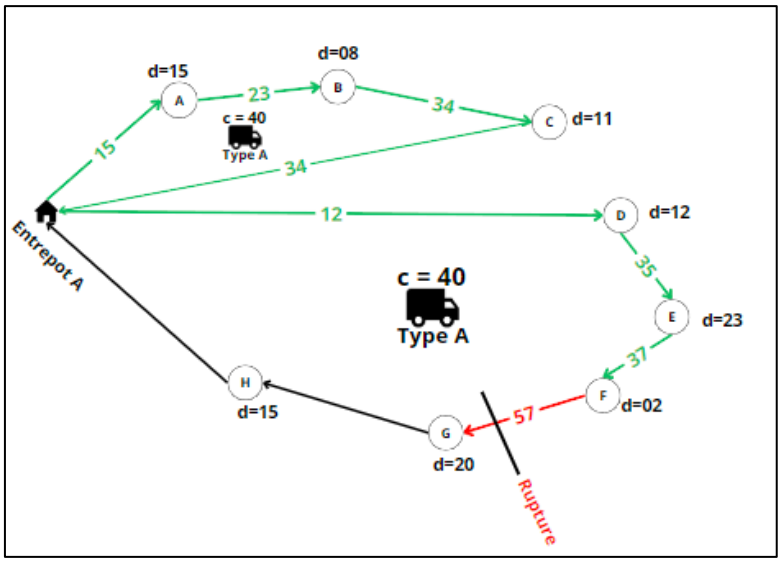


FIG 3.17 – Rupture de la capacité #4

La rupture sera sur l'arc (G,F) comme la somme des demandes selon l'ordre de la nouvelle route dépasse la capacité de véhicule de type « A » associé.

Par rapport les étapes restant possèdent mêmes processus, cette méthode arrêtera quand toutes les routes soient adaptées à la capacité de véhicule.

❖ Mutation

- Modification Génétique : Les descendants peuvent subir des mutations, types Insertion ou Echange/Inversion, pour introduire des variations et explorer de nouvelles solutions potentielles. Ces mutations respectent les contraintes de capacité par des ajustements post-mutation si nécessaire.

1- Mutation par Insertion

Est une technique qui altère la séquence des clients dans une tournée pour générer de la diversité dans les solutions. Dans cette mutation, un client est aléatoirement sélectionné et retiré de son emplacement actuel dans l'itinéraire. Ensuite, ce client est inséré à une autre position aléatoire au sein de la même tournée ou dans une autre tournée du même chromosome. Cette méthode peut aider à découvrir des configurations plus efficaces en termes de coût et de distance parcourue, qui n'auraient pas été générées par simple croisement. Après l'insertion, des ajustements peuvent être nécessaires pour s'assurer que la solution reste viable, par exemple, en vérifiant et en respectant les contraintes de capacité des véhicules.

2- Mutation par Echange/Inversion

La mutation par Swap (échange) ou Inversion consiste à sélectionner deux clients ou plusieurs dans une tournée et à échanger leurs positions (Swap) ou à inverser l'ordre de tous les clients entre deux points sélectionnés (Inversion). Le Swap modifie l'ordre en échangeant simplement les positions de deux clients, potentiellement réduisant la distance totale ou les coûts de la tournée si les clients échangés produisent un itinéraire plus direct ou plus conforme aux contraintes de fenêtres de temps. L'Inversion, quant à elle, prend une séquence de clients et la renverse, ce qui peut radicalement modifier la structure de la tournée et découvrir de nouvelles solutions efficaces que les opérations de croisement conventionnelles ne révèlent pas. Ces mutations sont appliquées aléatoirement et peuvent nécessiter des ajustements post-mutation pour garantir le respect des contraintes logistiques et de capacité.

❖ Evaluation :

- Calcul de Fitness pour les Descendants : La fitness des nouveaux chromosomes, générés par croisement et mutation, est à nouveau évaluée pour vérifier leur viabilité face aux contraintes du problème.

❖ Sélection :

- Formation de la Nouvelle Population : Une nouvelle génération de chromosomes est formée en sélectionnant les meilleurs parmi les anciens et les nouveaux chromosomes, en utilisant l'élitisme pour préserver les nouvelles meilleures solutions tout en veillant à maintenir une diversité génétique suffisante pour une exploration efficace de l'espace de solutions.

3.4.3.3 Finalisation

❖ Solution Finale :

- Critère d'Arrêt : L'algorithme est répété jusqu'à atteindre le nombre maximal de générations ou jusqu'à ce qu'une amélioration significative ne soit plus observée, permettant de converger vers une solution optimale ou satisfaisante.

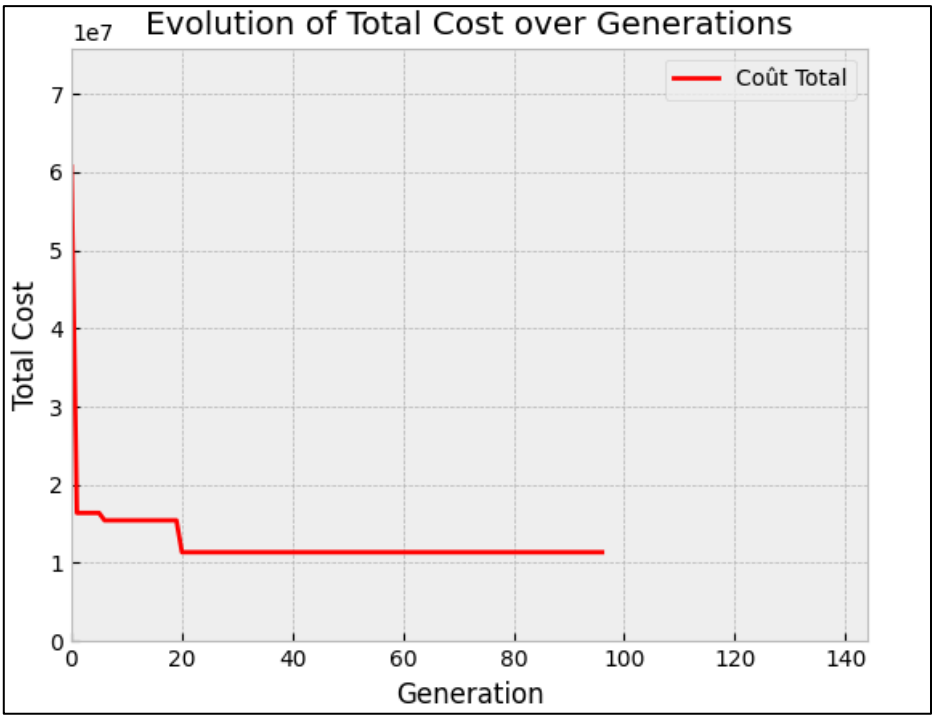


FIG 3.18 – Evolution de coût total en fonction de génération

Le graphique illustre l'évolution du coût total au fil des générations pour un algorithme génétique appliqué à un problème de routage de véhicules. Initialement, on observe une chute rapide du coût, signe d'une amélioration significative des solutions grâce à l'exploration efficace de l'espace des solutions par l'algorithme et les méthodes implémentées. Cette phase est caractéristique des premières générations où l'impact des croisements et mutations est le plus marqué. Suite à cette diminution, le coût se stabilise, indiquant que l'algorithme est entré dans une phase de régime stationnaire. Cela signifie que les modifications apportées à la population ne génèrent plus d'améliorations notables du coût, ce qui pourrait être dû à une convergence vers un minimum local/global ou à une saturation de la diversité génétique nécessaire pour poursuivre l'amélioration. Cette stagnation suggère que l'algorithme a potentiellement atteint les limites de son efficacité avec les paramètres et les stratégies actuelles, nécessitant ainsi une réévaluation des taux de mutation, de croisement, ou l'introduction de nouvelles stratégies pour dépasser ce plateau.

3.5 Conclusion

Dans ce chapitre nous avons présenté deux approches différentes pour tenter de résoudre le problème de MDHFVRP pour l'entreprise CPH Agriculture. D'une part, la méthode exacte du solveur IBM OPL ILOG CPLEX basée sur l'algorithme de Branch-and-Cut pourra permettre de trouver la solution optimale, mais son temps de calcul pourrait devenir prohibitif sur des instances de grande taille. D'autre part, l'algorithme génétique, en tant que méthode approchée, devrait

fournir des solutions de bonne qualité dans un temps raisonnable, même si l'optimalité n'est pas garantie. Dans le chapitre suivant, nous mettrons en œuvre et testerons ces deux méthodes afin d'évaluer leur performance respective sur les données réelles de CPH Agriculture. Les résultats de ces expérimentations permettront de déterminer la méthode la plus appropriée pour répondre aux besoins spécifiques de l'entreprise en matière d'optimisation des tournées de livraison.

Chapitre 4 :
Implémentation et test

4.1 Méthode exacte avec IBM-CPLEX

4.1.1 Configuration du solveur

Sur l'interface du solveur, dans le fichier « mod », nous avons commencé par saisir les instructions suivantes :

```
execute {
    // Configurer CPLEX pour forcer la recherche de la solution optimale exacte
    cplex.epgap = 0;  // Écart d'optimisation relatif
    cplex.epint = 0;  // Tolérance d'intégralité pour les variables entières
}
```

Ces instructions, regroupées dans un bloc « execute », permettent de configurer CPLEX pour forcer la recherche de la solution optimale exacte lors de la résolution du problème MDHFVRP. Elles définissent deux paramètres clés :

- cplex.epagap = 0 ; fixe l'écart d'optimisation relatif (relative gap) à 0, ce qui indique à CPLEX de ne s'arrêter que lorsque la solution optimale exacte est trouvée, sans tolérer aucun écart par rapport à l'optimal.
- cplex.epint = 0 ; définit la tolérance d'intégralité (integrality tolerance) pour les variables entières à 0, garantissant que toutes les variables entières auront des valeurs entières exactes dans la solution finale, sans aucune tolérance de fractionnalité.

En configurant ces paramètres à 0, nous forçons CPLEX à rechercher et trouver la solution optimale exacte pour le problème MDHFVRP, sans se contenter d'une solution approchée ou d'un écart par rapport à l'optimum. Cependant, cela peut augmenter considérablement les temps de calcul, en particulier pour les instances de grande taille ou très complexes.

4.1.2 Modèle mathématique et de la fonction d'affichage

❖ Le modèle mathématique sur le fichier « mod » :

```
// Déclarations des ensembles et indices
int n = ...;
int m = ...;
int K = ...;
// Définition des ensembles
range N = 1..m+n;
range entrepots = 1..m;
range clients = m+1..m+n;
range V = 1..K;
// Paramètres
float PO[N] = ...;
float VO[N] = ...;
float CPo[V] = ...;
float CV[V] = ...;
int W[V] =...;
float Cf[V] =...;
float Cva[V] =...;
float D[N][N] = ...;
// Variables de décision
dvar boolean X[N][N][V][entrepots];
dvar float+ Y[N][N];
dvar float+ Z[N][N];
// Fonction Objectif
minimize
    sum(d in 1..m,k in 1..K,i in 1..m,j in clients)Cf[k]*X[i][j][k][d]
    +sum(d in 1..m, k in 1..K, i in 1..m+n, j in 1..m+n )Cva[k]*D[i][j]*X[i][j][k][d];
// Contraintes
subject to {
     // Contrainte (1.5)
     forall( j in m+1..m+n )
       sum( d in 1..m, k in 1..K, i in 1..m+n)X[i][j][k][d]==1;
    // Contrainte (1.6)
    forall( i in m+1..m+n )
        sum( d in 1..m, k in 1..K, j in 1..m+n)X[i][j][k][d]==1;
    // Contrainte (1.7)
    forall(k in 1..K, j in 1..m+n, d in 1..m )
        sum(i in 1..m+n)X[i][j][k][d]==sum(i in 1..m+n)X[j][i][k][d] ;
    // Contrainte (1.8)
    sum( i in 1..m, j in m+1..m+n)Y[i][j]==sum( j in m+1..m+n)PO[j];
    // Contrainte (1.9)
    sum( i in 1..m, j in m+1..m+n)Z[i][j]==sum( j in m+1..m+n)VO[j];
     // Contrainte (1.10)
     forall(j in m+1..m+n)
        sum(i in 1..m+n )Y[i][j]- sum(i in 1..m+n )Y[j][i] == PO[j];
     // Contrainte (1.11)
     forall(j in m+1..m+n)
        sum(i in 1..m+n )Z[i][j]- sum(i in 1..m+n )Z[j][i] == VO[j];
    // Contrainte (1.12)
    forall( i in 1..n+m, j in m+1..m+n )
        Y[i][j]<=sum( d in 1..m, k in 1..K)CPo[k]*X[i][j][k][d];
    // Contrainte (1.13)
    forall( i in 1..n+m, j in m+1..m+n )
        Z[i][j]<=sum( d in 1..m, k in 1..K)CV[k]*X[i][j][k][d];
   // Contrainte (1.14)
   forall(i in m+1..m+n, k in 1..K, d in 1..m, b in 1..m : b!= d) X[d][i][k][b] == 0;
    // Contrainte (1.15)
    forall(i in m+1..m+n, k in 1..K, d in 1..m, b in 1..m : b!= d) X[i][d][k][b] == 0;
    // Contrainte (1.16)
    forall(i in 1..m+n, j in m+1..m+n )
        Y[i][j]<=sum(d in 1..m, k in 1..K)(CPo[k]-PO[i])*X[i][j][k][d];
     // Contrainte (1.17)
     forall(i in 1..m+n, j in m+1..m+n )
         Z[i][j]<=sum(d in 1..m, k in 1..K)(CV[k]-VO[i])*X[i][j][k][d];
   // Contrainte (1.18)
   forall(i in m+1..m+n, j in m+1..m+n)
         Y[i][j]>=sum(d in 1..m, k in 1..K)PO[j]*X[i][j][k][d];
   // Contrainte (1.19)
   forall(i in m+1..m+n, j in m+1..m+n)
         Z[i][j]>=sum(d in 1..m, k in 1..K)VO[j]*X[i][j][k][d];
    // Contrainte (1.20)
    forall(i in m+1..m+n, j in 1..m) Y[i][j]==0;
    // Contrainte (1.21)
    forall(i in m+1..m+n, j in 1..m) Z[i][j]==0;
    // Contrainte (1.22)
    forall(i in 1..m, j in 1..m) Y[i][j]==0;
    // Contrainte (1.23)
    forall(i in 1..m, j in 1..m) Z[i][j]==0;
    // Contrainte (1.24)
    forall(i in m+1..m+n) Y[i][i]==0;
    // Contrainte (1.25)
    forall(i in m+1..m+n) Z[i][i]==0;
   // Contrainte (1.26)
   forall( i in 1..m, j in 1..m, k in 1..K, d in 1..m)X[i][j][k][d]==0;
   // Contrainte (1.27)
   forall( i in 1..m+n, k in 1..K, d in 1..m)X[i][i][k][d]==0;
   // Contrainte (1.28)
   forall( k in 1..K)
     sum( d in 1..m, i in 1..m, j in m+1..m+n )  X[i][j][k][d]  <= W[k];
}
```

La fonction d'affichage :

```
// Affichage des résultats
execute {
    writeln();
    writeln("Affichage des parcours et des coûts regroupés par entrepot:");
    writeln();
    for(var d in entrepots) {  // Boucle sur les entrepots
        writeln();
        writeln("                          Entrepot ", d, ":");
        writeln();
        for(var k in V) {  // Boucle sur les types de véhicules
            var totalVariableCost = 0;
            var totalDistance = 0;
            var numTrips = 0;
            var totalPoids = 0;
            var totalVolume = 0;

            // Compter le nombre de départs du dépôt pour chaque véhicule
            for(var i in entrepots) {
                if (i == d) {  // Seulement si i est le dépôt en question
                    for(var j in N) {
                        if (X[i][j][k][d] == 1) {
                            numTrips += 1;
                        }
                    }
                }
            }
            // Calculer les distances et coûts des parcours
            for(var i in N) {
                for(var j in N) {
                    if (X[i][j][k][d] == 1) {
                        var distance = D[i][j];
                        var cost = Cva[k] * distance;
                        totalPoids  += PO[j];
                        totalVolume += VO[j];
                        totalDistance += distance;
                        totalVariableCost += cost;
                        writeln("  Type de Véhicule ", k, " - Parcours de ", i, " à ", j, " - Distance: ", distance,
                                " - Poids  livré au client  ", j, " : ", PO[j], " kg - Volume livré au client ", j, " :  ", VO[j], " m³ - Coût variable: ", cost);
                    }
                }
            }

            if (totalDistance > 0) { // Vérifier si des parcours ont été enregistrés
                writeln(); // Ajoute une ligne vide après les détails du parcours
                var totalFixedCost = Cf[k] * numTrips;  // Coût fixe multiplié par le nombre de départs
                var totalCost = totalVariableCost + totalFixedCost;
                writeln("  Distance totale pour le véhicule ", k, " depuis le dépôt ", d, ": ", totalDistance);
                writeln();
                writeln( " Poids totale transporté par le(s) véhicule(s) de type ", k, " depuis le dépôt ", d, " :  ", totalPoids);
                writeln( " Volume totale transporté par le(s) véhicule(s) ", k, " depuis le dépôt ", d, " :  ", totalVolume);
                writeln();
                writeln("  Coût variable total pour le véhicule ", k, " depuis le dépôt ", d, ": ", totalVariableCost);
                writeln("  Coût fixe total pour le véhicule ", k, " depuis le dépôt ", d, ": ", totalFixedCost);
                writeln("  Coût total pour le véhicule ", k, " depuis le dépôt ", d, ": ", totalCost);
                writeln(); // Ajout d'une ligne vide après le coût total pour un meilleur espacement
            }
        }
        writeln(); // Ajoute une ligne vide entre les dépôts pour une meilleure lisibilité
    }
}
```

