



TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

TIPE 2025

Problème du transport optimal :
Au cœur de l'animation météorologique

Candidat n°17073



Introduction

Le problème en lui-même

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- Le problème de Monge-Kantorovich :



(a) Gaspard Monge
(1746-1818)



(b) Leonid Kantorovich
(1912-1968)



Introduction

Le problème en lui-même

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

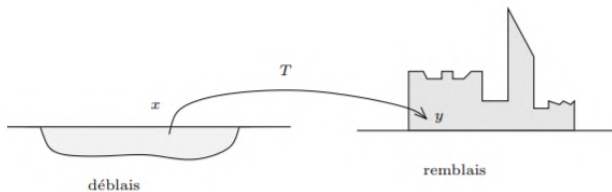


Figure 2: Exemple du déblais/remblais.



Introduction

Mon objectif

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe



(a)



(b)

Figure 3: (TF1)

Introduction

Mon objectif



TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

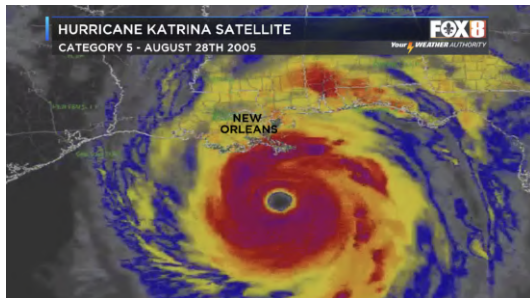


Figure 4: Application à la météorologie. (Fox 8)



Introduction

Le problème des livreurs/boulangeries

TIPE 2025

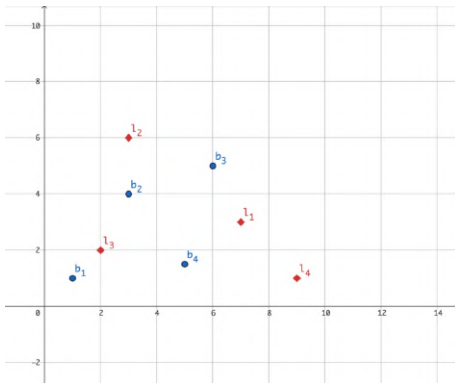
Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- **Contexte** : n livreurs $\mathcal{L} = \{l_1, \dots, l_n\}$ et n boulangeries $\mathcal{B} = \{b_1, \dots, b_n\}$ des points fixes dans le plan euclidien $E = \mathbb{R}^2$.
- *Exemple* :





Introduction

Le problème des livreurs/boulangeries

TIPE 2025

Introduction :
le cas discret

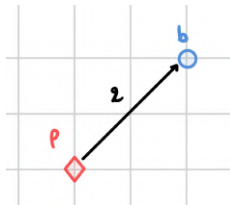
Le cas continu

Méthode
Gloutonne

Annexe

- **Fonction de coût** : On considère une fonction de coût

$$c : E \times E \rightarrow \mathbb{R}_+$$





Introduction

Le problème des livreurs/boulangeries

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- **Plan de transport** : Un plan de transport est une bijection

$$T : \mathcal{L} \rightarrow \mathcal{B}$$

On notera $\mathcal{T}$ l'ensemble des plans de transport.

$$T_{\text{ex}} : \left\{ \begin{array}{lcl} l_1 & \mapsto & b_3 \\ l_2 & \mapsto & b_2 \\ l_3 & \mapsto & b_1 \\ l_4 & \mapsto & b_4 \end{array} \right.$$



Introduction

Le problème des livreurs/boulangeries

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- **Objectif** : on désire trouver le *plan de transport optimal*, i.e le plan qui minimise le coût total :

