CORDIEZ Théophile
KARCHER Victor

Projet 2A

Optimisation de tournées de véhicules : retraits et livraisons avec fenêtres horaires

Tuteur : Patrick Charpentier
12 Juin 2020

Plan

I - Présentation du problème

II - Construction d’une solution initiale

III - Amélioration de la solution

IV - Résultats expérimentaux

I - Présentation du problème

Introduction

Constat

- 25% des camions roulent à vide en France
- 50% ne sont qu'à demi-remplis

Problématique

- Sous-optimalité => pollution & surcoûts

Vers des solutions collaboratives

Poule avant l'oeuf

Avec qui collaborer si chacun privilégie son intérêt propre ?

Absence de transports collaboratifs

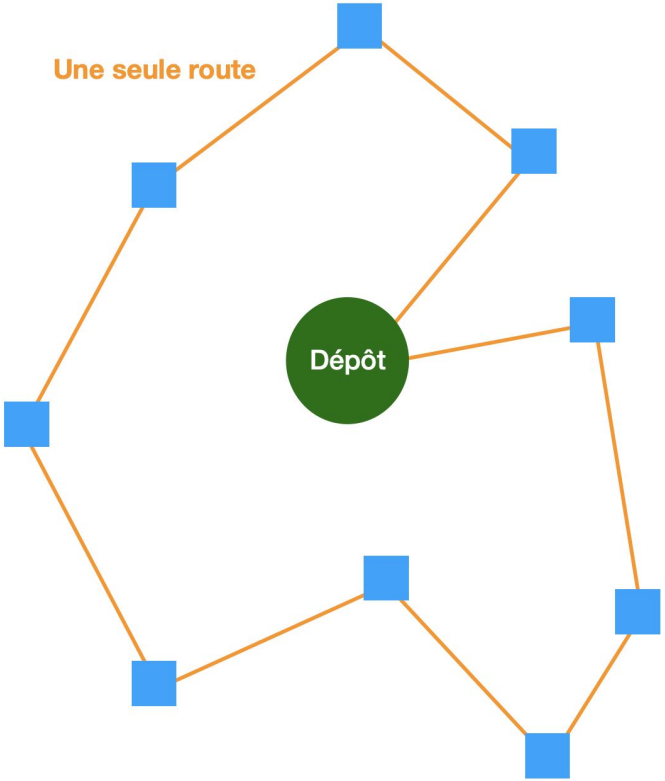
↓

Maximisation de l'utilité individuelle

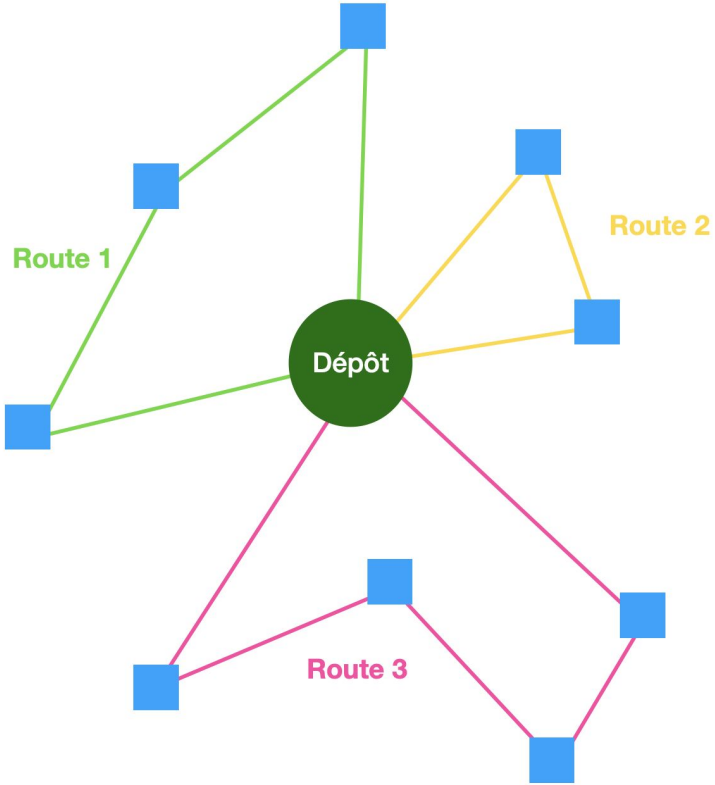
↓

Sous-optimalité

Problèmes de tournées de véhicules

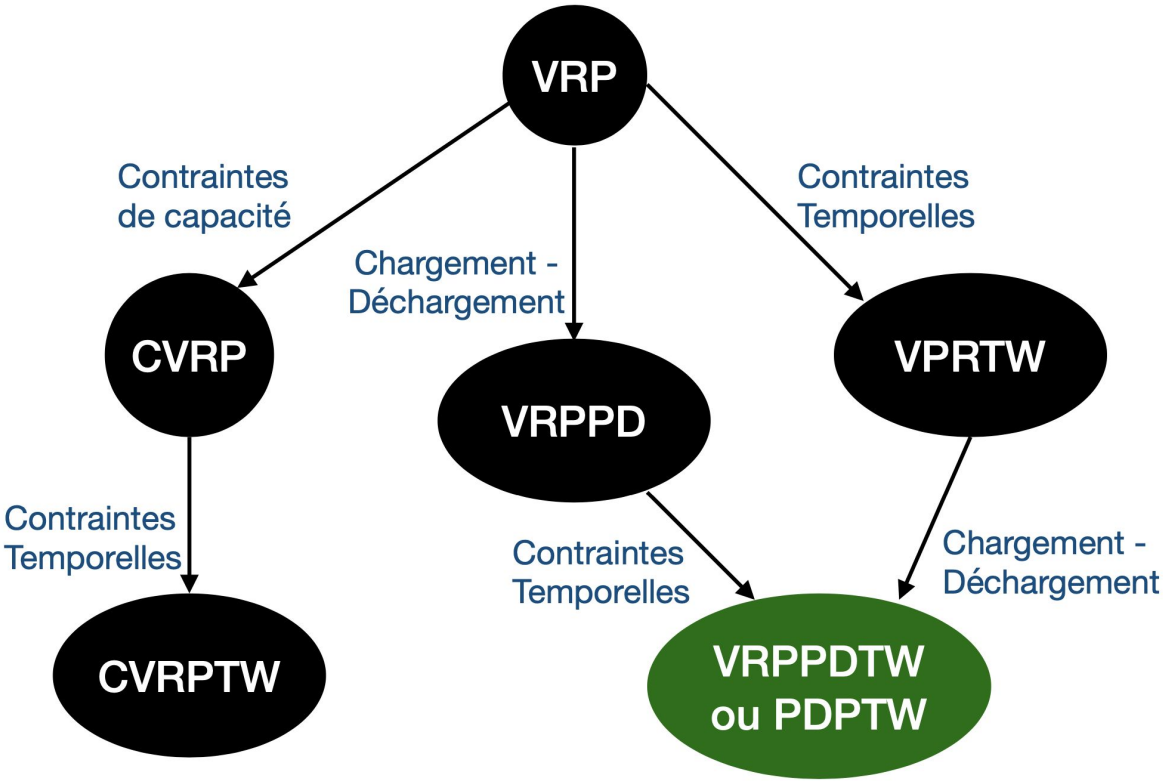


(a) Traveling Salesman Problem

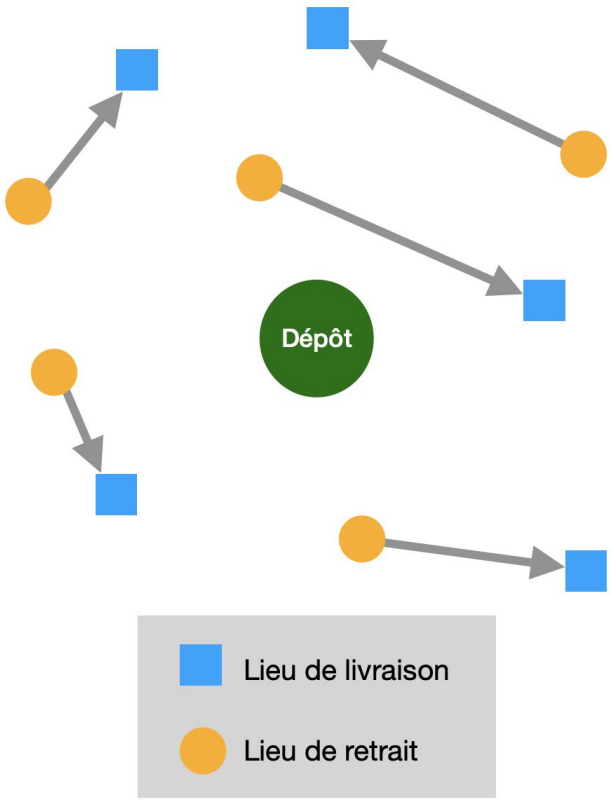


(b) Vehicule Routing Problem

Problèmes de tournées de véhicules



Pick-up and Delivery Problem with Time Windows

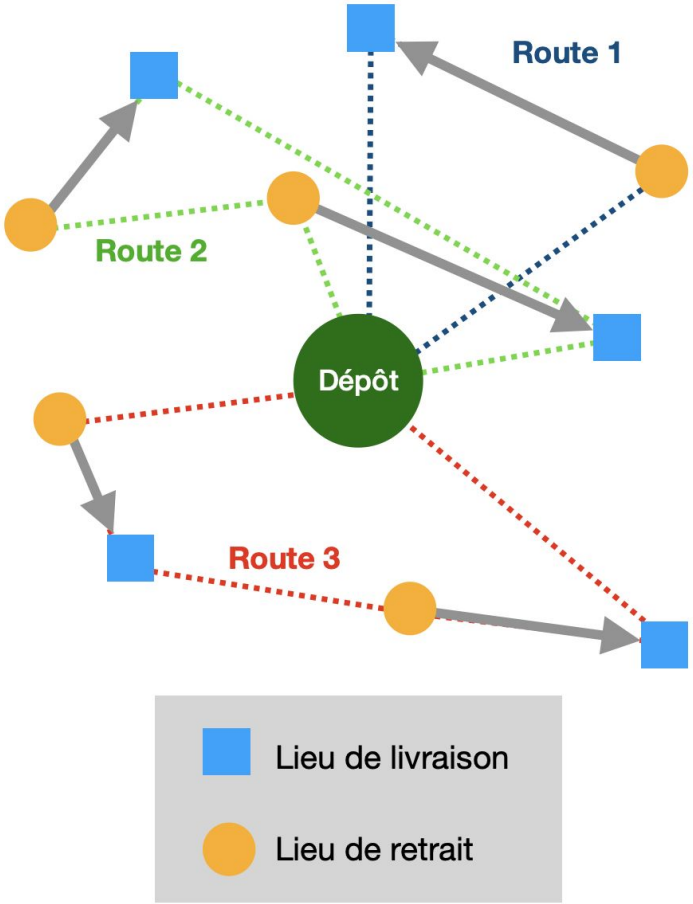


- M véhicules
  - capacité limitée
  - vitesse constante
- N requêtes retrait-livraison
  - fenêtres horaires
  - quantités à livrer
- Minimiser
  - le nombre de véhicules
  - la distance totale

