République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université SAAD DAHLAB de BLIDA
FACULTE de science

DEPARTEMENT : informatique

MEMOIRE

En vue de l'obtention du diplôme de

MASTER

Ingénierie de logicielle

Présenté par :
Rahi Amina & Cherbi Hamza

Thème

Amélioration d'un algorithme pour la création dynamique des lignes de transporte de personnel

Promoteur :
Mme. Boumahdi Fatima

Encadreur :
Mr. Chennouf Fouad

Devant le jury :
Mr.Bala Mahfoud
Mr.Derrar Hacene : président

Organisme d'accueil :
Wilab Technology

Promotion : 2016/2017

Dédicaces

A l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que dieu te garde dans son vaste paradis, à toi mon père.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur ; maman que j'adore.

Aux personnes dont j'ai bien aimé la présence dans ce jour, à tous mes frères et mes sœurs

Fatima , Mohamed Amine ,Oussama ,Abd raouf ,Kawthar

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, et qui m'ont accompagnaient durant mon chemin d'études supérieures, mes aimables amis,

Dounya, zahera, farida,razika,hadjer ,manel

et spécialement mon binôme hamera.

et a mes amis qui sont très loin mais toujours dans ma cœur Namjoon , Seokjin ,Yoongi ,Hoseok ,Jimin , Taehyung, Jungkook .

Et à tous ceux que j'aime et à toutes les personnes qui m'ont prodigué des encouragements et se sont donné la peine de me soutenir durant cette formation

Rahi Amina

Remerciement

Nous remercions tout d'abord notre dieu (Allah) pour la patience qui nous offert pour finaliser ce travail dans des bonnes conditions.

Au terme de ce travail, nous tenons à exprimer nos vifs remerciements et nos profondes reconnaissances à tous particulièrement notre promoteur madame Boumahdi Fatima , d'avoir accepté de diriger ce travail.

Nous remercions chaleureusement Mr. Chennouf Fouad qui aide durant notre travail et par sa patience et ses Précieux conseils.

Nous remercions vivement les membres du jury qui m'out fait l'honneur d'accepter du juger notre travail.

Nous remercions tous les enseignants du département d'informatique, aussi notre promotion 2016-2017 Master informatique.

Nous remercions toutes les personnes qui nous out soutenues de prés de loin au cours de la réalisation de ce modeste travail.

Résumé

Ce mémoire porte sur la modélisation et la résolution de différents problèmes de tournées de véhicules et plus particulièrement sur des problèmes de transport de personnes (dans notre cas : des employés). Ces problèmes demandent entre autre de respecter une qualité de service minimale pour les solutions proposées. Pour résoudre ces problèmes, notre système permettra de mettre en relation les personnes de différentes entreprises afin de leur permettre d'avoir un transport mutualisé. Les personnes faire inscrire sur une plateforme indiquent leur lieu d'habitation et de travail, pour obtenir des solutions de bonne qualité dans des temps raisonnables. Deux problèmes sont traités :

Le premier est de trouver les zones ne regroupant de personnes semblables, et le deuxième de relier les zones trouvées par le minimum de bus.

Mots- clé : transport à la demande, CHA, Data mining.

Abstract

In this thesis, we are interested in modeling and solving various vehicle routing problems (VRP), especially passenger transportation problems. These problems aim at finding solutions which guarantee a required quality of service.

To solve these problems, our system will make it possible to connect people of different companies in order to allow them to have a shared transport. The persons have to register on a platform indicate their place of residence and work, to obtain good quality solutions in reasonable times. two problems are addressed:
The first is to find areas grouping similar people, and the second to link the areas found by the minimum bus.

Keywords : Dial-a-Ride, HCA, Datamining

Table des matières

Table de figure

Introduction

1.1 Introduction

Lors de ces dernières années, l'offre de transport public n'a cessé de se développer et de s'améliorer, et avec elle les demandes et les exigences des usages, rendant ces moyens de transport moins compétitifs surtout dans les milieux ruraux et périurbains.
Les problèmes de transport jouent un rôle très important dans de nombreux secteurs d'activités.

On peut considérer que résoudre un problème de transport consiste à organiser dans le temps et dans l'espace un ensemble de tournées affectées à divers véhicules. Ce type de problème est rencontré, par exemple, dans les chaînes logistiques de grands groupes industriels, dans la grande distribution, dans les systèmes de transports publics (métro, bus…) ou privés (Uber, Blablacar…). Les problèmes se rencontrent aussi bien dans le cadre des services à la personne (transport de personnes à mobilité réduite, transport de personnes âgées, transport scolaire…) que dans le cadre du transport de marchandises (livraison de colis, collecte de déchets ménagers…).

