Projet : Métaheuristique multi-objectifs

Optimisation multi-objectifs des chaînes de montage de RENAULT

Oryan Rampon, Corentin Pelhâtre, Mathis Ocquident, Leonard Thaddeus, Adrien Cassaigue, Xavier Pillet

UNIVERSITÉ DE NANTES

M2 Informatique ORO

12 décembre 2019

Introduction

Plan

Car sequencing problem :

Données en sortie :

- Une séquence de véhicules
- Un score associé à la séquence :
  - (RAF) : Nombre de purges dans séquence
  - (EP) : Nombre de contraintes haute priorité violées
  - (ENP) : Nombre de contraintes basse priorité violées

Recherche local :

Pseudo-code général :

Require: TEMPS_MAX
  Calcul d'une séquence initiale (glouton)
  while TEMPS_MAX n'est pas encore atteint **do**
    Choisir une transformation et des positions où l'appliquer
    if la transformation est bonne **then**
      Mise-à-jour de la séquence courante par cette transformation
    end if
  end while
  return Séquence courante

Mouvements :

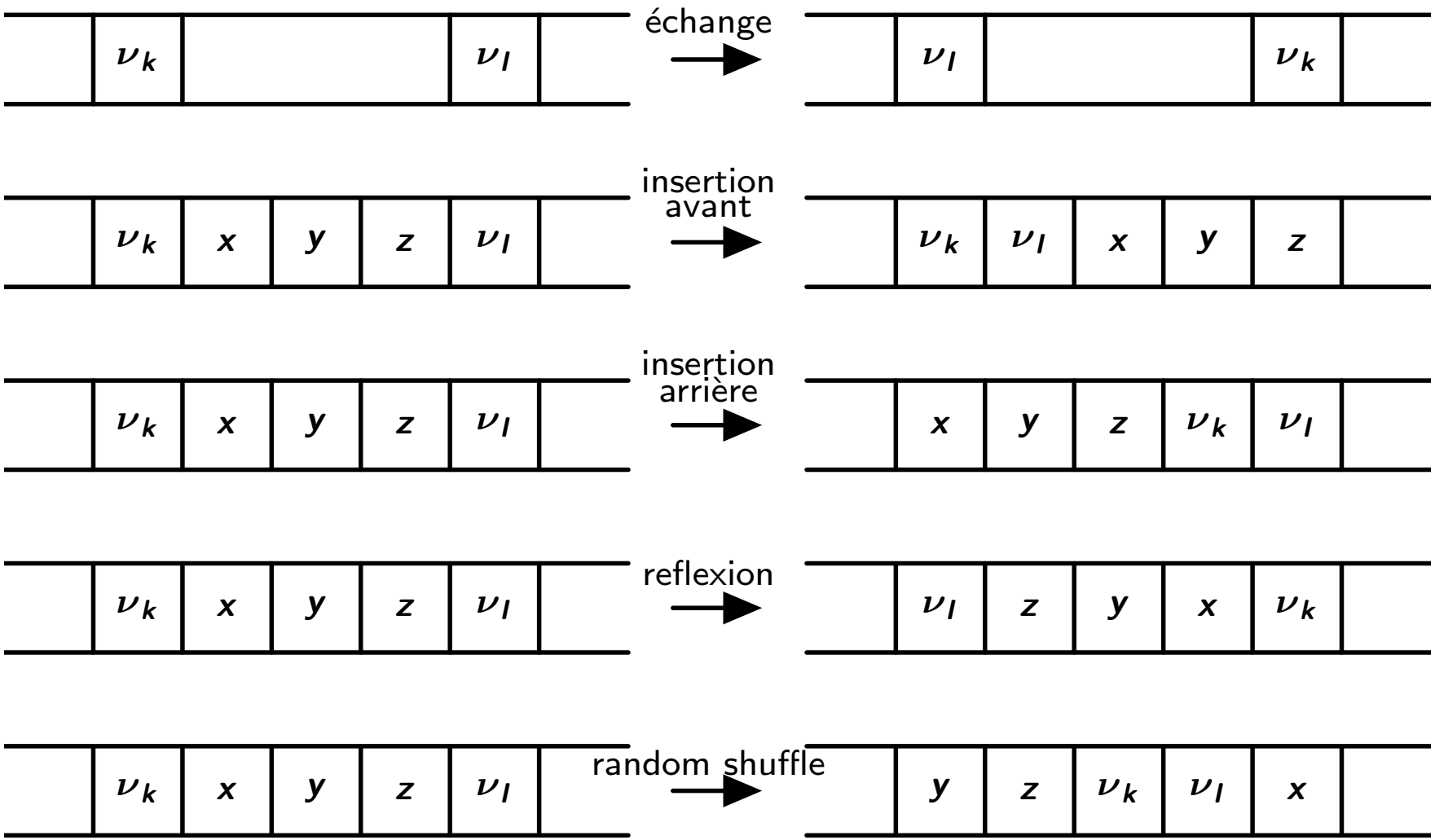


FIGURE – Les cinq mouvements de recherche locale

Structure de données :

Séquence de véhicules :

Séquence :

Voiture 1	Voiture 2	...	Voiture N

Voiture :

Num	Couleur	HPO	LPO	Début	Fin	ID

Tableau de violations

Tableau de violations :

Tableau : Taille de la séquence * Tableau : Taille de la séquence * nombre d'options
nombre d'options
Exemple : Considérant la ligne de l'option i ainsi qu'une séquence de 10 véhicules pour un ratio de 1/3

Séquence	0	1	0	0	1	1	1	1	0	0

Option i	-1	0	0	0	0	1	2	2	1	0

Plan

ND tree :

Structure de données optimisée pour savoir si une solution est efficace et les stocker [1].

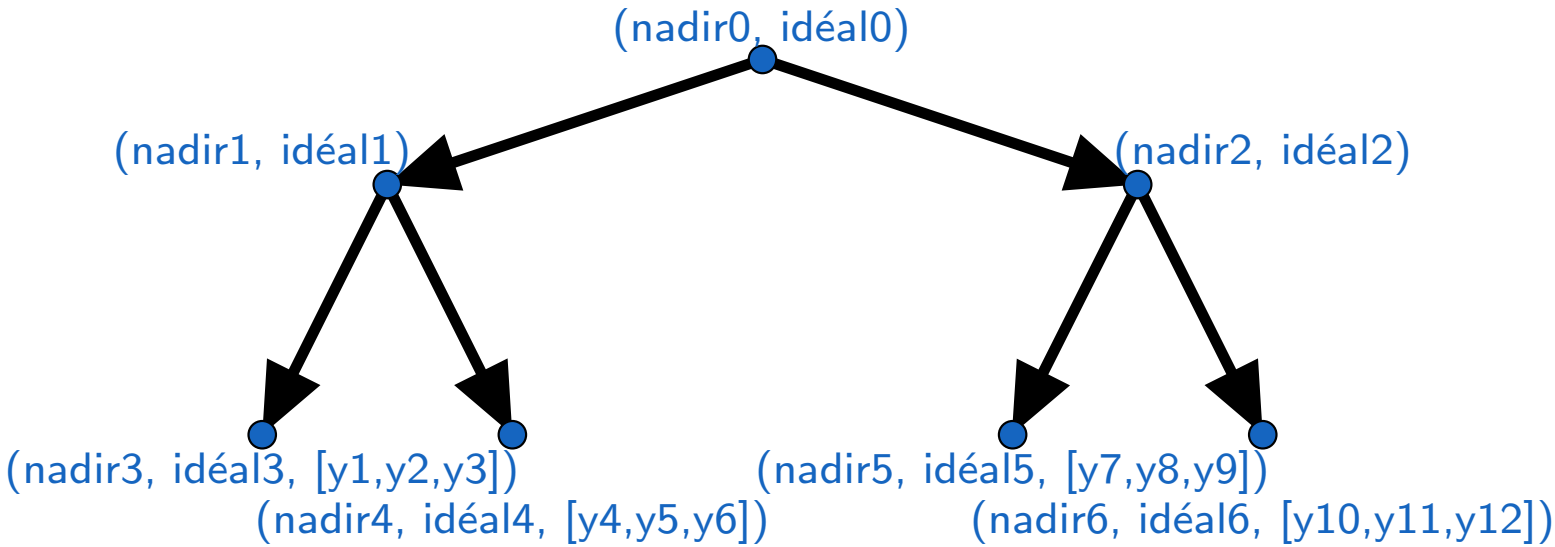


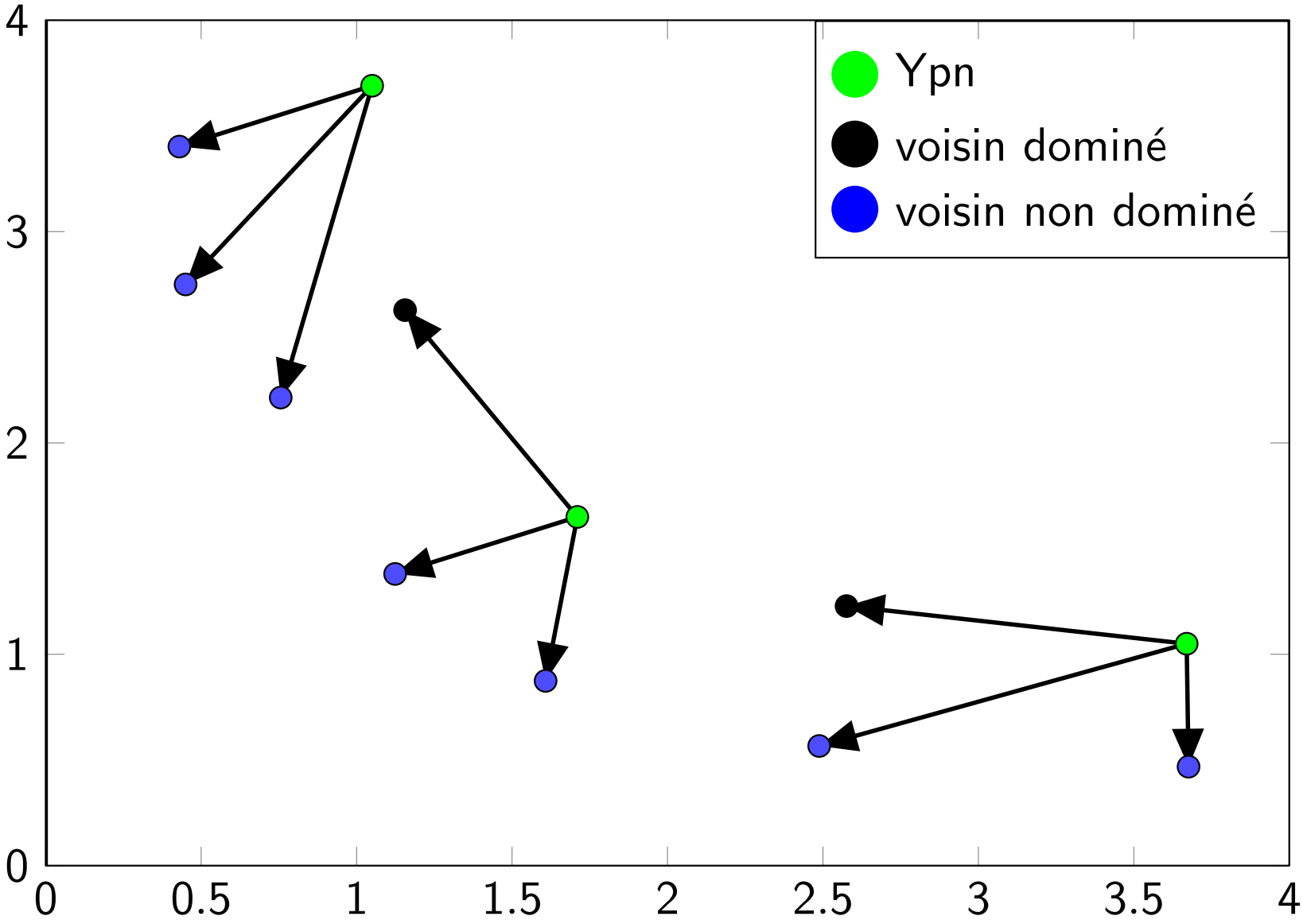
FIGURE – Schéma d'un ND tree

Pareto Local Search :

Pseudo-code général :

while Il existe des elements pas encore visités dans Y_{PN} **do**
  for y dans Y_{PN} **do**
    while Il existe un voisin non dominé de y **do**
      if voisin non dominé par un element de Y_{PN} **then**
        on l'ajoute a Y_{PN}
      end if
    end while
  end for
end while
return Y_{PN}

Pareto Local Search :



Solutions élites

Solutions initiales pour la PLS : solutions élites

6 solutions élites lexicographiques

- RAF EP ENP (algo optimal pour RAF)
- RAF ENP EP
- EP RAF ENP
- EP ENP RAF
- ENP EP RAF
- ENP RAF EP

Chaque solution élite est construite l'une après l'autre avec une VFLS.
On stock les voisins rencontrés dans un ND-Tree pour commencer à avoir Y_{PN}.

Plan

Algorithme génétique

Pseudo-code général :

Require: TEMPS_MAX, NDTree
  Création d'une population initiale
  while TEMPS_MAX n'est pas atteint **do**
    Sélection de deux parents
    Crossover
    Mutation
    Insertion
  end while
  return population finale, NDTree

Génération de la population initiale :

Population élite :

les 6 solutions élites sont obtenues par le principe exposé précédemment.

- 1 solution élite initiale (300 secondes)
- 5 solutions élites a partir de la première (200 secondes)

Population non élite :

Les solutions non élites sont obtenues à partir d'un ordre lexicographique aléatoire.

