RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude
Présenté pour l'obtention du Diplôme de

LICENCE ACADEMIQUE

Domaine : **Sciences Exactes**
Filière : **Informatique**
Spécialité : **Systèmes Informatiques**

Thème

Implémentation de deux algorithmes heuristiques pour le problème de tournés de véhicules

Présenté par :

- **HOMMADI abd allah**
- **SOUACI mohamed amine**
- **LATRECH el hadi**

Proposé et Encadré par : M.**Abd elKamel Ben Ali**

Soutenue le 21-05- 2018 Devant le jury:

M. …… Président

M. …… Rapporteur

Année Universitaire: 2017-2018

Sommaire

1 Problème V.R.P

1.1 Introduction

Le problème de la gestion de tournées de véhicules, ou véhiculé routing problème (V.R.P) en anglais, est l'un des problèmes combinatoires difficiles les plus connus dans la communauté scientifique RO (Recherche Opérationnelle). Ce problème consiste à déterminer un ensemble optimal de tournées de distribution ou de ramassage à moyen d'une flotte de véhicules localisés sur un ou plusieurs dépôts afin de desservir un ensemble de clients répartis géographiquement en respectant des contraintes (voir figure 1.1).

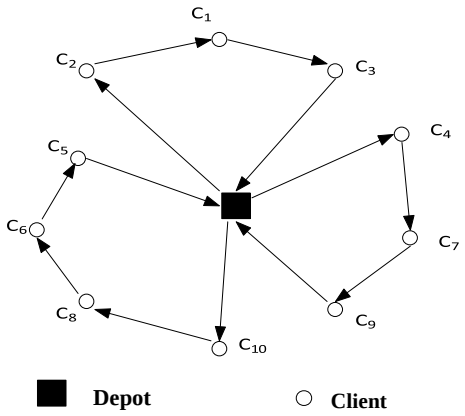

Figure 1.1 Le problème V.R.P

Nous allons présenter dans la section 1.1, les variantes du V.R.P les plus connues par rapport à leurs caractéristiques principales. Les variantes du V.R.P directement en lien avec notre problème à traiter sont expliquées plus en détail dans les sections 1.2 et 1.3. En montrant la complexité du V.R.P dans la section 1.4, nous allons exposer les méthodes de résolution pour le V.R.P ainsi que leurs applications dans la section 1.5. Pour conclure, le cadre général de notre travail sera présenté dans la section 1.6

1.2 Domaines d'applications

Le problème de tournées de véhicules (V.R.P) est appliqué dans plusieurs domaines économiques. La livraison de colis, la collecte du lait, la collecte d'argent (distributeurs), le ravitaillement de stations essences sont des exemples parmi les plus connus. Il apparaît aussi dans la distribution de services : visites à domicile de médecins, de représentants de commerce, d'agents de maintenance.

1.3 Les caractéristiques du V.R.P

Par rapport aux problèmes réels, certains auteurs ont tenté de définir une classification des problèmes de tournées à partir des caractéristiques décrites ci-dessous. Leur combinaison permet de mieux approcher les problèmes réels [Bodin et al. 1981], [Christo ides 1985], [Desrochers et al. 1990].

1.3.1 La distance entre les clients

a- ces distances sont ou ne sont pas euclidiennes .
b- la distance pour aller d'un client A à un autre client B est la même que pour aller du client B au client A (Problème Symétrique).
c- ces distances sont différentes (Problème Asymétrique) .

1.3.2 Le nombre de dépôts

a- un dépôt (V.R.P Classique).
b- plusieurs depots.

1.3.3 Le nombre de véhicules disponibles

a- un véhicule.
b- plusieurs véhicules (V.R.P Classique).

1.3.4 La capacité des véhicules

a- dentique pour tous les véhicules (homogène).
b- différente pour au moins un véhicule (hétérogène).
c- pas de restriction de capacité (capacité infinie).

1.3.5 La nature de la demande des clients

a- statique (connue en avance).
b- dynamique (apparaît au cours de temps).
c- déterministe (connue avec certitude).
d- stochastique (connue avec incertitude).

1.3.6 Les horaires de service d'un client

a- non spécifiques (V.R.P Classique).

b- spécifiques par intervalle de temps (notion de fenêtres de temps).

1.3.7 Le type de services proposés

a- un type unique.

b- plusieurs types mais un véhicule ne fournit qu'un seul type de service.

c- plusieurs types pour un même véhicule.

1.3.8 La période (ou horizon) considérée

a- une journée.

b- une semaine.

c- périodique.

1.3.9 La longueur maximal d'une tournée

a- donnée et identique pour toutes les tournées .

b- donnée et non identique pour toutes les tournées .

c- non donnée.

1.3.10 Le temps maximal d'une tournée

a- donné et identique pour toutes les tournées.

b- donné et non identique pour toutes les tournées.

c- non donné.

De même, l'objectif considéré peut différer. La minimisation de la distance totale parcourue ou la minimisation du temps parcours sont les objectifs les plus souvent considérés. Mais on peut rencontrer également les objectifs suivants :

- minimisation du nombre de véhicules utilisés pour servir l'ensemble des clients .
- minimisation de la violation de contraintes, en particulier pour les contraintes de fenêtres de temps dites « souples ».

- maximisation des gains lorsque la visite de la totalité de la clientèle est impossible, un gain étant lié à la visite d'un client.

Enfin, plus récemment, les recherches s'intéressent à des versions multi-objectifs rencontrées dans des cas pratiques [Jozefowiez et al. 2002].

1.4 Les variantes du problème

Quelques variantes classiques du problème de tournées de véhicules :

• Problème de tournées de véhicules avec contraintes de capacité : Les véhicules ont une capacité d'emport limitée (quantité, taille, poids, etc.).

• Problème de tournées de véhicules avec fenêtre de temps : Pour chaque client on impose une fenêtre de temps dans laquelle la livraison doit être exécutée.

• Problème de tournées de véhicules avec collecte et livraison: Un certain nombre de marchandises doivent être déplacées de sites de collecte vers des sites de livraisons.

1.5 Le problème C.V.R.P

1.5.1 Description

La version standard du V.R.P est le problème de tournées de véhicules avec contraintes de capacité (C.V.R.P) qui a été décrite pour la première fois en 1959 par [Dantzig et al. 1959]. Dans le C.V.R.P, tous les clients demandent le même type de service : ramassage ou livraison. Les quantités de service sont connues avec certitude. Une flotte homogène de véhicules de capacité finie, partent d'un unique dépôt, va desservir l'ensemble des clients, avant de retourner au dépôt. L'objectif est de minimiser le coût total des tournées (la distance totale parcourue ou d'autres mesures telles que le temps total) en respectant la contrainte de capacité des véhicules.

Le C.V.R.P est représenté sous la forme d'un graphe complet G = (S, A), où:

- S = {0,1,...,n} représente l'ensemble des sommets;
- A = {(i, j) tel que $i \neq j$, i, $j \in S$} représente l'ensemble des arcs entre les sommets.

Chaque sommet est associé à une position de coordonnées. Le sommet 0 représente le dépôt et les autres sommets représentent l'ensemble des clients, noté C = {1,...,n}, qui désirent obtenir une livraison de marchandise provenant du dépôt. Donc on a $S = C \cup \{0\}$. Une demande positive de marchandise d_i est associée à chaque client i appartenant à C.

Chaque arc (i, j) ∈A est associé à une valeur non négative c_{ij} qui correspond le coût pour aller du sommet i au sommet j. Il est courant de prendre la distance Euclidienne de la position du sommet i à la position du sommet j pour déterminer le coût de l'arc. Nous supposerons dans la suite que le graphe G est non orienté. Donc la matrice des coûts est symétrique, $c_{ij} = c_{ji}$. Elle satisfait l'inéquation triangulaire suivante: $c_{ij} \leq c_{ik} + c_{kj}$,∀i, j,k ∈S.

Une flotte de véhicules V = {1,...,m} est stationnée au dépôt pour desservir les clients. Tous les véhicules possèdent la même capacité Q (flotte homogène) qui est supérieure à la plus grande demande des clients, $Q \geq \max d_i$,∀i∈S.

L'objectif du problème est de minimiser le coût total des tournées tout en respectant les contraintes suivantes :

- Chaque client ne peut être servi qu'une et une seule fois par un seul véhicule.
- Toutes les tournées des véhicules commencent et se terminent par le dépôt.
- La demande totale des clients à servir de chaque tournée ne doit pas dépasser la capacité Q du véhicule.

1.5.2 Formulation

Il existe de nombreux modèles pour le V.R.P dans la littérature et nous pouvons trouver plusieurs méthodes de résolution pour un même modèle. Inversement, une même méthode de résolution peut être appliquée à plusieurs modèles. Toth et Vigo classent les différents modèles en trois catégories [Toth et al. 2002] :

- Les modèles de flot de commodités.
- Les modèles de partitionnement d'ensemble.
- Les modèles de flot de véhicules :

Les modèles de flot de commodités permettent d'associer aux arcs une demande qui circule sur ceux-ci. Ils sont intéressants de par leur adaptabilité à des problèmes réels comme la distribution d'essence par exemple [Garvin et al. 1957].

Dans les modèles de partitionnement d'ensemble, un ensemble de tournées est généré, et à chaque tournée est associé un coût. Ce type de modèle est très général puisque toutes les contraintes sur la constitution des tournées n'apparaissent pas explicitement. La difficulté principale est la génération de l'ensemble des tournées. Dès lors, suivant que l'on détermine un

sous-ensemble de tournées a priori ou que l'on génère les tournées au cours de la résolution, cela donne naissance à des heuristiques ou à des méthodes exactes parmi les plus performantes notamment pour le V.R.P.

