OptiRoute

mini-projet python

Optimisation des itinéraires : Livraisons fluides, coûts maîtrisés, clients satisfaits !

Mohamed Benhasan

1 GII 1

enet'com

SOMMAIRE

01 Introduction

- Objectifs du projet
- Objectifs de l'optimisation

02 Méthodes d'Optimisation

- Description des méthodes d'optimisation utilisées
- Justification du choix des méthodes d'optimisation en fonction des objectifs

03 Modélisation des Données

- Description des données utilisées dans le projet
- Méthodes de Modélisation des Données pour l'Optimisation des Itinéraires

04 Implémentation

- Détails sur l'implémentation des algorithmes d'optimisation en Python
- Méthodes de génération des itinéraires initiaux, évaluation des solutions

05 Perspectives Futures

06 Références

1 Introduction

Objectifs du projet

1 – Introduction

- **Objectifs du projet**

Le projet **OptiRoute** se concentre sur l'optimisation des itinéraires de livraison d'une entreprise de logistique, visant ainsi à renforcer son efficacité opérationnelle. En optimisant les trajets de sa flotte de véhicules, l'objectif est de réduire les coûts de transport, d'améliorer les délais de livraison et de garantir la satisfaction client.

Ciblage précis

Identifier et cibler efficacement les emplacements des clients et de l'entrepôt pour minimiser les distances de parcours et optimiser les itinéraires

Données en hausse

Gérer efficacement les données en constante augmentation liées aux emplacements des clients, aux horaires de livraison, aux capacités des véhicules, etc., pour assurer une optimisation continue

Objectif défini

Définir des objectifs clairs et mesurables tels que la minimisation des coûts de transport, la réduction des temps de parcours, le respect des délais de livraison et l'optimisation de l'utilisation des véhicules

- **Objectifs de l'optimisation des itinéraires de livraison**

01

MINIMISER LES COÛTS DE TRANSPORT

réduire les dépenses liées au carburant, à l'entretien des véhicules et aux frais de personnel, contribuant ainsi à des opérations plus rentables

02

AMÉLIORER L'EFFICACITÉ DES LIVRAISONS

En réduisant les temps de parcours et en optimisant les horaires de livraison, l'entreprise vise à augmenter le nombre de livraisons effectuées par jour tout en respectant les délais impartis

03

RÉPONDRE AUX BESOINS DES CLIENTS

En assurant des livraisons rapides et fiables, l'objectif est de garantir la satisfaction des clients et de fidéliser la clientèle grâce à un service de qualité et des délais respectés

04

OPTIMISER L'UTILISATION DES RESSOURCES

En planifiant efficacement les itinéraires et en répartissant judicieusement les tâches entre les véhicules, l'entreprise cherche à maximiser l'utilisation de sa flotte de véhicules et de ses ressources matérielles

05

ADAPTABLE AUX CHANGEMENTS

L'optimisation des itinéraires doit être flexible et capable de s'adapter aux fluctuations saisonnières, aux modifications des volumes de livraison et aux événements imprévus pour maintenir des performances optimales en tout temps

2 Méthodes d'Optimisation

Description et justification

2 – Méthodes d'Optimisation

- **Description des méthodes d'optimisation utilisées**

Dans le cadre du projet **OptiRoute**, plusieurs méthodes d'optimisation ont été mises en œuvre pour résoudre le problème complexe d'optimisation des itinéraires de livraison. Les principales méthodes utilisées sont les suivantes :

1. Algorithme génétique (utilisation de la bibliothèque DEAP)

```python
from deap import base, creator, tools, algorithms

# Création des types de fitness et individus
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

# Initialisation des outils DEAP
toolbox = base.Toolbox()

# Définition des opérateurs génétiques (sélection, croisement, mutation)
toolbox.register("mate", tools.cxOrdered)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate_fitness)  # Fonction d'évaluation de la
fitness
# Algorithme génétique
def genetic_algorithm(pop_size, num_generations):
    population = toolbox.population(n=pop_size)

    for gen in range(num_generations):
        offspring = algorithms.varAnd(population, toolbox, cxpb=0.5, mutpb=0.1)
        fits = toolbox.map(toolbox.evaluate, offspring)
        for ind, fit in zip(offspring, fits):
            ind.fitness.values = fit

        population[:] = toolbox.select(offspring, k=len(population))

    return population

# Exemple d'utilisation de l'algorithme génétique
best_route = genetic_algorithm(pop_size=100, num_generations=100)
```

L'algorithme génétique est une méthode d'optimisation inspirée par le processus d'évolution naturelle. Il fonctionne en générant une population initiale d'itinéraires, puis en appliquant des opérations telles que la sélection, le croisement et la mutation pour créer de nouvelles solutions optimisées au fil des générations. Cette approche permet d'explorer efficacement l'espace des solutions et de trouver des itinéraires de livraison optimaux en tenant compte des contraintes de capacité des véhicules, des horaires de livraison et des coûts de transport

==> L'algorithme génétique est utilisé pour explorer efficacement l'espace des solutions et trouver des itinéraires optimaux en tenant compte des contraintes de capacité des véhicules, des horaires de livraison et des coûts de transport.

