



Mini-Projet d'Optimisation Combinatoire

Placement de Centres Logistiques

Abdelillah SELHI et Amziane HAMRANI

Université Sorbonne Paris Nord – SupGalilée

13 juin 2025

Table des matières

1	Introduction	2
2	Partie 1 : Problème des P-médian	3
2.1	Définition	3
2.2	Coûts pris en compte	3
2.3	Formulation mathématique pour PLNE	3
2.4	Modèles exacts (PLNE)	4
2.5	Heuristiques gloutonnes	6
2.6	P_median_descente_stoch_iteree	9
3	Partie 2 : Problème des P-centre	11
3.1	Définition	11
3.2	Coûts pris en compte	11
3.3	Formulation mathématique pour PLNE	11
3.4	PLNE_P_centre	11
3.5	P_centre_greedy_random	12
3.6	P_centre_choix_villes_elognées	13
3.7	P_centre_greedy_metaheuristique	14
3.8	P_centre_descente_stoch_iteree	16
4	Calcul de l'écart relatif	17
4.1	P-médian	17
4.2	P-centre	19
5	Scénario opérationnel et économique	22
5.1	Mise à l'échelle des distances	22
5.2	Coût du transport	22
5.3	Coût de construction des centres	22
5.4	Coût global	22
6	Comparaison expérimentale : P-médian vs P-centre	23
6.1	Présentation des résultats	23
6.2	Résultats pour le P-médian	23
6.3	Résultats pour le P-centre	23
6.4	Analyse critique	23
6.5	P-médian avec coût d'installation (PLNE avec f_j)	24

1 Introduction

Dans ce projet, nous simulons une entreprise de livraison de type Amazon désirant implanter plusieurs centres logistiques à travers la France. L'objectif principal est de minimiser le coût global de distribution, qui se compose de deux éléments majeurs :

- Les coûts de transport, proportionnels à la distance (dépendant de la consommation de carburant),
- Les coûts fixes d'installation de chaque centre.

Nous étudions deux problèmes fondamentaux d'optimisation de localisation :

1. Le problème des **P-médian**, centré sur la distance totale,
2. Le problème des **P-centre**, centré sur la distance maximale.

Pour chacun de ces problèmes, nous présenterons des modèles exacts (PLNE) et des méthodes heuristiques (gloutonnes et métahéuristique).

2 Partie 1 : Problème des P-médian

2.1 Définition

Le problème des P-médian consiste à sélectionner P centres parmi n villes de façon à minimiser la somme des distances entre chaque ville et le centre auquel elle est assignée.

2.2 Coûts pris en compte

Deux variantes :

- **Sans coûts fixes** : seul le coût de transport $c_{ij} = d_{ij} \times \text{tarif_essence}$ est considéré,
- **Avec coûts fixes** : ajout d'un coût fixe f_j pour chaque centre ouvert.

2.3 Formulation mathématique pour PLNE

Variables :

$$x_{ij} = \begin{cases} 1 & \text{si la ville } i \text{ est affectée au centre } j, \\ 0 & \text{sinon,} \end{cases} \quad y_j = \begin{cases} 1 & \text{si centre ouvert à } j, \\ 0 & \text{sinon.} \end{cases}$$

Objectif sans coûts fixes :

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}.$$

Objectif avec coûts fixes :

$$\min \sum_{i,j} c_{ij} x_{ij} + \sum_j f_j y_j.$$

Contraintes :

$$\begin{aligned} \sum_{j=1}^n x_{ij} &= 1, & \forall i, \\ x_{ij} &\leq y_j, & \forall i, j, \\ \sum_{j=1}^n y_j &= P, \\ x_{ij}, y_j &\in \{0, 1\}. \end{aligned}$$

2.4 Modèles exacts (PLNE)

PLNE_P_median sans f

Entrées : filename, P.

Étapes clés :

1. Lecture du fichier et extraction des coordonnées (`tabX`, `tabY`) et du vecteur des coûts fixes f (non utilisées dans cette variante),
2. Calcul de la matrice des distances d_{ij} entre chaque paire de villes,
3. Définition de la variable binaire $x[i, j]$ indiquant si la ville i est desservie par le centre j (on utilise implicitement $y[j]=x[j, j]$ pour modéliser l'ouverture du centre en j),
4. Ajout des contraintes :
 - ouverture d'au plus P centres via `sum(x[j, j] for j in 1:n) <= P`,
 - affectation unique de chaque ville via `sum(x[i, j] for j in 1:n) == 1`,
 - desserte conditionnée à l'ouverture du centre via $x[i, j] \leq x[j, j]$.
5. Objectif : Min `sum(d[i, j] * x[i, j] for i in 1:n, j in 1:n)`,
6. Résolution du modèle PLNE avec CPLEX et extraction de la valeur objectif et des variables x .

