

Rapport

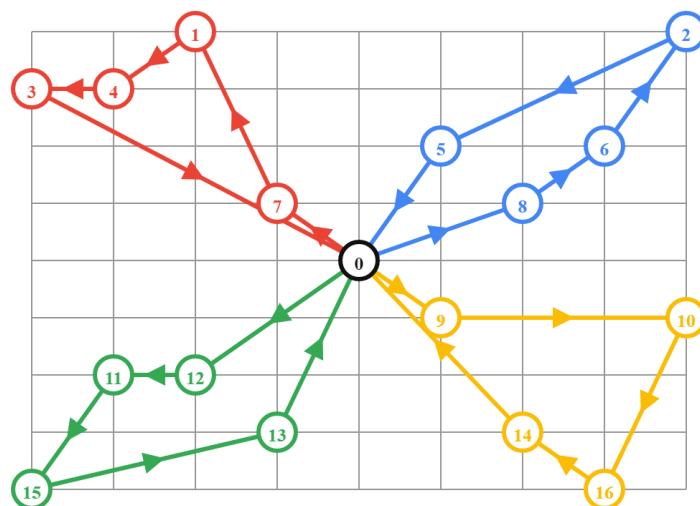
Projet - Capacited Vehicle Routing Problem

PROMO 2020-2023 | 4A Ingénieurs Informatique Apprentissage

Lien repository GitHub : <https://github.com/Kawow2/TravellingSalesManV2>

Objectif 

L'objectif est de trouver des solutions au problème du "CVRP" en utilisant 2 météheuristiques parmi : **le recuit simulé**, **la méthode Tabou** et **les algos évolutionnaires**.



Sommaire

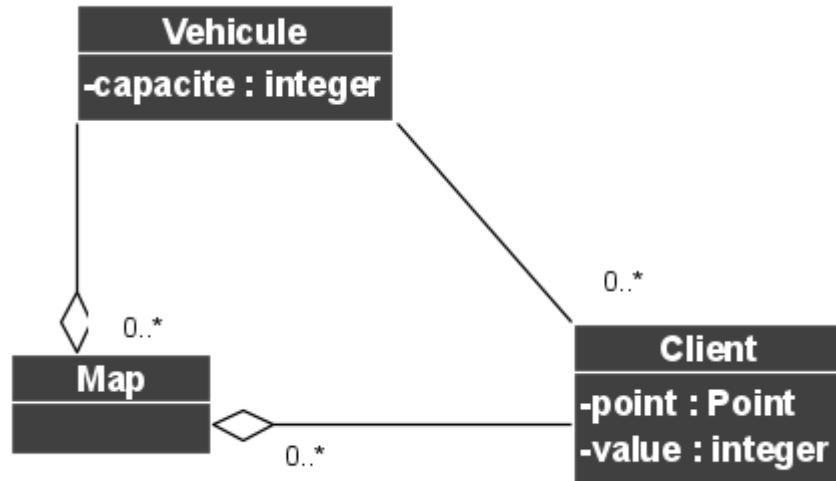
Sommaire 	1
Modélisation du problème & structuration du code 	5
Min. véhicules à utiliser pour les jeux de données 	7
Générateur aléatoire de solutions 	11
Opérateurs de voisinage & opérateurs évolutionnaires 	12
Les transformations locales intra-routes	12
2-OPT	12
Relocate	12
Exchange	13
Les transformations locales inter-routes	13
Relocate	13
Exchange	13
Implémentation métaheuristiques et tests sur notre dataset 	14
Recuit simulé	14
Tabou	16
Comparaison algorithmes 	17
	99
Analyse de nos résultats 	100
Fitness finale	100
Temps d'exécution	101
Modification de la valeur de MU	102
Solutions générées par tabou	103
Différence avec le nombre de véhicule minimal	104

Modélisation du problème & structuration du code

Pour modéliser le problème “**Capacited Vehicle Routing Problem**”, nous avons créé différentes classes java :

- La classe **Client**, qui représente un client à livrer. Il possède un **Point** (*classe Java qui permet de simuler des coordonnées, avec un x et un y*), mais aussi une valeur, qui représente la quantité de marchandises à livrer chez ce **Client**.
- La classe **Véhicule**, qui représente un camion de livraison. Il possède une liste chaînée de **Clients**, qui représente sa liste de clients à livrer, mais aussi une capacité maximale de marchandises, qui correspond à la quantité maximum d'objets que peut contenir le camion.
- La classe **Map**, qui représente l'ensemble le fait de gérer des livraisons. Elle possède une liste de **Clients**, mais aussi une liste de véhicules. Sa liste de clients représente l'état initial de la map, lors de la phase d'initialisation. La liste de véhicules représente la flotte de véhicules de l'on possède pour effectuer les livraisons.
 - La classe abstraite **Algorithme**, qui définit un algorithme de CVR_p (Recuit et Tabu) permet de définir les fonctions importantes des algorithmes (notamment la fonction lancer qui lance les itérations de l'algorithme) avec donc les classes **Recuit** et **Tabu** qui héritent de Algorithme.
 - La classe abstraite **VoisinAlgo**, qui définit les transformations élémentaires. Ainsi, **Relocate OPT**, **ExchangeInter** et **ExchangeIntra** héritent de VoisinAlgo

Diagramme de classes



Min. véhicules à utiliser pour les jeux de données

Pour déterminer le nombre minimal de véhicules à utiliser pour chaque fichier, nous avons commencé par utiliser un algorithme tout simple. Pour chaque fichier, nous avons fait la somme des valeurs de tous les clients, et nous avons divisé cette somme par la capacité totale d'un véhicule (*dans cette exemple, 100*). Nous arrondissons à l'unité supérieure ce qui nous permet d'obtenir suffisamment de camion pour transporter tous les colis.

Voici pour chaque fichier les résultats que nous avons obtenu : (**implémenté dans la classe MinRequiredVeh dans la package Tools**)

Numéro du fichier	Capacité totale	Nombre de véhicule minimale
A3205	410	5
A3305	446	5
A3306	541	6
A3405	460	5
A3605	442	5
A3705	407	5
A3706	570	6
A3805	481	5
A3905	475	5
A3906	526	6
A4406	570	6
A4506	593	6
A4507	634	7
A4607	603	7
A5307	664	7
A5407	669	7
A5509	839	9
A6009	829	9
A6109	885	9
A6208	733	8

A6310	932	10
A6409	848	9
A6509	877	9
A6909	845	9
A8010	942	10
C101	905	10
C201	905	10
R101	701	8

Cependant, après réflexions, nous avons détecté un problème potentiel dans notre algorithme. En effet, il ne prenait pas en compte la quantité de place disponible dans chaque camion.

Prenons un exemple, imaginons que nous devons **livrer des colis à 3 clients différents**. Le **1er client** doit recevoir un colis prenant **98 unités** dans le camion, le **second client** doit recevoir le **même colis (98 unités)** et le **3ème client** seulement **3 unités**.

 **Chaque camion peut contenir 100 unités de colis.**

L'algorithme que nous avons choisi nous retourne comme résultat **deux** camions. En effet, **98 + 98 + 3 = 199**, ce qui nous donnerait 2 véhicules. Cependant, ces trois colis ne peuvent pas tenir dans 2 véhicules. En effet, comme $98 + 3 > 100$, cela est impossible.

 **La plupart des résultats peuvent être juste mais certains sont sans doute faux.**

Nous avons donc décidé de recalculer les nombres de véhicules minimaux, en utilisant l'algorithme de **best-fit (bin-packing)** introduit en cours :

```

for (Client cli : allClientOfFile)
{
    for (Vehicule v : VehiculeList)
    {
        if (cli.package fits in v)
        {
            calcule la capacité restante du véhicule après avoir ajouté le package
        }
    }
    mettre le package dans le véhicule qui possède le minimum de place après avoir
    ajouté le package

    if (client.package doesn't fit in any vehicle)
        créer un nouveau véhicule et ajouter le package
}

```

Voici les résultats que nous avons obtenu :

Numéro du fichier	Capacité totale	Nombre de véhicule minimale
A3205	410	5
A3305	446	5
A3306	541	6
A3405	460	5
A3605	442	5
A3705	407	5
A3706	570	6
A3805	481	5
A3905	475	5
A3906	526	6
A4406	570	6
A4506	593	7
A4507	634	7
A4607	603	7
A5307	664	7
A5407	669	7
A5509	839	9
A6009	829	9
A6109	885	10
A6208	733	8
A6310	932	10
A6409	848	9
A6509	877	9
A6909	845	9
A8010	942	10
C101	905	10

C201	905	10
R101	701	8

(Finalement, cet algorithme ne donne pas de bons résultats. Si le nom des fichiers correspond au nombre de clients et au nombre minimal de véhicules, alors l'algorithme best-fit ne fonctionne pas. Cependant, nous avons trouvé cela intéressant à noter quand même).

L'implémentation de cet algorithme a été effectuée dans la classe Best Bin Packing dans le package Tools.

Ainsi, avec l'algorithme de (best-fit), il nous a permis de rectifier deux valeurs qui étaient fausses avec l'ancien algorithme.

Pour parler rapidement de la complexité du bin-packing, il est en $O(n^2)$, ce qui est assez élevé, mais ce n'est pas dérangeant au vu du nombre de fichiers que nous avons à traiter. Si celui-ci augmente, nous pourrions passer sur une version plus optimisée du best-fit.

(cf. Best-Fit-Heap (voir pseudo-code ci-dessous, source : [Basic Analysis of Bin-Packing Heuristics](#) [Bastian Rieck](#)))

Algorithm 8 Best-Fit-Heap

```

1: for All objects  $i = 1, 2, \dots, n$  do
2:   Perform a Breadth-First-Search in the heap to determine the best bin.
3:   if Best bin has been found then
4:     Pack object  $i$  in this bin.
5:     Restore the heap property.
6:   else
7:     Open a new bin and add the object.
8:   end if
9: end for

```

Générateur aléatoire de solutions

Pour notre générateur aléatoire de solutions nous avons implémenté une fonction du même nom dans notre projet, nous permettant pour un fichier donné de proposer un résultat aléatoirement généré.

Nous procédons donc de la manière suivante pour la génération :

(cf. fonction "generateurSolutionAleatoire" dans le package "Tools" de notre projet)

1	<p>Nous récupérons la liste des clients, que nous copions dans une liste tampon (<i>secondaire</i>) de clients.</p> <p>Nous définissons une nouvelle liste de véhicules. Nous y <i>ajoutons un 1er véhicule</i>.</p> <p>Pour ce 1er véhicule nous y ajoutons le 1er client (entrepôt) de notre liste passé en param. lors de l'initialisation. Et nous supprimons donc ce client de cette liste.</p>
2	<p>TANT QUE notre liste de client n'est pas vide ALORS</p> <ul style="list-style-type: none"> → On tire un nb aléatoire et l'on pioche donc le client x dans notre liste client.
3	<p>SI la capacité restante du véhicule actuel est inférieur à la charge à transporter chez le client ALORS</p> <ul style="list-style-type: none"> → On met à jour la capacité restante du véhicule actuel avec : sa capacité - la quantité à transporter du 1er client (entrepôt) de notre liste tampon de clients. → On ajoute l'entrepôt à notre fin parcours de livraison du véhicule actuel. → On crée un nouveau véhicule et l'ajoute à la liste des véhicules. → On met à jour sa capacité et ajouté l'entrepôt comme premier client (<i>point de départ</i>).
4	<p>On ajoute un client au véhicule actuel ALORS</p> <ul style="list-style-type: none"> → On met à jour la capacité restante du véhicule. → On ajoute le client au véhicule actuel.
5	Une fois sortie de la boucle on ajoute au dernier véhicule le 1er client (entrepôt) de notre liste tampon clients pour finaliser le parcours.
6	On retourne une nouvelle Map composée de notre liste tampon de client MàJ et de notre liste de véhicules .

Opérateurs de voisinage & opérateurs évolutionnaires



Pour réaliser nos algorithmes de voisinage, nous avions besoin de trouver des voisins. Pour trouver ces voisins, nous avons utilisé des opérateurs de voisinage. Ces opérateurs de voisinage permettent de trouver des voisins de l'objet Map, en modifiant un client dans la liste des clients de la carte.

Ces opérateurs de voisinage se distinguent en deux catégories différentes : les transformations locales intra-routes et les transformations locales inter-routes.

Les transformations locales intra-routes

2-OPT

Le principe de l'algorithme de voisinage “2-OPT” est de venir échanger deux arêtes disjointes de la solution actuelle et reconnecter le nouveau parcours ainsi obtenu.

En ce qui concerne l'implémentation, nous avons imaginé la solution suivante, proposer 2 fonctions :

- L'une permettant de réaliser “2-OPT” sur 2 arêtes du parcours d'un camion de la Map concernée.
- L'autre permettant de réaliser “2-OPT” sur toutes les possibilités de tous les parcours de tous les camions de la Map concernée.

Relocate

Le principe de cet algorithme de voisinage “intra-routes” est tout simplement de venir prendre un client d'un circuit d'un camion x et l'insérer dans le parcours de ce camion à une position α différente dans son circuit.

Pour implémenter cela nous avons imaginé la solution suivante, proposer 2 méthodes permettant :

- Pour l'une de réaliser le “relocate” sur un client spécifique dans une tournée spécifique.
- La seconde méthode permet de réaliser tous les “relocate” possible sur tous les clients de tous les camions de la Map.

Exchange

Le principe de cet algorithme de voisinage “intra-routes” est de pouvoir échanger de place, dans le parcours d’un camion, deux clients.

Pour l’implémentation de cette solution nous avons réalisé cela de la façon suivante, en proposant 2 fonctions :

- Pour l’une, réaliser l’action “d’exchange” entre deux clients définis (passer en paramètre de la fonction) sur un parcours d’un camion spécifié.
- Pour l’autre, elle permet de réaliser “l’exchange” sur tous les clients composant le parcours de tous les camions présents dans la Map.

Les transformations locales inter-routes

Relocate

Le principe de cet algorithme de voisinage “inter-routes” est tout simplement de venir prendre un client d’un circuit d’un camion x et l’insérer dans le parcours d’un camion y à la position α dans son circuit.

(Son implémentation est la même que celle mentionnée précédemment dans la section “intra-routes” au souci près que nous passerons en paramètres ici des clients et véhicules pouvant être dans des tournées différentes.)

Exchange

“L’exchange” en “inter-routes” signifie tout simplement de venir prendre 2 clients n’étant pas dans le même circuit et les échanger de place, respectivement à la place à laquelle était le client bougé.

(L’implémentation est similaire à son homologue “intra-route”, au détail près que nous passons en paramètre désormais 2 clients et 2 camions et que notre fonction testant toutes les possibilités testera “l’exchange” entre circuit désormais.

Implémentation métaheuristiques et tests sur notre dataset



Reçut simulé

Nous avons donc décidé d'implémenter parmi les métaheuristiques possible le recuit simulé dans un premier temps. Je vais donc revenir, dans cette partie, sur son implémentation et quelles notions du cours nous avons pu appliquer dans son implémentation.

(screenshot algo recuit simulé cours métahéuristiques)

```

function SIMULATED-ANNEALING( $x_0$  : initial solution,  $t_0$  : initial temperature)
     $x_{min} \leftarrow x_0$ ,  $f_{min} \leftarrow f(x_0)$ ,  $i \leftarrow 0$ 
    for  $k \leftarrow 0$  to  $n_1$  do
        for  $l \leftarrow 1$  to  $n_2$  do
            Randomly select  $y \in V(x_i)$ 
             $\Delta f \leftarrow f(y) - f(x_i)$ 
            if  $\Delta f \leq 0$  then
                 $x_{i+1} \leftarrow y$ 
                if  $f(x_{i+1}) < f_{min}$  then  $x_{min} \leftarrow x_{i+1}$ ,  $f_{min} \leftarrow f(x_{i+1})$  end if
            else
                randomly draw  $p \in [0, 1]$  according to uniform distribution
                if  $p \leq \exp(-\Delta f/t_k)$  then  $x_{i+1} \leftarrow y$ 
                else  $x_{i+1} \leftarrow x_i$ 
                end if
            end if
             $i \leftarrow i + 1$ 
        end for
         $t_{k+1} \leftarrow \mu \cdot t_k$ 
    end for
    return  $x_{min}$ 
end function

```

▷ n_1 changes of temperature
▷ n_2 moves at temperature t_k

▷ Metropolis rule

▷ temperature decrease, with $\mu < 1$

Avant de nous lancer dans l'itération des solutions de la méthode de recuit nous allons tout d'abord initialiser quelques variables utiles à son exécution.

- Un entier “nbTemp” qui sera le nombre d’itération à réaliser dans la 1ère boucle du recuit (nb. températures à tester). Nous avons implanté cela via la formule vu en cours :
$$[\log(\log(0.8)/\log(0.01))] / \log(\mu) \times 3$$
- Un double “température”.
- Une liste de double qui servira de base de stockage des solutions retenues par notre recuit.
- Une Map qui est le clone de celle passée en paramètre (il s’agit d’un buffer)
- Un double “fitnessBestSolution” qui prend à l’initialisation la valeur de la fitness de la map passée.
- Une Map “nextVoisin” qui stockera le nouveau voisin choisi par recuit.

Puis nous entamons donc l’itération sur “nbTemp” fois, itérant sur “N2” itérations (nombre que nous définissons à la main).

1. A chaque itération “nbTemp” nous recalculons la température avec la formule vu en cours : $newTempérature = \mu \times température$
2. A chaque itération “N2” nous faisons :
 - 2.1. Nous clonons la map de la précédente itération dans notre buffer de Map.
 - 2.2. Nous sélectionnons au hasard un algorithme de voisinage à utiliser.
 - 2.3. Nous sélectionnons au hasard deux clients à échanger.
 - 2.4. Nous lançons notre algo de voisinage et calculons sa fitness.
 - 2.5. Si sa fitness est meilleur que la précédente solution ou si nous passons la phase de repêchage (formule : $proba \leq e^{\Delta fitness / température}$).
 - 2.5.1. nous assignons à “nextVoisin” le nouveau voisin calculé.
 - 2.5.2. sinon nous gardons la solution précédente.
 - 2.6. Finalement nous assignons “nextVoisin” à map (qui sera la Map retournée comme résultat de notre recuit simulé).

Tabou

Dans un second temps nous avons donc implémenté la météuristiche Tabu. Nous allons donc revenir sur son implémentation et la façon dont nous avons imaginé cette solution à l'aide de nos connaissances et cours d'optimisation discrète.

(screenshot algo Tabu cours météuristiche)

```

function TABU-SEARCH( $x_0$  : initial solution)
     $x_{min} \leftarrow x_0$ ,  $f_{min} \leftarrow f(x_0)$ ,  $i \leftarrow 0$ 
     $T \leftarrow \emptyset$ 
    for  $i \leftarrow 0$  to maxIter do
         $C \leftarrow V(x_i) - \{m(x_i)/m \in T\}$ 
        Select  $x_{i+1} \in C$  such as  $f(x_{i+1}) = \min\{f(y)/y \in C\}$   $\triangleright x_{i+1} = m(x_i)$ 
         $\Delta f \leftarrow f(x_{i+1}) - f(x_i)$ 
        if  $\Delta f \geq 0$  then Add  $m^{-1}$  in  $T$  end if
        if  $f(x_{i+1}) < f_{min}$  then  $x_{min} \leftarrow x_{i+1}$ ,  $f_{min} \leftarrow f(x_{i+1})$  end if
    end for
    return  $x_{min}$ 
end function

```

Avant de nous lancer dans l'itération des solutions de la méthode de Tabu nous allons tout d'abord initialiser quelques variables utiles à son exécution.

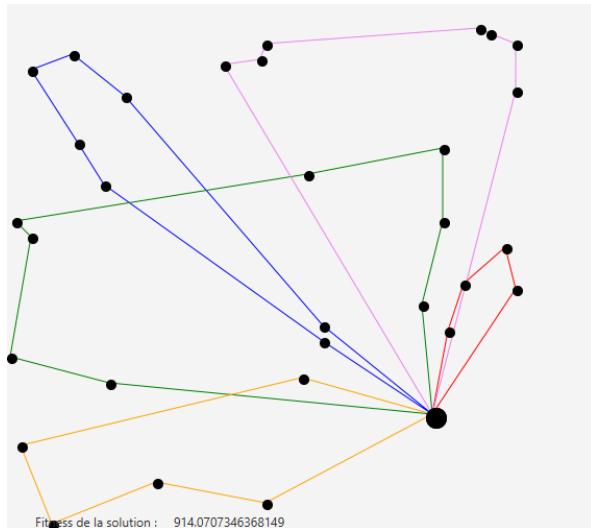
- Un entier “tailleTabu” représentant donc la taille du tableau Tabu excluant les solutions sur lesquelles nous ne pouvons pas revenir durant l’algo.
 - Un boolean “TABUCCONDITION” permettant d’itérer tant que l’algo / parcours Tabu n’est pas terminé.
 - Une liste de double “listToWrite” représentant la fitness des solutions de voisinage générées.
 - Un entier “count” comptant le nombre d’itérations de notre algo Tabu.
 - Une Map “bestSolution” contenant la meilleure solution possible pour notre fichier.
 - Une Map “bestCandidat” qui correspond à la solution calculée en cours à comparer avec d’autres.
 - Une liste de double “tabuClients” contenant la fitness des candidats à Tabu dans l’itération en cours.
1. Tant que notre Condition “TABUCCONDITION” vaut TRUE nous itérons de la manière suivante :
 - 1.1. Pour chaque algos de voisinage nous ajoutons dans une liste de Map “voisin” les solutions générées.
 - 1.2. SI “bestCandidat” est nul alors cela veut dire que nous avons terminé le parcours de Tabu et donc que notre solution précédente et la meilleure. (Nous ne réitérerons donc pas)
 - 1.3. Pour chaque solution dans notre liste nous allons comparer cette dernière avec la solution vérifiée et meilleure précédente afin d’extraire le “bestCandidat”.
 - 1.4. On calcule la fitness de “bestCandidat”
 - 1.5. SI cette fitness est meilleure que la “bestSolution” alors elle le devient.
 - 1.6. On ajoute ce candidat à notre liste “tabuClients”.

Comparaison algorithmes ▼

Pour comparer les deux algorithmes (Tabu et recuit), nous avons donc décidé de lancer tabou avec une taille de 10 et Recui avec deux températures différentes : 50 et 200.

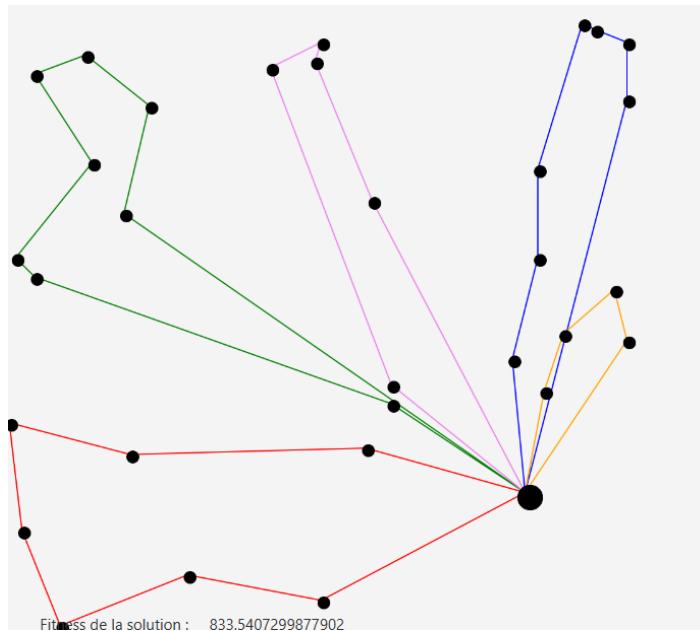
A3205

Tabou Taille 10



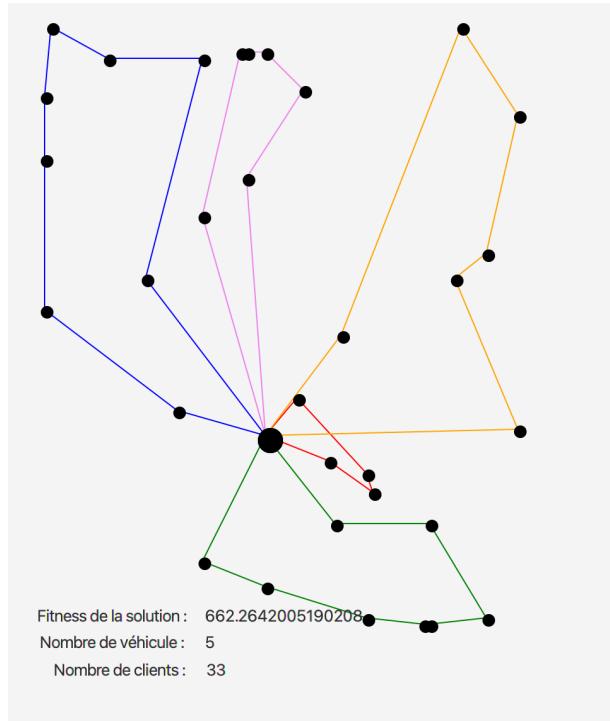
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
287ms	914	386129	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1084ms	833	839916	5

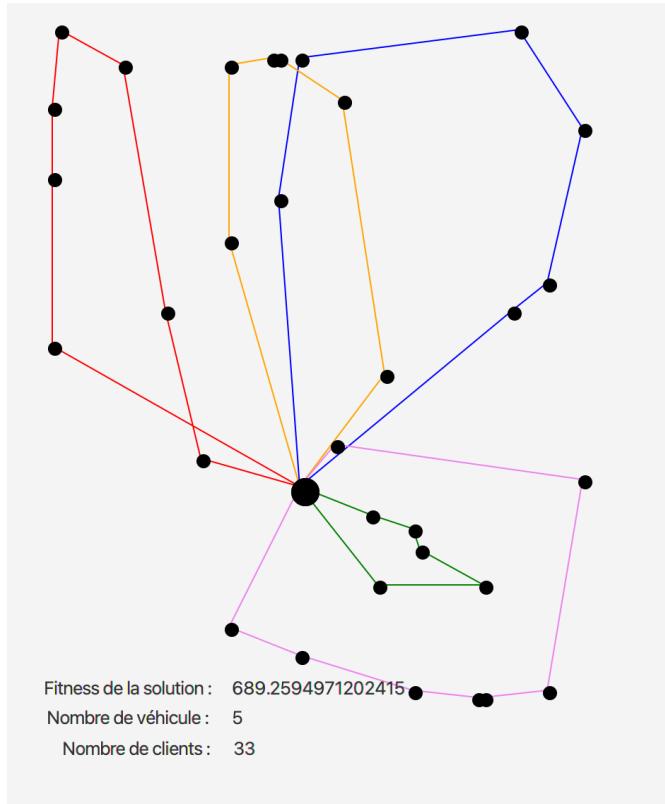
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
629ms	662	839916	5

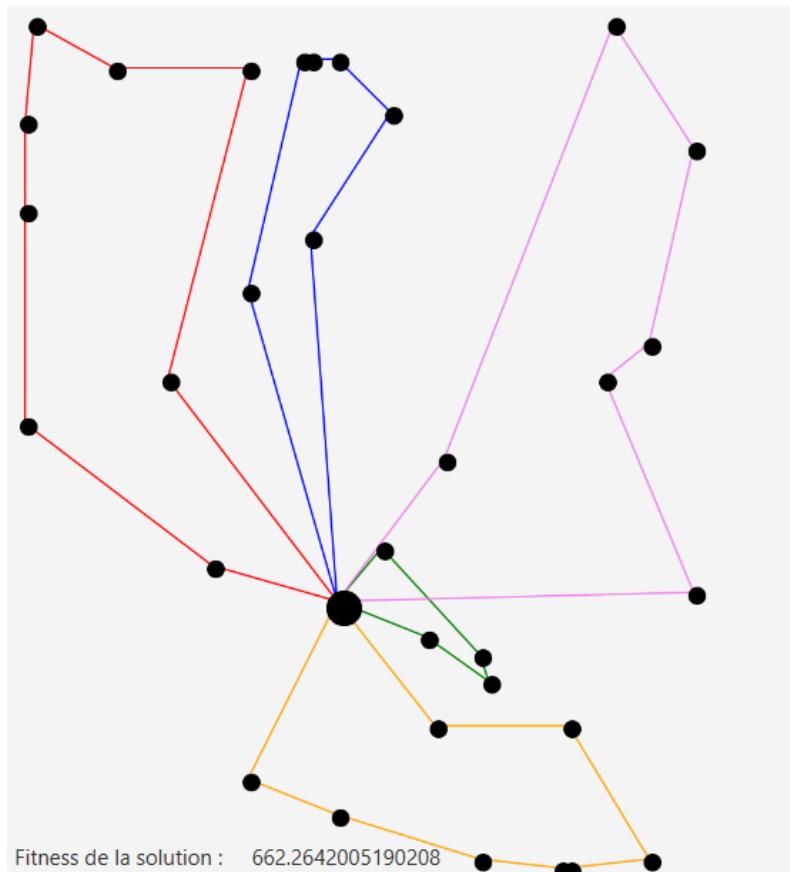
A3305

Tabou Taille 10



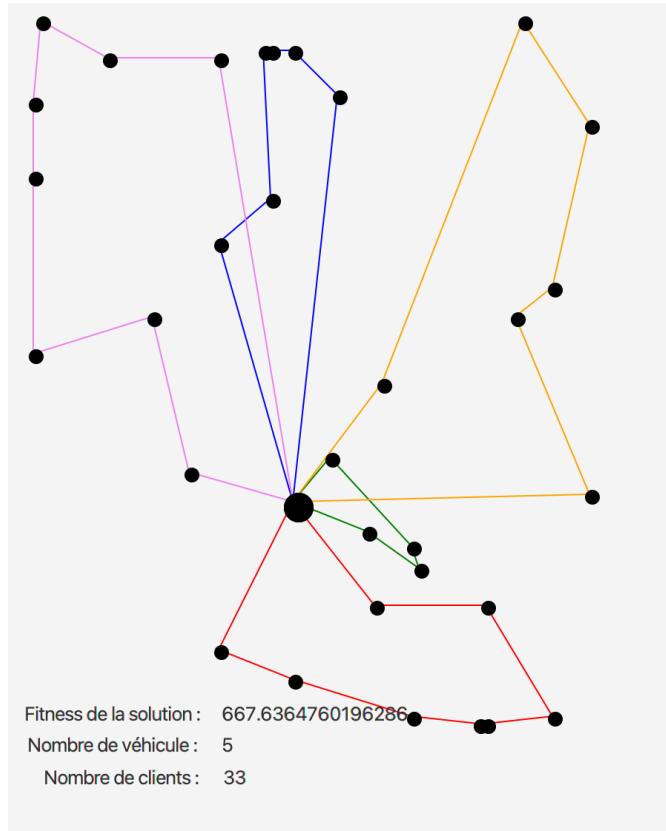
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
314ms	689	168966	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1333 ms	662	839916	5

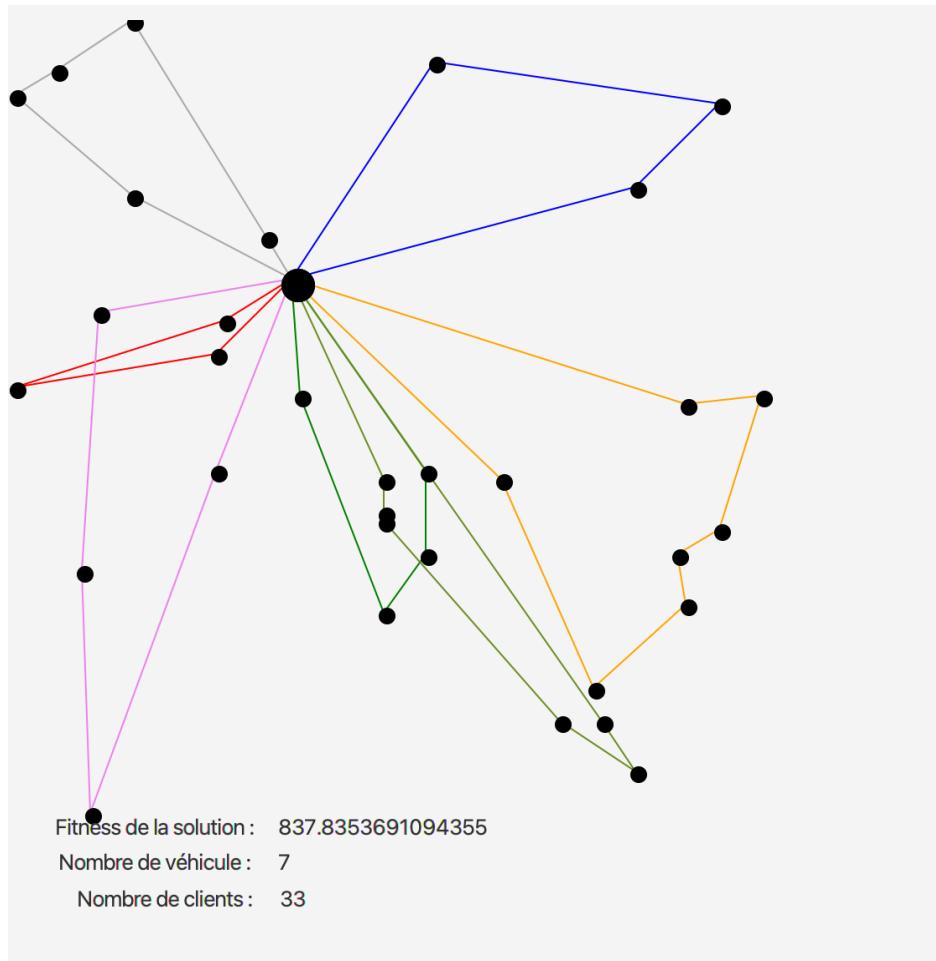
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
583ms	667	839916	5