4.1.3 Saisi des données sur le fichier dat

❖ Implémentation des données sur fichier « dat » :

```
n = 12; // Nombre des clients

m = 3; // Nombre des dépots

K = 4; // Nombre des types des véhicules


// Poids de la commande de chaque client, PO[i] = 0 pour chaque dépot i
PO = [0, 0, 0, 1598, 4025, 1388, 1620, 1154, 2512, 6285,675, 1598, 4025, 1388, 1620];

// Volume de la commande de chaque client i, VO[i] = 0 pour chaque dépot i
VO = [0, 0, 0, 8.1, 15, 7.1, 5.3, 3.5, 18.2, 20.5,3.2, 8.1, 15, 7.1, 5.3];

// Nombre de véhicules pour chaque type

W = [4,3,2,1];


// Capacité maximale de de poids pour chaque véhicule de type k

CPo = [4000,5000,10000,20000];

// Capacité volumique maximale  pour chaque véhicule de type k

CV = [14,16,27,60];

// Cout fixe pour chaque véhicule de type k

Cf =[815,990,1375,1985];

// Cout Variable ( Kilométrique ) pour chaque véhicule de type k

Cva = [70, 80];

// Distances entre les Noeuds (entrepôts et clients)
D = [
    [0.00, 396.29, 579.59, 108.94, 388.59, 315.36, 317.80, 374.91, 388.24, 192.35, 663.18, 135.08, 268.69, 347.02, 172.30],
    [396.29, 0.00, 687.39, 497.08, 496.40, 226.86, 93.55, 268.10, 78.70, 311.20, 733.28, 474.63, 210.17, 686.57, 371.01],
    [579.59, 687.39, 0.00, 666.73, 250.77, 446.52, 632.43, 465.48, 681.47, 446.78, 242.22, 580.17, 477.31, 794.10, 424.62],
    [108.94, 497.08, 666.73, 0.00, 488.73, 423.76, 368.86, 483.31, 494.06, 300.75, 686.89, 104.43, 377.09, 257.02, 250.40],
    [388.59, 496.40, 250.77, 488.73, 0.00, 255.46, 441.37, 274.42, 490.41, 255.71, 378.83, 402.78, 286.25, 616.70, 233.56],
    [315.36, 226.86, 446.52, 423.76, 255.46, 0.00, 232.91, 58.63, 245.94, 170.73, 554.25, 407.78, 68.38, 619.72, 224.17],
    [317.80, 93.55, 632.43, 368.86, 441.37, 232.91, 0.00, 280.48, 55.01, 254.35, 676.43, 417.78, 191.07, 629.72, 314.16],
    [374.91, 268.10, 465.48, 483.31, 274.42, 58.63, 280.48, 0.00, 274.64, 230.30, 593.60, 467.36, 127.95, 679.30, 283.75],
    [388.24, 78.70, 681.47, 494.06, 490.41, 245.94, 55.01, 274.64, 0.00, 303.29, 725.37, 466.72, 240.00, 678.66, 363.10],
    [192.35, 311.20, 446.78, 300.75, 255.71, 170.73, 254.35, 230.30, 303.29, 0.00, 459.18, 280.03, 126.52, 491.97, 95.30],
    [663.18, 733.28, 242.22, 686.89, 378.83, 554.25, 676.43, 593.60, 725.37, 459.18, 0.00, 601.00, 528.81, 714.27, 437.07],
    [135.08, 474.63, 580.17, 104.43, 402.78, 407.78, 417.78, 467.36, 466.72, 280.03, 601.00, 0.00, 354.43, 235.34, 227.74],
    [268.69, 210.17, 477.31, 377.09, 286.25, 68.38, 191.07, 127.95, 240.00, 126.52, 528.81, 354.43, 0.00, 572.71, 191.26],
    [347.02, 686.57, 794.10, 257.02, 616.70, 619.72, 629.72, 679.30, 678.66, 491.97, 714.27, 235.34, 572.71, 0.00, 438.04],
    [172.30, 371.01, 424.62, 250.40, 233.56, 224.17, 314.16, 283.75, 363.10, 95.30, 437.07, 227.74, 191.26, 438.04, 0.00]];
```

4.1.4 Discussions des résultats obtenus

```
// solution (optimal) with objective 18647

Affichage des parcours et des coûts regroupés par entrepot:


                        Entrepot 1:




                        Entrepot 2:


 Type de Véhicule 1 - Parcours de 2 à 7 - Distance: 93.55 - Poids  livré au client  7 : 1620 kg - Volume livré au client 7 :  5.3 m³ - Coût variable: 6548.5
 Type de Véhicule 1 - Parcours de 7 à 2 - Distance: 93.55 - Poids  livré au client  2 : 0 kg - Volume livré au client 2 :  0 m³ - Coût variable: 6548.5

 Distance totale pour le véhicule 1 depuis le dépôt 2: 187.1

Poids totale transporté par le(s) véhicule(s) de type 1 depuis le dépôt 2 :  1620
Volume totale transporté par le(s) véhicule(s) 1 depuis le dépôt 2 :  5.3

 Coût variable total pour le véhicule 1 depuis le dépôt 2: 13097
 Coût fixe total pour le véhicule 1 depuis le dépôt 2: 815
 Coût total pour le véhicule 1 depuis le dépôt 2: 13912



                        Entrepot 3:


 Type de Véhicule 3 - Parcours de 3 à 5 - Distance: 250.77 - Poids  livré au client  5 : 4025 kg - Volume livré au client 5 :  15 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 3 à 6 - Distance: 446.52 - Poids  livré au client  6 : 1388 kg - Volume livré au client 6 :  7.1 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 5 à 15 - Distance: 233.56 - Poids  livré au client  15 : 1620 kg - Volume livré au client 15 :  5.3 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 6 à 9 - Distance: 245.94 - Poids  livré au client  9 : 2512 kg - Volume livré au client 9 :  18.2 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 8 à 3 - Distance: 465.48 - Poids  livré au client  3 : 0 kg - Volume livré au client 3 :  0 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 9 à 3 - Distance: 681.47 - Poids  livré au client  3 : 0 kg - Volume livré au client 3 :  0 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 11 à 8 - Distance: 593.6 - Poids  livré au client  8 : 1154 kg - Volume livré au client 8 :  3.5 m³ - Coût variable: 0
 Type de Véhicule 3 - Parcours de 15 à 11 - Distance: 437.07 - Poids  livré au client  11 : 675 kg - Volume livré au client 11 :  3.2 m³ - Coût variable: 0

 Distance totale pour le véhicule 3 depuis le dépôt 3: 3354.41

Poids totale transporté par le(s) véhicule(s) de type 3 depuis le dépôt 3 :  11374
Volume totale transporté par le(s) véhicule(s) 3 depuis le dépôt 3 :  52.3

 Coût variable total pour le véhicule 3 depuis le dépôt 3: 0
 Coût fixe total pour le véhicule 3 depuis le dépôt 3: 2750
 Coût total pour le véhicule 3 depuis le dépôt 3: 2750

 Type de Véhicule 4 - Parcours de 3 à 10 - Distance: 446.78 - Poids  livré au client  10 : 6285 kg - Volume livré au client 10 :  20.5 m³ - Coût variable: 0
 Type de Véhicule 4 - Parcours de 4 à 13 - Distance: 377.09 - Poids  livré au client  13 : 4025 kg - Volume livré au client 13 :  15 m³ - Coût variable: 0
 Type de Véhicule 4 - Parcours de 10 à 4 - Distance: 300.75 - Poids  livré au client  4 : 1598 kg - Volume livré au client 4 :  8.1 m³ - Coût variable: 0
 Type de Véhicule 4 - Parcours de 12 à 3 - Distance: 580.17 - Poids  livré au client  3 : 0 kg - Volume livré au client 3 :  0 m³ - Coût variable: 0
 Type de Véhicule 4 - Parcours de 13 à 14 - Distance: 572.71 - Poids  livré au client  14 : 1388 kg - Volume livré au client 14 :  7.1 m³ - Coût variable: 0
 Type de Véhicule 4 - Parcours de 14 à 12 - Distance: 235.34 - Poids  livré au client  12 : 1598 kg - Volume livré au client 12 :  8.1 m³ - Coût variable: 0

 Distance totale pour le véhicule 4 depuis le dépôt 3: 2512.84

Poids totale transporté par le(s) véhicule(s) de type 4 depuis le dépôt 3 :  14894
Volume totale transporté par le(s) véhicule(s) 4 depuis le dépôt 3 :  58.8

 Coût variable total pour le véhicule 4 depuis le dépôt 3: 0
 Coût fixe total pour le véhicule 4 depuis le dépôt 3: 1985
 Coût total pour le véhicule 4 depuis le dépôt 3: 1985
```

❖ Resultat

Selon le rapport affiché, la solution optimale présente un coût total de 18647. Cette solution a permis de servir tous les clients en respectant les contraintes de capacité des véhicules et en minimisant les coûts. Notons que Le test a été réalisé par un seul cœur d'un CPU Intel Core i9-10900K et le temps d'exécution a été de 21,68 secondes.

Au départ de l'entrepôt 2, un véhicule de type 1 a effectué le parcours : 2 » 7 » 2, livrant le client 7 avant de retourner à l'entrepôt.

Pour l'entrepôt 3, deux véhicules ont été utilisés :

1. Un véhicule de type 3 a suivi le parcours : 3 » 5 » 15 » 6 » 9 » 8 » 11 » 3, desservant les clients 5, 6, 8, 9, 11 et 15.

2. Un véhicule de type 4 a pris l'itinéraire : 3 » 10 » 4 » 13 » 14 » 12 » 3, desservant les clients 10, 4, 13, 14 et 12.

Aucun véhicule n'a été utilisé depuis l'entrepôt 1.

Le rapport détaille également les distances parcourues, les poids et volumes livrés, ainsi que les coûts variables et fixes associés à chaque véhicule et chaque itinéraire. Les coûts totaux sont ventilés par type de véhicule et par entrepôt, avec une indication des poids et volumes totaux transportés.

4.2 Méthode approchée : Méta-Heuristique - l'Algorithme génétique

Pour aborder la complexité des problèmes de tournées de véhicules (VRP) chez CPH Agriculture, nous avons opté pour une méthode approchée basée sur les méta-heuristiques, en particulier l'algorithme génétique. Cette méthode permet de trouver des solutions efficaces même pour des instances de grande taille où les méthodes exactes deviennent impraticables en raison de la complexité computationnelle. Les tests réalisés dans cette étude visent à ajuster les paramètres de l'algorithme génétique afin d'optimiser la qualité des solutions obtenues et de minimiser les temps de calcul. Les paramètres sont définis aléatoirement dans le cadre des tests, et chaque scénario est testé plusieurs fois pour évaluer les performances de l'algorithme dans différentes configurations.

4.2.1 Choix de langage de programmation

Python est un choix populaire pour le développement d'algorithmes génétiques en raison de sa syntaxe claire, de sa flexibilité et du soutien étendu de sa com-

munauté. Voici un aperçu des bibliothèques et APIs utilisées, ainsi que des avantages et inconvénients associés à l'utilisation de Python dans ce contexte.

4.2.1.1 Bibliothèques et APIs Utilisées

- NumPy : Permet la manipulation efficace de tableaux et matrices, essentielle pour les calculs numériques intensifs.
- Pandas : Fournit des structures de données puissantes pour la manipulation et l'analyse de données complexes, telles que des séries temporelles ou des tableaux multidimensionnels.
- Matplotlib : Utilisé pour la création de visualisations graphiques, permettant de tracer l'évolution de la fitness ou d'autres métriques de performance au cours des générations.
- Folium : Une bibliothèque qui permet de créer des cartes interactives facilement avec la capacité d'intégrer des marqueurs, des lignes, et des polygones basés sur des coordonnées géographiques. Folium est particulièrement utile car il génère des cartes en utilisant la technologie Leaflet.js, qui est performante et largement utilisée dans les applications web.
- OSRM (Open Source Routing Machine) : Est une API de routage open source qui fournit des itinéraires rapides entre les points en utilisant le graphe de routes d'OpenStreetMap. OSRM est particulièrement utile pour les problèmes de VRP car il peut calculer rapidement des itinéraires optimaux et des distances routiers entre plusieurs points [29].

4.2.1.2 Avantages de Python

- Communauté Étendue et Support : Large base d'utilisateurs et communauté active, offrant une abondance de ressources d'apprentissage, de forums de discussion, et de modules tiers.
- Intégration avec D'autres Technologies : Facilité d'intégration avec d'autres applications et technologies, y compris des interfaces avec des langages de bas niveau comme C ou Fortran pour des calculs intensifs.
- Rapidité de Développement : Permet un développement rapide et un prototypage efficace grâce à sa syntaxe concise et claire.

4.2.1.3 Inconvénients de Python

- Performance : Python n'est pas aussi rapide que les langages compilés comme C++ ou Java en raison de sa nature interprétée. Cela peut être un inconvénient pour les simulations à grande échelle ou les calculs très intensifs.

❖ Utilisation de la Mémoire : La gestion automatique de la mémoire dans Python peut entraîner une consommation de mémoire supérieure, surtout avec de grandes données.

❖ Concurrence et Parallélisme : Python a des limitations en termes de concurrence due au Global Interpreter Lock (GIL), bien que des modules comme « multiprocessing » ou « asyncio » offrent des alternatives pour le parallélisme.

4.2.2 Paramétrisation de l'algorithme

4.2.2.1 Processus de test

Pour déterminer les paramètres optimaux de l'algorithme génétique, environ 1200 tests sont réalisés, où chaque configuration de paramètres est évaluée en fonction de la qualité des solutions obtenues et du temps nécessaire pour atteindre la convergence. Ces tests permettent de comprendre l'impact de chaque paramètre et d'ajuster leur valeur pour obtenir les meilleures performances possibles dans le contexte spécifique de VRP et ses variantes.

❖ Modélisation statistique

Les paramètres de l'algorithme sont définis aléatoirement dans le cadre des tests. Pour chaque configuration, les valeurs des paramètres sont choisies aléatoirement parmi un ensemble prédéfini. Les paramètres incluent :

- Taille de la population (population_size) : Choisie aléatoirement parmi [10, 20, 30, 40].
- Valeur de pénalité (penalty_value) : Fixée à une valeur élevée pour imposer des contraintes strictes.
- Taux de croisement (crossover_rate) : Choisi aléatoirement dans l'intervalle [0.1, 0.9].
- Taux de mutation (mutation_rate) : Choisi aléatoirement dans l'intervalle [0.1, 0.9].
- Élitisme (elite) : Choisi aléatoirement parmi [0, 1, 2, 3].
- Nombre de générations (generations) : Choisi aléatoirement parmi [100, 300, 500, 750, 1000, 1500, 2000, 3000].

Les paramètres spécifiques au nombre de dépôts, de clients et de types de véhicules sont définis manuellement pour chaque scénario. Un fichier Excel est utilisé pour entrer les données des clients et des entrepôts. Chaque scénario est

testé 100 fois avec une sélection aléatoire des paramètres pour évaluer les performances de l'algorithme.

❖ Objectif des tests

L'objectif des tests est de déterminer les seuils S_i de changement entre les classes de paramètres en fonction de la taille du problème en utilisant la régression linéaire. Ces tests évaluent chaque configuration de paramètres en termes de qualité des solutions obtenues et du temps nécessaire pour atteindre la convergence.