Les problèmes de tournées sont des problèmes combinatoires, c'est-à-dire des problèmes pour lesquels il existe un grand nombre de solutions qu'il est, en général, impossible de parcourir en totalité. Au cours de ce mémoire, plusieurs problèmes de tournées sont traités, avec une approche non agrégatif basé sur l'amélioration des algorithmes permettent la création de lignes de bus dynamiquement Ces méthodes ne garantissent pas l'optimalité de la solution mais permettent de fournir une solution de bonne qualité avec des temps de calcul raisonnables.

1.2 Objectif et Problématique

Il y a trop des gens qui ne disposent pas de leur propre véhicule, ou tout simplement ceux qui souhaitent se rendre au travail sans leur véhicule, la solution reste de prendre le bus ou autre moyen de transport, parfois obligé de faire des changements (escales) qui peuvent faire durer le trajet plus longtemps.

Dans le cadre de ce mémoire, nous nous intéressons en particulier à trouver une solution informatique permettant le pilotage en temps réel ou différé des véhicules mis en jeux pour

résoudre le problème de TAD. En plus de l'intérêt écologique et environnemental, le système ainsi mis au point doit optimiser les solutions.

Cette optimisation nécessite la satisfaction de plusieurs critères qui peuvent êtres contradictoires. Les critères à optimiser dans le système de TAD sont respectivement : La distance parcourue, le temps de service des véhicules, le nombre de véhicules utilisés dans le service.

Notre objectif est de modéliser et de résoudre d'une manière optimale un PTD qui consiste à transporter des personnes qui sont des employés.

1.3 Plan de la mémoire

Ce mémoire se compose de quatre chapitres.

Dans le chapitre 1, nous présentons le service de Transport À la Demande (TAD). Ensuite nous rappelons les différents éléments qui composent ce problème, ainsi que les contraintes à satisfaire et les objectifs. Nous présentons également une étude bibliographique concernant les problèmes de TAD dans les cas statique et dynamique.

Dans le chapitre 2, est consacré à une présentation générale du différentes techniques et algorithmes de classification de data mining qui sont utilisés, notamment, pour la classification automatique de données (non supervisée).

Dans le chapitre 3, ce chapitre décrit la partie conceptuelle. Nous commencerons par comprendre le contexte du système, déterminer les principaux cas d'utilisation, déterminer les besoins fonctionnels et les besoins non fonctionnels. Puis en présentant l'architecteur de notre système.

Dans le chapitre 4, le dernier chapitre est consacré aux tests et résultats où les algorithmes sont testés face à différents jeux de données réels et nous allons présenter dans la réalisation de l'application Linkibus.

- Enfin, dans le dernier chapitre nous présentons une conclusion de ce mémoire.

Chapitre 1 : Etat de L'art sur Le Problème de Transport à la Demande

1 Introduction

Le Problème de Transport à la Demande (PTD) ou encore en anglais Dial a Ride Problème (DRP), C'est un problème de la famille des problèmes de tournées de véhicules qui consiste à transporter un ensemble de personnes de leur sommet d'origine vers leur sommet de destination.

La résolution d'un PTD est une solution de transport alternative permettant l'augmentation des taux d'occupation des véhicules.

2 *Représentation de Transport à la Demande (PTD)*

Le Problème de Transport à la Demande (PTD ou DRP) consiste à prendre en charge le transport d'un certain nombre de voyageurs d'un lieu de départ à un lieu d'arrivé.

Ce dernier est caractérisé par un ensemble de demande de transport de taille « n » et d'un nombre de véhicules « m » pour servir ces dernies. Chaque demande de transport est modélisée par une requête contenant des informations sur la demande. L'exécution d'une requête consiste à prendre en charge une personne du point de ramassage « i » et de la déposer au point « i + n ».

Le ramassage en « i » doit démarrer dans la fenêtre de temps [ai, bi] et l'arrivée à la destination doit se faire dans la fenêtre de temps [ai + n, bi + n]. En effet, le PTD est une extension du Problème de Tournées de Véhicules (PTV) avec pour le PTD une contrainte supplémentaire qui est la cohérence de l'ordre de passage d'un véhicule pour servir une demande [1].

