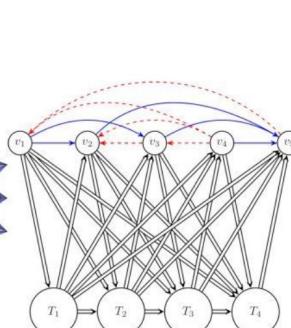
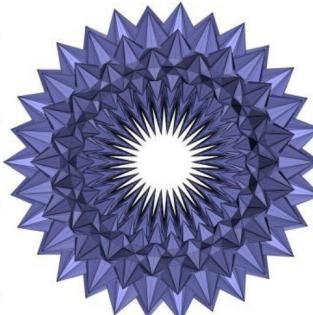
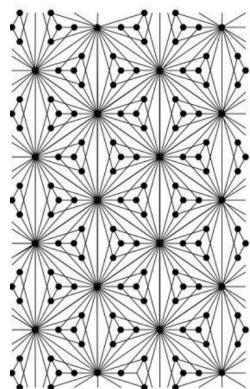


SUP GALILÉE
UNIVERSITÉ SORBONNE PARIS NORD

Mini-projet d'Optimisation Combinatoire

Auteures :
Catherine Sidar CALI
Katell GOUZERH

Professeur encadrant :
Mr FOUILHOUX



2024/2025

Introduction

Le placement optimal de centre logistiques représente un défi stratégique pour les entreprises qui souhaitent optimiser leur chaîne d'approvisionnement et ainsi réduire leurs coûts opérationnel.

Ce problème d'optimisation combinatoire consiste à déterminer le placement idéal de centres logistiques d'une nouvelle enseigne s'installant dans un pays.

Ce mini-projet que nous effectuons en binôme s'inscrit dans l'application des concepts fondamentaux que nous avons vu en cours d'optimisation linéaire et combinatoire, nous permettant ainsi de lier théorique et pratique.

Dans le cadre de ce projet, nous aborderons ce problème de placement de centre en considérant deux approches de modélisations différentes issue de la littérature scientifique en Recherche Opérationnelle.

D'une part nous allons modéliser les problèmes en utilisant la programmation linéaire en nombres entiers (PLNE), d'autre part nous allons développer des méthodes de recherches gloutonnes et méta-heuristiques pour résoudre des instances de plus grande taille.

Pour finir, nous allons effectuer une analyse comparative des performances des différentes méthodes de résolution en terme de qualité et temps de calcul, puis comparer les solutions aux deux problèmes et ainsi conclure sur l'approche la plus optimale pour le placement de centres logistiques.

Sommaire

Introduction	i
1 Choix, modélisation et résolution de deux problèmes de placement	1
1.1 Choix de deux problèmes à traiter	1
1.2 Modélisation PLNE	3
1.3 Méthodes gloutonnes et méta-heuristiques	11
2 Comparaison expérimentale des méthodes de résolution	15
3 Comparaisons des solutions des deux problèmes P1 et P2	21
4 Conclusion	23

Choix, modélisation et résolution de deux problèmes de placement

1.1 Choix de deux problèmes à traiter

Question 1 Choisissez deux problèmes $P1$ et $P2$ répondant à la démarche de placement. Vous devez les définir complètement. Ces deux problèmes n'ont pas l'obligation d'être éloignés l'un de l'autre, mais ils doivent permettre d'avoir deux visions du sujet du projet (par exemple UFLP/CFLP, p -médian/ p -centre, UFLP/ p -médian, ou inventer vos propres variantes).

Les deux modélisations que nous avons choisies sont :

- *Uncapacitated Facility Location Problem (UFLP)* : ce problème n'ajoute aucune contrainte supplémentaire à l'énoncé de base, mais nous permet de minimiser la somme des coûts d'installation et de gestion :

$$\sum_{i \in S} f_i + \sum_{j \in I \setminus S} c_{\sigma(j)j}$$

où :

- S est l'ensemble des centres ouverts,
- I est l'ensemble des villes,
- f_i est le coût d'installation du centre i ,
- $c_{\sigma(j)j}$ est le coût de desserte de la ville j par le centre auquel elle est affectée, noté $\sigma(j)$.
- *Capacitated Facility Location Problem (CFLP)* : ce problème reprend le modèle de l'UFLP en ajoutant une contrainte de capacité : chaque centre peut desservir au plus q villes.

Nous avons choisi ces deux problèmes, premièrement car UFLP est le modèle de base, on essaye de minimiser la somme des coûts d'installation et de gestion sans ajouter de contraintes. C'est donc un bon point de départ pour aborder le problème, comprendre les points fondamentaux et valider les méthodes de résolutions sur des cas simples.

Dans un second temps, CFLP (Capacitated Facility Location Problem) permet d'être en situation réelle. Il reprend le UFLP et ajoute la contrainte qu'un centre ne peut desservir qu'un nombre maximal q de villes, ce qui correspond bien à des limitations opérationnelles courantes de la vie quotidienne.

Question 2 Préciser et justifier, pour chaque problème, comment vous calculez les coûts c_{ij} et f_i en fonction des données des instances d_{ij} et \bar{f}_i .

Modélisation des problèmes UFLP et CFLP

Variables

- $u_i = \begin{cases} 1 & \text{si la ville } i \text{ a un centre} \\ 0 & \text{sinon} \end{cases}$
- $v_{ij} = \begin{cases} 1 & \text{si le centre de la ville } i \text{ dessert la ville } j \\ 0 & \text{sinon} \end{cases}$

Fonction objective

$$\min \sum_{i=1}^m \left(f_i \cdot u_i + \sum_{j=1}^n c_{ij} \cdot v_{ij} \right)$$

Formule de distance

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Définition des coûts

- *Coût c_{ij}* :

Le coût c_{ij} est défini proportionnellement à la distance d_{ij} entre la ville j et le centre i , c'est-à-dire :

$$c_{ij} = \alpha \cdot d_{ij} + \beta \cdot u_i$$

où α est un coefficient de pondération, qui peut représenter le coût de transport par km, augmenter au cours du temps (l'inflation). Ce coefficient nous permettra d'étudier différents scénarios et représenter visuellement les coûts en fonction de ce coût.

où β est le coût de maintenance fixe. Il nous permet de limiter le nombre de centres selon leur coût ou le nombre d'échanges entre les centres, puisque selon ce montant il peut être plus intéressant d'en construire plusieurs ou d'en construire quelques uns et alors privilégier les échanges entre eux.

- *Coût f_i* :

Le coût d'installation f_i est directement issu des données de l'instance :

$$f_i = \bar{f}_i$$

Contraintes du modèle

Contraintes UFLP :

- $\sum_{j=1}^n v_{ij} = 1 \quad \forall i \in \{1, \dots, n\}$ (chaque ville est affectée à un centre)
- $v_{ij} \leq u_j \quad \forall i, j \in \{1, \dots, n\}$ (affectation seulement à un centre ouvert)
- $v_{ii} = u_i \quad \forall i \in \{1, \dots, n\}$ (si une ville est un centre, elle se dessert elle-même)

Contrainte CFLP : les contraintes d'UFLP +

$$\sum_{j=1}^n v_{ij} \leq q \quad \forall i \quad (\text{un centre peut desservir au plus } q \text{ villes})$$

1.2 Modélisation PLNE

Question 3 En utilisant le solveur CPLEX au travers de JuMP, trouver et implémenter un PLNE pour chacun des problèmes P1 et P2.