Sortie : Tuple (table des indices de centres, table binaire de toutes les villes (1 pour ville contenant un centre, 0 sinon)).

Listing 1 – PLNE without f

```
using JuMP, CPLEX
function PLNE_P_median(filename, P)
    n=0
    tabX=Float64 []
    tabY=Float64 []
    f= Float64 []
    println("Lecture du fichier: ", filename)

    n= Lit_fichier_Placement(filename, tabX, tabY, f)

    println("Le fichier contient ",n, " villes")
    d=[dist(tabX[i],tabY[i],tabX[j],tabY[j]) for i in 1:n, j in 1:n
       ]

    m = Model(CPLEX.Optimizer)
    @variable(m, x[1:n,1:n], Bin)
    @objective(m, Min, sum(d[i,j]*x[i,j] for i in 1:n, j in 1:n))
    @constraint(m,sum(x[j,j] for j in 1:n)<=P)
    for i in 1:n
        @constraint(m,sum(x[i,j] for j in 1:n)==1)
        for j in 1:n
            @constraint(m,x[i,j] <=x[j,j])
        end
    end
end
```

```

optimize!(m)
...
end

```

PLNE_P_median avec f

Entrées : filename, P.

Étapes clés : même structure que pour le modèle sans coûts fixes, avec en plus :

- ajout du terme `sum(f[j] * x[j,j] for j in 1:n)` à la fonction objectif pour prendre en compte les coûts d'installation,

le reste (variables, contraintes d'affectation et d'ouverture, résolution) est identique.

Sortie : tuple (indices des centres ouverts, vecteur binaire de desserte des villes).

Listing 2 – PLNE_{P median with f}

```

function PLNE_P_median_with_f(filename ,P)
    n=0
    tabX=Float64 []
    tabY=Float64 []
    f= Float64 []

    println("Lecture du fichier: ", filename)

    n= Lit_fichier_Placement(filename , tabX , tabY , f)

    println("Le fichier contient ",n, " villes")
    d=[dist(tabX[i],tabY[i],tabX[j],tabY[j]) for i in 1:n, j in 1:n
       ]

    m = Model(CPLEX.Optimizer)
    @variable(m, x[1:n,1:n], Bin)

    @objective(m, Min, sum(d[i,j]*x[i,j] for i in 1:n, j in 1:n)+
               sum(f[j]*x[j,j] for j in 1:n))
    @constraint(m,sum(x[j,j] for j in 1:n)<=P)
    for i in 1:n
        @constraint(m,sum(x[i,j] for j in 1:n)==1)
        for j in 1:n
            @constraint(m,x[i,j] <=x[j,j])
        end
    end

    optimize!(m)
    ...
end

```

2.5 Heuristiques gloutonnes

P_median_greedy_random

Entrées : filename, P.

Étapes clés :

1. Lecture et calcul de la matrice d,
2. Tirage aléatoire de P centres distincts via `randperm(n) [1:P]`,
3. Construction du vecteur binaire S (1 si la ville est centre),
4. Pour chaque ville, calcul de la distance minimale à l'un des centres et sommation pour obtenir le coût total.

Sortie : (liste des indices des centres, vecteur binaire de desserte).

Listing 3 – P median greedy random

```
using Random
function P_median_greedy_random(filename , P)
    n = 0
    tabX = Float64 []
    tabY = Float64 []
    f = Float64 []

    n = Lit_fichier_Placement(filename , tabX , tabY , f)
    d = [dist(tabX[i] , tabY[i] , tabX[j] , tabY[j]) for i in 1:n , j
         in 1:n]
    # Tirage alatoire de P centres diff rents
    centres = randperm(n) [1:P]
    S = zeros(Int , n)
    for j in centres
        S[j] = 1
    end
    return centres , S
end
```

P_median_greedy**Entrées :** filename, P.**Étapes clés :**

1. Lecture et calcul de d,
2. Initialisation d'une liste vide S des centres,
3. Boucle jusqu'à $|S| = P$:
 - pour chaque ville non déjà sélectionnée, simuler son ajout à S et calculer le coût total,
 - choisir la ville qui minimise ce coût,
4. Conversion de S en vecteur binaire de desserte, calcul du coût final.

Sortie : (indices des centres, vecteur binaire).

Listing 4 – P median greedy

```

function P_median_greedy(filename, P)
    ... #initialisation + calcul tableau d de distances

    # M langer l ordre des villes pour diversifier les resultats
    ordre_villes = randperm(n)
    while length(S) < P
        meilleur = -1
        for j in ordre_villes
            if j in S
                continue
            end
            tempS = push!(copy(S), j)
            total = 0.0
            for i in 1:n
                min_dist = minimum(d[i, k] for k in tempS)
                total += min_dist
            end
            if total < meilleur_cout
                meilleur_cout = total
                meilleur = j
            end
        end
        push!(S, meilleur)
    end

    solution = zeros(Int, n)
    for j in S
        solution[j] = 1
    end
    return S, solution
end

```

P_median_greedy_metaheuristique

Entrées : filename, P, maxIter.