3 Travaux de recherche sur le Transport à la Demande (TAD)

Le TAD est un service de transport qui doit être présent dans différents lieux avec une flotte de véhicules. Le problème dans les TAD est de déterminer un ensemble optimisé (selon certains critères) d'itinéraires de véhicules permettant de répondre aux différentes demandes de transport. La nature des services à accomplir conditionne fortement la structure des problèmes. On distingue ainsi des sous-familles de problèmes de calcul de tournées. En effet, le Problème de Transport à la Demande (PTD) généralise un certain nombre de problèmes de tournées de véhicules tels que le ramassage et la livraison ou en anglais Pickup and Delivery Vehicle Routing Problem (PDVRP) .

3.1 Le Problème de Transport à la Demande (PTD) avec un seul véhicule

Un des cas les plus simples du problème de transport à la demande est celui où tous les utilisateurs sont desservis par un seul véhicule. Nous allons présenter dans cette section des algorithmes pour le cas statique, où toutes les demandes sont connues à l'avance, et pour le cas dynamique, où les demandes de transport arrivent peu à peu en temps réel.

3.1.1 Le PTD statique

En 1980[2], Psaraftis, a formulé et résolu le PTD avec un programme dynamique dans lequel les fonctions objectif est la minimisation de la somme pondérée des temps de service pour un véhicule et le nombre de clients insatisfaits [2].

L'insatisfaction du client est exprimée comme une combinaison pondérée des temps d'attente avant l'instant de ramassage et le temps de parcours. Dans cette solution pour le PTD, le transporteur impose une contrainte limitant la différence entre la position d'un utilisateur dans la liste d'appel et sa position dans la route d'un véhicule.

Cet algorithme a ensuite été mis à jour par le même auteur (Psaraftis, 1983[3]). Pour traiter des fenêtres de temps spécifiées par les voyageurs.

Comme c'est souvent le cas des approches basées sur la programmation dynamique, l'algorithme ne peut résoudre avec optimalité que des problèmes à petite taille, car la procédure a une complexité de O(n2) [3].

Le plus grand PTD résolu en utilisant cette approche contient neuf utilisateurs. Alors que la plupart des PTDs dans la pratique sont formulé par un nombre de demandes de transport plus important. En effet l'approche proposée pourrait encore s'avérer utile en tant que sous-routine dans un algorithme multi-véhicules, à condition que le nombre d'utilisateurs desservis par le même véhicule reste relativement faible.

Sexton (1979) [4], et Sexton et Bodin (1985a, b) [5][6], ont également résolue le PTD avec un seul véhicule. Cette résolution est une étape dans le problème avec plusieurs véhicules.

Leur approche se base sur une heuristique dans laquelle les voyageurs ont déjà été regroupés. Leur algorithme alterne entre la résolution d'un problème de routage par le biais d'une heuristique d'insertion et la résolution du problème d'ordonnancement associé.

3.1.2 Le PTD dynamique

Le PTD dynamique a été également étudié par Psaraftis (1980) [2]. Dans le PTD dynamique des nouvelles demandes de transport sont produites de façon dynamique au cours du temps, mais aucune information sur le futur des demandes n'est disponible (contrairement à ce qui se passe dans la programmation stochastique). Lorsqu'une nouvelle demande est produite, une solution prévisionnelle sera disponible.

Toutes les demandes qui ont été déjà traitées avant l'instant t ne seront pas retirées de la solution. Le problème est alors de ré-optimiser la partie de la solution après l'instant t, y compris la nouvelle demande.

Ceci est réalisé par l'application de l'algorithme de programmation dynamique développé pour le cas statique.

Une difficulté pratique qui paralyse cette approche est celle de la possibilité de résoudre le problème à l'instant t avant l'arrivée de la prochaine requête, ce qui ne peut être possible que si l'algorithme est lent et que les demandes arrivent rapidement.

Une solution a été proposée dans le cadre de la localisation dynamique d'ambulance par Gendreau, Laporte et Semet (2001) pour contourner cette difficulté. La solution consiste à pré-calculer plusieurs scénarios, en utilisant le calcul parallèle par la prévision de demandes de transport futures.

Malgré ses limites, le travail de Psaraftis sur le PTD dynamique a aidé à définir les concepts utilisés dans les recherches faites sur les problèmes de routage dynamique

3.2 *Le Problème de Transport à la Demande (PTD) avec plusieurs véhicules*

Beaucoup de recherches ont été effectués sur le PTD à plusieurs véhicules. Notre état de l'art concernant ce problème est subdivisé encore en cas statique et dynamique.