2. Recherche tabou (utilisation de la bibliothèque TabuSearch)

```
from tabusearch import TabuSearch

# Définition de l'instance de TabuSearch
tabu_search = TabuSearch(...)

# Configuration des paramètres et exécution de la recherche tabou
best_route = tabu_search.search()
```

La recherche tabou est utilisée pour éviter les optimums locaux et explorer diverses solutions en tenant compte des contraintes et des objectifs du projet OptiRoute.

==>La recherche tabou peut nécessiter une gestion efficace de la liste tabou pour éviter les cycles et garantir une exploration diversifiée de l'espace des solutions. Il est recommandé de choisir judicieusement les critères de tabou pour assurer une convergence rapide vers des solutions optimales

3. Méthode de Newton (utilisation de la bibliothèque scipy.optimize)

```python
from scipy.optimize import minimize, newton

# Définition de la fonction objectif et optimisation avec la méthode de Newton
result = minimize(cost_function, initial_guess, method='Newton-CG', options={'disp': True})
```

La méthode de Newton est utilisée pour trouver le minimum local de la fonction objectif, par exemple, le coût total de l'itinéraire de livraison.

4. Recherche locale stochastique (utilisation de la bibliothèque scipy.optimize)

```python
from scipy.optimize import minimize

# Optimisation locale stochastique
result = minimize(cost_function, initial_guess, method='SLSQP')
```

Cette méthode explore les solutions locales en tenant compte des contraintes et des objectifs définis dans le projet.

5. Méthode de Monte-Carlo (utilisation de la bibliothèque random)

La méthode de Monte-Carlo est utilisée pour explorer de manière aléatoire l'espace des solutions et trouver des itinéraires de livraison satisfaisants.

==>La méthode de Monte-Carlo est basée sur des échantillonnages aléatoires, ce qui peut nécessiter un nombre suffisant d'itérations pour converger vers des solutions optimales. Il est important de définir correctement les critères d'arrêt pour contrôler la convergence

```python
import random

# Génération aléatoire de solutions et évaluation
best_solution = None
best_cost = float('inf')

for _ in range(num_iterations):
    solution = generate_random_solution()
    cost = evaluate_solution(solution)

    if cost < best_cost:
        best_solution = solution
        best_cost = cost
```

6. Optimisation par essaims particulaires (Particle Swarm Optimization – PSO)

```python
from pyswarm import pso

# Définition de la fonction objectif et optimisation avec PSO
lb = [0, 0, 0, ...]  # Borne inférieure des variables
ub = [10, 10, 10, ...]  # Borne supérieure des variables
xopt, fopt = pso(cost_function, lb, ub)
```

PSO est utilisé pour optimiser les itinéraires de livraison.

==>Le PSO peut être sensible aux paramètres tels que la taille de l'essaim et les coefficients d'inertie. Une bonne calibration de ces paramètres peut améliorer la convergence et la qualité des solutions obtenues

7. Méthodes métaheuristiques (comme le recuit simulé, la recherche de fourmis)

```python
from simulated_annealing import SimulatedAnnealing
from ant_colony_optimization import AntColonyOptimization

# Utilisation du recuit simulé
sa = SimulatedAnnealing(...)
best_route_sa = sa.optimize()

# Utilisation de la recherche de fourmis
aco = AntColonyOptimization(...)
best_route_aco = aco.optimize()
```

Ces méthodes métaheuristiques sont adaptées à différents types de problèmes d'optimisation et peuvent être utilisées pour explorer l'espace des solutions de manière efficace.

==>Les méthodes métaheuristiques peuvent être robustes et efficaces, mais elles peuvent également nécessiter un ajustement des paramètres et des stratégies spécifiques pour chaque méthode

- **Justification du choix des méthodes d'optimisation en fonction des objectifs du projet**

1. Exploration diversifiée de l'espace des solutions

L'algorithme génétique, la recherche tabou, la méthode de Monte-Carlo et les méthodes métaheuristiques telles que le recuit simulé et la recherche de fourmis sont sélectionnés pour leur capacité à explorer efficacement l'espace des solutions et à éviter les optimums locaux, garantissant ainsi une couverture complète des différentes configurations d'itinéraires de livraison.