Exemple d’itinéraires

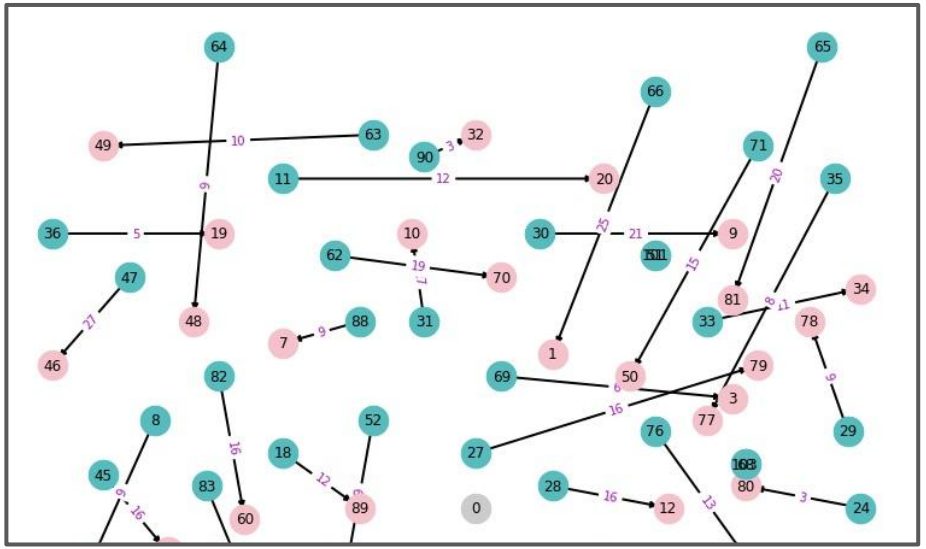
Contraintes

- Départ du dépôt & retour au dépôt
- Livrer toutes les requêtes
- Paire retrait-livraison sur une même route
- Visiter le lieu de retrait avant le lieu de livraison
- Capacité des véhicules
- Fenêtres horaires des clients



Exemple d’instance

Instances des auteurs
Li & Lim **(100 noeuds)**



	node	x	y	demands	earliest_time	latest_time	service_time	pickup_index	delivery_index
0	0	35	35	0	0	230	0	0	0
1	1	41	49	−25	161	171	10	66	0
2	2	35	17	7	50	60	10	0	73
3	3	55	45	−6	116	126	10	69	0

Formalisme mathématique

Plan de route $\rho := \{r_1, r_2, \dots, r_m\}$

Route d'un véhicule $r =< 0, v_1, \dots, v_n, 0 >$

Coût d'une route $t(r) = c_{0,v_1} + c_{v_1,v_2} + \dots + c_{v_{n-1},v_n} + c_{v_n,0}$

Fonction objectif $\gamma(\rho) := \begin{pmatrix} |\rho| \\ \sum_{r \in \rho} t(r) \end{pmatrix}$

Nombre de véhicules
critère n°1

Distance totale
critère n°2

Formalisme mathématique

Date de départ en i

$$\begin{cases} \delta_0 = 0 \\ \delta_i = \max(\delta_{i^-} + c_{i^- i}\ ,\ e_i) + s_i \quad (i \in Customers). \end{cases}$$