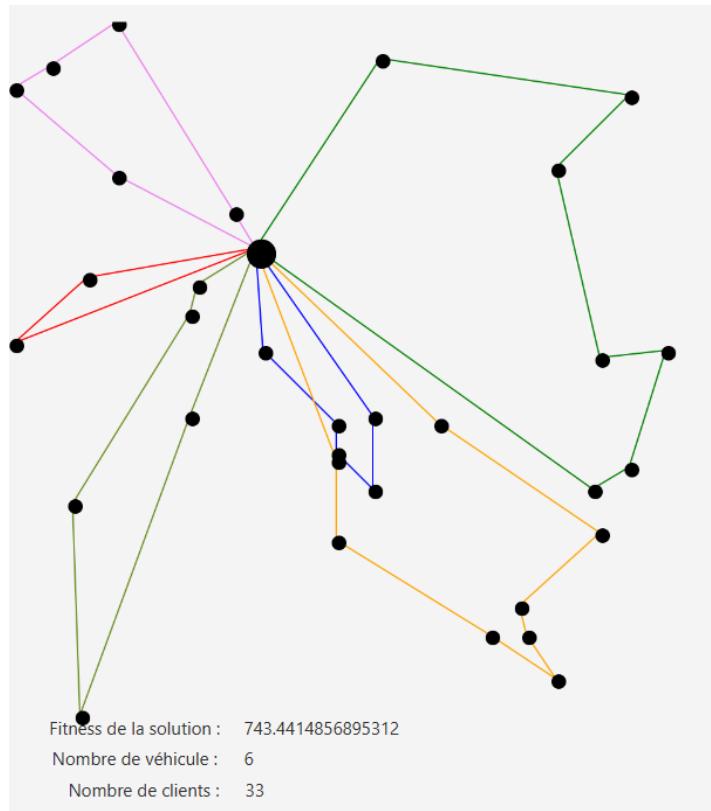
A3306

Tabou Taille 10



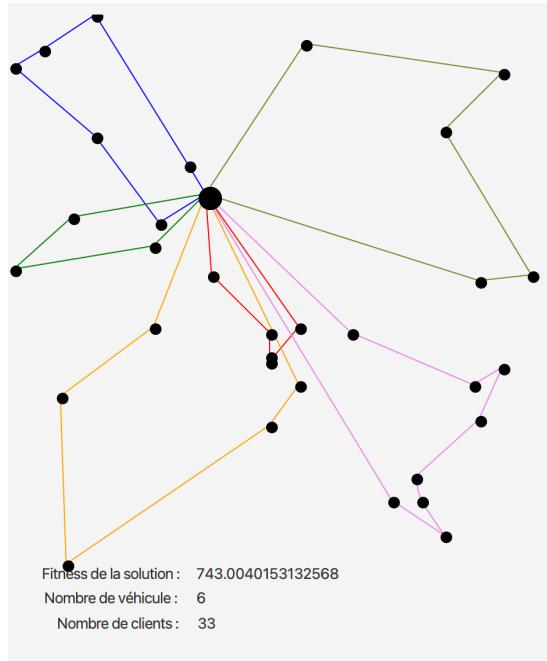
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
439ms	837	294642	7

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1155	743	839916	6

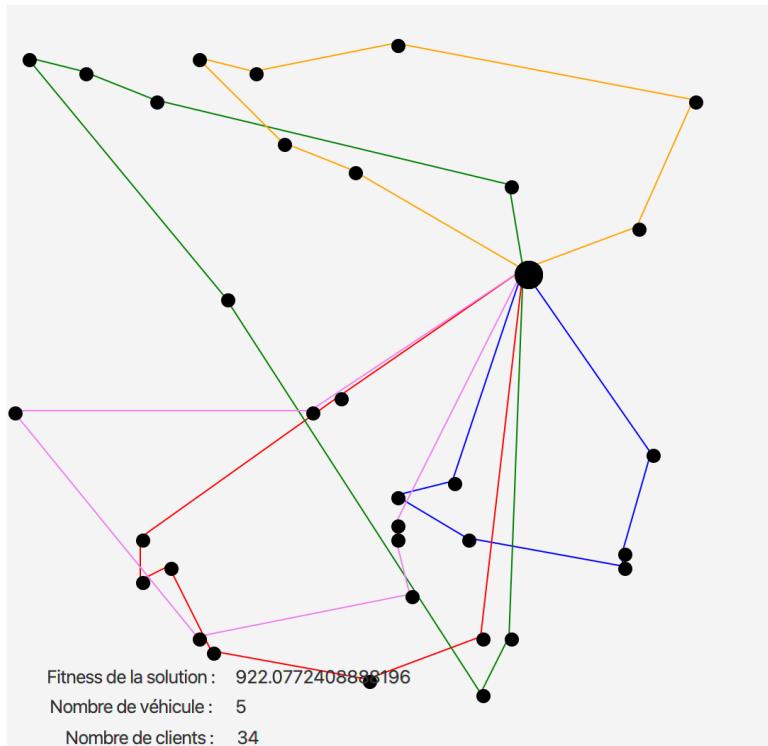
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
624ms	743	839916	6

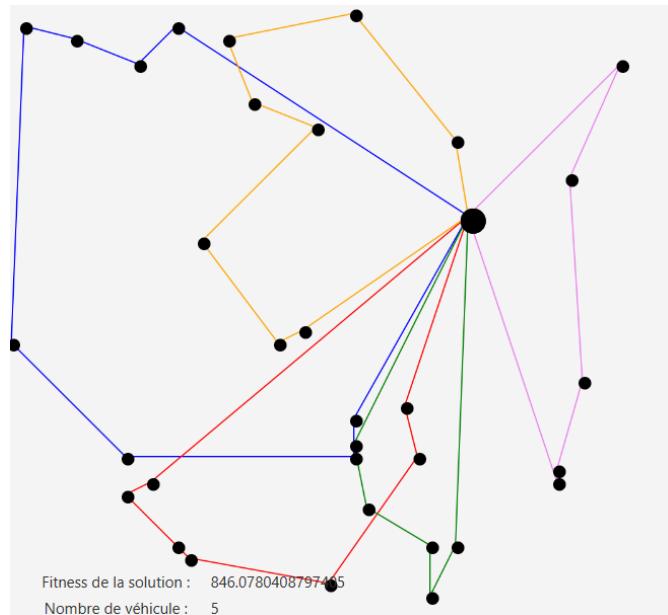
A3405

Tabou Taille 10



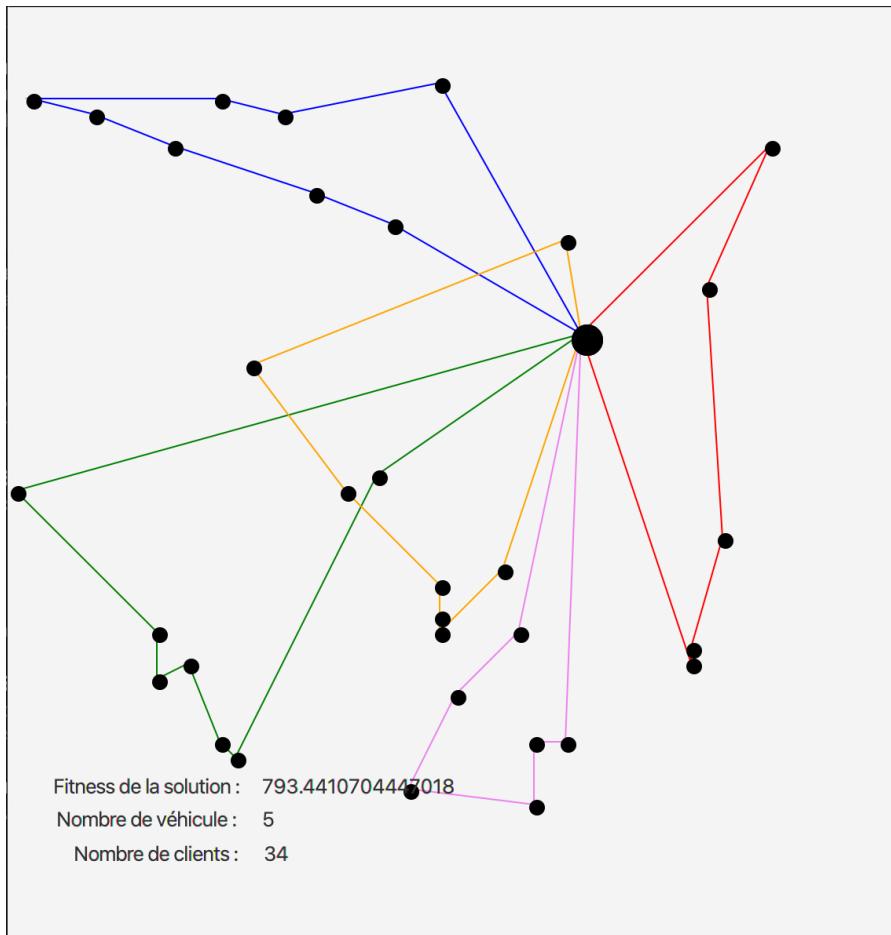
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
635ms	922	839916	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1161	846	839916	5

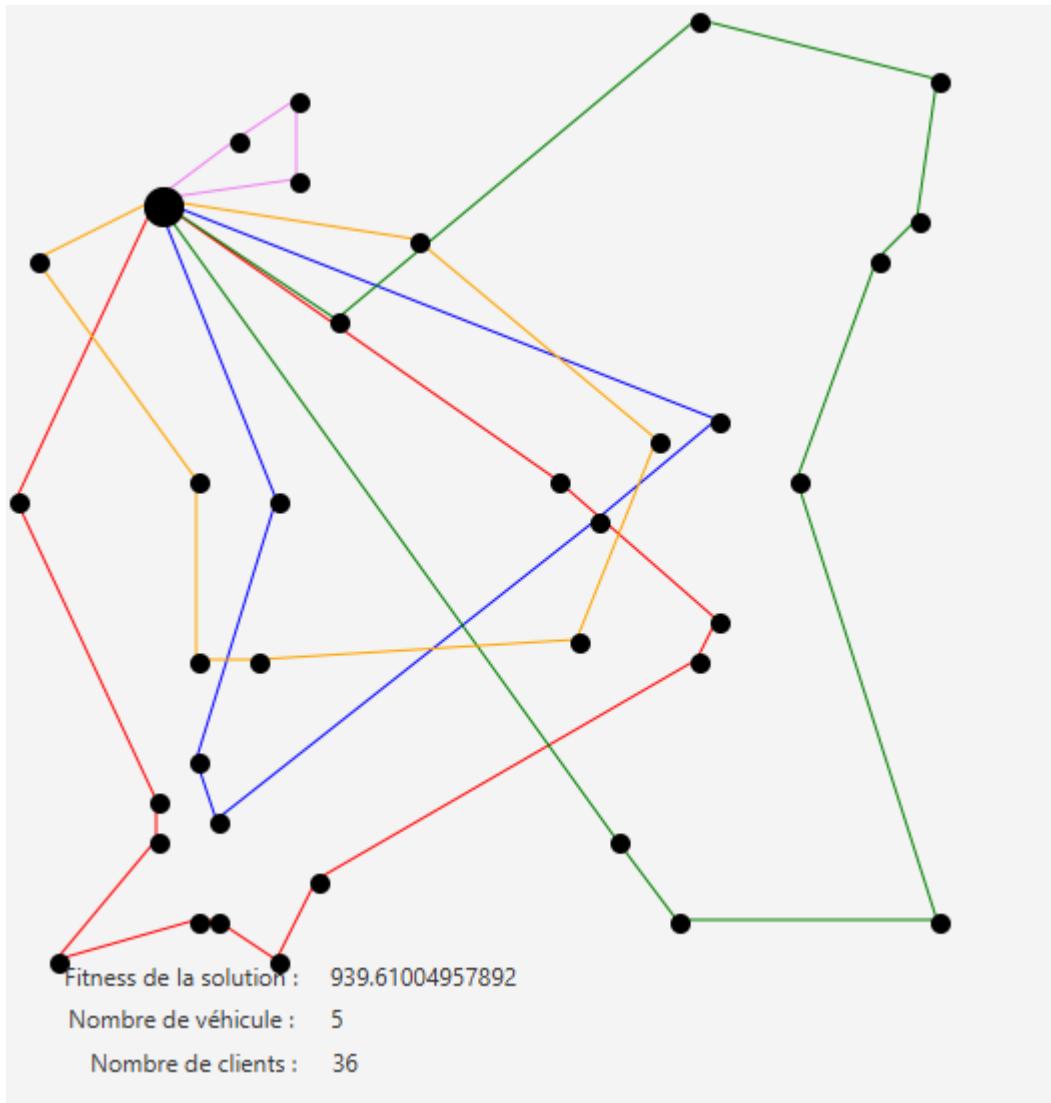
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
590ms	793	839916	5

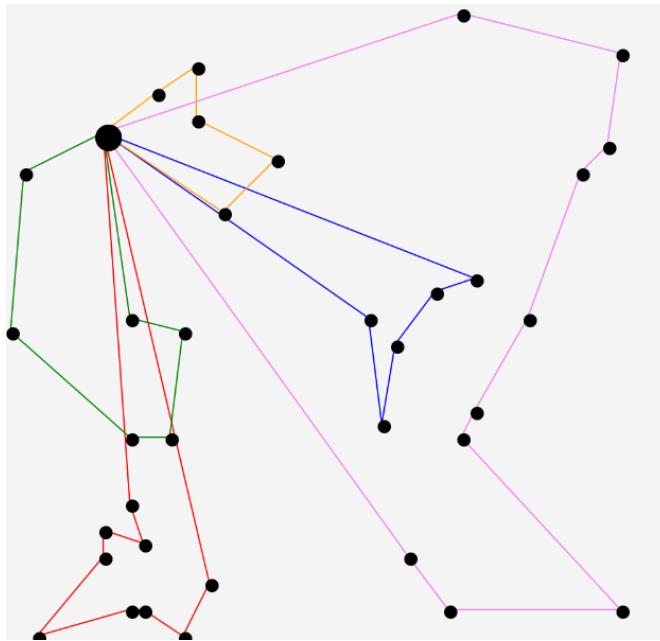
A3605

Tabou Taille 10



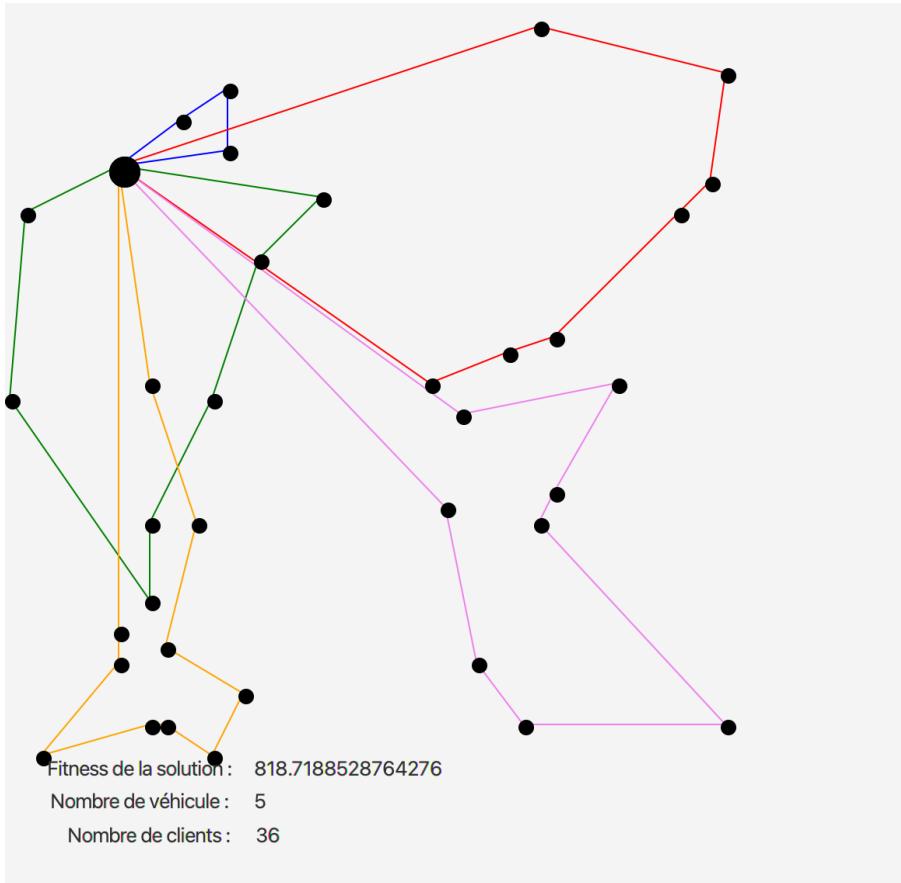
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
684	939	156970	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1212	823	839916	5

Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
669ms	818.71	839916	5

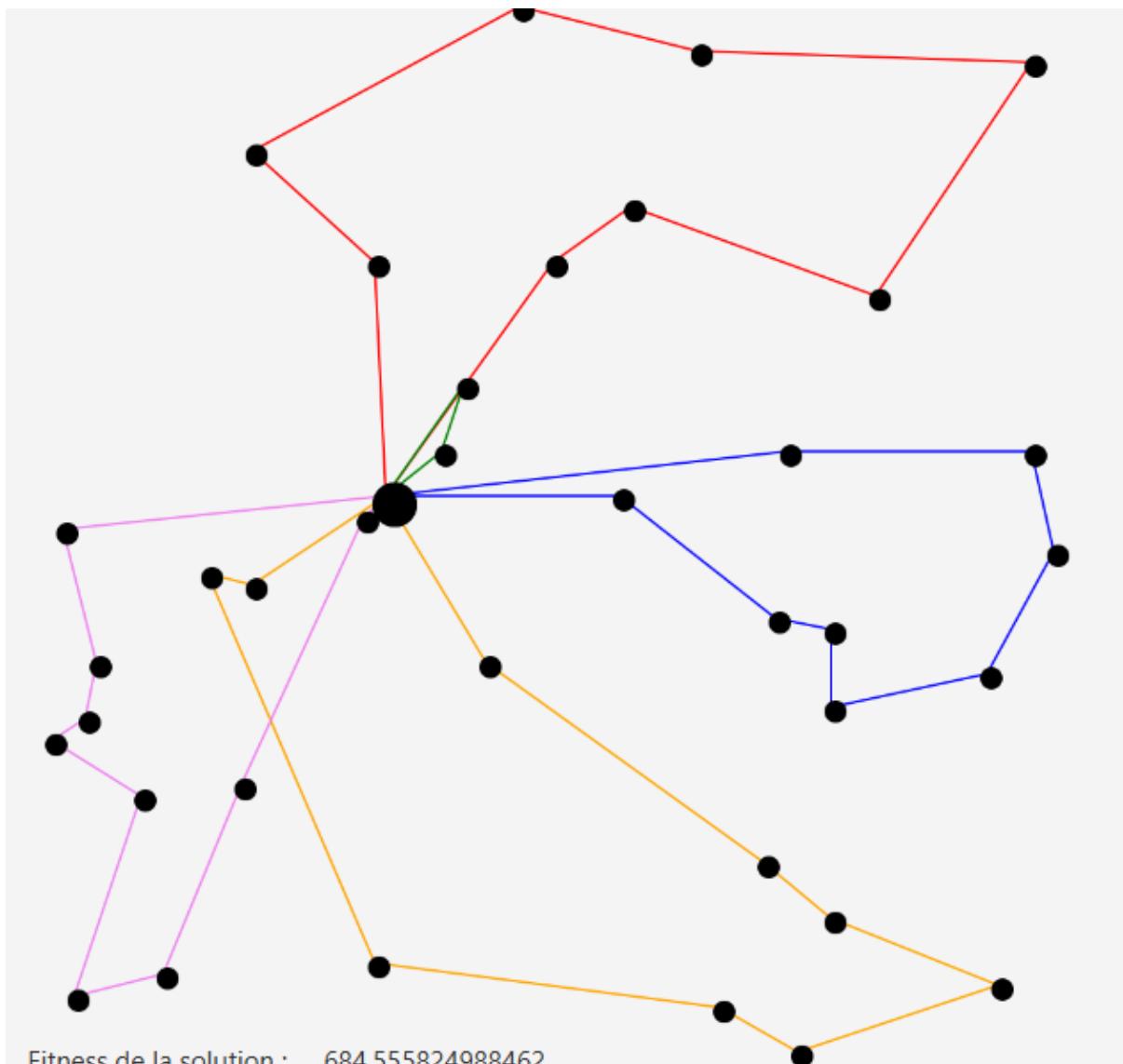
A3705

Tabou Taille 10



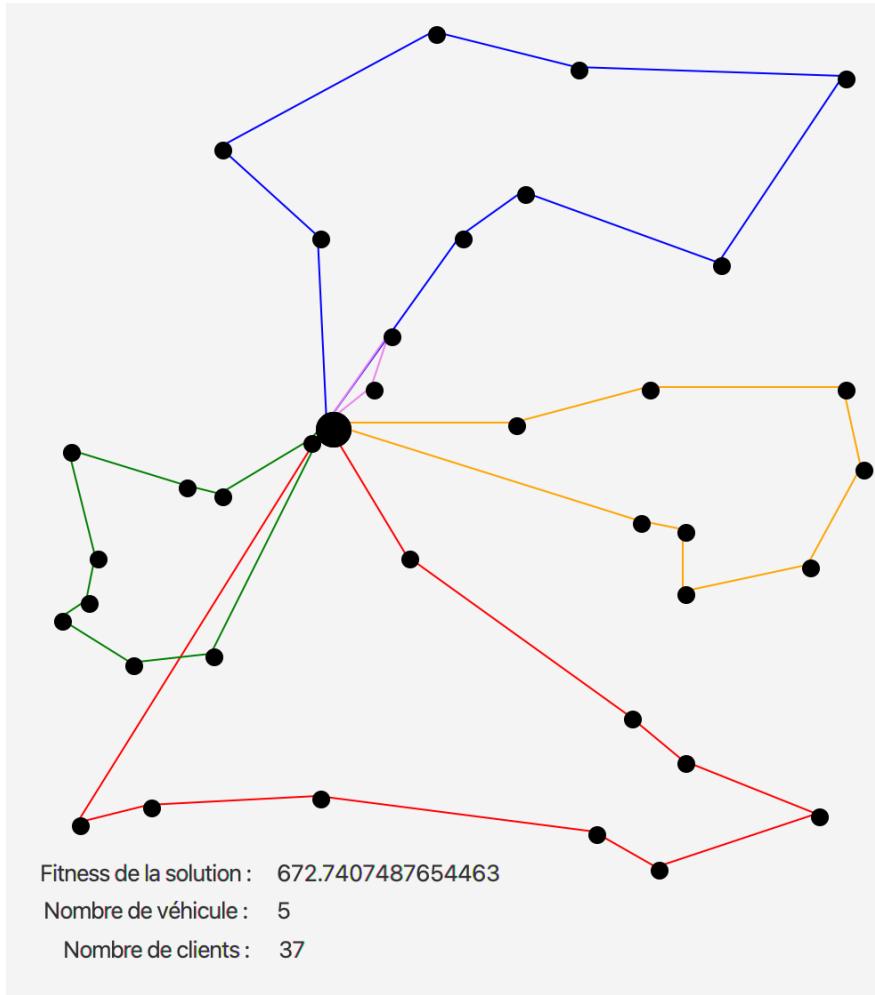
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2313	724	891610	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1272	684	839916	5

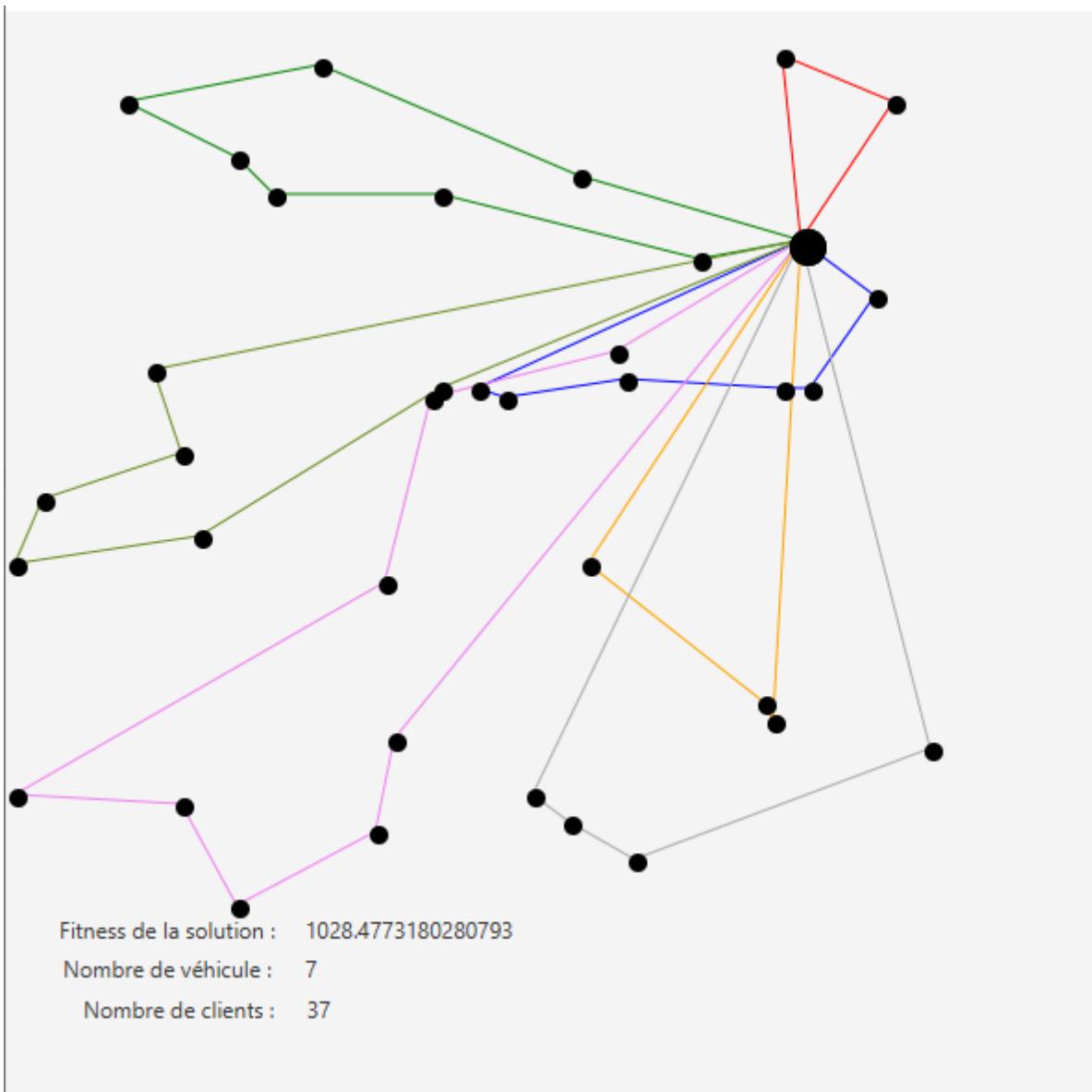
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
633ms	672.74	839916	5

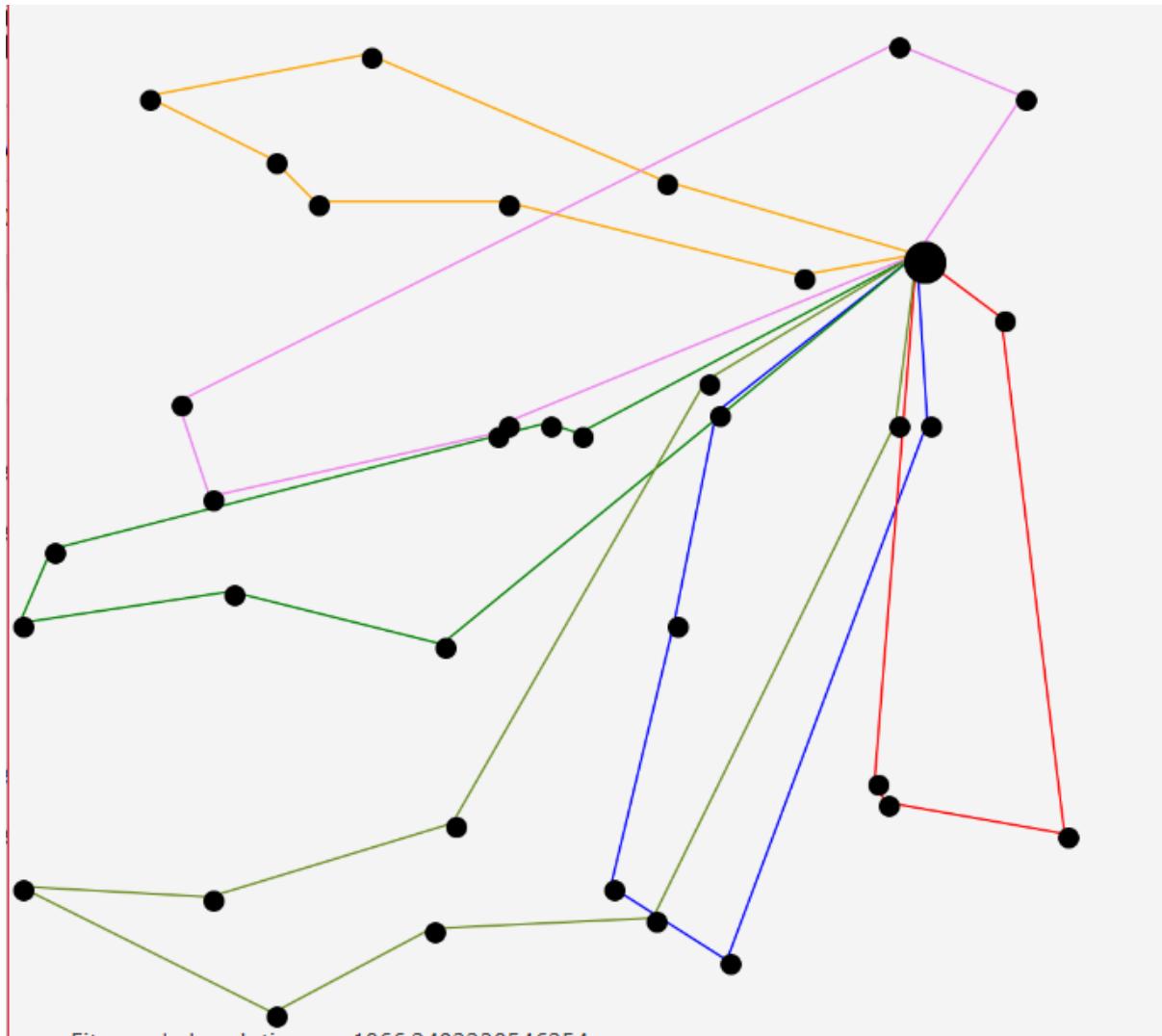
A3706

Tabou Taille 10



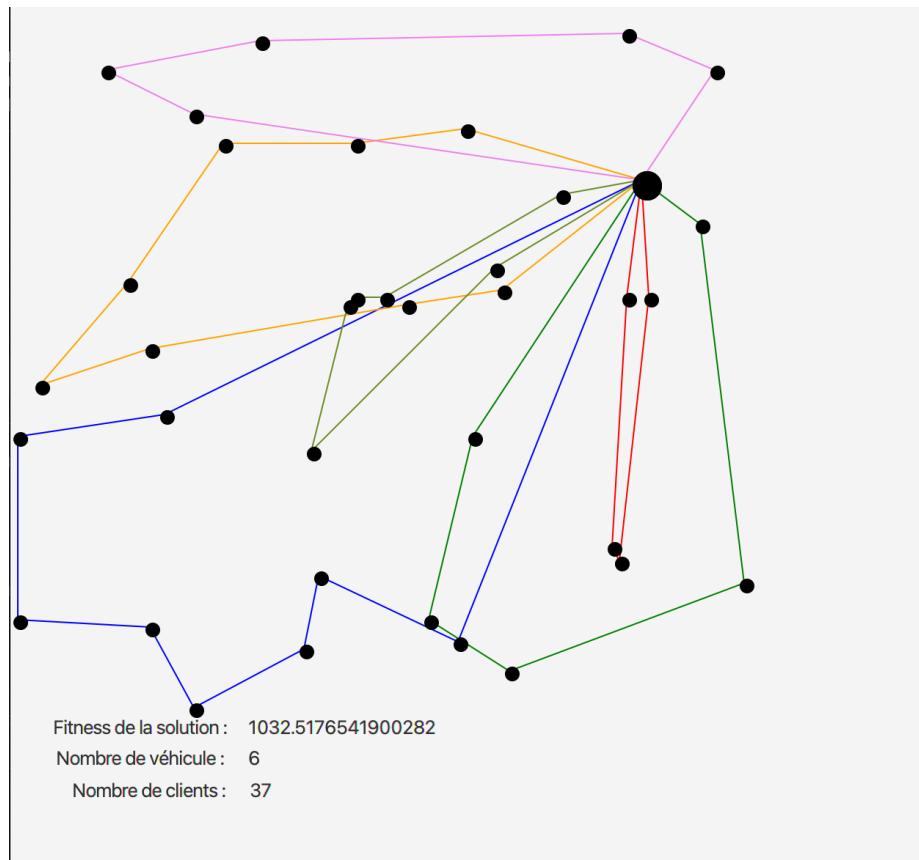
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2313	1028	910125	7