Voici le PLNE pour le problème 1 UFLP :

Fonction UFLP

```

function UFLP(alpha, beta, tabX, tabY, f, n, nom_fichier_base)
    m = Model(CPLEX.Optimizer)
    tabDistance = distance(tabX, tabY)

    @variable(m, u[1:n], Bin) # u[i] = 1 si la ville i est un
                                centre
    @variable(m, v[1:n, 1:n], Bin) # v[i,j] = 1 si i est
                                desservie par j

    @objective(m, Min,
        sum(f[j] * u[j] for j in 1:n) + # Cout fixe d'ouverture
        sum(alpha * tabDistance[i,j] * v[i,j] + beta * v[i,j] for
            i in 1:n, j in 1:n) # Cout de transport et gestion
    )

    @constraint(m, c1[i in 1:n], sum(v[i,j] for j in 1:n) == 1) #
        Chaque ville doit etre desservie
    @constraint(m, c2[i in 1:n, j in 1:n], v[i,j] <= u[j]) #
        Un centre doit etre ouvert pour desservir
    @constraint(m, c3[i in 1:n], v[i,i] == u[i]) #
        Une ville centre se dessert elle-meme

    t1 = now()
    set_silent(m) # Desactive les logs
    optimize!(m)
    t2 = now()
    temps = t2 - t1

    status = termination_status(m)
    val = -1.0

    if status == OPTIMAL || status == FEASIBLE_POINT
        val = objective_value(m)
        solution_u = [value(u[i]) for i in 1:n]
        nb_centres = count(x -> x > 0.5, solution_u)

        println("Solution optimale trouvée : ")
        println("Nombre total de villes : ", n)
        println("Nombre de centres ouverts : ", nb_centres)
        println("Objectif (cout total) : ", round(val, digits=2))
        println("Temps de résolution : ", round(Millisecond(temps
            ).value / 1000, digits=3), " secondes")

        Dessine_Solution_Placement(replace(nom_fichier_base, "."
            f"flp" => "_UFLP.flp"), n, tabX, tabY, solution_u)
    else
        println("Pas de solution optimale. Statut : ", status)
    end

    return status, val
end

```

Voici le PLNE pour le problème 2 CFLP :

Fonction UFLP

```

function CFLP(alpha, beta, tabX, tabY, f, n, q, nom_fichier_base)
    m = Model(CPLEX.Optimizer)
    tabDistance = distance(tabX, tabY)

    @variable(m, u[1:n], Bin)
    @variable(m, v[1:n, 1:n], Bin)

    @objective(m, Min,
        sum(f[j] * u[j] for j in 1:n) +
        sum(alpha * tabDistance[i,j] * v[i,j] + beta * v[i,j] for
            i in 1:n, j in 1:n)
    )

    @constraint(m, c1[i in 1:n], sum(v[i,j] for j in 1:n) == 1)
        # Chaque ville est affectee
    @constraint(m, c2[i in 1:n, j in 1:n], v[i,j] <= u[j])
        # Affectation seulement a un centre
        ouvert
    @constraint(m, c3[i in 1:n], v[i,i] == u[i])
        # Si une ville est un centre
        , elle se dessert
    @constraint(m, c4[j in 1:n], sum(v[i,j] for i in 1:n) <= q *
        u[j])      # Contrainte de capacite

    t1 = now()
    set_silent(m)
    optimize!(m)
    t2 = now()
    temps = t2 - t1

    status = termination_status(m)
    val = -1.0

    if status == OPTIMAL || status == FEASIBLE_POINT
        val = objective_value(m)
        solution_u = [value(u[i]) for i in 1:n]
        nb_centres = count(x -> x > 0.5, solution_u)

        println("Solution optimale trouvée : ")
        println("Nombre total de villes : ", n)
        println("Nombre de centres ouverts : ", nb_centres)
        println("Objectif (cout total) : ", round(val, digits=2))
        println("Temps de resolution : ", round(Millisecond(temps
            ).value / 1000, digits=3), " secondes")

        Dessine_Solution_Placement(replace(nom_fichier_base, "."
            flp" => "_CFLP.flp"), n, tabX, tabY, solution_u)
    else
        println("Pas de solution optimale. Statut : ", status)
    end

    return status, val
end

```

Question 4 Jusqu'à quelle taille peut-on résoudre les instances pour P1 ? pour P2 ? (au besoin vous pouvez prendre une partie réduite d'une instance fournie).

Pour répondre à cette question, nous allons fixer les paramètres suivants :

$$\alpha = 1, \quad \beta = 0.5, \quad \text{et} \quad q = 9.$$

Ces valeurs permettent de pondérer respectivement le coût de transport, le coût de gestion, ainsi que la capacité maximale des centres dans notre modèle.

Tests de résolution sur différentes instances

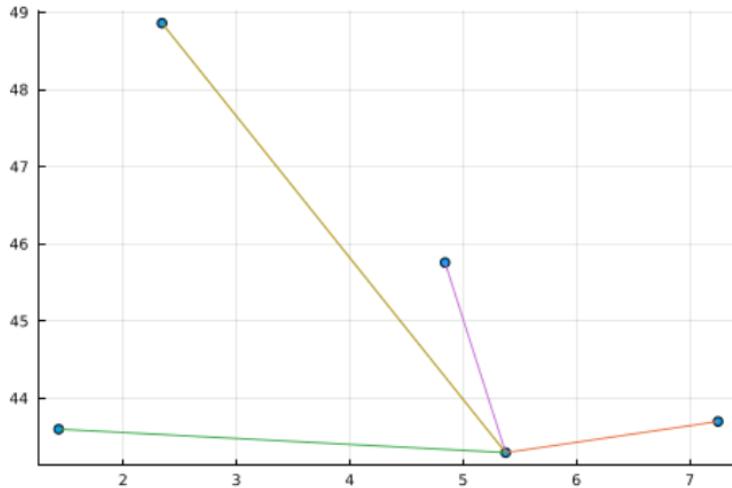
Afin d'évaluer les limites de performance des algorithmes UFLP et CFLP, nous avons réalisé plusieurs tests sur des instances de tailles croissantes. Pour chaque test, nous affichons la sortie du terminal ainsi que les représentations graphiques des solutions obtenues.

- **1^{er} test :** inst_300000 — Instance contenant 5 villes.