Nous présentons ici le modèle de flot de véhicules à trois indices qui a été développé par [Fisher et al. 1978] pour le C.V.R.P, en supposant que la taille de la flotte est fixée à m. Tout d'abord, nous listons les indices et les constantes, puis nous définissons les variables de décision comme suit :

Indices et constants connues:

- i, j ∈ S = {0,1,...,n} : le dépôt ou les clients .
- v ∈V = {1,...,m} : les véhicules.
- d_i : la demande du client i, i∈C .
- Q : la capacité de chaque véhicule v ∈V.
- c_{ij} : le coût (la distance) du sommet i au sommet j, i, j ∈ S

Variables de decision :

- x_{ij}^v : la variable binaire, égale 1 si le véhicule v visite le sommet j après le sommet i.

La formulation mathématique du C.V.R.P s'écrit :

$$\text{Minimiser} \sum_{v\in V}\sum_{i\in S}\sum_{j\in S} c_{ij}x_{ij}^v \tag{1.1}$$

Sujette aux:

$$\sum_{v\in V}\sum_{j\in S} x_{ij}^v = 1, \forall i \in C \tag{1.2}$$

$$\sum_{j\in S} x_{ij}^v - \sum_{j\in S} x_{ji}^v = 0, \forall i \in C, \forall v \in V \tag{1.3}$$

$$\sum_{j\in C} x_{0j}^v = 1, \forall v \in V \tag{1.4}$$

$$\sum_{j\in C} x_{j0}^v = 1, \forall v \in V \tag{1.5}$$

$$\sum_{i\in C}\sum_{j\in S} d_i x_{ij}^v \le Q, \forall v \in V \tag{1.6}$$

$$x_{ij}^v \in \{0,1\}, \forall i, j \in S, v \in V \tag{1.7}$$

- Dans cette formulation, 1.1 représente l'objectif de minimiser la somme des coûts de parcours.
- Les équations 1.2-1.7 constituent les contraintes à satisfaire.
- L'équation 1.2 impose que chaque client soit desservi une et une seule fois par un seul véhicule.
- L'équation 1.3 assure que le véhicule qui arrive chez un client est le même que celui qui part de ce client.
- L'équation 1.4 assure que chaque véhicule ne sort qu'une seule fois du dépôt.
- L'équation 1.5 assure le retour unique au dépôt pour chaque véhicule.
- L'inéquation 1.6 définit la contrainte de capacité qui impose que la demande totale des clients de chaque tournée ne doit pas dépasser la capacité Q du véhicule.
- 1.7 est la définition des types de variables utilisés (i.e. contraintes d'intégrité).

Ce type de formulation a souvent été utilisé comme modèle mathématique pour de nouvelles variantes du V.R.P. En effet, ayant un indice par véhicule, on peut aisément spécifier l'ensemble des contraintes du problème et il est en général facile d'y ajouter de nouvelles contraintes. Une formulation du V.R.PTW, qui en découle en introduisant de nouvelles variables et en ajoutant des contraintes est présentée dans la section suivante.

2 L'heuristique de Clarke et Wright (1964)

Cette heuristique fonctionne de manière itérative en fusionnant à chaque itération deux routes en une seule ce qui procure une économie de distance. L'heuristique débute par une solution initiale où chacun des n clients est visité individuellement sur une route aller-retour. À chaque étape, les deux routes procurant la plus grande économie sont fusionnées ensemble. La fusion de la route (0, …, i, 0) avec la route (0, j, …, 0) en une plus grande route (0, …, i, j, …, 0) procure le gain suivant : c0i + c0j - cij. On dénotera ainsi sij = c0i + c0j - cij comme étant l'économie réalisée en fusionnant une route débutant (ou se terminant) avec le client i avec une route débutant (ou se terminant) avec le client j. Évidemment, une économie est réalisable seulement si la route résultante respecte les contraintes de capacité et de durée.

L'heuristique débute donc par calculer toutes les économies si possibles pour ensuite les trier en ordre décroissant. L'algorithme considère alors la plus grande économie disponible et regarde si il est possible de fusionner les routes correspondantes tout en respectant les contraintes. L'algorithme

considère alors l'économie suivante. La solution est obtenue lorsqu'il n'est plus possible de fusionner ensemble des routes ou lorsque la liste des économies disponibles est épuisée.

Cette description correspond à la version parallèle de l'heuristique puisque plusieurs routes peuvent être construites en parallèle. Il est possible d'implanter l'algorithme des économies de façon séquentielle en restreignant le choix des économies de façon à construire une seule route à la fois. Lorsque cette dernière ne peut plus être agrandie à cause des contraintes, une nouvelle route est créée.

Dans cette section nous présentons une méthode de construction, CW^+, qui utilise une version perturbée du calcul des économies. Cette méthode est à son tour utilisée à l'intérieur d'une approche d'amélioration, $D–CW^+$, qui exploite plusieurs décompositions du problème à résoudre. Ces méthodes sont présentées ci-dessous.

(i) phase d'initialisation
Initialiser la liste des économies $L \leftarrow \{S_{ij} \mid \forall (i,j) \in E\}$;
Trier L dans l'ordre décroissant ;
Initialiser l'ensemble des $|V\backslash\{0\}|$ tournées $T \leftarrow \{T_i \mid \forall i \in V\backslash\{0\}\}$;

(ii) phase d'amélioration itérative
$S_{ij} \leftarrow$ première élément dans L ;
Tant que (L non vide) **et** ($S_{ij} > 0$) **faire**
| **Si** la tournée T_i **et** la tournée T_j peuvent être fusionnées **alors**
| | $T \leftarrow T \setminus \{T_i, T_j\}$;
| | $T \leftarrow T$ U Fusion(T_i, T_j) ;
| **Fin si**
| $L \leftarrow L \setminus \{S_{ij}\}$;
| $S_{ij} \leftarrow$ premier élément dans L ;
Fin Tant que
Retourner T ;

Algorithme 1.2 : Principe de l'algorithme de Clarke et Wright

La Figure 1.14 donne un exemple de cette méthode. Le graphe et la solution initiale sont présentés Figure 1.14 (a). On considère que la capacité des véhicules Q est égale à 10. Le chiffre sur les arcs pointillés reliant 2 sommets i et **j** représente le coût Cij de déplacement du véhicule entre **i** et **j**. Le chiffre indiqué près d'un sommet **i** est la quantité **D** demandé par le client. La solution obtenue est illustrée dans la figure à droite. Dans l'itération (2), la fusion liée à $\mathbf{S_{34}}$ est rejetée à cause de la capacité de véhicule. La solution présentée cette figure contient deux tournées : (0, 4, 5, 0) et (0, 1, 2, 3, 0).

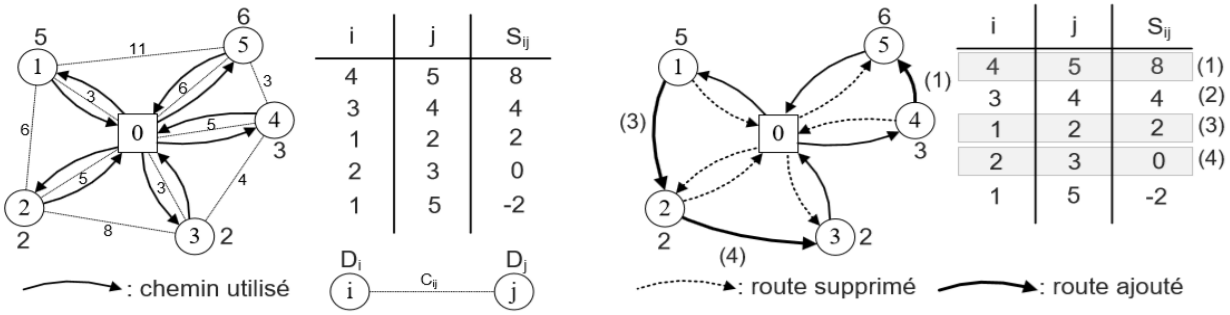

Figure 2.1 Illustration de la méthode de Clarke et Wright

3 L'heuristique Nearest Neighbor

Une façon de calculer une approximation du voyageur du commerce est d'appliquer l'heuristique du « plus proche voisin » qui fonctionne comme suit :

1. On sélectionne une ville au hasard.

2. Trouver une ville plus proche et non visitée et y aller.

3. Répéter l'étape **2** s'il en existe des villes non encore visitées.

4. Retour à la première ville.

Le fonctionnement de l'algorithme est illustré par l'exemple suivant :

On dispose de 7 villes (a, b, c, d, e, f, g) ayant les coordonnées décrites à l'aide du tableau suivant :

ville	***x***	***y***
a	**2**	**1**
b	**5**	**2**
c	**1**	**3**
d	**4**	**3**
e	**6**	**3**
f	**2**	**4**
g	**5**	**5**

Figure 3.1 Tableau de Coordonnées des villes

Initialement tous les sommets sont blancs, on commence par le chois d'une ville de départ (a) dans l'exemple, puis on colorise ce sommet en noire. À chaque fois on colorise un sommet et on l'ajoute au tour. On continue ce processus jusqu'à la colorisation de tous les sommets existants. Au moment de chaque passage d'un sommet à un autre on calcule le poids de cet arc on utilisant la formule suivant : $\sqrt{(x2\text{-}x1)2x(y2\text{-}y1)2}$