Étapes clés :

1. Génération d'une solution initiale aléatoire via la heuristique P_median_greedy_random,
2. Boucle de météahuristique pour maxIter itérations :
 - si trop de générations stagnent, régénérer par heuristique aléatoire,
 - sinon, échanger un centre existant par une ville non-centre aléatoire et recalculer le coût,
 - accepter le voisin si le coût baisse, sinon incrémenter le compteur de stagnation.

Sortie : (indices des centres, vecteur binaire).

Listing 5 – P median greedy metaheuristique

```
function P_median_greedy_meta(filename, P, maxIter)

  ... #initialisation + d[i,j]

  # Tirage alatoire de P centres diff rents via une
  # heuristique existante
  centres, S = P_median_greedy_random(filename, P)

  for i in 1:n
    min_dist_init = minimum(d[i, k] for k in centres)
    min_dist_total += min_dist_init
  end
  nbIter = 1
  nb_stable = 0

  while nbIter <= maxIter
    if nb_stable > (1/5)*maxIter
      centresT, STemp = P_median_greedy_random(filename, P)
      min_dist_totalT = sum(minimum(d[i, k] for k in centresT)
        ) for i in 1:n)
      if min_dist_totalT < min_dist_total
        centres = centresT
        S = STemp
        min_dist_total = min_dist_totalT
        nb_stable = 0
      end
    else
      centres_copy = copy(centres)
      nouvelle_ville = rand(setdiff(1:n, centres))
      to_be_replaced = rand(1:length(centres))
      centres_copy[to_be_replaced] = nouvelle_ville
      S_copy = zeros(Int, n)
      for j in centres_copy
        S_copy[j] = 1
      end
    end
  end
end
```

```

        total = sum(minimum(d[i, k] for k in centres_copy) for
                     i in 1:n)
        if total < min_dist_total
            centres = centres_copy
            S = S_copy
            min_dist_total = total
            nb_stable = 0
        else
            nb_stable += 1
        end
    end
    nbIter += 1
end
return centres, S
end

```

2.6 P_median_descente_stoch_iteree

Entrées : filename, P, maxIter, nbLancements.

Étapes clés :

1. Lecture des données (positions, coûts, etc.),
2. Boucle sur nbLancements essais :
 - Application de la météahuristique P_median_greedy_meta à chaque essai,
 - Calcul du coût total (somme des distances) pour la solution trouvée,
 - Conservation de la meilleure solution sur l'ensemble des essais.
3. Retour de la meilleure solution (centres et affectation) trouvée sur toutes les descentes stochastiques.

Sortie : (indices des centres de la meilleure solution, vecteur binaire d'affectation).

Listing 6 – P_median_descente_stoch_iteree

```

function P_median_descente_stoch_iteree(filename, P, maxIter,
                                         nbLancements)
    n = 0
    tabX = Float64[]
    tabY = Float64[]
    f = Float64[]
    # Lecture des données depuis le fichier
    n = Lit_fichier_Placement(filename, tabX, tabY, f)

    dist_tot = 10e10
    best_centres = []
    best_S = []

    for essai in 1:nbLancements
        centres, S = P_median_greedy_meta(filename, P, maxIter)
        dist_loc = calcul_distance_totale(centres, tabX, tabY, n)

```

```
# Meilleure solution trouv e
if dist_loc < dist_tot
    dist_tot = dist_loc
    best_centres = centres
    best_S = S
end
end
return best_centres, best_S
end
```

3 Partie 2 : Problème des P-centre

3.1 Définition

Le problème des P-centre cherche à positionner P centres afin de minimiser la distance maximale entre chaque ville et son centre.

3.2 Coûts pris en compte

Seule la distance maximale z est considérée.

3.3 Formulation mathématique pour PLNE

$$\min z$$

$$\begin{aligned} \sum_j x_{ij} &= 1, \\ x_{ij} &\leq y_j, \\ \sum_j y_j &= P, \\ \sum_j d_{ij} x_{ij} &\leq z, \\ x_{ij}, y_j &\in \{0, 1\}. \end{aligned}$$

3.4 PLNE_P_centre

Entrées : filename, P .

Étapes clés :

1. Lecture et calcul de la matrice d ,
2. Définition des variables $x[i, j]$, $y[j]$ et du rayon z ,
3. Ajout des contraintes d'affectation, d'ouverture et de respect du rayon maximal via `sum(d[i, j]*x[i, j] for j) <= z`,
4. Objectif : minimiser z ,
5. Résolution avec CPLEX.

Sortie : (indices des centres, vecteur binaire de desserte).