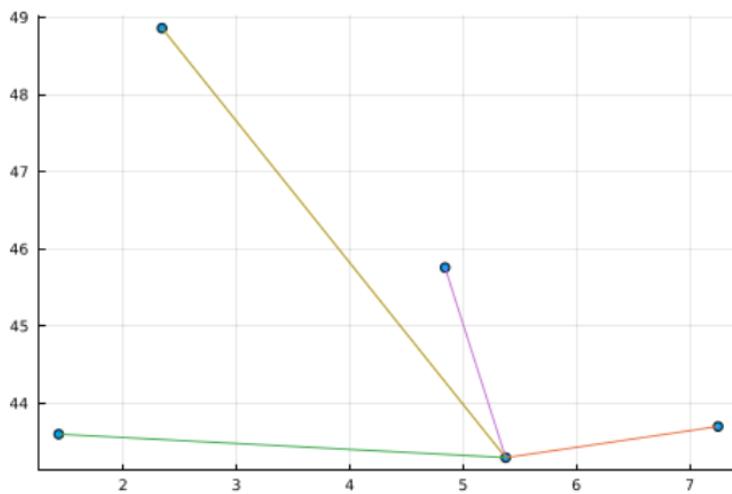
```
--- Résolution du problème UFLP ---
 Solution optimale trouvée :
Nombre total de villes : 5
Nombre de centres ouverts : 1
Objectif (coût total) : 27.15
⌚ Temps de résolution : 0.047 secondes
Création du fichier pdf de la solution: inst_300000_UFLP_sol.pdf
Statut de résolution UFLP : OPTIMAL
Valeur optimale (UFLP) : 27.15431328225381

--- Résolution du problème CFLP ---
 Solution optimale trouvée :
Nombre total de villes : 5
Nombre de centres ouverts : 1
Objectif (coût total) : 27.15
⌚ Temps de résolution : 0.005 secondes
Création du fichier pdf de la solution: inst_300000_CFLP_sol.pdf
Statut de résolution CFLP : OPTIMAL
Valeur optimale (CFLP) : 27.15431328225381
Création du fichier pdf de l'instance: inst_300000_instance.pdf
```

Sortie du terminal — test 1 (5 villes)



Affichage graphique de la solution UFLP — test 1



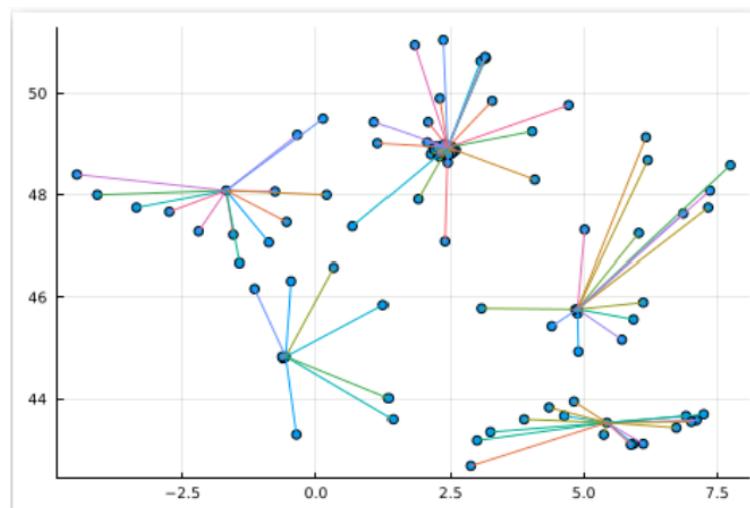
Affichage graphique de la solution CFLP — test 1

- 2^e test : inst_50000 — Instance contenant 114 villes.

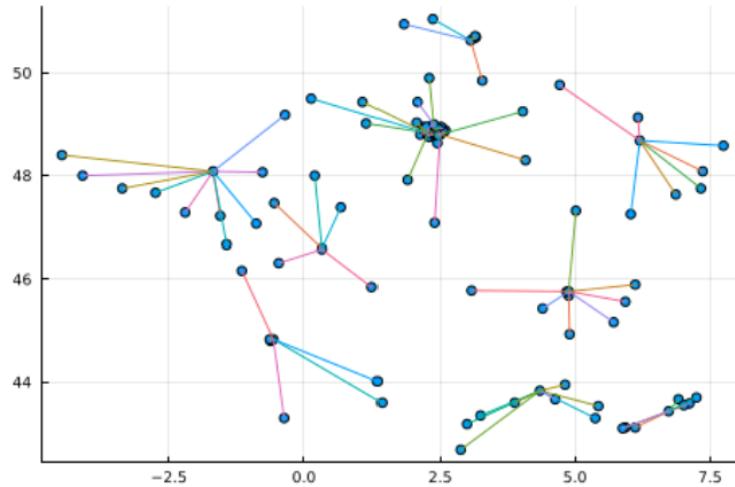
```
--- Résolution du problème UFLP ---
✓ Solution optimale trouvée :
Nombre total de villes : 114
Nombre de centres ouverts : 5
Objectif (coût total) : 250.39
⌚ Temps de résolution : 0.607 secondes
Création du fichier pdf de la solution: inst_50000_UFLP_sol.pdf
Statut de résolution UFLP : OPTIMAL
Valeur optimale (UFLP) : 250.38547085149725

--- Résolution du problème CFLP ---
✓ Solution optimale trouvée :
Nombre total de villes : 114
Nombre de centres ouverts : 13
Objectif (coût total) : 326.54
⌚ Temps de résolution : 4.544 secondes
Création du fichier pdf de la solution: inst_50000_CFLP_sol.pdf
Statut de résolution CFLP : OPTIMAL
Valeur optimale (CFLP) : 326.5382986066668
Création du fichier pdf de l'instance: inst_50000_instance.pdf
```

Sortie du terminal — test 2 (114 villes)



Affichage graphique de la solution UFLP — test 2

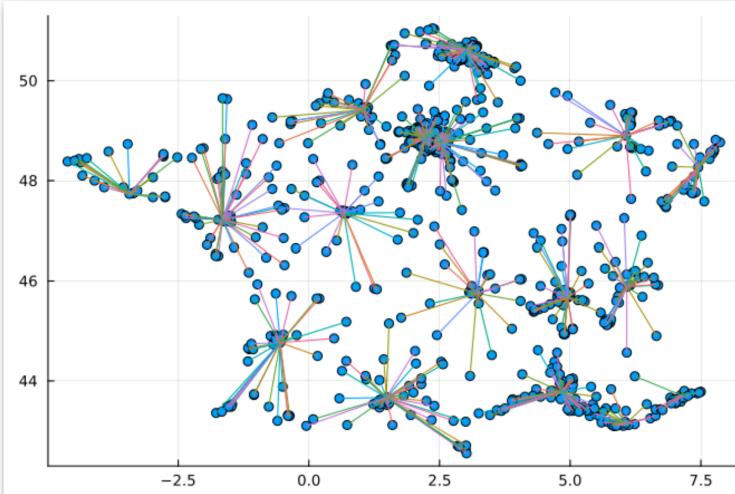


Affichage graphique de la solution CFLP — test 2

- 3^e test : inst_10000 — Instance de 891 villes.

```
--- Résolution du problème UFLP ---
✓ Solution optimale trouvée :
Nombre total de villes : 891
Nombre de centres ouverts : 17
Objectif (coût total) : 1133.45
⌚ Temps de résolution : 55.51 secondes
Création du fichier pdf de la solution: inst_10000_UFLP_sol.pdf
□
```

Sortie du terminal — test 3 (891 villes)



Affichage graphique de la solution UFLP — test 3

Remarque : Pour cette instance, l'algorithme CFLP n'a pas pu aboutir. La résolution échoue en raison de contraintes mémoire ou de temps sur notre machine. Nous avons donc décidé de réduire progressivement la taille de l'instance pour évaluer les capacités de résolution de CFLP.