Les schémas montrent l'application de la méthode Nearest Neighbor

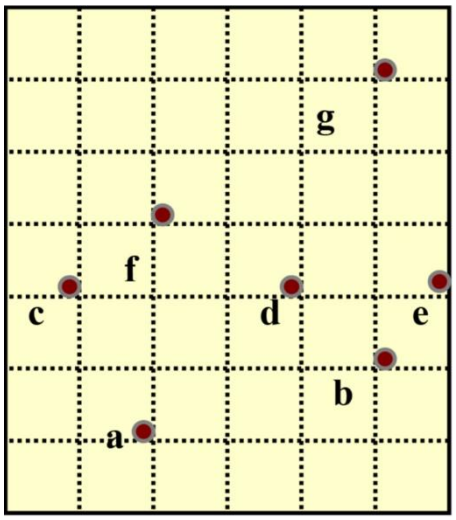

Figure 3.2 a est la ville de départ

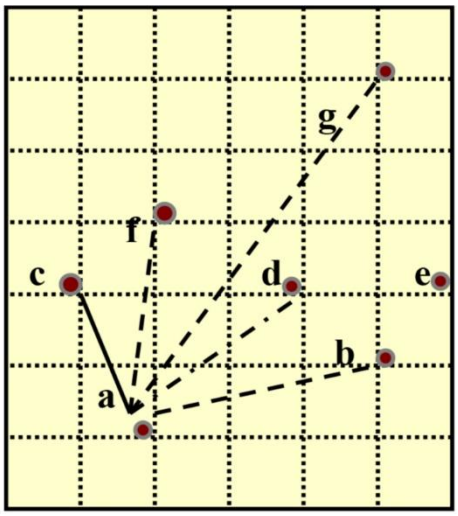

Figure 3.3 Après le choix de la ville c

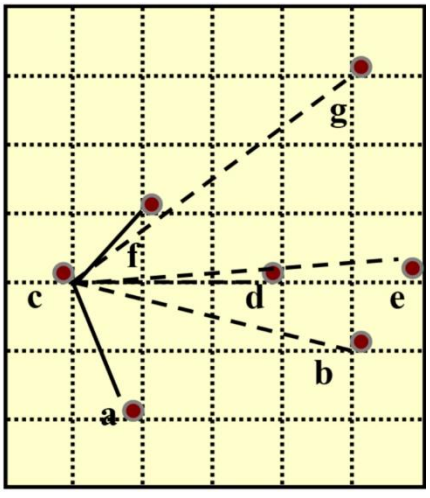

Figure 3.4 Après le choix de la ville f

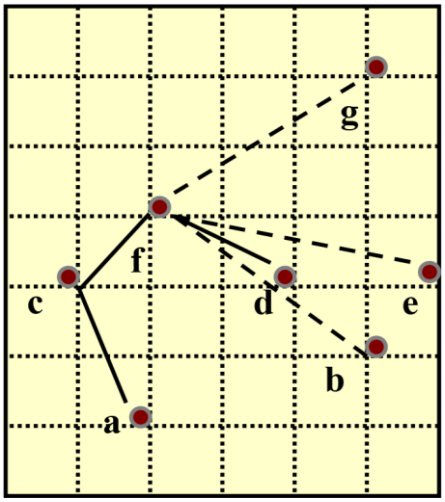

Figure 3.5 Après le choix de la ville d

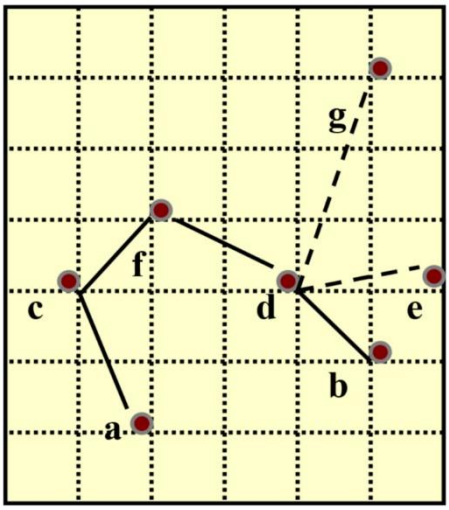

Figure 3.6 Après le choix de la ville b

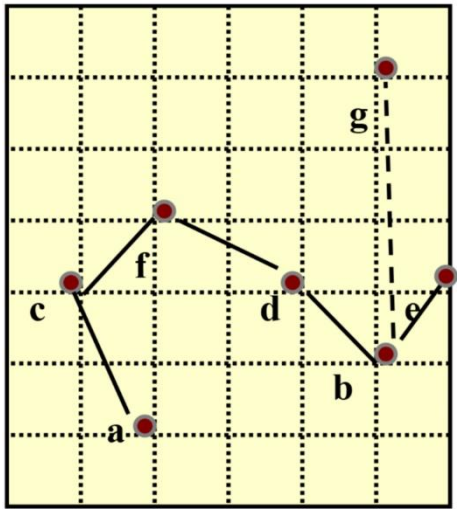

Figure 3.7 Après le choix de la ville e

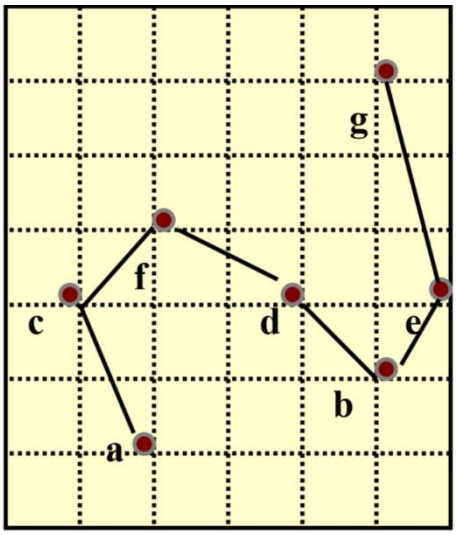

Figure 3.8 Après le choix de la ville g

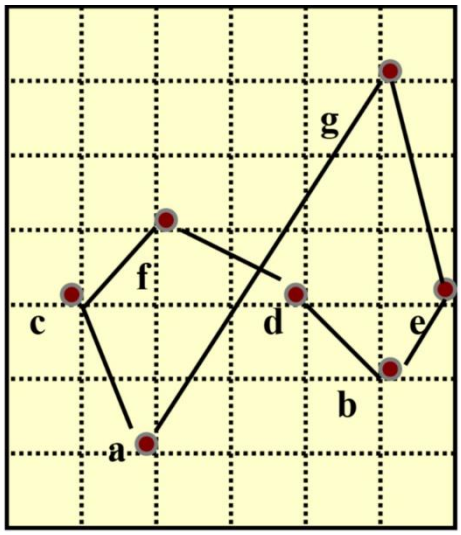

Figure 3.9 le tour après le choix de la ville a

Les figures (3.2 à 3.9) : Application de l'algorithme du plus proche voisin

La longueur de la tour obtenue par l'application de Nearest Neighbor est 15.95 Cette algorithme est basé sur le principe de choix « semble le meilleure pour l'instant.

4 Implémentation

Pour traiter le problème C.V.R.P, nous avons développé un simulateur ayant la capacité de simuler une journée de service. Le simulateur s'appuie sur deux Processus. L'Un de ce Processus simuler l'Algorithme Clarke and Wright et l'autre Proc simuler l'Algorithme Nearest Neighbors.

Le but de cette partie est de présenter quelques aspects visuels de notre application. Les différentes fonctionnalités et les principaux objets constituant les interfaces.

4.1 Environnement de travail

4.1.1 Environnement matériel

Pour implémenter notre application, nous avons utilisé un PC dont leurs configurations sont les suivantes :

- Systèmes d'exploitation Microsoft Windows 10.

- Processeurs Intel(R) I5 2.20 GHz.

- Mémoires : 8 GO RAM.

- Disque dur : 500 GO.

4.1.2 Environnement logiciel

Le langage de programmation utilisée : JAVA 1.8.0 client VM 25.121-b13.

I.D.E.: Neat Beans IDE 8.2.

4.2 Classes

4.2.1 Classe Problème

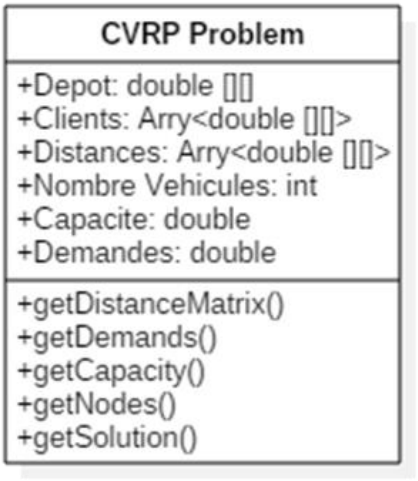

Figure 4.1 Diagramme de classe du Problème C.V.R.P

- Depot (double): c'est le node localisé de vecteur (x,y) represente le point de depart.
- Clients (list double): c'est le node localisé de vecteur (x,y) represente les points.
- Distances (list double): c'est la distance entre deux nodes.
- Nombre de Vehicule (int): c'est un flot de vehicule (Route).
- Capacite (double): le maximale des vehicules.
- Demandes(double): le quantité de service demande de chaque client.
- getDistanceMatrice: une fonction qui donne toutes les distances entre les nodes.
- getDemands:une function qui donne les demands des clients.
- getCapacity: une function qui donne les capacite des vehicules.
- getNodes: une function qui represente les points de clients.
- getSolution: c'est un objet qui calculi la solution.