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1135	1066	839916	6

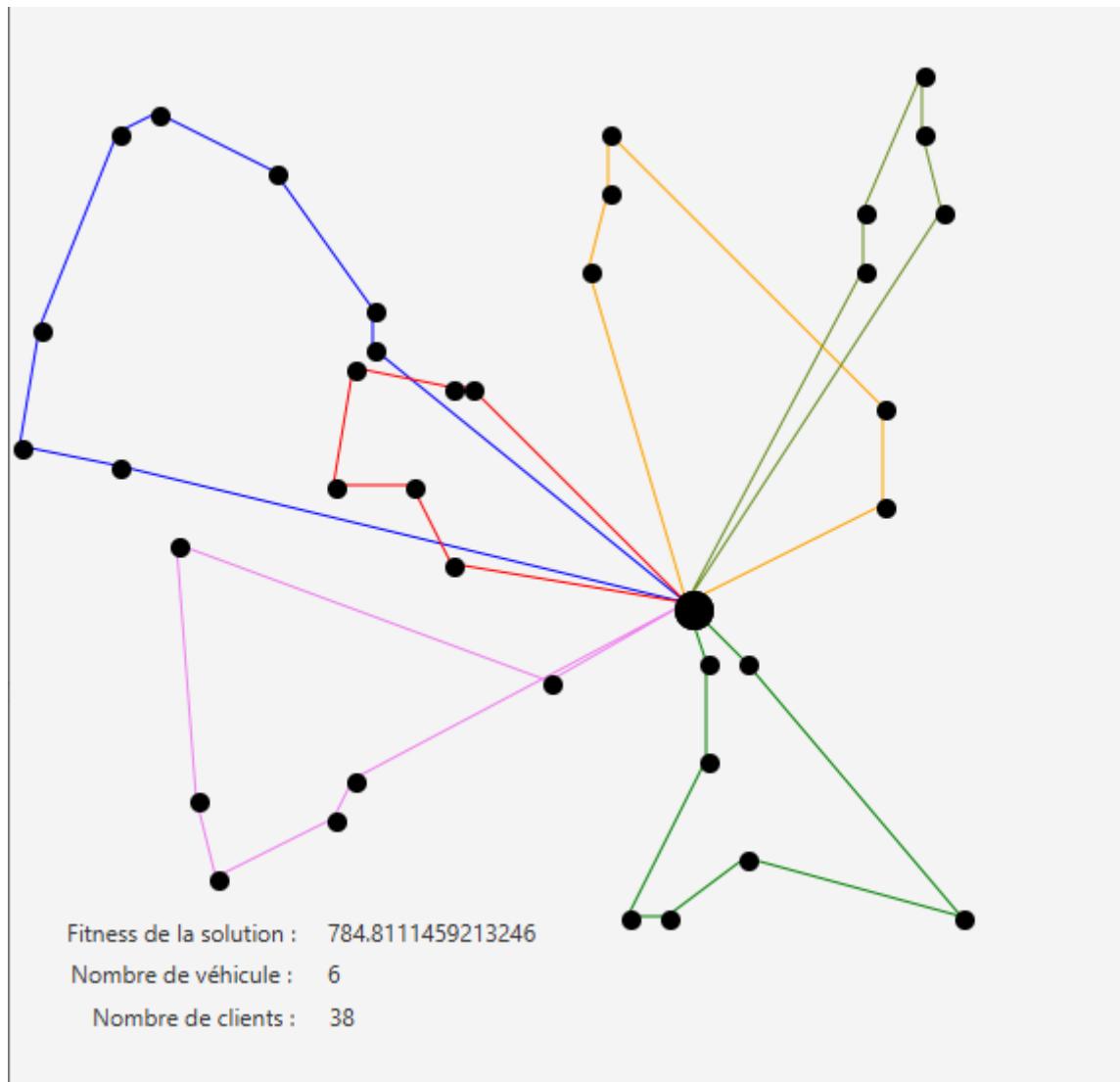
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
625ms	1032.51	839916	6

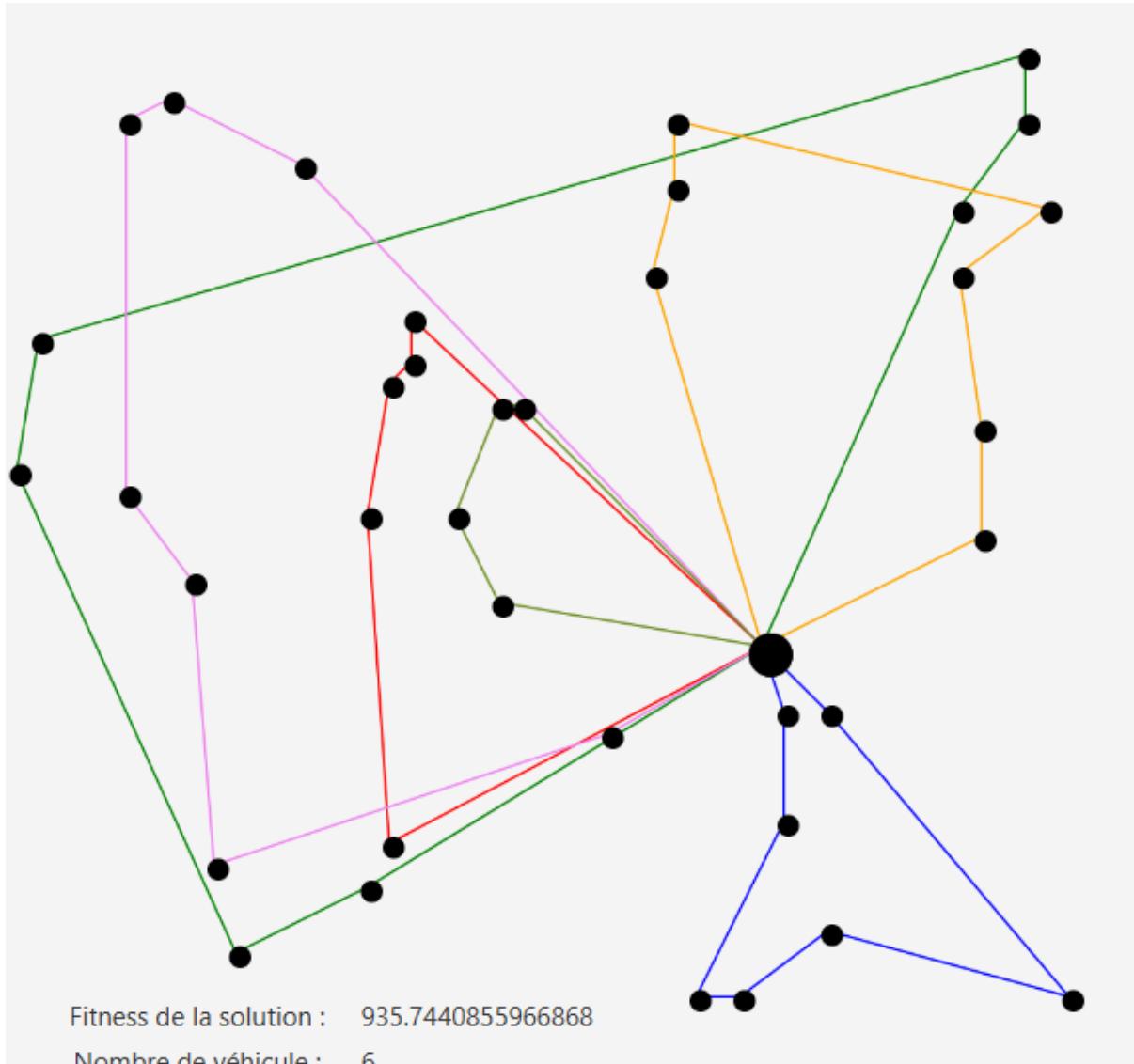
A3805

Tabou Taille 10



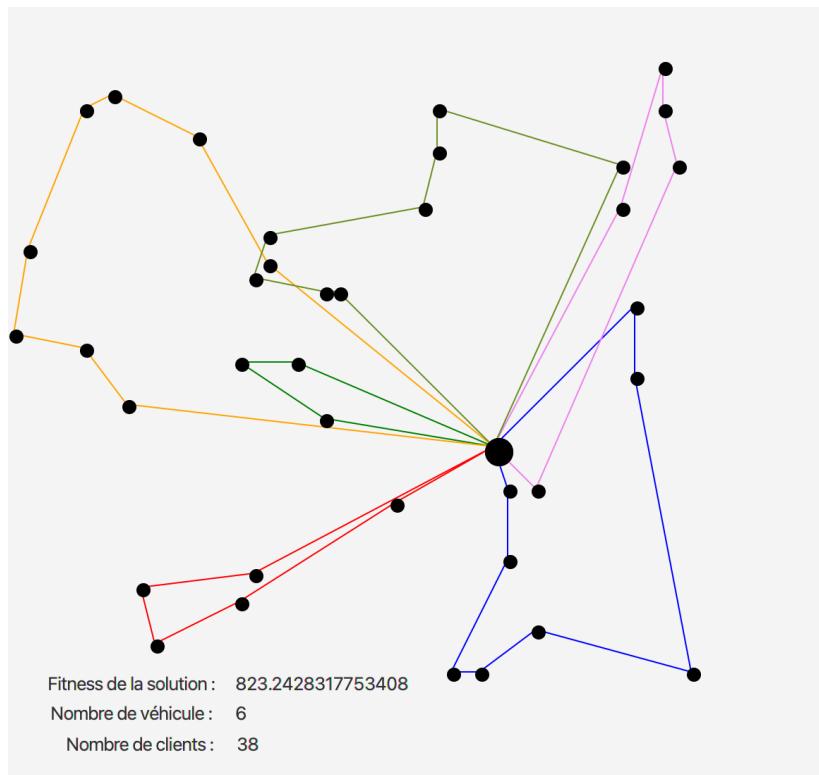
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2509	784	944329	6

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1204	935	839916	6

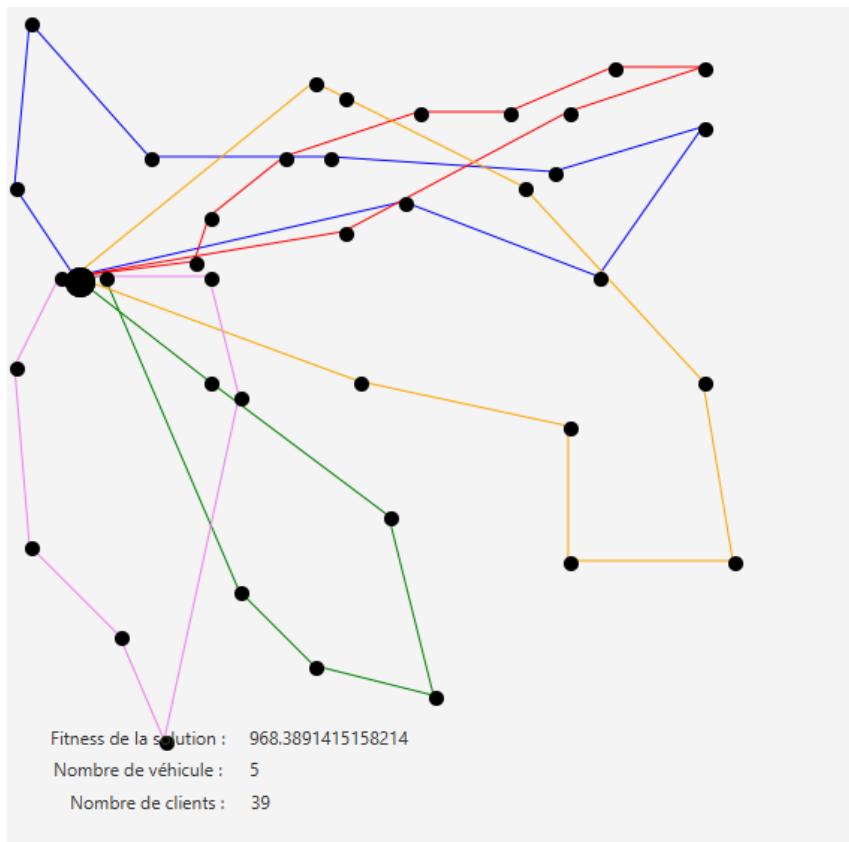
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
666ms	823.24	839916	6

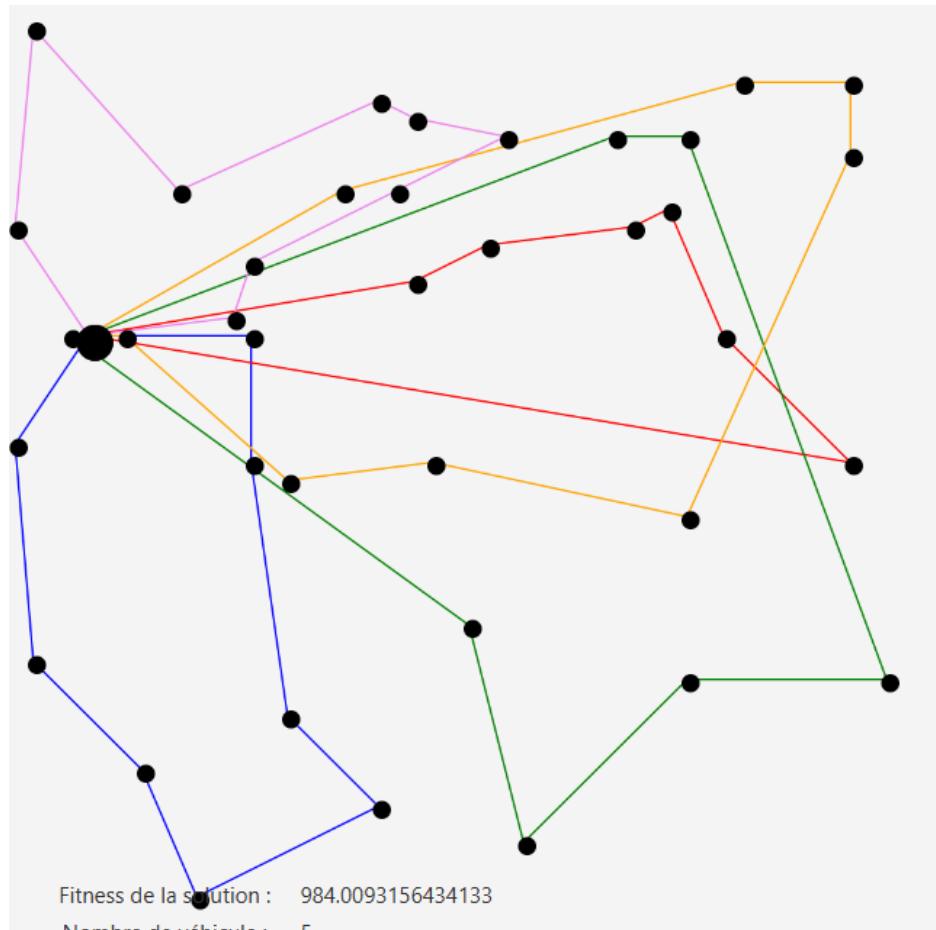
A3905

Tabou Taille 10



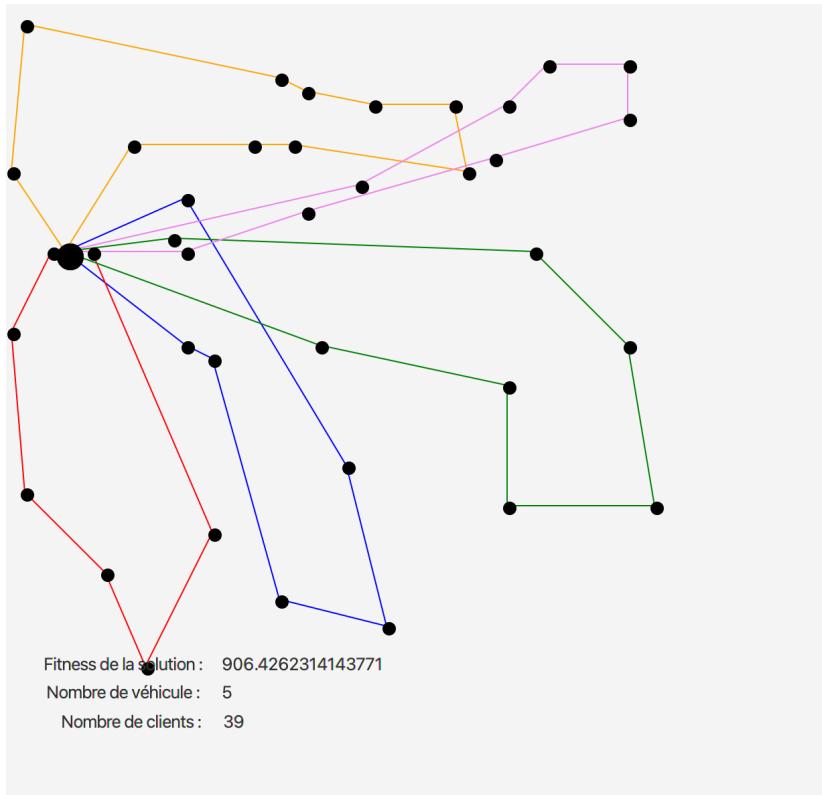
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
487	968	197502	5

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1155	984	839916	5

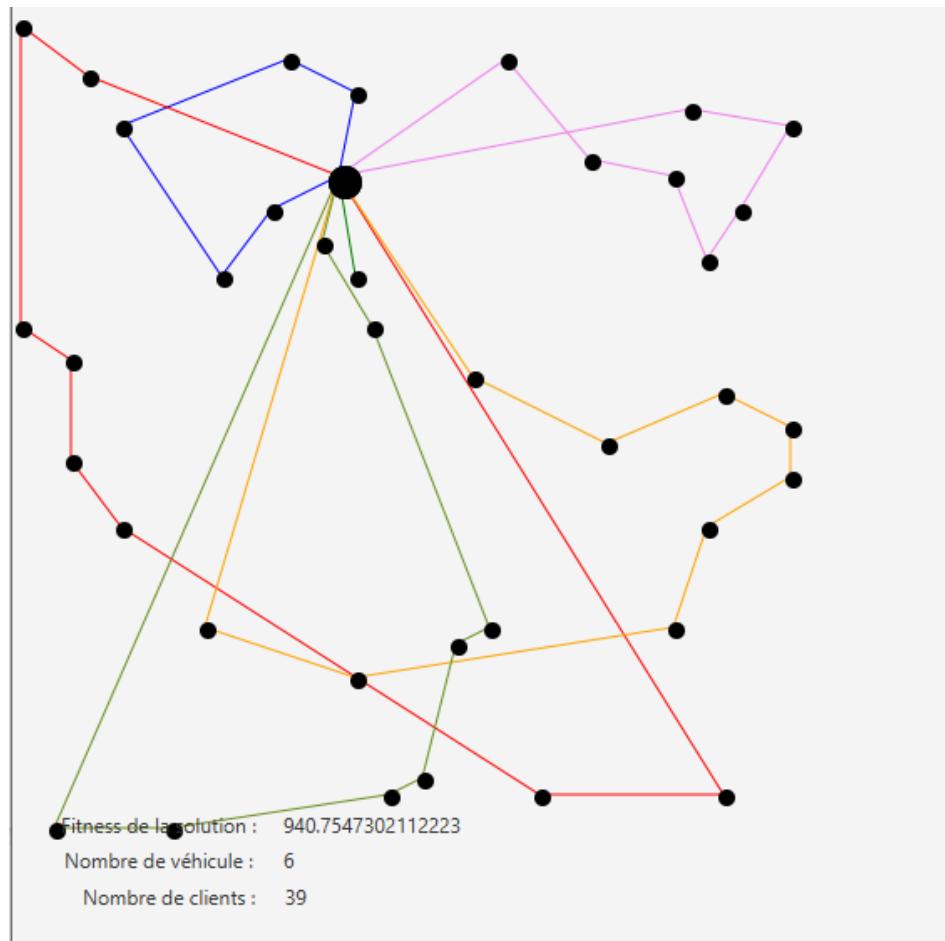
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
623ms	906	839916	5

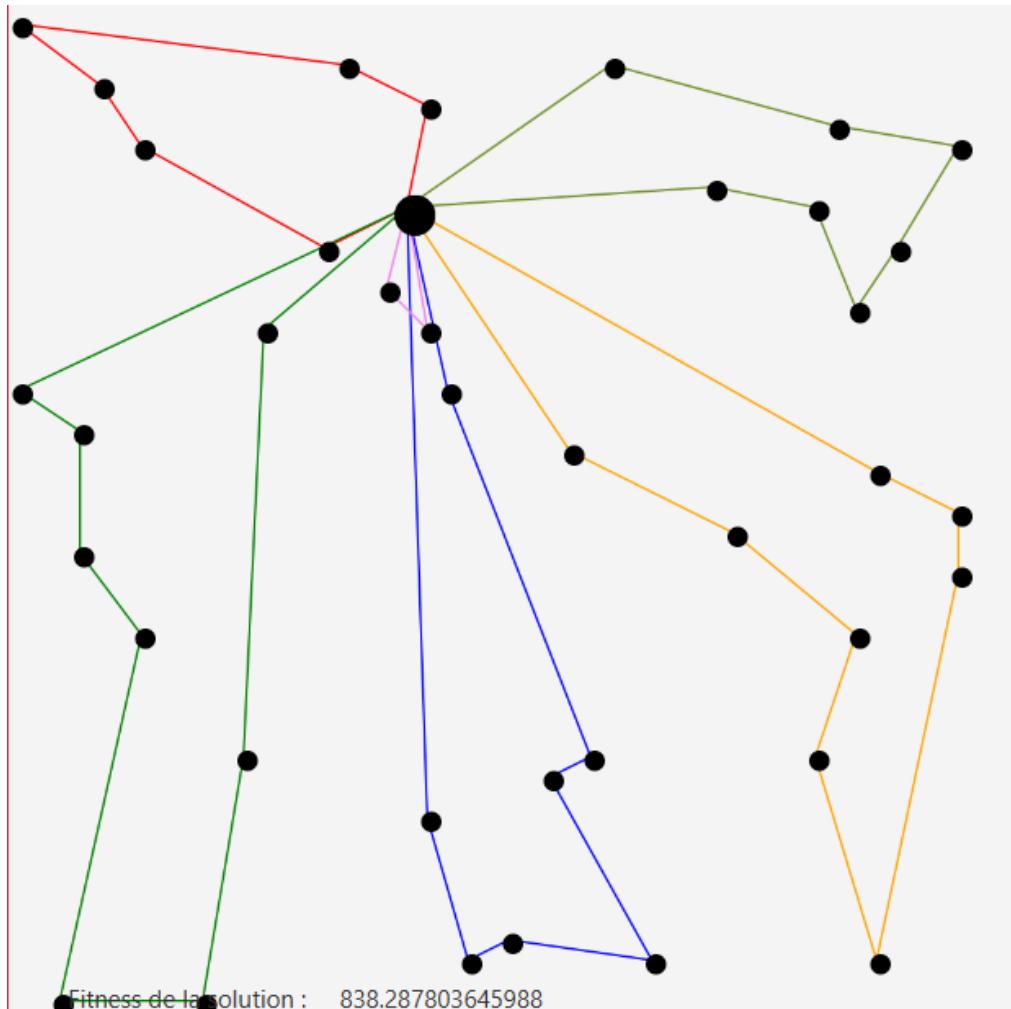
A3906

Tabou Taille 10



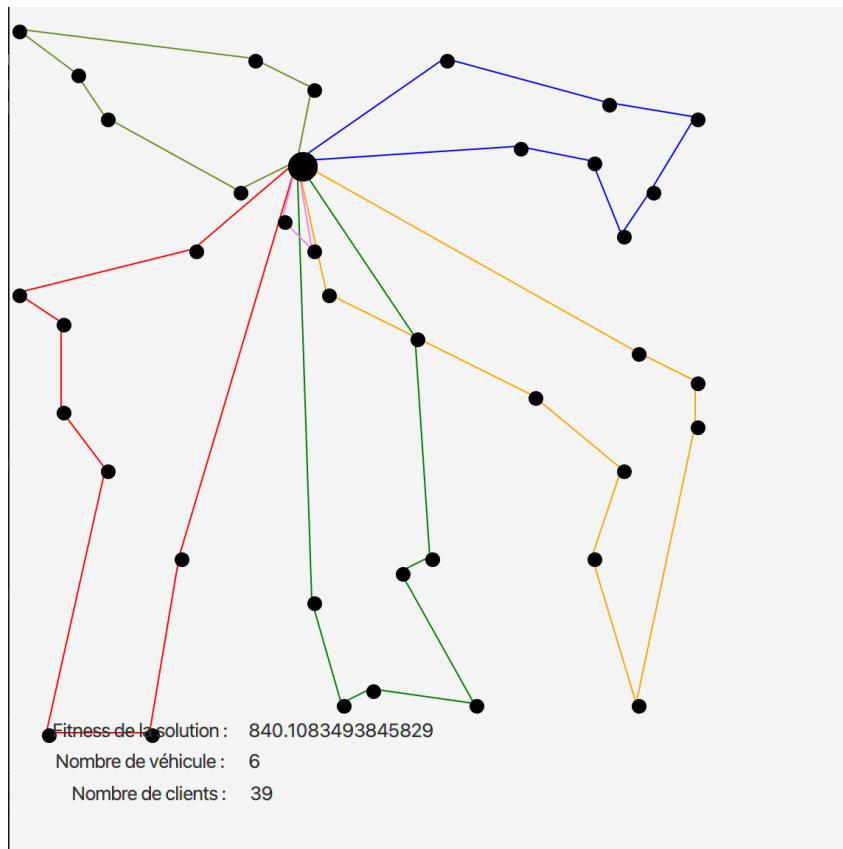
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1037	940	413227	6

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1578	828	839916	6

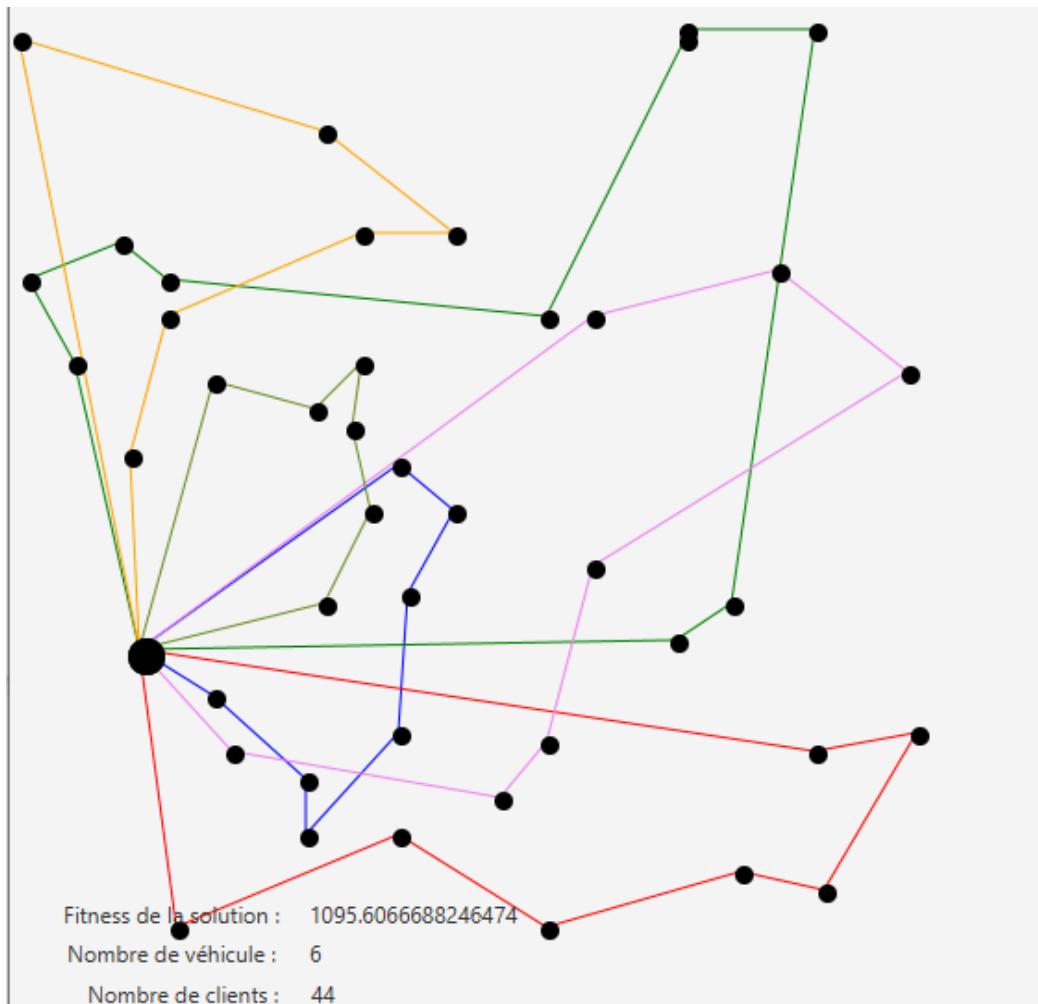
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
714ms	840	839916	6

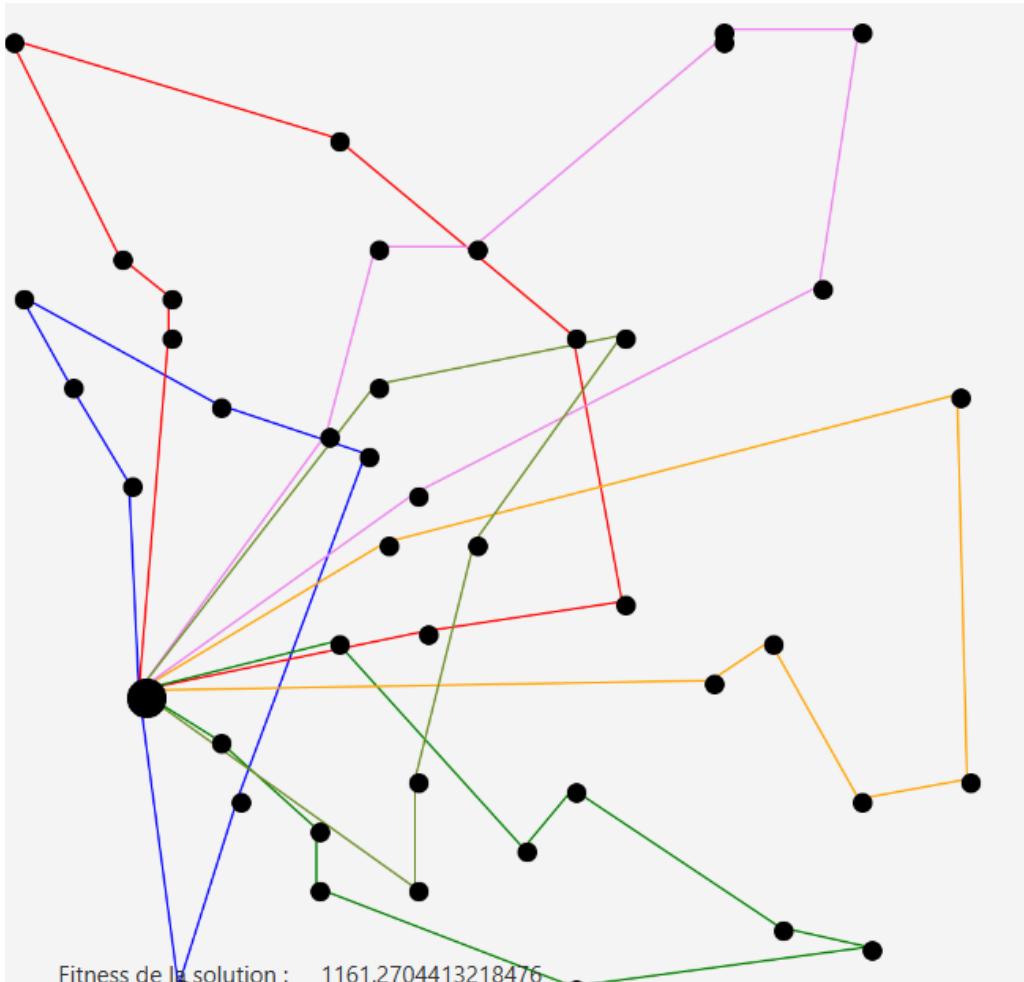
A4406

Tabou Taille 10



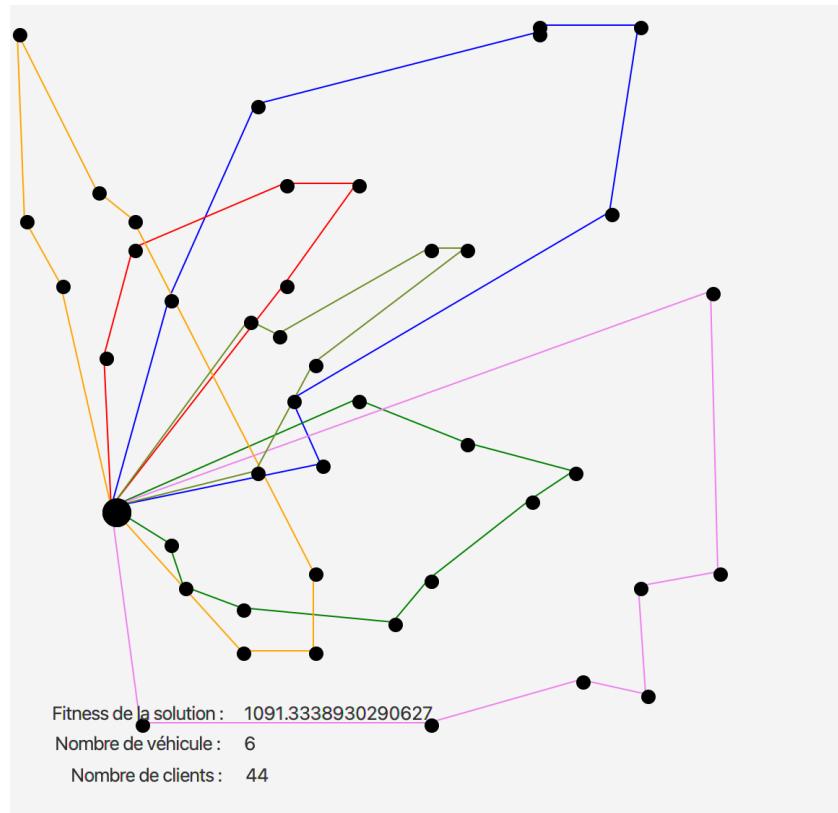
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
528	1095	195609	6