Date d'arrivée en i

$$a_i = \max(\delta_{i^-} + c_{i^- i}\ ,\ e_i) \quad (i \in Customers).$$

Chargement en c

$$q(c) = \sum_{i \in cust(r)\ \&\ \delta_i \leq \delta_c} q_i.$$

Contraintes

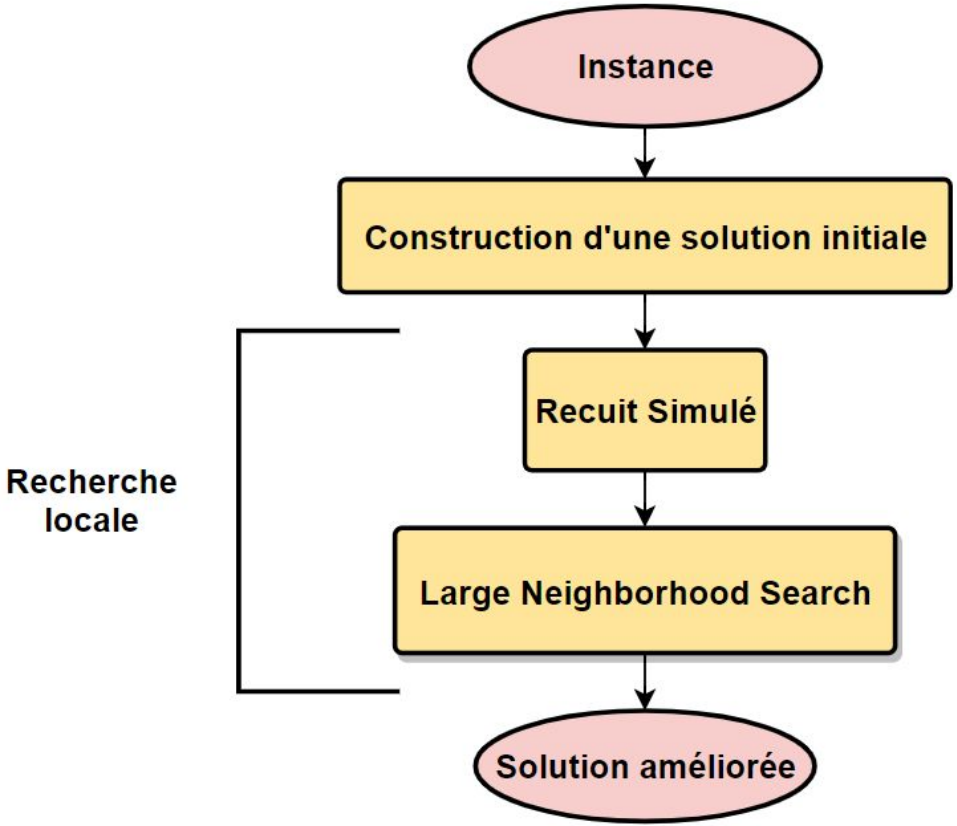
$$\begin{cases} q(i) \leq Q & (i \in Customers) \\ a(r_j) \leq l_0 & (1 \leq j \leq m) \\ a_i \leq l_i & (i \in Customers) \\ route(i) = route(@i) & (i \in Customers^p) \\ \delta_i \leq \delta_{@i} & (i \in Customers^p) \end{cases}$$

Fenêtres temporelles

Fenêtre horaire du client $\mathbf{v_i}$: $[\ e_i\ ,\ l_i\ ]$

- Départ à $\mathbf{t_0 = 0}$ de tous les véhicules
- Arrivée au client $\mathbf{v_i}$ à l'instant $\mathbf{a(v_i)}$
  - $\mathbf{a(v_i) < e_i}$ ➡ attente pendant $\mathbf{e_i - a(v_i)}$ puis service de manutention pendant $\mathbf{s_i}$
  - $\mathbf{e_i \leq a(v_i) \leq l_i}$ ➡ service de manutention pendant $\mathbf{s_i}$
  - $\mathbf{a(v_i) > l_i}$ ➡ violation de la fenêtre horaire

Structure du programme



II - Construction d’une solution initiale

Construction d'une solution initiale

Algorithme de descente de gradient

1: Soit une route r

2: Répéter

3: Pour chaque paire (c_1 , c_2) de clients dans r :

4: Si la fenêtre horaire de c_2 ferme avant celle de c_1 :

5: Échanger les positions de c_1 et c_2 dans r

6: D = coût(r') - coût(r)

7: Si D < 0:

8: r = r'

9: Jusqu'à ce qu'à l'absence d'amélioration

Construction d’une solution initiale

Algorithme de construction séquentielle

```
1: Répéter
2: Initialiser une route vide r
3:     Pour (toutes les requêtes non assignées) :
4:         Obtenir une requête non assignée i
5:         Insérer i à la fin de r
6:         Appeler la descente de gradient sur r
7:         Si r respecte les contraintes alors :
8:             Marquer i comme inséré
9:         Sinon :
10:            Retirer i de la route r
11: Jusqu’à ce qu’à toutes les requêtes soient insérées
```

III - Amélioration de la solution

Amélioration de la solution

Méthode de destruction-réparation des solutions

$r_1 = < 0\ p_1\ p_2\ d_1\ d_2\ p_3\ p_4\ d_4\ d_3\ 0 >$

$r_2 = < 0\ p_5\ p_6\ d_6\ p_7\ d_7\ d_5\ 0 >$

Déplacement de paires (p_2, p_6, d_7)

- Placer p_2 après p_6
- Placer d_2 après d_7

$r_1' = < 0\ p_1\ \cancel{p_2}\ d_1\ \cancel{d_2}\ p_3\ p_4\ d_4\ d_3\ 0 >$

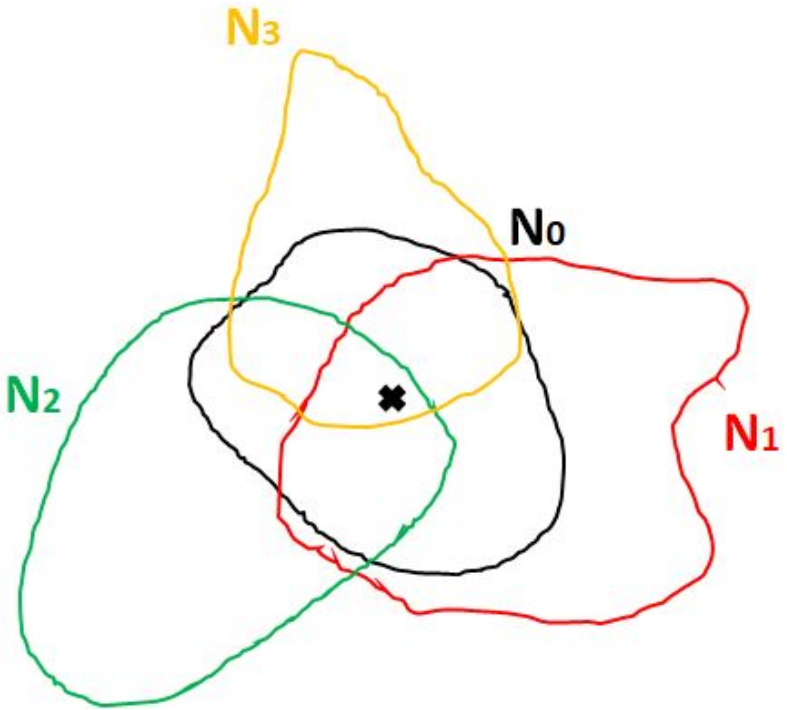
$r_2' = < 0\ p_5\ p_6\ p_2\ d_6\ p_7\ d_7\ d_2\ d_5\ 0 >$