3.2.1 Le cas statique

En 2003 [7], Cordeau et Laporte ont proposé une heuristique basée sur l'algorithme de recherche tabou pour résoudre le PTD avec plusieurs véhicules [7]. Dans leur travail, les utilisateurs spécifient une fenêtre de temps sur l'heure d'arrivée et de départ de leur voyage. En effet, dans cette approche un temps de trajet maximal est associé à chaque demande de transport. Ce temps peut être soit le même pour toutes les demandes, ou calculé en utilisant un facteur de déviation maximale de la durée du trajet direct de chaque demande en particulier.

Dans l'article de Aldaihani et Dessouky (2003) [8], les auteurs traitent un PTD hybride dans lequel deux types de services sont offerts : le transport de porte à porte, et le transport vers et depuis un arrêt de bus [8]. Le choix entre les deux modes est fait par l'algorithme proposé.

Diana et Dessouky (2004) ont appliqué une procédure pour résoudre une version du PTD dont l'objectif est une somme pondérée de la distance, du temps de trajet, et du temps d'inactivité des véhicules. Ils ont fournie des résultats à partir des tests effectués sur des PTDs de tailles qui varient entre 500 et 1000 demandes.

En 2006, quatre heuristiques ont été publiées pour résoudre PTD à plusieurs véhicules. Dans l'article rédigé par Rekiek, Delchambre et Saleh (2006) [9], le principal objectif est la minimisation du nombre de véhicules utilisés pour servir les voyageurs. Les auteurs proposent un algorithme génétique pour la phase de regroupement de voyageurs et un mécanisme d'insertion pour la phase de routage.

Plus récemment, Claudio et al (2009) [10] ont proposé un algorithme génétique pour la résolution du PTD avec plusieurs véhicules [10]. Les auteurs ont testé leur approche sur le benchmark de Cordeau et Lapote (2003) [11].

3.2.2 *Le cas dynamique*

Colorni et Righini (2001) [12] ont testé trois objectifs différents du PTD dynamique, à savoir la maximisation des demandes de transport servies, la maximisation de la perception du niveau de service par des voyageurs, et la minimisation de la distance parcourue [12].

Dans leur recherche, un mécanisme d'insertion en alternance est mis en place. En effet, ce mécanisme
consiste en une phase de regroupement de voyageur et d'une phase de routage de véhicules. L'algorithme de routage est un algorithme branch-and-cut appliqué à un ensemble de demandes dont les fenêtres de temps ne sont pas trop éloignées dans le futur. Les auteurs motionnent qu'ils ont réalisé des expériences avec leur approche à Crema et Verbania, situés dans le nord de l'Italie.

Un algorithme développé par Coslovich, Pesenti et Ukovich (2006) [13] suit une stratégie en deux phases pour l'insertion d'une nouvelle demande de transport dans un itinéraire existant [13]. Une phase offline est d'abord utilisée pour créer un ensemble possible d'itinéraires concernant les demandes connues en se basant sur le mécanisme 2-opt. Après Une phase online est utilisée pour insérer la nouvelle demande en tenant compte de l'objectif de minimisation de l'insatisfaction des utilisateurs.

4 *Conclusion*

Dans ce chapitre nous avons présenté un état de l'art des problèmes de transport à la demande (PTD).

Le PTD est un problème de routage important et difficile à résoudre. En effet, ce problème est rencontré dans plusieurs contextes et est susceptible de devenir très important dans les prochaines années. Il partage plusieurs caractéristiques avec les problèmes de ramassage et de livraison, mais comme il est un problème de transport de personnes, le niveau des critères de service devient plus important. Ainsi, la ponctualité, la réduction des temps d'inactivité et le coût de transport sont plus critiques dans le PTD.

Chapitre 2 : l'algorithmes de classification de data mining

1 Introduction

La classification est une tâche appliquée dans la vie courante. Elle est utilisée pour expliquer les nouveaux phénomènes rencontrés en les comparants avec des concepts et des phénomènes connus et en essayant de rapprocher les caractéristiques le plus possible.

C'est un sujet de recherche actif qui se place au cœur de data mining, ce qui justifie pleinement l'intérêt qui lui est porté. Les techniques de classification sont réparties essentiellement en deux approches : la classification supervisée et la classification non supervisée (appelée aussi ''clustering'').