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1366	1161	839916	6

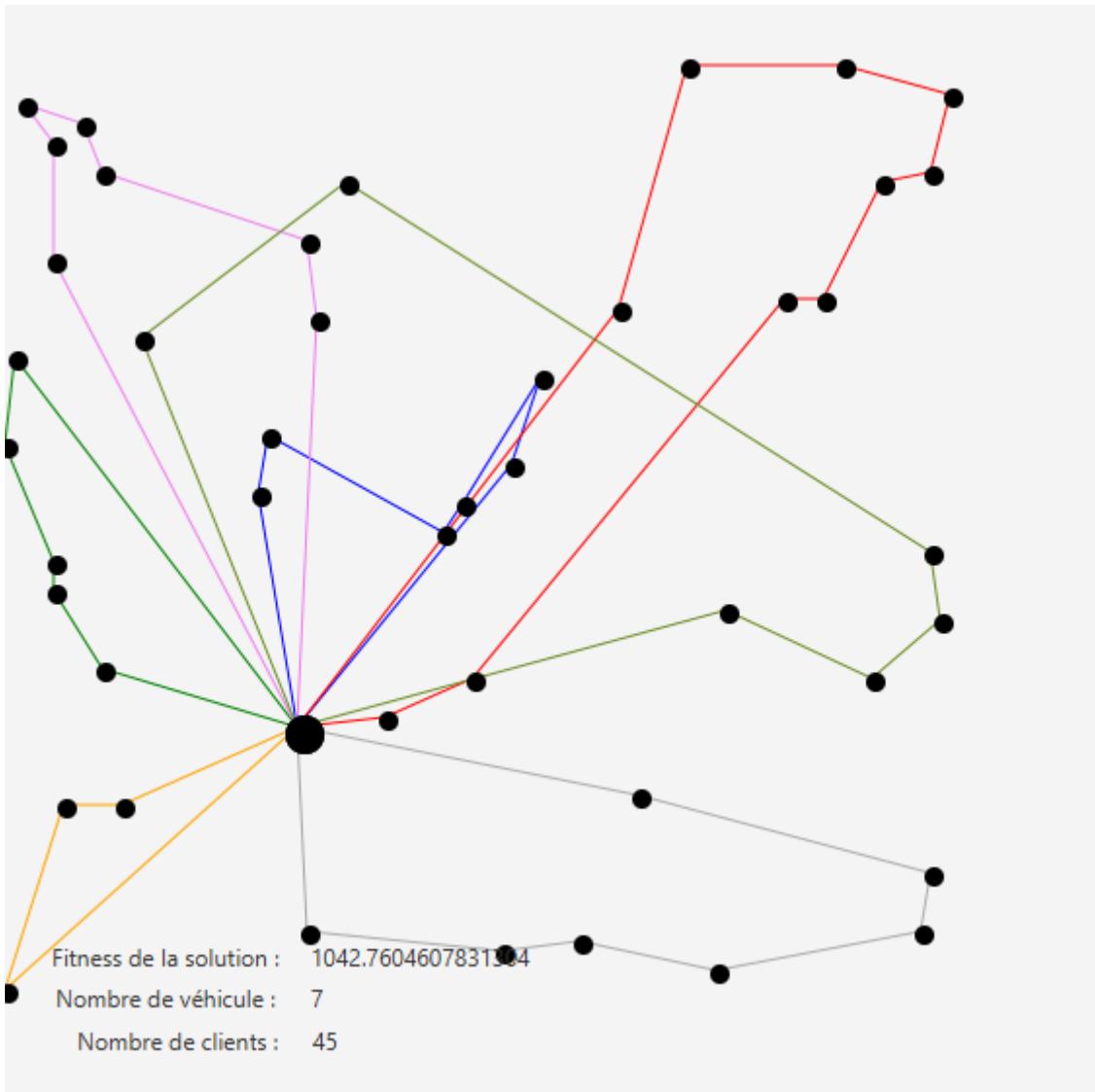
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
738ms	1091	839916	6

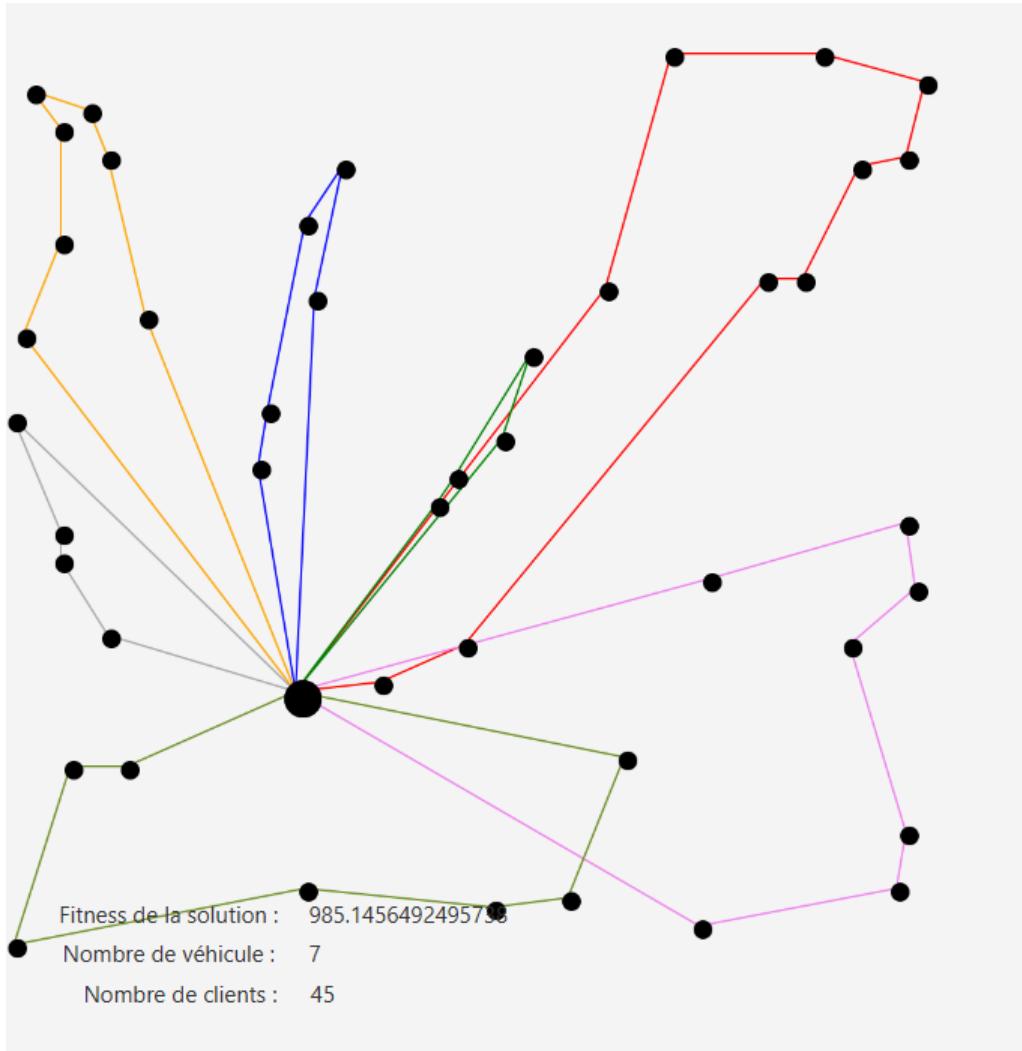
A4506

Tabou Taille 10



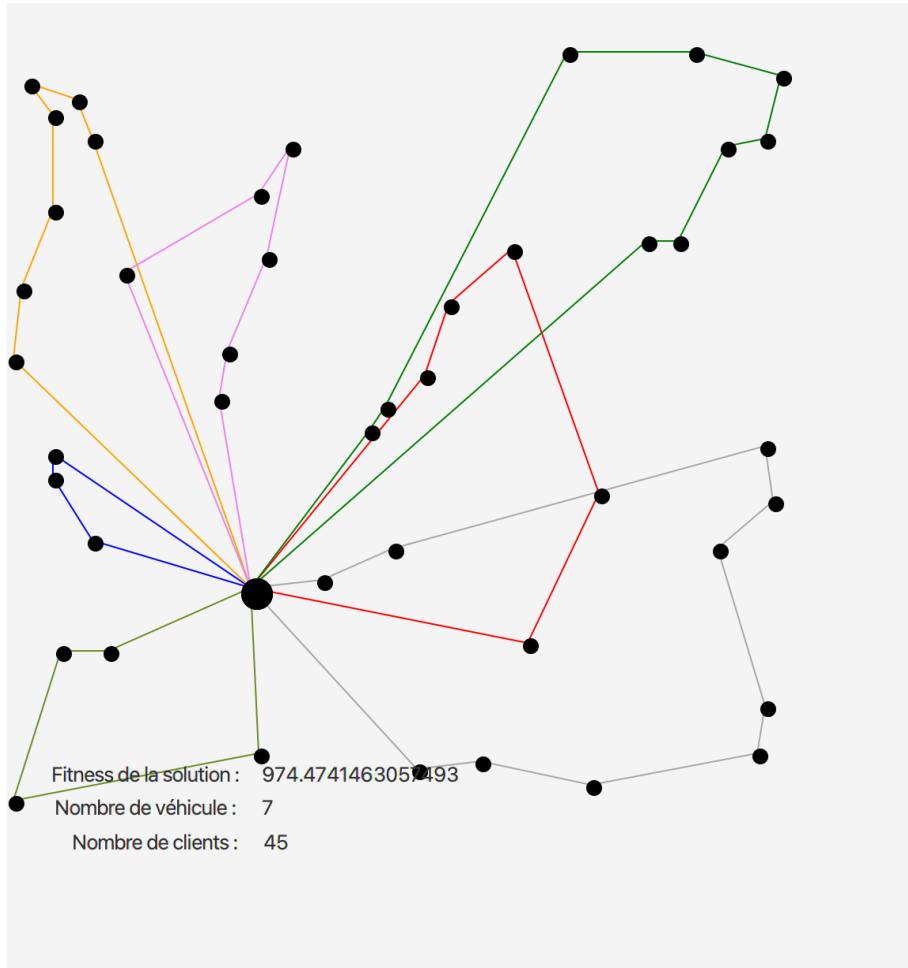
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2770	1042	927646	7

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1650	985	839916	7

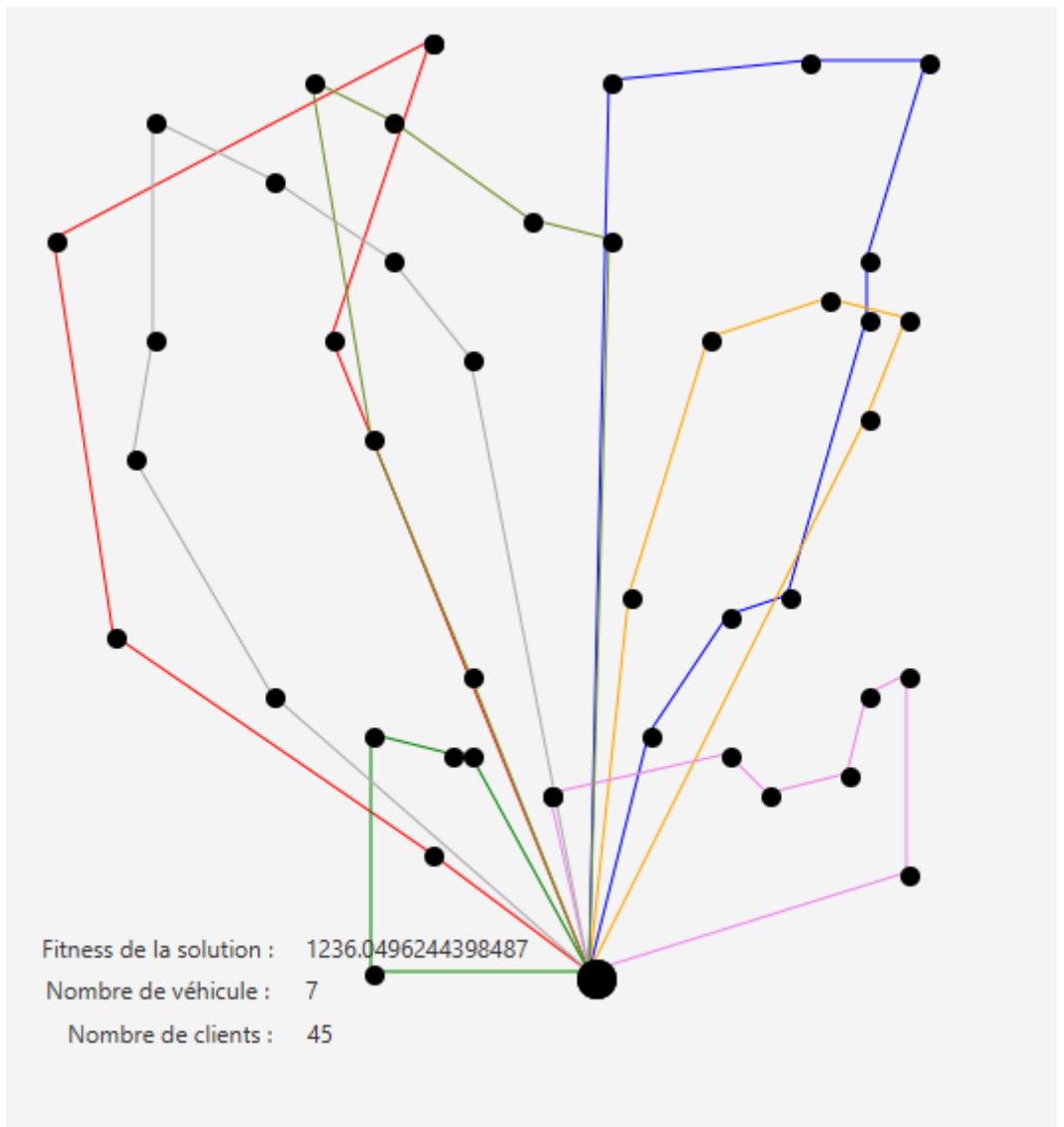
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
731ms	974	839916	7

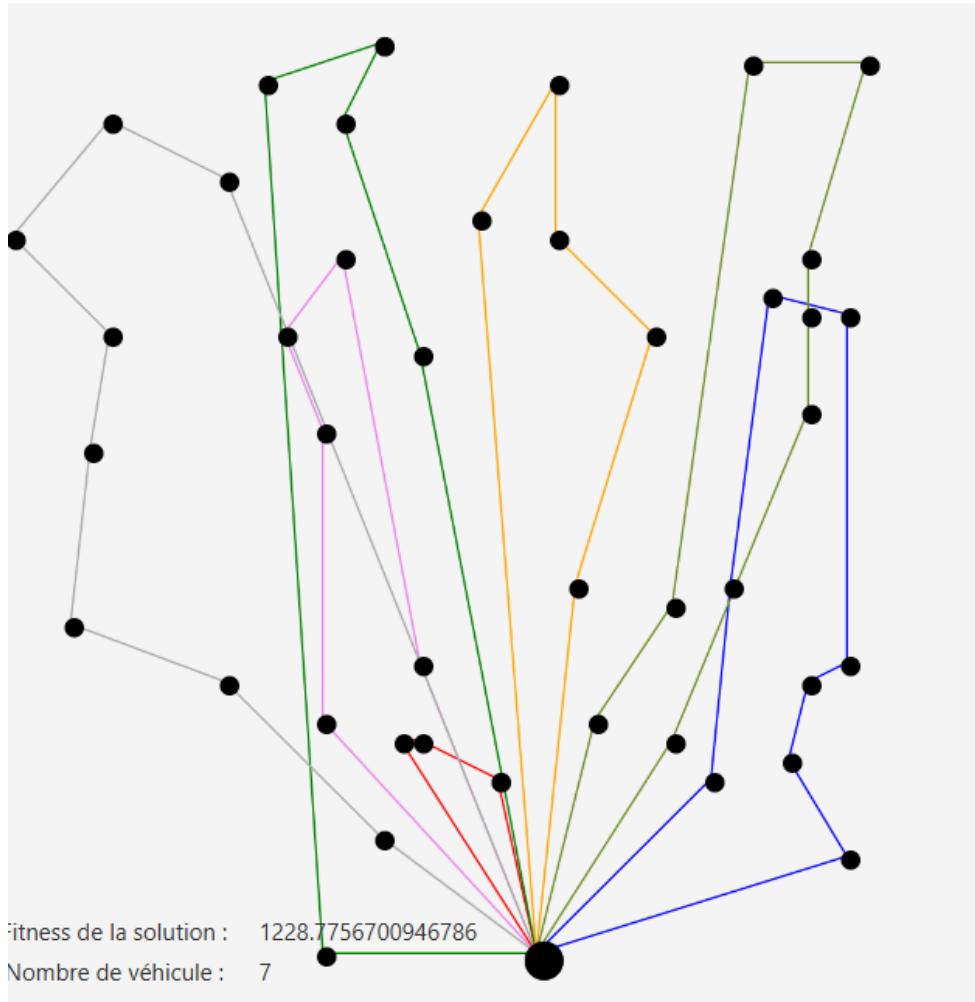
A4507

Tabou Taille 10



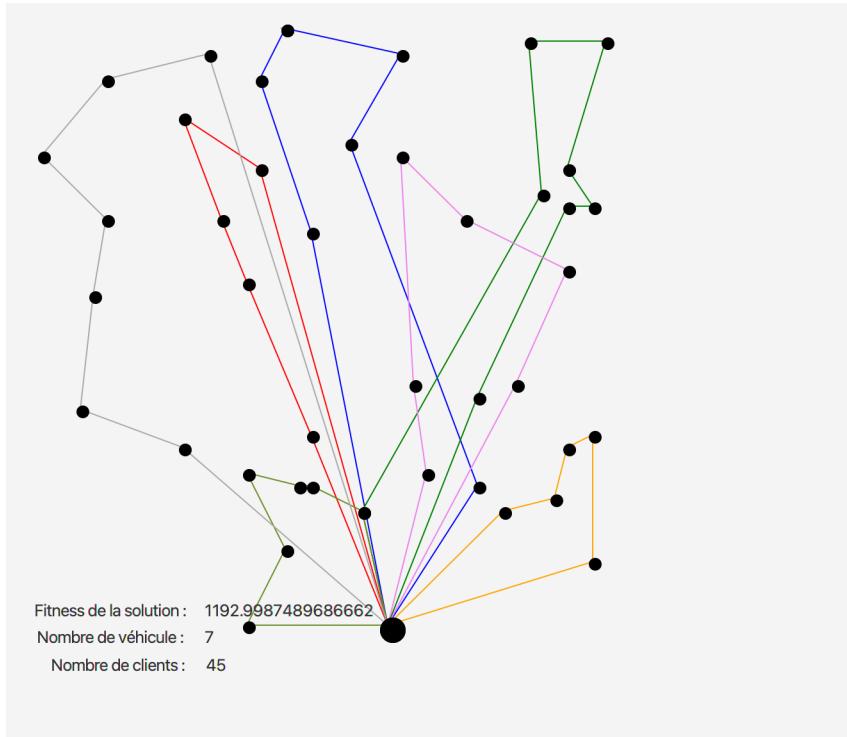
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
714	1236	265173	7

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1317	1228	839916	7

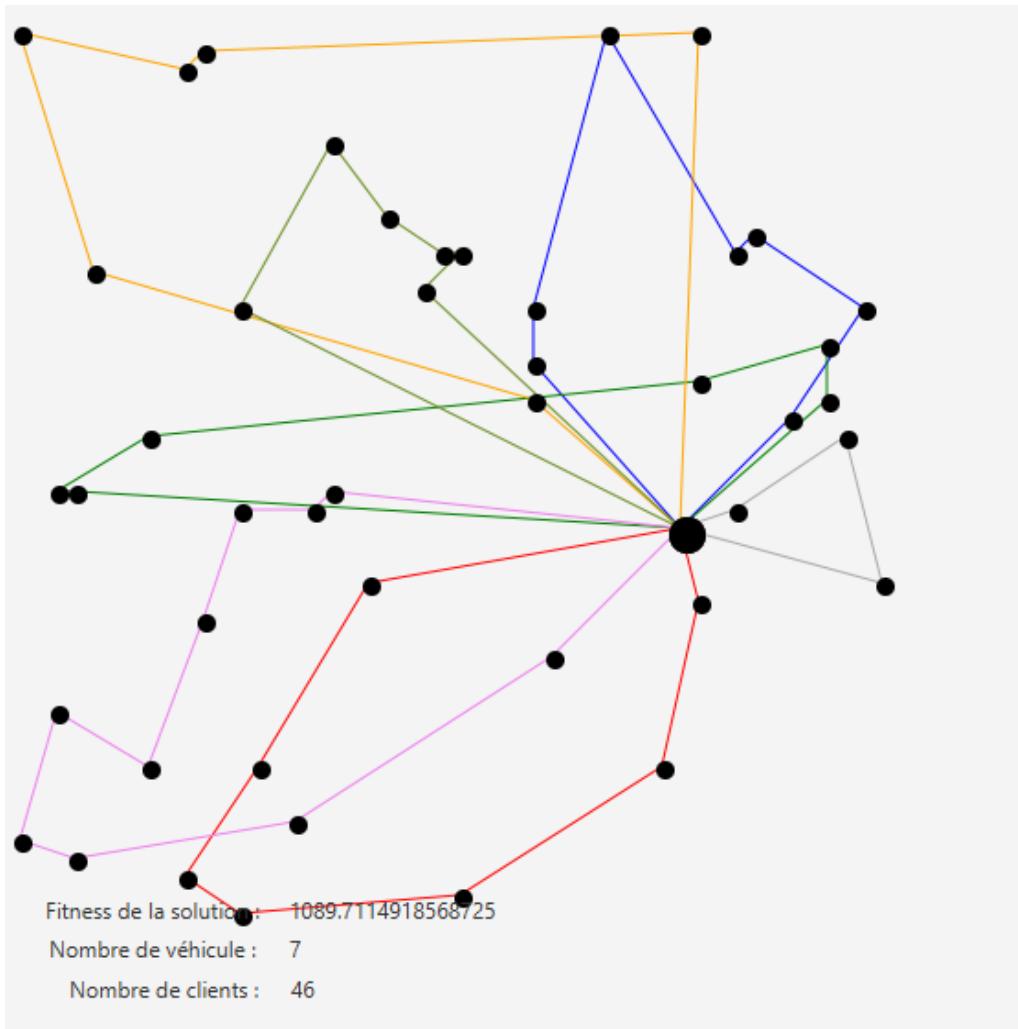
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
743ms	1192	839916	7

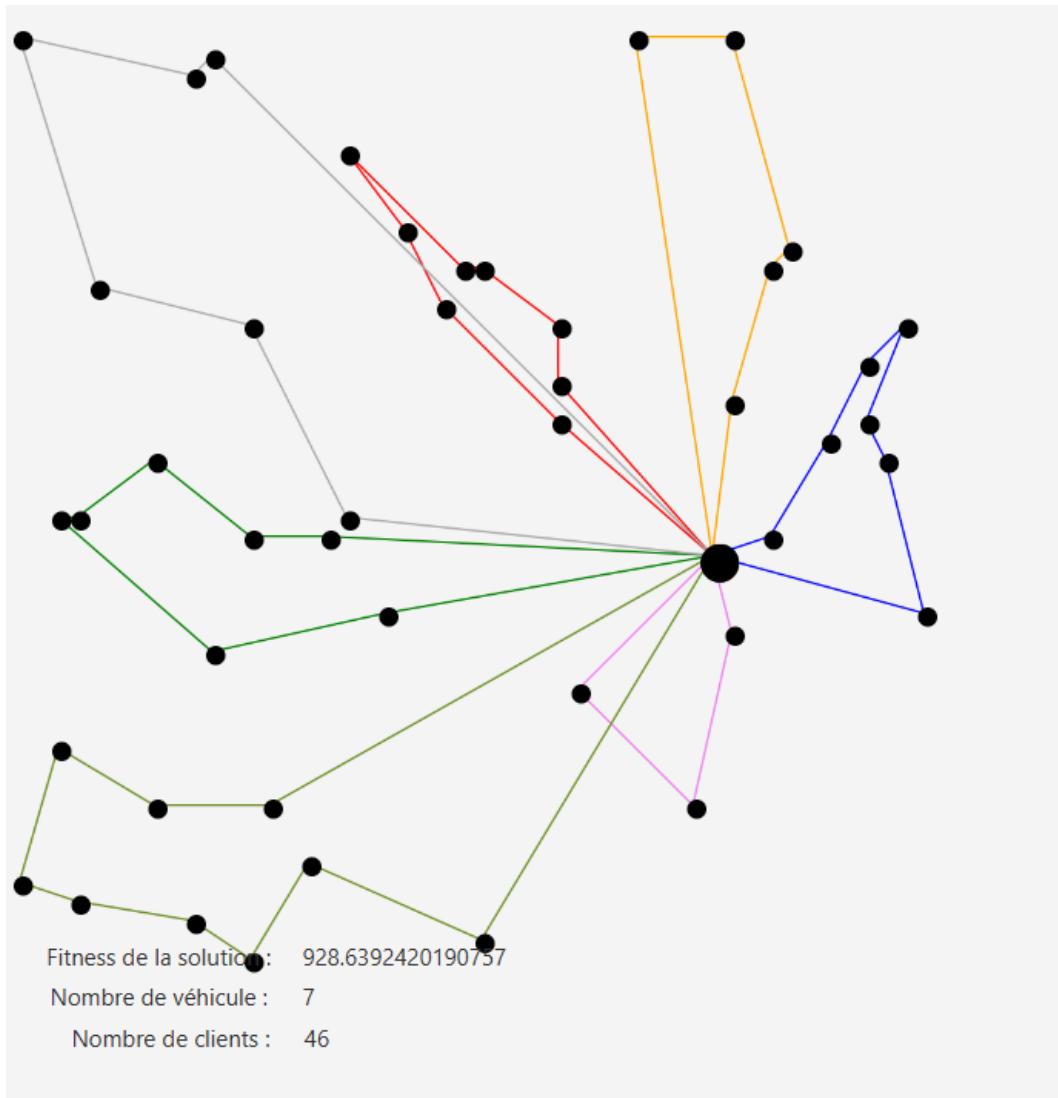
A4607

Tabou Taille 10



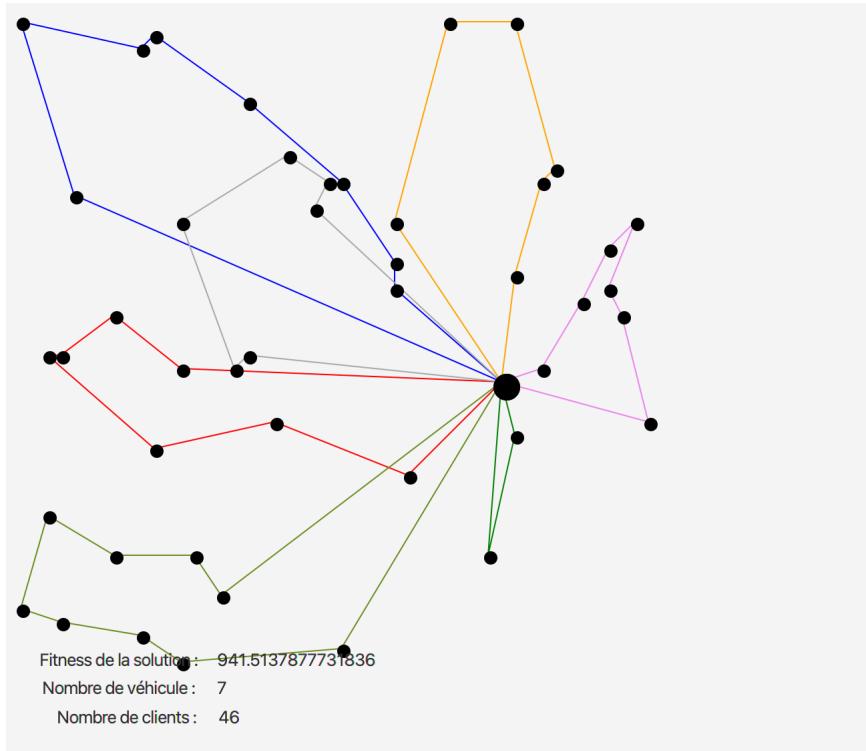
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
4044	1089	1273795	7

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1650	928	839916	7

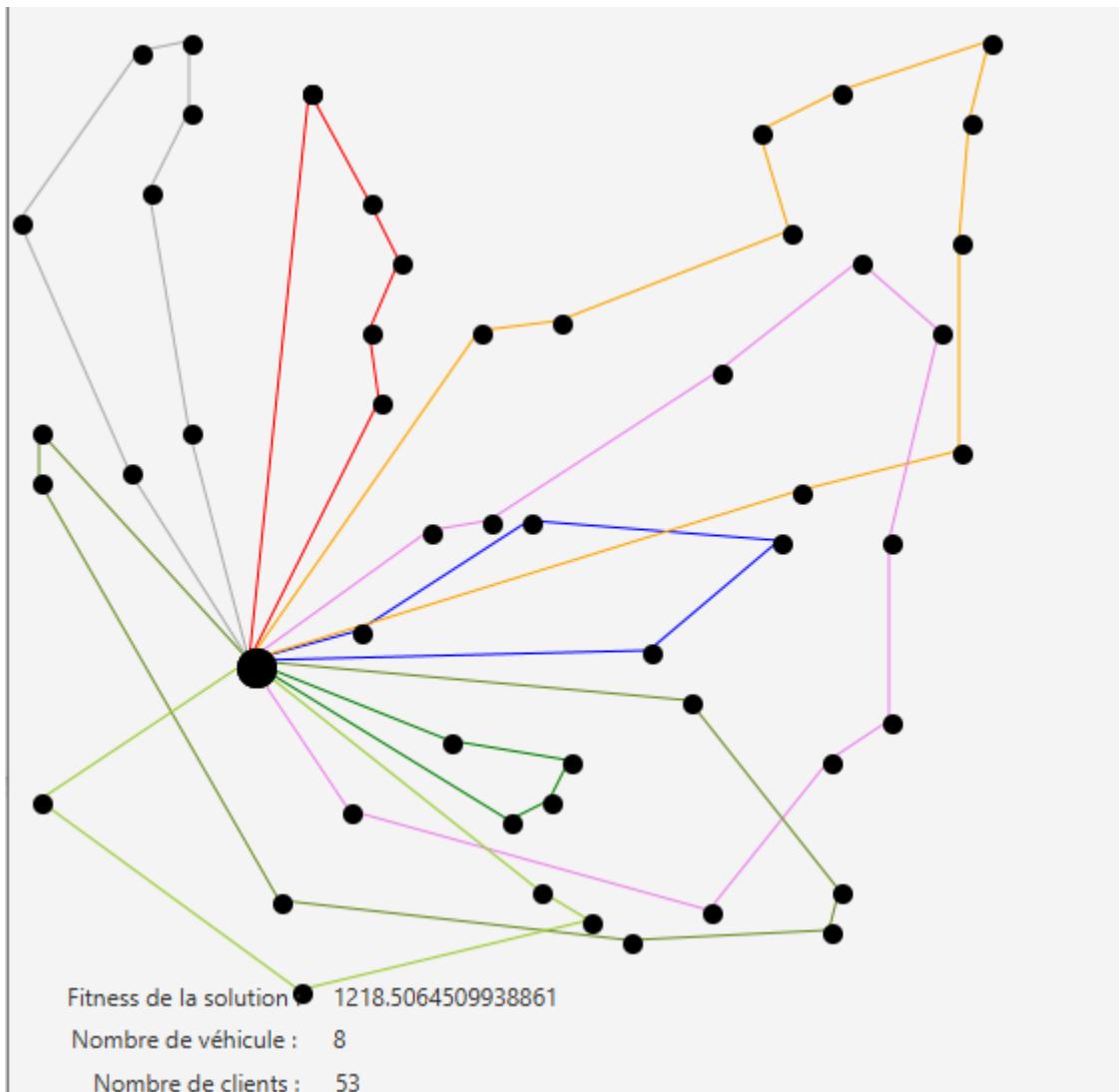
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
737ms	941	839916	7