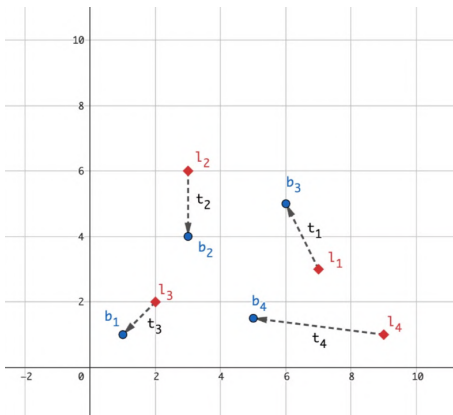
$$T_{\min} = \arg \min_{T \in \mathcal{T}} \sum_{l \in \mathcal{L}} c(l, T(l))$$



Introduction

Le problème des livreurs/boulangeries

- *Résolution de l'exemple* : on prend c la distance euclidienne usuelle, on a alors la solution suivante :





Introduction

L'importance de la fonction de coût : Cas de la dimension 1

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Maintenant, $E = \mathbb{R}$,

$\mathcal{L}$ est un ensemble de n livres collés entre eux.

$$\mathcal{B} = \{l + 1 \mid l \in \mathcal{L}\}$$



Deux solutions *optimales* se présentent :

- n petits déplacements
- 1 grand déplacement



Introduction

L'importance de la fonction de coût : déplacement de livres

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- Pour c la distance euclidienne,

$$c : (x, y) \mapsto \|x - y\|$$

- Version petits déplacements, $:= T_{\text{petits}}$,

$$\sum_{l \in \mathcal{L}} c(l, T_{\text{petits}}(l)) = \sum_{l \in \mathcal{L}} 1 = n$$

- Version grand déplacement, $:= T_{\text{grand}}$,

$$\sum_{l \in \mathcal{L}} c(l, T_{\text{grand}}(l)) = n + \sum_{l \in \mathcal{L}, l \neq l_1} \underbrace{c(l, T_{\text{grand}}(l))}_{=0} = n$$

$\Rightarrow$ Les deux versions sont bien *optimales*.



Introduction

L'importance de la fonction de coût : déplacement de livres

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

- Pour c la distance euclidienne au carré,

$$c : (x, y) \mapsto \|x - y\|^2$$

- Version petits déplacements, $:= T_{\text{petits}}$,

$$\sum_{l \in \mathcal{L}} c(l, T_{\text{petits}}(l)) = \sum_{l \in \mathcal{L}} 1^2 = n$$

- Version grand déplacement, $:= T_{\text{grand}}$,

$$\sum_{l \in \mathcal{L}} c(l, T_{\text{grand}}(l)) = n^2 + \sum_{l \in \mathcal{L}, l \neq l_1} \underbrace{c(l, T_{\text{grand}}(l))}_{=0} = n^2$$

$\Rightarrow$ La version 'petits déplacements' est la seule *optimale*.



Introduction

L'importance de la fonction de coût : caractère convexe

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Utilité de la norme quadratique : Théorie de Brenier

- Unicité de la solution ;
- Régularité de la solution ;
- Usage du gradient ;

Le cas continu

Retour à l'animation météorologique



TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

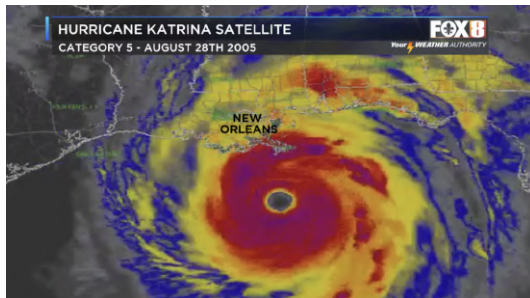


Figure 5: Ouragan Katrina, Fox 8



Le cas continu

Transport optimal entre deux espaces dans le cas continu

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Soient μ, ν deux mesures de probabilité sur $E = \mathbb{R}^2$, de même masse totale.

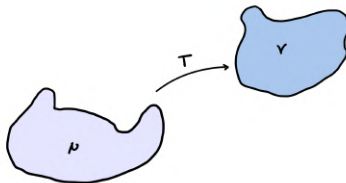
Soit c une fonction de coût *mesurable*.

On veut trouver une *application de transport* $T : E \rightarrow E$ qui :

- **Transfère μ sur ν :**

$$T_{\#}\mu = \nu$$

notation *push-forward* : $T_{\#}\mu$ est l'image directe de μ par T .