- 44 solutions non élite (10 secondes chacune)

Sélection :

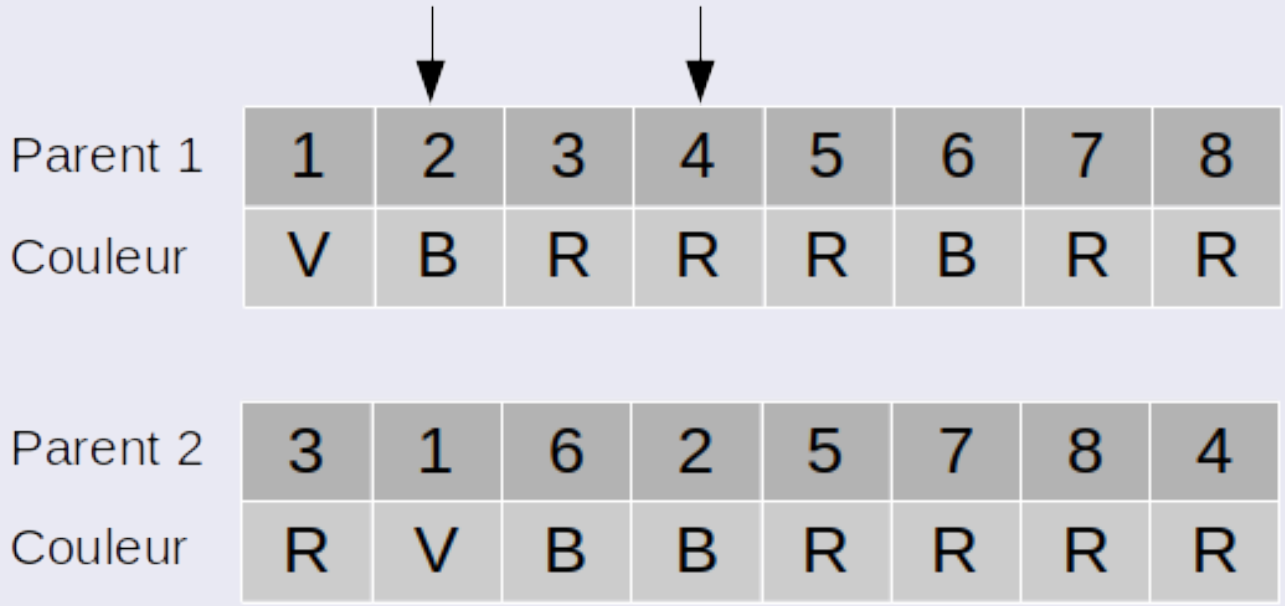
Critère de sélection :

Une sélection très aléatoire mais conservant un fort impact sur la qualité des solutions :

- Un des objectifs (RAF, EP, ENP) est choisi aléatoirement.
- Sélection de deux parents aléatoirement pondérés par l'objectif choisi.

Crossover couleur

Cut points aléatoires sur le père :

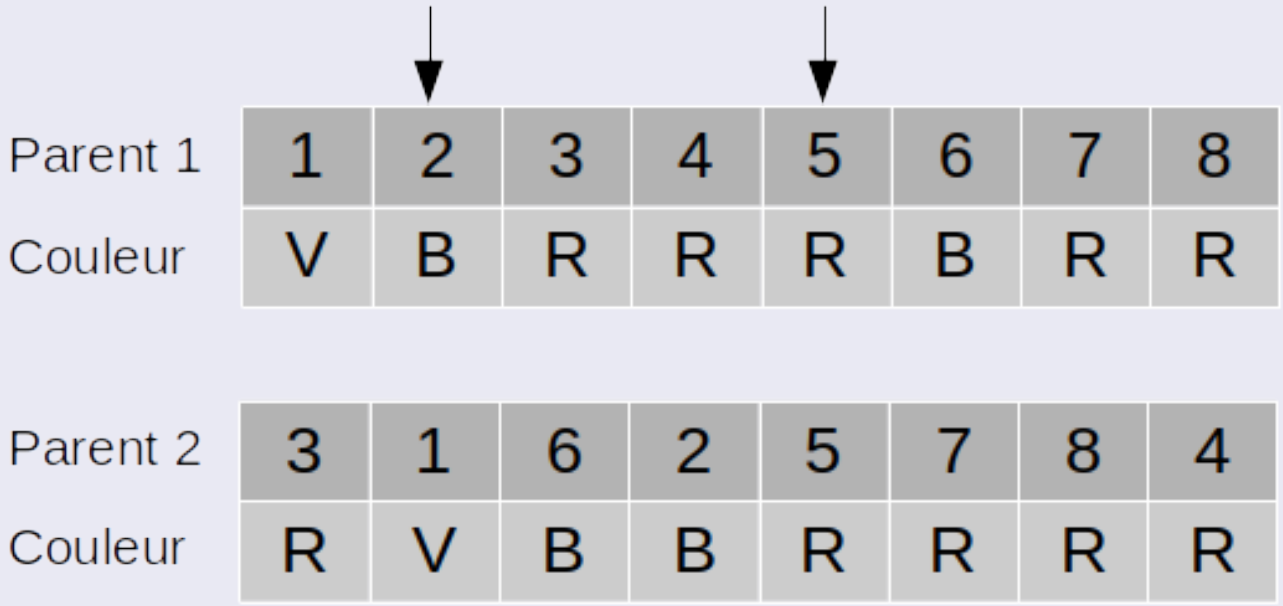


Parent 1	1	2	3	4	5	6	7	8
Couleur	V	B	R	R	R	B	R	R

Parent 2	3	1	6	2	5	7	8	4
Couleur	R	V	B	B	R	R	R	R

Crossover couleur

Déplacer les cut points :

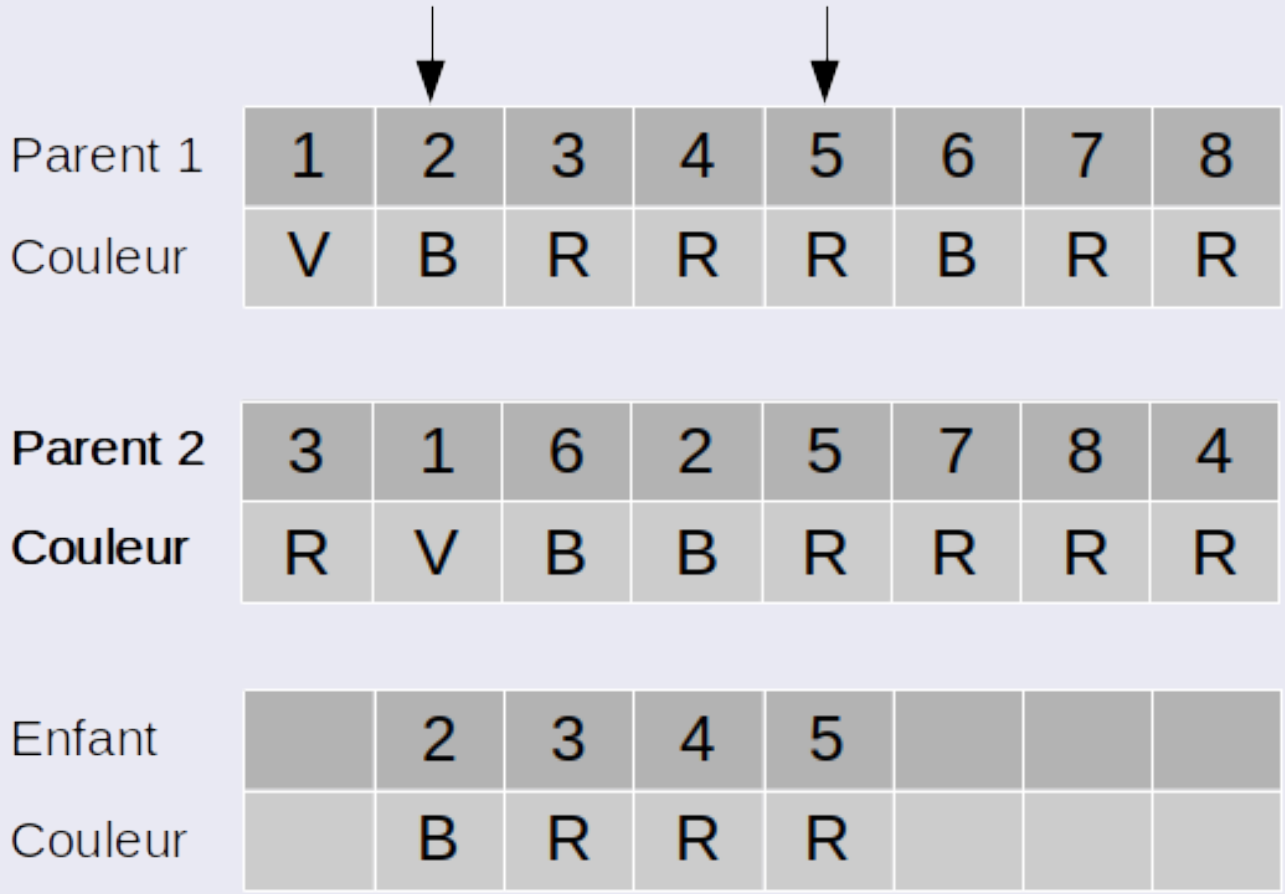


Parent 1	1	2	3	4	5	6	7	8
Couleur	V	B	R	R	R	B	R	R

Parent 2	3	1	6	2	5	7	8	4
Couleur	R	V	B	B	R	R	R	R

Crossover couleur

Copie du parent 1 entre les cuts points :



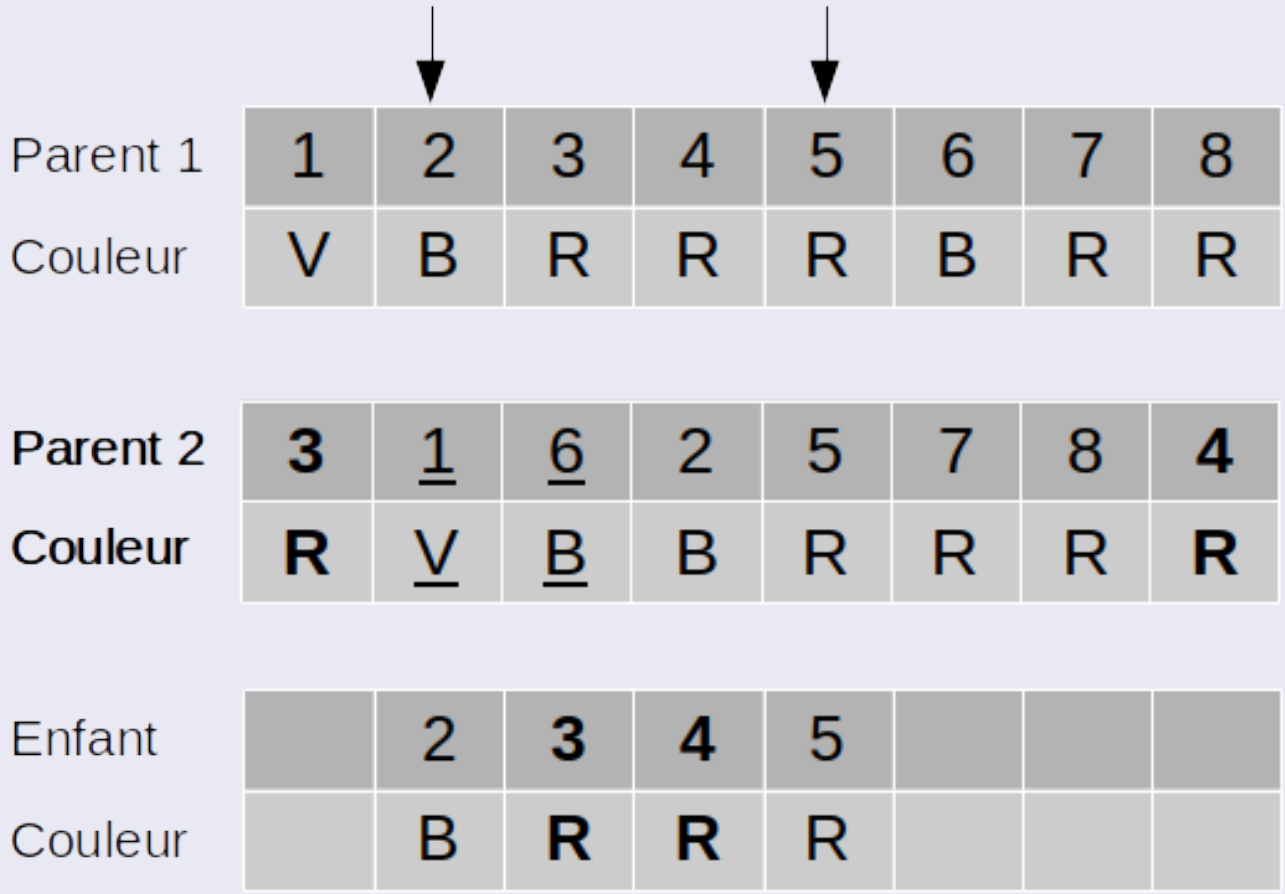
Parent 1	1	2	3	4	5	6	7	8
Couleur	V	B	R	R	R	B	R	R