Listing 7 – PLNE P centre

```

function PLNE_P_centre(filename ,P)
    n=0
    tabX=Float64 []
    tabY=Float64 []
    f= Float64 []

    println("Lecture du fichier: ", filename)

    n= Lit_fichier_Placement(filename , tabX , tabY , f)

    println("Le fichier contient ",n, " villes")
    d=[dist(tabX[i],tabY[i],tabX[j],tabY[j]) for i in 1:n, j in 1:n
        ]

    m = Model(CPLEX.Optimizer)
    @variable(m, x[1:n,1:n], Bin)
    @variable(m, z>=0)

    @objective(m, Min, z)
    @constraint(m,sum(x[j,j] for j in 1:n)<=P)
    for i in 1:n
        @constraint(m,sum(x[i,j] for j in 1:n)==1)
        @constraint(m,sum(d[i,j]*x[i,j] for j in 1:n)<=z)
        for j in 1:n
            @constraint(m,x[i,j] <=x[j,j])
        end
    end
    optimize!(m)
end

```

3.5 P_centre_greedy_random

Entrées : filename, P , RCL_size.

Étapes clés :

1. Lecture et calcul de la matrice d ,
2. Initialisation avec un centre aléatoire,
3. Tant que $|S| < P$:
 - pour chaque ville non sélectionnée, calculer sa distance minimale à S ,
 - construire une RCL des RCL_size villes les plus mal desservies,
 - sélectionner aléatoirement un centre dans la RCL,
4. Construction du vecteur binaire et calcul du rayon maximal.

Sortie : (indices des centres, vecteur binaire).

Listing 8 – P centre greedy random

```

function P_centre_greedy_random(filename, P; RCL_size=3)
    ...

    n = Lit_fichier_Placement(filename, tabX, tabY, f)

    # Initialisation avec un centre alatoire
    centre_initial = rand(1:n)
    S = [centre_initial]

    while length(S) < P
        dist_min = Dict{Int, Float64}()

        for j in setdiff(1:n, S)
            dist_min[j] = minimum(d[j, k] for k in S)
        end

        # Selection des RCL_size villes les plus mal desservies
        pires = sort(collect(keys(dist_min)), by = j -> -dist_min[j])
        RCL = pires[1:min(RCL_size, length(pires))]

        # Choix alatoire dans la liste restreinte
        choisi = rand(RCL)
        push!(S, choisi)
    end
    solution = zeros(Int, n)
    for j in S
        solution[j] = 1
    end
    return S, solution
end

```

3.6 P_centre_choix_villes_eloignées

Entrées : filename, P .

Étapes clés :

1. Lecture et calcul de la matrice d ,
2. Sélection des deux villes les plus éloignées pour initier S ,
3. Tant que $|S| < P$:
 - pour chaque ville non sélectionnée, calculer sa distance minimale au set S ,
 - ajouter la ville qui maximise cette distance,
4. Construction du vecteur binaire et calcul du rayon final.

Sortie : (indices des centres, vecteur binaire).

Listing 9 – $P_{centregreedy}$

```

function P_centre_choix_villes_eloignees(filename , P)
    ...
    # choisir les 2 villes les plus éloignées
    max_dist = -1.0
    a, b = 1, 2
    for i in 1:n
        for j in i+1:n
            if d[i,j] > max_dist
                max_dist = d[i,j]
                a, b = i, j
            end
        end
    end
    S = [a, b]
    while length(S) < P
        # Trouver la ville la plus éloignée des centres actuels
        max_min_dist = -1.0
        meilleur = -1
        for j in setdiff(1:n, S)
            dist_min = minimum(d[j, k] for k in S)
            if dist_min > max_min_dist
                max_min_dist = dist_min
                meilleur = j
            end
        end
        push!(S, meilleur)
    end

    solution = zeros(Int , n)
    for j in S
        solution[j] = 1
    end
    return S,solution
end

```

3.7 P_centre_greedy_metaheuristique

Entrées : filename, P , maxIter. **Étapes clés :**

1. Solution initiale par P_centre_greedy_random,
2. Boucle météahuristique :
 - relance aléatoire si stagnation,
 - sinon, échange aléatoire d'un centre avec une ville non-centre et calcul du nouveau rayon,
 - acceptation si le rayon diminue, sinon compter une itération sans amélioration.

Sortie : (indices des centres, vecteur binaire).