- 4^e test : Instance partielle — 444 villes.

Résultat : Échec de la résolution CFLP. Nous avons poursuivi les essais en diminuant encore le nombre de villes.

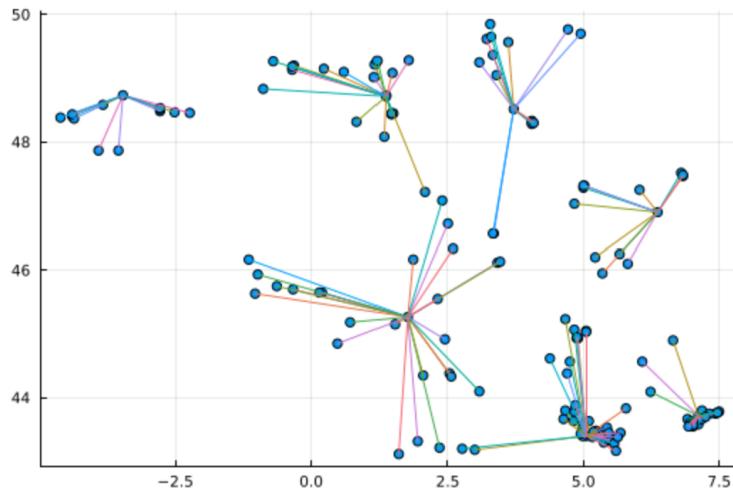
- **5^e test :** Instance partielle — 151 villes.

```
julia> include("PLNE.jl")
Le fichier contient 151 villes

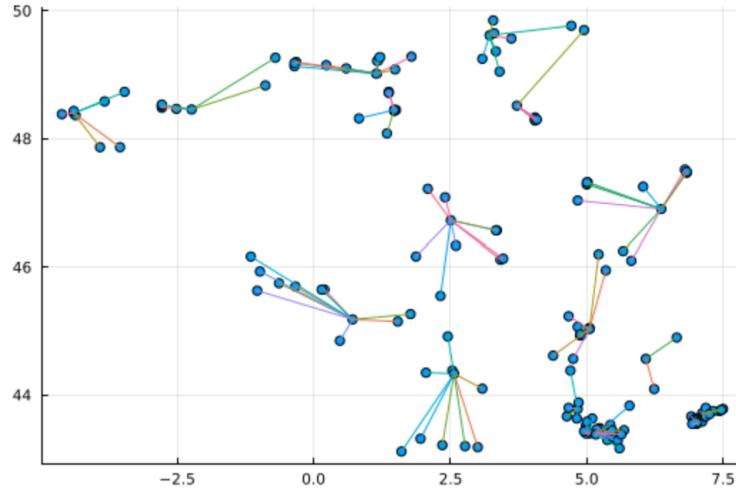
--- Résolution du problème UFLP ---
✓ Solution optimale trouvée :
Nombre total de villes : 151
Nombre de centres ouverts : 7
Objectif (coût total) : 311.77
⌚ Temps de résolution : 0.444 secondes
Création du fichier pdf de la solution: inst_223villes_UFLP_sol.pdf
Statut de résolution UFLP : OPTIMAL
Valeur optimale (UFLP) : 311.77305924588546

--- Résolution du problème CFLP ---
✓ Solution optimale trouvée :
Nombre total de villes : 151
Nombre de centres ouverts : 17
Objectif (coût total) : 412.19
⌚ Temps de résolution : 38.713 secondes
Création du fichier pdf de la solution: inst_223villes_CFLP_sol.pdf
Statut de résolution CFLP : OPTIMAL
Valeur optimale (CFLP) : 412.1920613528115
Création du fichier pdf de l'instance: inst_223villes_instance.pdf
```

Sortie du terminal — test 5 (151 villes)



Affichage graphique de la solution UFLP — test 5



Affichage graphique de la solution CFLP — test 5

- **6^e test :** Tentatives sur de très grandes instances (plus de 1000 villes).

Résultat : Les essais avec des instances plus grandes (ex : 1000, 3000, 9000 villes) n'ont pas abouti, tant pour UFLP que CFLP. La dernière instance fonctionnelle pour UFLP reste celle de 891 villes (test 3). Pour CFLP, la limite est atteinte à environ 151 villes (test 2).

1.3 Méthodes gloutonnes et méta-heuristiques

Question 5 En utilisant un langage de programmation de votre choix, vous développerez plusieurs méthodes de recherche gloutonnes pour le problème P1 et pour le problème P2. Il est recommandé d'introduire du hasard dans ces recherches gloutonnes, afin que chaque exécution puisse produire une solution différente. On rappelle qu'une méthode gloutonne est une méthode qui construit une solution réalisable en prenant des décisions partielles de manière itérative, sans revenir en arrière. Vous pouvez commencer par une méthode simple, par exemple basée sur le hasard, puis proposer une (ou plusieurs) méthodes plus élaborées exploitant les poids et/ou la nature géographique des instances.

Nous avons implémenter chacune un algorithme glouton adapté au problème P1 et P2.

Algorithme glouton (UFLP / CFLP basé sur la distance)

Le premier algorithme consiste :

- A ouvrir nbcentre centre, qui est la moitié du nombre de villes, les placer de manière aléatoire
- Chaque ville est ensuite affectée au **centre le plus proche**, en se basant uniquement sur la **distance géographique**.

Pour la version CFLP (avec capacité limitée) :

- Chaque centre ne peut desservir qu'un nombre limité de villes (par exemple $q = 9$).

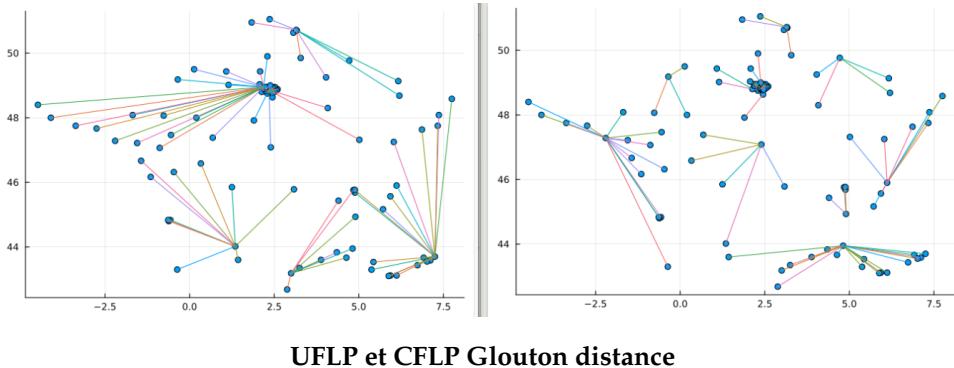
- Si un centre atteint sa capacité maximale, la ville est alors affectée au centre valide le plus proche restant.