4.2.2.2 Détermination l'approche de paramètres

❖ Sélection des Meilleures Paramètres pour Chaque Configuration

D'après 1200 individus qui contiennent pour chaque configuration 150 lignes possédant des paramètres de l'algorithme dispersés de manière aléatoire, un meilleur paramètre est sélectionné pour chaque scénario :

- m : Nombre d'entrepôts
- n : Nombre de clients
- v : Type de véhicules
- Ps : Taille de population
- Pv : Valeur de pénalité (choisie élevée selon le problème et ses données)
- E : Elitisme
- Mr : Taux de mutation
- Cr : Taux de croisement
- G : Nombre de générations
- Et : Temps d'exécution

OS : WINDOWS, CPU : Intel Core i7-10700 – 2.90 GHz, RAM : 16GB

Scenario	m	n	v	Ps	Pv	E	Mr	Cr	G	Et
1	2	8	13	40	10000000	1	0.29	0.60	300	61.64
2	3	15	5	20	10000000	0	0.72	0.78	500	187.09

3	6	15	5	20	10000000	2	0.12	0.46	750	291.96
4	3	20	5	30	10000000	1	0.29	0.64	750	622.09
5	3	15	8	30	10000000	0	0.45	0.73	750	265.65
6	3	35	10	30	10000000	3	0.74	0.16	500	648.58
7	5	35	10	10	10000000	1	0.68	0.23	1000	563.65
8	3	35	13	20	10000000	1	0.43	0.33	500	598.13
9	3	50	10	40	10000000	2	0.76	0.33	500	1621.15
10	6	60	20	40	10000000	3	0.71	0.41	1000	1923.11

TAB 4.1 – Meilleur paramètres sélectionnés pour chaque scénario

Le choix de ces paramètres se concentre sur l'optimalité et le temps d'exécution observés.

❖ Classification des paramètres

Les paramètres de l'algorithme génétique sont classés en différentes catégories (A, B, C), chacune correspondant à une taille spécifique de problème pour des routes fermées. Cette classification simplifie la gestion des risques et permet d'améliorer les contraintes dues à la nature aléatoire de l'algorithme génétique. La capacité des véhicules et la taille de la flotte ne sont pas incluses dans ces classes pour éviter une complexification excessive, ce qui nécessiterait des tests plus détaillés.

Classe	Ps	Pv	E	Mr	Cr	G
A	20	10000000	0 - 1	0.12 … 0.72	0.46 … 0.78	500 - 750
B	30	10000000	1 - 2	0.29 … 0.74	0.16 … 0.73	750
C	40	10000000	1 … 3	0.71 … 0.76	0.33 … 041	1000

TAB 4.2 – Classification des paramètres

Ces classes sont déterminées à partir du résultat final des tests, les fixations et les variations des paramètres pour chaque classe sont déterminées approximativement.

- Classe A : (Scenarios 1, 2, 3) représente la petite taille
- Classe B : (Scenarios 4, 5, 6, 7, 8) représente la moyenne taille
- Classe C : (Scenarios 9, 10) représente la grande taille

❖ Régression linéaire

Appliquant la méthode des Moindres Carrés Ordinaires sur la variable dépendante « Et » et les variables indépendantes « m – n – v ».

OLS Regression Results

Dep. Variable:	Execution Time	R-squared:	0.880
Model:	OLS	Adj. R-squared:	0.820
Method:	Least Squares	F-statistic:	14.68
Date:	Mon, 10 Jun 2024	Prob (F-statistic):	0.00360
Time:	20:39:26	Log-Likelihood:	-67.274
No. Observations:	10	AIC:	142.5
Df Residuals:	6	BIC:	143.8
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-295.3927	280.756	-1.052	0.333	-982.379	391.594
m	4.4448	66.810	0.067	0.949	-159.033	167.923
n	34.1412	7.268	4.697	0.003	16.357	51.926
v	-2.6276	24.958	-0.105	0.920	-63.697	58.442

Omnibus:	1.949	Durbin-Watson:	1.081
Prob(Omnibus):	0.377	Jarque-Bera (JB):	1.056
Skew:	-0.473	Prob(JB):	0.590
Kurtosis:	1.719	Cond. No.	120.

FIG 4.1 – Résultat statistique de la régression linéaire sur les données de paramètres de l'algorithme génétique

- Résumé des Résultats

R-squared (R^2) : Indique que 88% de la variance dans le temps d'exécution est expliquée par les variables indépendantes.

Adj. R-squared : Ajusté pour le nombre de variables dans le modèle, confirmant la bonne capacité explicative du modèle.

F-statistic: Montre que le modèle global est statistiquement significatif avec une p-value de $0.00360 < 0.05$.

- Coefficients

coef(const) δ : -295.3927 - Valeur du temps d'exécution lorsque toutes les variables explicatives sont à zéro.

coef(m) α : 4.4448 - Chaque dépôt supplémentaire augmente le temps d'exécution en moyenne de 4.4448 unités, mais ce coefficient n'est pas statistiquement significatif (p-value = 0.949).

coef(n) β : 34.1412 - Chaque client supplémentaire augmente le temps d'exécution en moyenne de 34.1412 unités. Ce coefficient est statistiquement significatif (p-value = 0.003).

coef (v) γ : -2.6276 - Chaque type de véhicule supplémentaire diminue le temps d'exécution en moyenne de 2.6276 unités, mais ce coefficient n'est pas statistiquement significatif (p-value = 0.920).

Les coefficients pour le nombre d'entrepôts « m » et le nombre de types de véhicules « v » ne sont pas significatifs, mais leur inclusion est justifiée pour une vue complète du système. La petite taille de notre échantillon limite la détection de relations significatives, et une augmentation de la taille pourrait révéler des significativités statistiques, reflétant mieux les relations programmées dans l'algorithme. Le modèle global reste significatif, indiquant que les variables indépendantes expliquent collectivement une part significative de la variance du temps d'exécution.

En conclusion, il est acceptable d'inclure les coefficients non significatifs pour des raisons théoriques et pratiques, tout en notant leur non-significativité et en continuant à affiner le modèle avec plus de données.

❖ Calcul des Seuils

Pour calculer manuellement X pour une configuration (m – n – v) donnée, on doit utiliser la formule suivante :

$$X = \alpha m + \beta n + \gamma v + \delta$$

Ensuite, comparant la valeur de X aux seuils S_i pour déterminer la classe de paramètres :

- Si $X < S_1$, la classe de paramètres A est choisie.
- Si $S_1 \leq X < S_2$, la classe de paramètres B est choisie.
- Si $S_2 \leq X$, la classe de paramètres C est choisie.

Pour calculer les seuils S_1 et S_2 avec notre programme, nous devons d'abord calculer les valeurs de X_i pour chaque scénario (i) en utilisant les coefficients de

la régression linéaire. Ensuite, nous déterminons les seuils S_1 et S_2 en fonction les quantiles q_1 et q_2 des valeurs de X_i.

Prenant les 10 scenarios mentionnés sur table, $q_1 = \frac{3}{10}, q_2 = \frac{8}{10}$ ces répartitions sont déterminées d'après les mesures de risque de l'algorithme qui se base sur des processus et sélections aléatoires.

Ensuite,

Soit :

$\alpha = 4.44, \beta = 34.14, \gamma = -2.63, \delta = -295.39$

$X = 4.44m + 34.14n - 2.63v - 295.39$

On obtient le résultat suivant d'après le programme :

- $S_1 = 226.26$
- $S_2 = 996.14$

Ces valeurs de seuil ont été obtenues en utilisant une méthode de régression linéaire suivie de l'interpolation des quantiles de la distribution des valeurs prédictives X

Donc

- Si $X < 226.26$, la classe de paramètres A est choisie.
- Si $226.26 \leq X < 996.14$, la classe de paramètres B est choisie.
- Si $996.14 \leq X$, la classe de paramètres C est choisie.

Exemple :

Soit :

$m = 4, n = 17, v = 6$

Calculant X

$X = 4.44 \times 4 + 34.14 \times 17 - 2.63 \times 6 - 295.39$

$X = 286.97$

Déterminant la meilleure classe de paramètres

$226.26 \leq X < 996.14$, donc la classe B est préférable.

4.2.3 Tests et validation

OS : WINDOWS, CPU : Intel Core i9-10900K, RAM : 32GB

CPU-T : Temps d'exécution utilisant seulement 01 cœur de processeur possède 3.7 GHz comme vitesse

Pour chaque scénario, la configuration est testée 5 fois. Si des solutions de qualité différente (Optimal ou Proche-Optimal) sont trouvées parmi les tests, le CPU-T est déterminé comme la moyenne des temps d'exécution correspondant à la qualité de solution majoritaire.

4.2.3.1 Tests

	m	n	v	Paramètre	CPU-T (s)	Qualité de Solution
1	2	5	3	Classe A	51.25	Optimal
2	2	5	5	Classe A	57.22	Optimal
3	2	8	3	Classe A	63.14	Optimal
4	5	8	3	Classe A	75.66	Optimal
5	2	13	3	Classe A	158.02	Optimal
6	2	8	6	Classe A	96.39	Optimal
7	3	25	10	Classe B	314.18	Proche-Optimal
8	5	25	10	Classe B	343.72	Proche-Optimal
9	3	35	10	Classe B	498.18	Proche-Optimal
10	3	25	13	Classe B	475.46	Proche-Optimal
11	3	50	15	Classe C	1546.07	Proche-Optimal
12	5	50	15	Classe C	1565.85	Proche-Optimal
13	3	60	15	Classe C	1899.62	Proche-Optimal
14	3	50	20	Classe C	1701.40	Proche-Optimal
15	8	100	25	Classe C	3468.54	Non-Réalisable

TAB 4. 3 – Résultat de tests effectué sur différentes configurations

4.2.3.2 Validation

La décision sur la qualité de la solution (optimale ou approchée) est prise à partir de 5 tests réalisés, selon les scénarios suivants :

❖ Optimal :

- La solution n'est considérée comme optimale que si au moins 3 essais parmi les 5 effectués donnent le même résultat objectif et que ce dernier soit identique à celui obtenu grâce à la méthode exacte.

Ainsi, la méthode paramétrée des classes "A, B, C" est valable pour les contraintes et pour l'optimalité.

❖ Proche-Optimal :

- De scenario 7 au scenario 14 : La solution n'est considérée comme optimale que si au moins 4 essais parmi les 5 effectués donnent le même résultat objectif, faute de quoi elle sera considérée comme approchée

Donc, la méthode paramétrée des classes "B, C" est valable uniquement pour les contraintes.

❖ Non-Réalisable :

- 15ème scenario : La solution est considérée comme non réalisable si au moins un essai donne une solution relaxée par rapport aux contraintes.

Donc la méthode paramétrée de classe « A, B, C » est non-valable par rapport les contraintes.

Pour obtenir une solution réalisable dans ce cas, il est nécessaire d'effectuer d'autres tests avec une taille de problème plus importante et d'ajuster correctement les paramètres de l'algorithme. Plus précisément, la taille actuelle de la population initiale et le nombre de générations utilisés sont insuffisants pour garantir la réalisabilité de la solution. Augmenter la taille de la population initiale permet d'explorer un espace plus large de solutions potentielles, améliorant ainsi les chances d'identifier des régions prometteuses de l'espace de recherche. De même, accroître le nombre de générations offre à l'algorithme davantage d'opportunités pour affiner ces solutions par sélection, croisement et mutation, conduisant à une amélioration progressive des solutions et augmentant la probabilité d'atteindre une solution non seulement optimale mais également réalisable, en particulier dans des contextes de problèmes complexes ou de grande taille.

4.3 Comparaison des performances des méthodes

4.3.1 Méthode exacte vs Méthode Approchée

Dans le cadre de la résolution de problèmes d'optimisation complexes comme les problèmes de tournées de véhicules (VRP), on a recours à deux grandes catégories de méthodes : les méthodes exactes et les méthodes approchées. Chacune présente des avantages et des inconvénients qu'il convient d'examiner avant de faire un choix en fonction du contexte et des objectifs.

Les méthodes exactes, comme l'algorithme "Branch-and-Cut" implémenté dans CPLEX, garantissent de trouver la solution optimale. Elles fournissent également des bornes supérieures et inférieures permettant d'évaluer la qualité de la solution. Cependant, le temps de calcul peut être extrêmement long pour les instances de grande taille ou très contraintes, avec une utilisation intensive des ressources processeur. Ces méthodes sont donc adaptées aux problèmes de taille modérée où l'optimalité de la solution est cruciale, par exemple dans des contextes réglementés ou très compétitifs.

À l'inverse, les méthodes approchées comme les algorithmes génétiques sont plus rapides pour obtenir des solutions acceptables, surtout sur les grandes instances. Elles sont capables de fournir de bonnes solutions même en présence de contraintes complexes. Néanmoins, elles ne garantissent pas l'optimalité et la qualité de la solution dépend du paramétrage de l'algorithme et des caractéristiques de l'instance. Ces méthodes sont préférées lorsque le temps de calcul est contraint ou qu'une solution "suffisamment bonne" est acceptable.

Pour comparer ces deux types de méthodes, on mesure généralement le temps de calcul (CPU time) et la qualité de la solution obtenue par rapport à l'optimum ou une borne inférieure connue. Des expériences systématiques sur des instances de tailles variées permettent de documenter l'efficacité mais aussi l'efficience en termes de ressources utilisées.

4.3.2 Analyse de performance

L'objectif de cette analyse est de comparer les performances de deux approches de résolution du problème de tournées de véhicules : une méthode exacte utilisant CPLEX, et une méthode approchée via un algorithme génétique. Les performances sont évaluées en termes de qualité des solutions obtenues et de temps de calcul.

Données :

m : Nombre d'entrepôts définis

n : Nombre de clients à visiter

v : Type de véhicules

Résultats :

OS : WINDOWS, CPU : Intel Core i9-10900K – 3.7 GHz, RAM : 32GB

CPU-T : Temps d'exécution utilisant seulement 01 cœur de processeur

- Taille du problème : Petite

Configuration Testée	Méthode	m	n	v	CPU-T (s)	Qualité de Solution
Base	Exacte	2	8	3	80.10	Optimal
Base	Approchée	2	8	3	63.14	Optimal
Augmente m	Exacte	5	8	3	28.05	Optimal
Augmente m	Approchée	5	8	3	75.66	Optimal
Augmente n	Exacte	2	13	3	61.53	Optimal
Augmente n	Approchée	2	13	3	158.02	Optimal
Augmente v	Exacte	2	8	6	75.72	Optimal
Augmente v	Approchée	2	8	6	96.39	Optimal
Augmente n, v	Exacte	3	12	12	563.94	Optimal
Augmente n, v	Approchée	3	12	12	204.38	Optimal

TAB 4.4 – Comparaison entre méthode exacte et approchée pour des données de petite taille