- Minimise le coût du transport total :

$$\inf_{\substack{T \in \mathcal{T} \\ T_{\#} \mu = \nu}} \int_E c(x, T(x)) d\mu(x)$$



Le cas continu

La théorie de Brenier

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Hypothèses supplémentaires : μ doit être *absolument continue* (par rapport à la mesure de Lebesgue), et c est le coût quadratique :

On prendra ici $c : (x, y) \mapsto \frac{1}{2} \|x - y\|^2$

Théorie de Brenier

Il existe une application de transport optimale $T : E \rightarrow E$ qui répond au problème de Monge et cette application est de la forme :

$$T(x) = \vec{\nabla} \varphi(x)$$

où $\varphi : E \rightarrow \mathbb{R}$ est une fonction convexe.



Méthode gloutonne

Fonction de coût

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Conformément à la théorie de Brenier, on prendra

$$c : (x, y) \mapsto \frac{1}{2} \|x - y\|^2 = \frac{1}{2} (x - y)^T (x - y)$$

Proposition 3

c est convexe.



Méthode gloutonne

Trouver le φ convexe : descente de gradient

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Proposition 4 - Gradient opposé d'une fonction

Soient $(E, \langle, \rangle)$, $U \subset E$ un ouvert, $a \in U$, $f : U \rightarrow \mathbb{R}$ une fonction et différentiable en a , alors :

$-\vec{\nabla} f(a)$ donne la direction selon laquelle f décroît le plus rapidement en a .

$\Rightarrow$ Pourrait-on alors simplement s'intéresser au coût quadratique c ?



Méthode gloutonne

Représentation d'un nuage

TIPE 2025

Prenons l'exemple d'un cercle décrit par P points :

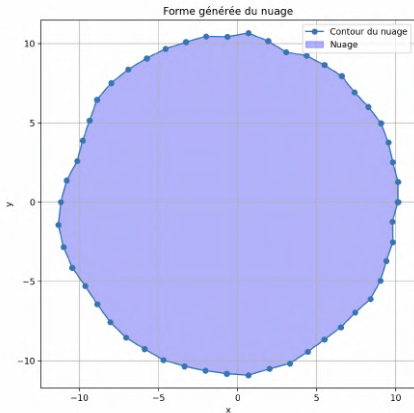


Figure 6: Nuage en python décrit avec une liste de coordonnées (x, y) .



Méthode gloutonne

Trouver le φ convexe : exemple du cercle

TIPE 2025

Introduction :
le cas discret

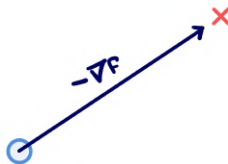
Le cas continu

Méthode
Gloutonne

Annexe



(a) Point source et point cible.



(b) Transition la moins coûteuse.

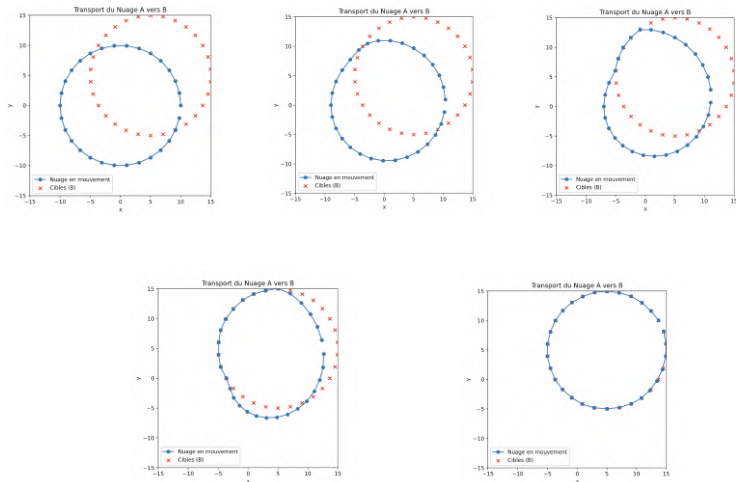
on applique alors la descente de gradient aux P points.