2 L'objectif de clustering

L'objectif le plus simple de clustering est de répartir une population d'objets en groupes d'observations homogènes (classes), chaque classe étant bien différenciée des autres, pour former une partition.

Le clustering est un sujet abordé par plusieurs métiers. Ce fait lui donne l'avantage d'être utilisé par divers domaines d'application qui peuvent être catalogués suivant le but recherché selon trois grands types [14] :

- Extraction des connaissances : Dans ce type d'applications, on cherche généralement à donner une description sémantique aux groupes constitués afin de donner un sens à l'information dont on dispose ;

- Réduction des bases de données : L'objectif visé est de réduire l'espace des données sur lequel on travaille en espérant que le problème qu'on cherche à résoudre devient beaucoup plus simple ;

- Etude de comportement (*profiling*) : Le but est de détecter des sous populations qui partagent des caractéristiques proches, ce qui leurs donne un certain comportement commun. Les systèmes de décision se basent souvent sur l'étude de comportement afin d'entreprendre des actions convenables à chacune des sous population.

Dans ce qui suit, nous allons présenter les principaux algorithmes de clustering parmi les plus représentatifs .

3 Définition de clustering

Le clustering est une approche de classification dont les classes existent a posteriori. Au départ, on dispose d'un ensemble d'objets non étiquetés (dont la classe est inconnue).
A partir de ces objets, l'idée est de parvenir à détecter des objets ''similaires'' afin de les regrouper dans des classes [14].
Selon [15], le clustering consiste à diviser une population d'objets en sous-ensembles d'objets appelés classes pour que tous les objets dans une même classe soient similaires et les objets de classes distinctes soient dissimilaires. Parmi les objectifs que le clustering doit permettre d'atteindre, citons la réduction des données et la prédiction basée sur les groupes.

4 Terminologie

Avant d'entamer le processus de clustering, il est indispensable de définir quelques concepts :

4.1 Représentation d'une classe

C'est la description de l'ensemble des objets constituant la classe. Cette description, caractérisée par un ensemble de paramètres, permet de définir la forme et la taille de la classe dans un espace de données.
Dans la plupart des algorithmes de classification, la description d'une classe est simplement une combinaison linéaire des objets (centroide), ou par un axe médian (médoide) [16].

4.1.1 Le centroide :

Un centroide d'une classe *C* est un vecteur où chacune de ses composantes correspond à la moyenne arithmétique (quand elle existe) des objets de la classe *C* [16].

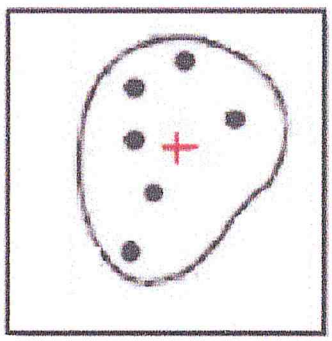

Figure 2.1 Représentation d'un centroide [16]

4.1.2 Le médoide

On prend *k* objets parmi les objets de la classe. Ces *k* éléments sont centraux vis-à-vis de la classe, c'est-à-dire qu'ils sont proches du centre géométrique de la classe [16]. Ils permettent de représenter la classe non pas par un objet unique, qui n'a pas toujours d'existence réelle, mais plutôt par un noyau médian censé définir au mieux la classe.

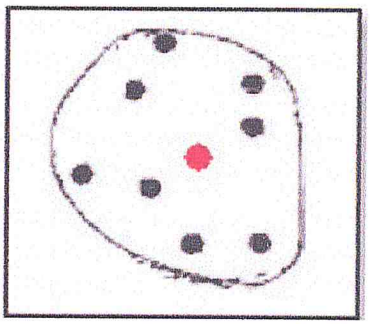

Figure2.2 Représentation d'un médoide (k=1) [16]

4.2 La notion de proximité (distance)

Une mesure de distance (appelée aussi mesure de similarité ou de proximité) exprime une similarité ou une dis-similarité : plus deux objets se rassemblent, plus leur similarité est grande et plus deux objets s'éloignent, plus leur similarité est petite.
La mesure de distance peut s'appliquer entre deux objets ou entre deux classes.

