

# TIPE

## Optimisation des transports

ROCHER Kilian — WILLEM Logan

2020 - 2021

## ① Situation du problème

Qu'est ce que l'optimisation des transports ?

Quelques exemples

## ② Problème de tournée des véhicules

Contexte

Variantes

## ③ Première résolution

L'algorithme de Clarke & Wright

Insuffisance de l'algorithme seul

## ④ Amélioration

Le 2-opt

Résultats avec 2-opt

Complexité des algorithmes

## Définition et enjeux de l'optimisation des transports

Quelques exemples de transport optimal avec images explicatives

Dans ce type de problèmes, il s'agit de minimiser le coût total (en distance par exemple) de la tournée de tous les véhicules, ayant pour objectif de livrer un nombre défini de clients.

## Différentes variantes du problème de tournée des véhicules

### — Classique (VRP)

## Différentes variantes du problème de tournée des véhicules

- Classique (VRP)
- Contrainte de capacité (CVRP)

## Différentes variantes du problème de tournée des véhicules

- Classique (VRP)
- Contrainte de capacité (CVRP)
- Dépôts multiples (MDVRP)



## Différentes variantes du problème de tournée des véhicules

- Classique (VRP)
- Contrainte de capacité (CVRP)
- Dépôts multiples (MDVRP)
- Retours des colis (VRPPD et VRPB)

## Différentes variantes du problème de tournée des véhicules

- Classique (VRP)
- Contrainte de capacité (CVRP)
- Dépôts multiples (MDVRP)
- Retours des colis (VRPPD et VRPB)
- ...

On s'est intéressé à la variante classique afin de s'approprier au mieux le problème.

## Données

- Un point  $D$  de coordonnées  $(0, 0)$  : le dépôt

## Données

- Un point  $D$  de coordonnées  $(0, 0)$  : le dépôt
- Une famille de points  $(i_1, \dots, i_k) \in [-100, 100]^2)^k$  pour un certain  $k \in [2, +\infty[$  représentant les clients

## Données

- Un point  $D$  de coordonnées  $(0, 0)$  : le dépôt
- Une famille de points  $(i_1, \dots, i_k) \in [-100, 100]^2)^k$  pour un certain  $k \in [2, +\infty[$  représentant les clients
- Une fonction  $d$  qui calcule la distance entre deux points.

## Données

- Un point  $D$  de coordonnées  $(0, 0)$  : le dépôt
- Une famille de points  $(i_1, \dots, i_k) \in ([-100, 100]^2)^k$  pour un certain  $k \in [2, +\infty[$  représentant les clients
- Une fonction  $d$  qui calcule la distance entre deux points.

Remarque : Dans cette méthode de résolution, le nombre de véhicules n'est pas fixé. C'est l'algorithme qui décide du nombre optimal de véhicules à utiliser.

## Definition (Fonction *gain*)

Fonction  $s$  : calcule le gain après raccord de deux routes.

Pour deux points  $i$  et  $j$ ,  $s$  calcule la différence de distance entre le chemin  $D - i - D + D - j - D$  qui vaut  $2d(D, i) + 2d(j, D)$  au chemin  $D - i - j - D$  de distance  $d(D, i) + d(i, j) + d(j, D)$

$$s(i, j) = 2d(D, i) + 2d(j, D) - [d(D, i) + d(i, j) + d(j, D)]$$

$$s(i, j) = d(D, i) + d(j, D) - d(i, j)$$

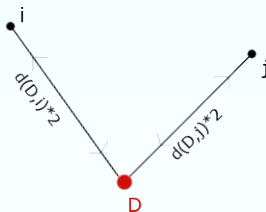
## Definition (Fonction *gain*)

Fonction  $s$  : calcule le gain après raccord de deux routes.

Pour deux points  $i$  et  $j$ ,  $s$  calcule la différence de distance entre le chemin  $D - i - D + D - j - D$  qui vaut  $2d(D, i) + 2d(j, D)$  au chemin  $D - i - j - D$  de distance  $d(D, i) + d(i, j) + d(j, D)$

$$s(i, j) = 2d(D, i) + 2d(j, D) - [d(D, i) + d(i, j) + d(j, D)]$$

$$s(i, j) = d(D, i) + d(j, D) - d(i, j)$$





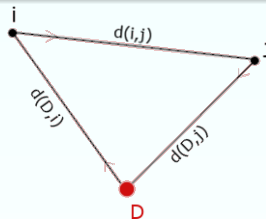
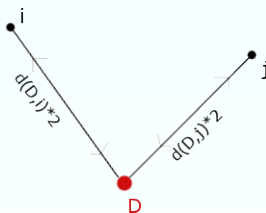
## Definition (Fonction *gain*)

Fonction  $s$  : calcule le gain après raccord de deux routes.

Pour deux points  $i$  et  $j$ ,  $s$  calcule la différence de distance entre le chemin  $D - i - D + D - j - D$  qui vaut  $2d(D, i) + 2d(j, D)$  au chemin  $D - i - j - D$  de distance  $d(D, i) + d(i, j) + d(j, D)$

$$s(i, j) = 2d(D, i) + 2d(j, D) - [d(D, i) + d(i, j) + d(j, D)]$$

$$s(i, j) = d(D, i) + d(j, D) - d(i, j)$$



## L'algorithme de Clarke &amp; Wright

- Étape 1 :
- Étape 2 :