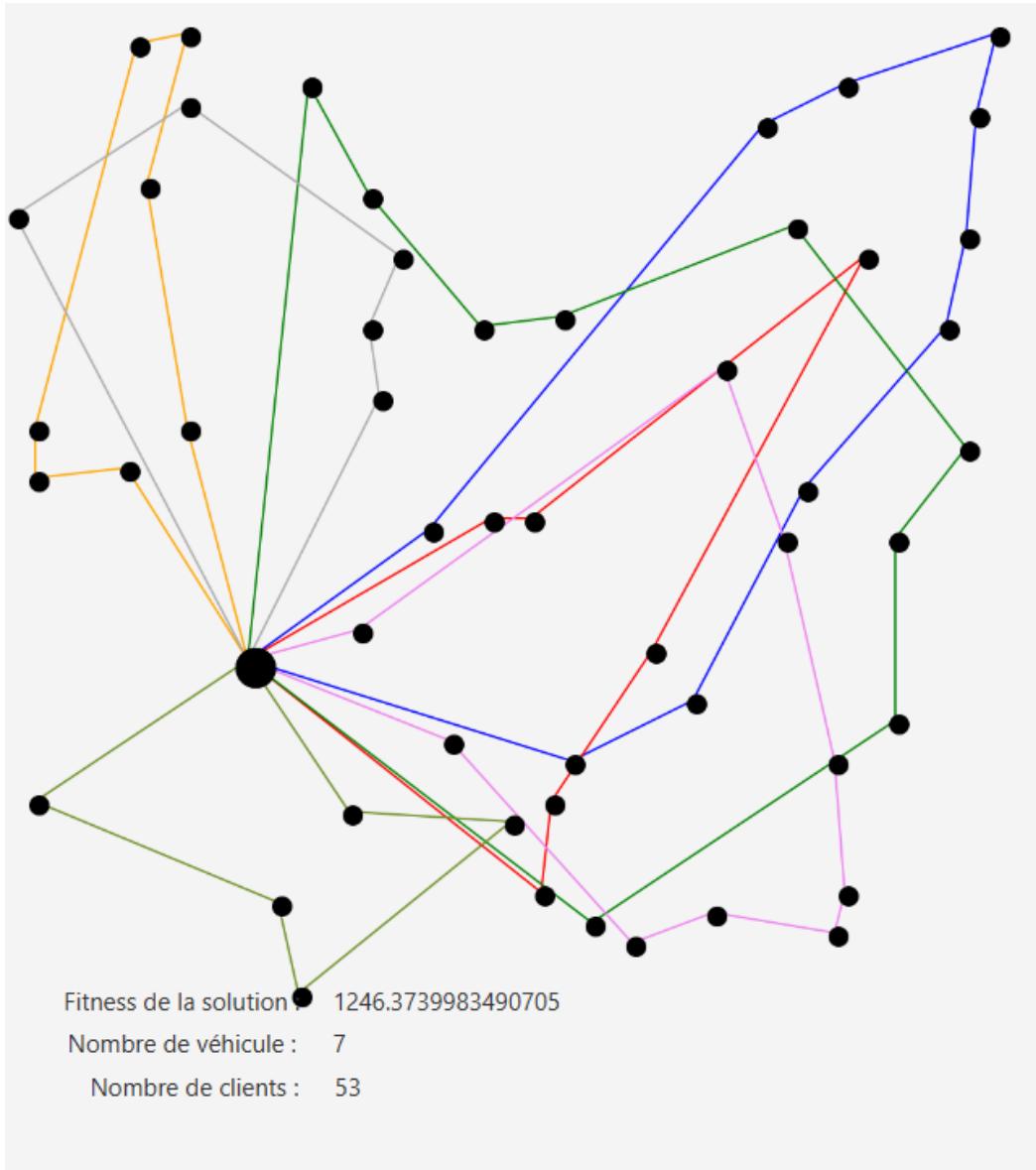
A5307

Tabou Taille 10



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2031	1218	2012253	8

Recuit Température = 50



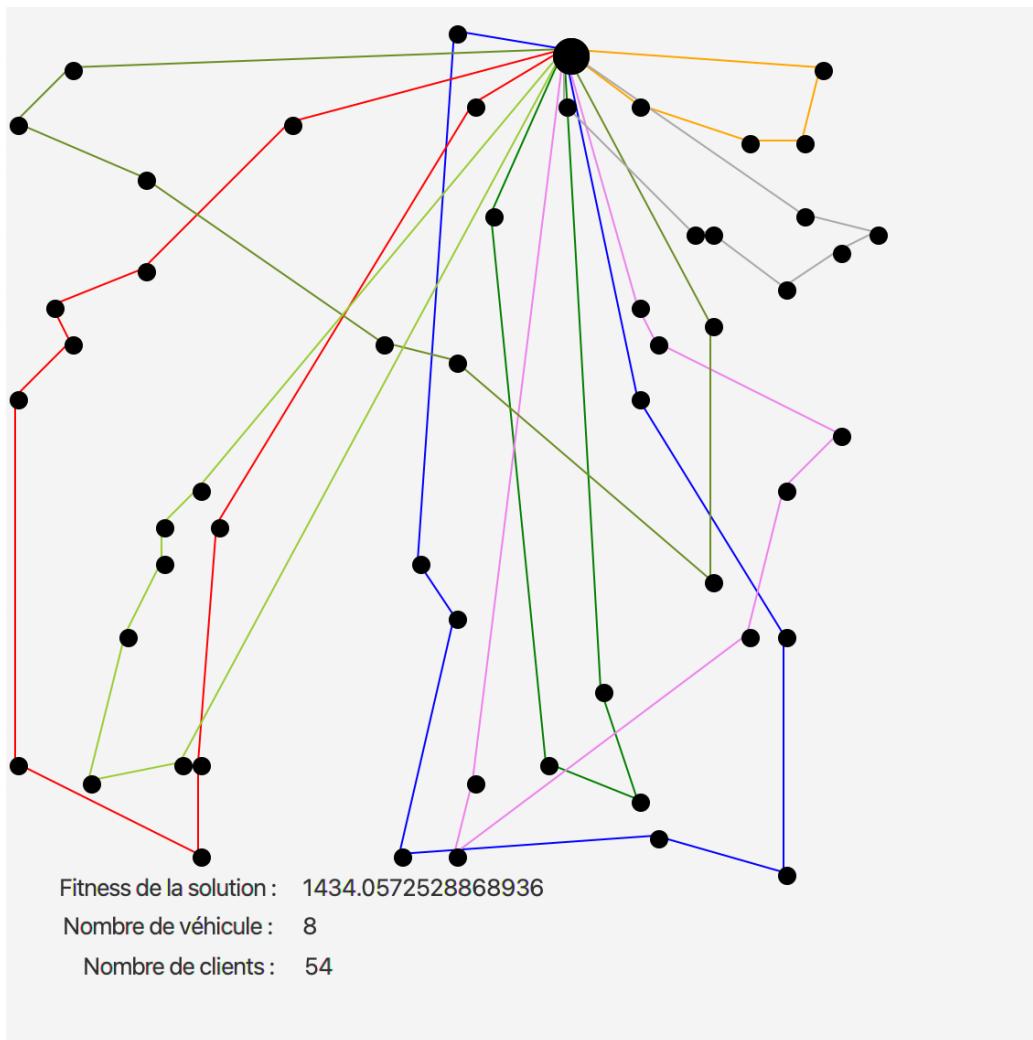
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1707	1246	839916	7

Recuit Température = 200

Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1301	1162	839916	7

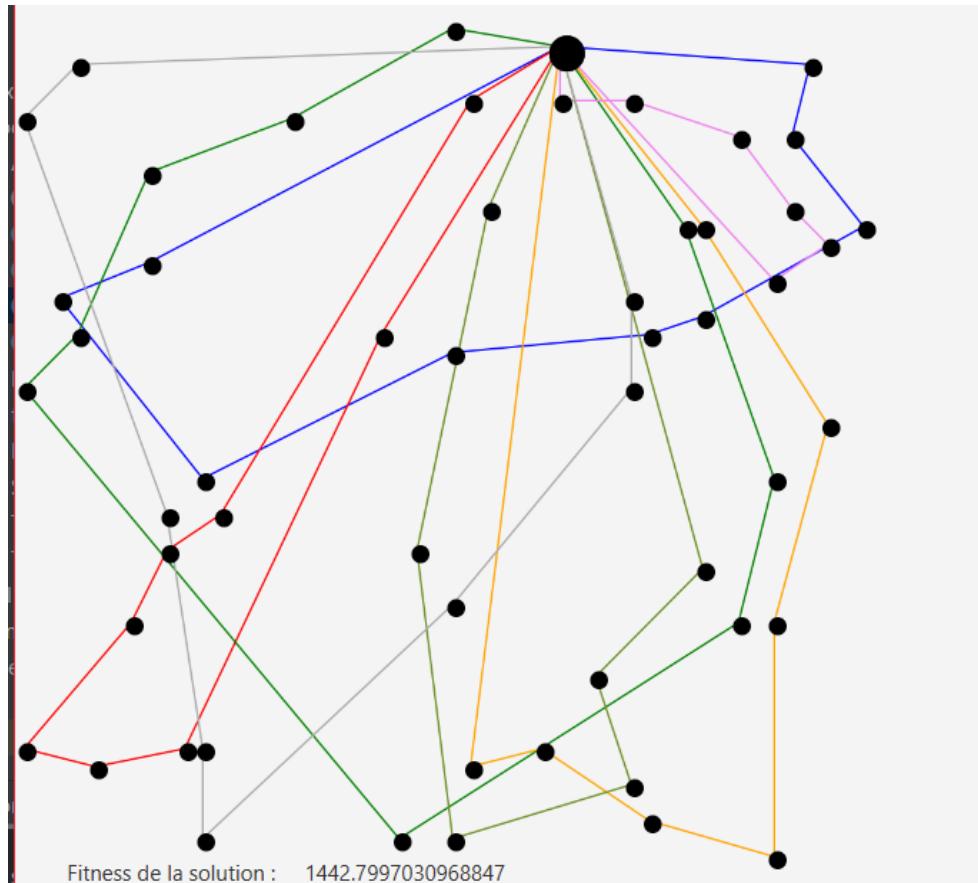
A5407

Tabou Taille 10



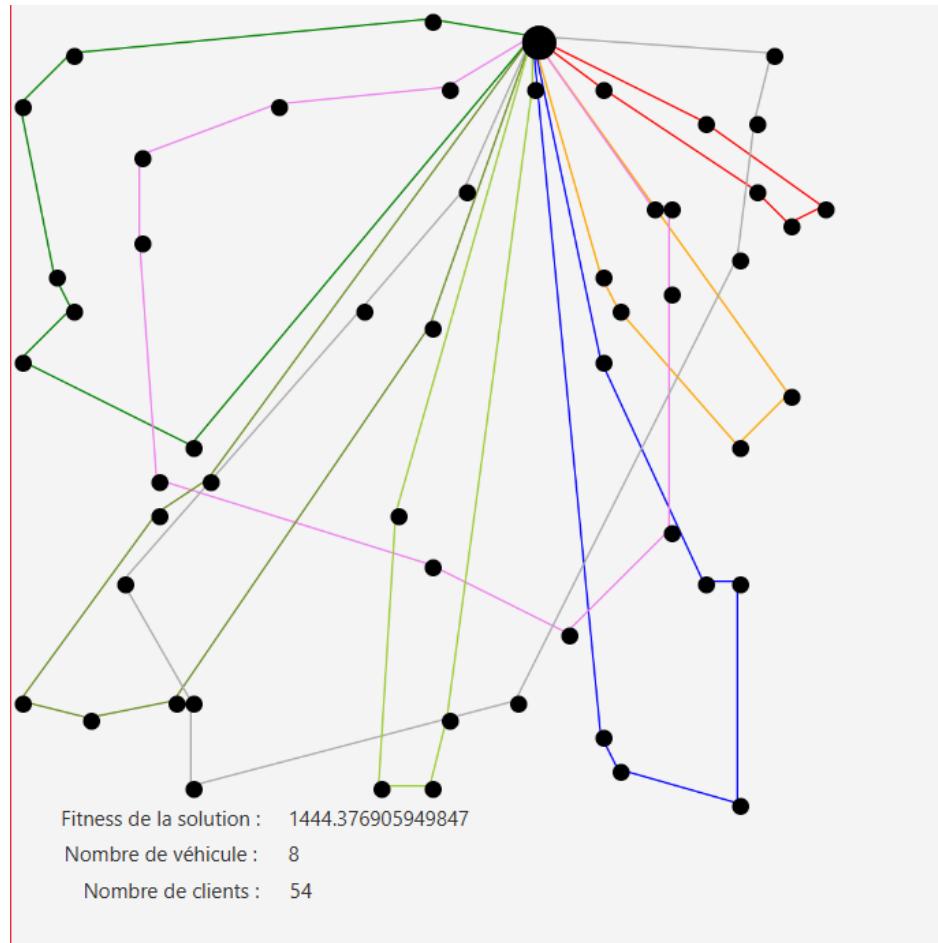
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
3734ms	1434	1656056	8

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1599	1442	839916	7

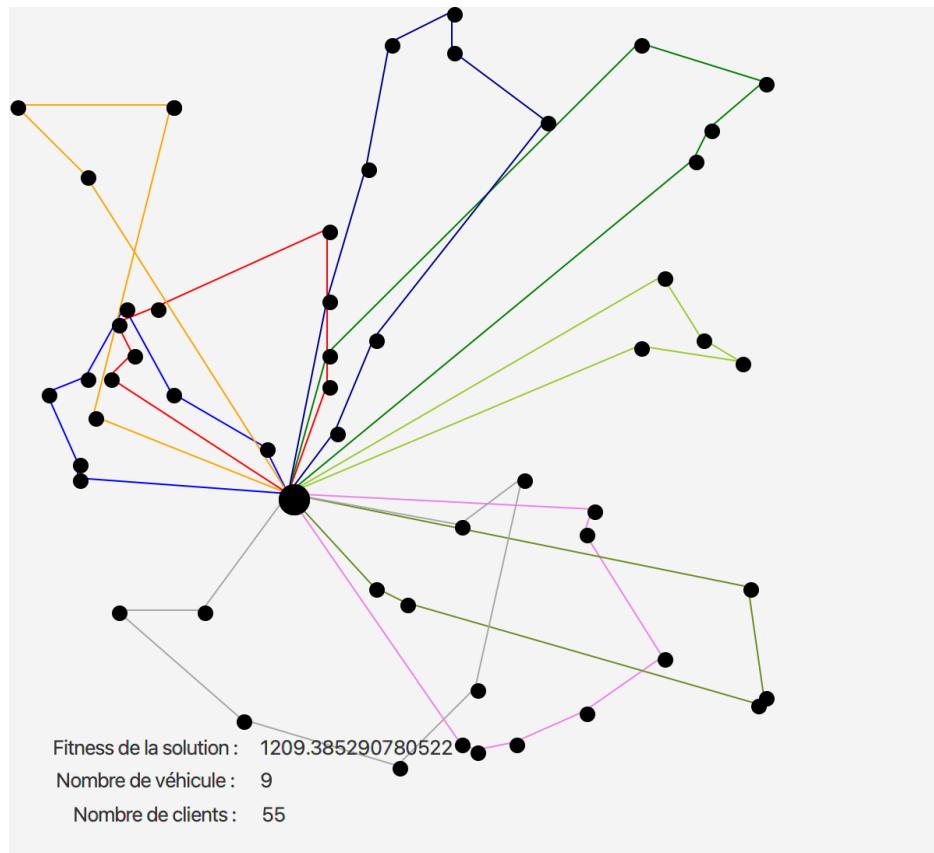
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1504	1444	839916	8

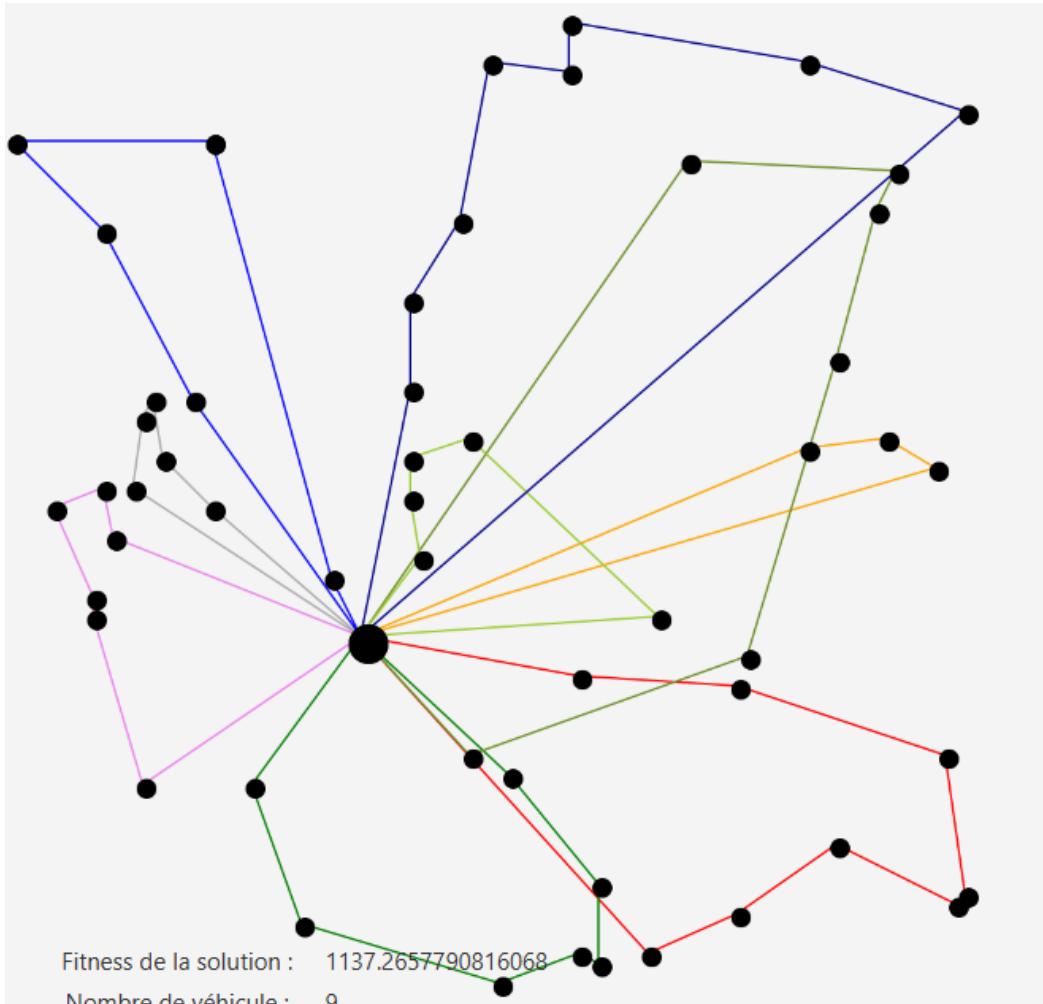
A5509

Tabou Taille 10



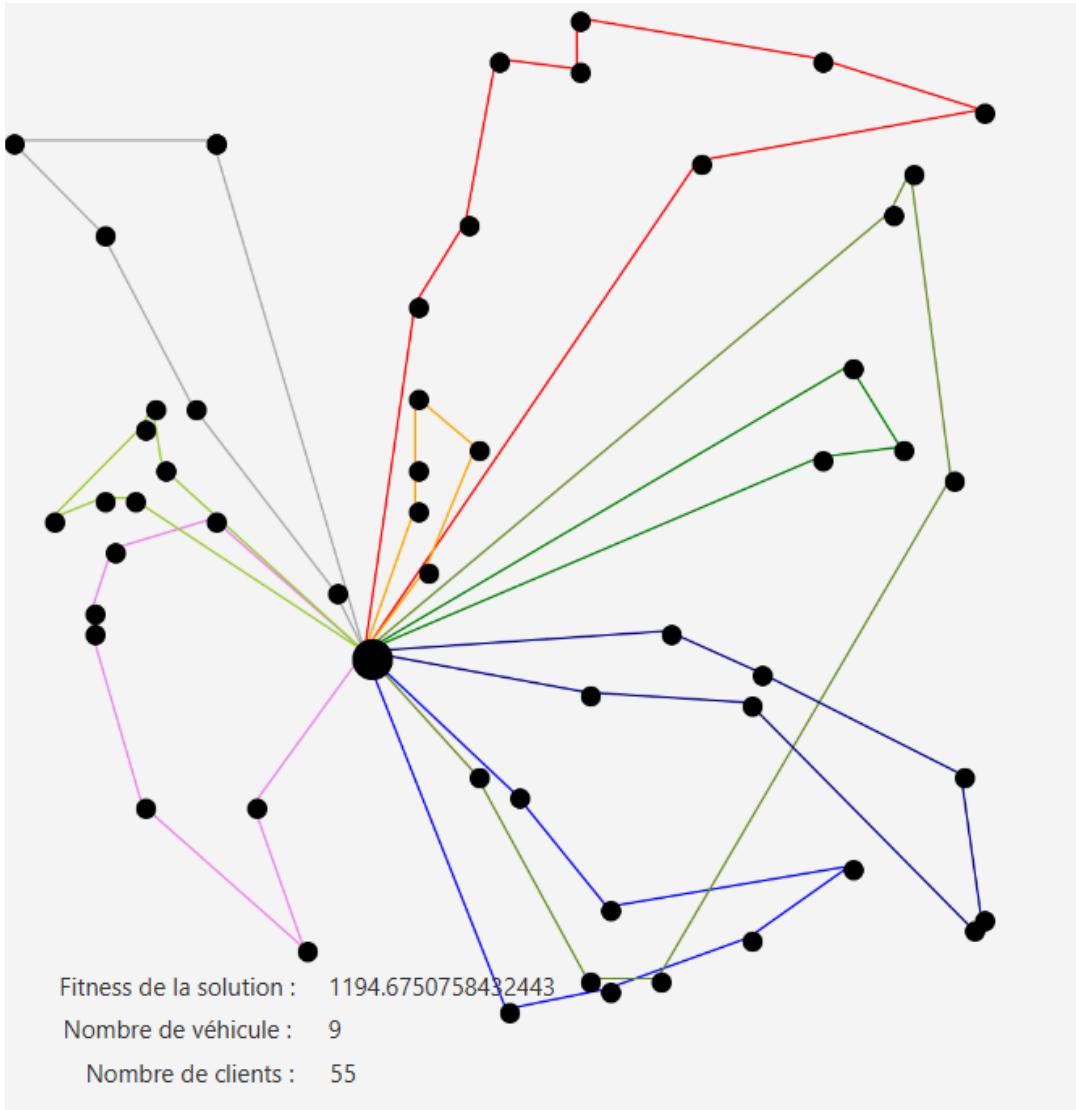
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
885ms	1209	452184	9

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1584	1137	839916	9

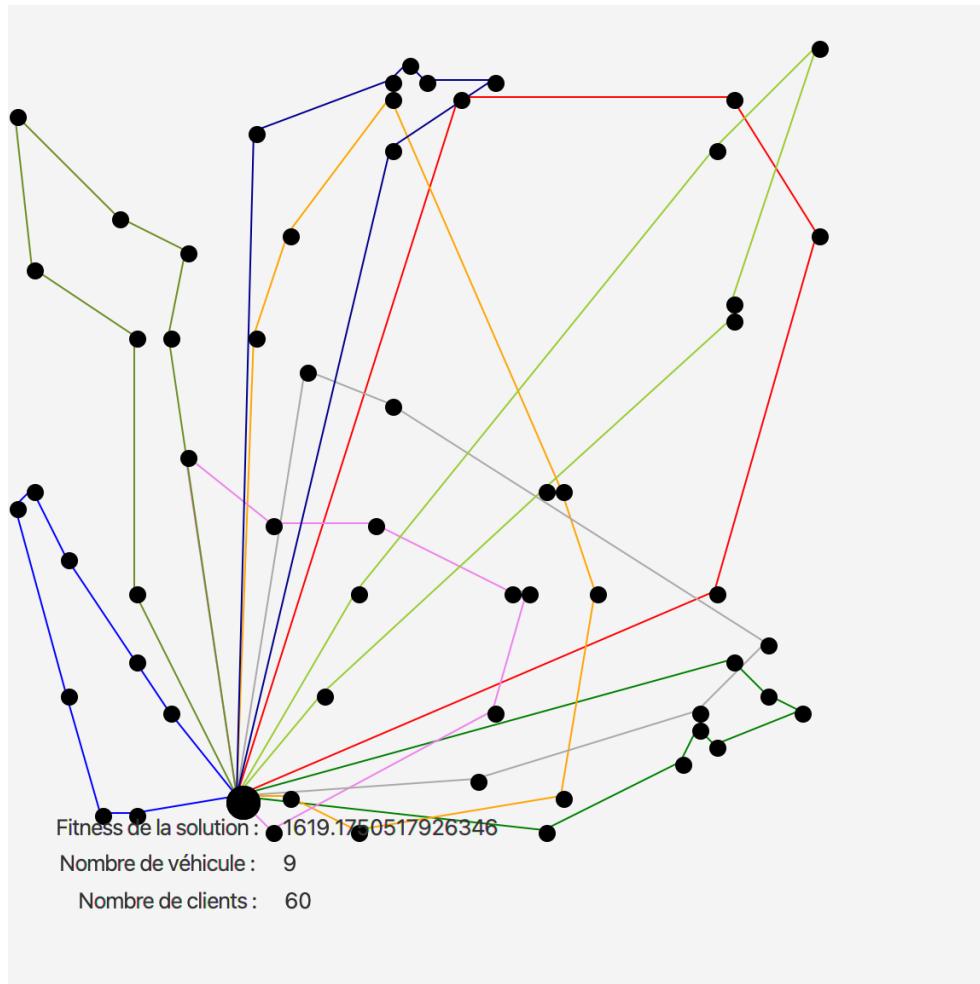
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1435	1194	839916	9

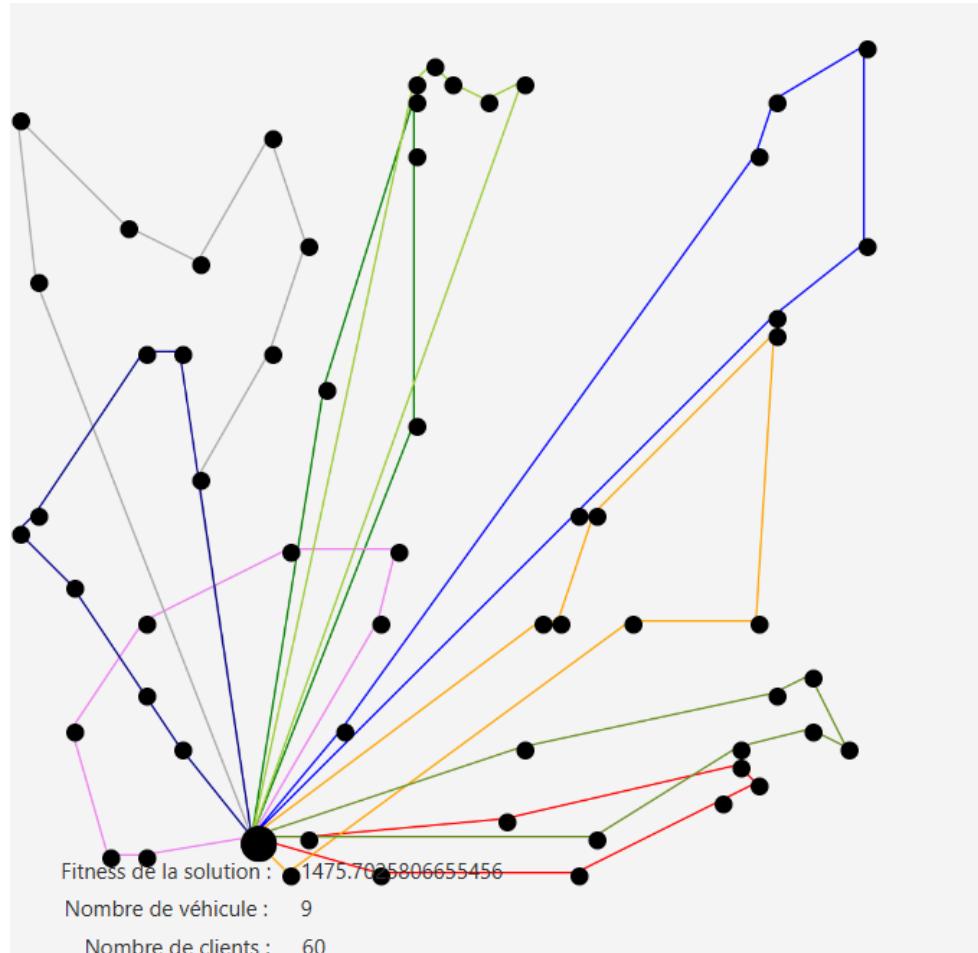
A6009

Tabou Taille 10



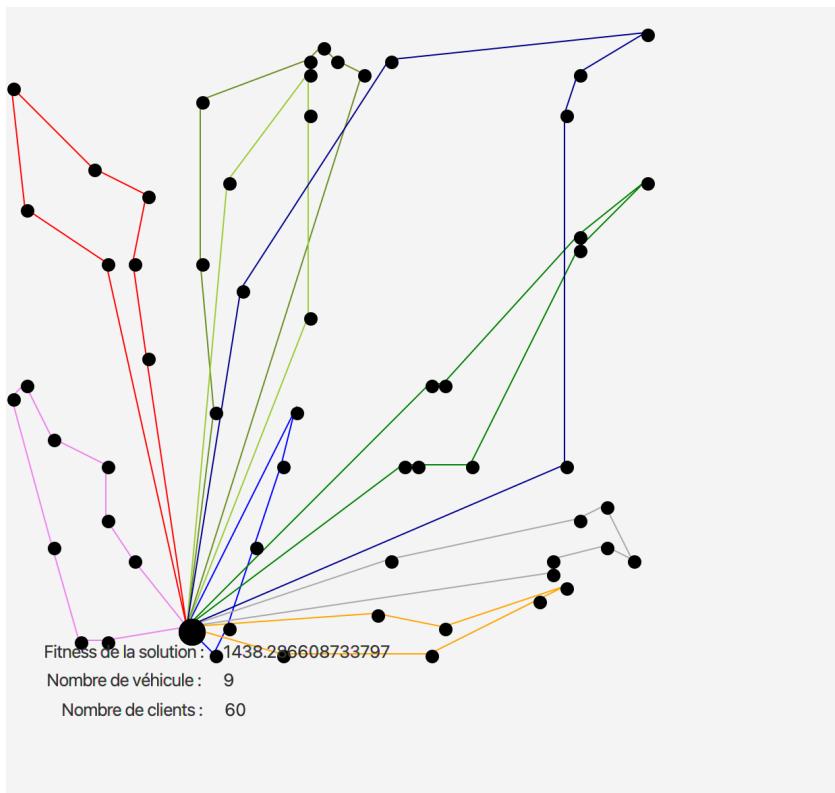
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2736ms	1619	1214185	9

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2046	1476	839916	9

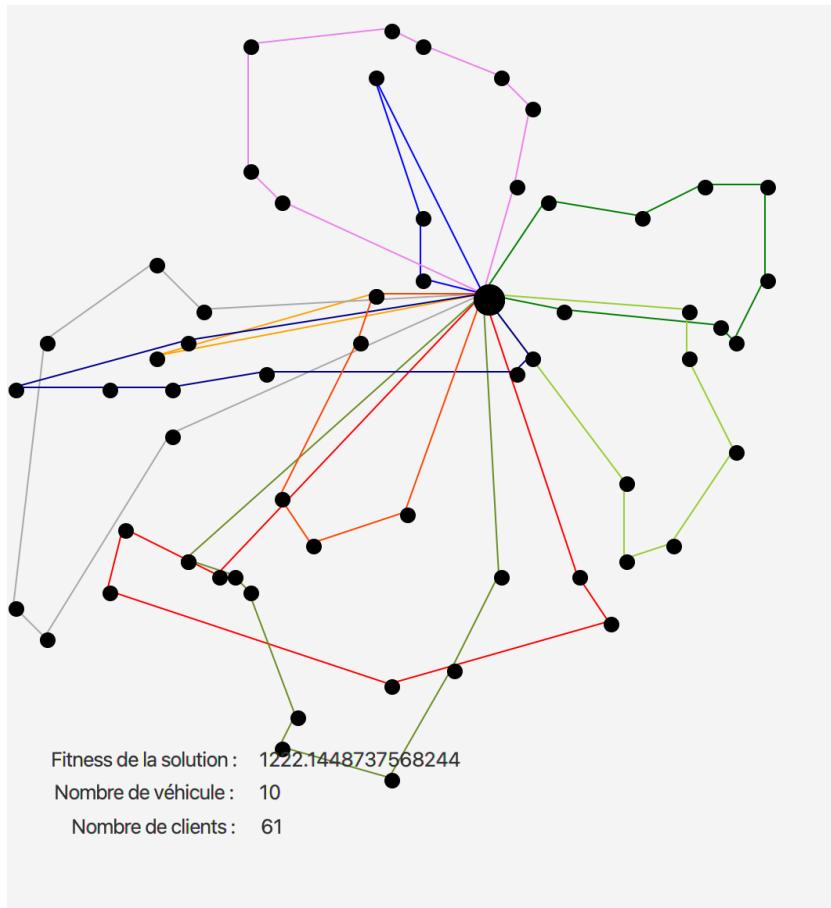
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1562	1442	839916	9