Parent 2	3	1	6	2	5	7	8	4
Couleur	R	V	B	B	R	R	R	R

Enfant		2	3	4	5			
Couleur		B	R	R	R			

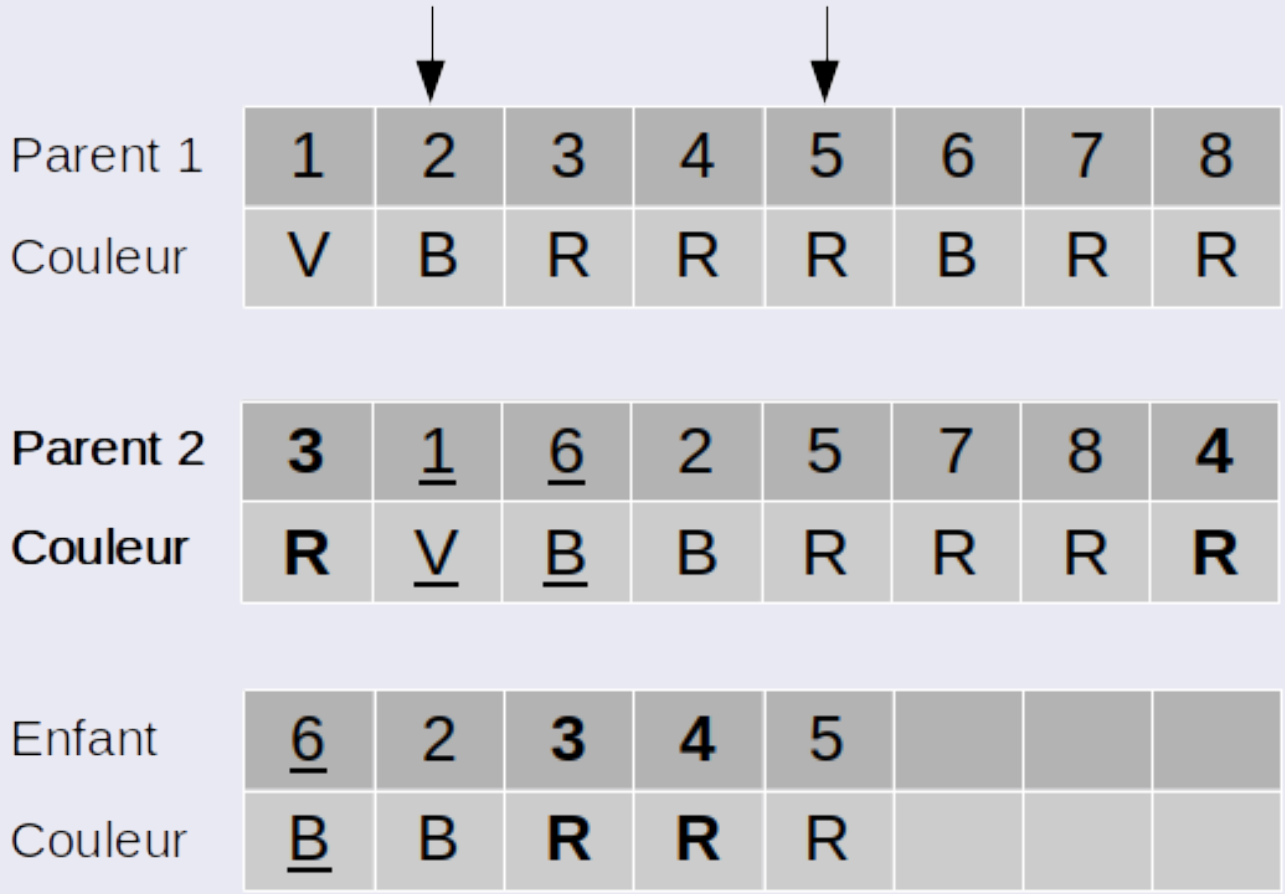
Crossover couleur

Copie du parent 2 :



Crossover couleur

Copie du parent 2 :



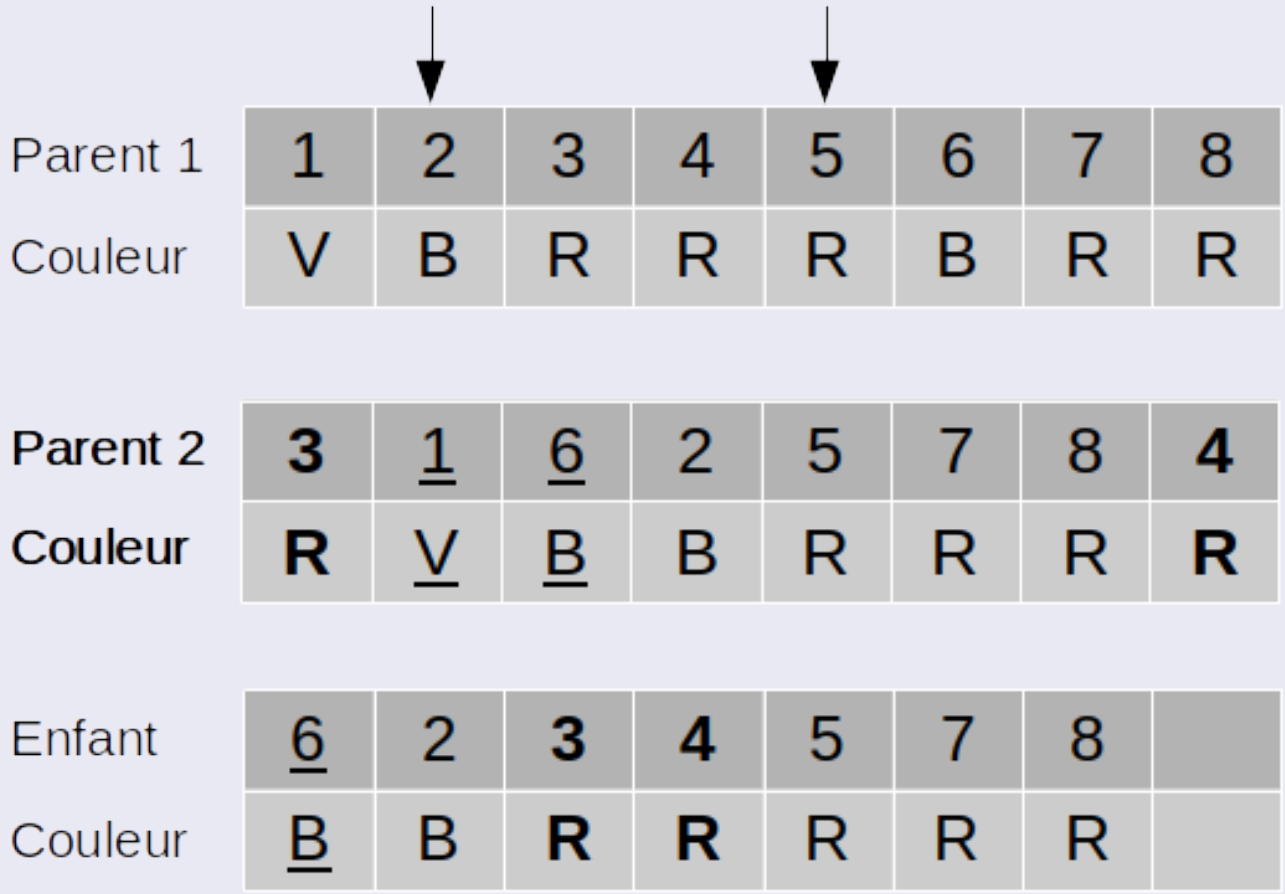
Parent 1	1	2	3	4	5	6	7	8
Couleur	V	B	R	R	R	B	R	R

Parent 2	**3**	1	6	2	5	7	8	**4**
Couleur	**R**	V	B	B	R	R	R	**R**

Enfant	6	2	**3**	**4**	5			
Couleur	B	B	**R**	**R**	R			

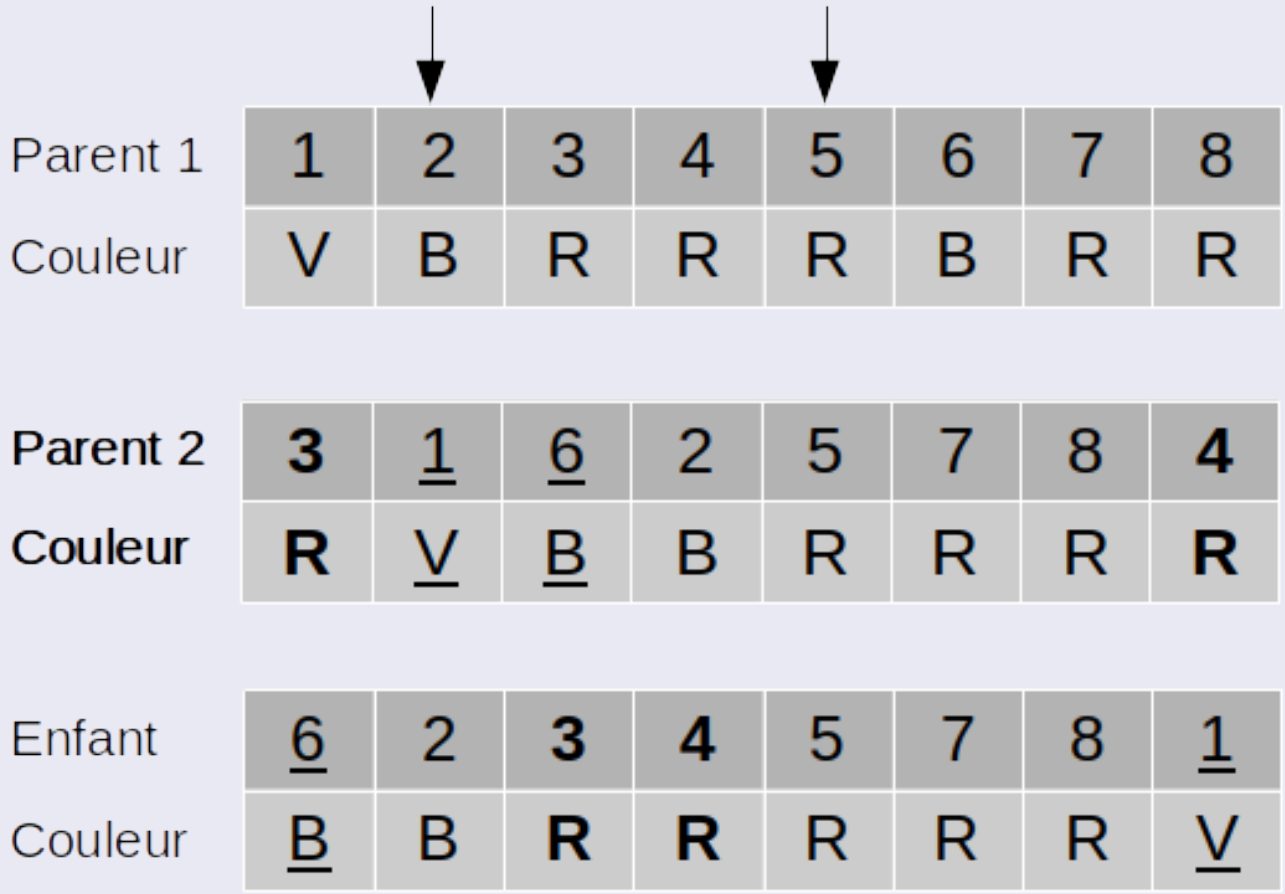
Crossover couleur

Copie du parent 2 :



Crossover couleur

Copie du parent 2 :



Parent 1	1	2	3	4	5	6	7	8
Couleur	V	B	R	R	R	B	R	R

Parent 2	**3**	1	6	2	5	7	8	**4**
Couleur	**R**	V	B	B	R	R	R	**R**

Enfant	6	2	**3**	**4**	5	7	8	1
Couleur	B	B	**R**	**R**	R	R	R	V

Crossover options

Crossover sur ENP ou EP :

Parent	1	2	3	4	5	6	7	8	9	10
Conflit	0	0	2	1	0	3	1	0	0	1

En triant en fonction des conflits, on obtient :

1	2	5	8	9

3	4	6	7	10

Random1 = 4, Random2 = 3 ; on conserve 4 voitures sans conflits à partir de la $3^{ème}$:

Enfant	1	-	-	-	5	-	-	8	9	-

Crossover options

Crossover sur ENP ou EP :

Le tableau de *conflit* est construit à partir des *tableaux de violations* de chacune des options HPO ou LPO.

Parent	1	2	3	4	5	6	7	8	9	10
Option1	-1	-1	0	1	0	1	1	0	0	-1
Option2	-1	0	1	0	0	1	0	0	0	1
Option3	-1	0	1	0	0	1	0	-1	-2	-1
Conflit	0	0	2	1	0	3	1	0	0	1

Crossover options

Crossover sur ENP ou EP :

Parent	1	2	3	4	5	6	7	8	9	10
Conflit	0	0	2	1	0	3	1	0	0	1

En triant en fonction des conflits, on obtient :

1	2	5	8	9

3	4	6	7	10

Random1 = 4, Random2 = 3 ; on conserve 4 voitures sans conflits à partir de la $3^{ème}$:

Enfant	1	-	-	-	5	-	-	8	9	-

Crossover options

Crossover sur ENP ou EP :

On mélange les voitures qu'il reste à insérer :

6	3	10	2	7	4

Random3 = 7 ; on complète la séquence de l'enfant en partant de la position 7 et en minimisant les conflits

Enfant	1	-	-	-	5	-	2	8	9	-

En renouvelant l'opération, on obtient :

Enfant	1	3	6	10	5	4	2	8	9	7

Mutation par VFLS :

Principe :

- Réalisation de la VFLS pendant une durée déterminée (0.1 seconde)

Rappel : la VFLS ne prend que les mouvements améliorants.

Problème :

Nous avons remarquer que cela nuisait à toute diversification de la population.

Mutation par basique swap :

Principe :

- Réalisation d'un nombre prédéfini de *swap* (5)

Rappel : La mutation sert à éviter une convergence prématurée de l'algorithme.

Insertion, première méthode :

Premier algorithme :

Require: population, enfant, NDTree
  mélanger(population)
  while reste un élément dans la population && aucun n'a été dominé **do**
    if enfant domine fortement l'élément courant de la population **then**
      remplacer l'élément courant par l'enfant
      ajouter l'enfant dans le NDTree
    end if
  end while
  while reste un élément dans la population && aucun n'a été dominé **do**
    if enfant domine l'élément courant de la population **then**
      remplacer l'élément courant par l'enfant
      ajouter l'enfant dans le NDTree
    end if
  end while
  return population modifié, NDTree

Insertion, première méthode :

Principe :

- Insertion des enfants 1 par 1
- Insertion lorsque l'enfant domine un élément de la population

Problèmes :

- De nombreuses solutions intéressantes non conservées
- Les solutions de bonne qualité pas assez valorisées

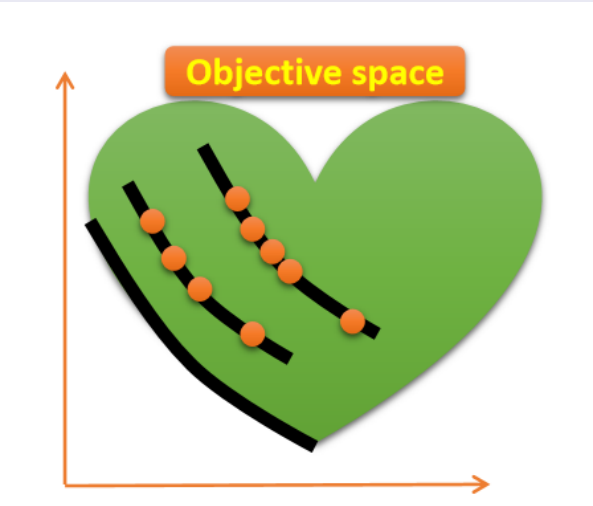
Insertion, seconde méthode :

Principe :

- Création d'une population d'enfants de même taille que l'initiale (50)
- Insertion si non dominé dans le sous ensemble restant

Propriété :

Tout ensemble de solution possède au moins une solution non dominé.