Cette méthode est gloutonne et on peut le voir suite aux différents essais que l'on a effectué avec les instances. Par exemple pour l'instance 50 000

Création du fichier pdf de la solution: inst_50000_CFLP_sol.pdf			
Méthode	Centres ouverts	Coût total	Statut
UFLP exact	5	250.39	OPTIMAL
UFLP glouton	57	1177.08	FAISABLE
CFLP exact	13	326.54	OPTIMAL
CFLP glouton	57	1166.24	FAISABLE

Création du fichier pdf de l'instance: inst_50000_instance.pdf

Sortie du terminal



UFLP et CFLP Glouton distance

Algorithme glouton géographique

Cet algorithme construit une solution en ouvrant des centres progressivement, qui dépendant de la position de la ville. L'idée principale est de commencer avec une ville bien placée, d'y affecter les villes proches, puis d'ouvrir un nouveau centre dans une ville éloignée lorsque les précédents ne peuvent plus accueillir de villes.

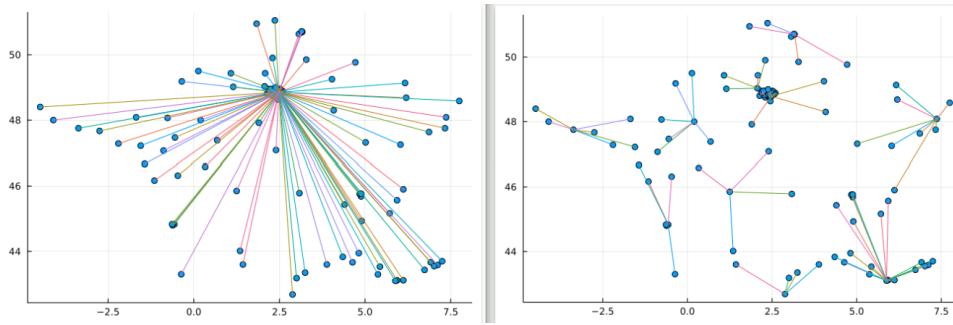
Le deuxième algorithme consiste :

- **Choisir une ville de départ** : on sélectionne une ville qui semble bien placée géographiquement avec de l'aléatoire pour varier les résultats entre différentes exécutions.
- **Ouvrir un centre dans cette ville.**
- **Affecter les villes proches** : on affecte à ce centre toutes les villes situées dans un rayon donné en paramètre, tant que cela respecte les contraintes (par exemple la capacité du centre).
- **Trouver une ville éloignée** : si certaines villes ne sont toujours pas assignées, on cherche la plus éloignée des centres existants. On ne choisit pas exactement la plus éloignée, mais on en prend une parmi celles qui sont au-delà d'un certain seuil (par exemple 80% de la distance maximale) pour introduire un peu d'aléatoire.
- **5. Répéter** : on recommence les étapes 2 à 4 jusqu'à ce que toutes les villes soient assignées à un centre.

Par exemple pour l'instance 50 000

Méthode	Centres ouverts	Coût total	Statut
UFLP exact	5.0	250.39	OPTIMAL
CFLP exact	13.0	326.54	OPTIMAL
UFLP glouton	7	322.24	FAISABLE
CFLP glouton	13	381.73	FAISABLE
UFLP glouton géo	1	402.73	FAISABLE
CFLP glouton géo	13	657.43	FAISABLE

Sortie du terminal



UFLP et CFLP Glouton géographique

Question 6 Implémentez une méta-heuristique de descente stochastique pour le problème P1, et une autre pour le problème P2. La descente stochastique itérée consiste à relancer plusieurs fois une descente stochastique à partir de solutions initiales générées aléatoirement, dans le but d'explorer l'espace des solutions plus efficacement.

Descente stochastique itérée

On doit écrire un algorithme métahéuristique qui va générer des solutions aléatoires et les améliorer localement.

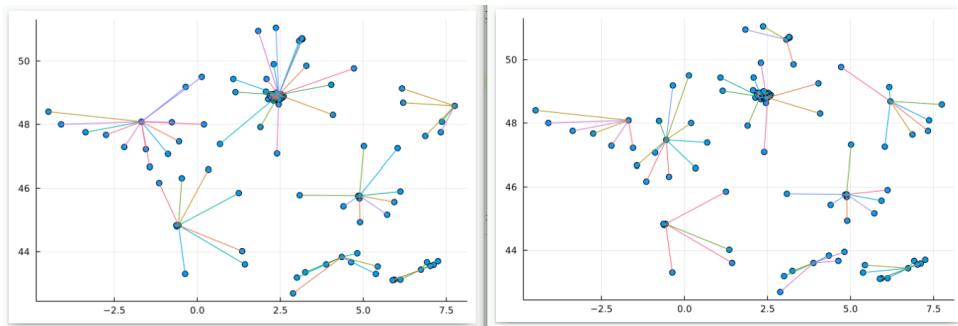
Notre algorithme consiste à :

- On commence par générer une solution aléatoire réalisable, c'est-à-dire en ouvrant p centres dans des villes aléatoires.
- Ensuite, à chaque itération, on modifie légèrement cette solution (par exemple en fermant un centre et en ouvrant un autre ailleurs), et on garde la solution si elle améliore le coût.
- On répète ce processus plusieurs fois (défini par `itermax`), puis on recommence tout avec une nouvelle solution initiale. Cela permet d'explorer plusieurs zones de l'espace des solutions.
- À la fin, on conserve la meilleure solution trouvée pendant tous les essais.

C'est une méthode assez efficace car elle explore plusieurs solutions possibles sans rester bloquée dans un mauvais minimum local. Voici un exemple sur l'instance 50 000

Méthode	Centres ouverts	Coût total	Statut
UFLP exact	5.0	250.39	OPTIMAL
CFLP exact	13.0	326.54	OPTIMAL
UFLP glouton	7	322.24	FAISABLE
CFLP glouton	13	381.73	FAISABLE
UFLP glouton géo	1	402.73	FAISABLE
CFLP glouton géo	13	657.43	FAISABLE
UFLP descente	7	251.74	FAISABLE
CFLP descente	13	331.48	FAISABLE

Sortie du terminal



UFLP et CFLP Glouton géographique

Nous avons choisi d'utiliser le langage Julia car ce projet représentait pour nous une bonne opportunité de découvrir et approfondir son utilisation dans un contexte d'optimisation hors PL et PLNE. Julia nous a permis de comparer facilement la mise en œuvre des modèles exacts (UFLP et CFLP formulés en PLNE) avec celle des algorithmes approchés, comme les méthodes gloutonnes ou la métahéuristique de descente locale. Enfin, la gestion et l'importation des données étaient plus simple car certaines fonctions de traitement nous ont été fournies par l'enseignant. Cela nous a permis de nous concentrer davantage sur la modélisation et l'analyse des résultats, tout en gagnant du temps dans la préparation des tests.