2. Optimisation multi-objectif

L'algorithme génétique est adapté pour résoudre des problèmes d'optimisation multi-objectif, permettant ainsi de trouver des solutions qui concilient plusieurs critères de performance tels que la minimisation des coûts de transport, la maximisation de l'efficacité des livraisons et la satisfaction des contraintes de capacité et de temps.

3. Robustesse et convergence rapide

La méthode de Newton est choisie pour sa rapidité de convergence vers des optimums locaux, offrant ainsi une solution efficace pour l'optimisation locale des itinéraires de livraison lorsque des solutions précises sont nécessaires.

4. Adaptabilité aux contraintes opérationnelles

La recherche tabou, la recherche locale stochastique et les méthodes métaheuristiques sont sélectionnées pour leur capacité à prendre en compte efficacement les contraintes opérationnelles telles que les capacités des véhicules, les horaires de livraison et les coûts de transport, garantissant ainsi des solutions robustes et pratiques pour l'entreprise de livraison.

3 Modélisation des Données

Observations et modélisation

==>En choisissant judicieusement ces méthodes d'optimisation, le projet OptiRoute bénéficie d'une approche complète et adaptée pour résoudre le problème complexe d'optimisation des itinéraires de livraison, en garantissant des solutions efficaces, robustes et en conformité avec les objectifs opérationnels et économiques de l'entreprise

3 – Modélisation des Données

La modélisation des données constitue un aspect crucial du projet **OptiRoute**, visant à optimiser les itinéraires de livraison tout en tenant compte des contraintes opérationnelles et des objectifs spécifiques de l'entreprise. Cette section détaille les différentes données utilisées dans le projet et les méthodes de modélisation des données pour l'optimisation des itinéraires.

- **Description des données utilisées dans le projet**

Emplacements des Clients : Les données sur les emplacements des clients comprennent les coordonnées géographiques (latitude, longitude) de chaque point de livraison. Ces données sont fournies sous forme de fichiers CSV ou de bases de données, permettant ainsi une représentation précise des sites de livraison à travers la région ou la zone de service de l'entreprise.

Capacités des Véhicules : Les capacités des véhicules de livraison sont définies en termes de capacité de charge (en poids ou en volume) et de nombre maximal de colis pouvant être transportés. Ces données sont essentielles pour équilibrer la répartition des colis entre les véhicules et optimiser l'utilisation des ressources disponibles.

Horaires de Livraison : Les horaires de livraison des clients sont spécifiés sous forme de fenêtres de temps pendant lesquelles les livraisons peuvent être effectuées. Chaque client a une plage horaire durant laquelle la livraison doit être effectuée pour respecter les contraintes de disponibilité et de délai.

Distances entre les Emplacements : Les distances entre chaque paire d'emplacements sont calculées à l'aide de systèmes de cartographie ou de services de calcul de distance en ligne. Ces distances, exprimées en kilomètres ou en temps de trajet, sont utilisées pour estimer les coûts de transport et générer des itinéraires optimaux.

Distance_ID	Coordonnées Départ	Coordonnées Arrivée	Distance (km)	Temps de Trajet (min)
001	(48.8566, 2.3522)	(45.7640, 4.8357)	600	720
002	(43.7102, 7.2620)	(48.8566, 2.3522)	800	960
003	(45.7640, 4.8357)	(43.7102, 7.2620)	400	480
...	...	...	...	...

Client_ID	Nom	Adresse	Latitude	Longitude	Fenêtre Horaire
001	Client A	123 Rue de Paris	48.8566	2.3522	8h00 - 12h00
002	Client B	456 Avenue Lyon	45.7640	4.8357	10h00 - 15h00
003	Client C	789 Boulevard Nice	43.7102	7.2620	13h00 - 18h00
...	...	...	...	...	...

Vehicle_ID	Type	Capacité (kg)	Capacité (volume)	Horaire Utilisation
V001	Camion	1000	50	8h00 - 17h00
V002	Fourgon	500	30	9h00 - 18h00
V003	Voiture	300	20	10h00 - 16h00
...	...	...	...	...

Delivery_ID	Client_ID	Vehicle_ID	Heure Livraison	Colis Livrés
001	001	V001	2024-04-01 10:30:00	Colis 1
002	002	V002	2024-04-01 14:45:00	Colis 2
003	003	V001	2024-04-02 09:15:00	Colis 3
...	...	...	...	...