Variantes de l'insertion :

Insertion avec un quota d'enfant :

Require: population, enfants, NDTree
  enfantsGardés, parentsGardés = nouvelle population vide
  while taille(population)-quota d'enfants non respecter **do**
    while Il reste des éléments dans la population **do**
      if Courant non dominé dans population **then**
        parentsGardés = parentsGardés ∪ courant
      end if
    end while
    population = population - parentsGardés
  end while
  Similairement on recupère le quota d'enfants
  ajouter tous les enfantsGardés dans le NDTree
  population = enfantsGardés ∪ parentsGardés
  return population, NDTree

Variantes de l'insertion :

Principe de la seconde variante :

- On fusionne les parents et les enfants
- On réalise a nouveau une recherche récursive de tous les non dominés

Avantage par rapport aux autres méthodes :

- Pas de dégradation de la population
- Conservation de toutes les solutions intéressantes
- Spécialisation sur un objectif légèrement plus lente

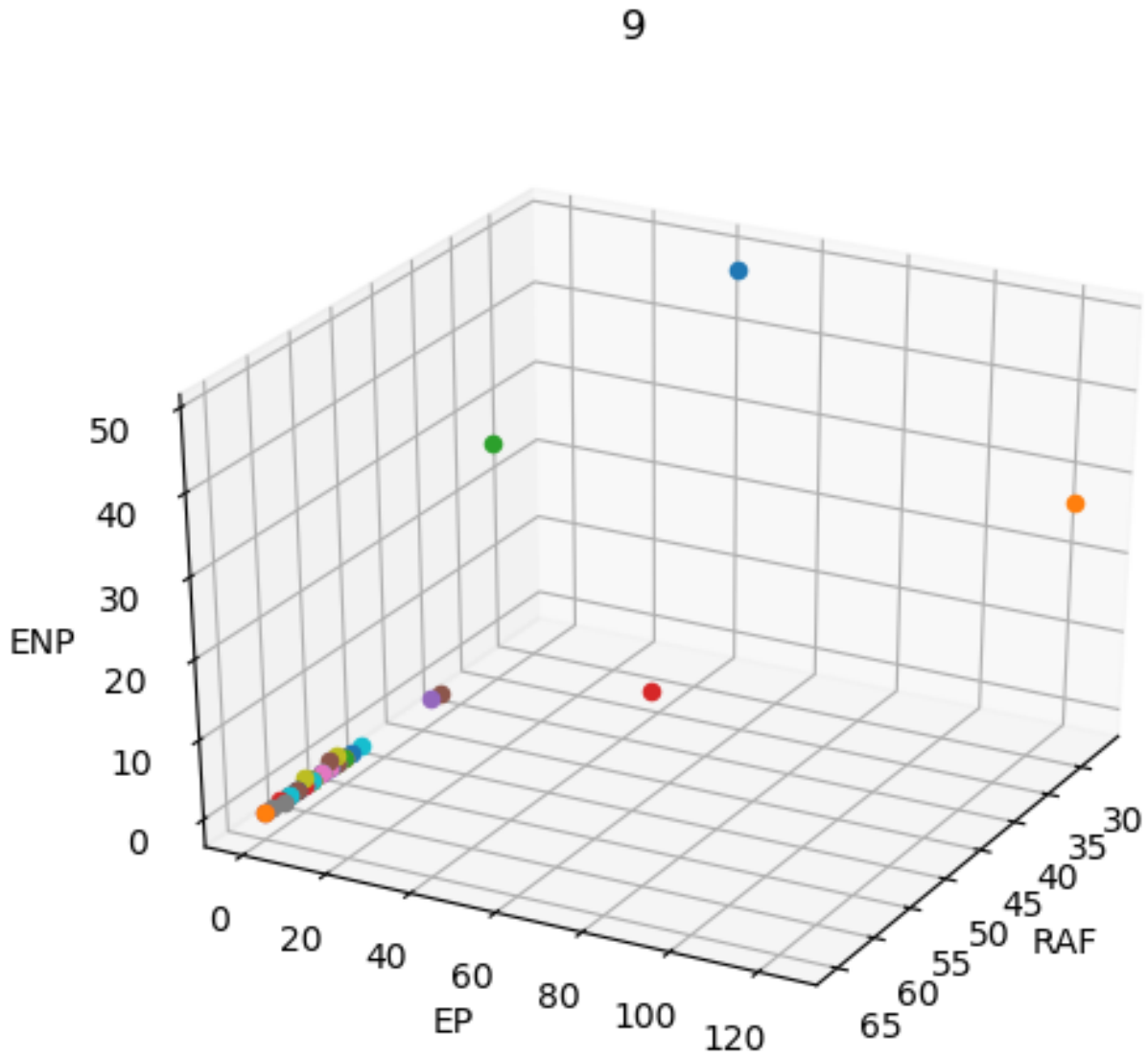
Résultats

Résultats : Génétique

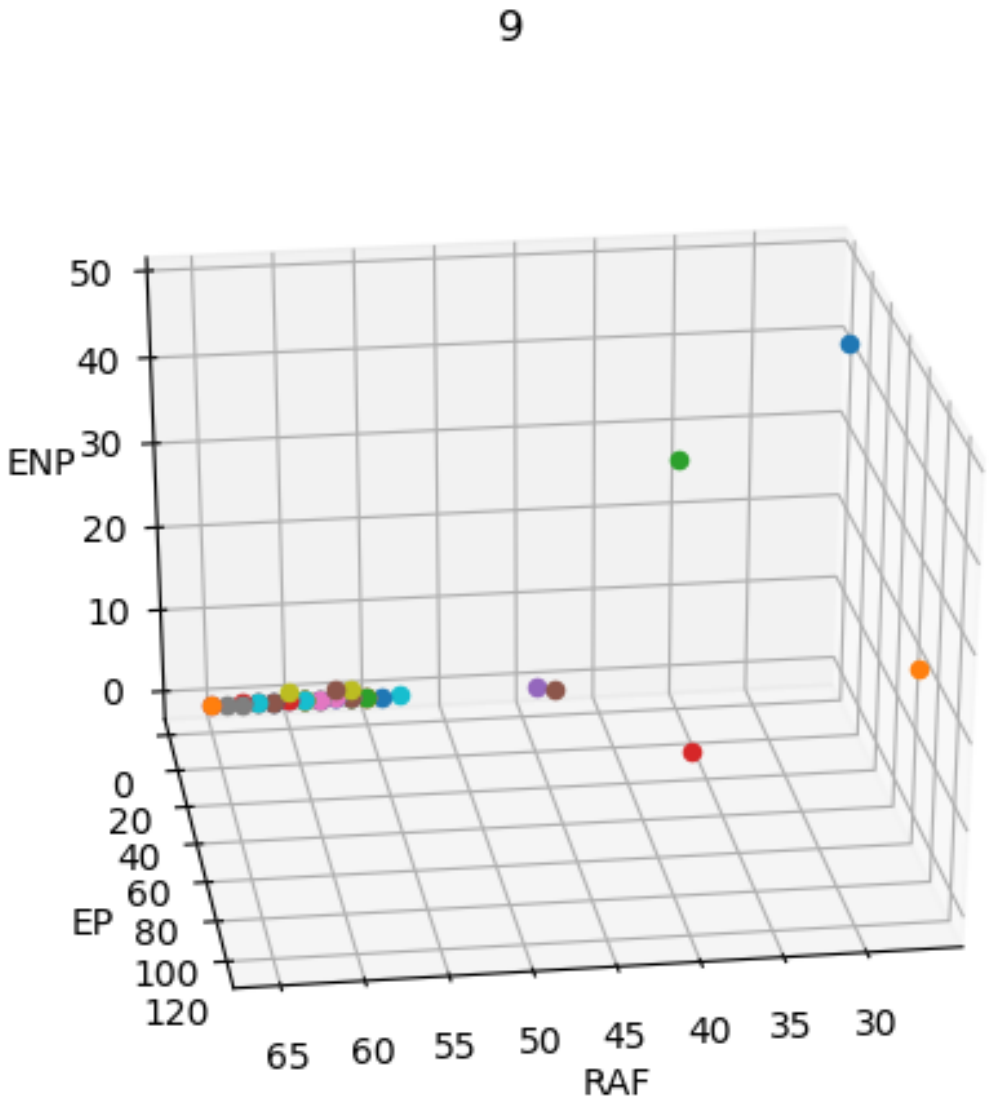
Observations

- Forte spécialisation de la population
- Convergence vers certains points non dominés pour la population finale
- Ajout dans ND_Tree
- Visualisation globale intéressante

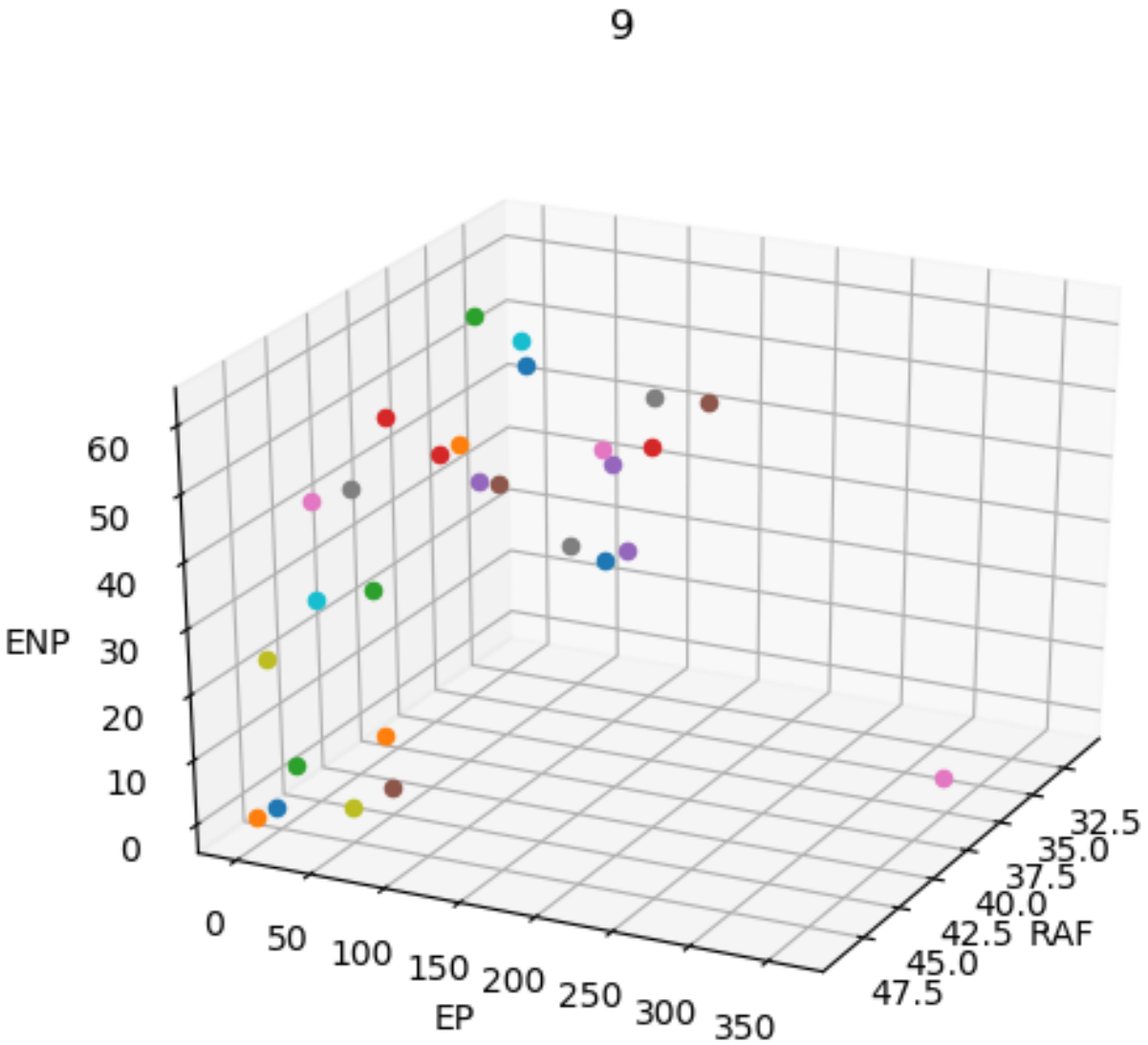
Résultats : Génétique, instance 064_38_2_RAF_EP_ENP_ch2