Comparaison expérimentale des méthodes de résolution

Analyse expérimentale à l'aide du gap entre borne inférieure et borne supérieure

Ici on va évaluer les performances des heuristiques à l'aide de la mesure de **gap**, définie par la formule suivante :

$$\text{gap} = \frac{z_{\text{bestsol}} - z_b}{z_{\text{bestsol}}}$$

où z_b est la *borne inférieure* obtenue par la **relaxation linéaire** du programme en nombres entiers, et z_{bestsol} est la meilleure valeur réalisable obtenue par l'heuristique considérée. Cette mesure donne une indication de la proximité d'une solution réalisable par rapport à la borne théorique minimale.

Pour chaque problème (UFLP ou CFLP), nous avons résolu la relaxation linéaire en relâchant les contraintes d'intégralité des variables u et v , les remplaçant par des variables continues dans $[0, 1]$. Grâce à ça, nous avons une estimation inférieur de la solution opt entière. Nous avons écrit 2 fonctions différentes pour cela :

- `borne_inferieure_relax_UFLP` pour le problème UFLP ;
- `borne_inferieure_relax_CFLP` pour le problème CFLP.

```
 Analyse UFLP :
Borne inférieure UFLP (relax) : 250.15
 Gap UFLP (glouton) : 43.16%
 Limité : UFLP glouton peine avec la borne relaxée
 Gap UFLP (glouton géo) : 53.86%
 Limité : UFLP glouton géo peine avec la borne relaxée
 Gap UFLP (descente) : 0.63%
 Excellent : UFLP descente est très proche de la borne relaxée
```

Gap pour UFLP

```
 Analyse CFLP :
Borne inférieure CFLP (relax) : 323.97
 Gap CFLP (glouton) : 24.83%
 Limité : CFLP glouton peine avec la borne relaxée
 Gap CFLP (glouton géo) : 54.5%
 Limité : CFLP glouton géo peine avec la borne relaxée
 Gap CFLP (descente) : 2.43%
 Excellent : CFLP descente est très proche de la borne relaxée
```

Gap pour CFLP

Un **gap faible** (proche de 0%) indique que la méthode heuristique fournit une solution très proche de l'optimal (ou de sa relaxation). Nous avons constaté que :

- Les algorithmes gloutons simples obtiennent des gaps de l'ordre de 7 à 65%, montrant une certaine efficacité dans certains cas mais ce n'est pas optimal.
- Tandis qu'une approche, comme la *descente stochastique itérée*, nous permet d'observer des gaps souvent inférieurs à 3%, ce que l'on peut interpréter comme une excellente performance.

Cette analyse nous montre donc l'intérêt de comparer les heuristiques non seulement à la solution optimale, mais aussi à la borne inférieure obtenue via la relaxation linéaire.

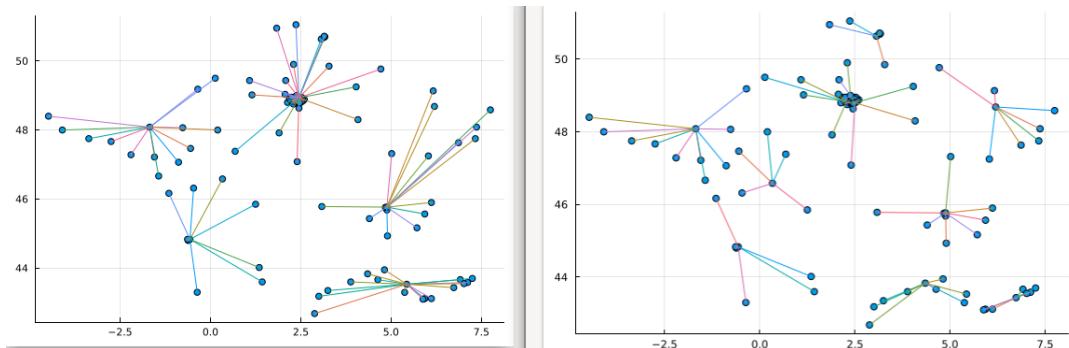
Question 8 Pour chacun des deux problèmes P1 et P2, vous devez comparer expérimentalement la validité de votre code sur des instances de tailles réduites

- valider que vous obtenez une solution réalisable
- visualiser vos solutions.