4.2.2 Classe Clarke et Wright

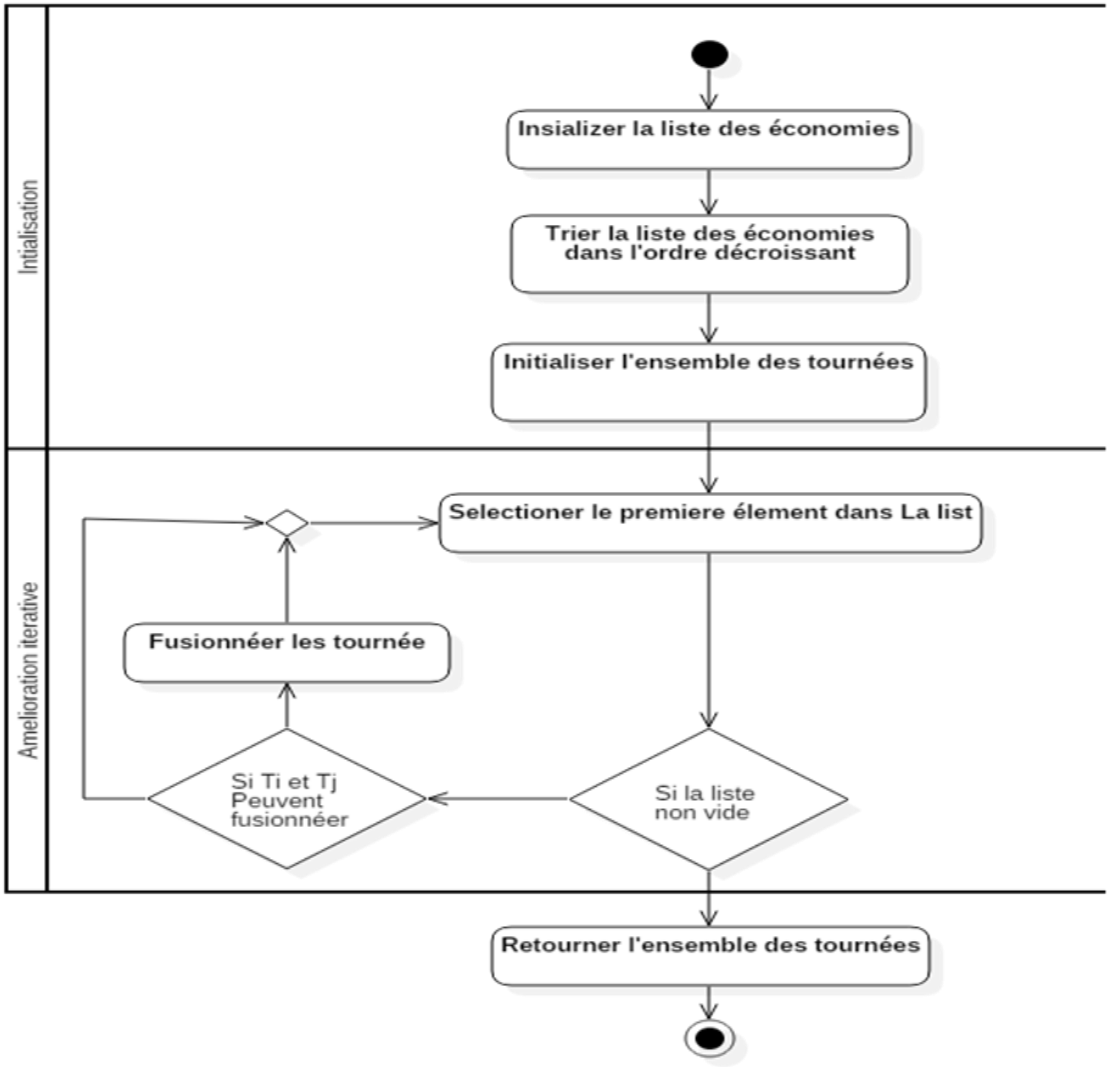

Figure 4.2 Diagramme d'activité du Problème de Clarke et Wright

4.2.3 Classe Nearest Neighbor

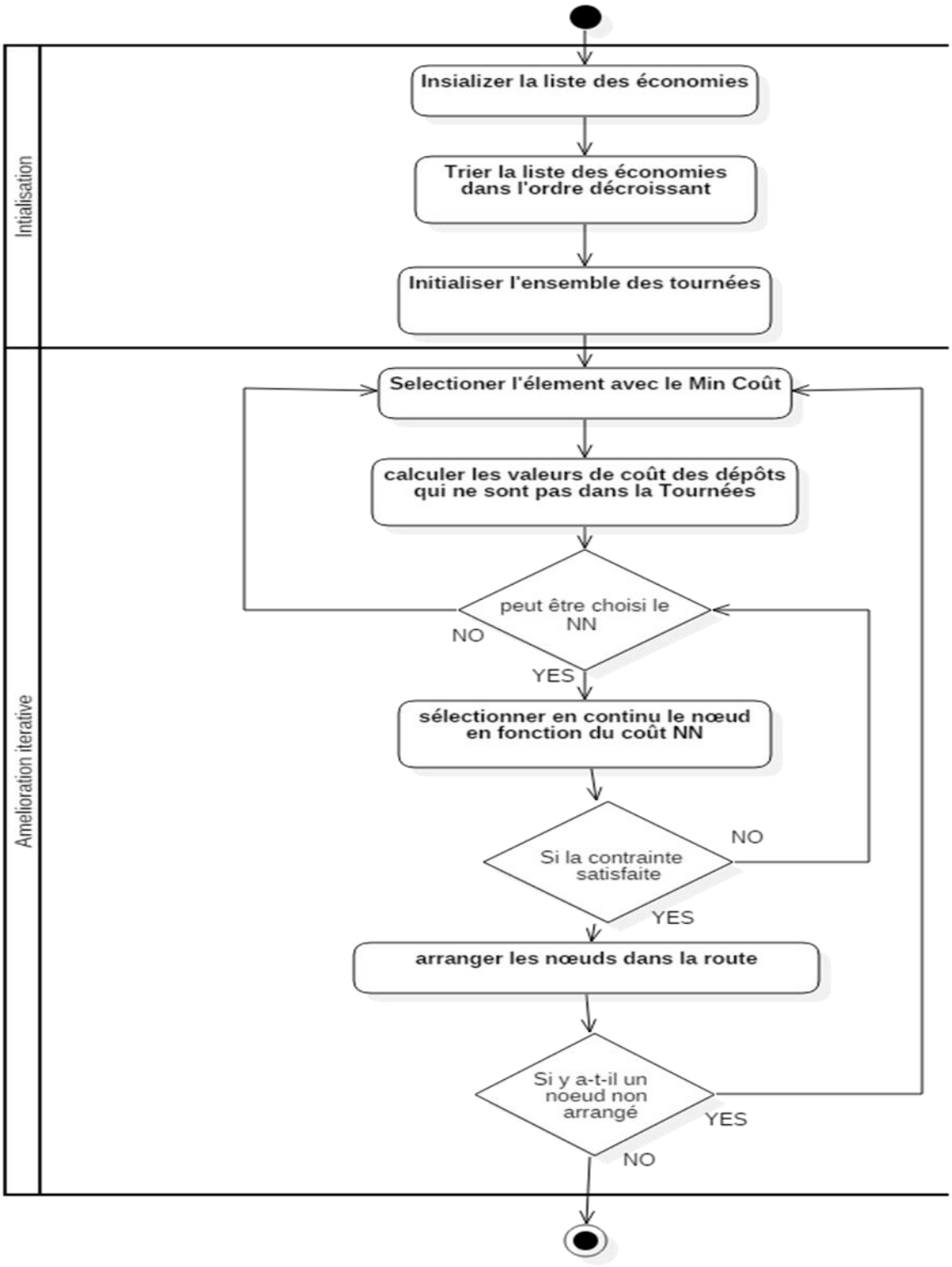

Figure 4.3 Diagramme d'activité du Problème de Nearest Neighbor

4.3 Page d’accueil

La Figure 4.1présente l'interface graphique de l’application, que ce compose de fonctionnalité suivants.