## L'algorithme de Clarke &amp; Wright

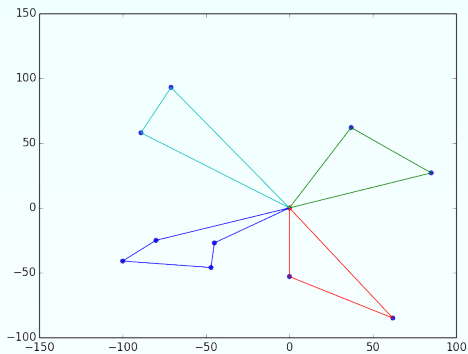


Figure – Résolution parfaite d'un problème

## Insuffisance de l'algorithme seul

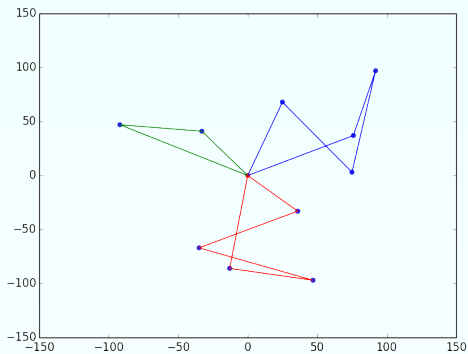


Figure – Résultat non satisfaisant : 10 clients

## Insuffisance de l'algorithme seul

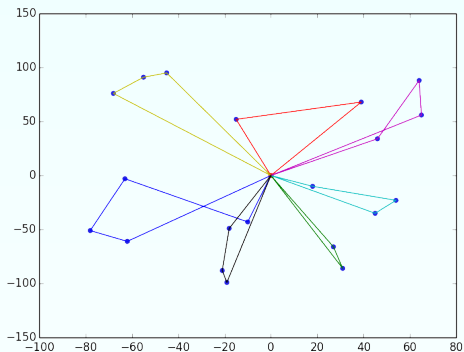
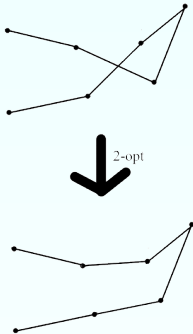


Figure – Résultat non satisfaisant : 20 clients



## Le 2-opt



Principe du 2-opt

Suppression des liaisons sécantes

```

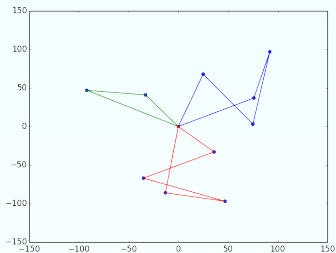
# 2-opt pour régler les problèmes de croisements
def deux_opt(routes/FINAL):
    continuer = True
    while continuer:
        continuer = False
        for route in routes:
            if len(route) > 4:
                for i in range(len(route)-1):
                    for j in range(len(route)-1):
                        if i != j and j != i-1 and j != i+1:
                            if route[i] == 0:
                                if distance(DEPOT, CLIENTS[route[i+1]-1]) + distance(CLIENTS[route[i]-1], CLIENTS[route[i+1]-1]) >
                                   distance(DEPOT, CLIENTS[route[i]-1]) + distance(CLIENTS[route[i+1]-1], CLIENTS[route[i+1]-1]):
                                    (route[i+1], route[j])
                                    ) = (route[j], route[i+1])
                                    continuer = True
                            elif route[j] == 0:
                                if distance(CLIENTS[route[i]-1], CLIENTS[route[i+1]-1]) + distance(DEPOT, CLIENTS[route[i+1]-1]) >
                                   distance(CLIENTS[route[i]-1], DEPOT) + distance(CLIENTS[route[i+1]-1], CLIENTS[route[i+1]-1]):
                                    (route[i], route[j])
                                    ) = (route[j], route[i])
                                    continuer = True
                            elif distance(CLIENTS[route[i]-1], CLIENTS[route[i+1]-1]) + distance(CLIENTS[route[j]-1], CLIENTS[route[i+1]-1]) >
                                   distance(CLIENTS[route[i]-1], CLIENTS[route[j]-1]) + distance(CLIENTS[route[i+1]-1], CLIENTS[route[i+1]-1]):
                                    (route[i+1], route[j]) = (route[j], route[i+1])
                                    continuer = True
    return routes

```

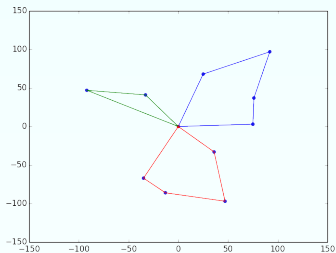
Vue générale de l'algorithme



## 10 clients

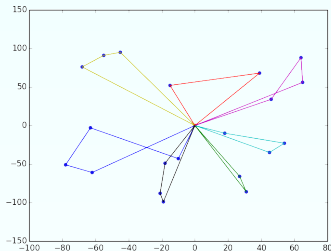


Avant

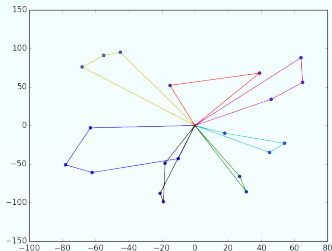


Après

## 20 clients



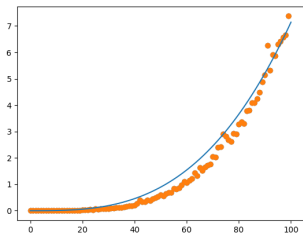
Distance : 1442km  
Avant



Distance : 1403km  
Après

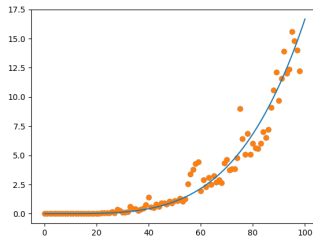


## Sans 2-opt



$$f(x) = x^3/14000$$

## Avec 2-opt



$$f(x) = x^4/6000000$$