Amélioration de la solution

Définition d’un sous-voisinage

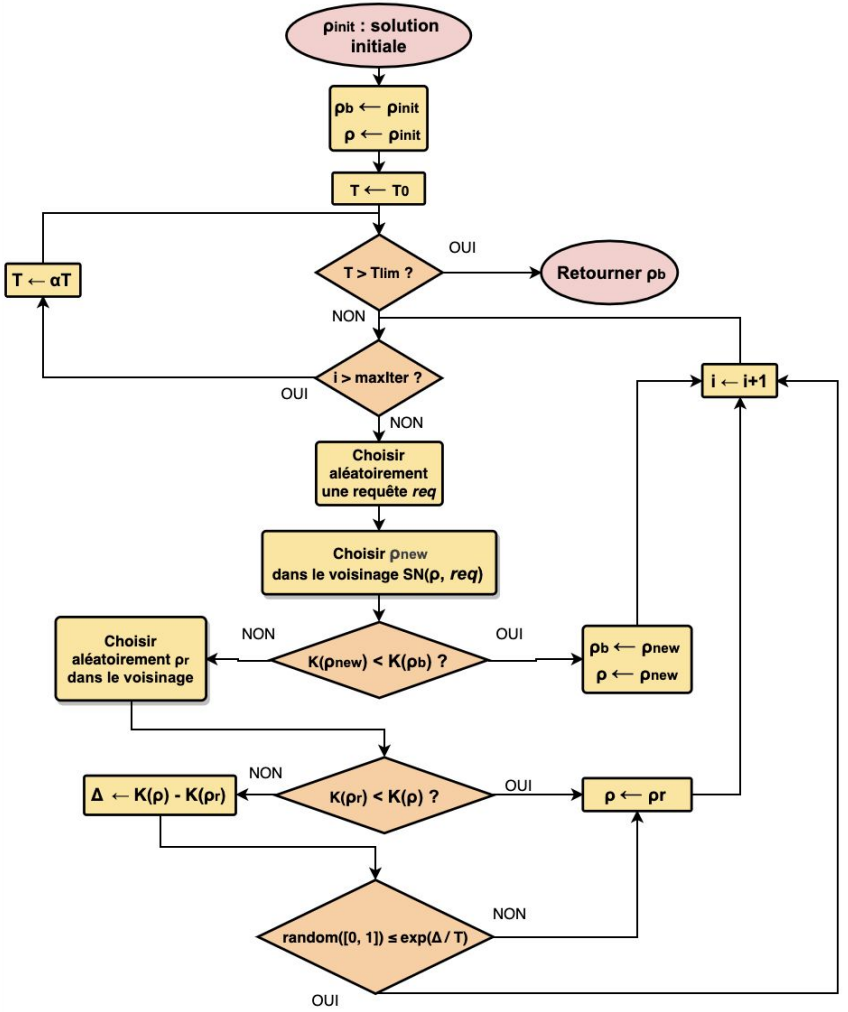
Pour construire un sous-voisinage :

- une solution
- un client choisi aléatoirement
- un opérateur de destruction/réparation