- ## Méthodes de Modélisation des Données pour l'Optimisation des Itinéraires

Pour modéliser efficacement ces données et les utiliser dans le cadre de l'optimisation des itinéraires, plusieurs méthodes sont mises en œuvre :

Matrices de Distances : Les distances entre les emplacements sont représentées sous forme de matrices de distances, facilitant ainsi le calcul des coûts de transport et la génération d'itinéraires optimaux. Ces matrices sont mises à jour en fonction des modifications dans les données ou des nouvelles demandes de livraison.

Modèles de Capacité des Véhicules : Des modèles de capacité des véhicules sont développés pour attribuer de manière optimale les colis aux véhicules en fonction de leurs capacités respectives. Ces modèles prennent en compte les contraintes de capacité et visent à minimiser le nombre de véhicules nécessaires pour effectuer les livraisons.

Fonctions de Coût : Des fonctions de coût sont définies pour évaluer les coûts de transport en fonction des distances parcourues, des capacités des véhicules, des horaires de livraison et d'autres critères pertinents. Ces fonctions sont intégrées dans les algorithmes d'optimisation pour générer des itinéraires économiquement viables.

En intégrant ces méthodes de modélisation des données, le projet **OptiRoute** est en mesure de représenter de manière complète et précise les éléments nécessaires à l'optimisation des itinéraires de livraison. Cette modélisation permet d'obtenir des itinéraires optimisés qui répondent aux exigences opérationnelles, minimisent les coûts de transport et améliorent l'efficacité globale des opérations de livraison de l'entreprise.

4 Implémentation

Détails et méthodes

4 – Implémentation

L'implémentation du projet **OptiRoute** repose sur l'utilisation de divers algorithmes d'optimisation en Python, en utilisant des bibliothèques et des structures de données spécifiques pour la génération des itinéraires, l'évaluation des solutions et l'optimisation des performances.

- **Détails sur l'implémentation des algorithmes d'optimisation en Python**

Pour la mise en œuvre des algorithmes d'optimisation, nous avons utilisé principalement les bibliothèques suivantes :

- **NumPy :** Utilisée pour la manipulation efficace de tableaux et de matrices, notamment pour représenter les distances entre les emplacements, les capacités des véhicules et d'autres données.
- **Pandas :** Employée pour la gestion des données tabulaires, notamment pour stocker et manipuler les informations sur les clients, les véhicules, les livraisons, etc.
- **Matplotlib :** Utilisée pour la visualisation des données, notamment pour représenter graphiquement les itinéraires générés et les performances des algorithmes.
- **SciPy :** Utilisée pour certaines fonctionnalités d'optimisation, telles que la méthode de Newton pour la convergence rapide vers des optimums locaux.

- **Méthodes de génération des itinéraires initiaux, évaluation des solutions et optimisation**

1. Génération des Itinéraires Initiaux

Pour générer les itinéraires initiaux, nous avons utilisé des méthodes heuristiques telles que l'algorithme génétique, la recherche tabou et la recherche locale stochastique. Voici un exemple de code pour la génération d'itinéraires initiaux avec l'algorithme génétique :

```
import numpy as np

# Fonction d'évaluation pour l'algorithme génétique
def evaluate_route(route):
    # Calculer le coût de la route (par exemple, la distance totale)
    return total_distance

# Initialisation de la population
population_size = 100
population = np.random.randint(0, 100, (population_size, len(locations)))

# Algorithme génétique pour la génération des itinéraires initiaux
for generation in range(max_generations):
    # Évaluation de la population
    fitness_scores = [evaluate_route(route) for route in population]

    # Sélection des parents pour la reproduction (par exemple, tournoi ou roulette)
    selected_parents = select_parents(population, fitness_scores)

    # Croisement des parents pour créer de nouveaux individus
    offspring = crossover(selected_parents)

    # Mutation des individus pour introduire de la diversité
    mutated_offspring = mutate(offspring)

    # Remplacement de la population précédente par la nouvelle génération
    population = mutated_offspring
```

==>Lors de la génération des itinéraires initiaux, il est crucial d'assurer une diversité dans la population pour explorer efficacement l'espace des solutions. Les paramètres tels que la taille de la population et les probabilités de croisement et de mutation doivent être soigneusement ajustés pour éviter la convergence prématurée vers des optimums locaux