4.2.1 Distance entre deux objets

Cette mesure peut être exprimée par une formule qui génère un nombre positif reflétant la proximité ou l'éloignement entre deux objets.
Selon [17], pour qu'une fonction de deux objets $d(x, y)$ soit une vraie métrique de distance, elle doit remplir les conditions suivantes :

- $d(x, y) = 0$ si et seulement si $x = y$
- $d(x, y) \geq 0$ quels que soient x et y
- $d(x, y) = d(y, x)$. (Symétrie)
- $d(x, y) \leq d(x, z) + d(z, y)$. (Inégalité triangulaire)

Les mesures de distance entre objets dépendent notamment du type de données. Parmi les mesures qui traitent des données numériques très populaires, on trouve la distance de Minkowski [18], notée *dm*, entre deux objets *x*, *y* définis chacun par ses attributs.

4.2.2 Distance entre deux classes

Les mesures inter-classe les plus populaires dans la littérature sont : le lien simple(*singlelink)*, le lien moyen (*average link)* et le lien complet (*complete link)* [19].
La distance D entre 2 classes *C*1, *C*2 notée *D* (*C*1, *C*2) est calculée pour toute paire d'objets, un objet de chaque classe. Une opération (Minimum pour le lien simple, Moyenne pour le lien moyen et Maximum pour le lien complet) est ensuite appliquée pour chaque paire d'objets.
D (*C*1, *C*2) = Opération { *d(x, y)* | $x \in C1, y \in C2$ } [19].
Tel que : *d(x, y)* = distance entre les deux objet *x* et *y*.

4.3 L'inertie

L'inertie d'une population d'objets est la moyenne des carrés des distances des individus à leur centre de gravité. Cette entité mesure l'homogénéité des objets de la population. L'inertie intra-classe est la somme des inerties des classes. Plus cette entité est petite, plus les objets de la même classe sont homogènes.
L'inertie inter-classe est la moyenne (pondérée par le nombre de classes) des carrées des distances des centres des classes au centre global. Plus cette entité est importante, plus les classes sont éloignées [20].
Un bon clustering possède deux propriétés :

- Inertie intra-classe (variance des objets dans la même classe) minimale ;
- Inertie inter-classe (variance des centres des classes) maximale.

5 Les algorithmes de clustering

La majorité des algorithmes de clustering peuvent être répartis en trois grandes familles [21], [22] et [23] :

- Les algorithmes de partitionnement : Ils adoptent une recherche itérative des classes jusqu'à l'optimisation d'un critère d'arrêt (par exemple atteindre une certaine inertie).

- Les algorithmes hiérarchiques : Ils sont descendants si, à partir d'une seule classe, ils cherchent à établir une partition par division ; ou ascendants s'ils cherchent à former des classes plus grandes par fusion de classes jusqu'à la satisfaction d'un critère d'arrêt.

- Les algorithmes à base de densité : Les classes sont formées selon le voisinage des objets et le niveau de densité de chaque objet.

5.1 Les algorithmes de partitionnement

Leur principe général est de démarrer à partir d'un certain nombre de classes qui sont partitionnés d'une manière aléatoire en effectuant une ''redistribution'' des objets ou en essayant d'identifier les classes comme étant des régions très peuplées jusqu'à la rencontre d'un critère d'arrêt. Les algorithmes de partitionnement que nous allons développer sont k-Means, k-Medoïds, PAM, CLARA et CLARANS.

5.1.1 L'algorithme k-Means

L'algorithme k-Means (ou l'algorithme des centres mobiles) a été introduit par J.MacQuenn et mis en œuvre sous sa forme actuelle par E.Forgy [21]. Il est le plus utilisé dans les applications scientifiques et industrielles car il est le plus simple et le plus rapide.
Dans cet algorithme, chaque classe est représentée par la moyenne (centroide). K-Means est un algorithme itératif. Il commence avec un ensemble de k individus de référence choisis de façon aléatoire. Les individus de données sont ainsi partitionnés dans k classes ; un individu appartient à une classe si le centre de cette classe est le plus proche de lui (en terme de distance). La mise à jour des centroides et l'affectation des individus de données aux classes sont réalisées pendant les itérations successives [24].
La *Figure 3* montre un exemple de déroulement de l'algorithme k-Means sur un nuage d'objets bidimensionnels, avec $k = 3$.