Méthode gloutonne

Trouver le φ convexe : exemple du cercle

On a alors la transition suivante :



TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe



Méthode gloutonne

Trouver le φ convexe : prendre en compte les obstacles

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

De manière empirique, on peut déduire des zones sur une carte qu'un cumulus pourrait difficilement traverser (zone montagneuse) ou au contraire très facilement (zone de grands vents) :



Figure 10: Nuage accroché au Mont Fuji au Japon.



Méthode gloutonne

Trouver le φ convexe : prendre en compte les obstacles

TIPE 2025

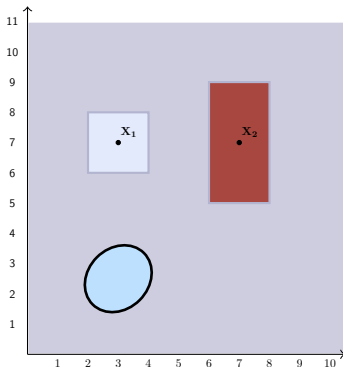
Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

On va alors simplement cartographier les zone *difficiles/faciles* en modifiant le coût :



$$c(x, y) = \frac{1}{2} \|x - y\|^2 + \sum_i v_i \|y - X_i\|^2$$

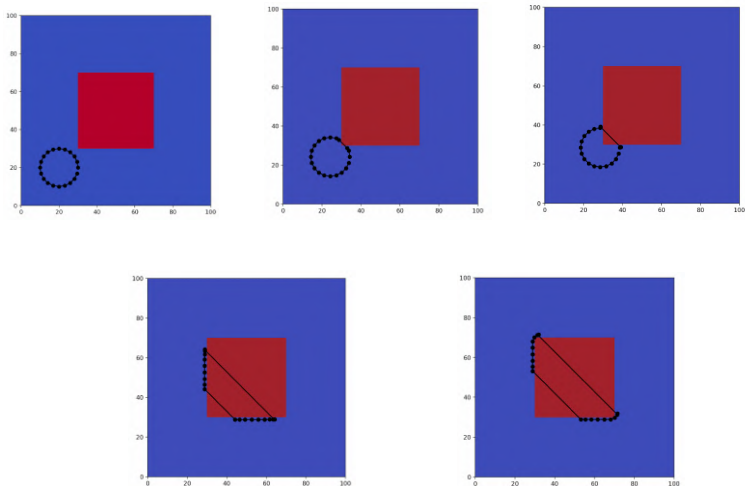


Méthode gloutonne

Premier test : problème de séparation du nuage

TIPE 2025

On a le premier test suivant :





Méthode gloutonne

Premier test : problème de séparation du nuage

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Correction du problème : calculer l'écart moyen entre chaque points voisins et faire en sorte qu'il n'y ait aucun écart significativement plus grand que la moyenne.

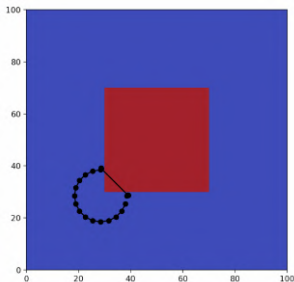


Figure 13: Exemple de situation à éviter.



Méthode gloutonne

Deuxième test : problème de densité

TIPE 2025

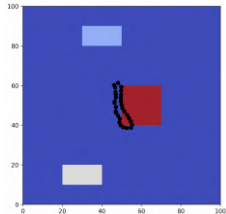
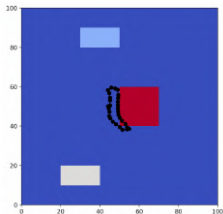
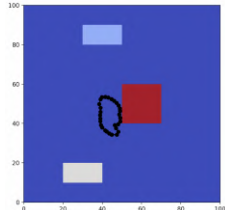
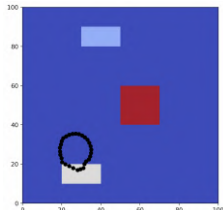
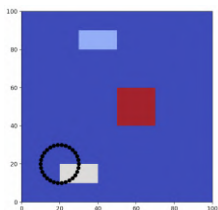
Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