- Taille du problème : Moyenne

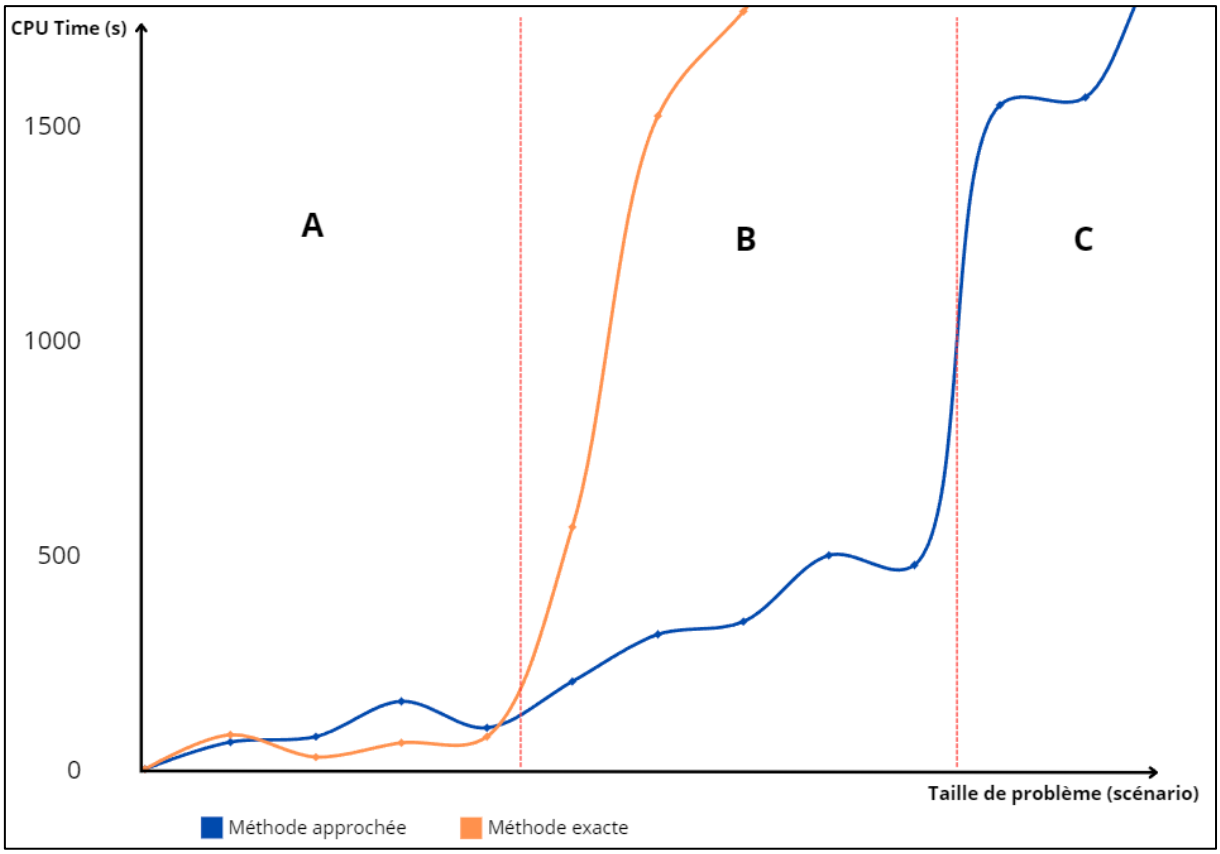
Configuration Testée	Méthode	m	n	v	CPU-T (s)	Qualité de Solution
Base	Exacte	3	25	10	1521.59	Optimal
Base	Approchée	3	25	10	314.18	Proche-Optimal
Augmente m	Exacte	5	25	10	+1600.00	/
Augmente m	Approchée	5	25	10	343.72	Proche-Optimal
Augmente n	Exacte	3	35	10	+1600.00	/
Augmente n	Approchée	3	35	10	498.18	Proche-Optimal
Augmente v	Exacte	3	25	13	+1600.00	/
Augmente v	Approchée	3	25	13	475.46	Proche-Optimal

TAB 4.5 – Comparaison entre méthode exacte et approchée pour des données de moyenne taille

- Taille du problème : Grande

Configuration Testée	Méthode	m	n	v	CPU-T (s)	Qualité de Solution
Base	Exacte	3	50	15	+1600.00	/
Base	Approchée	3	50	15	1546.07	Proche-Optimal
Augmente m	Exacte	5	50	15	+1600.00	/
Augmente m	Approchée	5	50	15	1565.85	Proche-Optimal
Augmente n	Exacte	3	60	15	+1600.00	/
Augmente n	Approchée	3	60	15	1899.62	Proche-Optimal
Augmente v	Exacte	3	50	20	+1600.00	/
Augmente v	Approchée	3	50	20	1701.40	Proche-Optimal

TAB 4.6 – Comparaison entre méthode exacte et approchée pour des données de grande taille



GRA 4.1 – Comparaison entre la méthode exacte et la méthode approchée

Ce graphe compare le temps de calcul (CPU Time) entre deux méthodes de résolution de problèmes de tournées de véhicules spécifiquement pour le cas étudié MDHFVRP : la méthode approchée et la méthode exacte. L'axe vertical représente le temps de calcul en secondes (CPU Time), tandis que l'axe horizontal représente la taille du problème en termes de nombre d'entrepôts (m), de clients (n), et de types de véhicules (v).

Les deux courbes passent par différentes phases :

Phase A : représente la petite taille de problème « TAB 4.7 »

Phase B : représente la moyenne taille de problème « TAB 4.5 »

Phase C : représente la grande taille de problème « TAB 4.6 »

4.3.3 Interprétation des Résultats

❖ Petite Taille de Problème :

La méthode Branch-and-Cut offre des temps d'exécution généralement meilleurs que l'algorithme génétique pour les petites configurations, atteignant des solutions optimales dans tous les cas testés.

L'algorithme génétique, bien qu'ayant des temps d'exécution plus longs, maintient une qualité de solution optimale, ce qui indique une efficacité raisonnable dans la gestion des petits ensembles de données.

❖ Taille Moyenne de Problème :

Pour les configurations moyennes, La méthode exacte dépasse le seuil de temps (>1600 secondes), indiquant des limites pratiques en termes de complexité calculatoire.

L'algorithme génétique reste en deçà de ce seuil et offre des solutions "Proche-Optimal", suggérant une alternative viable lorsque le temps est un facteur critique.

❖ Grande Taille de Problème :

Dans les configurations plus grandes, La méthode exacte atteint systématiquement les limites de temps, tandis que l'algorithme génétique, bien que prenant plus de temps que dans les tailles moyennes, parvient à produire des solutions proches de l'optimal.

Ceci met en évidence l'adaptabilité de l'algorithme génétique à des problèmes de grande échelle, où la méthode exacte devient impraticable en raison des exigences temporelles.

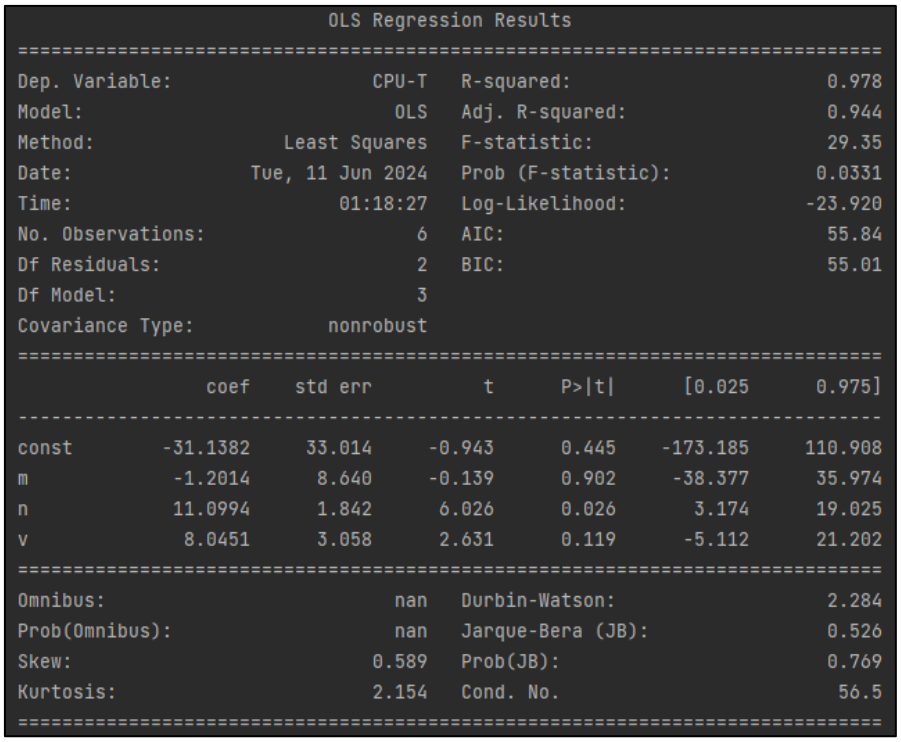
4.3.3.1 Facteur de Décision pour le Choix de la Méthode

En se basant sur les performances observées, nous pouvons établir un seuil de décision pour le choix de la méthode de résolution en fonction de la taille du problème, représentée par le nombre combiné d'entrepôts, de clients et de véhicules.

❖ Régression Linéaire

Appliquant la méthode des Moindres Carrés Ordinaires sur les mêmes variables pour chaque Méthode.

- Méthode approché



OLS Regression Results

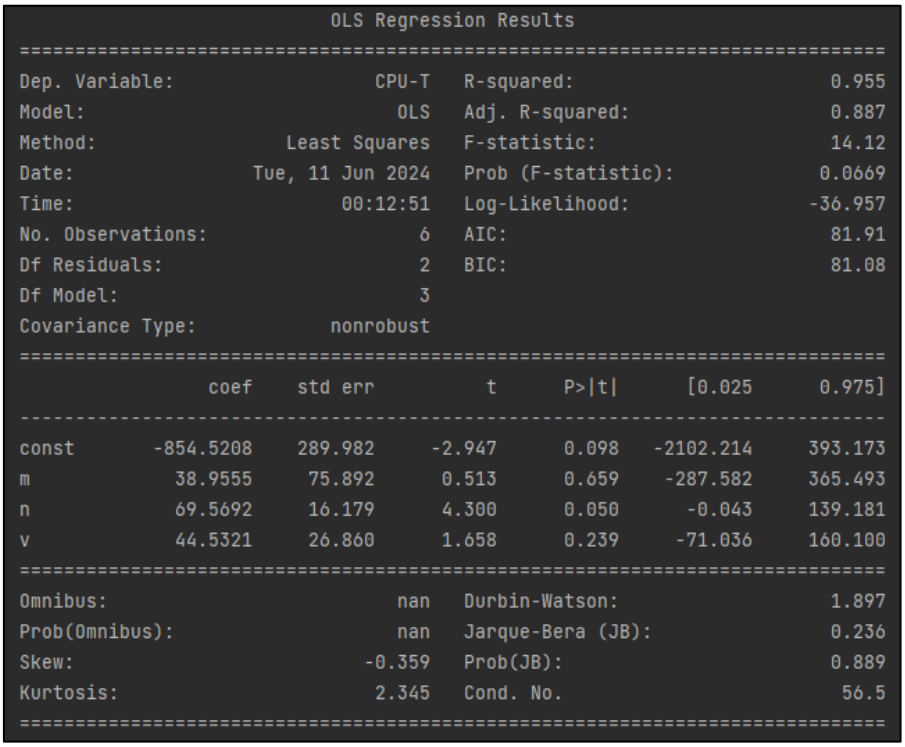
Dep. Variable:	CPU-T	R-squared:	0.978
Model:	OLS	Adj. R-squared:	0.944
Method:	Least Squares	F-statistic:	29.35
Date:	Tue, 11 Jun 2024	Prob (F-statistic):	0.0331
Time:	01:18:27	Log-Likelihood:	-23.920
No. Observations:	6	AIC:	55.84
Df Residuals:	2	BIC:	55.01
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-31.1382	33.014	-0.943	0.445	-173.185	110.908
m	-1.2014	8.640	-0.139	0.902	-38.377	35.974
n	11.0994	1.842	6.026	0.026	3.174	19.025
v	8.0451	3.058	2.631	0.119	-5.112	21.202

Omnibus:	nan	Durbin-Watson:	2.284
Prob(Omnibus):	nan	Jarque-Bera (JB):	0.526
Skew:	0.589	Prob(JB):	0.769
Kurtosis:	2.154	Cond. No.	56.5

FIG 4.2 – Résultat Statistique de la régression linéaire sur les données de la méthode approchée

- Méthode exacte



OLS Regression Results

Dep. Variable:	CPU-T	R-squared:	0.955
Model:	OLS	Adj. R-squared:	0.887
Method:	Least Squares	F-statistic:	14.12
Date:	Tue, 11 Jun 2024	Prob (F-statistic):	0.0669
Time:	00:12:51	Log-Likelihood:	-36.957
No. Observations:	6	AIC:	81.91
Df Residuals:	2	BIC:	81.08
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-854.5208	289.982	-2.947	0.098	-2102.214	393.173
m	38.9555	75.892	0.513	0.659	-287.582	365.493
n	69.5692	16.179	4.300	0.050	-0.043	139.181
v	44.5321	26.860	1.658	0.239	-71.036	160.100

Omnibus:	nan	Durbin-Watson:	1.897
Prob(Omnibus):	nan	Jarque-Bera (JB):	0.236
Skew:	-0.359	Prob(JB):	0.889
Kurtosis:	2.345	Cond. No.	56.5

FIG 4.3 – Résultat Statistique de la régression linéaire sur les données de la méthode exacte

- Approchée :
    - Constant δ : -31.14
    - Coefficients α, β, γ : [-1.20, 11.10, 38.05]

- Exacte :
    - Constant δ : -854.52
    - Coefficients α, β, γ : [38.96, 69.57, 44.53]

- Différences :
    - $a_{diff} = -40.16$
    - $b_{diff} = -58.47$
    - $c_{diff} = -36.49$
    - $d_{diff} = -823.38$

La forme générale des équations linéaires pour l'estimation du temps CPU peut être écrite comme suit :

Pour la méthode approchée :

$X_a = \alpha_a m + \beta_a n + \gamma_a v + \delta_a$

Pour la méthode exacte :

$X_b = \alpha_b m + \beta_b n + \gamma_b v + \delta_b$

Pour trouver le seuil où les deux méthodes prennent le même temps CPU :

$\alpha_a m + \beta_a n + \gamma_a v + \delta_a = \alpha_b m + \beta_b n + \gamma_b v + \delta_b$

Cela se simplifie en :

$(\alpha_a - \alpha_b)m + (\beta_a - \beta_b)n + (\gamma_a - \gamma_b)v = \delta_b - \delta_a$

Substituons les différences :

$\alpha_{diff}m + \beta_{diff}n + \gamma_{diff}v = \delta_{diff}$

$-40.16m - 58.47n - 36.49v = -823.38$

En simplifiant :

$X = 40.16m + 58.47n + 36.49v$

Cette équation représente le seuil X. Pour décider quelle méthode utiliser, nous comparons la valeur calculée de X pour une instance de problème donnée à un seuil prédéterminé :

- Si $X < 823.38$, la méthode exacte est préférable.
- Sinon, la méthode approchée est préférable.

4.3.4 Conclusion

Les méthodes exactes sont idéales pour obtenir des solutions optimales pour des problèmes de petite à moyenne taille mais deviennent inefficaces pour de grandes instances en raison de leur complexité et de leur temps de calcul élevé. Les méthodes approchées, comme les algorithmes génétiques, offrent des solutions acceptables plus rapidement et sont adaptées aux grands problèmes. La régression linéaire permet de définir un seuil X (823.38) pour décider de la méthode à utiliser : en dessous de ce seuil, on privilégie la méthode exacte ; au-dessus, la méthode approchée est recommandée.

4.4 Fonctionnement de VRS (Vehicule Routing Solver)

FIG 4.4 – L'interface de Solveur VRS

Le VRS est un logiciel solveur conçu pour résoudre efficacement les problèmes complexes de routage de véhicules. Son objectif principal est d'optimiser les trajets des flottes afin de réduire les coûts opérationnels et les temps de transport. Doté d'une interface intuitive.

4.4.1 Architecture logicielle

L'interface utilisateur du VRS est structurée en plusieurs onglets :

- Onglet "Clients"
- Onglet "Entrepôts"
- Onglet "Commandes"
- Onglet "Véhicules"
- Onglet "Solveur"
- Onglet "Rapport"
- Onglet "Carte Géographique"

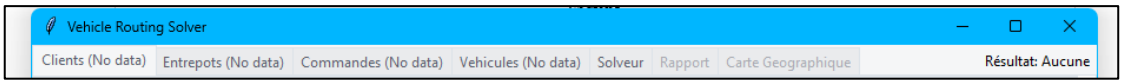


FIG 4.5 – Les Onglets de VRS

Chaque onglet est équipé de contrôles interactifs adaptés à ses fonctions spécifiques, rendant le VRS un outil puissant pour les opérations de planification de transport. L'interface est conçue pour être simple mais complète, permettant aux utilisateurs de gérer efficacement toutes les phases du processus de routage depuis une seule application.

4.4.2 Fonctionnalités et interaction utilisateur

❖ Clients

- Importation : Les utilisateurs peuvent charger les données clients à partir d'un fichier Excel.
- Gestion des données : Possibilité d'ajouter, modifier, et supprimer des informations client directement via l'interface graphique.