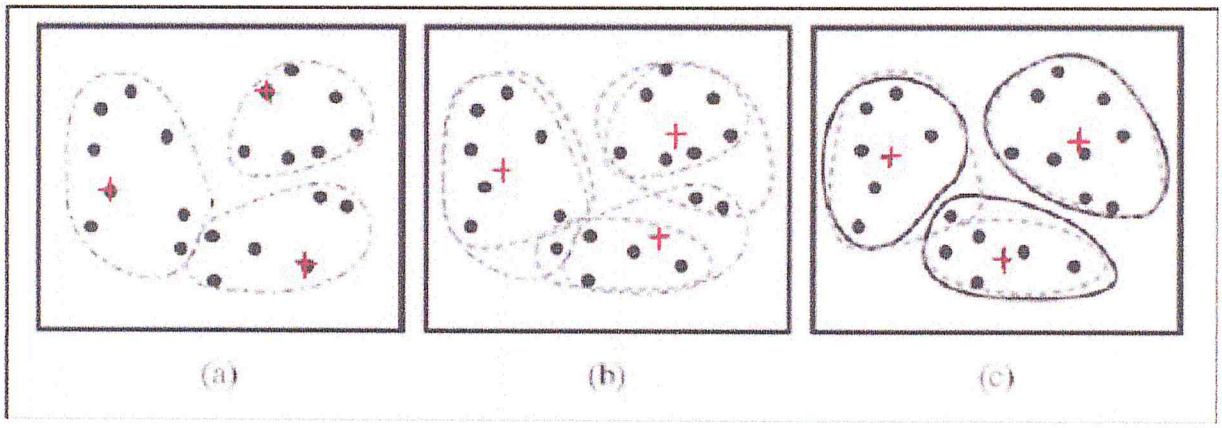

Figure 2.3 Partitionnement basé sur k-Means [18].

Figure 3.a : Choix aléatoire de trois centre initiaux (objets) marqués par (+) et affectation de chaque objet restant au centre le plus proche ;

Figure3.*b* : Calcul du nouveau centre pour chaque classe (moyenne des objets de la classe) et redistribution des objets selon les nouveaux centres ;

Figure 3.*c* : Le processus est répété jusqu'à stabilité des classes.

Comme avantages de cet algorithme, on cite [24]

- ✓ Il s'adapte bien pour des populations de tailles importantes ;
- ✓ Il est relativement efficace ; si n le nombre d'objets, k le nombre de classes et t le nombre d'itération, l'algorithme converge, généralement, avec k et t suffisamment inférieur à n (k, $t << n$) ;
- ✓ Il est indépendant de l'ordre d'arrivée des données.

Parmi les inconvénients de cet algorithme, on cite [24] :

- Il est applicable seulement dans le cas où la moyenne des objets est définie ;
- Le nombre de classes k, doit être spécifié à priori ;
- Il converge souvent vers un optimum local ;
- Il est sensible aux objets isolés (bruits) ;
- Il n'est pas adapté pour découvrir des classes avec des structures non-convexes, et des classes de tailles et formes différentes.

5.1.2 L'algorithme k-Medoids

Il est introduit par Kaufman et Rousseew [21]. L'esquisse de l'algorithme k-Medoids ressemble à celle de k-Means sauf que, contrairement à l'algorithme k-Means où la classe est représentée par une valeur moyenne, le centroide, dans l'algorithme k-Medoids, une classe est représentée par un de ses objets prédominants, le médoide.
L'algorithme k-Medoids utilise une fonction objectif qui définit la distance moyenne entre un objet et le médoide.

La Figure 4 est une illustration du déroulement de l'algorithme k-Medoids sur un nuage d'objets bidimensionnels avec $k = 3$.

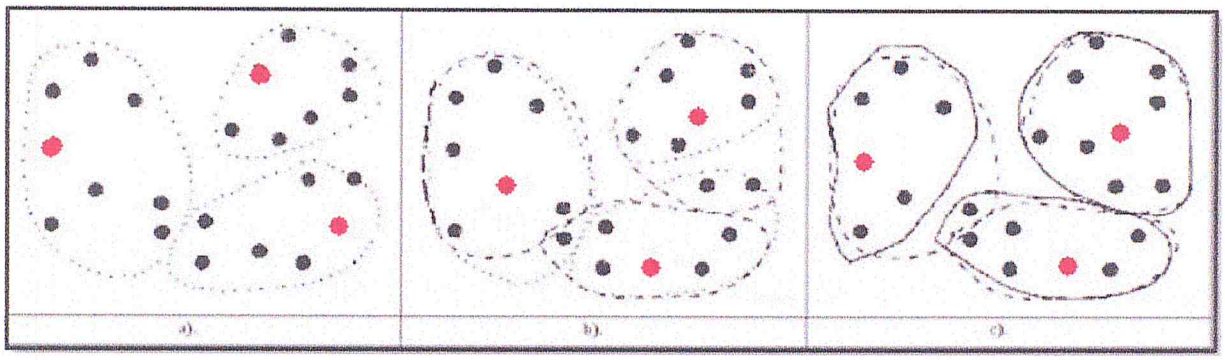

Figure 2.4 Partitionnement basé sur k-Medoids [21].