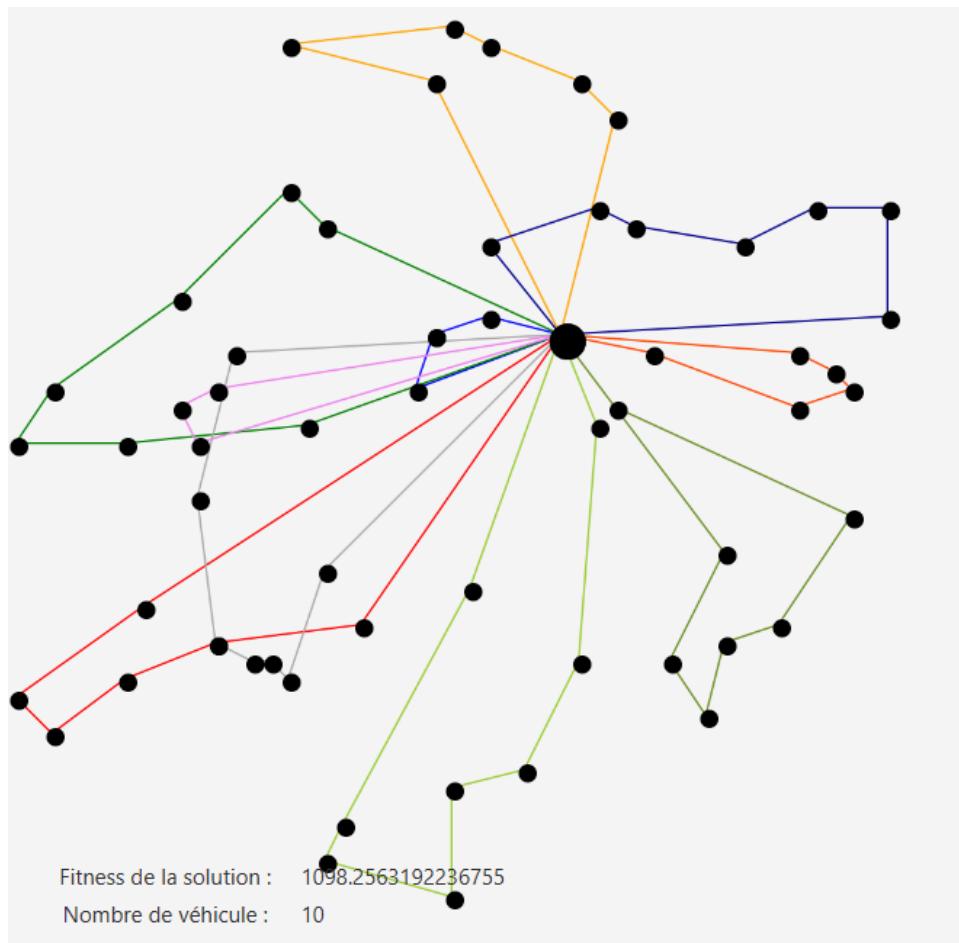
A6109

Tabou Taille 10



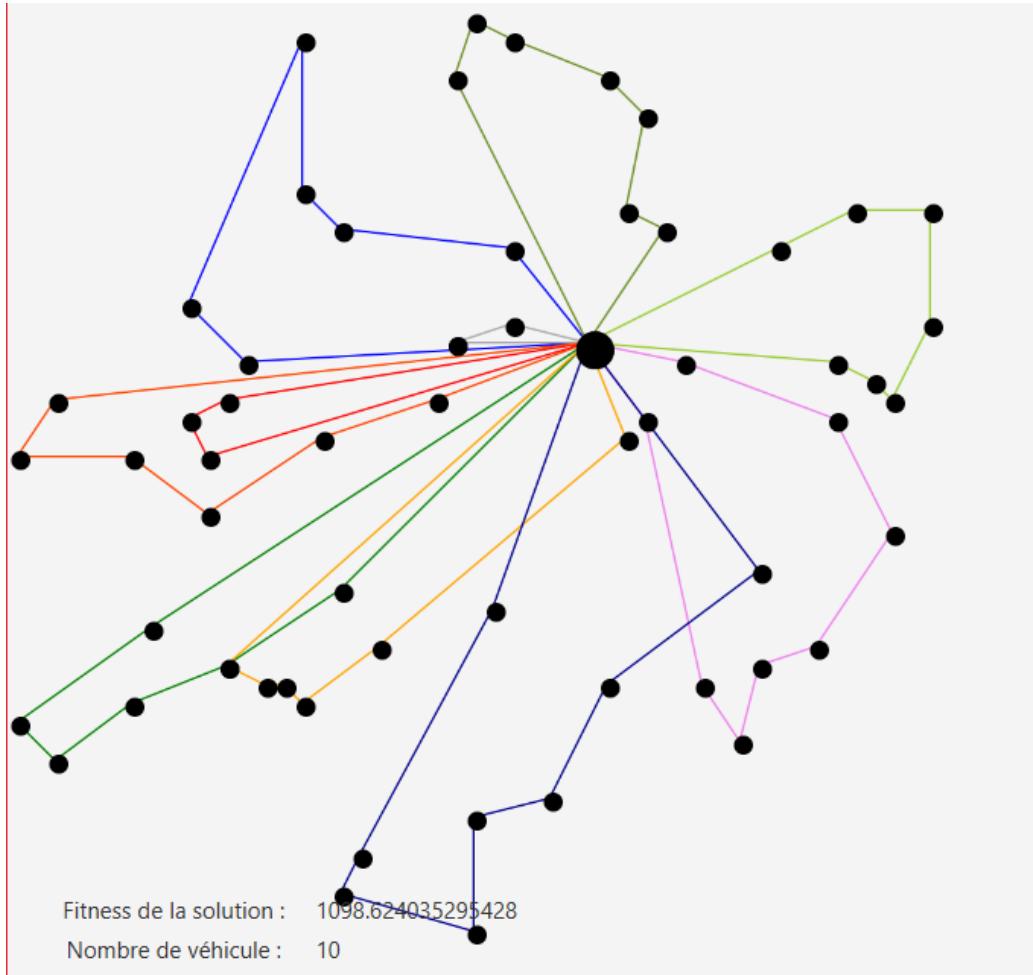
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2363ms	1222	1057393	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2046	1098	839916	10

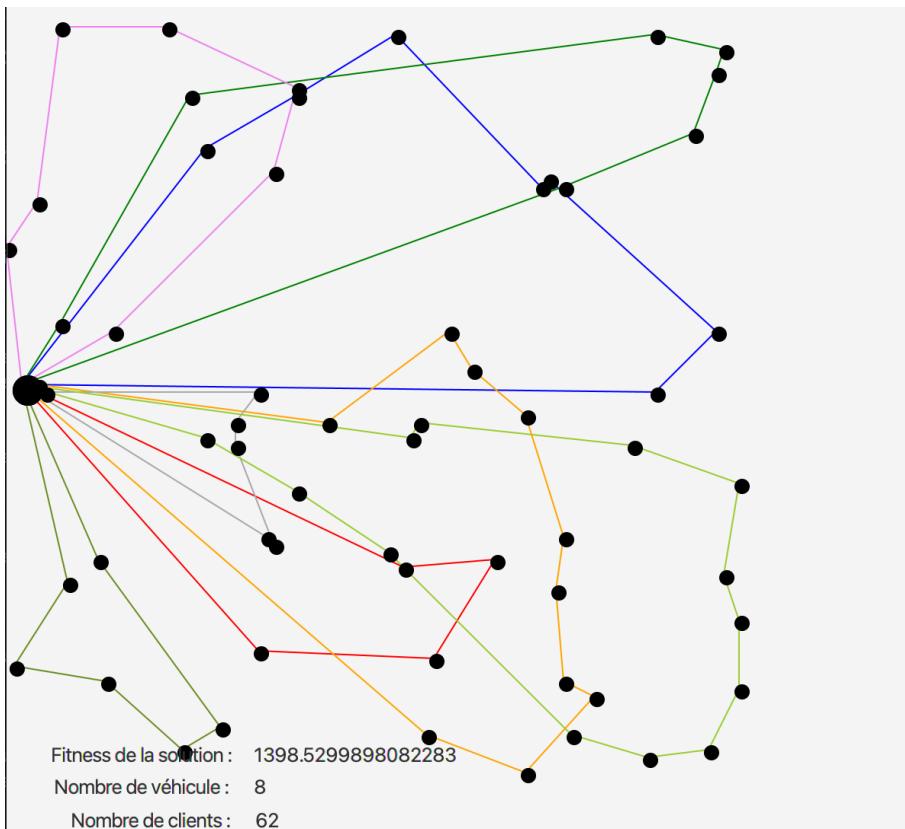
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1554	1098	839916	10

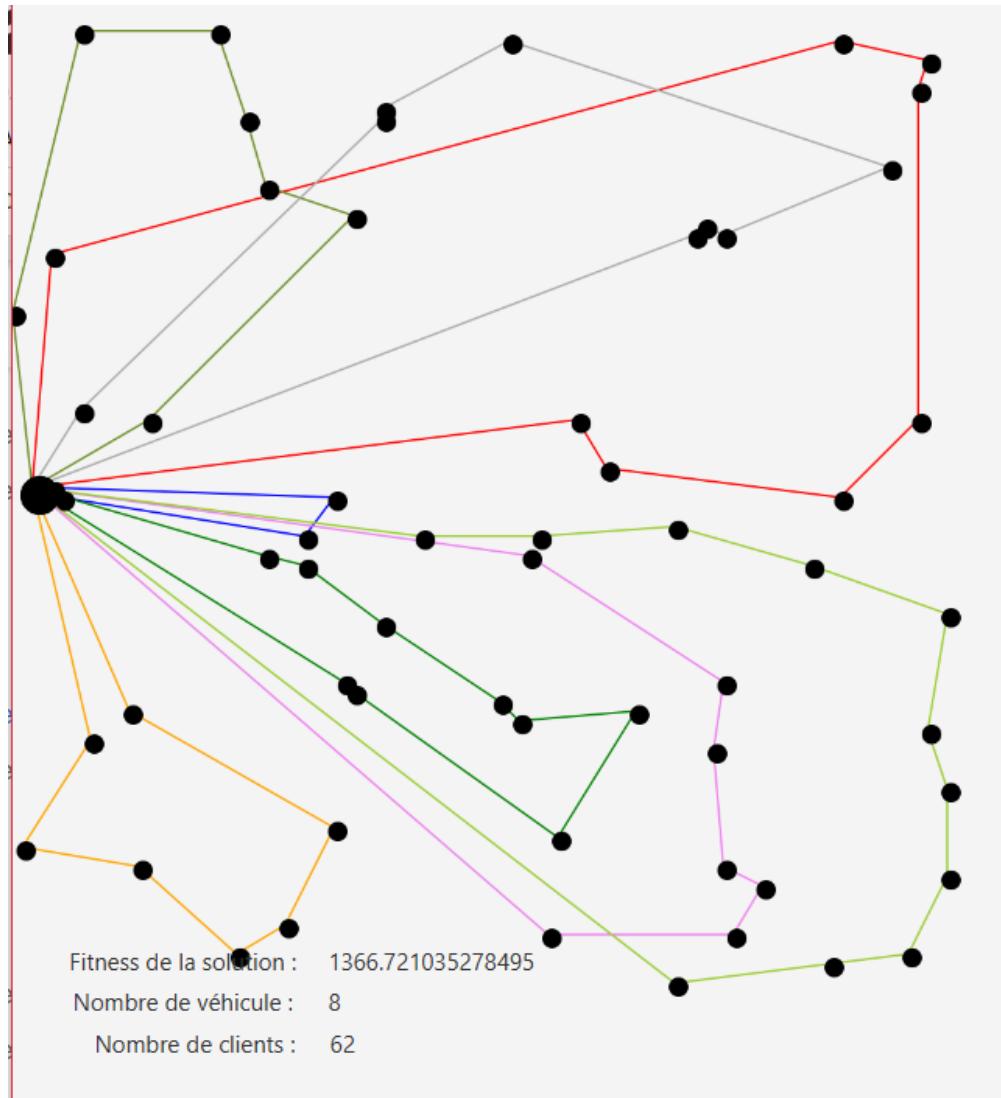
A6208

Tabou Taille 10



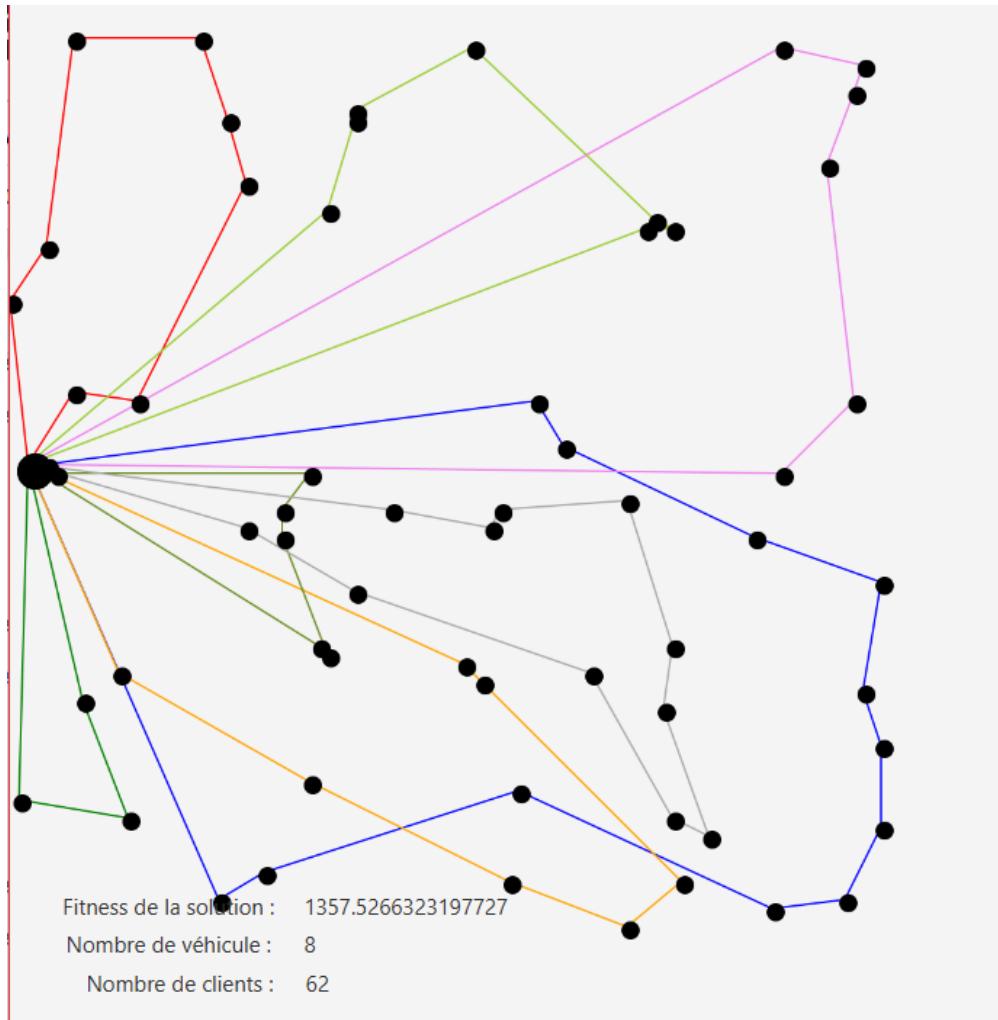
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
5064ms	1398	839916	8

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1537	1366	839916	8

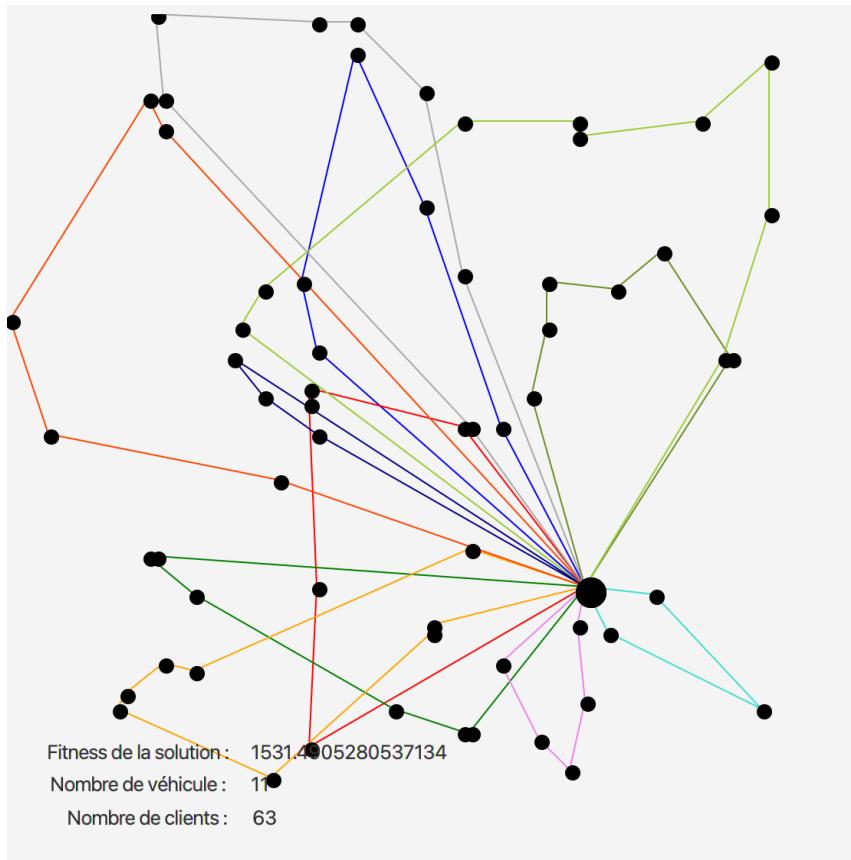
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1611	1357	839916	8

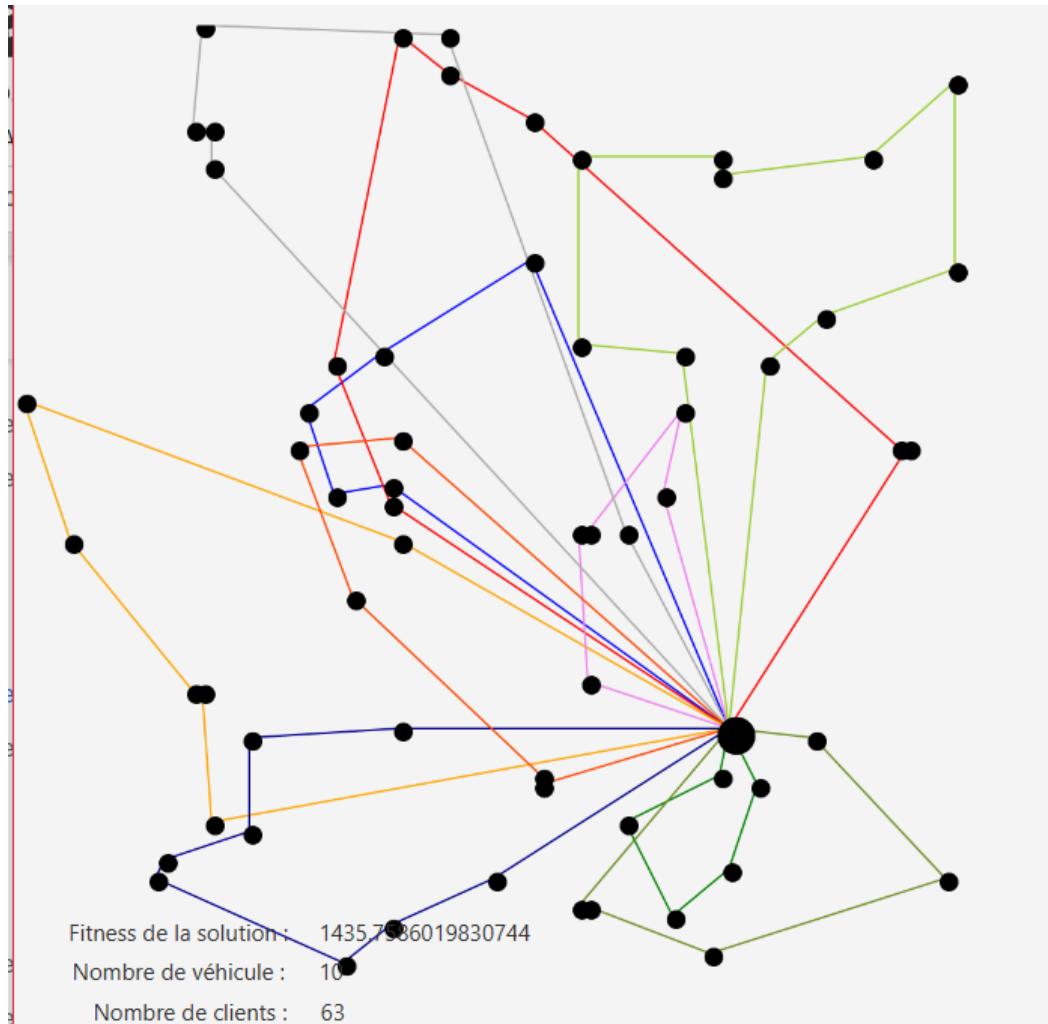
A6310

Tabou Taille 10



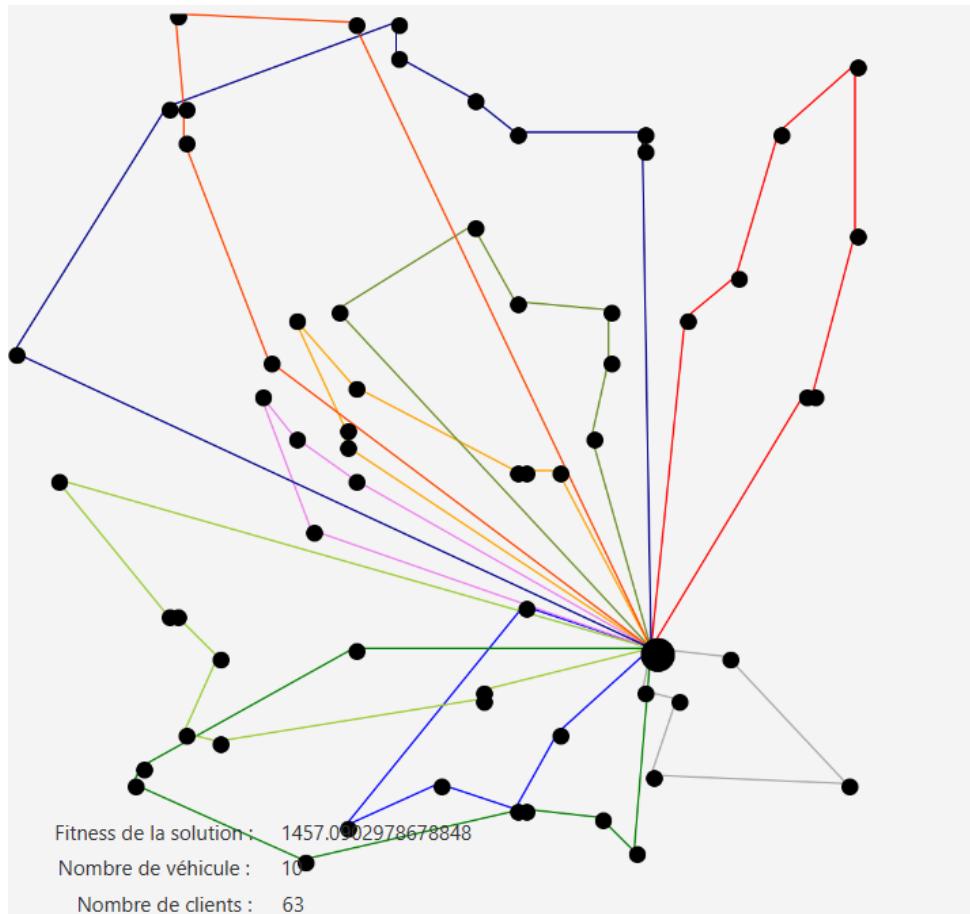
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
7835ms	1531	2940856	11

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1805	1435	839916	10

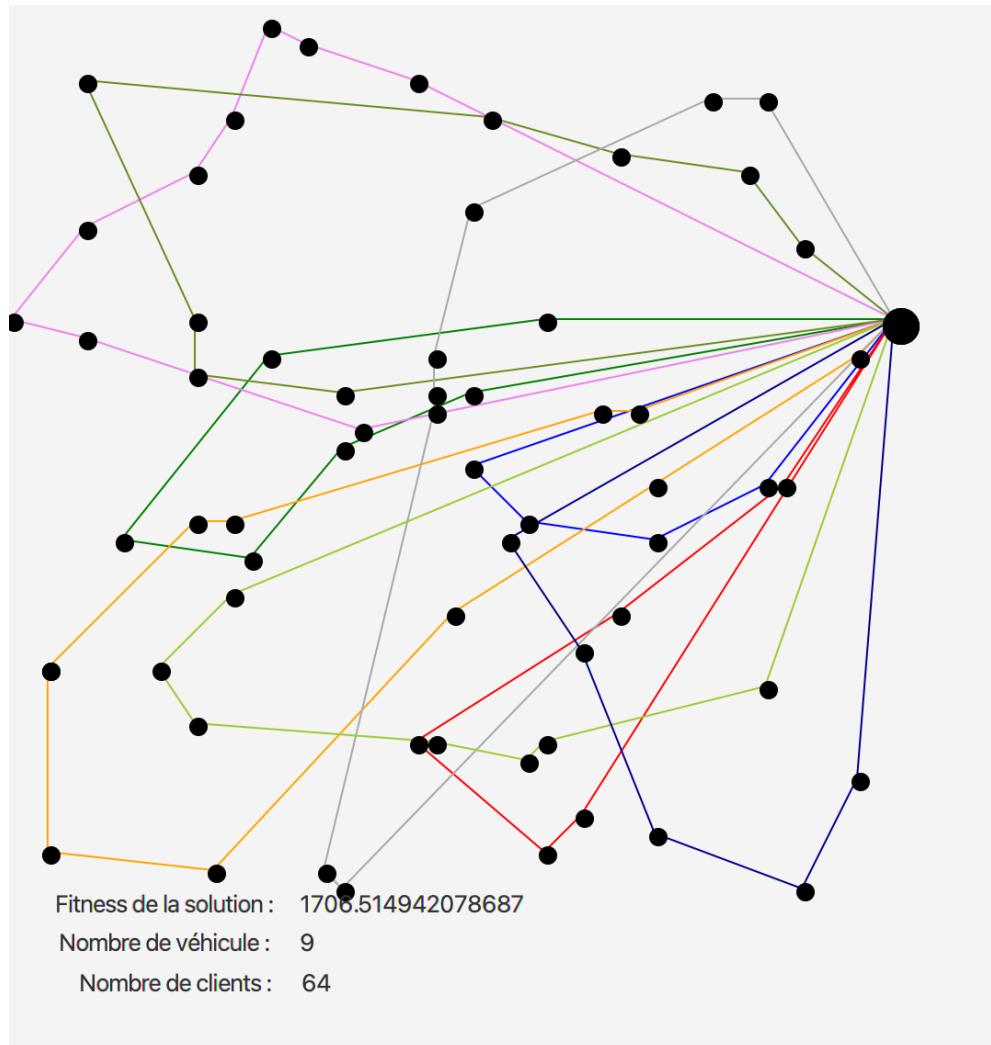
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1648	1457	839916	10

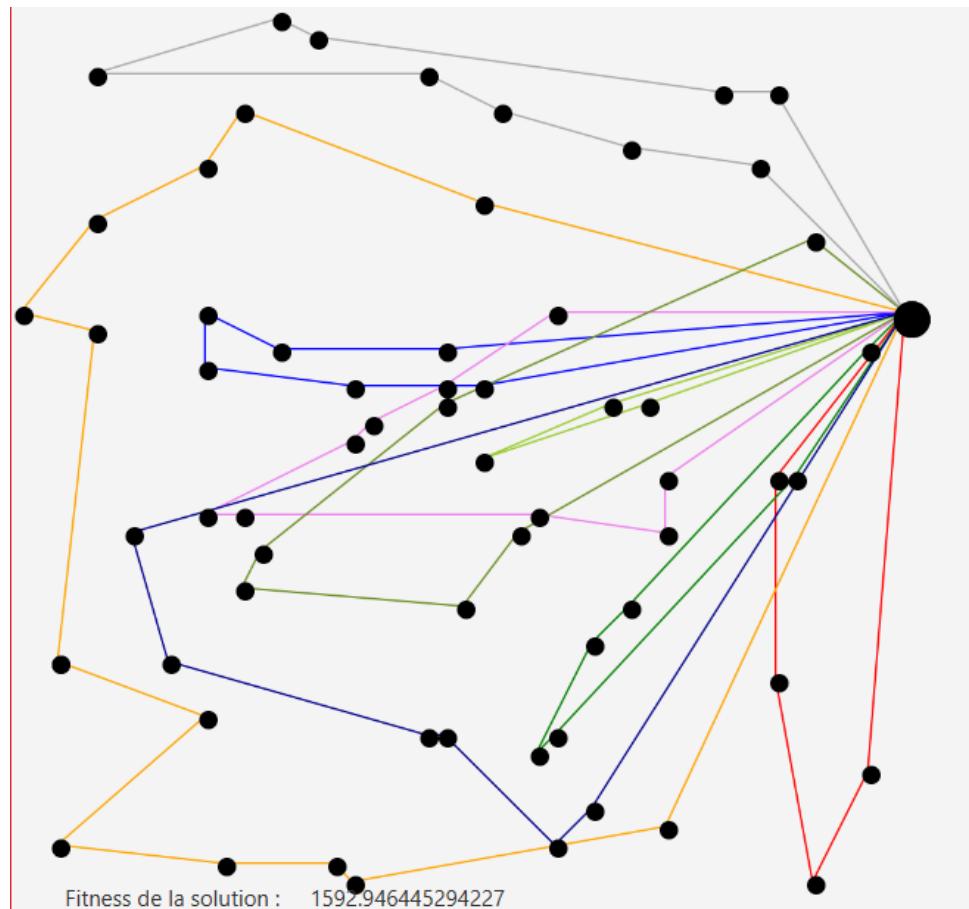
A6409

Tabou Taille 10



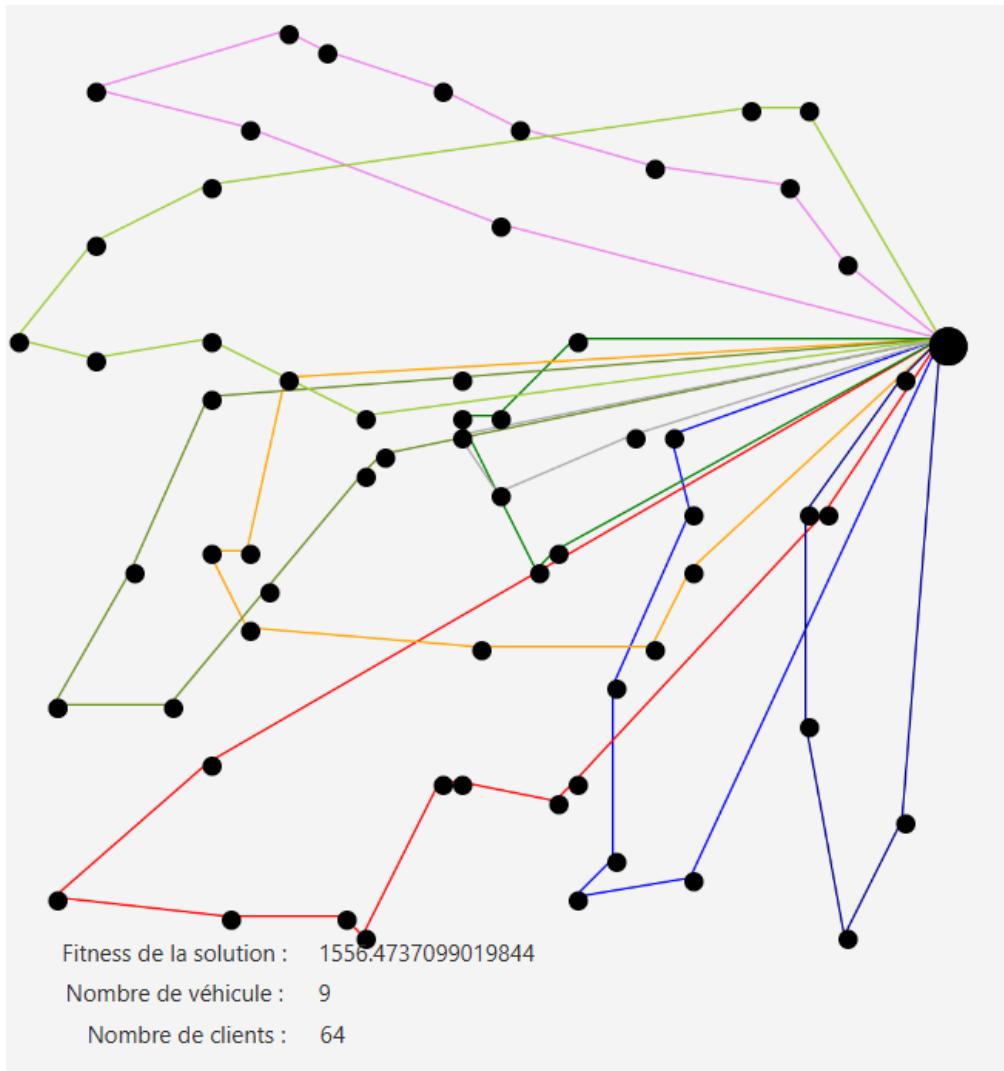
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2116ms	1706	911476	9