Listing 10 – $P_centre_{greedy_meta}$ heuristique

```

function P_centre_greedy_meta(filename, P, maxIter)
    ...

    # Tirage alatoire de P centres diff rents via une
    # heuristique existante
    centres, S = P_centre_greedy_random(filename, P)

    max_min_dist = maximum([minimum(d[i, j] for j in centres) for i
                           in 1:n])
    nbIter = 1
    nb_stable = 0

    while nbIter <= maxIter
        if nb_stable > (1/5)*maxIter
            centresT, STemp = P_centre_greedy_random(filename, P)
            max_min_distTemp = maximum([minimum(d[i, j] for j in
                                                 centresT) for i in 1:n])
            if max_min_distTemp < max_min_dist
                centres = centresT
                S = STemp
                max_min_dist = max_min_distTemp
                nb_stable = 0
            end
        else
            centres_copy = copy(centres)
            nouvelle_ville = rand(setdiff(1:n, centres))
            to_be_replaced = rand(1:length(centres))
            centres_copy[to_be_replaced] = nouvelle_ville
            S_copy = zeros(Int, n)
            for j in centres_copy
                S_copy[j] = 1
            end
            max_min_distTemp = maximum([minimum(d[i, j] for j in
                                                 centres_copy) for i in 1:n])
            if max_min_distTemp < max_min_dist
                centres = centres_copy
                S = S_copy
                max_min_dist = max_min_distTemp
                nb_stable = 0
            else
                nb_stable += 1
            end
        end
        nbIter += 1
    end

    return centres, S
end

```

3.8 P _ centre _ descente _ stoch _ iteree

Entrées : filename, P, maxIter, nbLancements.

Étapes clés :

1. Lecture des données (positions, coûts, etc.),
2. Boucle sur nbLancements essais :
 - Application de la métaheuristique P_centre_greedy_meta à chaque essai,
 - Calcul du rayon maximal pour la solution trouvée,
 - Conservation de la meilleure solution sur l'ensemble des essais.
3. Retour de la meilleure solution (centres et affectation) trouvée sur toutes les descentes stochastiques.

Sortie : (indices des centres de la meilleure solution, vecteur binaire d'affectation).

Listing 11 – P_centre_descente_stoch_iteree

```
function P_centre_descente_stoch_iteree(filename, P, maxIter,
                                         nbLancements)
    n = 0
    tabX = Float64[]
    tabY = Float64[]
    f = Float64[]
    # Lecture des données depuis le fichier
    n = Lit_fichier_Placement(filename, tabX, tabY, f)

    min_dist = 10e10
    best_centres = []
    best_S = []

    for essai in 1:nbLancements
        centres, S = P_centre_greedy_meta(filename, P, maxIter)
        min_dist_loc = calcul_max_distance(centres, tabX, tabY, n)
        # Meilleure solution trouvée
        if min_dist_loc < min_dist
            min_dist = min_dist_loc
            best_centres = centres
            best_S = S
        end
    end

    return best_centres, best_S
end
```

4 Calcul de l'écart relatif

Nous évaluons la qualité de chaque heuristique sur un échantillon de 900 villes (fichier inst_10000.flp) à l'aide de l'écart relatif (ou *gap*) défini par :

$$\text{gap} = \frac{z_{\text{heur}} - z_{\text{LP}}}{z_{\text{heur}}} \times 100\% ,$$

où z_{heur} est la valeur objective de la solution heuristique et z_{LP} la borne inférieure issue de la relaxation linéaire du PLNE. Un gap plus faible traduit une solution plus proche de l'optimum.

4.1 P-médian

Pour le problème des P-médian, nous avons comparé quatre méthodes :

(a) **Glouton aléatoire**

```
Valeur optimale = 39.52501997011254
Gap : 0.5005155201766553
```

FIGURE 1 – Gap % — P-médian, glouton aléatoire

«La méthode glouton aléatoire présente un gap de l'ordre de 50%, ce qui signifie que sa solution est au plus 50% au-dessus de l'optimum.»

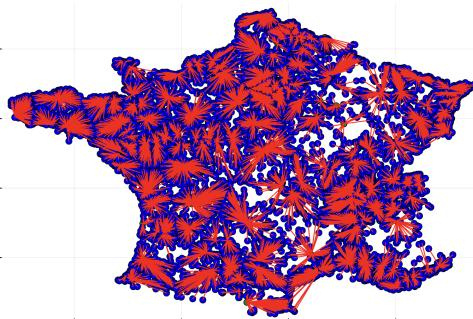


FIGURE 2 – Map % — P-médian, glouton aléatoire

(b) **Glouton par insertion**

```
Valeur optimale = 39.52501997011254
Gap : 0.08088576123783056
```

FIGURE 3 – Gap % — P-médian, glouton insertion

«Avec l'insertion gloutonne, le gap moyen est de 8%, montrant une amélioration

d'environ 42% par rapport au glouton aléatoire.»

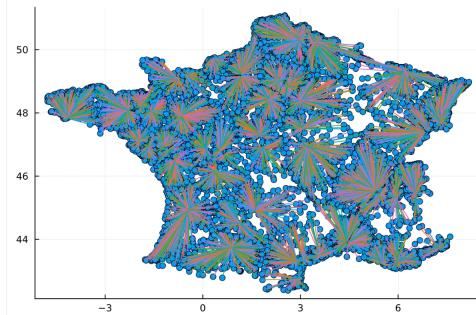


FIGURE 4 – Map % — P-médian, glouton insertion

Limite pratique : jusqu'à 7 000 villes en moins de 5 minutes.