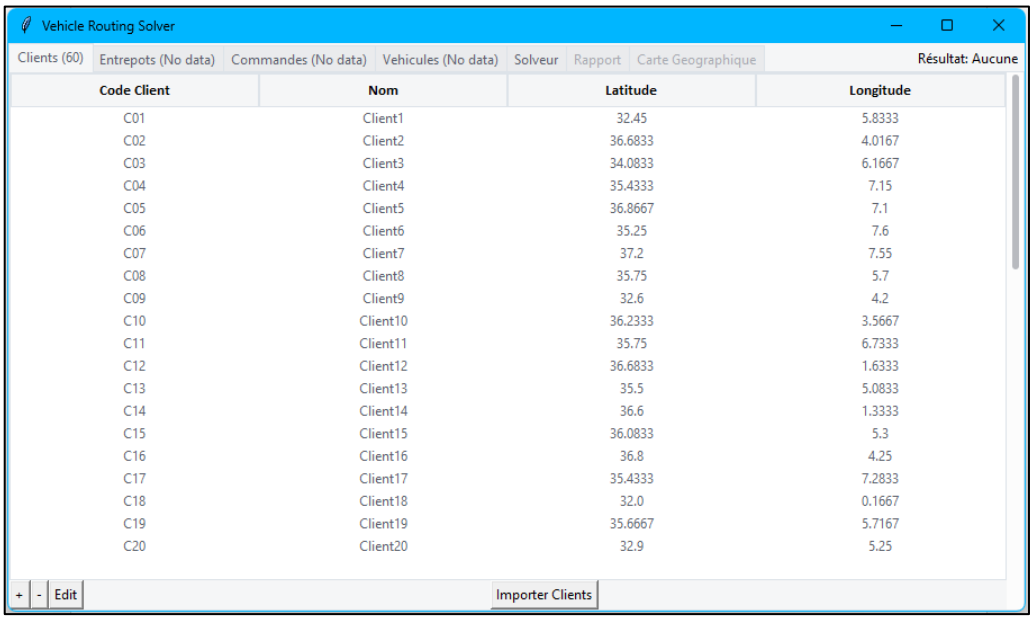


FIG 4.6 – VRS - Onglet « Clients »

❖ Entrepôts

- Importation : Similaire à l'onglet "Clients", permet l'importation des données d'entrepôts.
- Gestion des données : Ajout, suppression ou modification des détails des entrepôts directement.

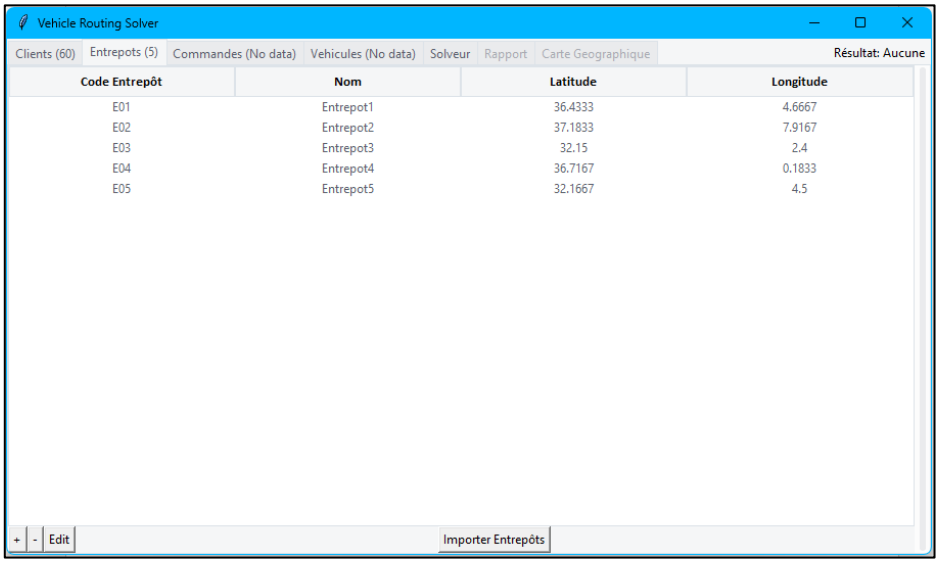


FIG 4.7 – VRS - Onglet « Entrepôts »

❖ Commandes

- Importation et gestion : Importation de données et manipulation (ajout, suppression, modification) des commandes à livrer.

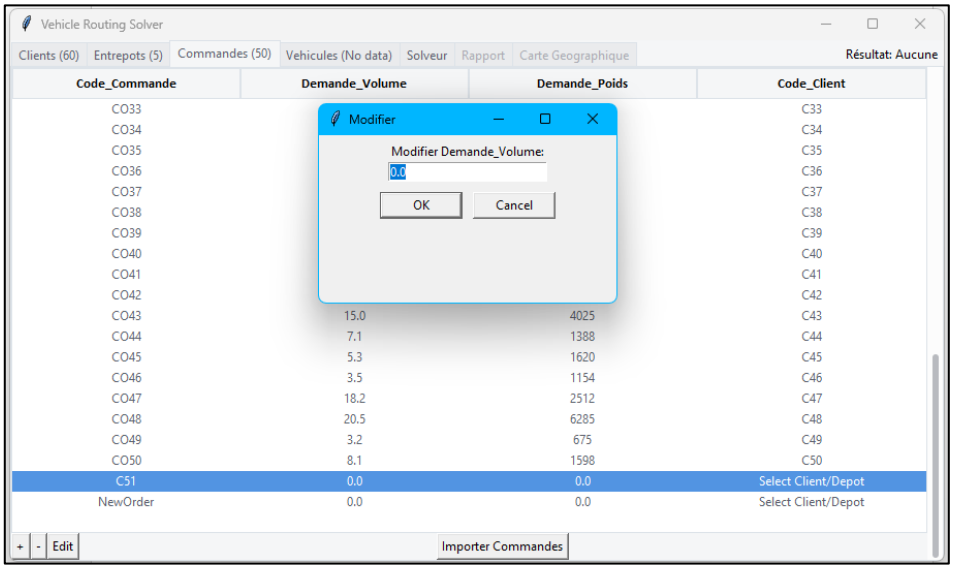


FIG 4.8 – VRS - Onglet « Commandes »

❖ Véhicules

- Configuration : Définition et ajustement des propriétés des véhicules, telles que le type, les capacités, et les coûts.

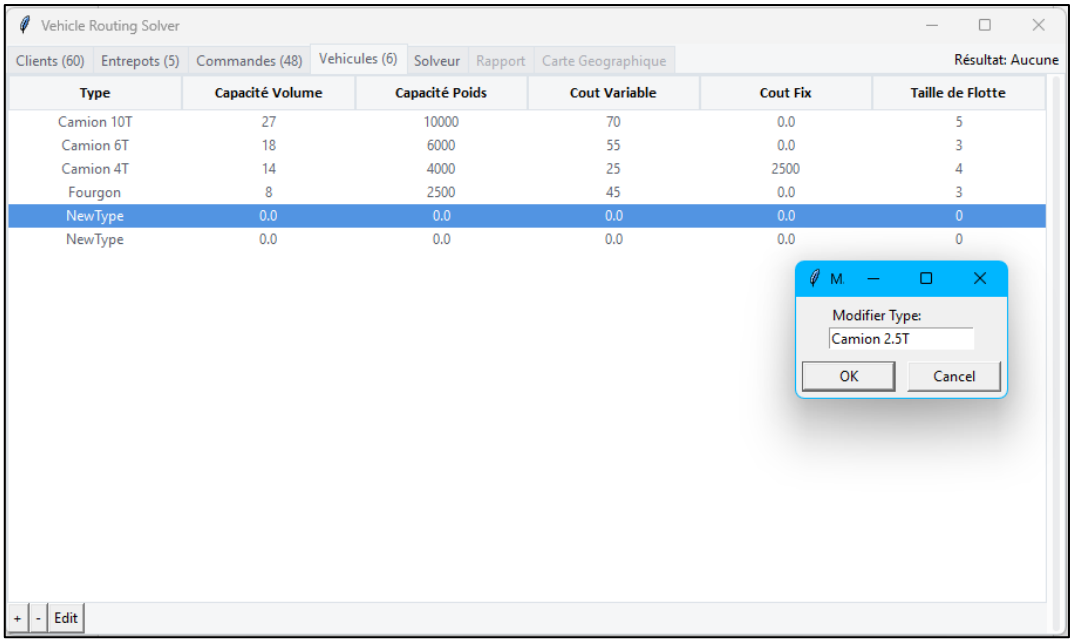


FIG 4.9 – VRS - Onglet « Véhicules »

❖ Solveur

- Configuration de la Route :

  Ouvert/Fermé : Choix entre une route ouverte où le véhicule ne retourne pas au point de départ, ou fermée où il retourne.

- Mode de résolution :

  Auto : Le système détermine automatiquement les meilleurs paramètres de l'algorithme basés sur les données.

  Manuelle : L'utilisateur peut ajuster les paramètres de l'algorithme génétique, tels que la taille de la population, le taux de croisement, le taux de mutation, etc.

- Démarrage de l'algorithme :

  Bouton "Démarrer" : Lance l'exécution de l'algorithme avec les paramètres fournis et montre la progression en temps réel.

  Affichage des logs : Visualisation en temps réel des logs de l'opération, aidant à diagnostiquer les problèmes ou suivre l'avancement.

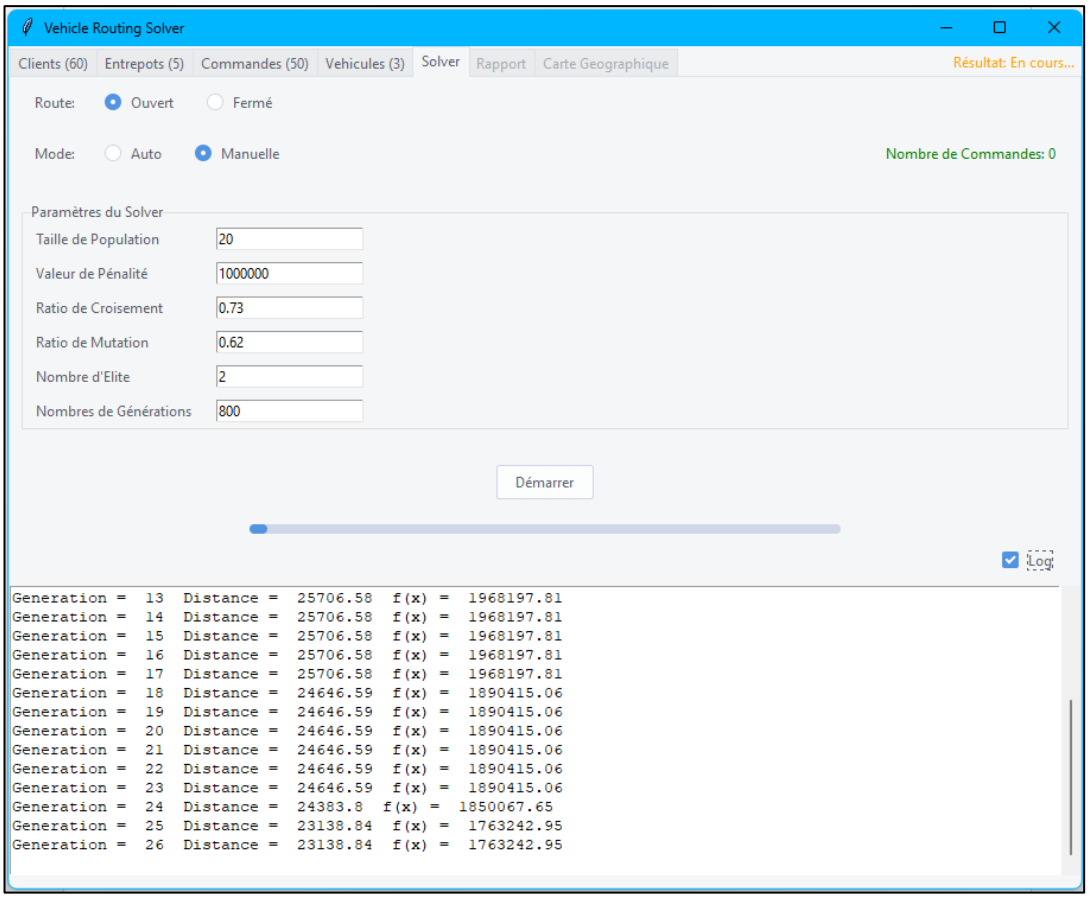


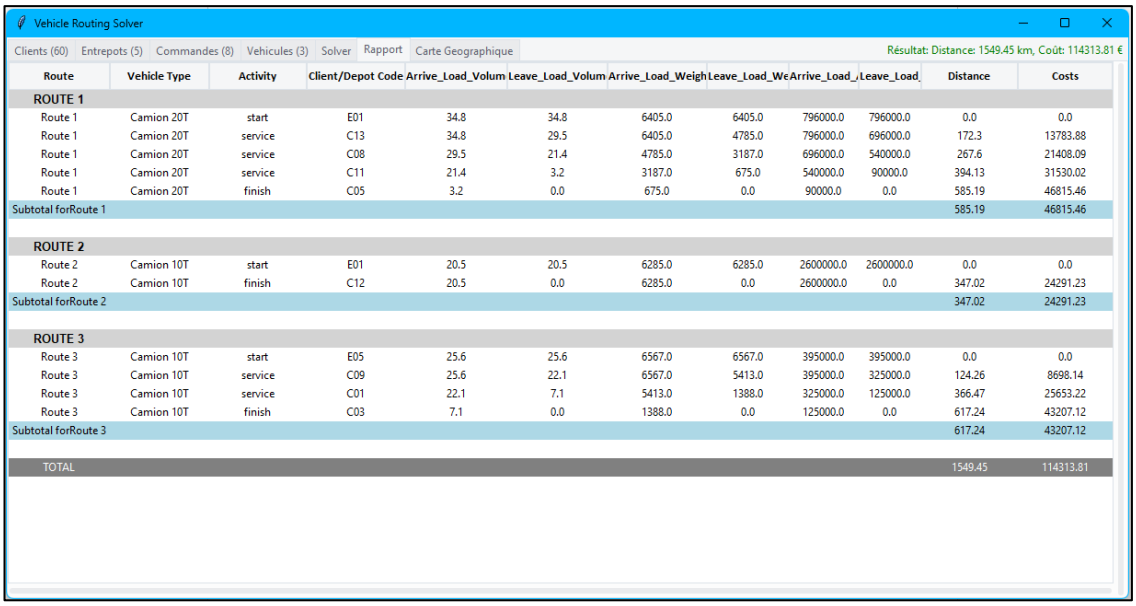
FIG 4.10 – VRS - Onglet « Solveur »

L'utilisateur doit attendre jusqu'à la barre de progression soit remplis a fin les onglets Rapport et Carte Géographique seront débloqués

4.4.3 Intégration des résultats

❖ Rapport

- Visualisation des résultats : Affiche les résultats de l'optimisation sous forme de tableau, montrant les détails des tournées par véhicule.



Route	Vehicle Type	Activity	Client/Depot Code	Arrive_Load_Volum	Leave_Load_Volum	Arrive_Load_Weigh	Leave_Load_We	Arrive_Load_	Leave_Load	Distance	Costs
ROUTE 1											
Route 1	Camion 20T	start	E01	34.8	34.8	6405.0	6405.0	796000.0	796000.0	0.0	0.0
Route 1	Camion 20T	service	C13	34.8	29.5	6405.0	4785.0	796000.0	696000.0	172.3	13783.88
Route 1	Camion 20T	service	C08	29.5	21.4	4785.0	3187.0	696000.0	540000.0	267.6	21408.09
Route 1	Camion 20T	service	C11	21.4	3.2	3187.0	675.0	540000.0	90000.0	394.13	31530.02
Route 1	Camion 20T	finish	C05	3.2	0.0	675.0	0.0	90000.0	0.0	585.19	46815.46
Subtotal forRoute 1										585.19	46815.46
ROUTE 2											
Route 2	Camion 10T	start	E01	20.5	20.5	6285.0	6285.0	2600000.0	2600000.0	0.0	0.0
Route 2	Camion 10T	finish	C12	20.5	0.0	6285.0	0.0	2600000.0	0.0	347.02	24291.23
Subtotal forRoute 2										347.02	24291.23
ROUTE 3											
Route 3	Camion 10T	start	E05	25.6	25.6	6567.0	6567.0	395000.0	395000.0	0.0	0.0
Route 3	Camion 10T	service	C09	25.6	22.1	6567.0	5413.0	395000.0	325000.0	124.26	8698.14
Route 3	Camion 10T	service	C01	22.1	7.1	5413.0	1388.0	325000.0	125000.0	366.47	25653.22
Route 3	Camion 10T	finish	C03	7.1	0.0	1388.0	0.0	125000.0	0.0	617.24	43207.12
Subtotal forRoute 3										617.24	43207.12
TOTAL										1549.45	114313.81

FIG 4.11 – VRS - Onglet « Rapport »

Dans cet exemple les trois routes combinées parcourent une distance totale de 1549.45 km, avec un coût total de 114313.81. Chaque route a été planifiée pour optimiser l'utilisation des véhicules, minimiser les coûts associés, tout en respectant les contraintes de chargement et de livraison des clients.