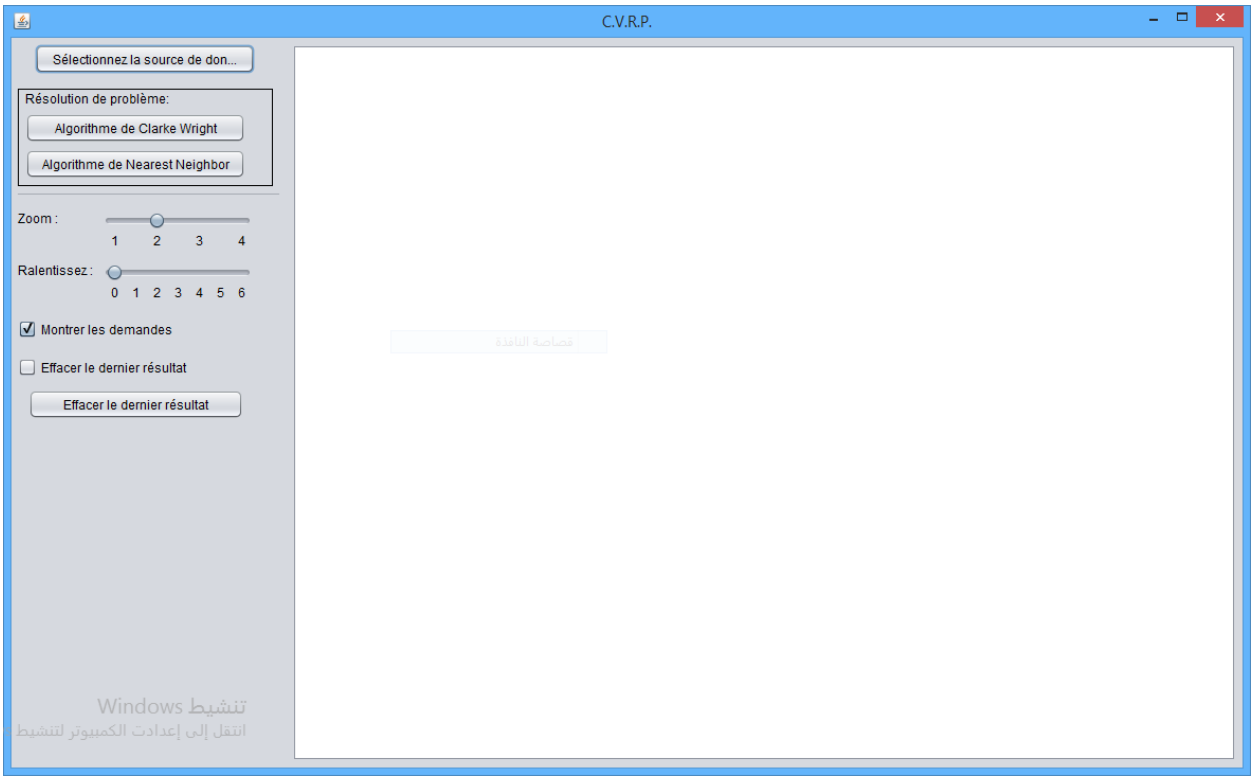

Figure 4.4 Parties d'interface graphique de l’application

- Botton pour la sélectionnèment de la source de données.

- Botton de résolution de probleme avbec l’algorithm clarke and wright.

- Botton de résolution de probleme avbec l’algorithm Nearest neighbor.

- Une barre de progression pour la fonction de zoom.

- Une barre de progression pour la vites d’exécution.

- Case à coucher pour voire ou montrer les demandes.

- Case à coucher pour efface Automatiquement le résultat dans le console.

- Botton pour efface le résultat dans le console.

- Un console d’affichage.

4.4 Page de Présentation graphique

La Figure 4.5 présente l'interface de Présentation graphique.

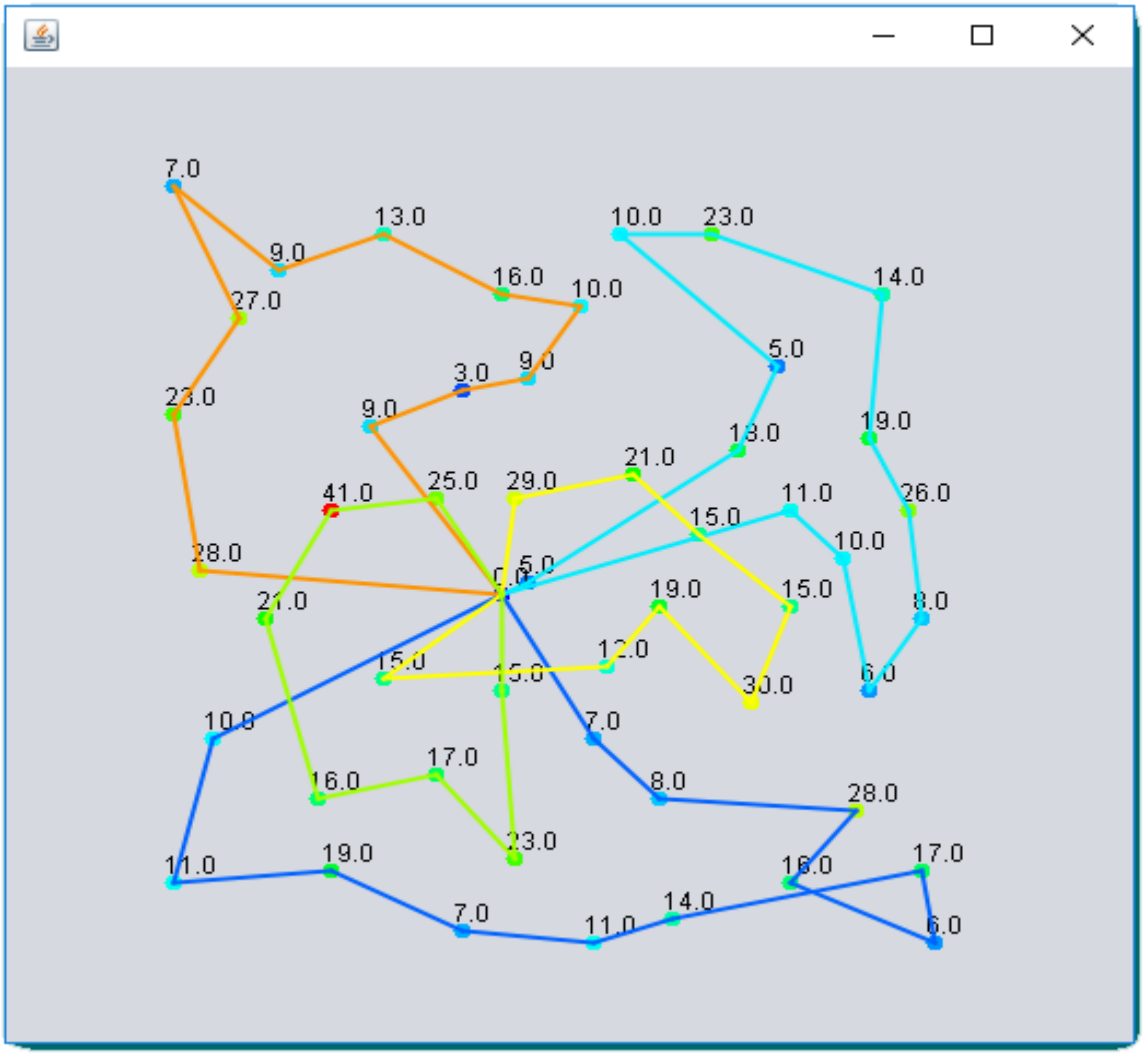

Figure 4.5 l'interface de Présentation graphique

4.5 Simulation

Le but de cette section est de simuler les deux algorithmes Clarke et Wright et Nearest Neighbor pour vérifier la performance de chaque algorithme et le fonctionnalité de notre application de simulation.

4.5.1 Data

Le fichier d'entrée que nous voulons tester dans la plate-forme, est un fichier XML, et nous choisire le type de fichier XML pour que est'e orienté pour la saisie humaine de texte structuré, et autorise beaucoup de raccourcis, XML s'est imposé comme format de référence pour l'échange de données, notamment de métadonnées.

4.5.2 Structure de donnée

Notre fichier de donnée XML et structuré come ce de suite :

- Instance : le node principale.
- Network : présente une liste de nôdes (Client avec leurs localisation x y).
- Node : .
- Fleet : Présente une liste de véhicules.
- vehicle_profile : définir les paramètre véhicule.
- departure_node: le node de départ.
- arrival_node: le node d'arrive.
- Capacity : la capacité maximale de véhicule.
- Requests : Présente la liste de requetés.
- Request : définir les paramètre de requête de chaque client.
- Quantity : la quantité demandé par chaque client.

```
<instance>
<network>
<nodes>
<node id="1" type="1">
<cx>37.0</cx>
<cy>52.0</cy>
</node>
</nodes>
</network>
<fleet>
<vehicle_profile type="0">
<departure_node>51</departure_node>
<arrival_node>51</arrival_node>
<capacity>160.0</capacity>
</vehicle_profile>
</fleet>
<requests>
<request id="0" node="1">
<quantity>7.0</quantity>
</request>
</requests>
</instance>
```

Figure 4.6 Les données d'une instance affiché dans un éditeur de données XML

4.6 Exécution

4.6.1 Exemple Instance 01 résolu avec l'algorithme Clarke et Wright

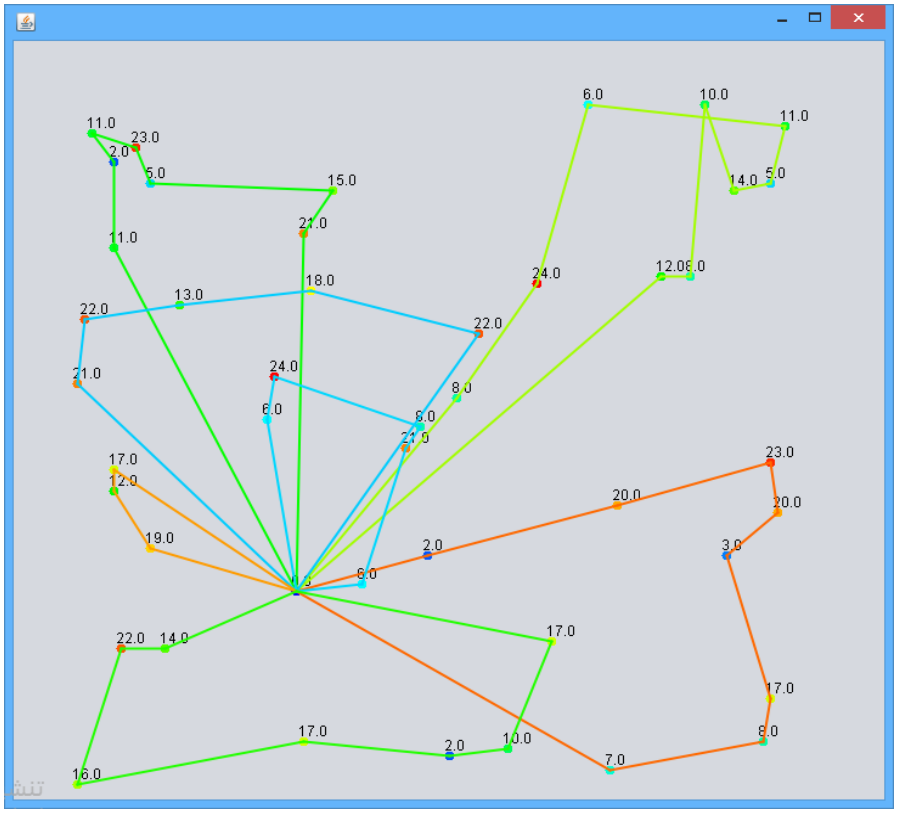

Figure 4.7 Présentation graphique de l'instance 1 avec l'algorithme Clarke et Wright