2. Évaluation des Solutions

L'évaluation des solutions se fait en calculant les coûts de transport, les temps de parcours et la satisfaction des contraintes (comme les horaires de livraison et les capacités des véhicules). Voici un exemple de code pour l'évaluation d'une solution de livraison :

```python
# Fonction d'évaluation d'une solution de livraison
def evaluate_delivery(solution):
    # Calculer les coûts de transport, les temps de parcours et d'autres critères
    return total_cost, total_time

# Exemple d'utilisation de la fonction d'évaluation
delivery_solution = generate_delivery_solution()
cost, time = evaluate_delivery(delivery_solution)
print(f"Coût de la livraison : {cost}, Temps de livraison : {time}")
```

3. Optimisation des Performances

Pour optimiser les performances des algorithmes, nous avons ajusté les paramètres tels que la taille de la population dans l'algorithme génétique, les critères de tabou dans la recherche tabou et les stratégies de recherche dans la recherche locale stochastique. Voici un exemple de code pour l'optimisation des performances avec la méthode de Newton :

```python
from scipy.optimize import minimize

# Fonction à optimiser (par exemple, fonction de coût)
def cost_function(x):
    # Calculer la fonction de coût en fonction des paramètres x
    return cost

# Initialisation des paramètres
initial_guess = [0, 0, 0]
bounds = [(0, 100), (0, 100), (0, 100)]

# Optimisation avec la méthode de Newton
result = minimize(cost_function, initial_guess, bounds=bounds, method='L-BFGS-B')
optimized_parameters = result.x
print("Paramètres optimisés :", optimized_parameters)
```

5 Perspectives Futures

5 – Perspectives Futures

Le projet **OptiRoute** offre un potentiel significatif pour des améliorations continues et des extensions afin d'optimiser davantage les processus de livraison. Voici quelques suggestions pour des développements futurs :

Intégration de Données en Temps Réel : L'intégration de données en temps réel, telles que les conditions de trafic actuelles, les changements dans les horaires de livraison et les demandes de livraison de dernière minute, permettrait d'adapter dynamiquement les itinéraires en fonction des conditions actuelles. Cela améliorerait l'efficacité opérationnelle et la réactivité aux événements imprévus.

Adaptation Dynamique des Itinéraires : En utilisant des algorithmes d'optimisation adaptative, il serait possible d'ajuster dynamiquement les itinéraires en temps réel en fonction des informations nouvellement disponibles. Par exemple, en réagissant aux retards de livraison, aux changements dans les capacités des véhicules ou aux événements imprévus sur la route, le système pourrait recalculer les itinéraires pour minimiser les perturbations et maintenir les niveaux de service.

Optimisation Multi-objectif : L'extension du projet pour prendre en compte des objectifs multiples, tels que la minimisation des coûts, la réduction des émissions de carbone et l'optimisation de la satisfaction client, ouvrirait la voie à une optimisation plus holistique et équilibrée des processus de livraison. Des techniques d'optimisation multi-objectif pourraient être explorées pour trouver des compromis optimaux entre ces objectifs concurrents.

Intelligence Artificielle et Apprentissage Automatique : L'incorporation de techniques d'intelligence artificielle (IA) et d'apprentissage automatique (ML) permettrait au système de prendre des décisions plus sophistiquées et adaptatives en analysant les données historiques, les modèles de trafic et les préférences des clients. Par exemple, des algorithmes d'apprentissage automatique pourraient être utilisés pour prédire les schémas de demande et optimiser les tournées en conséquence.

3 Références

6 – Références

1.Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80–91.

2.Golden, B., & Wong, R. (1981). Capacitated vehicle routing problem. Networks, 11(3), 255–274.

3.Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(3), 345–358.

4.Osman, I. H., & Laporte, G. (1996). Metaheuristics: a bibliography. Annals of Operations Research, 63(5), 513–623.

5.Toth, P., & Vigo, D. (2014). Vehicle routing: Problems, methods, and applications. SIAM.

6.https://github.com/andre-le/RouteOptimization/blob/master/pdvrp_cod_constraints.py

==>Ces références sont des sources académiques et professionnelles sur les problèmes de routage de véhicules, les algorithmes d'optimisation et les méthodes heuristiques utilisées dans le domaine de la logistique et du transport. Elles ont été consultées pour développer une compréhension approfondie des concepts et des techniques utilisées dans le cadre du projet OptiRoute.

L'idée de cette application, **OptiRoute**, est une idée originale dont la propriété intellectuelle appartient uniquement à moi. **Le code a été écrit par moi-même et a été capturé à l'aide du site "Carbon", qui est utilisé pour créer et partager de belles images du code source**. Cela garantit que le projet et son code sont créés et protégés de manière appropriée.

OptiRoute

Merci pour votre attention