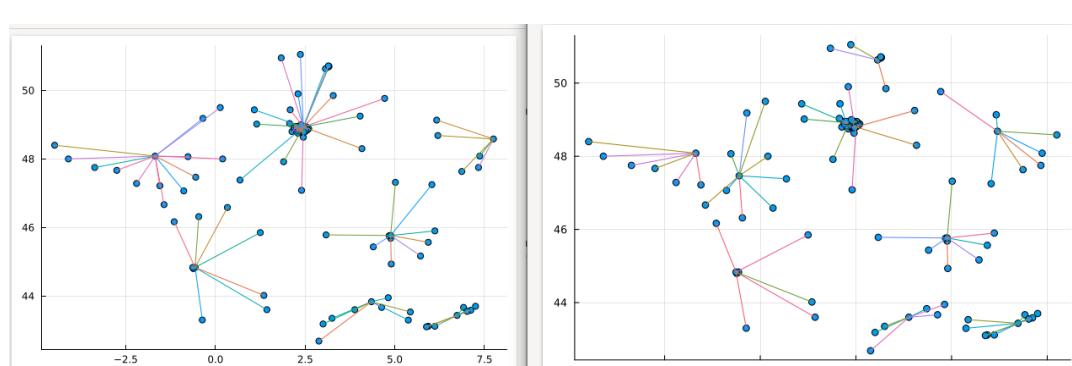
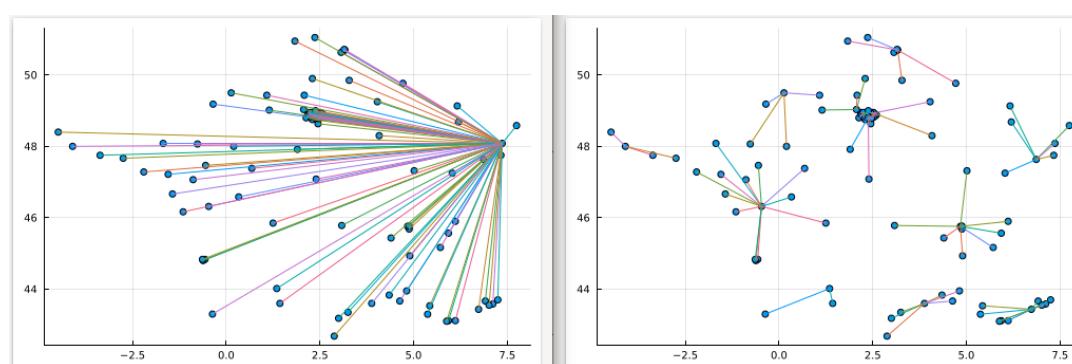
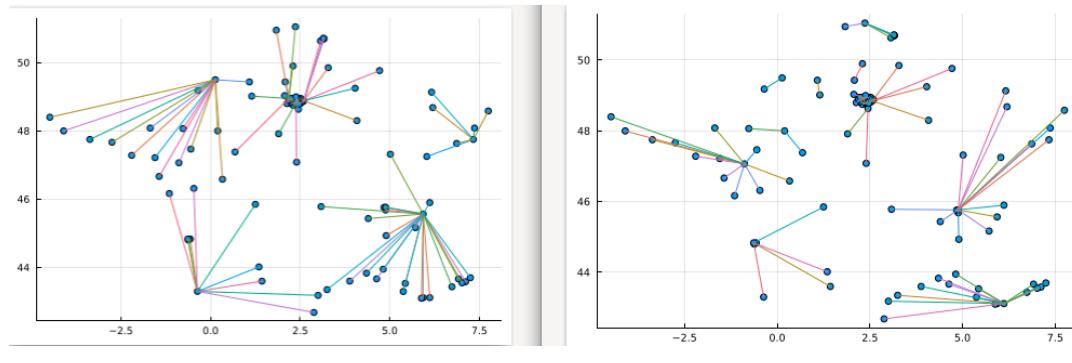
Instance	inst_50000	inst_10000	inst_1000	inst_500
Nb Villes	114	891	9730	17401
UFLP	nb centre ouvert : 5 coût : 250.39 temps : 4.349	nb centre ouvert : 17 coût : 1133.45 temps : 46.623	non résolvable	non résolvable
CFLP	nb centre ouvert : 13 coût : 326.54 temps : 4.302	non résolvable	non résolvable	non résolvable
glouton distance UFLP	nb centre ouvert : 7 coût : 283.93 temps : 0.007	nb centre ouvert : 24 coût : 1152 temps : 0.005	nb centre ouvert : 79 coût : 5630.14 temps : 2.111	nb centre ouvert : 106 coût : 8879.74 temps : 8.388
glouton distance CFLP	nb centre ouvert : 13 coût : 441.44 temps : 0.001	nb centre ouvert : 99 coût : 2522.77 temps : 0.009	non résolvable	non résolvable
glouton geo UFLP	nb centre ouvert : 1 coût : 699.92 temps : 0.0	nb centre ouvert : 1 coût : 5596.82 temps : 0.019	non résolvable	non résolvable
glouton geo CFLP	nb centre ouvert : 13 coût : 631.68 temps : 0.001	nb centre ouvert : 99 coût : 5114.48 temps : 0.209	non résolvable	non résolvable
descente stochastique UFLP	nb centre ouvert : 7 coût : 251.74 temps : 6.058	non résolvable	non résolvable	non résolvable
descente stochastique CFLP	nb centre ouvert : 13 coût : 332.24 temps : 15.811	non résolvable	non résolvable	non résolvable

TABLE 2.1: Résultats expérimentaux pour les instances de différentes tailles

Instance 50 000:

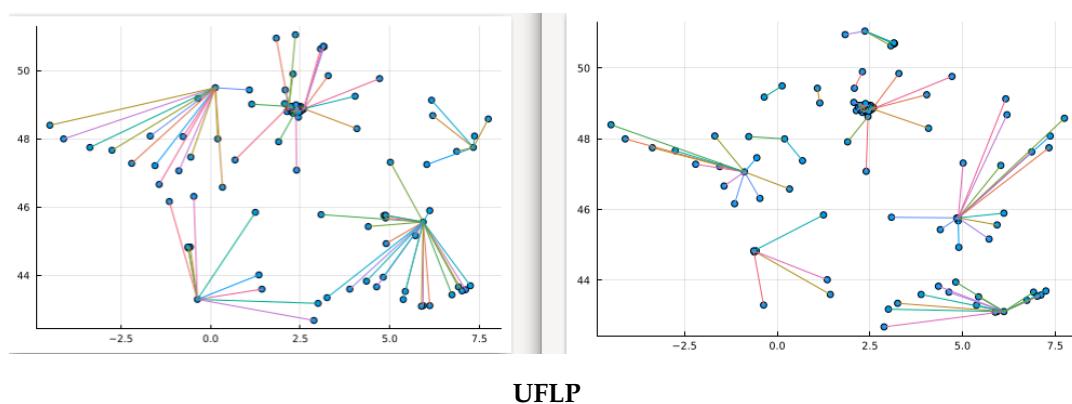


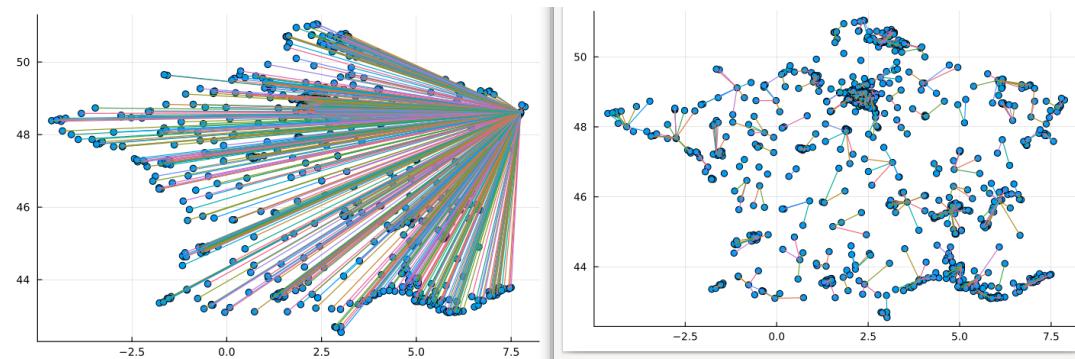
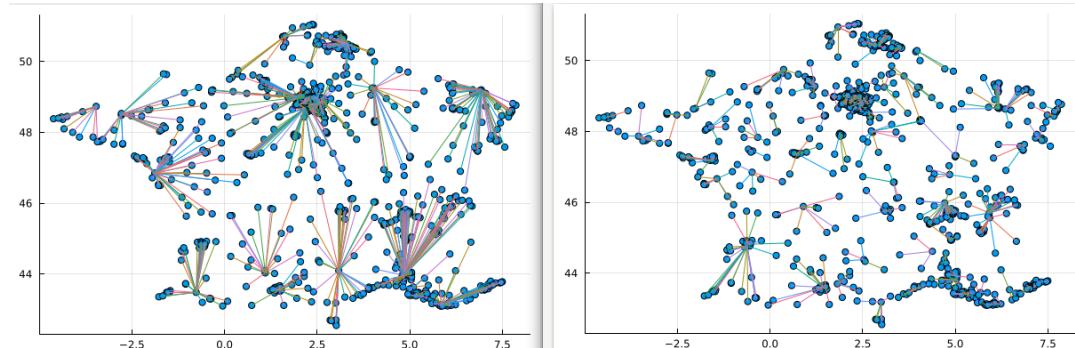
UFLP et CFLP