On a le second test suivant :





Méthode gloutonne

Deuxième test : problème de densité

TIPE 2025

Introduction :
le cas discret

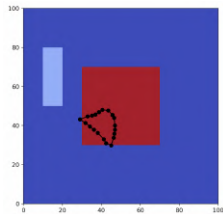
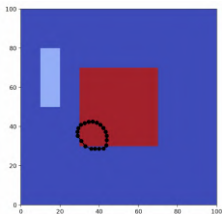
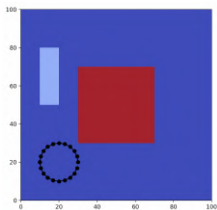
Le cas continu

Méthode
Gloutonne

Annexe

Correction du problème : Calculer la densité initiale d_0 et l'écart autorisé $\varepsilon_d > 0$ à cette densité pour qu'à chaque instant de la transition :

$$d \in [d_0 - \varepsilon_d, d_0 + \varepsilon_d]$$





Méthode gloutonne

Troisième test : dans des conditions plus compliquées

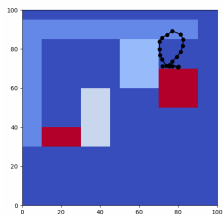
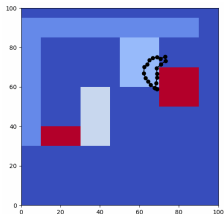
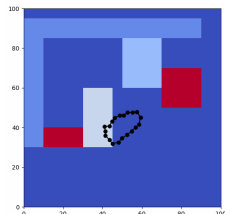
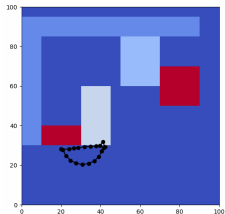
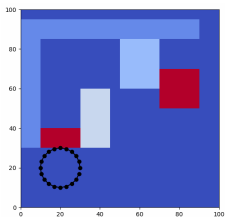
TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe



Méthode gloutonne

Conservation de la masse

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Conservation de la masse entre une forme A et une forme B du nuage :

$$\int_E \mu(x) dx = \boxed{\int_A \mu(x) dx = 1 = \int_B \nu(x) dx} = \int_E \nu(x) dx$$

Hypothèse forte : répartition uniforme de la matière¹,

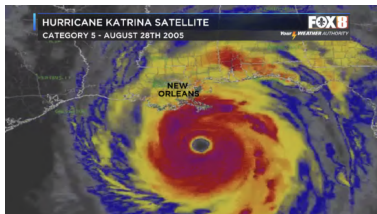


Figure 19: Ouragan Katrina, Fox 8

¹cas d'une sous-surface



Méthode gloutonne

Conservation de la masse

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Si $x \in A$,

$$\mu(x) = \text{cte} = \frac{1}{\mathcal{A}(A)}$$

où $\mathcal{A}(A)$ est l'aire décrite par A .

Sinon,

$$\mu(x) = \text{cte} = 0$$

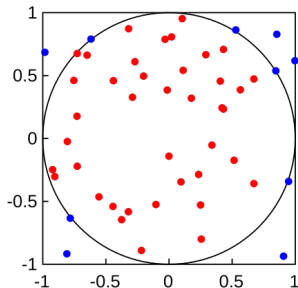


Méthode gloutonne

Calcul de l'aire d'un nuage : méthode de Monte-Carlo

TIPE 2025

On utilise pour cela la méthode de *Monte Carlo* :



À l'aide d'une loi uniforme, on lance N points sur une surface S comprenant notre nuage, on récupère alors le pourcentage p de points dans le nuage, l'aire du nuage est donc approximativement de $p \times S$.