Le Résultat après l'exécution d'une instance

Route: Cost : 228.05 Load : 98.0 3: Qnt: 12.0 10: Qnt: 8.0 21: Qnt: 10.0 8: Qnt: 14.0 41: Qnt: 5.0 33: Qnt: 11.0 5: Qnt: 6.0 20: Qnt: 24.0 32: Qnt: 8.0 X 97.0, Y 62.0 X 11.0, Y 16.0X 75.0, Y 61.0X 49.0, Y 68.0X 90.0, Y 68.0X 6.0, Y 25.0 X 6.0, Y 56.0 X 87.0, Y 5.0 X 98.0, Y 8.0
Route: Cost : 165.07 Load : 88.0 18: Qnt: 21.0 17: Qnt: 15.0 11: Qnt: 5.0 40: Qnt: 23.0 30: Qnt: 11.0 19: Qnt: 2.0 34: Qnt: 11.0 X 6.0, Y 13.0 X 32.0, Y 23.0X 74.0, Y 98.0X 96.0, Y 16.0X 1.0, Y 100.0X 64.0, Y 30.0X 7.0, Y 81.0

Route: Cost : 181.82 Load : 100.0 12: Qnt: 7.0 39: Qnt: 8.0 36: Qnt: 17.0 42: Qnt: 3.0 4: Qnt: 20.0 16: Qnt: 23.0 22: Qnt: 20.0 9: Qnt: 2.0 X 56.0, Y 37.0 X 9.0, Y 11.0 X 2.0, Y 35.0 X 33.0, Y 31.0 X 71.0, Y 5.0 X 36.0, Y 17.0 X 40.0, Y 72.0 X 85.0, Y 29.0
Route: Cost : 149.64 Load : 96.0 24: Qnt: 21.0 37: Qnt: 22.0 27: Qnt: 13.0 43: Qnt: 18.0 13: Qnt: 22.0 X 60.0, Y 95.0 X 32.0, Y 94.0 X 46.0, Y 53.0 X 6.0, Y 59.0 X 13.0, Y 81.0
Route: Cost : 84.19 Load : 65.0 26: Qnt: 6.0 29: Qnt: 24.0 7: Qnt: 8.0 28: Qnt: 21.0 23: Qnt: 6.0 X 15.0, Y 33.0 X 3.0, Y 9.0 X 91.0, Y 17.0 X 28.0, Y 43.0 X 1.0, Y 44.0
Route: Cost : 63.55 Load : 48.0 1: Qnt: 19.0 44: Qnt: 12.0 6: Qnt: 17.0 X 52.0, Y 96.0 X 31.0, Y 73.0 X 48.0, Y 50.0
Route: Cost : 157.21 Load : 98.0 14: Qnt: 14.0 35: Qnt: 22.0 31: Qnt: 16.0 38: Qnt: 17.0 2: Qnt: 2.0 25: Qnt: 10.0 15: Qnt: 17.0 X 66.0, Y 80.0 X 96.0, Y 88.0 X 53.0, Y 46.0 X 95.0, Y 94.0 X 81.0, Y 29.0 X 27.0, Y 49.0 X 96.0, Y 55.0
Total Cost : 1029.530861672903 , Total Load : 593.0 Valeur Optimal:944.0 Time : 17.866316

4.6.2 Exemple Instance 01 résolu avec l'algorithme Nearest Neighbor

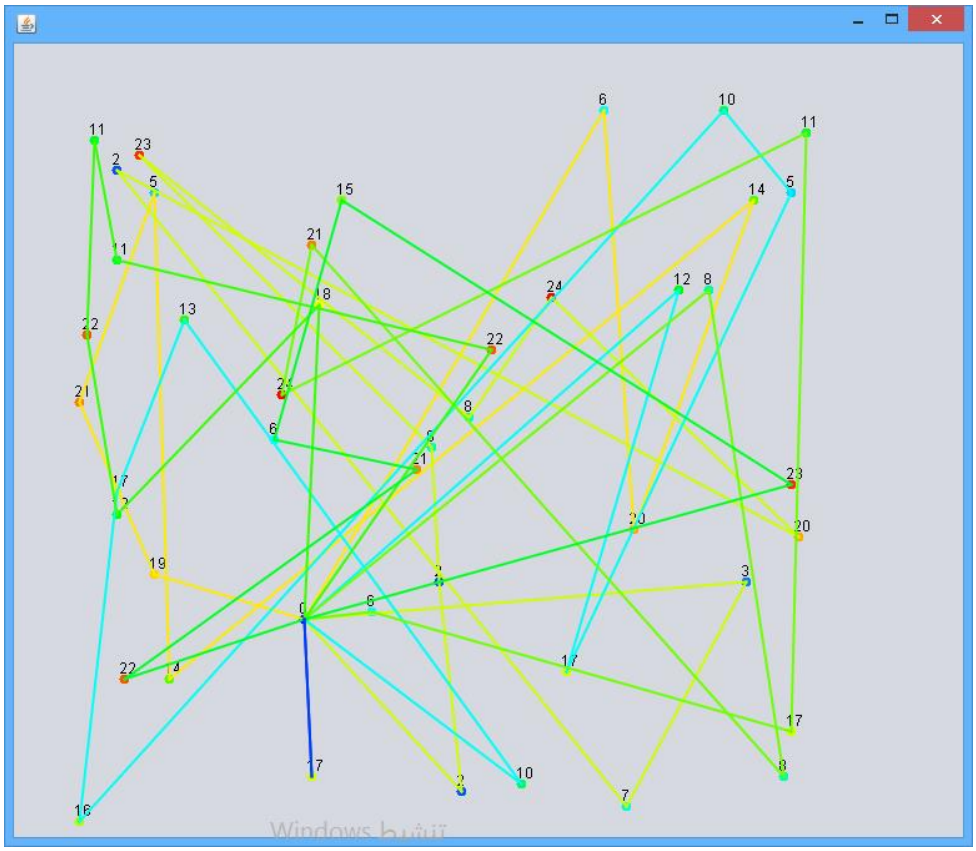

Figure 4.8 Présentation graphique de l'instance 2 avec l'algorithme Nearest Neighbor

Le Résultat après l'exécution d'une instance

Route: Cost : 111.00 Load : 99 1: Qnt: 19 24: Qnt: 21 11: Qnt: 5 14: Qnt: 14 8: Qnt: 14 22: Qnt: 20 5: Qnt: 6 X 11, Y 67 X 1, Y 44 X 11, Y 16 X 13, Y 81 X 91, Y 17 X 75, Y 61 X 71, Y 5
Route: Cost : 118.00 Load : 99 2: Qnt: 2 9: Qnt: 2 7: Qnt: 8 40: Qnt: 23 32: Qnt: 8 20: Qnt: 24 4: Qnt: 20 19: Qnt: 2 12: Qnt: 7 42: Qnt: 3 X 52, Y 96 X 49, Y 68 X 48, Y 50 X 9, Y 11 X 53, Y 46 X 64, Y 30 X 97, Y 62 X 6, Y 13 X 74, Y 98 X 90, Y 68
Route: Cost : 94.00 Load : 100 3: Qnt: 12 15: Qnt: 17 41: Qnt: 5 21: Qnt: 10 31: Qnt: 16 6: Qnt: 17 27: Qnt: 13 25: Qnt: 10 X 81, Y 29 X 66, Y 80 X 96, Y 16 X 87, Y 5 X 1, Y 100 X 6, Y 56 X 15, Y 33 X 60, Y 95
Route: Cost : 50.00 Load : 95 10: Qnt: 8 39: Qnt: 8 18: Qnt: 21 29: Qnt: 24 33: Qnt: 11 36: Qnt: 17 23: Qnt: 6 X 85, Y 29 X 95, Y 94 X 32, Y 23 X 28, Y 43 X 98, Y 8 X 96, Y 88 X 40, Y 72

Route: Cost : 108.00 Load : 96					
13: Qnt: 22	34: Qnt: 11	30: Qnt: 11	37: Qnt: 22	44: Qnt: 12	43: Qnt: 18
X 56, Y 37	X 6, Y 25	X 3, Y 9	X 2, Y 35	X 6, Y 59	X 33, Y 31
Route: Cost : 125.00 Load : 87					
16: Qnt: 23	17: Qnt: 15	26: Qnt: 6	28: Qnt: 21	35: Qnt: 22	
X 96, Y 55	X 36, Y 17	X 27, Y 49	X 46, Y 53	X 7, Y 81	
Route: Cost : 0.00 Load : 17					
38: Qnt: 17					
X 32, Y 94					
Total Cost : 606.0 , Total Load : 593.0 Valeur Optimal:944.0 Time : 0.853287					