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1631	1592	839916	9

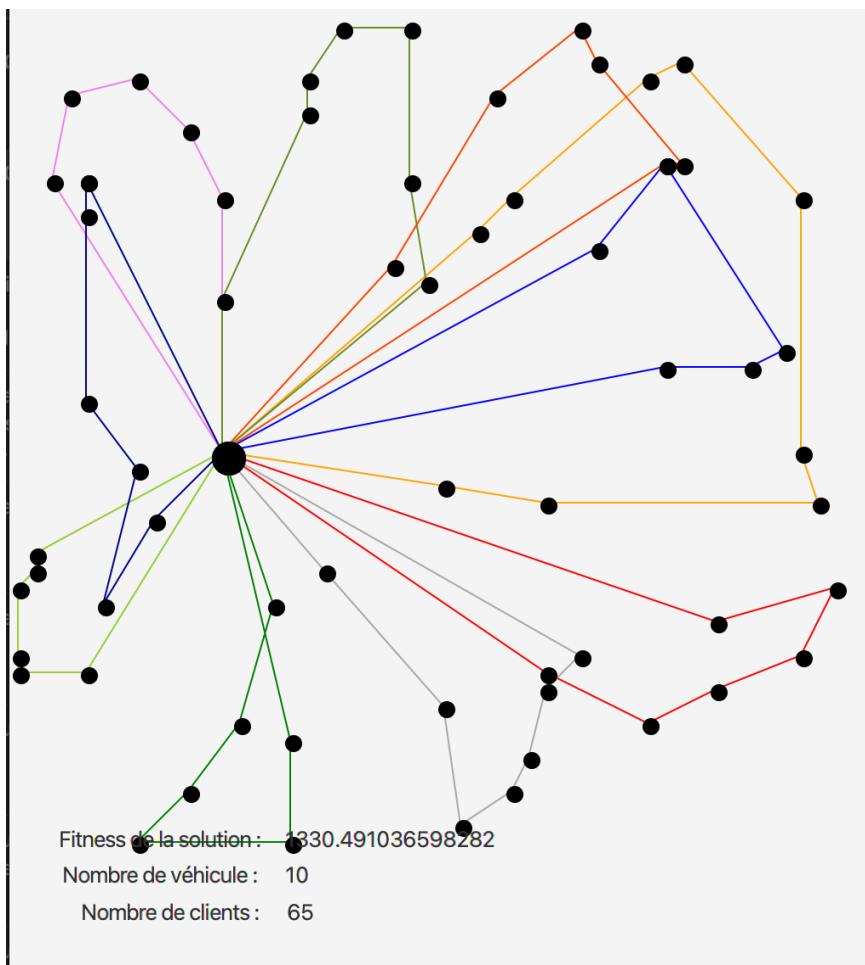
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1677	1556	839916	9

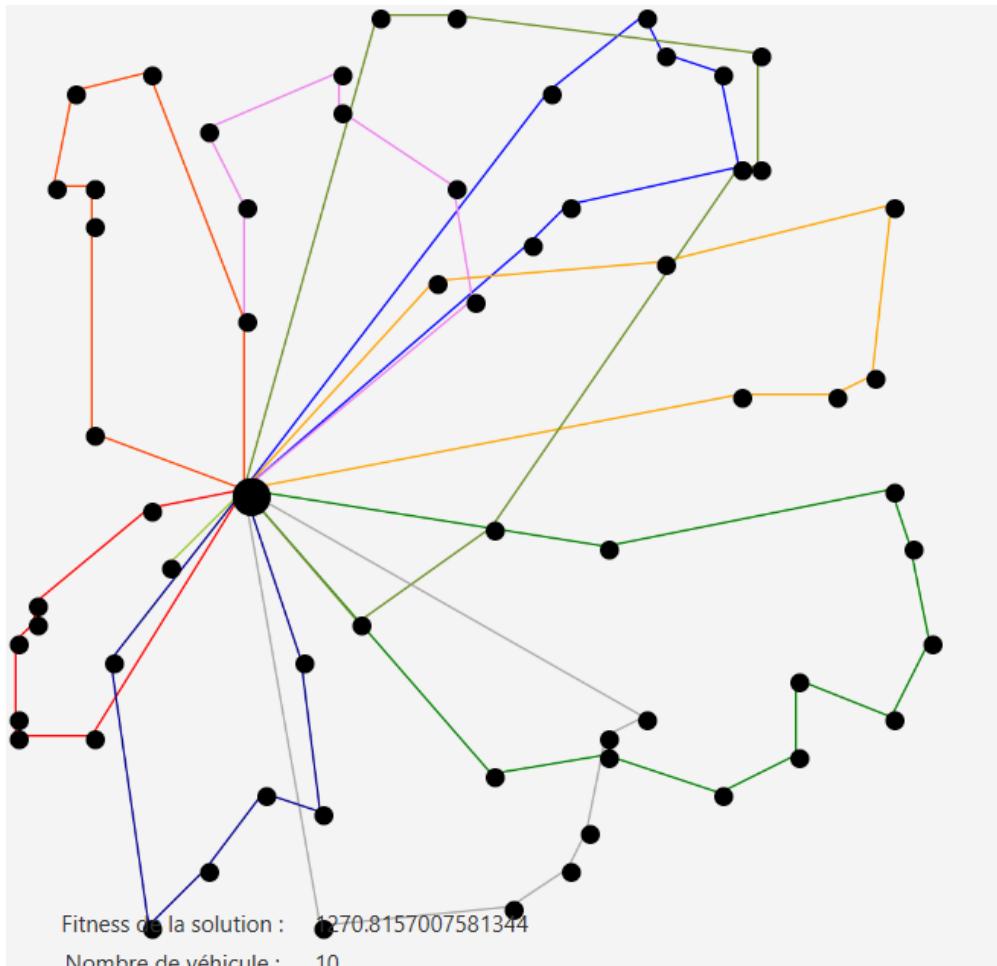
A6509

Tabou Taille 10



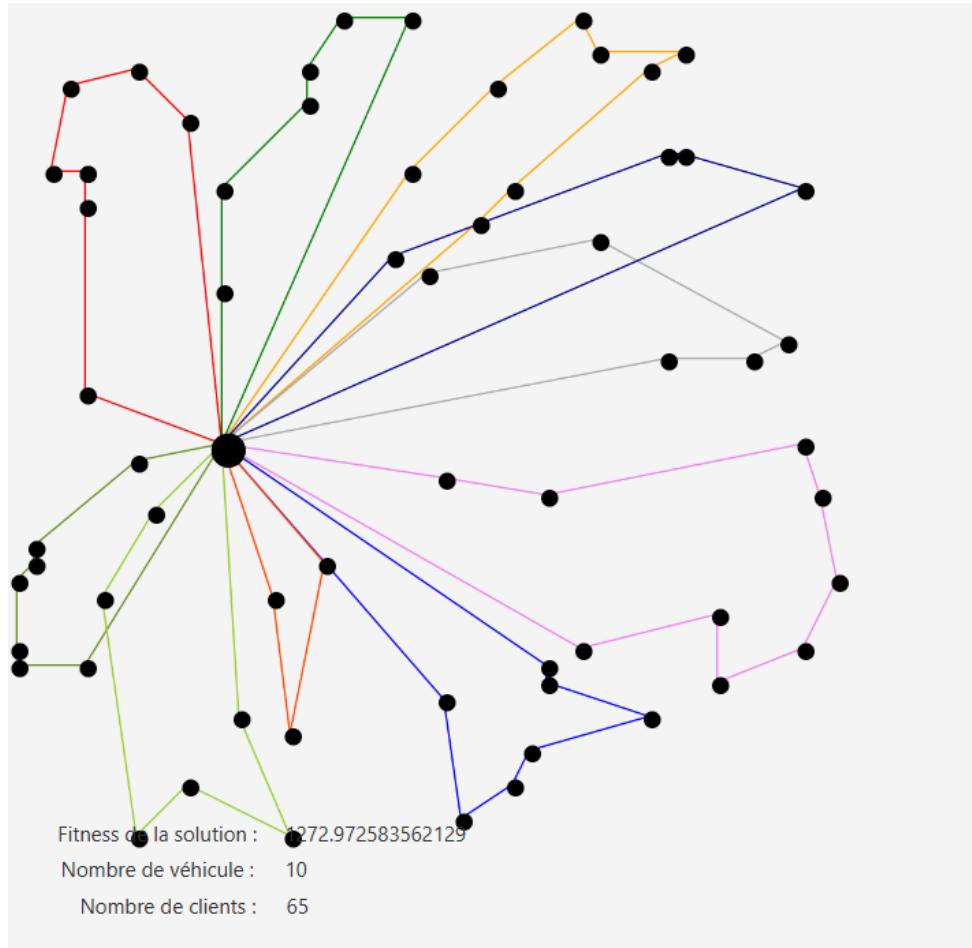
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
8329ms	1330	3103929	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1729	1270	839916	10

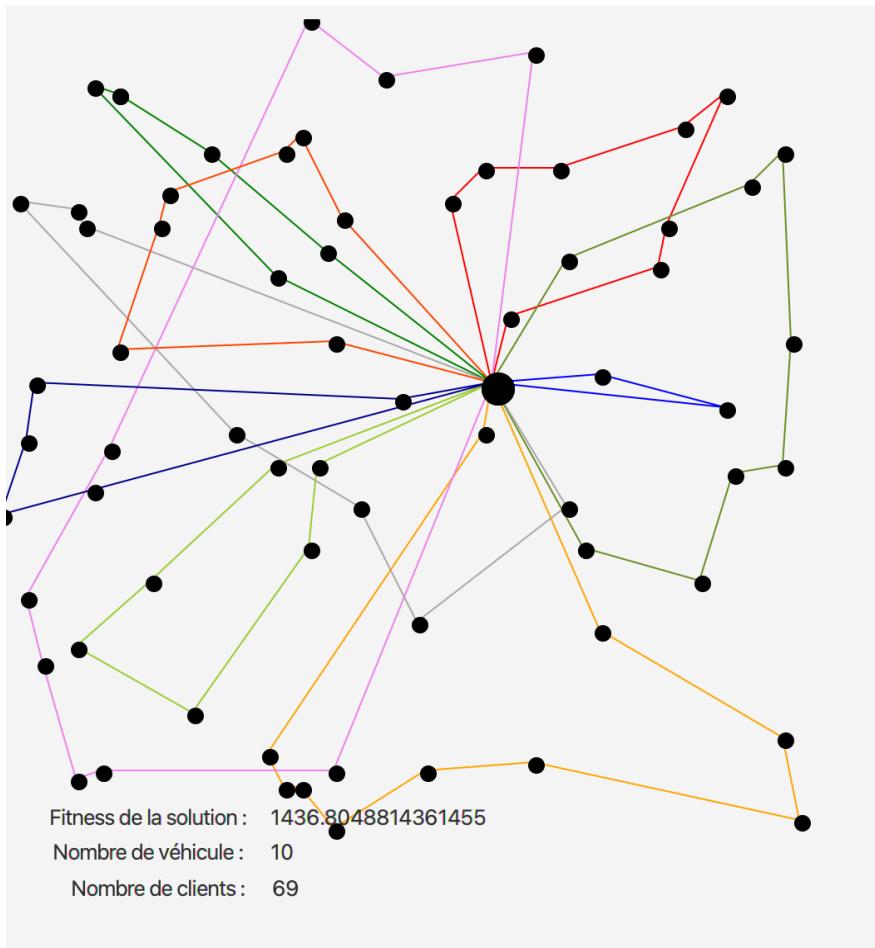
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1760	1272	839916	10

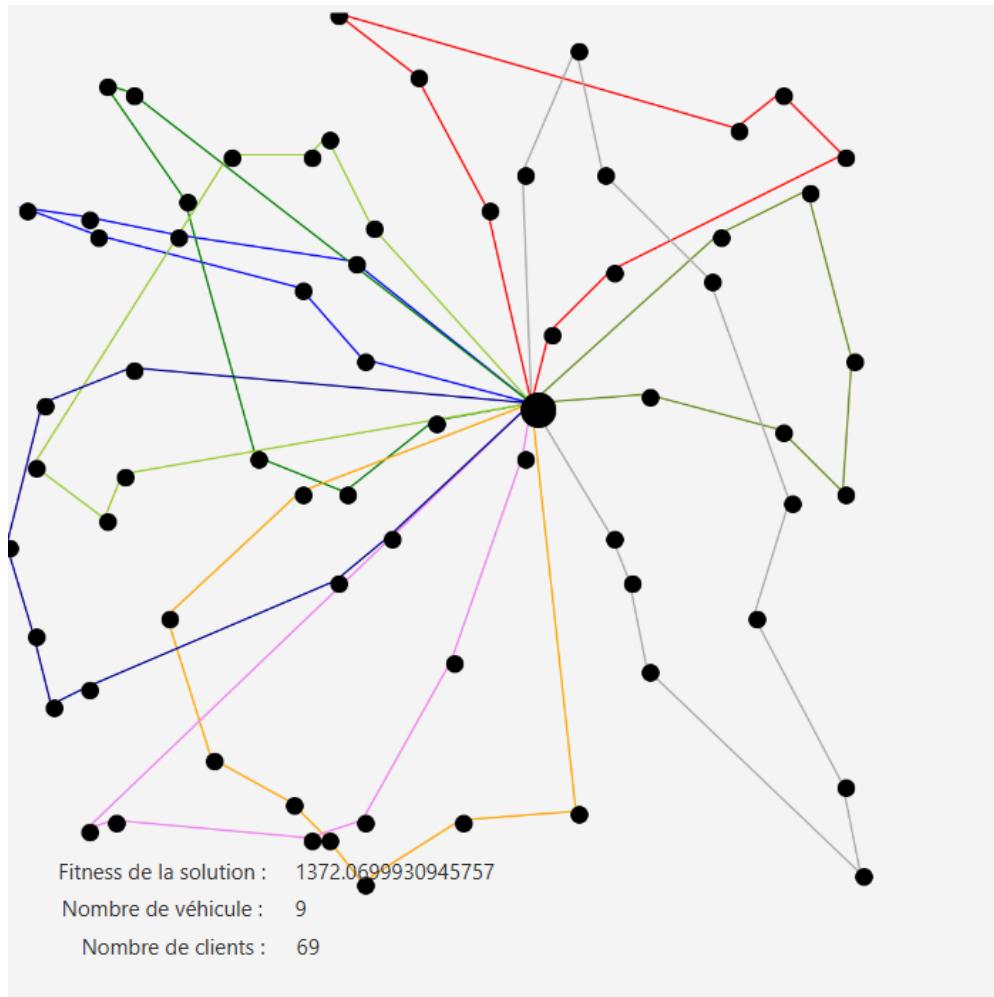
A6909

Tabou Taille 10



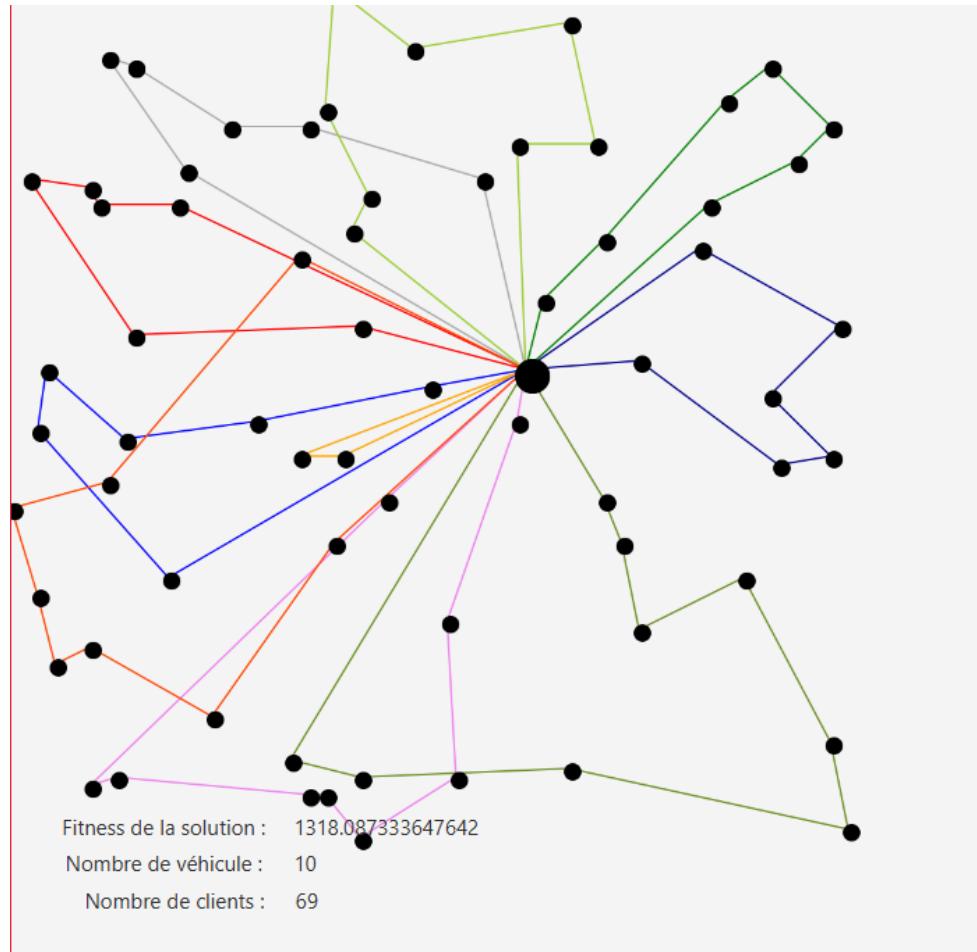
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
11927ms	1436	4000039	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1892	1372	839916	9

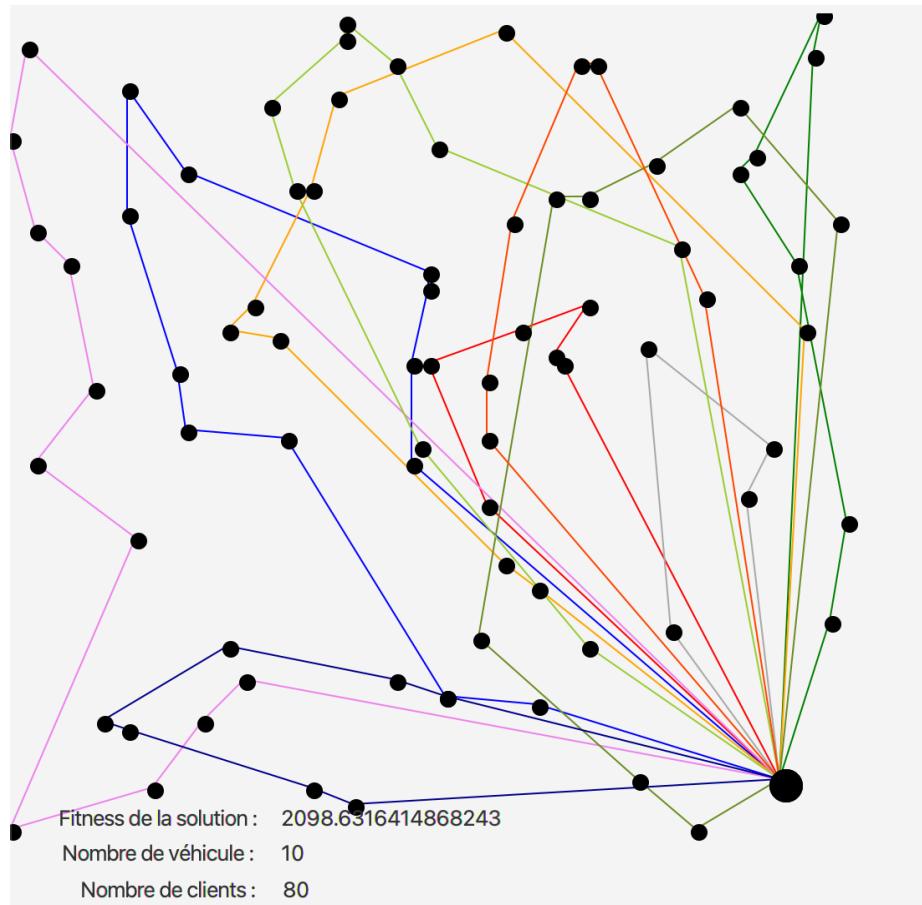
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1740	1318	839916	10

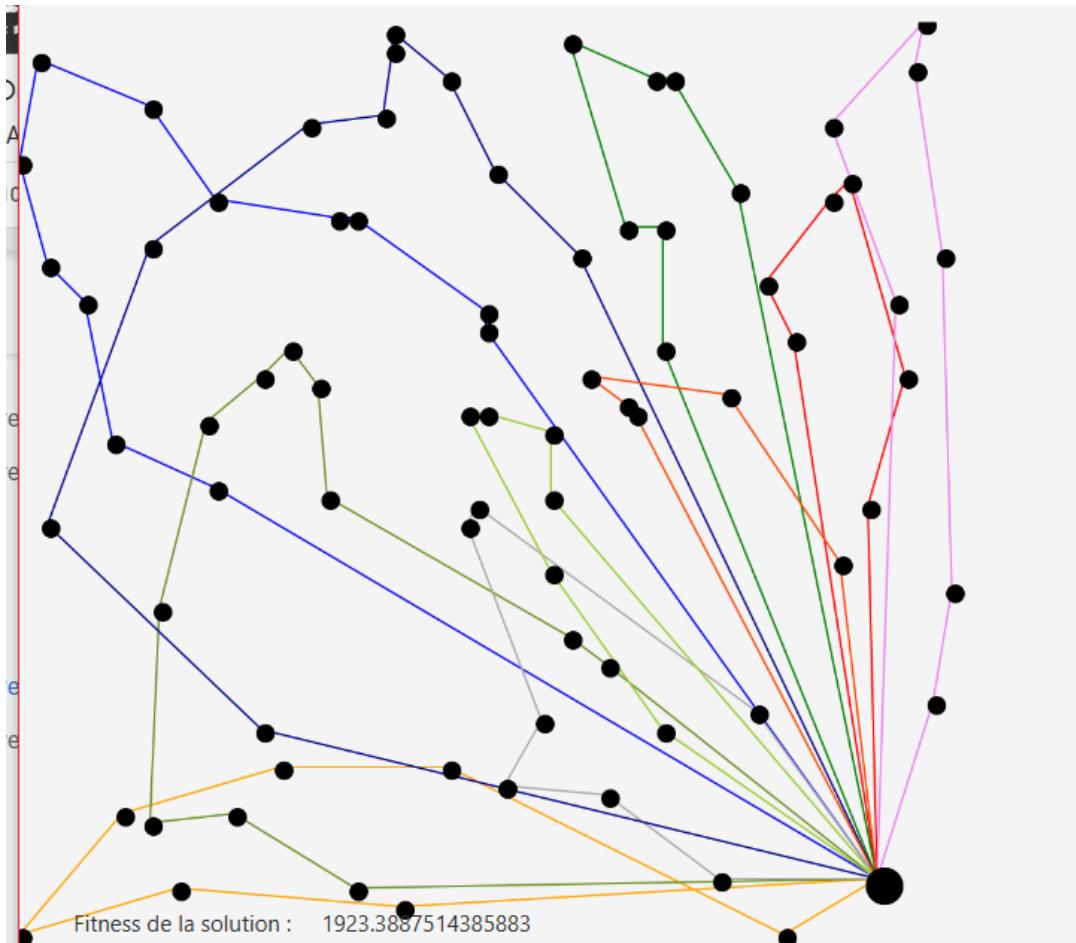
A8010

Tabou Taille 10



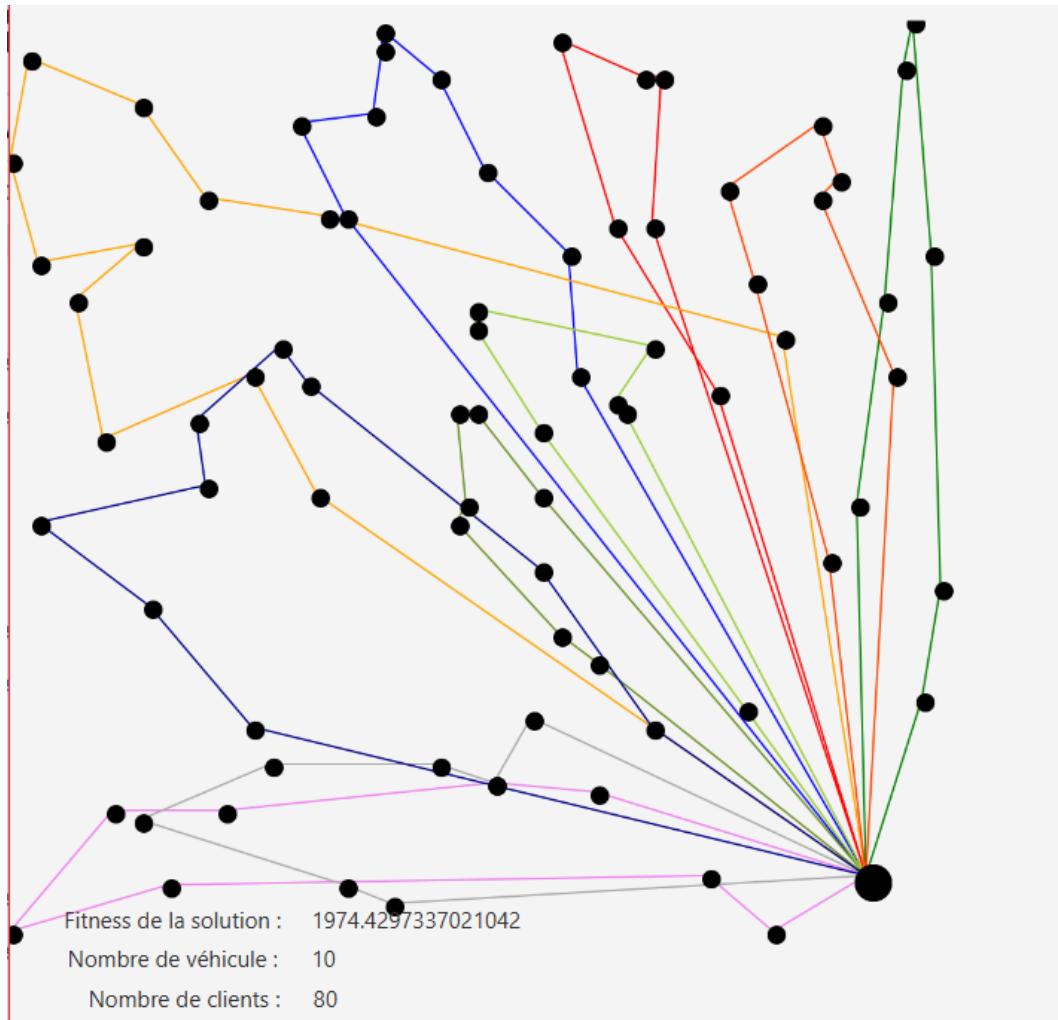
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
10587ms	2098	3368209	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2413	1923	839916	10

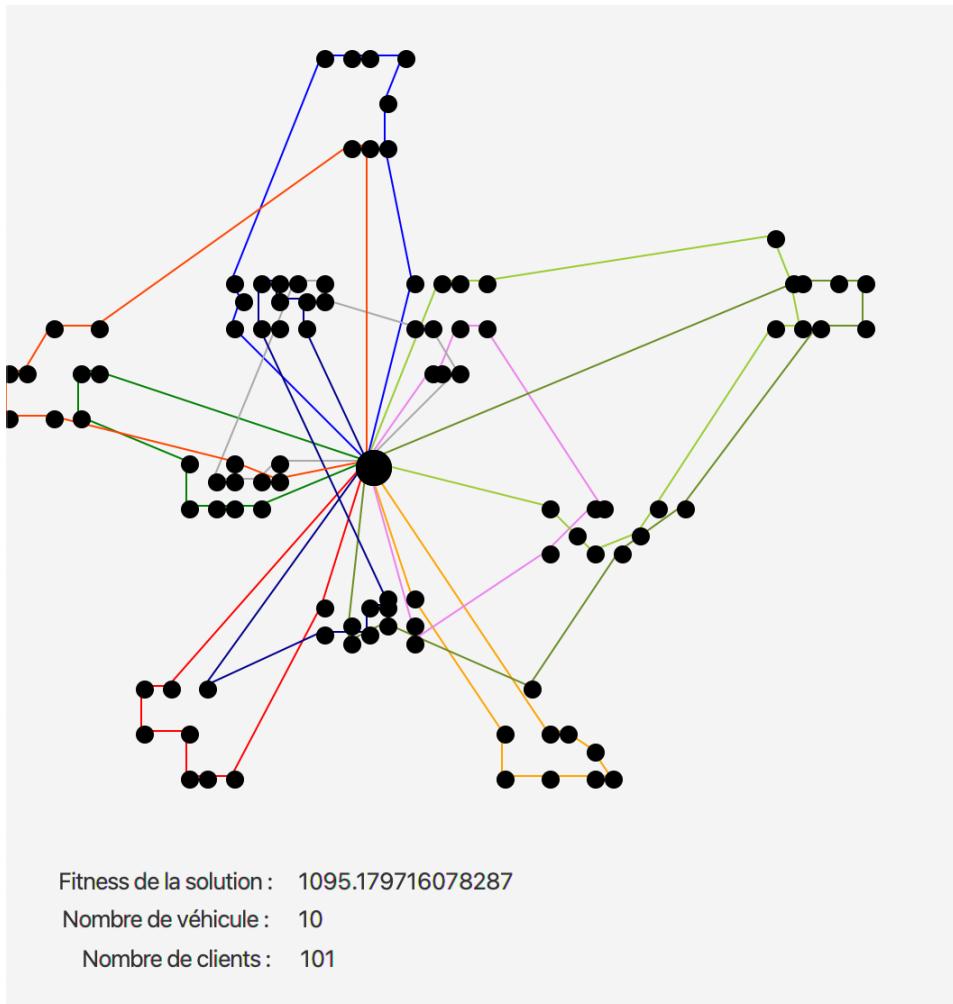
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
1975	1974	839916	10

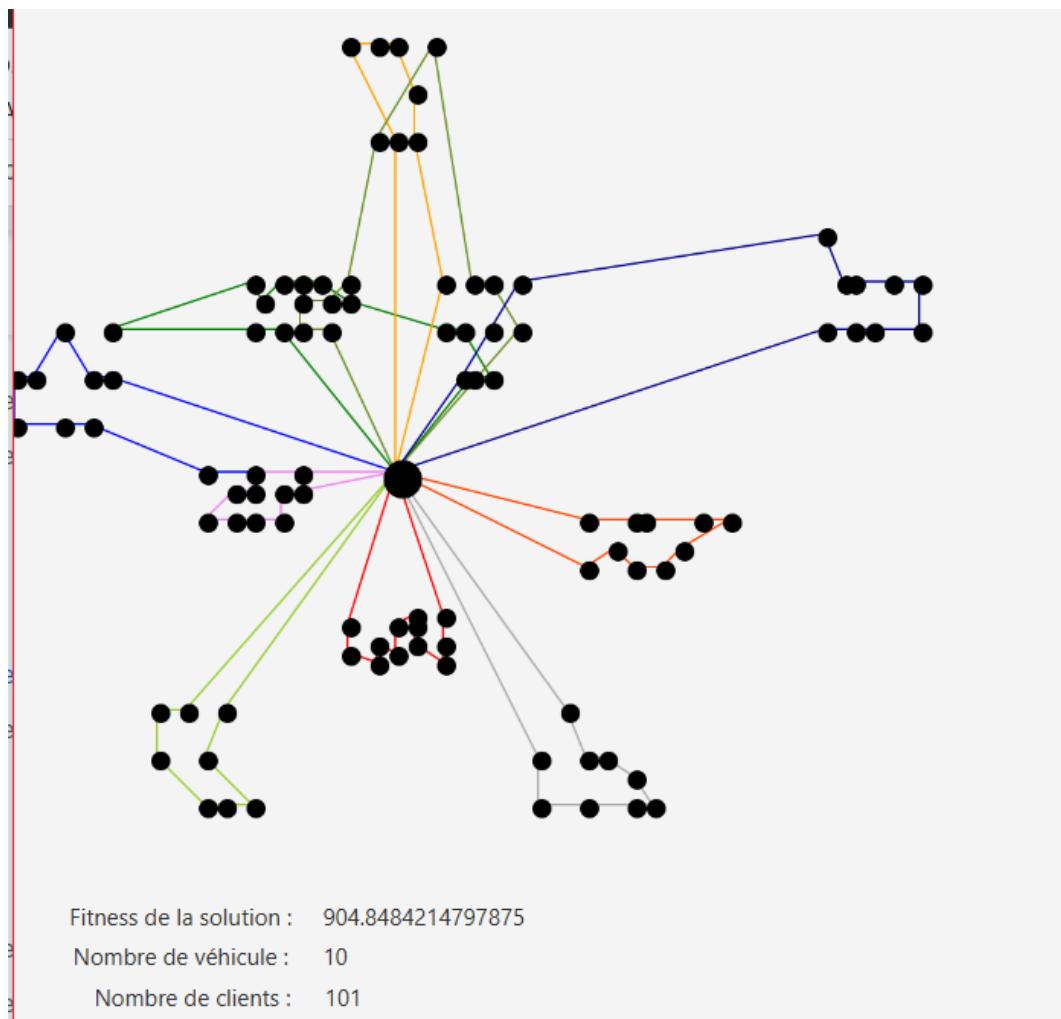
c101

Tabou Taille 10



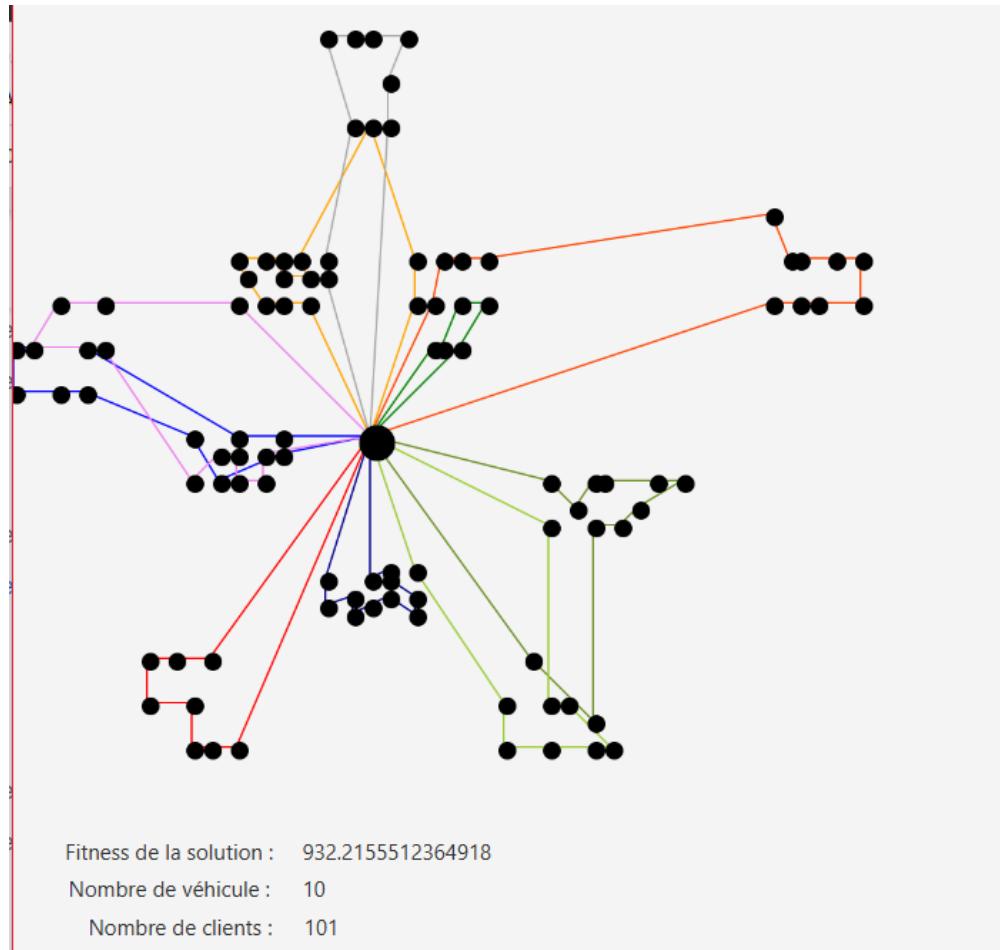
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
56185ms	1095	11 768 712	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2531	904	839916	10