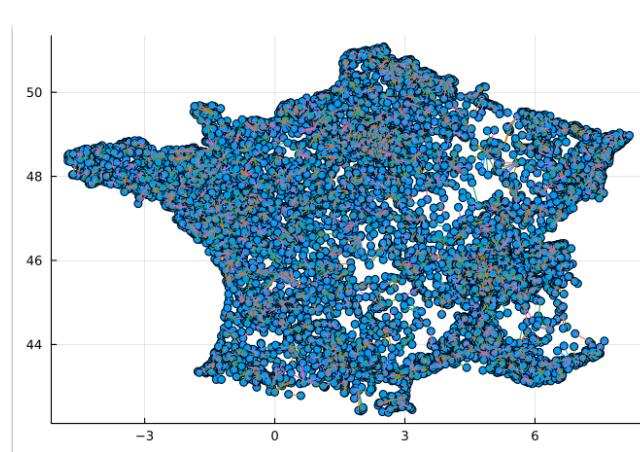
UFLP et CFLP descente stochastique

Instance 10 000:

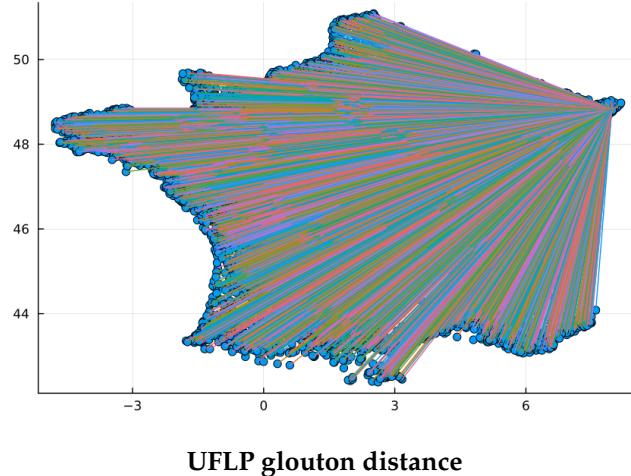




Instance 1000 :



Instance 500 :



Question 9 Votre mini-rapport doit comporter une comparaison des différentes méthodes exactes et approchées.

En répondant aux différentes questions suivantes :

- taille maximale des instances pouvant être résolues en moins de 5 minutes ;
- qualité des solutions obtenues (écart par rapport à la solution optimale quand vous l'avez, écart à la meilleure borne connue si possible).

Les comparaisons doivent porter sur un jeu d'essai de taille suffisante.

1. Taille maximale résolue en moins de 5 minutes

Problème	Méthode	Taille max (villes)	Temps typique
UFLP	PLNE exacte	891	46 s
	Descente stochastique	114 (échec > 114)	6 s
	Glouton distance / géo	17 401	≤ 8 s
CFLP	PLNE exacte	151	4 s
	Descente stochastique	114 (limite de capacité)	16 s
	Glouton distance / géo	9 730	≤ 0.21 s

TABLE 2.2: Taille maximale résolue en moins de 5 minutes

2. Qualité des solutions (écart à la borne relaxée)

Problème	Borne relaxée	Glouton	Glouton géo	Descente
UFLP	250.15	+43.16%	+53.86%	+0.63%
CFLP	323.97	+24.83%	+54.50%	+2.43%

TABLE 2.3: Écart relatif des méthodes approchées par rapport à la borne relaxée (voir tableau 2.2 du rapport)

Conclusion question 9

- Dans le tableau 2, on trouve les méthodes inférieures à 5 min pour les deux problèmes, on peut également voir dans le tableau 1 que certains ne peuvent pas être résolues sur notre machine ou avec la méthode.
- La méthode exacte (PLNE) permet d'obtenir la solution optimale, mais devient impraticable au-delà de 1 000 variables à cause de la taille.
- La descente stochastique est un bon compromis entre qualité et temps de calcul : écart $\leq 3\%$ pour un temps inférieur à 20 secondes. C'est donc une bonne option, mais également limitée en taille, il faudrait changer la valeur de certains paramètres pour pouvoir résoudre avec cette méthode pour de plus grande instance.
- Les deux versions gloutonnes (distances et géographique) sont presque instantanées, elles sont capables de traiter des instances de plus de 10 000 villes, mais le coût est beaucoup plus important : entre 25% et 55%.

Comparaisons des solutions des deux problèmes P1 et P2

Question 10 Comparer les solutions obtenues en termes opérationnels. En évaluant de ma manière critique laquelle est la plus intéressante pour le problème de placement de centres logistiques

Critère	UFLP (sans capacité)	CFLP (avec capacité q)	Analyse
Nb centres (instance 114)	5	13	UFLP ouvre moins de sites, coûts plus faible
Coût total (instance 114)	250	327	L'ajout de la contrainte de capacité augmente de $\sim 30\%$ le coût
Taille PLNE	891 villes	151 villes	la contrainte de capacité complique fortement la résolution
Pertinence	Peu pertinent car les centres n'ont pas une capacité illimitée, ils ne peuvent pas desservir un nombre de villes illimités	indispensable car souvent la capacité physique (surface, stock, personnel) est bornée	Le plus pertinent est donc de prendre le problème P2, CFPL car c'est le plus pertinent et réaliste

En pratique :

Dans le cas où l'entreprise est libre niveau capacité de dessert pour chaque plate-forme, la solution UFLP permet de minimiser les coûts fixes tout en conservant une structure simple.

Dès qu'il existe des limites opérationnelles (surface, équipes, réglementation), les contraintes CFLP s'impose malgré un coût plus élevé, car ce problème permet d'éviter la saturation des centres et de préserver la qualité de service sur le long terme.

Pour conclure, pour un réseau logistique réel où la capacité est rarement illimitée, le CFLP fournit une solution plus robuste et durable d'un point de vue opérationnel,

tandis que l'UFLP reste une bonne approximation économique lorsque la contrainte de capacité peut être relâchée.

Conclusion

Ce mini-projet nous a permis de mieux comprendre les enjeux liés à la localisation de centres logistiques, notamment la différence entre le modèle avec et sans contraintes. Nous avons expérimenter différentes approches, avec des méthodes exactes et heuristiques pour constater les avantages et limite de chacun. Pour conclure, les résultats montrent que la descente locale permet d'obtenir une solution de bonne qualité, de manière rapide, et on observe que le modèle avec capacité (CFLP) est le plus réaliste pour représenter les contraintes sur le terrain.