4.6.3 Exemple Instance 02 résolu avec l'algorithme Clarke et Wright

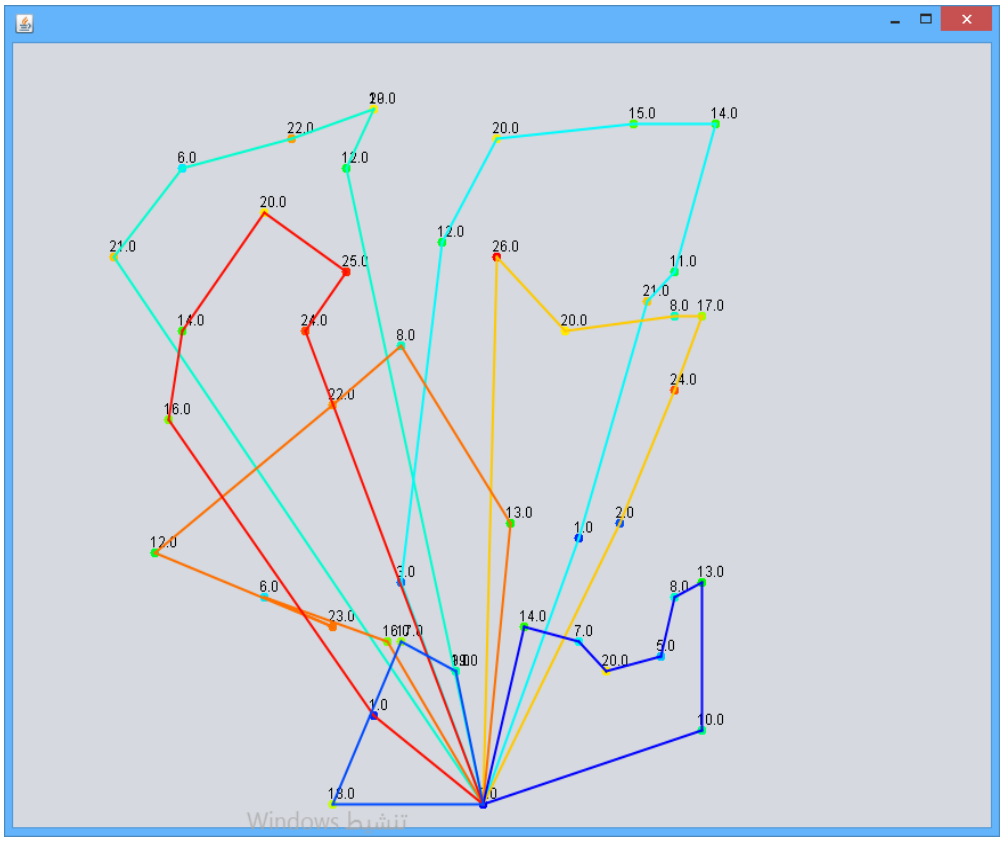

Figure 4.9 Présentation graphique de l'instance 2 avec l'algorithme Clarke et Wright

Le Résultat après l'exécution d'une instance

Route: Cost : 233.61 Load : 100.0 31: Qnt: 12.0 36: Qnt: 20.0 19: Qnt: 19.0 29: Qnt: 22.0 41: Qnt: 6.0 43: Qnt: 21.0 X 67.0, Y 75.0 X 83.0, Y 7.0 X 93.0, Y 69.0X 63.0, Y 9.0 X 89.0, Y 27.0X 35.0, Y 35.0
Route: Cost : 225.61 Load : 97.0 8: Qnt: 1.0 15: Qnt: 21.0 42: Qnt: 11.0 1: Qnt: 14.0 37: Qnt: 15.0 30: Qnt: 20.0 5: Qnt: 12.0 21: Qnt: 3.0 X 65.0, Y 61.0 X 89.0, Y 71.0X 7.0, Y 25.0 X 45.0, Y 87.0X 81.0, Y 61.0X 41.0, Y 13.0X 29.0, Y 71.0 X 63.0, Y 25.0 Route: Cost : 181.70 Load : 97.0 38: Qnt: 2.0 17: Qnt: 24.0 23: Qnt: 17.0 25: Qnt: 8.0 10: Qnt: 20.0 22: Qnt: 26.0 X 57.0, Y 81.0 X 79.0, Y 81.0X 39.0, Y 45.0X 47.0, Y 77.0X 17.0, Y 35.0X 93.0, Y 33.0
Route: Cost : 196.34 Load : 100.0 9: Qnt: 13.0 14: Qnt: 8.0 24: Qnt: 22.0 28: Qnt: 12.0 4: Qnt: 23.0 6: Qnt: 6.0 26: Qnt: 16.0 X 73.0, Y 35.0 X 85.0, Y 31.0X 89.0, Y 33.0X 33.0, Y 9.0 X 55.0, Y 23.0X 87.0, Y 79.0X 29.0, Y 19.0
Route: Cost : 113.13 Load : 77.0 40: Qnt: 10.0 20: Qnt: 13.0 16: Qnt: 8.0 7: Qnt: 5.0 18: Qnt: 20.0 13: Qnt: 7.0 32: Qnt: 14.0 X 17.0, Y 13.0 X 49.0, Y 69.0X 89.0, Y 43.0X 75.0, Y 63.0X 45.0, Y 5.0 X 49.0, Y 37.0X 41.0, Y 27.0
Route: Cost : 113.13 Load : 77.0 40: Qnt: 10.0 20: Qnt: 13.0 16: Qnt: 8.0 7: Qnt: 5.0 18: Qnt: 20.0 13: Qnt: 7.0 32: Qnt: 14.0 X 17.0, Y 13.0 X 49.0, Y 69.0X 89.0, Y 43.0X 75.0, Y 63.0X 45.0, Y 5.0 X 49.0, Y 37.0X 41.0, Y 27.0
Route: Cost : 195.67 Load : 100.0 2: Qnt: 1.0 3: Qnt: 16.0 11: Qnt: 14.0 27: Qnt: 20.0 33: Qnt: 25.0 44: Qnt: 24.0 X 15.0, Y 47.0 X 39.0, Y 75.0X 39.0, Y 99.0X 13.0, Y 65.0X 49.0, Y 77.0X 61.0, Y 99.0
Route: Cost : 73.55 Load : 63.0

39: Qnt: 9.0 35: Qnt: 19.0 34: Qnt: 17.0 12: Qnt: 18.0 X 93.0, Y 89.0 X 45.0, Y 5.0 X 57.0, Y 81.0X 75.0, Y 77.0
Total Cost : 1219.6174057581636 , Total Load : 634.0 Valeur Optimal:1146.0 Time : 14.53061

4.6.4 Exemple Instance 02 résolu avec l'algorithme Nearest Neighbor

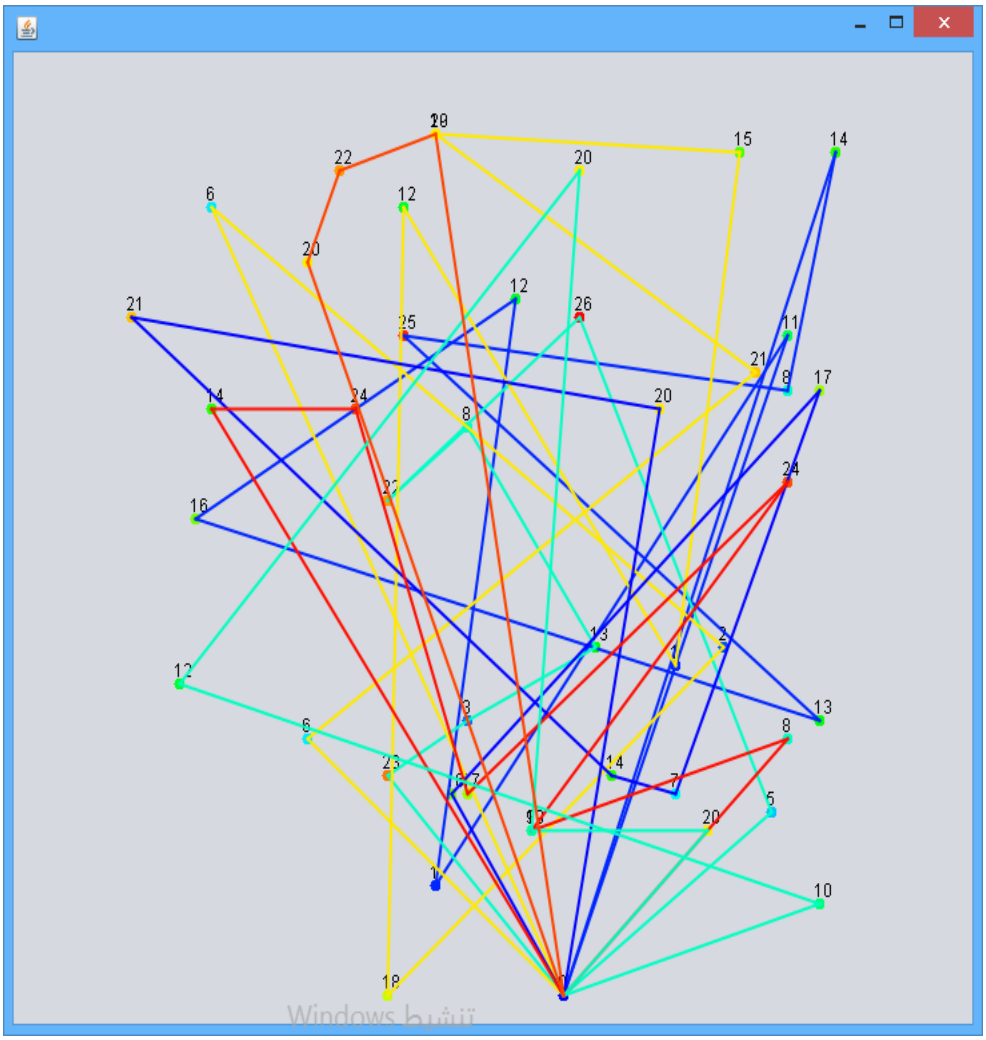

Figure 4.10 Présentation graphique de l'instance 2 avec l'algorithme Nearest Neighbor