❖ Carte Géographique

- Cartographie des routes : Affichage graphique des routes optimisées sur une carte visualisation dans un navigateur web avec l'option de séparation chaque tournée par véhicule.

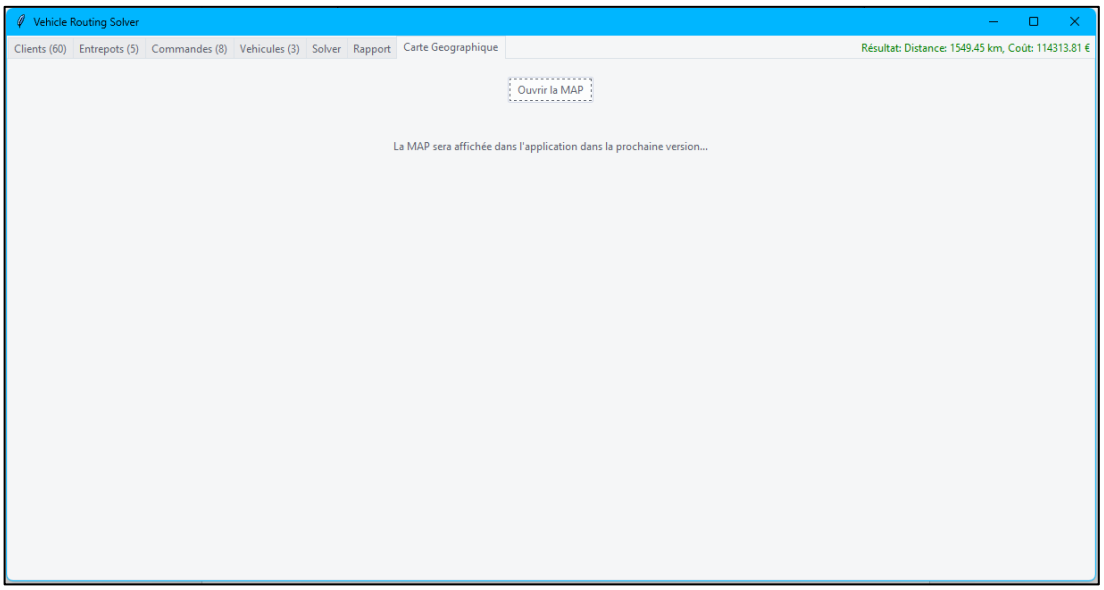


FIG 4.12 – VRS - Onglet « Carte Géographique »

FIG 4.13 – VRS - La tournée optimal tracée sur la carte géographique

Chaque tournée est représentée par une couleur différente, et les numéros inscrits sur les nœuds indiquent l'ordre dans lequel ils sont visités.

Dans les prochaines versions du programme, plusieurs améliorations sont prévues. La carte géographique sera intégrée directement dans l'interface utilisateur avec des options avancées telles que l'animation des trajets, permettant une visualisation dynamique des parcours des véhicules en temps réel, des options de filtrage et de zoom pour une meilleure gestion des trajets, et la visualisation des routes optimisées avec des légendes claires et des indicateurs de performance. De plus, la méthode exacte sera migrée vers un module CPLEX, qui a été observé comme étant particulièrement utile pour les problèmes de petite taille. Le programme sélectionnera automatiquement la méthode de résolution la plus appropriée (exacte ou approchée) en fonction de la taille et de la complexité du problème, assurant ainsi une optimisation efficace des ressources et des temps de calcul. Pour le solveur basé sur l'algorithme génétique, la parallélisation sera appliquée afin de tirer parti des cœurs processeur disponibles (multiprocessing), améliorant ainsi les performances. En outre, de nouvelles contraintes réelles seront ajoutées, comme les fenêtres de temps (TimeWindow) et les livraisons fractionnées (Split Delivery), permettant de traiter des problèmes encore plus complexes et réalistes.

4.4.4 Évaluation l'efficacité de Solveur VRS

Dans le but d'évaluer les performances du solveur VRS développé dans le cadre de ce projet, nous allons comparer les résultats qu'il fournit avec ceux de la méthode manuelle actuellement utilisée par le logisticien de l'entreprise.

La comparaison portera sur trois échantillons de données réelles fournis par l'entreprise, représentant différents cas de problèmes de tournées à résoudre. Pour chaque échantillon, nous présenterons d'abord les données. Ensuite, nous exposerons les résultats de résolution obtenus par l'approche manuelle du logisticien d'une part, et par le solveur VRS d'autre part.

Les résultats seront analysés et comparés selon deux critères principaux :

- L'optimalité de la solution trouvée, évaluée par la valeur de la fonction objectif à minimiser (coût total des livraisons).
- Le temps de résolution requis pour aboutir à la solution proposée.

Soit :

Dv : Demande volume

Dp : Demande poids

M : Montant (Unité Monétaire)

Tv : Type de véhicule

Dt : Distance totale

Ct : Coût total

T : Temps de programmation ou exécution

❖ Programme 01

- Données

  Commandes

Code_Commande	Dv (m^3)	Dp (Kg)	M (Um)	Code_Client
CO01	0.5638	158.15	46800	C01
CO02	0.1561	73.88	41625	C02

CO03	0.225	46.2	14377.5	C03
CO04	0.7551	252	134775	C04
CO05	1.6099	408.96	178650	C05
CO06	0.2074	100	48150	C06
CO07	0.0552	12	8010	C07
CO08	1.461	471.6	113400	C08
CO09	5.076	1612.9	389025	C09
CO10	0.8453	229.184	107766	C10
CO11	0.5	120	59400	C11

TAB 4.8 – Données de commandes de programme 01

Véhicules

Type	1	2	3	4	5	6	7	8	9	10	11	12	13
Disponibilité	9	4	7	4	3	2	3	6	2	3	4	5	3

TAB 4.9 – Données de véhicules disponibles pour le programme 01

- Résultats

Méthode	**Tournée**	**Tv**	**Dt (Km)**	**Ct (Um)**	**T (min)**
Entreprise	E03–C06–C04–C02–C01–E03	4	1345.15	60531.75	15
	E03–C03–E03	12			
	E02–C07–C09–C08–C11–C10–E02	1			
VRS	E02–C07–C09–C08–C11–C10-E02	5	1282.16	57697.39	1.8
	E03–C03–C05–C02–C04–C01–C06–E03	5			

TAB 4.10 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 01

La solution proposée par VRS montre une optimisation de la distance de trajet et des coûts par rapport à la méthode utilisée par l'entreprise, avec une réduction significative du temps de programmation.

❖ Programme 02

- Données

Commandes

Code_Commande	**Dv (m^3)**	**Dp (Kg)**	**M (Um)**	**Code_Client**
CO01	0.4636	129	35010	C03
CO02	0.6655	289	116910	C12
CO03	4.87	1572	378000	C13

CO04	1	240	118800	C14
CO05	0.2895	164.092	111177	C15
CO06	0.8012	260.516	101007	C16
CO07	0.1045	39.6	17793	C17
CO08	0.316	132	41040	C18
CO09	0.5371	246	75420	C19
CO10	0.124	41	22990.5	C20
CO11	0.1765	71.88	38815.2	C21
CO12	1.7424	660	296550	C22
CO13	0.8712	330	148275	C23

TAB 4.11 – Données de commandes de programme 02

Véhicules

Type	1	2	3	4	5	6	7	8	9	10	11	12	13
Disponibilité	8	4	7	3	3	2	3	6	2	3	4	4	3

TAB 4.12 – Données de véhicules disponibles pour le programme 02

- Résultats

Méthode	**Tournée**	**Tv**	**Dt (Km)**	**Ct (Um)**	**T (min)**
Entreprise	E03–C22–C23–C12–C03–E03	4	1511.22	68004.90	25
	E02–C13-C14-C15–E02	4			
	E01–C18–C21–C17–E01	11			
	E01-C19-C16-C20-E01	12			
VRS	E01–C16–C17–C21–C18–C20-C19-E01	5	1296.59	58346.51	2.9
	E02-C13-C15-C14-E02	5			
	E03–C03–C12–C23–C22–E03	5			

TAB 4.13 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 02

VRS améliore considérablement l'efficacité en réduisant à la fois la distance de trajet et les coûts pour les tournées, démontrant une optimisation supérieure comparée à la méthode de l'entreprise.

❖ Programme 03

- Données

Commandes

Code_Commande	**Dv (m^3)**	**Dp (Kg)**	**M (Um)**	**Code_Client**
CO01	2.0415	653.792	188145	C24
CO02	0.3049	75.02	40995	C25
CO03	1.4166	348.8	143550	C26

CO04	1.3644	388.57	151924.5	C27
CO05	0.248	38	45441	C28
CO06	0.58	181.36	76072.5	C29
CO07	0.1614	164	45108	C15
CO08	0.4795	200.96	98158.5	C07

TAB 4.14 – Données de commandes de programme 03

Véhicules

Type	1	2	3	4	5	6	7	8	9	10	11	12	13
Disponibilité	8	4	7	1	3	2	3	6	2	3	3	3	3

TAB 4.15 – Données de véhicules disponibles pour le programme 03

- Résultats

Méthode	**Tournée**	**Tv**	**Dt (Km)**	**Ct (Um)**	**T (min)**
Entreprise	E01–C24–C27–C25–C26–C28-C29-E01	4	1041.43	46864.35	10
	E02–C07-C15–E02	4			
VRS	E02–C15–C07-E02	5	1041.43	46864.35	1.5
	E02-C24-C29-C28-C26-C25-C27-E02	5			

TAB 4.16 – Comparaison entre résultat obtenu par VRS et le logisticien de l'entreprise pour le programme 03

La solution de VRS, bien que n'ayant pas amélioré les distances parcourues ou les coûts, par contre en réduisant ainsi considérablement le temps de programmation par rapport à la méthode de l'entreprise.

Cette analyse comparative permettra de mettre en évidence les forces et les faiblesses du solveur VRS par rapport à la méthode manuelle, et d'identifier les pistes d'améliorations possibles. Elle constituera ainsi une étape essentielle dans l'évaluation de la solution développée et dans la détermination de sa capacité à optimiser la gestion des tournées de livraison pour l'entreprise.

4.4.5 Impact sur CPH Agriculture

❖ Réduction des coûts de transport :

L'utilisation du VRS réduit les distances parcourues par les véhicules, diminuant ainsi les coûts liés au carburant et à l'entretien. Cela se traduit par une diminution des dépenses globales liées au transport.

❖ Amélioration de l'efficacité logistique :

Le VRS maximise l'utilisation de la capacité de chargement des véhicules, limitant le nombre de véhicules nécessaires et améliorant l'efficacité logistique.

❖ Gain de temps :

Le gain de temps réalisé dans la planification des livraisons grâce au VRS permet aux gestionnaires de se concentrer sur d'autres tâches stratégiques, augmentant la productivité globale.

❖ Réduction de l'empreinte carbone :

En réduisant les distances parcourues, le VRS contribue à diminuer les émissions de gaz à effet de serre, améliorant ainsi l'empreinte écologique de l'entreprise.

❖ Amélioration du service client :

L'optimisation des tournées assure des livraisons plus rapides et fiables, accroissant la satisfaction client et renforçant la fidélité de la clientèle.

❖ Adaptabilité aux contraintes réelles :

Le Future VRS prend en compte les contraintes spécifiques du secteur agricole, comme les fenêtres temporelles et les capacités de chargement variables, aussi des obstacles législatifs, permettant ainsi à CPH Agriculture de répondre efficacement aux exigences opérationnelles et réglementaires.

Conclusion

L'objectif principal de ce mémoire était de résoudre le problème de tournées de véhicules multi-dépôts avec flotte hétérogène (MDHFVRP) pour l'entreprise CPH Agriculture. Deux approches distinctes ont été explorées et implémentées, offrant ainsi des solutions à la fois exactes et approchées.

Dans un premier temps, une modélisation mathématique du problème a été formulée en utilisant la programmation linéaire en nombres entiers. Cette formulation rigoureuse a permis de capturer avec précision toutes les contraintes opérationnelles inhérentes au MDHFVRP. Le modèle a ensuite été résolu de manière exacte à l'aide du solveur CPLEX. Si cette approche garantit l'optimalité de la solution, elle reste cependant limitée par la complexité exponentielle du problème, rendant son utilisation difficile pour les instances de grande taille.

Afin de pallier cette limitation, une méta-heuristique basée sur l'algorithme génétique a été développée. Inspirée des principes de l'évolution naturelle, cette méthode approchée permet d'explorer efficacement l'espace de recherche pour trouver des solutions de bonne qualité en un temps de calcul raisonnable. Les résultats obtenus se sont avérés très satisfaisants, offrant des solutions quasi-optimales pour des instances de tailles réalistes.

Dans le but de faciliter l'utilisation de ces méthodes par les logisticiens de CPH Agriculture, une interface graphique conviviale a été conçue et implémentée. Celle-ci permet de saisir les données d'entrée, d'exécuter la méta-heuristique génétique et de visualiser les tournées optimisées obtenues. Cette interface offre ainsi un outil pratique pour la planification et l'optimisation des opérations logistiques de l'entreprise.

Les contributions majeures de ce travail résident dans le développement d'une solution complète, à la fois théorique et pratique, pour le MDHFVRP dans le contexte spécifique de CPH Agriculture. Les approches proposées ont démontré leur efficacité et leur capacité à générer des solutions de qualité supérieure par rapport aux méthodes de planification manuelles traditionnellement utilisées.

Cependant, certaines limitations subsistent et ouvrent la voie à des perspectives d'amélioration future. Notamment, l'intégration de contraintes supplémentaires telles que les fenêtres de temps ou les contraintes de précédence pourrait être envisagée pour une modélisation encore plus réaliste du problème. De plus, l'exploration d'autres méta-heuristiques ou l'hybridation de plusieurs approches pourrait potentiellement conduire à de meilleures performances.

En conclusion, ce mémoire a permis de proposer des solutions innovantes et performantes pour relever les défis logistiques complexes auxquels fait face CPH Agriculture. Les méthodes développées offrent une base solide pour une optimisation efficace des tournées de livraison, contribuant ainsi à une gestion plus durable et rentable des opérations logistiques de l'entreprise.