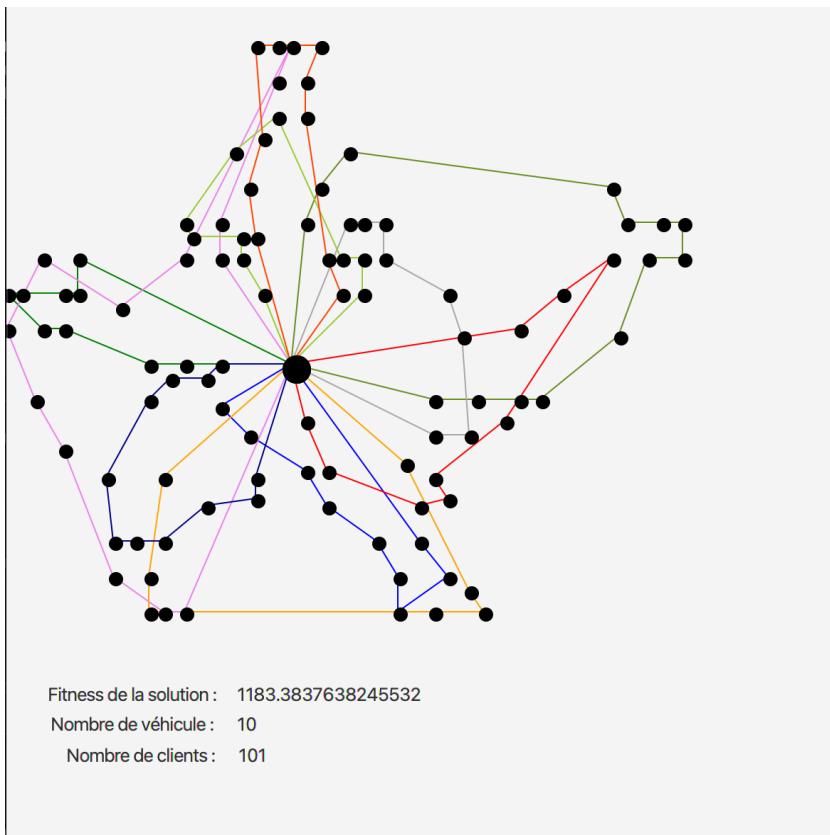
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2296	932	839916	10

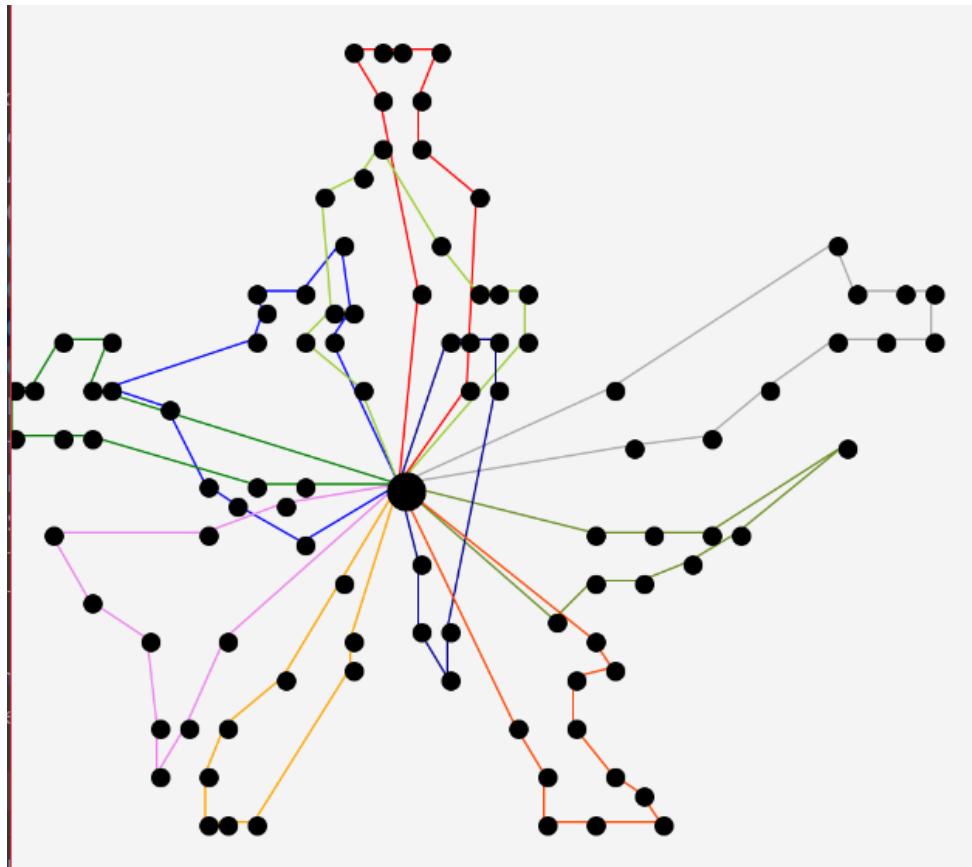
c201

Tabou Taille 10



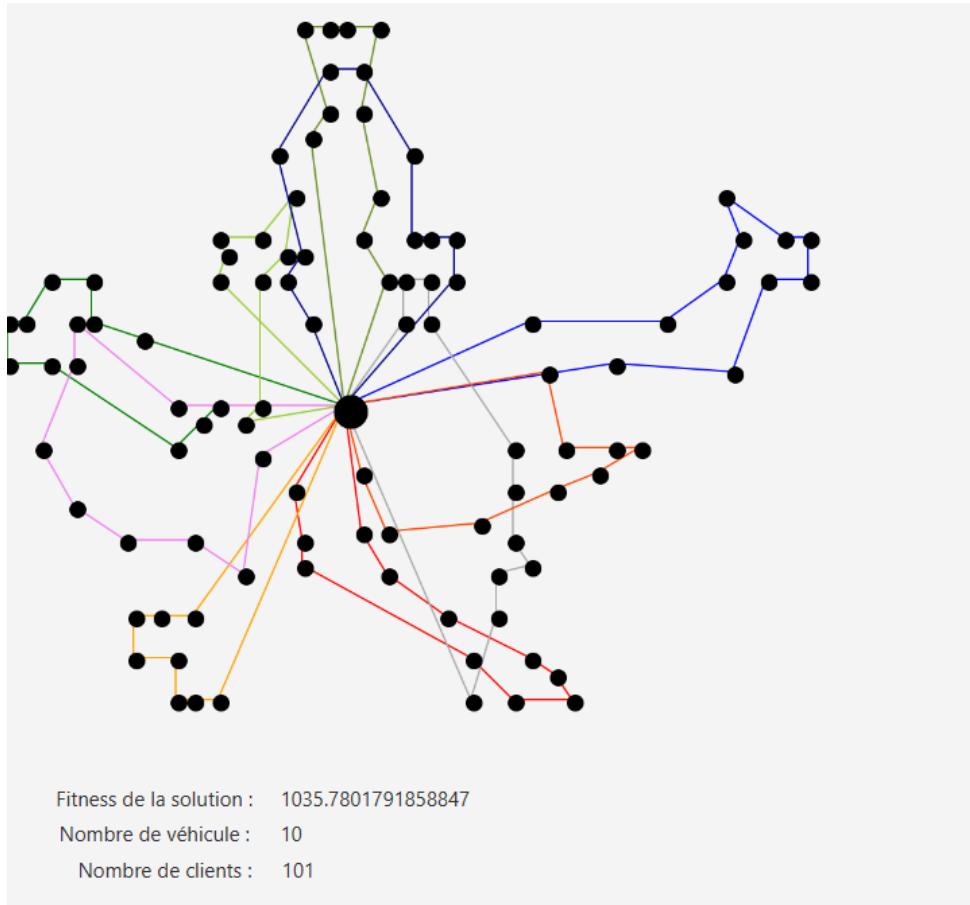
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
47134ms	1183	10299183	10

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2733	998	839916	10

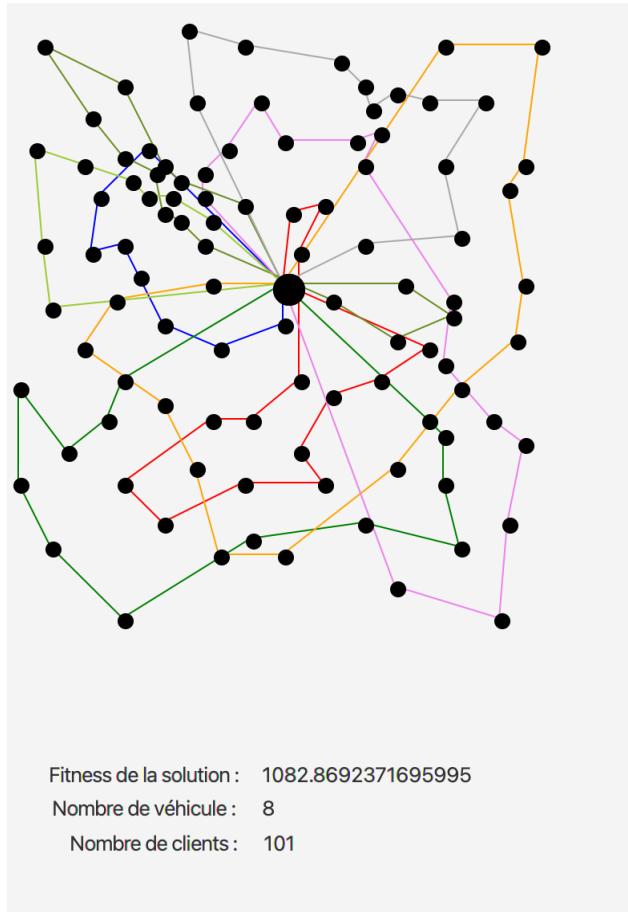
Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2336	1035	839916	10

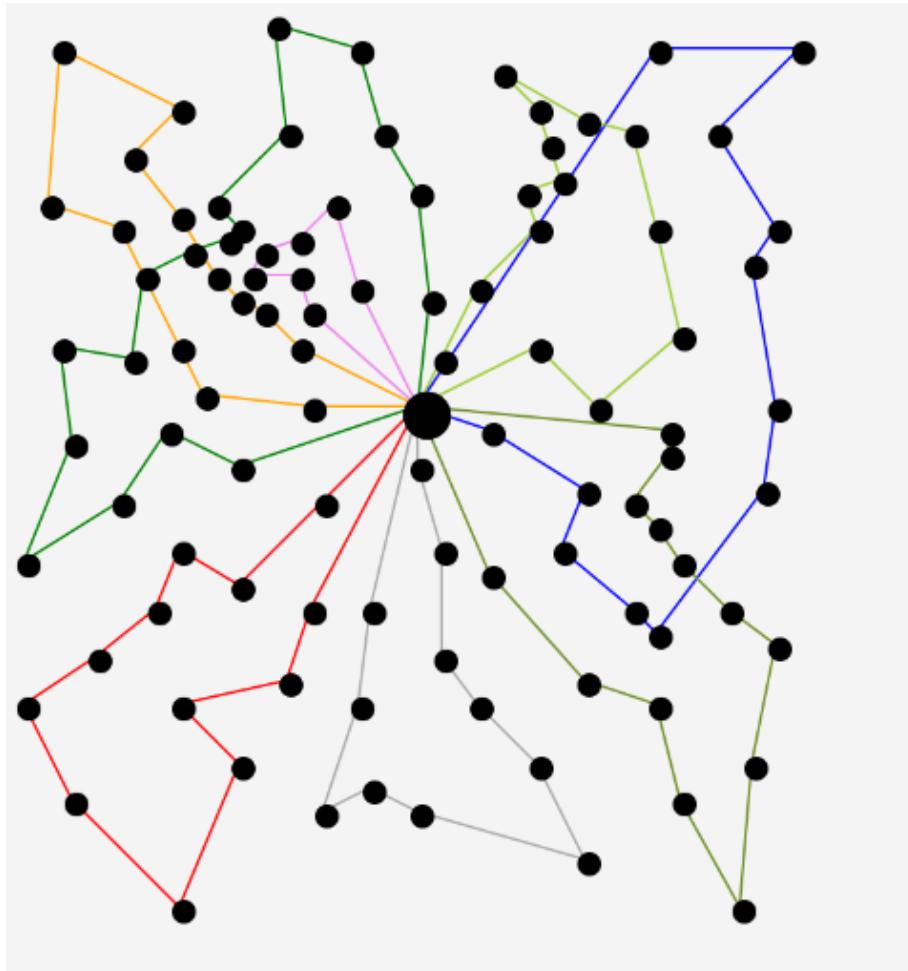
r101

Tabou Taille 10



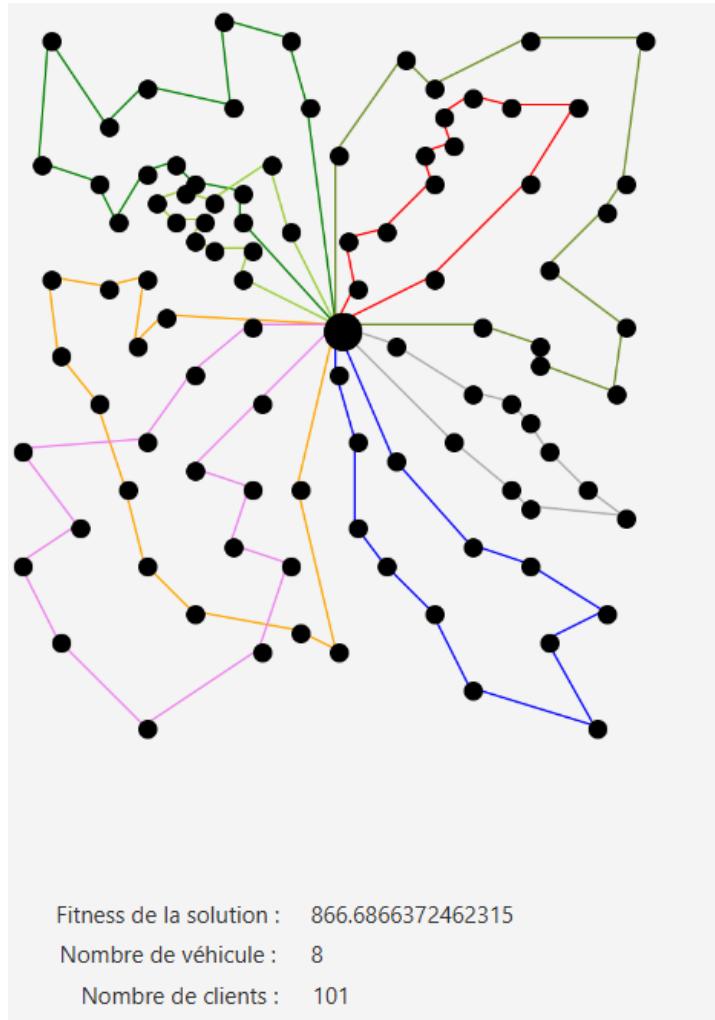
Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
144615ms	1082	24956807	8

Recuit Température = 50



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
3091	861	839916	10

Recuit Température = 200



Temps exécution	Fitness finale	Nombre de solutions générées	Nombre de véhicule final
2323	866	839916	8

Analyse de nos résultats



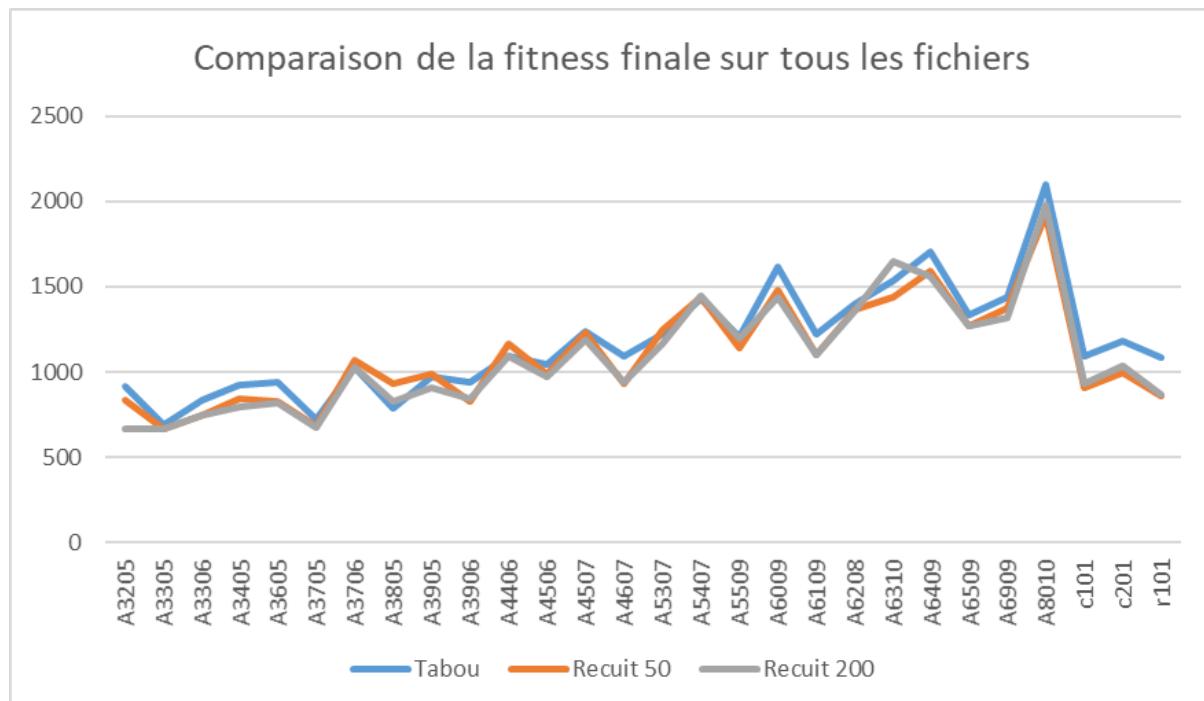
Fitness finale

Sur les 28 fichiers que nous avons testé, nous pouvons nous apercevoir sur le graphique suivant que globalement, nos deux algorithmes avec tous leurs paramétrages trouvent les mêmes résultats.

Sur les 3 derniers fichiers (*qui sont des fichiers qui contiennent chacun plus de 100 clients*), nous pouvons voir que Tabu possède des résultats qui sont bien moins optimisés que ceux réalisés par l'algorithme Recuit.

Si l'on compare l'algorithme Recuit qui à été lancé avec les deux paramétrages différents :

- Celui avec la **température à 50** possède un **meilleur résultat dans environ 40% des cas (39,29%)**.
- Celui avec la **température à 200** possède un **meilleur résultat dans 53% des cas (53.57%)**
- Les **résultats arrondis** sont les **mêmes dans 7% des cas (2 cas sur 28)**.



Temps d'exécution

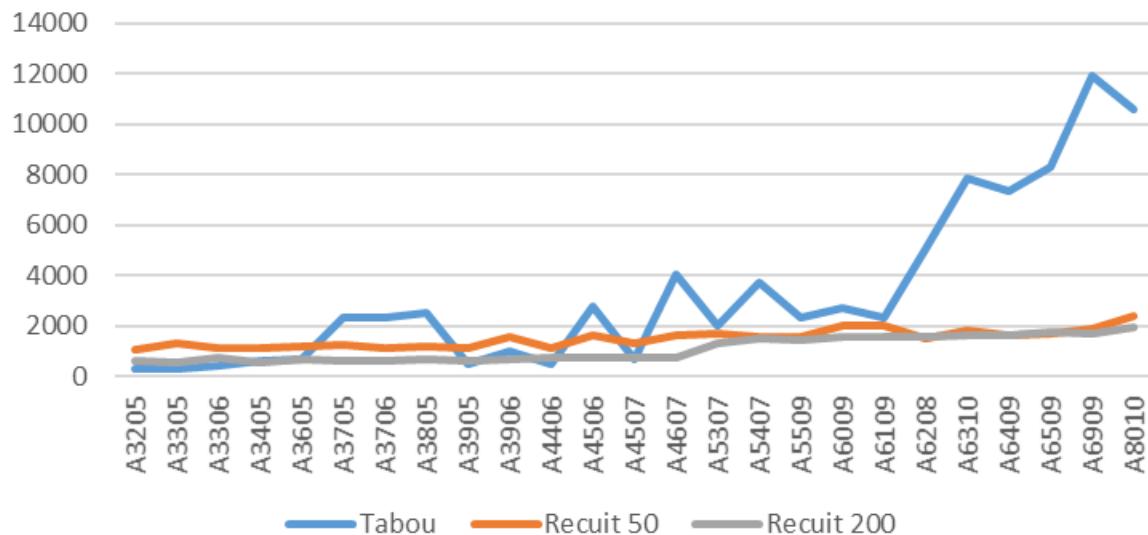
Pour le temps d'exécution, les algorithmes ont été lancés sur 3 machines très différentes. Ainsi, certains résultats concernant les temps d'exécution sont très variables et ne sont pas forcément représentatifs.

Toutefois, cela permet aussi de constater au travers de cet inconvénient que les composants utilisés influent donc sur le calcul de nos solutions.

Sur ce graphique, nous nous apercevons que :

- **Recuit** avec une **température à 50 et 200** possèdent des *temps d'exécution relativement similaires*.
- Sur la plupart des fichiers, **recuit 50 est plus rapide que recuit 200** (*ce qui est normal car avoir une température de 50 permet de réduire le champs des possibilités pour les voisins*).
- Notre **temps d'exécution de Tabou** est plus ou moins **constant lors des premiers fichiers**, mais lorsqu'on arrive aux **fichiers qui contiennent plus de données**, les **temps augmentent** très rapidement (*notamment à partir du fichier A6208*).

Temps d'exécution de l'algorithme en fonction du fichier et de l'algorithme choisi

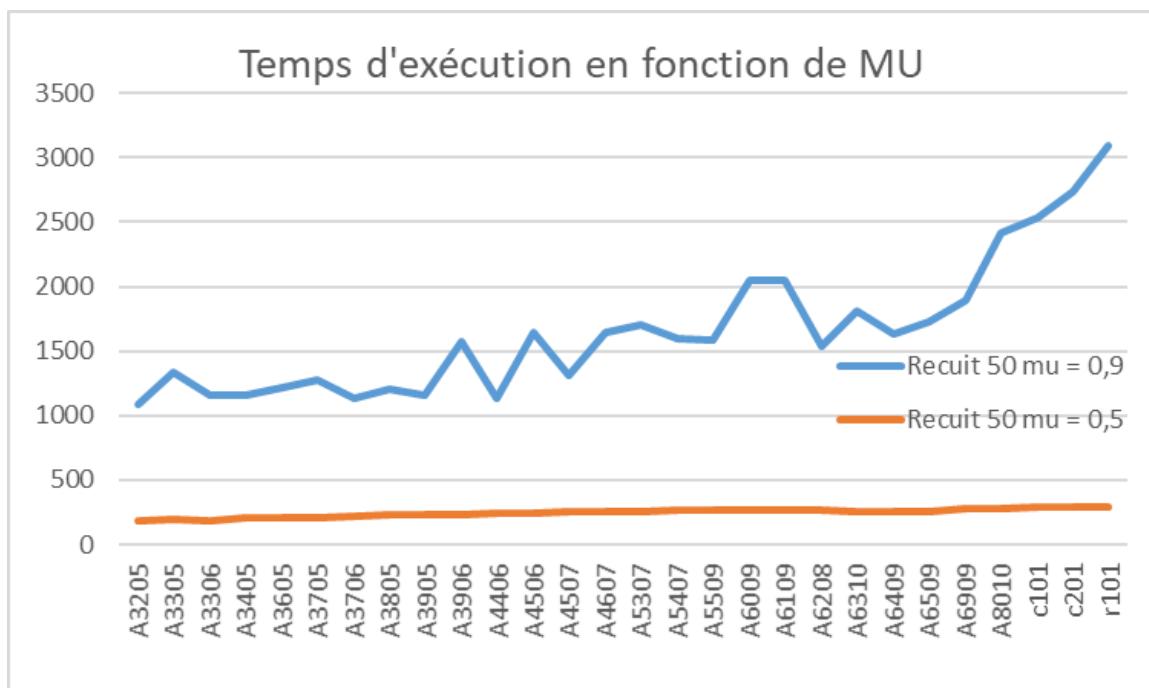


Modification de la valeur de MU

Un des paramètres que nous avons laissé modifiable est la valeur de μ . Cette valeur influe sur le nombre de solutions qui seront générées car elle influe sur le nombre d'itérations qui seront effectuées.

Exemple

- pour $\mu = 0.5$ nous avons **119988** solutions générées
- alors que nous en avons **839916** pour $\mu = 0.9$.

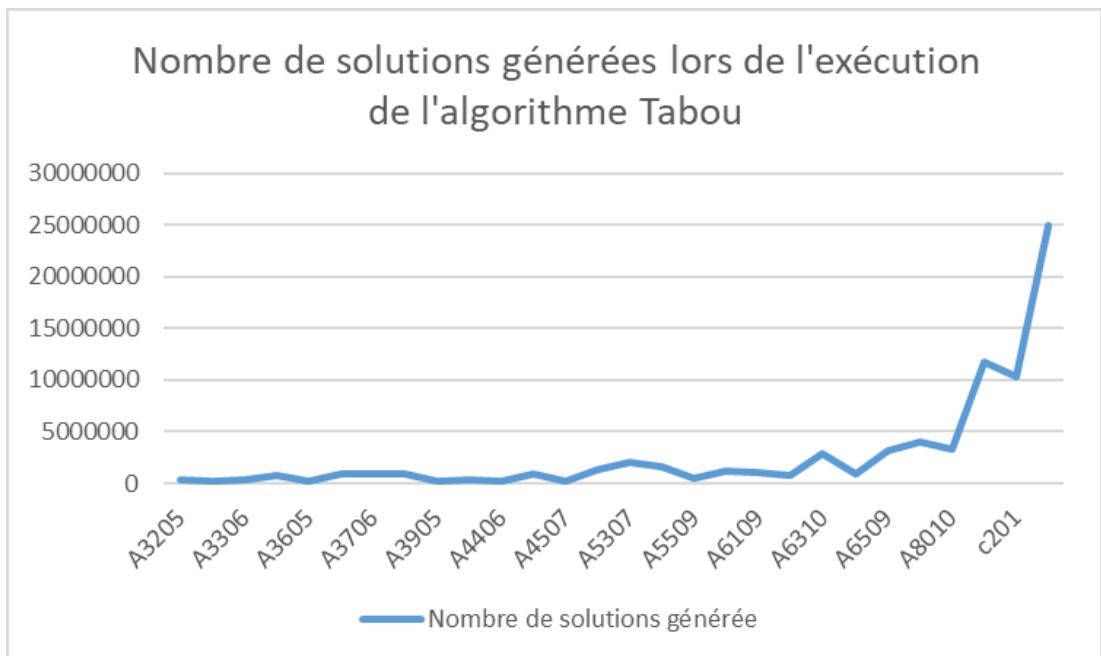


Comme nous pouvons le voir sur le graphique ci-dessus, cela à donc un impact sur le temps d'exécution.

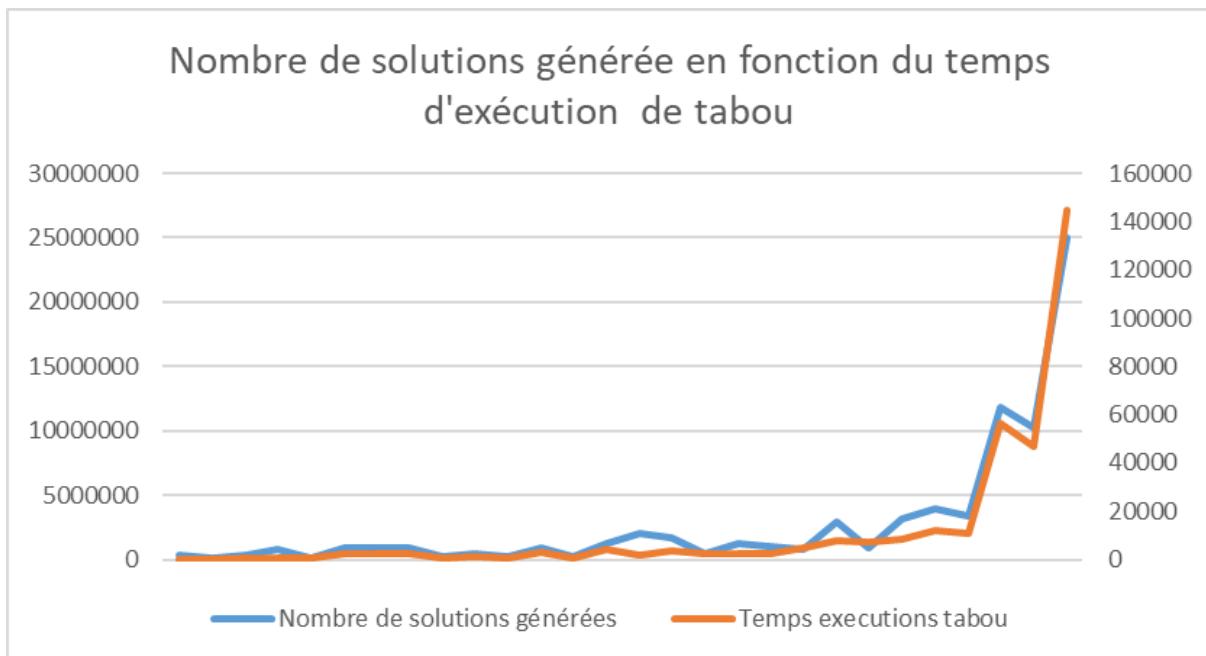
- Lorsque nous avons $\mu = 0.5$, le nombre de solution à généré étant assez faible, le *temps d'exécution est négligeable, entre 150 et 300 ms.*
- Dans le cas de $\mu = 0.9$, le *temps d'exécution augmente exponentiellement comme le nombre de solutions est plus important.*
- Le nombre de solutions à générer augmentant, les fichiers avec plus de clients sont donc les plus impactés par l'augmentation du temps d'exécution.

Solutions générées par tabou

L'algorithme de Tabou génère tous les voisins d'une carte pour trouver le meilleur candidat et réitère cela un grand nombre de fois. Le nombre de clients influe donc énormément sur le nombre de solutions générées. Plus il y a de clients, plus la carte à un nombre de voisin important.



Le graphique ci-dessus nous montre le nombre de voisins qui ont été générés avec Tabou. On peut voir qu'il augmente exponentiellement avec l'augmentation du nombre de clients.



Différence avec le nombre de véhicule minimal

Le graphique ci-dessous montre la différence pour chaque algorithme du nombre de véhicules que nos solutions contiennent. On peut voir que nous sommes généralement au bon résultat ou 1 véhicule supplémentaire.