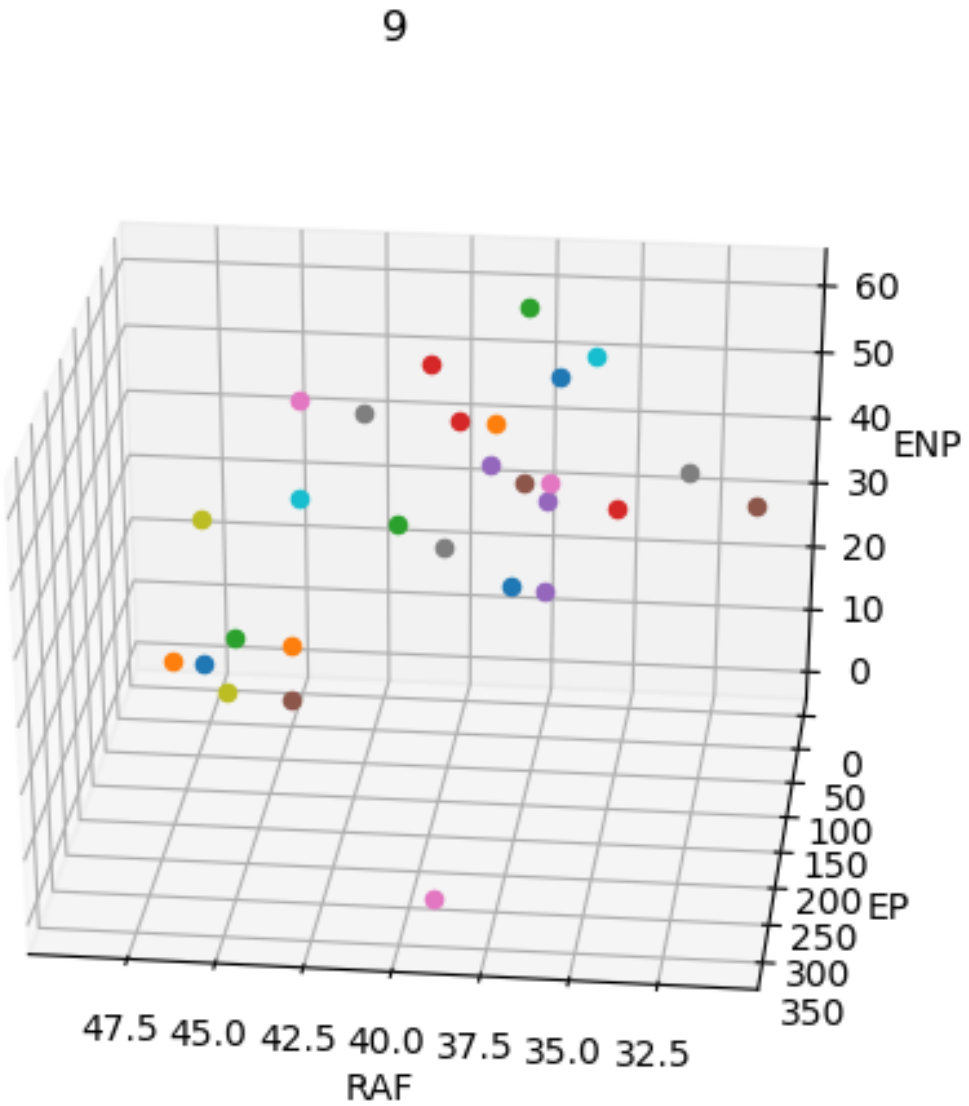
Résultats : Génétique, instance 064_38_2_RAF_EP_ENP_ch2



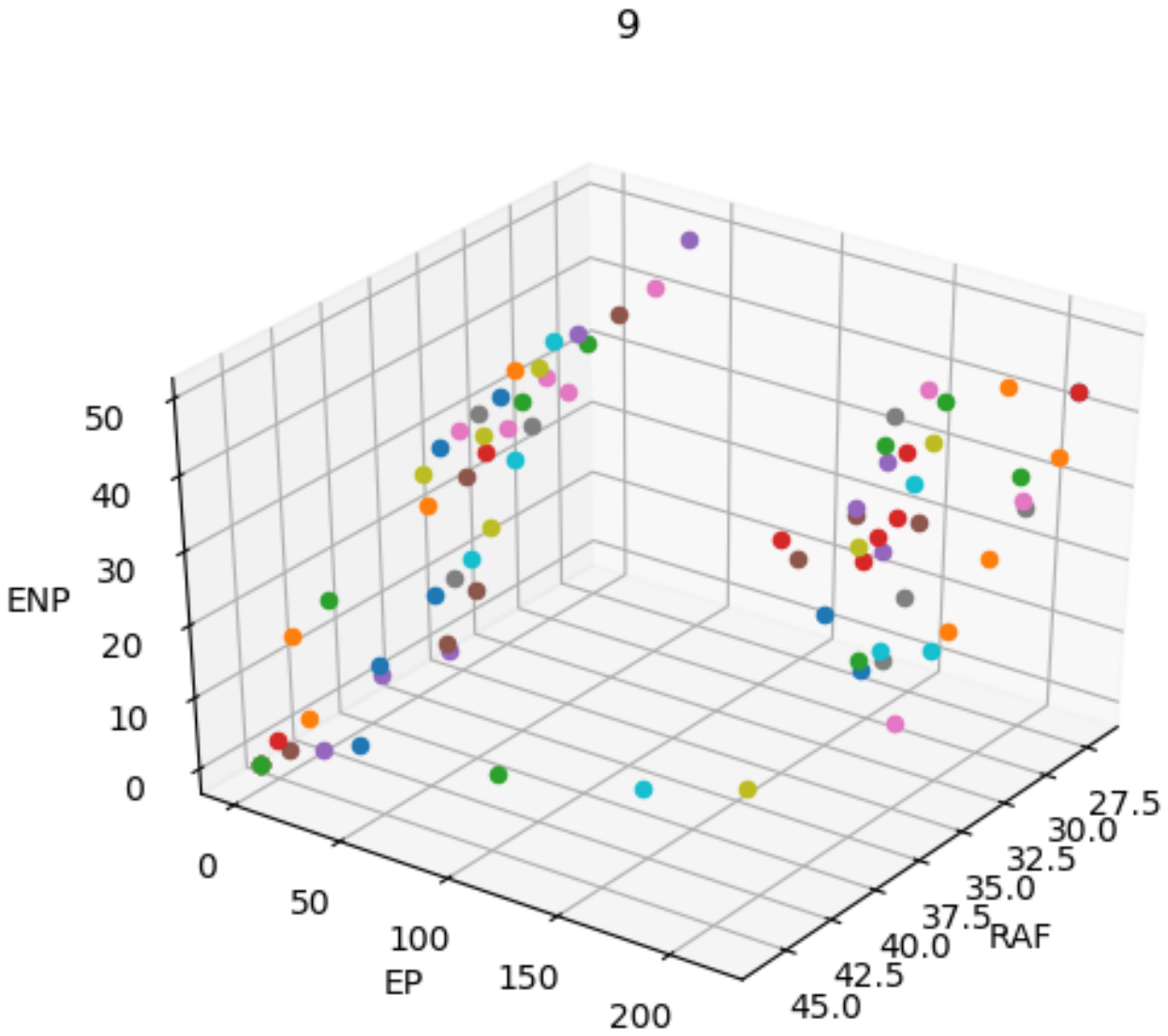
Résultats : Génétique, instance 064_38_2_RAF_EP_ENP_ch2



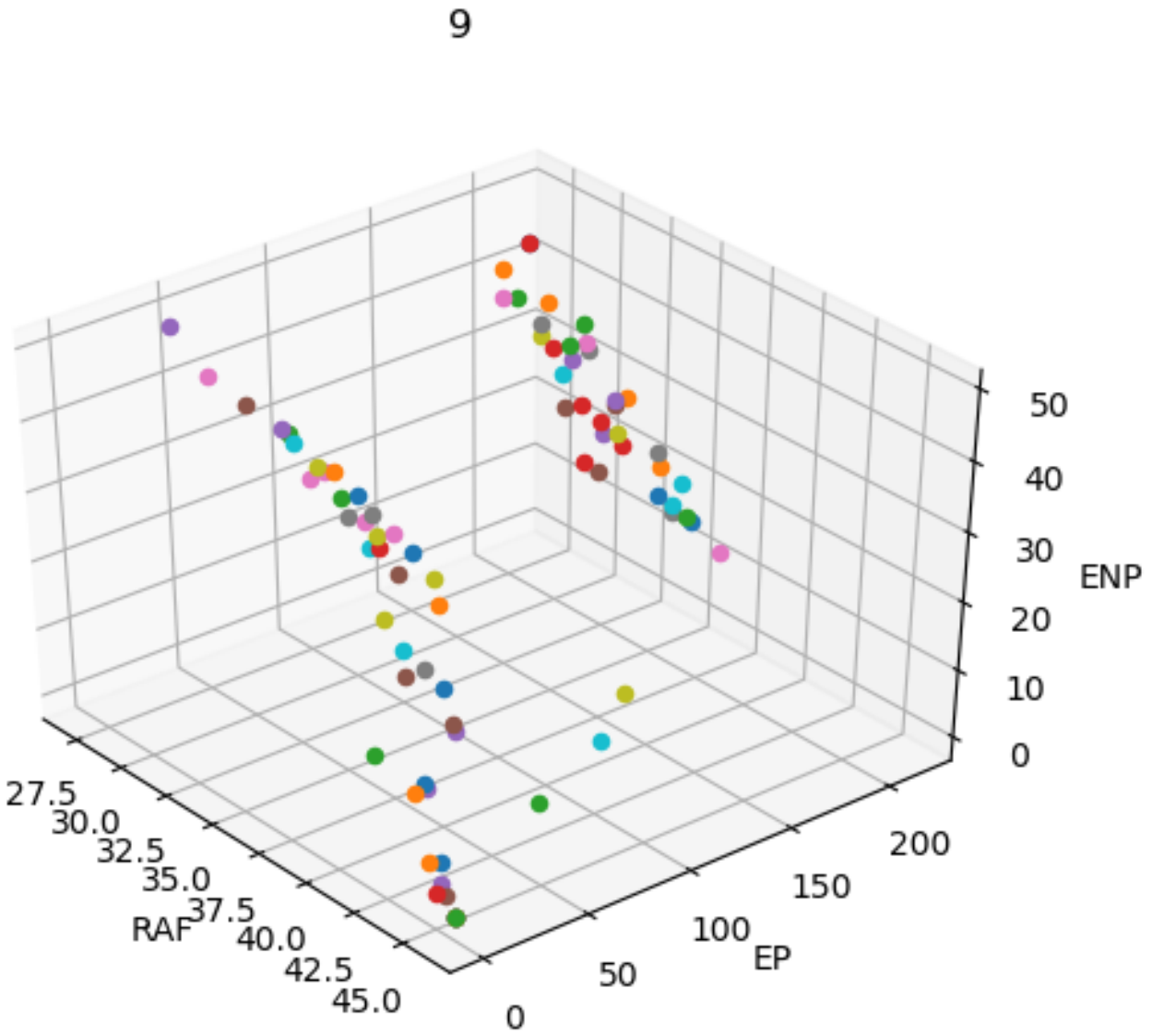
Résultats : Génétique, instance 064_38_2_RAF_EP_ENP_ch2



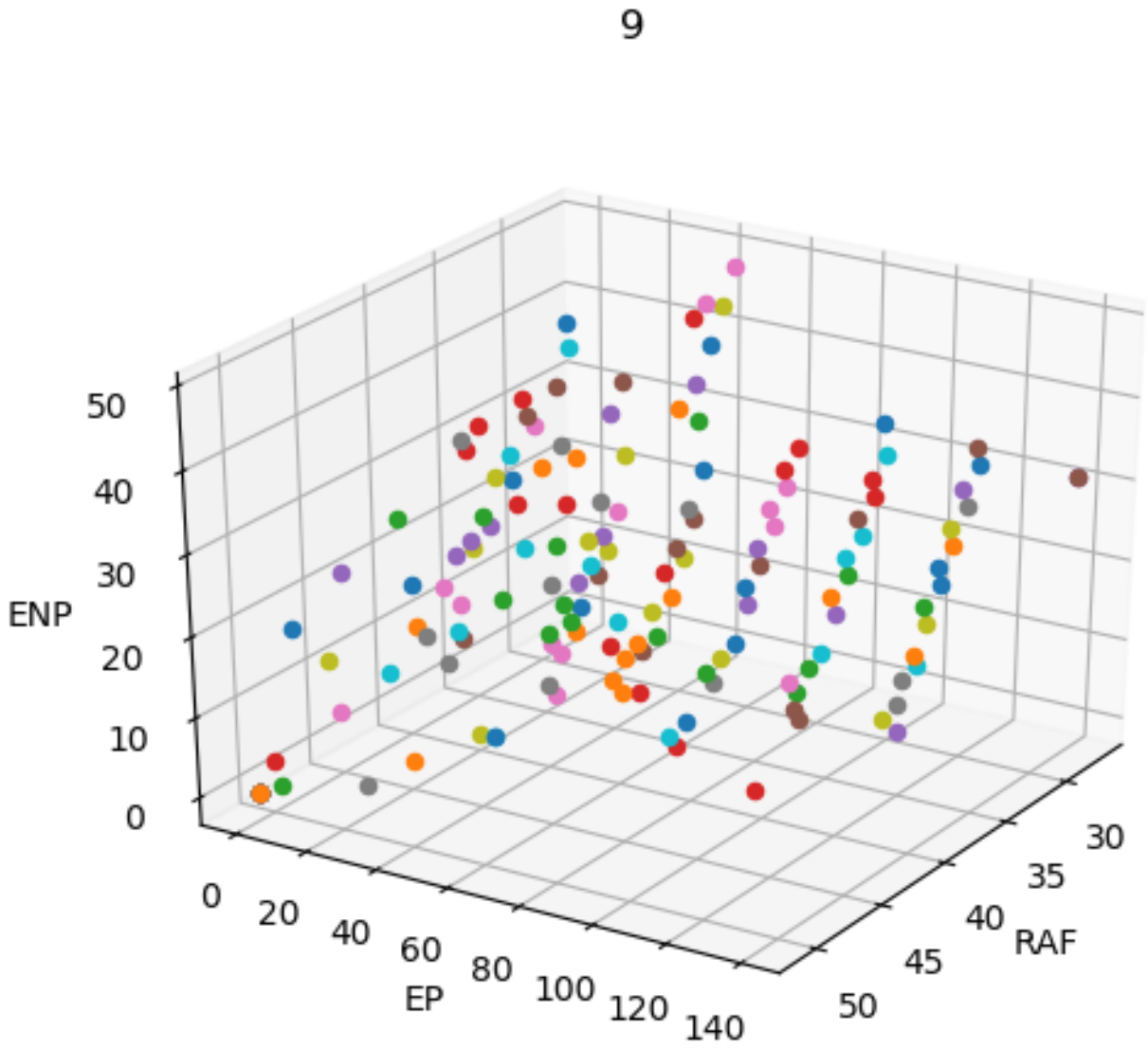
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch2