Méthode gloutonne

Calcul de l'aire d'un nuage : méthode de Monte-Carlo

TIPE 2025

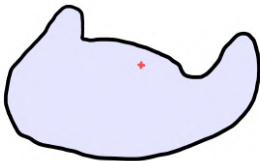
Introduction :
le cas discret

Le cas continu

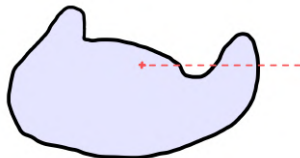
Méthode
Gloutonne

Annexe

Appartenance au nuage :



(a) Point dans le nuage.



(b) Calcul d'appartenance.

⇒ Calcul en temps constant : $\mathcal{O}(1)$.



Méthode gloutonne

Calcul de l'aire d'un nuage : méthode de Monte-Carlo

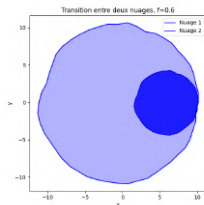
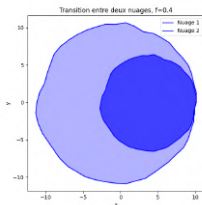
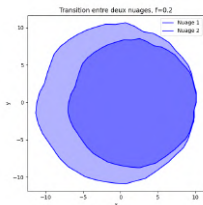
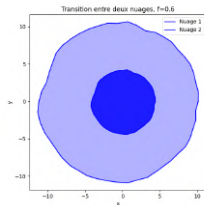
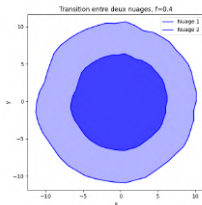
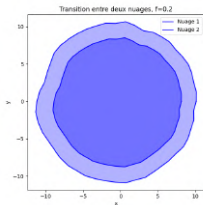
TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe





Méthode gloutonne

Calcul de l'aire d'un nuage : méthode de Monte-Carlo

TIPE 2025

Complexité :

$$\mathcal{O}(N)$$

avec N le nombre de points utilisés.

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

Pour le nuage de la figure 6, avec 3000 points :

Aire estimée (m^2)	Temps requis (s)
289.06	0.0816
289.58	0.0799
288.61	0.0783
289.35	0.0781
289.22	0.0780

⇒ Moins d'un dixième de seconde pour calculer l'aire.



Méthode gloutonne

Conclusion

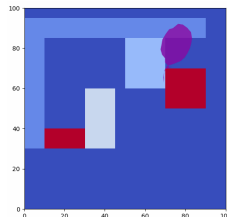
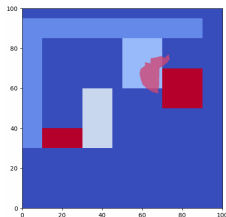
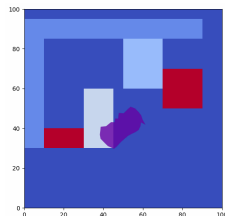
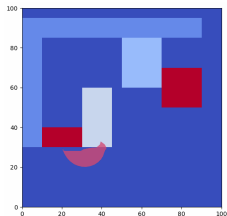
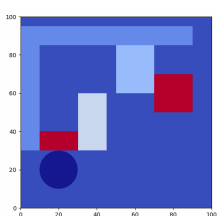
TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe





Extrait du code

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from scipy.ndimage import map_coordinates

# Creation d'une carte des couts (ex : 100x100 avec differentes zones)
def generer_carte_cout(taille=100):
    carte = np.ones((taille, taille)) # Fond facile (valeur 1)

    # Rectangle central - difficulte moyenne
    carte[30:45, 30:60] = 4

    # Rectangle easy - difficulte legere
    carte[0:10,30:95] = 2
    carte[10:90, 85:95] = 2

    # Rectangle en bas a gauche - difficulte elevee
    carte[10:30, 30:40] = 8
    carte[70:90,50:70]=8

    # Rectangle difficulte moyenne
    carte[50:70,60:85]=3

    return carte
```



Extrait du code

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

```
# Generer un nuage de points dans l'espace
def generer_nuage(nb_points, centre=(50, 50), rayon=10):
    angles = np.linspace(0, 2 * np.pi, nb_points, endpoint=False)
    points = [(centre[0] + rayon * np.cos(a), centre[1] + rayon * np.sin(a))
               for a in angles]
    return np.array(points)