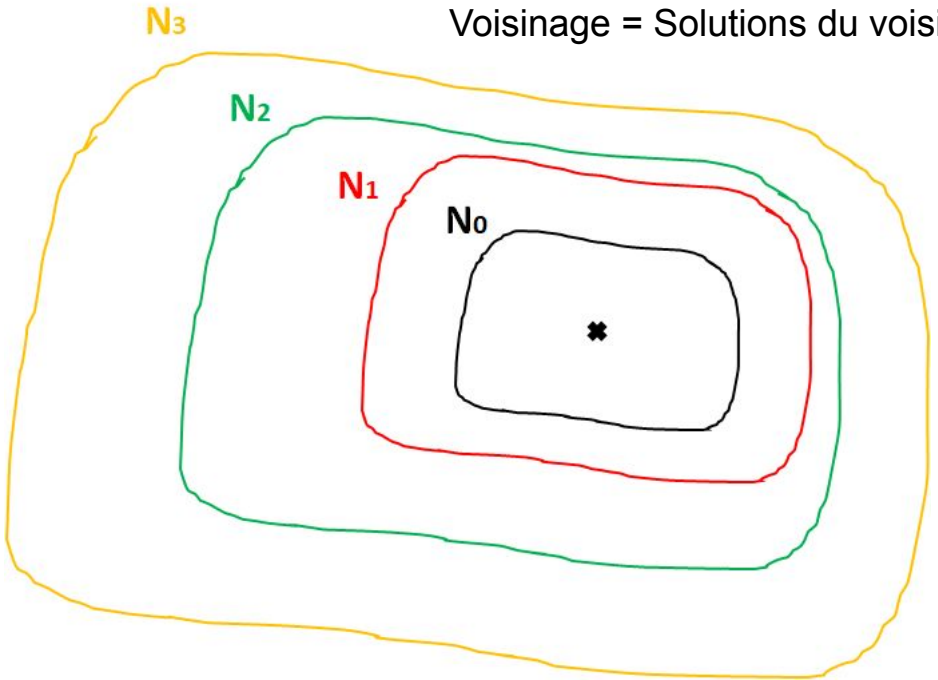
Amélioration de la solution

Algorithme du Recuit Simulé



Amélioration de la solution

Algorithme du LNS (*Large Search Neighborhood*) → voisinage plus étendu

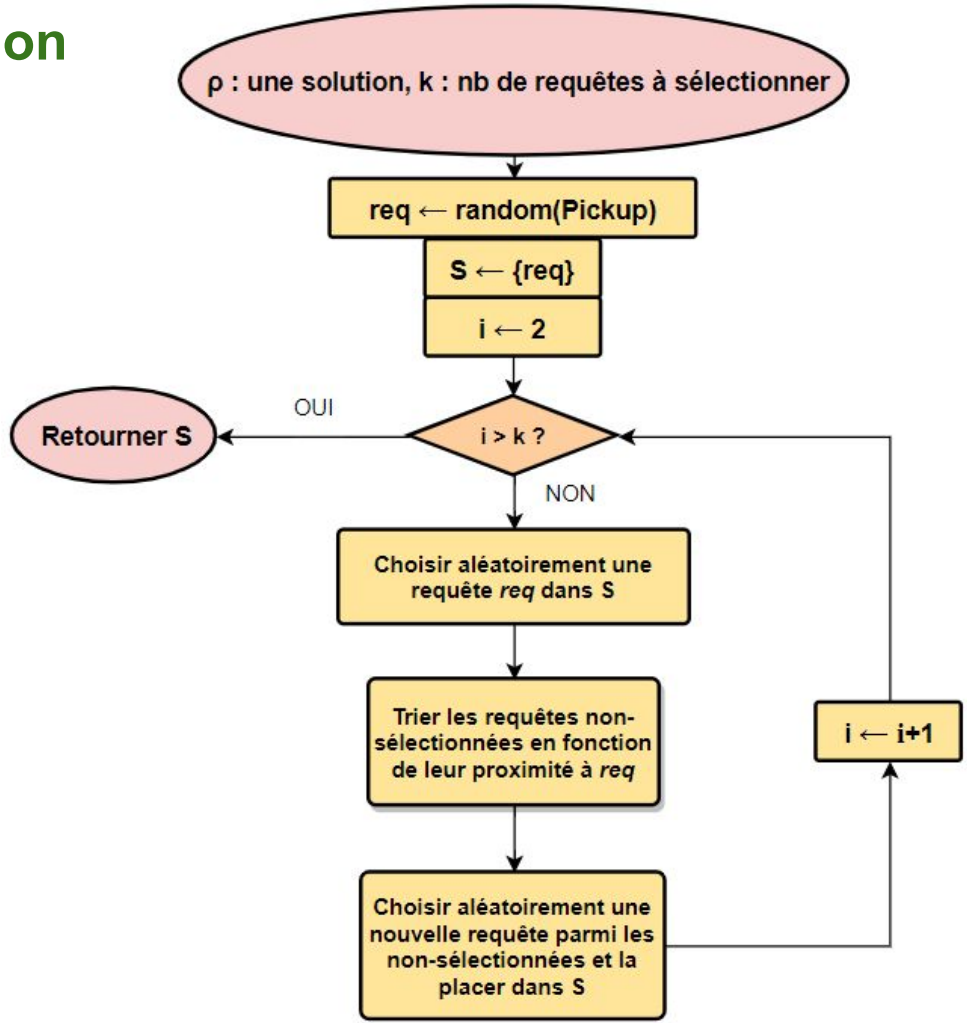


Voisinage = Solutions du voisinage précédent + client parmi une sélection

Sélection de clients S = { c1, c2, …, cp}