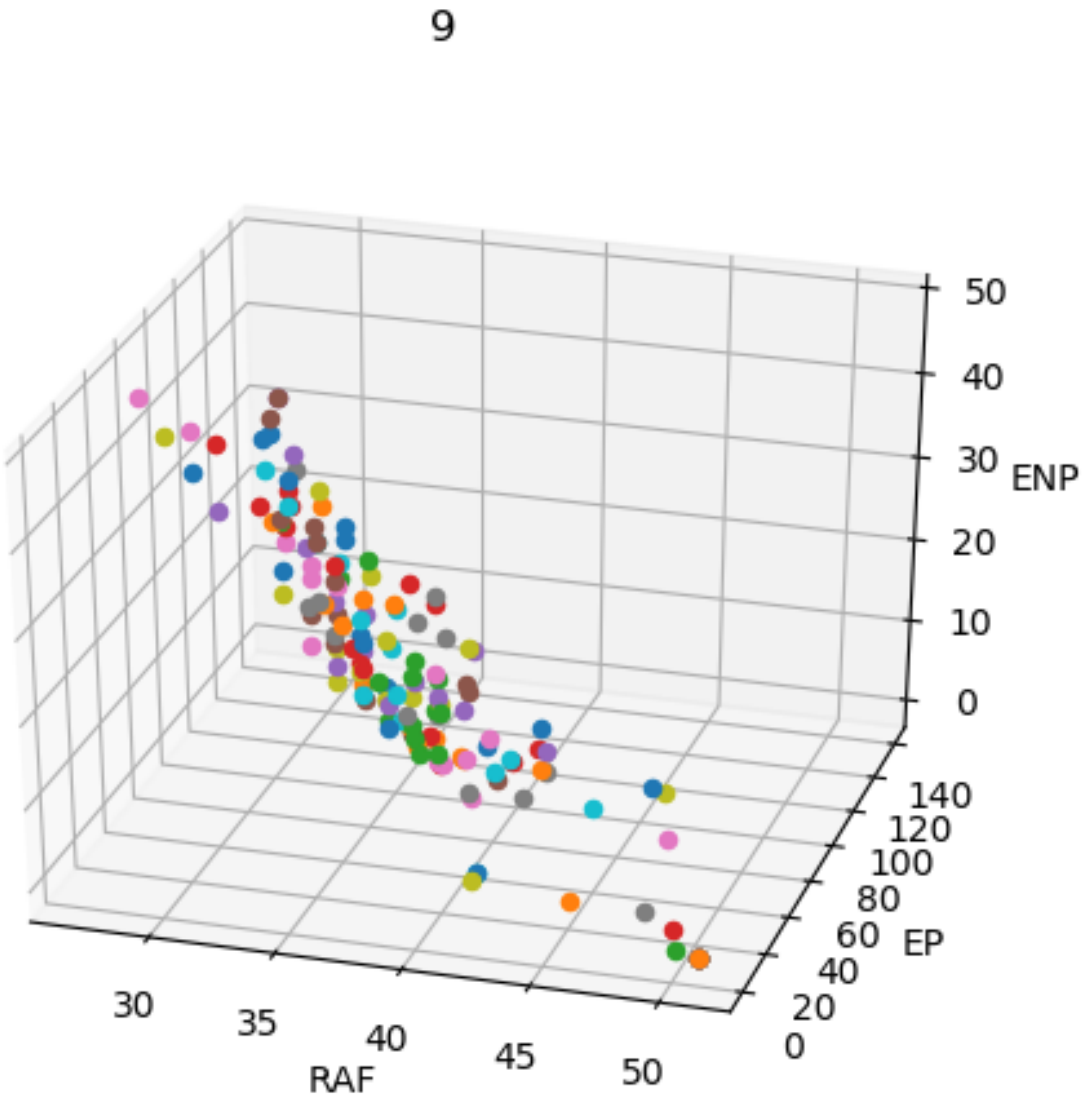
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch2



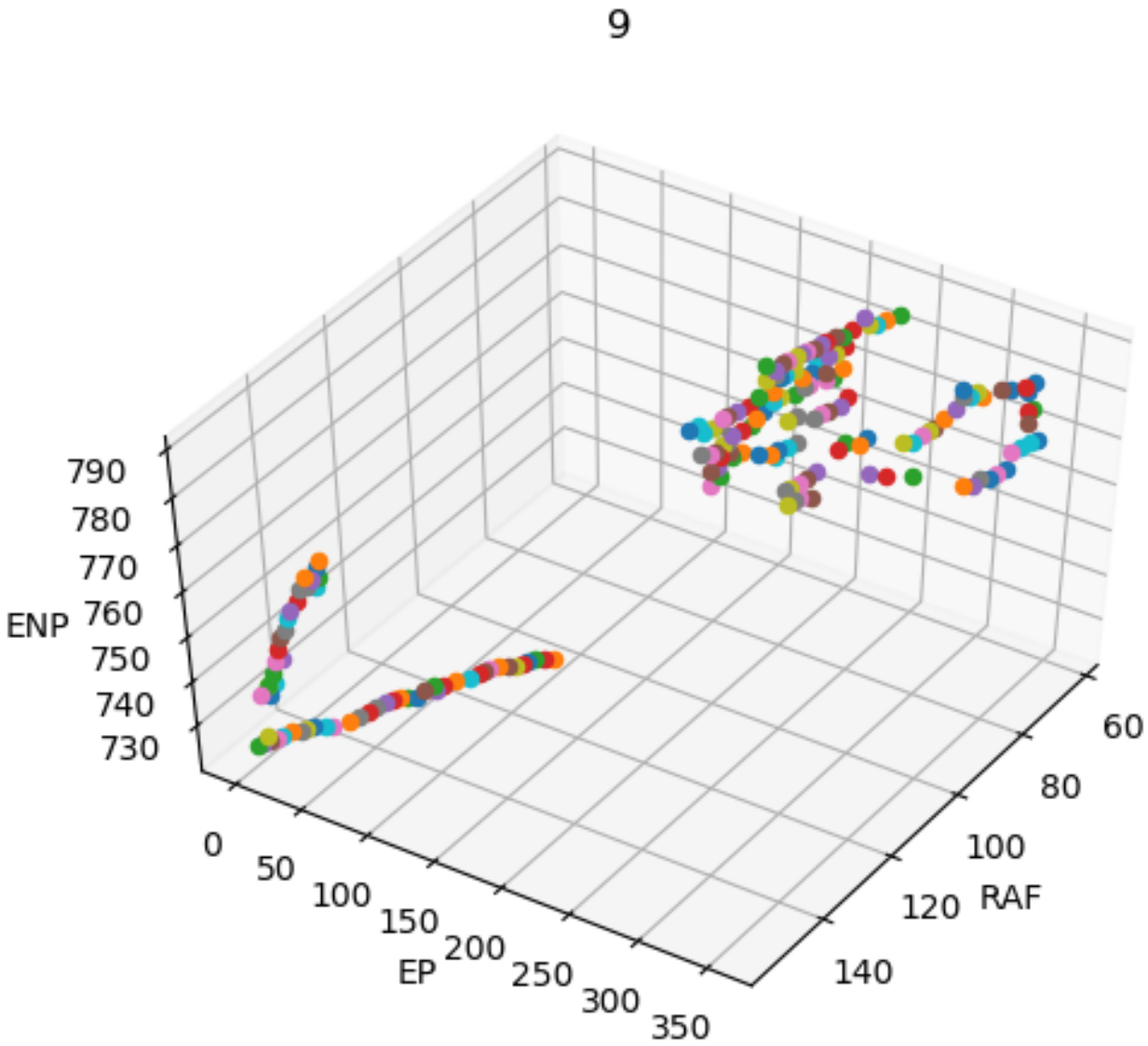
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_EP_ch2



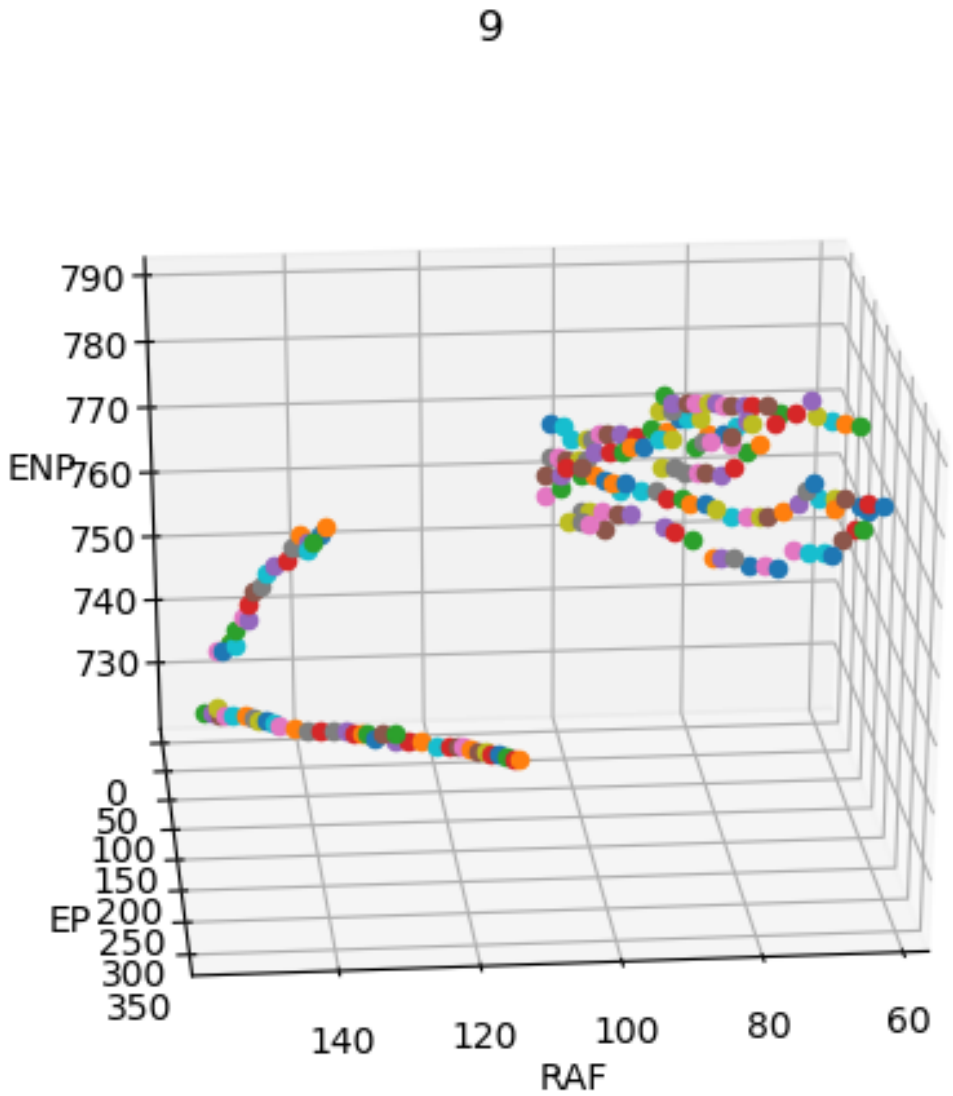
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch2



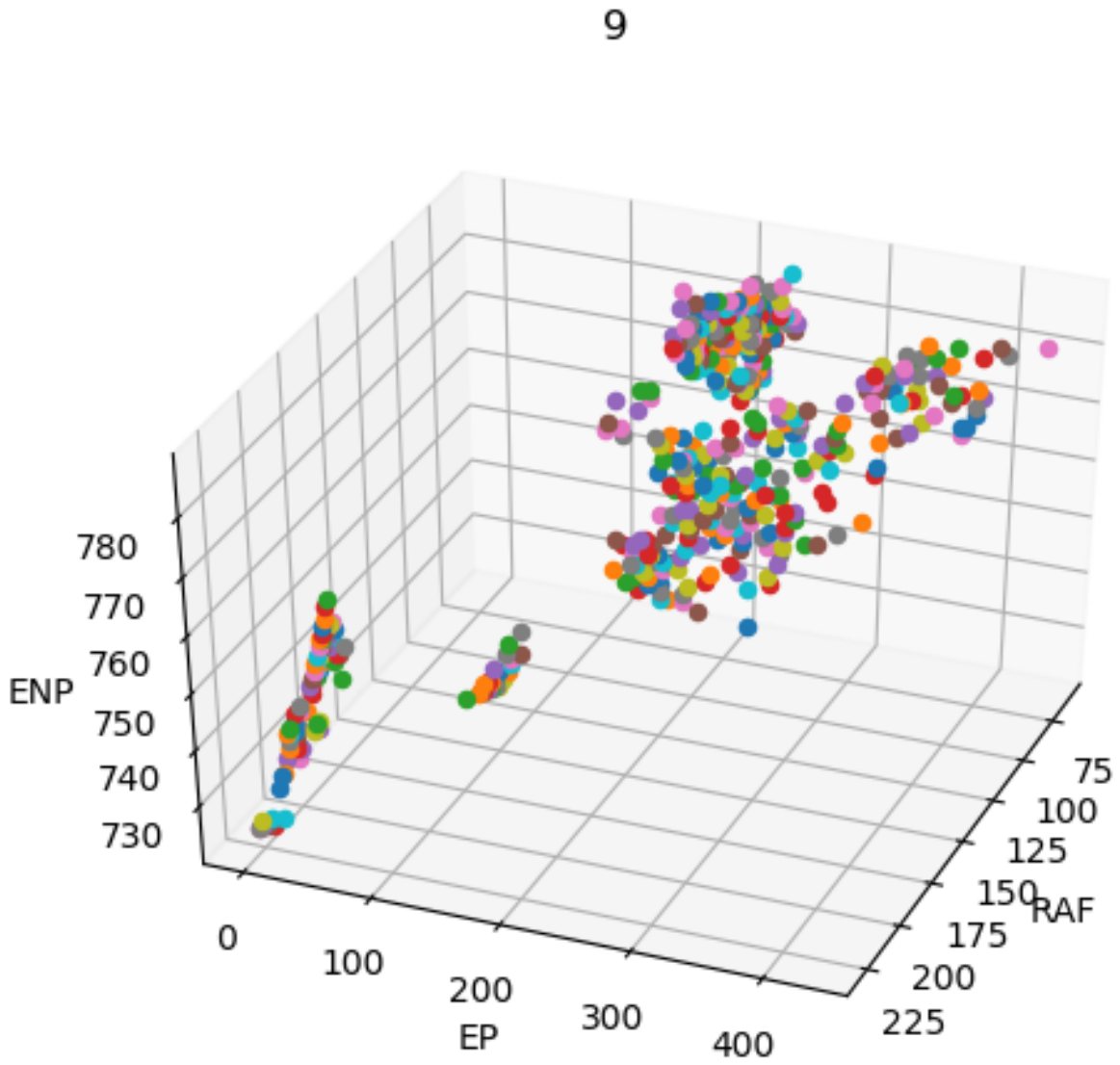
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch1