(c) Métaheuristique (échange aléatoire)

```
Valeur optimale = 39.52501997011254
Gap : 0.10809896481661943
```

FIGURE 5 – Gap % — P-médian, météahuristique

«La météahuristique réduit le gap à 11%, démontrant la capacité des échanges itératifs à se rapprocher du PLNE.»

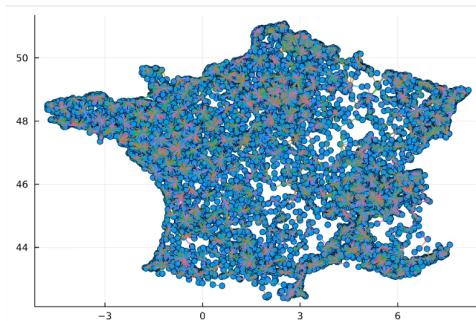


FIGURE 6 – Map % — P-médian, météahuristique

Limite pratique : jusqu'à 36 000 villes en moins de 5 minutes dans notre configuration.

(d) Métaheuristique + descente stochastique

«La descente stochastique atteint le gap le plus faible (10.5%), confirmant son efficacité pour peaufiner la solution.»

Valeur optimale = 39.52501997011254
 Gap : 0.10497164920079467

FIGURE 7 – Gap % — P-médian, descente stochastique

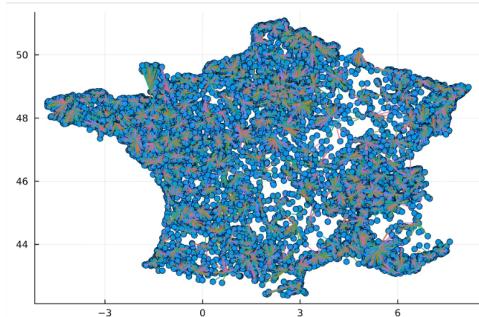


FIGURE 8 – Map % — P-médian, descente stochastique

Limite pratique. jusqu'à 36 000 villes, bien qu'elle nécessite un temps de calcul légèrement supérieur en raison de ses multiples relances (mais reste dans la limite des 5 minutes).

4.2 P-centre

Pour le problème des P-centre, les quatre méthodes étudiées sont :

- (a) Glouton aléatoire

Valeur optimale = 1.2405226501761264
 Gap : 0.2770928587793126

FIGURE 9 – Gap % — P-centre, glouton aléatoire

«Le glouton aléatoire affiche un gap de 27%, reflétant la variabilité de cette méthode simple.»

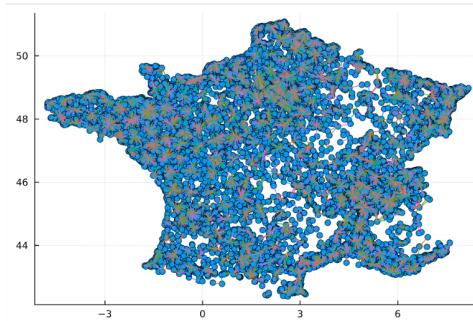


FIGURE 10 – Map % — P-centre, glouton aléatoire

(b) Choix des villes éloignées

```
Valeur optimale = 1.2405226501761264
Gap : 0.1325017172372409
```

FIGURE 11 – Gap % — P-centre, choix de villes éloignées

«En privilégiant les villes les plus éloignées, le gap obtenu est de 13%, indiquant une optimisation plus importante de la couverture maximale par rapport à la méthode aléatoire.»

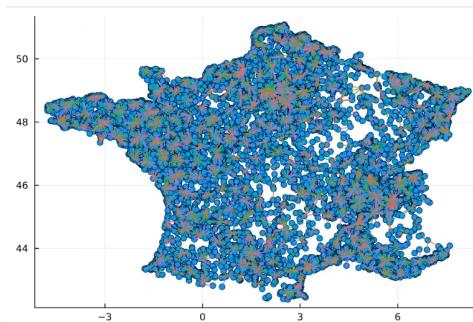


FIGURE 12 – Map % — P-centre, Choix des villes éloignées

Limite pratique : jusqu'à 36 000 villes en moins de 5 minutes dans notre configuration.

(c) Métaheuristique (échange aléatoire)

```
Valeur optimale = 1.2405226501761264
Gap : 0.08497774753003633
```

FIGURE 13 – Gap % — P-centre, métahéuristique

«La météahuristique ramène le gap à une valeur plus petite d'environ 8%, validant l'intérêt des échanges itératifs.»