# Fonction de cout en fonction de la carte des couts
def fonction_cout(point, carte):
    x, y = point
    return map_coordinates(carte, [[x], [y]], order=1)[0]

# Calcul du gradient du cout
def calculer_gradient(point, carte, epsilon=1.0):
    x, y = point
    grad_x = (fonction_cout((x + epsilon, y), carte) - fonction_cout((x -
        epsilon, y), carte)) / (2 * epsilon)
    grad_y = (fonction_cout((x, y + epsilon), carte) - fonction_cout((x, y -
        epsilon), carte)) / (2 * epsilon)
    return np.array([grad_x, grad_y])
```



Extrait du code

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

```
# Moyenner les ecarts
def regulariser_positions(points, facteur_seuil=1.5):

    # Ajuste les positions des points pour eviter qu'ils ne se dispersent
    trop.

    nouveaux_points = points.copy()
    nb_points = len(nouveaux_points)

    # Calcul des distances entre tous les points voisins
    distances = [np.linalg.norm(nouveaux_points[i] - nouveaux_points[i-1])
                 for i in range(nb_points)]
    distance_moyenne = np.mean(distances)

    for i in range(nb_points):
        j = (i + 1) % nb_points # Voisin suivant (avec boucle)
        d = np.linalg.norm(nouveaux_points[j] - nouveaux_points[i])

        if d > facteur_seuil * distance_moyenne: # Trop eloignes
            correction = (nouveaux_points[i] + nouveaux_points[j]) / 2 #
                           Moyenne entre les deux
            nouveaux_points[j] = correction # Ajustement

    return np.array(nouveaux_points)
```



Extrait du code

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

```
# Deplacement glouton en minimisant le cout total
def calculer_deplacement(A, B, carte, pas=1.0, force_repulsion=0.1,
    rayon_repulsion=10.0):
    nouveaux_A = []
    nb_points = len(A)
    for i, ((xA, yA), (xB, yB)) in enumerate(zip(A, B)):
        direction = np.array([xB - xA, yB - yA])
        direction /= np.linalg.norm(direction) + 1e-8 # Normalisation
        gradient = calculer_gradient((xA, yA), carte) # Gradient du cout
        direction -= 0.5 * gradient

        # Repulsion
        repulsion = np.zeros(2)
        for j, (xj, yj) in enumerate(A):
            if i != j:
                vecteur = np.array([xA - xj, yA - yj])
                distance = np.linalg.norm(vecteur)
                if 0 < distance < rayon_repulsion:
                    # Correction de la force de repulsion
                    repulsion += vecteur / (distance**2 + 1e-8)

        direction += force_repulsion * repulsion
        direction /= np.linalg.norm(direction) + 1e-8 # Renormalisation
        finale

    nouveaux_A.append((xA + pas * direction[0], yA + pas * direction[1])
    )

# On conserve la regularisation pour la cohesion
return regulariser_positions(np.array(nouveaux_A))
```



Extrait du code

TIPE 2025

Introduction :
le cas discret

Le cas continu

Méthode
Gloutonne

Annexe

```
# Simulation complete du transport
def simulation_transport(A, B, carte, iterations=100, pas=1.0):
    historique = [A]
    points = A
    for _ in range(iterations):
        points = calculer_deplacement(points, B, carte, pas)
        historique.append(points)
    return historique

# Parametres et generation des donnees
nb_points = 20
carte_cout = generer_carte_cout()
A = generer_nuage(nb_points, centre=(20, 20))
B = generer_nuage(nb_points, centre=(80, 80))
historique = simulation_transport(A, B, carte_cout)

# Animation
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(carte_cout.T, cmap='coolwarm', origin='lower', extent=[0, 100, 0, 100])
points_plot, = ax.plot([], [], 'o-', color='black')
ax.set_xlim(0, 100)
ax.set_ylim(0, 100)

plt.show()
```