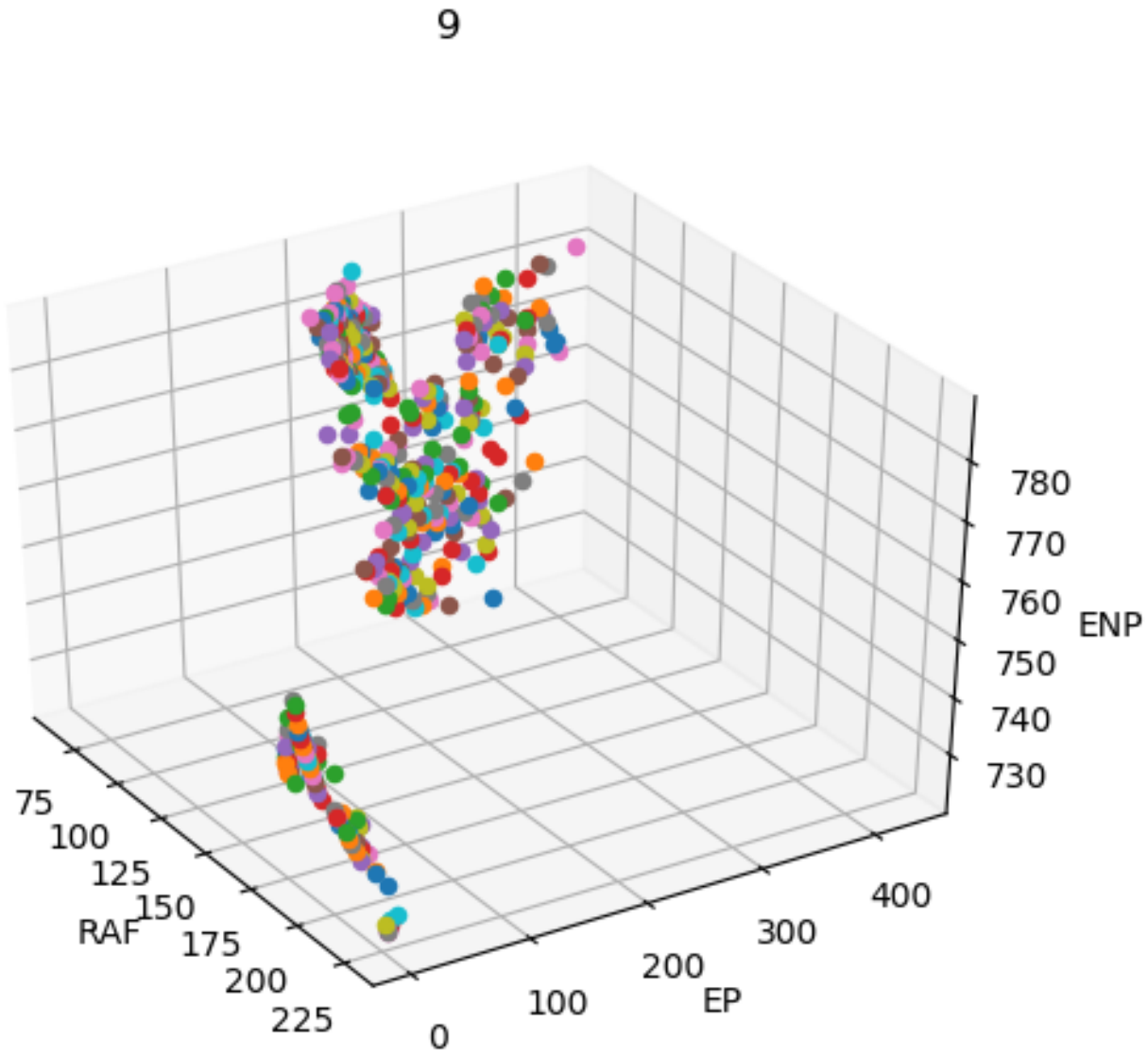
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch1



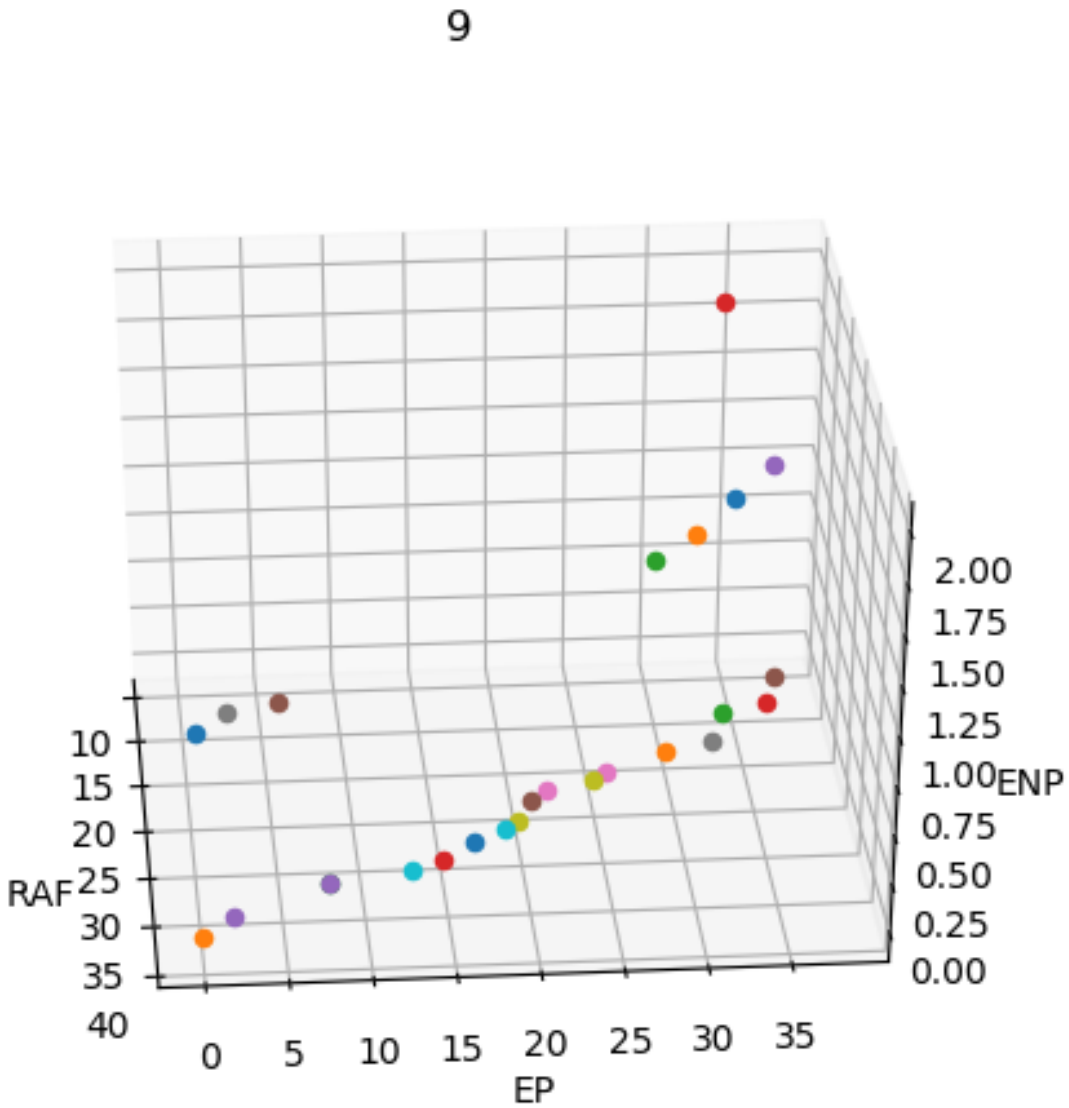
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_EP_ch1



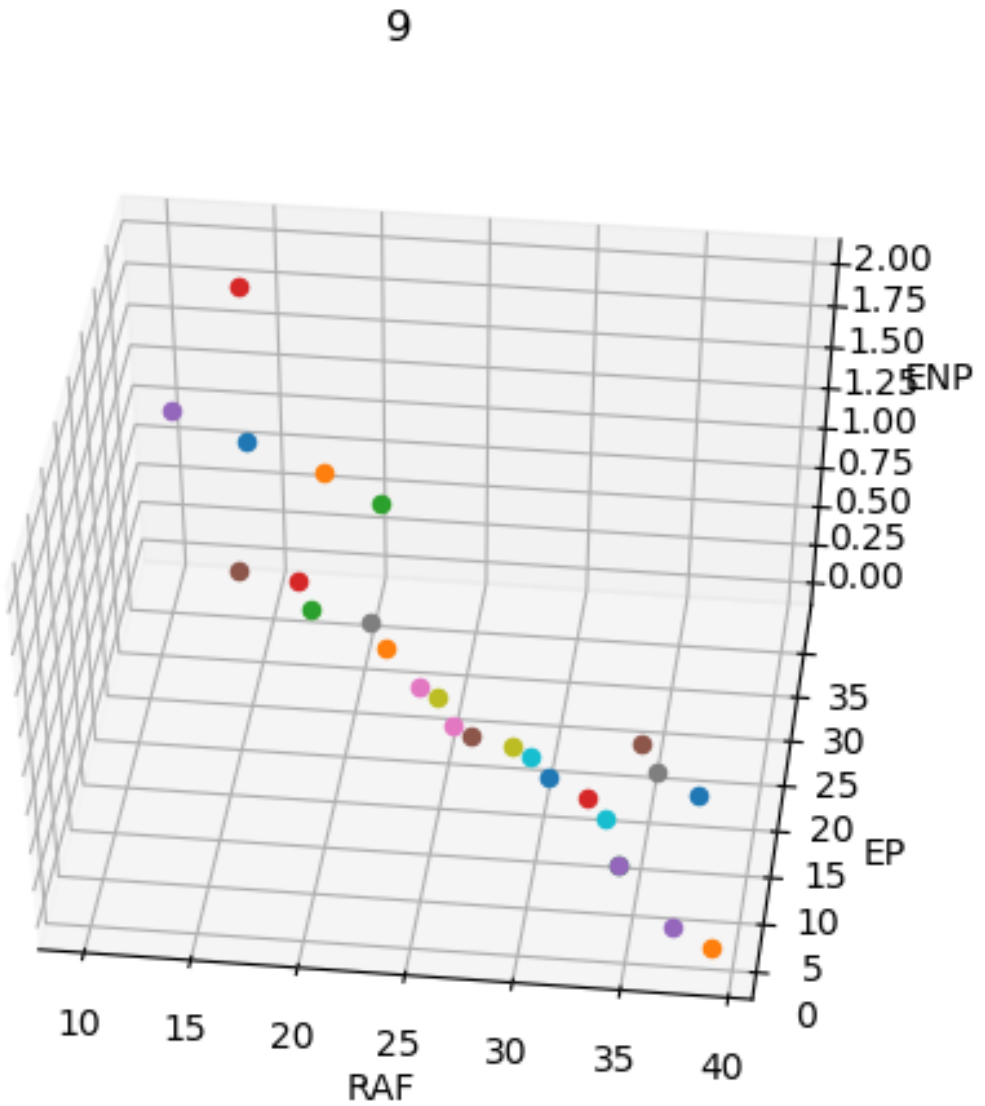
Résultats : Pareto Local Search, instance 064_38_2_RAF_EP_ENP_ch1



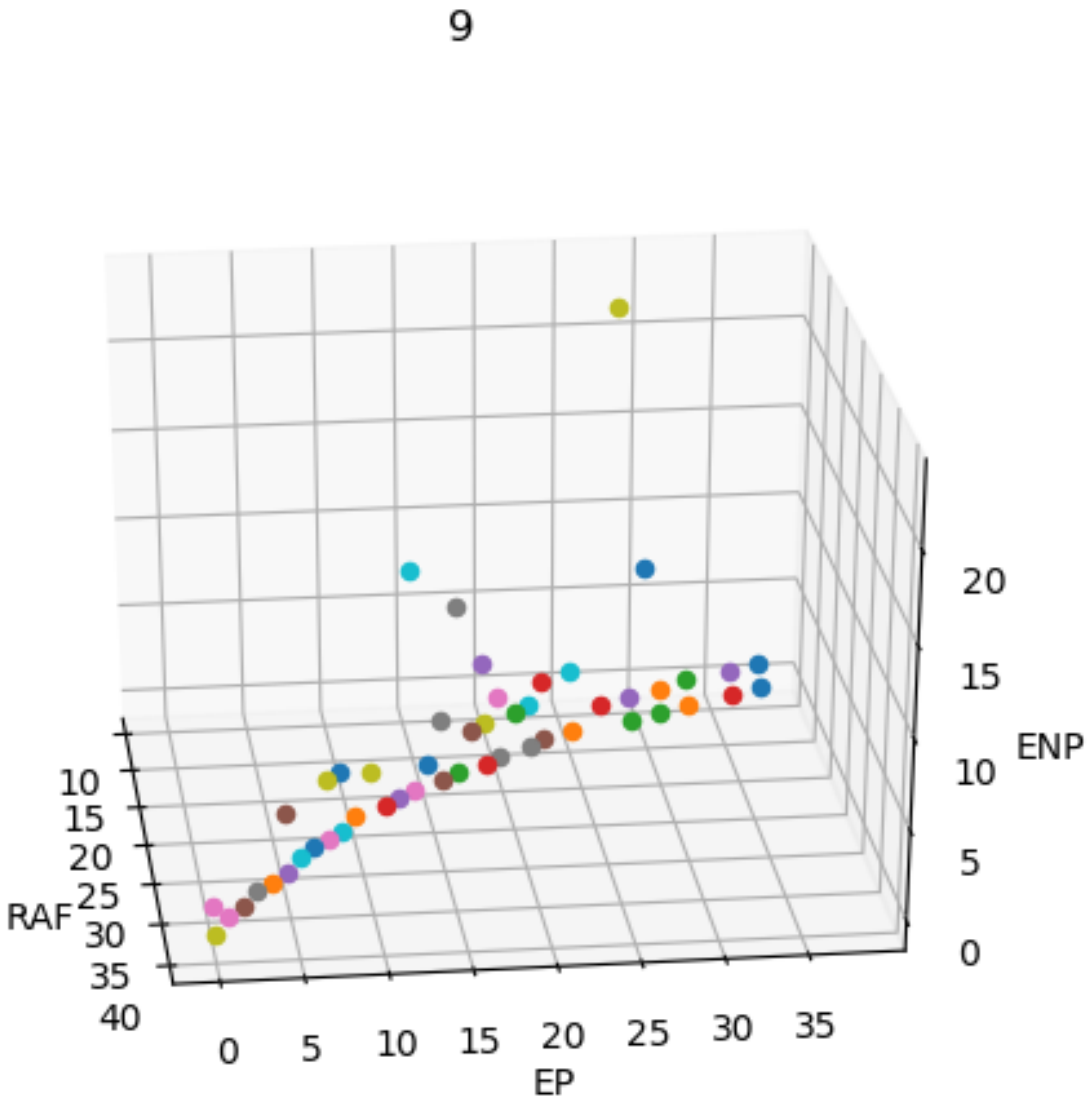
Résultats : Pareto Local Search, instance 022_3_4_EP_RAF_ENP