p < n

Amélioration de la solution

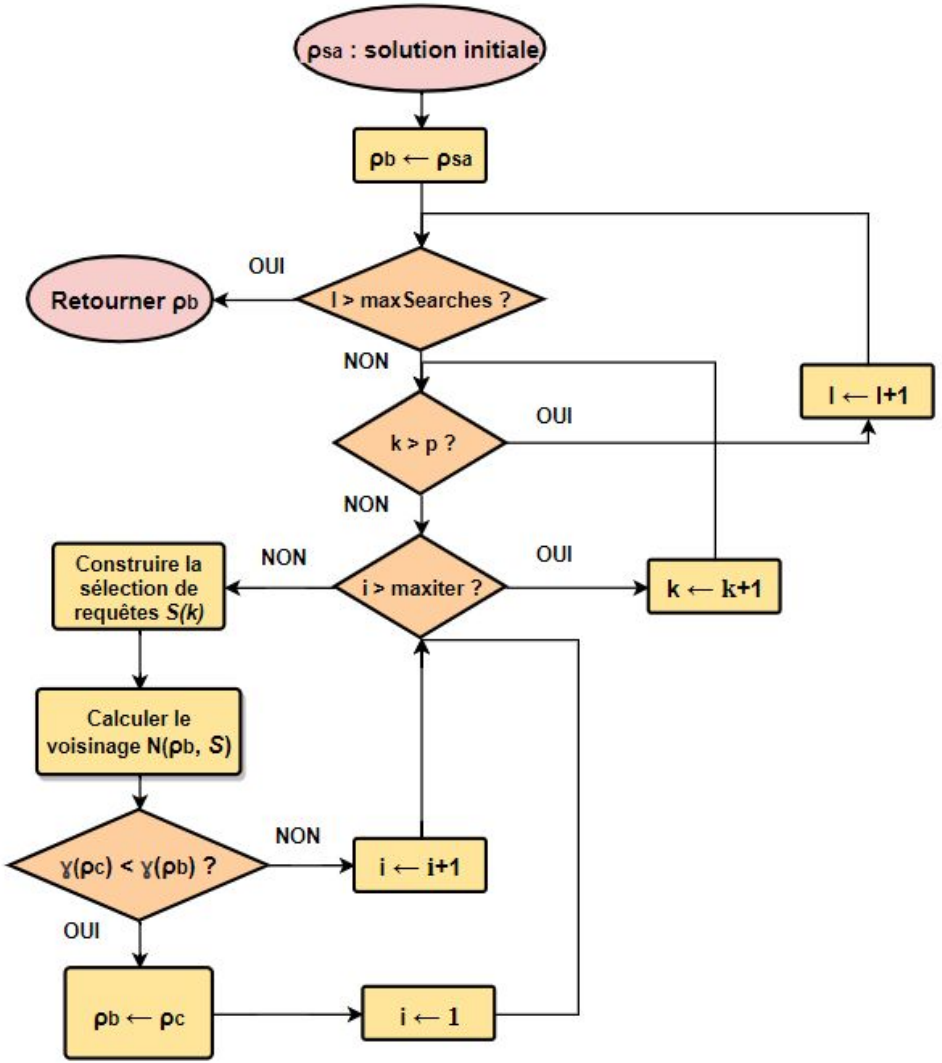
Sélection des requêtes du LNS



Résolution du problème

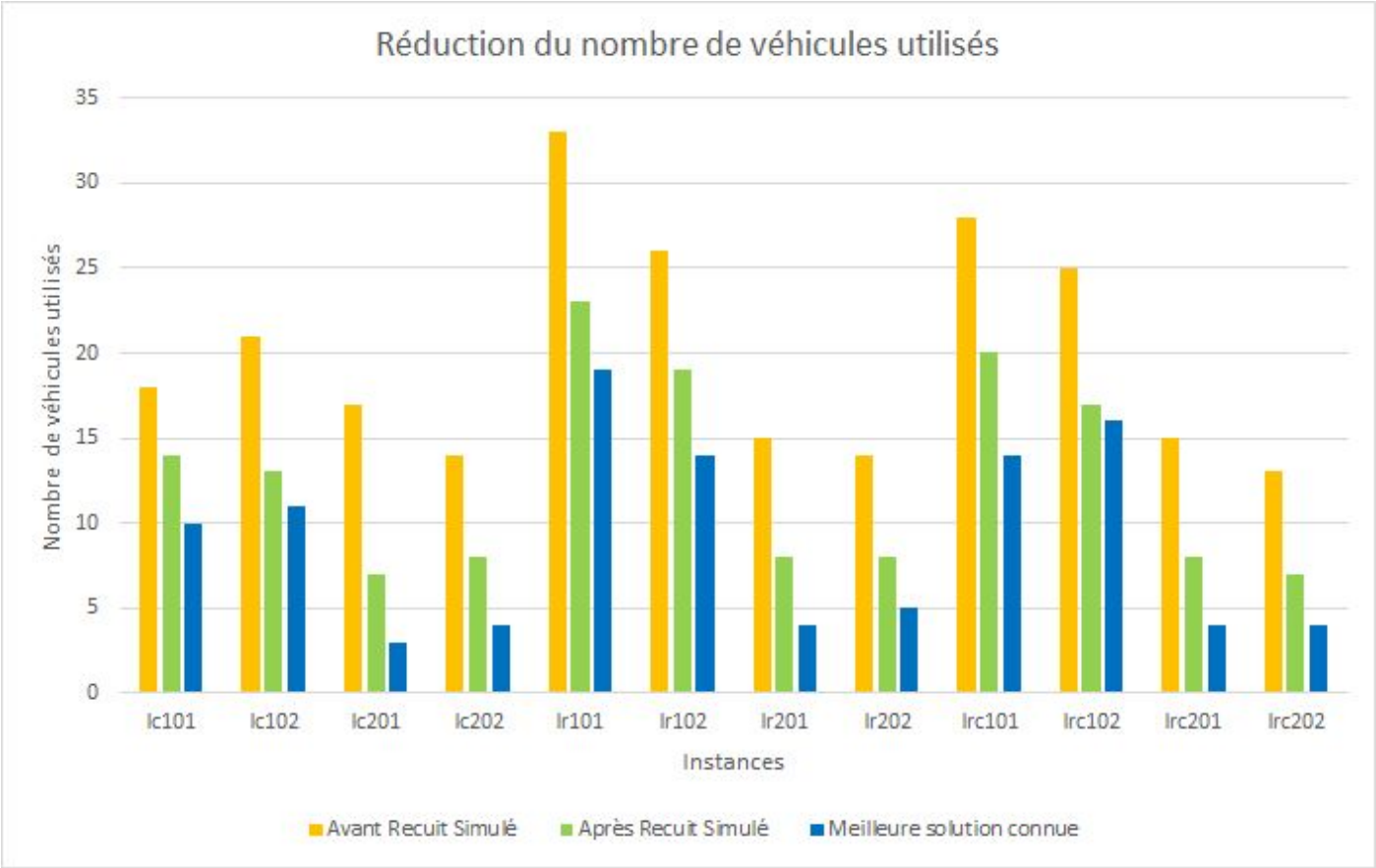
Algorithme du LNS

(Large Neighborhood Search)

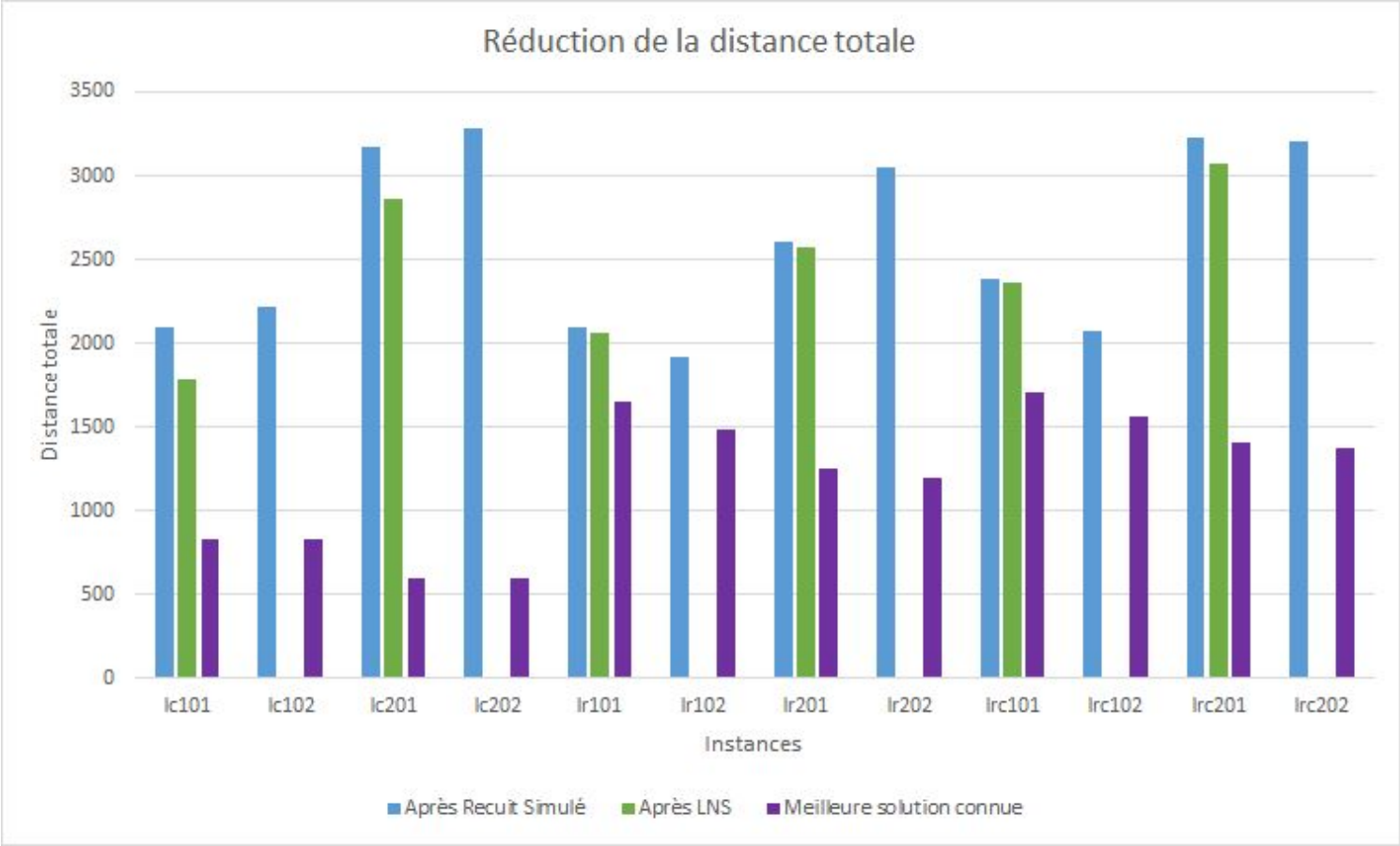


IV - Résultats expérimentaux

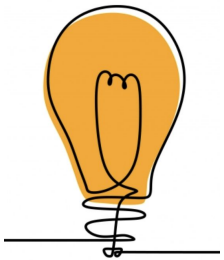
Résultats expérimentaux



Résultats expérimentaux



Conclusion



- Problème NP-difficile
- Diminution du nombre de véhicules mais pas de la distance totale
- D’autres opérateurs de destruction / réparation ?
- Recherche locale peu efficace pour se rapprocher des meilleures solutions
- Méthodes plus efficaces ?
- Optimisation du code