Annexe :

```python

import folium
import folium.plugins
from folium.plugins import BeautifyIcon
import pandas as pd
import random
import requests
import folium
import numpy as np
import copy
import os
import time as tm
import webbrowser

from itertools import cycle
import matplotlib
matplotlib.use('TkAgg')  # Vous pouvez aussi essayer 'Agg', 'Qt5Agg', 'GTK3Agg', etc.
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from geopy.distance import geodesic
```

Code : Ensemble des bibliothèques utilisées

```python
# GA-VRP Function
def genetic_algorithm_vrp(coordinates, distance_matrix_local, parameters, fixed_cost, variable_cost, capacity_volume,
                          capacity_weight, population_size=5, vehicle_types=1, n_depots=1, route='closed',fleet_size=[],
                          mutation_rate=0.1, crossover_rate=0.1, elite=0, generations=50, penalty_value=1000,
                          graph=True, map=True, selection='rw'):
    start = tm.time()
    count = 0
    solution_report = ['None']
    max_capacity_volume, max_capacity_weight = copy.deepcopy(capacity_volume), copy.deepcopy(capacity_weight)
    for i in range(0, n_depots):
        parameters[i, 0] = 0
        parameters[i, 1] = 0
        parameters[i, 2] = 0
    population = initial_population(coordinates, distance_matrix_local, population_size=population_size,
                                    vehicle_types=vehicle_types, n_depots=n_depots)
    cost, population = target_function(population, distance_matrix_local, parameters, fixed_cost, variable_cost,
                                       max_capacity_volume, max_capacity_weight, penalty_value,route=route,
                                       fleet_size=fleet_size)
    cost, population = (list(t) for t in zip(*sorted(zip(cost, population))))
    if (selection == 'rw'):
        fitness = fitness_function(cost, population_size)
    elif (selection == 'rb'):
        rank = [[i] for i in range(1, len(cost) + 1)]
        fitness = fitness_function(rank, population_size)
    elite_ind = elite_distance(population[0], distance_matrix_local, route=route)
    cost = copy.deepcopy(cost)
    elite_cst = copy.deepcopy(cost[0][0])
    solution = copy.deepcopy(population[0])
    print('Generation = ', count, ' Distance = ', elite_ind, ' f(x) = ', round(elite_cst, 2))

    # Définir le graphique initial
    plt.ion()  # Activer le mode interactif
    fig, ax = plt.subplots()
    line, = ax.plot([], [], 'r-', label='Coût Total')  # Initialiser une ligne vide
    ax.set_xlim(0, generations)
    ax.set_ylim(0, elite_cst*1.25)  # Estimation initiale de max_cost; ajustez selon vos données
    plt.title('Evolution of Total Cost over Generations')
    plt.xlabel('Generation')
    plt.ylabel('Total Cost')
```

Code : Algorithme génétique #1 [30]

```
    plt.legend()
    plt.show()

    elite_costs = []  # Stocker les coûts élites pour chaque génération
    generations_x = []  # Stocker les indices de génération pour l'axe x

    while (count <= generations - 1):
        offspring = breeding(cost, population, fitness, distance_matrix_local, n_depots, elite, max_capacity_volume,
                             max_capacity_weight, fixed_cost, variable_cost, penalty_value, parameters, route,
                             vehicle_types, fleet_size, crossover_rate=crossover_rate)
        offspring = mutation(offspring, mutation_rate=mutation_rate, elite=elite)
        cost, population = target_function(offspring, distance_matrix_local, parameters, fixed_cost, variable_cost,
                                           max_capacity_volume, max_capacity_weight, penalty_value, route=route,
                                           fleet_size=fleet_size)
        cost, population = (list(t) for t in zip(*sorted(zip(cost, population))))
        elite_child = elite_distance(population[0], distance_matrix_local, route=route)
        if (selection == 'rw'):
            fitness = fitness_function(cost, population_size)
        elif (selection == 'rb'):
            rank = [[i] for i in range(1, len(cost) + 1)]
            fitness = fitness_function(rank, population_size)
        if (elite_ind > elite_child):
            elite_ind = elite_child
            solution = copy.deepcopy(population[0])
            elite_cst = copy.deepcopy(cost[0][0])

        # Mise à jour des listes pour les générations et les coûts
        elite_costs.append(elite_cst)
        generations_x.append(count)
        ax.set_xlim(0, count*1.5)
        line.set_xdata(generations_x)
        line.set_ydata(elite_costs)
        ax.relim()
        ax.autoscale_view()
        fig.canvas.draw()
        fig.canvas.flush_events()

        count = count + 1
        print('Generation = ', count, ' Distance = ', elite_ind, ' f(x) = ', round(elite_cst, 2))

    plt.ioff()  # Désactiver le mode interactif
    plt.show()  # Afficher le graphique final


    if (graph == True):
        plot_tour_coordinates(coordinates, solution, n_depots=n_depots, route=route)

    if map == True:
        depot_indices = [i for i in range(n_depots)]
        draw_routes_on_map(coordinates, solution, depot_indices, output_file='final_solution_map.html')

        # Ouvrir la carte dans un navigateur web
        webbrowser.open_new_tab('final_solution_map.html')

    solution_report = show_report(solution, distance_matrix_local, parameters, fixed_cost, variable_cost,
                                  route=route)
    end = tm.time()
    print('Algorithm Time: ', round((end - start), 2), ' seconds')
    return solution_report, solution
```

Code : Algorithme génétique #2 [30]

```python
# Function: Initial Population
def initial_population(coordinates='none', distance_matrix_local='none', population_size=5, vehicle_types=1,n_depots=1):
    try:
        distance_matrix_local.shape[0]
    except:
        distance_matrix_local = calculate_real_distances(coordinates)
    depots = [[i] for i in range(0, n_depots)]
    vehicles = [[i] for i in range(0, vehicle_types)]
    clients = list(range(n_depots, distance_matrix_local.shape[0]))
    population = []
    for i in range(0, population_size):
        clients_temp = copy.deepcopy(clients)
        routes = []
        routes_depot = []
        routes_vehicles = []
        while (len(clients_temp) > 0):
            e = random.sample(vehicles, 1)[0]
            d = random.sample(depots, 1)[0]

            c = random.sample(clients_temp, random.randint(1, len(clients_temp)))
            routes_vehicles.append(e)
            routes_depot.append(d)
            routes.append(c)
            clients_temp = [item for item in clients_temp if item not in c]
        population.append([routes_depot, routes, routes_vehicles])
    return population

```

Code : Initialisation de la population [30]

```python
# Function: Route Evaluation & Correction
def target_function(population, distance_matrix_local, parameters, fixed_cost, variable_cost, capacity_volume,
                    capacity_weight, penalty_value, route, fleet_size=[]):
    cost = [[0] for i in range(len(population))]
    flt_cnt = [0] * len(fleet_size)
    if (route == 'open'):
        end = 2
    else:
        end = 1
    for k in range(0, len(population)):  # k individuals
        individual = copy.deepcopy(population[k])
        size = len(individual[1])
        i = 0
        pnlt = 0
        flt_cnt = [0] * len(fleet_size)
        while (size > i):  # i subroutes
            # Extraction des coordonnées de tous les points (dépôts et clients) dans la sous-route
            points_coord = [coordinates[depot_client_index] for depot_client_index in
                            individual[0][i] + individual[1][i]]
            # Vérification de doublons dans les coordonnées des points
            if len(points_coord) != len(set(tuple(coord) for coord in points_coord)):
                pnlt += penalty_value  # Appliquer une pénalité si des doublons sont trouvés

            dist = evaluate_distance(distance_matrix_local, individual[0][i], individual[1][i])

            cap_volume_used = evaluate_capacity_volume(parameters, depot=individual[0][i], subroute=individual[1][i])
            cap_weight_used = evaluate_capacity_weight(parameters, depot=individual[0][i], subroute=individual[1][i])

            cost_s = evaluate_cost(dist, depot=individual[0][i], subroute=individual[1][i],
                                   fixed_cost=[fixed_cost[individual[2][i][0]]],
                                   variable_cost=[variable_cost[individual[2][i][0]]])
            pnlt = pnlt + penalty_value * (sum(v > capacity_volume[individual[2][i][0]] for v in cap_volume_used[0:-1])
                                           + sum(w > capacity_weight[individual[2][i][0]] for w in cap_weight_used[0:-1]))
            if (len(fleet_size) > 0):
                flt_cnt[individual[2][i][0]] = flt_cnt[individual[2][i][0]] + 1
```

Code : Evaluation #1 [30]

```
            if (size <= i + 1):
                for v in range(0, len(fleet_size)):
                    v_sum = flt_cnt[v] - fleet_size[v]
                    if (v_sum > 0):
                        pnlt = pnlt + v_sum

            cost[k][0] = cost[k][0] + cost_s[-end] + pnlt
            size = len(individual[1])
            i = i + 1
    cost_total = copy.deepcopy(cost)
    return cost_total, population
```

Code : Evaluation #2 [30]

```
# Function: Breeding
def breeding(cost, population, fitness, distance_matrix_local, n_depots, elite, capacity_volume, capacity_weight,
             fixed_cost, variable_cost, penalty_value, parameters, route, vehicle_types, fleet_size, crossover_rate):
    offspring = copy.deepcopy(population)
    if (elite > 0):
        cost, population = (list(t) for t in zip(*sorted(zip(cost, population))))
        for i in range(0, elite):
            offspring[i] = copy.deepcopy(population[i])
    for i in range(elite, len(offspring)):
        parent_1, parent_2 = roulette_wheel(fitness), roulette_wheel(fitness)
        while parent_1 == parent_2:
            parent_2 = random.sample(range(0, len(population) - 1), 1)[0]
        parent_1 = copy.deepcopy(population[parent_1])
        parent_2 = copy.deepcopy(population[parent_2])
        probability = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
        if (probability <= crossover_rate):
            rand = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
            # VRP - Crossover
            if ((len(parent_1[1]) > 1 and len(parent_2[1]) > 1)):
                if (rand > 0.5):
                    offspring[i] = crossover_vrp_brbax(parent_1, parent_2)
                    offspring[i] = crossover_vrp_bcr(offspring[i], parent_2, distance_matrix_local, capacity_volume,
                                                     capacity_weight, fixed_cost, variable_cost, penalty_value,
                                                     parameters=parameters, route=route)
                elif (rand <= 0.5):
                    offspring[i] = crossover_vrp_brbax(parent_2, parent_1)
                    offspring[i] = crossover_vrp_bcr(offspring[i], parent_1, distance_matrix_local, capacity_volume,
                                                     capacity_weight, fixed_cost, variable_cost, penalty_value,
                                                     parameters=parameters, route=route)
        if (n_depots > 1):
            offspring[i] = evaluate_depot(n_depots, offspring[i], distance_matrix_local)
        if (vehicle_types > 1):
            offspring[i] = evaluate_vehicle(vehicle_types, offspring[i], distance_matrix_local, parameters,
                                            fixed_cost, variable_cost, capacity_volume, capacity_weight, penalty_value,
                                            route, fleet_size)
    offspring[i] = cap_break( offspring[i], parameters, capacity_volume, capacity_weight)
    return offspring
```

Code Reproduction (Sélection – Croisement) [30]

```python
# Function: Mutation
def mutation(offspring, mutation_rate, elite):
    for i in range(elite, len(offspring)):
        probability = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
        if (probability <= mutation_rate):
            rand = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
            if (rand <= 0.5):
                offspring[i] = mutation_vrp_insertion(offspring[i])
            elif (rand > 0.5):
                offspring[i] = mutation_vrp_swap(offspring[i])
        for k in range(0, len(offspring[i][1])):
            if (len(offspring[i][1][k]) >= 2):
                probability = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
                if (probability <= mutation_rate):
                    rand = int.from_bytes(os.urandom(8), byteorder='big') / ((1 << 64) - 1)
                    cut = random.sample(list(range(0, len(offspring[i][1][k]))), 2)
                    cut.sort()
                    C = offspring[i][1][k][cut[0]:cut[1] + 1]
                    if (rand <= 0.5):
                        random.shuffle(C)
                    elif (rand > 0.5):
                        C.reverse()
                    offspring[i][1][k][cut[0]:cut[1] + 1] = C
    return offspring

```

Code : Mutation [30]

```python
# Paramètres
n_depots     =  2          # Nombre d'entrepôts
route        = 'closed'     # 'open', 'closed', OpenVRP ou ClosedVRP
graph        = False        # True, False
map          = True         # True, False

# Paramètres - Véhicule
vehicle_types =   11        # Nombre de types de véhicule
fixed_cost    = [ 0,0,0,0,0,0,0,0,0,0,0]      # Cout fix pour chaque type
variable_cost = [ 45,70,45,45,45,50,55,55,60,50,80]      # Cout variable unitaire pour chaque type
capacity_volume      = [8,27,8,8,10,14,16,18,21,12,60]      # Capacité volume pour chaque type
capacity_weight      = [2500,10000,3000,2000,2500,4000,5000,6000,7000,3500,20000]      # Capacité poids pour chaque type
fleet_size    = [ 1,3,5,1,2,2,4,4,3,5,7]      # Taille de flotte pour chaque type

# Paramètres - Algorithme Génétique
penalty_value   = 1000000    # Valeur de pénalité
population_size = 20       # Taille de population
mutation_rate   = 0.9      # Taux de mutation
crossover_rate = 0.9  # Taux de croisement
elite           = 1        # Elitisme
generations     = 1000     # Nombre de générations


#########################################################################################################################
```

Code : Paramètres [30]

```python
import pandas as pd
import numpy as np
import os
import random
import datetime
import MDVRPUIPARAMTEST  # fichier MDVRPUIPARAMTEST.py possede le code de l'algorithme complet specifique pour les tests

def run_simulation(params):
    cost, execution_time = MDVRPUIPARAMTEST.run_mdvrp(**params)
    return cost, execution_time

def generate_random_configurations(num_configs):
    configurations = []
    for _ in range(num_configs):
        config = {
            'population_size': random.choice([10, 20, 30, 40]), # Séléction Aleatoire - la taille de la population
            'mutation_rate': round(random.uniform(0.1, 0.9), 2), # Séléction Aleatoire - le taux de mutation
            'crossover_rate': round(random.uniform(0.1, 0.9), 2), # Séléction Aleatoire - le taux de croisement
            'elite': random.randint(0, 3), # Sélection Aléatoire - Nombre d'elites

            # Sélection Aléatoire - Nombre de generations
            'generations': random.choice([100, 300, 500, 750, 1000, 1500, 2000, 3000])
        }
        configurations.append(config)
    return configurations

def main():
    desktop_path = os.path.join(os.path.expanduser('~'), 'Desktop')
    placement_data_path = os.path.join(desktop_path, 'Emplacements.xlsx')
    commands_data_path = os.path.join(desktop_path, 'Commandes.xlsx')

    placement_data = pd.read_excel(placement_data_path)
    commands_data = pd.read_excel(commands_data_path)

    n_depots = 2
    n_clients = len(placement_data) - n_depots
    vehicle_types = 13
    test_number = 3


    ga_configurations = generate_random_configurations(test_number)

    results = []

    for idx, config in enumerate(ga_configurations):
        params = {
            'placement_data': placement_data,
            'commands_data': commands_data,
            'n_depots': n_depots,
            'route': 'closed',
            'vehicle_types': vehicle_types,
            'fixed_cost': [0]*vehicle_types,
            'variable_cost': [45, 70, 45, 45, 45, 50, 55, 55, 60, 50, 45, 45, 80],
            'capacity_volume': [8, 27, 8, 8, 10, 14, 16, 18, 21, 12, 2.5, 2, 60],
            'capacity_weight': [2500, 10000, 3000, 2000, 2500, 4000, 5000, 6000, 7000, 3500, 1200, 1200, 20000],
            'fleet_size': [8]*vehicle_types,
            'penalty_value': 1000000,
            **config
        }
        cost, execution_time = run_simulation(params)
        results.append({
            'Number of Depots': n_depots,
            'Number of Clients': n_clients,
            'Number of Vehicles': vehicle_types,
            'Population Size': config['population_size'],
            'Penalty Value': params['penalty_value'],
            'Mutation Rate': config['mutation_rate'],
```

Code : Tests des paramètres #1

```
            'Crossover Rate': config['crossover_rate'],
            'Number of Generations': config['generations'],
            'Execution Time': execution_time,
            'Total Cost': cost
        })
        print(f"Test {idx + 1}/{len(ga_configurations)}: Execution time = {execution_time}s, Cost = {cost}")

    # Renommer le fichier avec la date et l'heure du dernier test
    current_time = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    final_results_file = os.path.join(desktop_path, f'ga_simulation_results_{current_time}.csv')
    results_df = pd.DataFrame(results)
    results_df.to_csv(final_results_file, index=False, sep=';')
    print(f"All simulations completed and results saved to {final_results_file}")

if __name__ == '__main__':
    main()
```

Code : Tests des paramètres #2

```
import pandas as pd
import statsmodels.api as sm

# Préparer les données
data = pd.DataFrame({
    'm': [2, 3, 6, 3, 3, 3, 5, 3, 3, 6],
    'n': [8, 15, 15, 20, 15, 35, 35, 35, 50, 60],
    'v': [13, 5, 5, 5, 8, 10, 10, 13, 10, 20],
    'Execution Time': [61.64, 187.09, 291.96, 622.09, 265.65, 648.58, 563.65, 598.13, 1621.15, 1923.11]
})

# Régression linéaire
X = data[['m', 'n', 'v']]
y = data['Execution Time']

# Ajouter une constante pour statsmodels
X_with_const = sm.add_constant(X)
model = sm.OLS(y, X_with_const).fit()

intercept = model.params[0]
coefficients = model.params[1:]

print(f"Intercept: {intercept}")
print(f"Coefficients: {coefficients}")

# Résultats du test de la régression linéaire
print(model.summary())

# Calculer X
data['X'] = coefficients[0] * data['m'] + coefficients[1] * data['n'] + coefficients[2] * data['v'] + intercept

# Déterminer les seuils
S1 = data['X'].quantile(0.30)
S2 = data['X'].quantile(0.80)

print(f"Seuil 1: {S1}")
print(f"Seuil 2: {S2}")
```