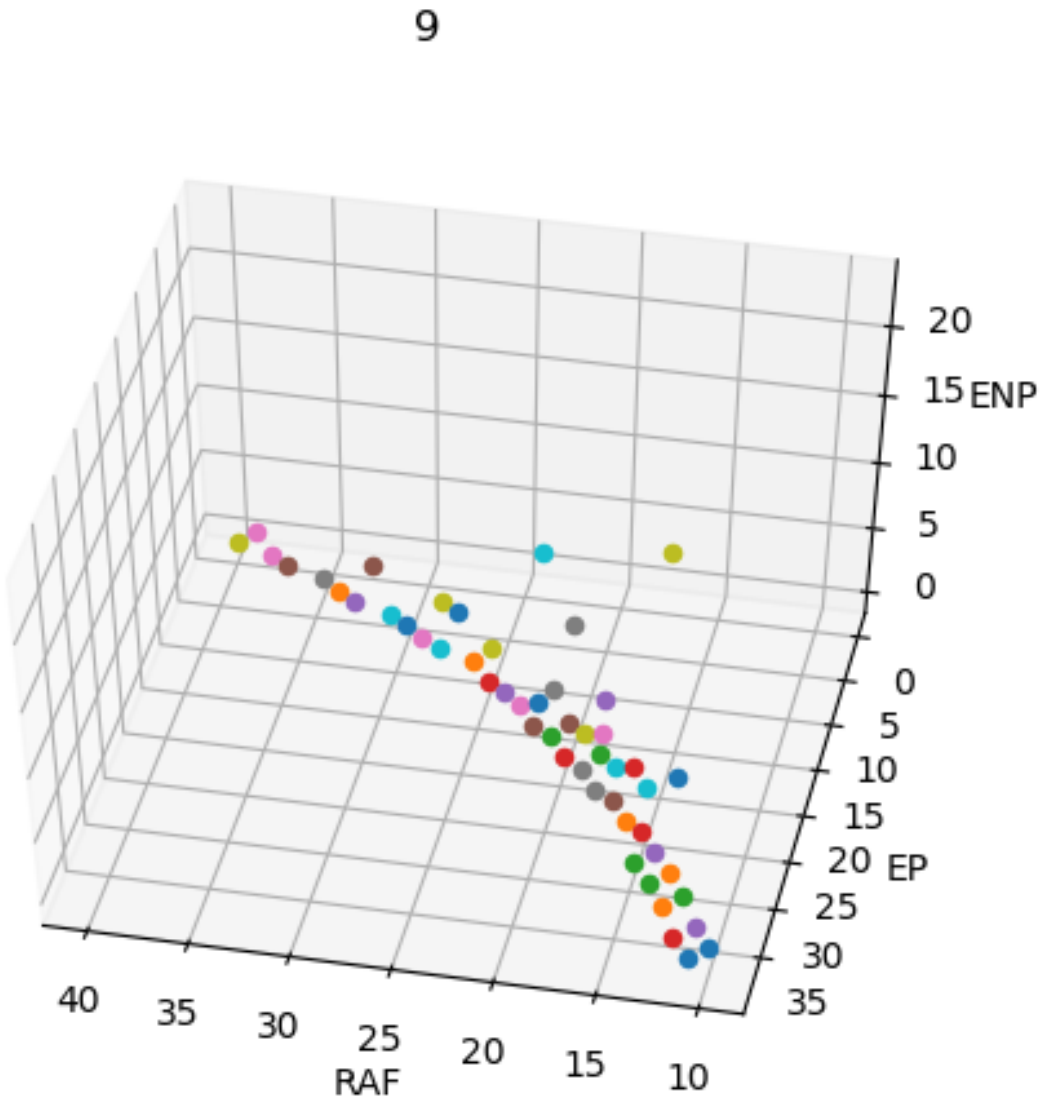
Résultats : Pareto Local Search, instance 022_3_4_EP_RAF_ENP



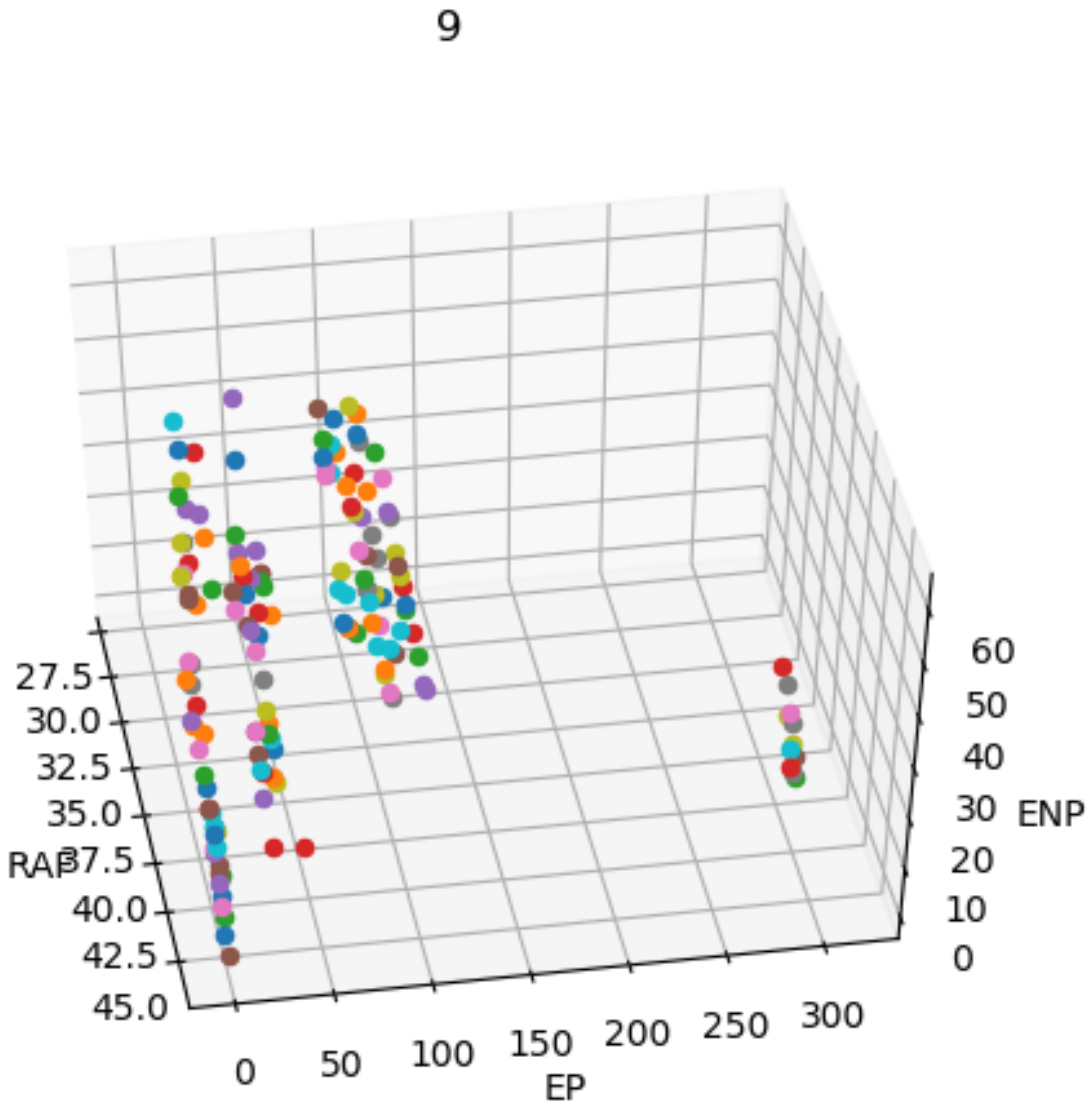
Résultats : Pareto Local Search, instance 022_3_4_EP_RAF_ENP



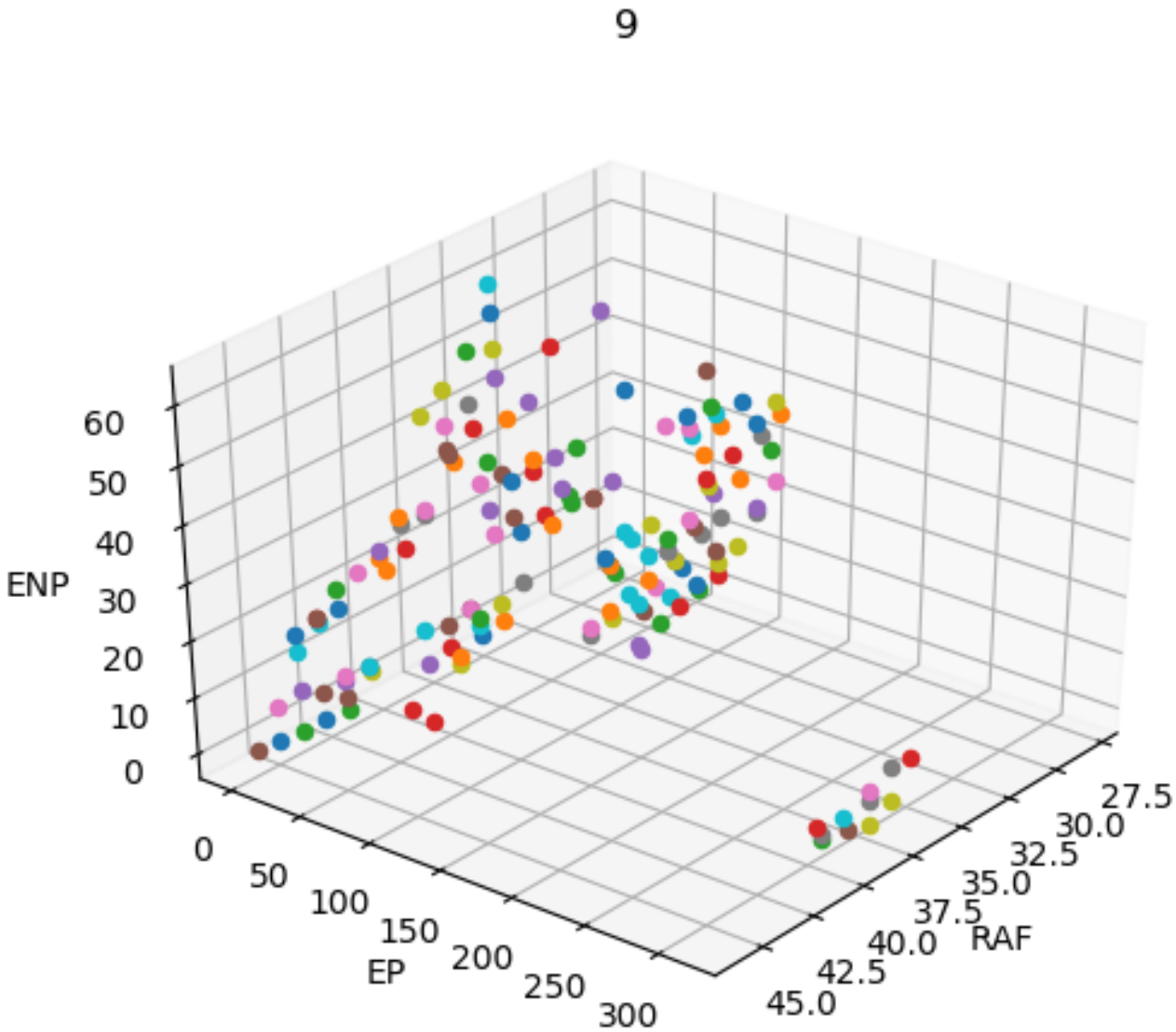
Résultats : Pareto Local Search, instance 022_3_4_EP_RAF_ENP



Résultats : Hybridation, instance 064_38_2_RAF_EP_ENP_ch2



Résultats : Hybridation, instance 064_38_2_RAF_EP_ENP_ch2



Conclusion :

- Évolution : VFLS $\Rightarrow$ Algo Génétique / PLS
- Choix de représentation des résultats : Hyper volume / Graphique 3D
- Paramétrage des algorithmes implémentés
- Résultats au moins aussi bon que ceux du papier
- Quelques pistes d'amélioration

Références I


Andrzej Jaszkiewicz and Thibaut Lust.
Nd-tree-based update : a fast algorithm for the dynamic nondominance problem.
IEEE Transactions on Evolutionary Computation, 22(5) :778–791, 2018.


Alain Nguyen and Van-Dat Cung.
Le problème du car sequencing renault et le challenge roadef'2005.
2005.


Bertrand Estellon, Frédéric Gardi, and Karim Nouioua.
Two local search approaches for solving real-life car sequencing problems.
European Journal of Operational Research, 191(3) :928–944, 2008.

Références II


C Ribeiro, D Aloise, T Noronha, C Rocha, and S Urrutia.
A hybrid heuristic for a real-life car sequencing problem with multiple requirements.
In *Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search*. Tenerife, Spain, 2005.


Matthias Prandtstetter and Günther Raidl.
A variable neihborhood search approach for solving the car sequencing problem.
na, 2005.


Jens Gottlieb, Markus Puchta, and Christine Solnon.
A study of greedy, local search, and ant colony optimization approaches for car sequencing problems.
In *Workshops on Applications of Evolutionary Computation*, pages 246–257. Springer, 2003.