Solution d'exécution

Route: Cost : 94.00 Load : 100						
1: Qnt: 14	25: Qnt: 8	33: Qnt: 25	20: Qnt: 13	3: Qnt: 16	5: Qnt: 12	2: Qnt: 1
42: Qnt: 11						
X 95, Y 7	X 89, Y 33	X 41, Y 27	X 93, Y 69	X 15, Y 47	X 55, Y 23	X 45, Y 87
X 89, Y 27						
Route: Cost : 78.00 Load : 100						
4: Qnt: 23	21: Qnt: 3	9: Qnt: 13	14: Qnt: 8	24: Qnt: 22	22: Qnt: 26	7: Qnt: 5
X 39, Y 75	X 49, Y 69	X 65, Y 61	X 49, Y 37	X 39, Y 45	X 63, Y 25	X 87, Y 79
Route: Cost : 145.00 Load : 100						
6: Qnt: 6	15: Qnt: 21	19: Qnt: 19	37: Qnt: 15	8: Qnt: 1	31: Qnt: 12	12: Qnt: 18
38: Qnt: 2	41: Qnt: 6					
X 29, Y 71	X 85, Y 31	X 45, Y 5	X 83, Y 7	X 75, Y 63	X 41, Y 13	X 39, Y 99
X 81, Y 61	X 17, Y 13					
Route: Cost : 82.00 Load : 95						
10: Qnt: 20	43: Qnt: 21	32: Qnt: 14	13: Qnt: 7	23: Qnt: 17	26: Qnt: 16	
X 73, Y 35	X 7, Y 25	X 67, Y 75	X 75, Y 77	X 93, Y 33	X 47, Y 77	
Route: Cost : 124.00 Load : 96						
11: Qnt: 14	44: Qnt: 24	34: Qnt: 17	17: Qnt: 24	39: Qnt: 9	16: Qnt: 8	
X 17, Y 35	X 35, Y 35	X 49, Y 77	X 89, Y 43	X 57, Y 81	X 89, Y 71	
Route: Cost : 34.00 Load : 81						
18: Qnt: 20	35: Qnt: 19	30: Qnt: 20	28: Qnt: 12	40: Qnt: 10		
X 79, Y 81	X 57, Y 81	X 63, Y 9	X 13, Y 65	X 93, Y 89		
Route: Cost : 95.00 Load : 62						
27: Qnt: 20	29: Qnt: 22	36: Qnt: 20				
X 29, Y 19	X 33, Y 9	X 45, Y 5				
Total Cost : 652.0 ,	Total Load : 634.0	Valeur Optimal:1146.0			Time : 0.280386	

4.7 Résultat des deux algorithmes

les instances	Valeur Optimale	Alg1 Clarke et Wright			Alg2 Nearest Neighbor		
		Solution	Déviation %	Le Tempes (ms)	Solution	Déviation %	Le Tempes (ms)
A-n32-k5	784	**843**	7.525510204	6.03319	924	17.85714286	0.10823
A-n33-k5	661	**722**	9.228441755	7.30625	752	13.76701967	0.27758
A-n33-k6	742	798	7.547169811	3.93986	**782**	5.39083558	0.28878
A-n37-k5	669	**737**	10.16442451	2.07933	807	20.62780269	0.29158
A-n38-k5	730	**771**	5.616438356	1.28716	820	12.32876712	0.20620
A-n39-k5	822	913	11.07055961	0.95545	**910**	10.70559611	0.19454
A-n39-k6	831	**869**	4.572803851	0.89387	1104	32.85198556	0.12129
A-n46-k7	914	**1004**	9.846827133	1.63006	1044	14.22319475	0.13109
A-n48-k7	1037	1160	11.8611379	2.22162	**1052**	1.446480231	0.12129
A-n53-k7	1001	1079	7.792207792	2.74367	**1048**	4.695304695	0.18381
A-n65-k9	1174	**1250**	6.473594549	2.14138	1434	22.14650767	0.15302
A-n69-k9	1159	1259	8.628127696	1.67251	**1163**	0.345125108	0.15348
A-n80-k10	1763	1881	6.693136699	1.40846	**1804**	2.325581395	0.15488

Comparaison

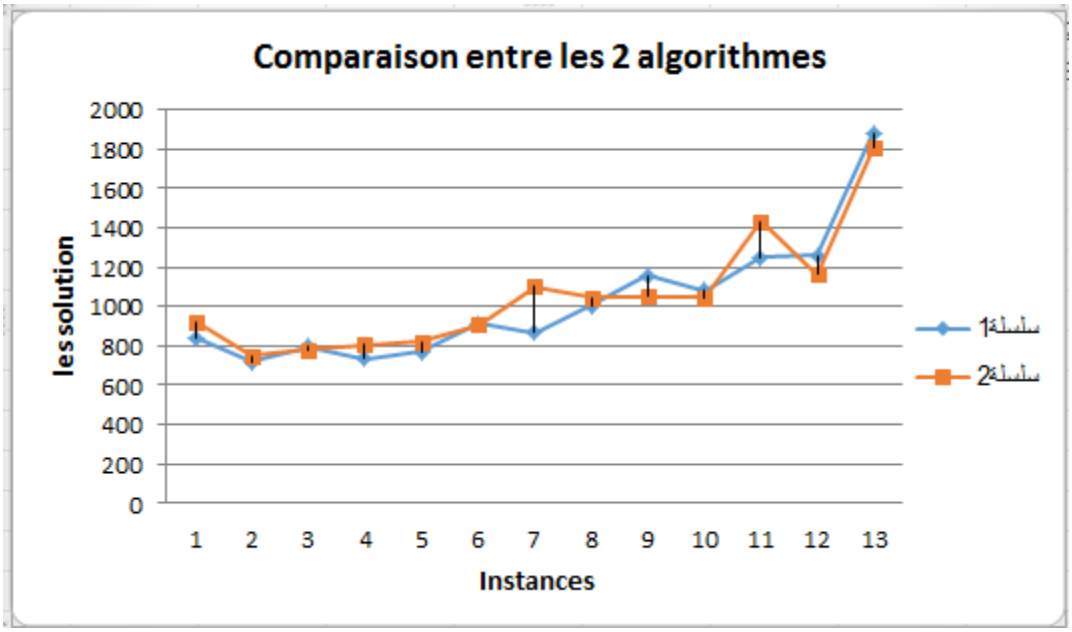

Figure 4.11 graph de Comparaison entre les deux algorithmes

Analyse de la courbe

Représente la courbe d'ascendance les résultats obtenus à partir de la table de résultats numériques tel que nous notons que l'algorithme **Clarke et Wright** meilleur de la **Nearest Neighbor** dans certain de l'instance (A-n65-k9,A-n46-k7,A-n39-k6,A-n38-k5,A-n37-k5, A-n32-k5,A-n33-k5) et dans les autres instances contraire .

Conclusion générale

La recherche examinée dans cette thèse se concentre sur la comparaison de deux algorithmes, à savoir, l'algorithme Clarke et Wright et l'algorithme Nearest Neighbor, pour résoudre le problème C.V.R.P. Ensuite, nous avons implémentation de deux algorithmes heuristiques pour obtenir des solutions de bonne qualité pour le problème, Après de nombreuses expériences, nous avons remarqué que l'algorithme Nearest Neighbor un mode de fonctionnement non sécurisé, en choisissant le plus proche voisin, indépendamment du coût. Ainsi, nous concluons que l'algorithme Clarke et Wright a les solutions les plus sûres.

Table de figures

Références

[Bodin et al. 1981] L. Bodin and B. Golden. « Classification in vehicle routing and scheduling ». Networks 11, pages 97-108, 1981.

[Christofides 1985] N. Christofides. « Vehicle Routing ». chapter 12 in Traveling Salesman Problem. pages 431- 448. John Wiley & Sons Ltd., 1985.

[Desrochers et al. 1990] M. Desrochers, J.K. Lenstra and M.W.P. Savelsbergh. « A classification scheme for vehicle routing and scheduling problems ». European Journal of Operational Research 46(3), pages 322-332,1990.

[Jozefowiez et al. 2002] N. Jozefowiez, F. Semet and E-G. Talbi. « Parallel and Hybrid Models for Multi-objective Optimization: Application to the Vehicle Routing Problem». J.J. Merelo Guervós et al. (Eds.): PPSN VII, LNCS 2439, Springer-Verlag, pages 271–280,2002.

[Dantzig et al. 1959] G.B. Dantzig and J.H. Ramser. « The truck dispatching problem ». Operations Research, Management Sciences 6(1), pages 80–91, 1959.

[Toth et al. 2002] P. Toth and D. Vigo. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[Garvin et al. 1957] W.M. Garvin, H.W. Crandall, J.B. Johnson and R.A. Spellman. « Application of linear programming to oil industry ». Management Science 3, pages 407-430, 1957.

[Fisher et al. 1978] M.L. Fisher and R. Jaikumar. « A Decomposition Algorithm for Large-Scale Vehicle Routing ». Rapport technique n_ 78-11-05, Departement of Decision Sciences, University of Pennsylvania, 1978.

[Source data] http://neo.lcc.uma.es/v.r.p/v.r.p-instances/capacitated-v.r.p-instances.