Code : Méthode régression linéaire suivie de l'interpolation des quantiles sur les résultats de tests sur les paramètres de l'algorithme génétique

```python
import pandas as pd
import numpy as np
import statsmodels.api as sm

# Données pour la méthode approchée
data_approche = pd.DataFrame({
    "m": [2, 5, 2, 2, 3, 3],
    "n": [8, 8, 13, 8, 12, 25],
    "v": [3, 3, 3, 6, 12, 10],
    "CPU-T": [63.14, 75.66, 158.02, 96.39, 204.38, 314.18]
})

# Données pour la méthode exacte
data_exact = pd.DataFrame({
    "m": [2, 5, 2, 2, 3, 3],
    "n": [8, 8, 13, 8, 12, 25],
    "v": [3, 3, 3, 6, 12, 10],
    "CPU-T": [80.1, 28.05, 61.53, 75.72, 563.94, 1521.59]
})

# Régression linéaire pour la méthode approchée avec statsmodels
X_approche = data_approche[["m", "n", "v"]]
y_approche = data_approche["CPU-T"]
X_approche_with_const = sm.add_constant(X_approche)
model_approche = sm.OLS(y_approche, X_approche_with_const).fit()

# Régression linéaire pour la méthode exacte avec statsmodels
X_exact = data_exact[["m", "n", "v"]]
y_exact = data_exact["CPU-T"]
X_exact_with_const = sm.add_constant(X_exact)
model_exact = sm.OLS(y_exact, X_exact_with_const).fit()

# Afficher les résultats de la régression
print("Résultats de la régression pour la méthode approchée:")
print(model_approche.summary())

print("\nRésultats de la régression pour la méthode exacte:")
print(model_exact.summary())

# Coefficients pour la méthode approchée
intercept_approche = model_approche.params[0]
coefficients_approche = model_approche.params[1:]

# Coefficients pour la méthode exacte
intercept_exact = model_exact.params[0]
coefficients_exact = model_exact.params[1:]

print(f"\nApprochée - Intercept: {intercept_approche}, Coefficients: {coefficients_approche}")
print(f"Exacte - Intercept: {intercept_exact}, Coefficients: {coefficients_exact}")

# Coefficients pour la méthode approchée
a_approche, b_approche, c_approche = coefficients_approche
d_approche = intercept_approche

# Coefficients pour la méthode exacte
a_exact, b_exact, c_exact = coefficients_exact
d_exact = intercept_exact

# Calcul des différences
a_diff = a_approche - a_exact
b_diff = b_approche - b_exact
c_diff = c_approche - c_exact
d_diff = d_exact - d_approche

print(f"Differences - a_diff: {a_diff}, b_diff: {b_diff}, c_diff: {c_diff}, d_diff: {d_diff}")
```

Code : Régression linéaire sur les résultats de test méthode exacte et approchée

	A	B	C	D	E	F	G	H	I	J	K
1	Number of Depots	Number of Clients	Number of Vehicles	Population Size	Penalty Value	Elite	Mutation Rate	Crossover Rate	Number of Gen	Execution	Total Cost
2	2	8	13	10	10000000	3	0.77	0.18	500	26.37	322057.02
3	2	8	13	40	10000000	1	0.29	0.6	300	61.64	322174.41
4	2	8	13	20	10000000	1	0.24	0.44	750	78.36	322174.41
5	2	8	13	20	10000000	2	0.16	0.5	750	83.99	322174.41
6	2	8	13	10	10000000	0	0.81	0.38	2000	94.48	322174.41
7	2	8	13	10	10000000	2	0.42	0.56	2000	108.22	322174.41
8	2	8	13	20	10000000	0	0.54	0.46	1000	116.03	322174.41
9	2	8	13	40	10000000	0	0.85	0.76	750	136.81	322174.41
10	2	8	13	30	10000000	0	0.14	0.66	1000	153.97	322174.41
11	2	8	13	30	10000000	3	0.72	0.88	1500	223.27	322174.41
12	2	8	13	50	10000000	1	0.31	0.26	1000	234.9	322174.41
13	2	8	13	10	10000000	3	0.46	0.55	5000	274.36	322174.41
14	2	8	13	50	10000000	0	0.52	0.26	2000	440.52	322174.41
15	2	8	13	20	10000000	2	0.48	0.45	5000	496.16	322174.41
16	2	8	13	20	10000000	1	0.72	0.89	5000	503.82	322174.41
17	2	8	13	20	10000000	0	0.45	0.9	5000	588.79	322174.41
18	2	8	13	40	10000000	1	0.52	0.74	5000	933.6	322174.41
19	2	8	13	40	10000000	0	0.28	0.71	5000	1071.87	322174.41
20	2	8	13	10	10000000	3	0.31	0.69	3000	1202.26	322174.41
21	2	8	13	10	10000000	0	0.31	0.4	100	4.45	324465.84
22	2	8	13	50	10000000	0	0.32	0.24	100	24.08	324465.84
23	2	8	13	50	10000000	0	0.29	0.7	100	25.62	324465.84
24	2	8	13	20	10000000	0	0.71	0.25	300	25.81	324465.84
25	2	8	13	10	10000000	1	0.38	0.26	750	35.51	324465.84
26	2	8	13	10	10000000	1	0.41	0.35	750	35.94	324465.84
27	2	8	13	30	10000000	3	0.31	0.21	300	43.42	324465.84
28	2	8	13	20	10000000	2	0.75	0.84	500	48.24	324465.84
29	2	8	13	20	10000000	3	0.51	0.24	750	63.27	324465.84
30	2	8	13	30	10000000	2	0.44	0.45	500	65.93	324465.84
31	2	8	13	20	10000000	3	0.26	0.46	1000	91.51	324465.84
32	2	8	13	10	10000000	3	0.15	0.13	2000	92.22	324465.84
33	2	8	13	20	10000000	2	0.14	0.15	1000	103.24	324465.84
34	2	8	13	20	10000000	3	0.12	0.14	1000	114.07	324465.84
35	2	8	13	40	10000000	2	0.54	0.26	750	127.61	324465.84
36	2	8	13	40	10000000	1	0.42	0.33	750	134.6	324465.84
37	2	8	13	20	10000000	1	0.4	0.73	1500	139.39	324465.84
38	2	8	13	20	10000000	1	0.36	0.85	1500	139.58	324465.84
39	2	8	13	10	10000000	2	0.28	0.33	3000	152.66	324465.84
40	2	8	13	20	10000000	3	0.13	0.28	1500	154.07	324465.84
41	2	8	13	20	10000000	3	0.14	0.17	1500	154.26	324465.84
42	2	8	13	50	10000000	1	0.45	0.13	750	160.84	324465.84
43	2	8	13	20	10000000	3	0.64	0.16	2000	166.27	324465.84
44	2	8	13	50	10000000	3	0.12	0.36	750	167.59	324465.84
45	2	8	13	20	10000000	1	0.66	0.82	2000	185.13	324465.84

Figure : Exemple sur les résultats de test sur les paramètres de l'algorithme génétique pour la configuration m = 2, n = 8, v = 13

F	G	H	I	J	K
	Type	V	W	Vcost	
1	Fourgon	8	2500	45	
2	Camion 10T	27	10000	70	
3	IVECO	8	3000	45	
4	MASTER	8	2000	45	
5	Camion 2.5T	10	2500	45	
6	Camion 4T	14	4000	50	
7	Camion 5T	16	5000	55	
8	Camion 6T	18	6000	55	
9	Camion 7T	21	7000	60	
10	Camion 3.5T	12	3500	50	
11	Hilux	2.5	1200	45	
12	Herbin	2	1200	45	
13	Camion 20T	60	20000	80	

Figure : Exemples sur les données de véhicules

	A	B	C	D	E
1	Code_Entrepot	Nom	Colonne1	Latitude	Longitude
2	E01	MAGASIN ALGER	E01	36.6975365	3.2702203
3	E02	MAGASIN ORAN	E02	35.6117590	-0.5748800
4	E03	MAGASIN BISKRA	E03	34.8643913	5.7191372
5					
6					
7					
8					
9					
10					
11					

Figure : Exemple sur les données d'entrepôts

	A	B	C	D	E
1	Code_Client	Nom	Latitude	Longitude	
2	C01	Client1	35.223355	4.1975	
3	C02	Client2	34.576547	5.344529	
4	C03	Client3	33.365009	6.874524	
5	C04	Client4	34.720239	5.283059	
6	C05	Client5	34.602212	5.380627	
7	C06	Client6	34.732349	5.304842	
8	C07	Client7	35.532011	-0.19294	
9	C08	Client8	34.92038	-0.214683	
10	C09	Client9	35.179509	-0.640222	
11	C10	Client10	35.052869	-1.42792	
12	C11	Client11	34.647223	-1.321924	
13	C12	Client12	33.594637	7.024078	
14	C13	Client13	35.283295	-1.339778	
15	C14	Client14	35.065981	-1.425472	
16	C15	Client15	34.846904	-1.722071	
17	C16	Client16	36.743513	2.879334	
18	C17	Client17	36.548407	1.703843	
19	C18	Client18	36.468606	2.618177	
20	C19	Client19	36.540397	2.915203	
21	C20	Client20	36.467503	2.68458	
22	C21	Client21	36.25598	1.954567	
23	C22	Client22	34.744483	5.910228	
24	C23	Client23	34.718721	6.291697	
25	C24	Client24	36.271767	2.710635	
26	C25	Client25	35.4082	1.971738	
27	C26	Client26	35.218476	2.316581	
28	C27	Client27	35.432873	2.214718	
29	C28	Client28	35.448562	2.894055	
30	C29	Client29	35.455215	2.9073	
31					

Figure : Exemple sur les données de clients

Bibliographie

[1] Revel, A. (1997). Sommet mondial de l'alimentation. Rome-11-18 novembre 1996. Économie rurale, 238(1), 44-46.

[2] Joya, R., & Hamon, J. cio.

[3] Benoit, S. (2024). Ancrage industriel en bioéconomie: quel rôle joue la gouvernance logistique?. Les Cahiers Scientifiques du Transport-Scientific Papers in Transportation, 78.

[4] Lulovicova, A. (2024). Évaluation environnementale des systèmes alimentaires territoriaux: Apports de l'analyse du cycle de vie territoriale à la construction et à l'évaluation des processus de reterritorialisation durables des systèmes agroalimentaires en France (Doctoral dissertation, Université Côte d'Azur).

[5] G. B. Dantzig and J. H. Ramser. (1959). The truck dispatching problem. Management Science, 6(1):80–91, 10.

[6] Soleilhac, G. (2022). Optimisation de la distribution de marchandises avec sous-traitance du transport : une problématique chargeur.

[7] Boudjerda, N., Djarit, I., & Alliouache, A. E. (2022). Le problème de routage dynamique des véhicules avec fenêtres de temps.

[8] Cokyasar, T., Subramanyam, A., Larson, J., Stinson, M., & Sahin, O. (2023). Time-constrained capacitated vehicle routing problem in urban e-commerce delivery. Transportation Research Record, 2677(2), 190-203.

[9] Martinez, C. (2020). Considération de la dimension humaine dans l'optimisation de tournées de soins et services à domicile soumises à des perturbations.

[10] Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). "A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations." *Journal für Betriebswirtschaft*, 58(2), 81-117.

[11] Baldacci, R., Bartolini, E., & Mingozzi, A. (2011). "An exact method for the vehicle routing problem with a heterogeneous fleet." Mathematical Programming, 131(1-2), 241-267.

[12] Bräysy, O., & Gendreau, M. (2005). "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." Transportation Science, 39(1), 104-118.

[13] Laporte, G., & Nobert, Y. (1987). "Exact algorithms for the vehicle routing problem." Annals of Discrete Mathematics, 31, 147-184.

[14] Gérard, O. (2022). LORH : outil pour la planification du parcours patient dans le milieu hospitalier.

[15] Classe, N. P. (). Classe NP. fr-academic.com.

[16] Robbes, A. (2021). Résolution de problèmes intégrés dans un contexte pharmaceutique : ordonnancement et tournée de véhicules.

[17] Daussogne, Q. & Catanzaro, D. (). A Metaheuristic for the Job Sequencing and Tool Switching Problem. dial.uclouvain.be

[18] Sakhri, M. S. A., Ben-Hassine, M., & Tlili, M. (2024, March). Optimisation des tournées de véhicules avec clients stochastiques par une approche d'apprentissage automatique. In La 25ème édition du congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision ROADEF 2024.

[19] ABDARRAZAKE, H. & SEDDIK, L. (2021). Une Approche pour la Résolution et l'Optimisation d'un Problème de Tournées de Véhicule (Capacitated Location Routing Problem (CLRP)).

[20] Ali, A., Mensour, H., & Abdelkamel, B. E. N. (2020). Adaptation d'un nouvel algorithme métaheuristique appelé TLBO pour la résolution du problème de tournées de véhicules. dspace.univ-ouargla.dz.

[21] Salhi, S., Imran, A., & Wassan, N. A. (2013). The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. Computers & Operations Research. https://doi.org/10.1016/j.cor.2013.05.011

[22] IBM Corporation. (2009). Optimization modeling with IBM ILOG OPL: Instructor workbook. IBM Corporation.

[23] Toussaint, H. (2016). IBM ILOG CPLEX Optimization Studio: Une introduction à OPL et Cplex Studio IDE. IBM. Dernière mise à jour : février 2019.

[24] IBM Corporation. (2009). *IBM ILOG OPL IDE Reference* (Version 6.3). IBM Corporation.

[25] ILOG, Inc. (2008). AMPL CPLEX System Version 11.0 User's Guide: Standard (Command-line) Version Including CPLEX Directives. ILOG, Inc.

[26] Bermudez, C., Graglia, P., Stark, N., Salto, C., & Alfonso, H. (2010). Comparison of recombination operators in panmictic and cellular GAs to solve a vehicle routing problem. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 14(46), 34-44.

[27] Sajid, M., Singh, J., Haidri, R. A., Prasad, M., Varadarajan, V., Kotecha, K., & Garg, D. (2021). A novel algorithm for capacitated vehicle routing problem for smart cities. Symmetry, 13(10), 1923.

[28] Pichpibul, T., & Kawtummachai, R. (2013). A heuristic approach based on clarke-wright algorithm for open vehicle routing problem. The scientific world journal, 2013.

[29] Project OSRM. (s.d.). Introduction to OSRM API. OSRM Documentation. project-osrm.org

[30] Valdecy. (n.d.). *pyVRP* [Software]. GitHub. https://github.com/Valdecy/pyVRP/blob/master/src/pyVRP.py

[31] Meliani, Y., Hani, Y., Elhaq, S. L., & El Mhamedi, A. (2019). A developed Tabu Search algorithm for heterogeneous fleet vehicle routing problem. IFAC-PapersOnLine, 52(13), 1051-1056.

[32] Housroum, H. (2005). Une approche génétique pour la résolution du problème VRPTW dynamique.

[33] Guenoune, H., & Dahane, M. (2019). Résolution du problème de tournées de véhicules avec collecte et livraison simultanées avec une approche coopérative de métaheuristiques.

Résumé :

Résumé

Ce mémoire de fin d'études présente une approche pour résoudre le problème de tournées de véhicules avec multi-dépôts et flotte hétérogène (MDHFVRP) pour l'entreprise CPH Agriculture. Deux méthodes distinctes sont explorées : une méthode exacte utilisant le solveur IBM ILOG CPLEX, et une méta-heuristique génétique pour une approche approchée. Une interface graphique permettant aux logisticiens d'utiliser la méta-heuristique génétique a également été développée. L'objectif est d'optimiser les tournées de livraison tout en minimisant les coûts et en respectant les contraintes opérationnelles.

Mots clés : problème de tournées de véhicules, multi-dépôts, flotte hétérogène, méthode exacte, cplex, méta-heuristique algorithme génétique, interface graphique, optimisation, logistique, transport

Abstract

This final year dissertation presents an approach to solve the Multi-Depot Vehicle Routing Problem with Heterogeneous Fleet (MDHFVRP) for the company CPH Agriculture. Two distinct methods are explored : an exact method using the IBM ILOG CPLEX, and a genetic metaheuristic for an approximate approach. A graphical user interface allowing logisticians to use the genetic metaheuristic has also been developed. The objective is to optimize delivery routes while minimizing costs and respecting operational constraints.

Keywords: vehicle routing problem, multi-depot, heterogeneous fleet, exact method, cplex, metaheuristic, genetic algorithm, graphical user interface, optimization, logistics, transportation

ملخّص

يقدم هذا البحث الختامي نهجًا لحل مشكلة جولات المركبات متعددة المستودعات وأساطيل غير متجانسة (MDHFVRP) لشركة CPH Agriculture, تم استكشاف طريقتين مختلفتين : طريقة دقيقة باستخدام حال IBM ILOG CPLEX، وطريقة تقريبية باستخدام الخوارزمية الجينية المستنبطة. تم أيضًا تطوير واجهة بيانية لتمكين أخصائي اللوجستيات من استخدام الخوارزمية الجينية المستنبطة. الهدف هو تحسين جولات التسليم مع تقليل التكاليف واحترام القيود التشغيلية.

كلمات مفتاحية: مشكلة جولات المركبات، متعدد المستودعات، أسطول غير متجانس، طريقة دقيقة، cplex، خوارزمية جينية مستنبطة، واجهة رسومية، تحسين، لوجستيات، النقل.