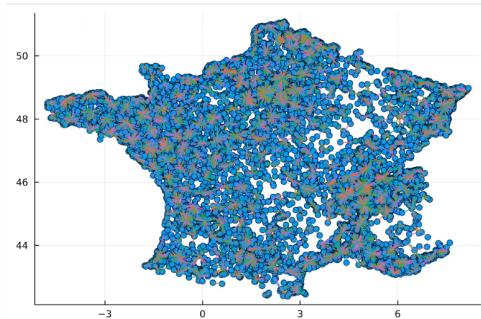


FIGURE 14 – Map % — P-centre, météahuristique

Limite pratique : jusqu'à 16 000 villes en moins de 5 minutes.

(d) **Métaheuristique + descente stochastique**

```
Valeur optimale = 1.2405226501761264
Gap : 0.03989076007905548
```

FIGURE 15 – Gap % — P-centre, descente stochastique

«La descente stochastique fournit le gap le plus compétitif (4%), montrant qu'elle affine au mieux la solution.»

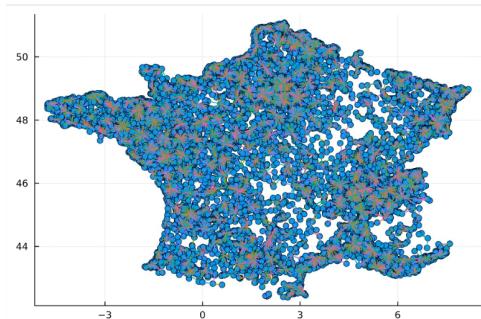


FIGURE 16 – Map % — P-centre, descente stochastique

Limite pratique : jusqu'à 3 500 villes en moins de 5 minutes dans notre configuration (on multiplie le temps par 10 par rapport à la météahuristique).

5 Scénario opérationnel et économique

Dans cette étude, nous considérons un cas réaliste où une entreprise souhaite desservir un réseau de **139 villes** réparties sur le territoire, en installant jusqu'à **15 centres logistiques**. L'objectif est de comparer différentes stratégies d'implantation des centres selon deux modèles d'optimisation (P-médian et P-centre), en tenant compte de coûts économiques concrets.

5.1 Mise à l'échelle des distances

Pour se rapprocher de distances réalistes, **toutes les distances calculées entre les villes ont été multipliées par 10**. Ainsi, chaque unité de distance dans le modèle correspond effectivement à 10 kilomètres sur le terrain. Cela permet d'obtenir une évaluation cohérente des coûts de transport en euros.

5.2 Coût du transport

Le coût de transport est estimé à partir des distances parcourues par les véhicules, selon l'hypothèse suivante :

- Consommation moyenne des véhicules : **7 L / 100 km** (soit 0,07 L/km),
- Prix moyen de l'essence : **1,54 € / L**,

d'où un coût kilométrique de :

$$0,07 \text{ L/km} \times 1,54 \text{ €/L} = 0,1078 \text{ €/km}$$

Le coût total de transport est donc égal à la somme des distances (en km) multipliée par ce coefficient.

5.3 Coût de construction des centres

À chaque centre ouvert dans la ville j , on associe un coût fixe d'installation noté f_j . Pour la comparaison, le coût de construction est pris égal à :

$$\text{Coût d'installation} = f_j \times 10\,000$$

où f_j provient des données du problème (par exemple, une estimation de foncier ou de travaux spécifiques à chaque ville).

5.4 Coût global

Le coût total d'une solution est donc la somme du coût de transport (proportionnel à la distance) et du coût de construction des centres :

$$\text{Coût total} = \left(\sum_{\text{villes}} \text{distance au centre le plus proche} \right) \times 0,1078 + \sum_{\text{centres ouverts } j} f_j \times 10\,000$$

Ce cadre permet d'évaluer objectivement la performance des solutions produites par chaque modèle (P-médian ou P-centre) dans une perspective opérationnelle et économique.

6 Comparaison expérimentale : P-médian vs P-centre

6.1 Présentation des résultats

Pour comparer objectivement les deux modèles, nous avons évalué sur le même scénario les solutions optimales produites par le P-médian et le P-centre, en calculant à chaque fois :

- Le coût total d'installation des centres,
- Le coût total du transport (somme des distances, convertie en euros selon le scénario précédent),
- Le coût global (installation + transport).

6.2 Résultats pour le P-médian

```
-----  
Total (root+branch&cut) = 0.49 sec. (239.18 ticks)  
Valeur optimale = 51.02304652347393  
Coût total transport : 110.01 €  
Coût total installation : 3.092988e6 €
```

FIGURE 17 – Résultats pour le modèle P-médian (exemple de sortie console)

On observe que :

- **Coût d'installation des centres** : environ 3 093 000 €
- **Coût total de transport** : environ 110 €

6.3 Résultats pour le P-centre

```
-----  
Total (root+branch&cut) = 40.13 sec. (2696.02 ticks)  
Valeur optimale = 1.5206906325745548  
Coût total transport : 207.01 €  
Coût total installation : 3.016202e6 €
```

FIGURE 18 – Résultats pour le modèle P-centre (exemple de sortie console)

On observe que :

- **Coût d'installation des centres** : environ 3 016 000 €
- **Coût total de transport** : environ 207 €

6.4 Analyse critique

Les deux modèles conduisent à l'ouverture d'un nombre comparable de centres, pour un coût d'installation quasiment équivalent (proche de 3 millions d'euros). En revanche, le coût du transport est nettement **plus faible avec le P-médian** (~110 € contre ~207 €

pour le P-centre), car le modèle P-médian est conçu pour optimiser la somme totale des distances, ce qui minimise naturellement les dépenses liées aux trajets de livraison.

En contrepartie, le modèle **P-centre** permet de garantir que **toutes les villes sont desservies dans un rayon limité**, offrant ainsi une meilleure équité de service et probablement des délais plus courts pour les villes les plus éloignées. Ce choix de confort et de qualité de service se fait toutefois au détriment du coût total de transport.

En résumé :

- Si l'objectif principal est la maîtrise du coût global, le modèle **P-médian** est le plus pertinent.
- Si l'objectif prioritaire est la rapidité et l'équité du service (personne n'est trop loin d'un centre), alors le **P-centre** s'avère préférable, quitte à supporter un coût logistique plus élevé.

Cette analyse met en avant les arbitrages classiques du monde réel entre performance économique (P-médian) et qualité de service (P-centre).

6.5 P-médian avec coût d'installation (PLNE avec f_j)

Nous avons également testé la version du PLNE P-médian intégrant les coûts fixes d'installation (f_j) pour chaque centre, en plus du coût de transport. Les résultats sont synthétisés ci-dessous :

```
-----  
Total (root+branch&cut) = 0.58 sec. (324.19 ticks)  
Valeur optimale = 193.38547085149727  
Coût total transport : 259.98 €  
Coût total installation : 728001.0 €
```

FIGURE 19 – Résultats pour le modèle P-médian avec coûts d'installation (exemple de sortie console)

On constate que :

- **Coût d'installation des centres** : **728 001 €**, soit nettement inférieur aux autres modèles testés jusqu'ici,
- **Coût total de transport** : **259,98 €**, donc **plus élevé** qu'avec le P-médian classique ou P-centre,

Ce résultat s'explique par le fait que l'algorithme privilégie l'ouverture de centres moins coûteux à installer, même si cela entraîne des trajets un peu plus longs pour desservir les villes.

Projection économique sur 10 ans

Afin de comparer les trois scénarios sur le long terme, nous avons estimé le coût global sur 10 ans, en tenant compte de la récurrence des trajets :

- Livraison prévue **3 fois par semaine** vers chaque ville,
- Donc : $3 \times 52 = 156$ livraisons par an et par ville,

— Sur 10 ans : 1560 livraisons par ville.

Pour chaque modèle, on applique la formule suivante :

$$\text{Coût transport (10 ans)} = \text{coût transport initial} \times 1560$$

et

$$\text{Coût global (10 ans)} = \text{coût installation} + \text{coût transport (10 ans)}$$

Par exemple, pour le P-médian avec coût d'installation variable :

$$259,98 \times 1560 = 405\,568,8$$

et

$$728\,001 + 405\,569 = 1\,133\,570$$

Pour mieux visualiser la comparaison, voici un tableau récapitulatif des trois modèles :

Modèle	Coût installation	Coût transport	Transport (10 ans)	Coût global (10 ans)
P-médian classique	3 092 988 €	110,01 €	171 615 €	3 264 603 €
P-centre	3 016 020 €	207,01 €	323 936 €	3 339 956 €
P-médian avec f_j	728 001 €	259,98 €	405 569 €	1 133 570 €

TABLE 1 – Comparaison des coûts projetés sur 10 ans pour chaque modèle (15 centres, 139 villes, 3 livraisons/ville/semaine)

On constate que le modèle P-médian classique reste le plus économique sur 10 ans si l'on ne tient pas compte des coûts d'installation variables, grâce à son optimisation globale des trajets. Le P-centre, en cherchant à limiter la distance maximale, engendre des coûts de transport nettement supérieurs sur la durée.

En revanche, le P-médian avec coûts d'installation (f_j) privilégie l'ouverture de centres moins onéreux, ce qui réduit considérablement le coût d'installation initial, mais allonge les trajets et donc augmente le coût de transport récurrent. Toutefois, sur 10 ans, le modèle P-médian avec coût d'installation variable demeure de loin le plus économique, même si le confort de desserte est moindre.

Ce scénario met en lumière l'importance de bien équilibrer coût initial d'installation et coûts récurrents de transport lors de la conception d'un réseau logistique, et démontre que la stratégie optimale dépend largement de l'horizon temporel et des priorités opérationnelles de l'entreprise.