Figure 4.a : Choix des trois centres initiaux marqués par ( ) et affectation de chaque objet restant au centre le plus proche ;
Figure 4.b : Calcul des nouveaux médoides pour chaque classe et redistribution des objets selon les nouveaux médoides (Le médoide étant l'objet le plus proche du centroide de la classe) ;
Figure 4.c : Le processus est répété jusqu'à stabilité des classes.

Comme avantages de cet algorithme, on cite [24] :

- ✓ Il s'adapte bien pour des populations de tailles importantes ;
- ✓ Il est relativement efficace ; si n le nombre d'objets, k le nombre de classes et t le nombre d'itération, l'algorithme converge, généralement, avec k et t suffisamment inférieur à n ($k, t << n$) ;
- ✓ Il est indépendant de l'ordre d'arrivée des données.

Parmi les inconvénients de cet algorithme, on cite [24] :

- ✓ Il est applicable seulement dans le cas où la moyenne des objets est définie ;
- ✓ Le nombre de classes *k*, doit être spécifié à priori ;
- ✓ Il converge souvent vers un optimum local ;
- ✓ Il est sensible aux objets isolés (bruits) ;
- ✓ Il n'est pas adapté pour découvrir des classes avec des structures non-convexes, et des classes de tailles et formes différentes.

5.2 Les algorithmes basés sur la densité

Ces algorithmes considèrent les classes comme étant des régions denses dans l'espace d'objets. Un objet de l'espace est dense si le nombre de ses voisins dépasse un certain seuil fixé à priori. Ils essayent alors d'identifier les classes en se basant sur la densité des objets dans une région. Les objets sont alors groupés non pas sur la base d'une distance mais sur la base de la densité de voisinage excède une certaine limite.

Parmi les algorithmes les plus connus dans cette catégorie, nous citons DBSCAN et OPTICS [21], [23].

Avant de développer les algorithmes, il faut d'abord définir quelques concepts [25] :

Soient :

X : ensemble d'objets ;

ε : rayon maximum de voisinage ;

MinPts : nombre minimum d'objets dans le voisinage d'un point.

Le voisinage d'un objet ***p*** par rapport à **ε** : $\mathbf{V\varepsilon\ (p) = \{q\ \epsilon\ X\ /\ distance\ (p, q) \leq \varepsilon\ \}.}$

Un objet ***p*** est ''directement densité-accessible'' à partir de ***q*** (Figure 5) si :

· $p\ \ V\varepsilon\ (q)$

· $|V\varepsilon\ (q)| >$ **MinPts** (**q** est dit noyau)

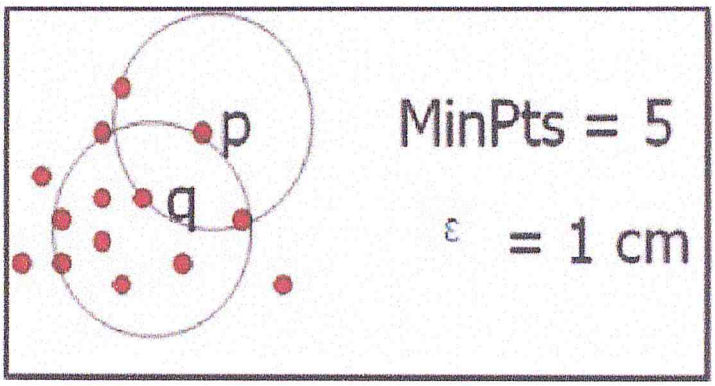

Figure 2.5 Illustration du concept "directement densité-accessible" [25]

Un objet *p* est ''accessible'' à partir de *q* s'il existe *p1*, ..., *pn*, *p1* = *q*, *pn* = *p* tel que *pi+1* est directement densité-accessible à partir de *pi* pour $1 < i < n$. (Figure 6)

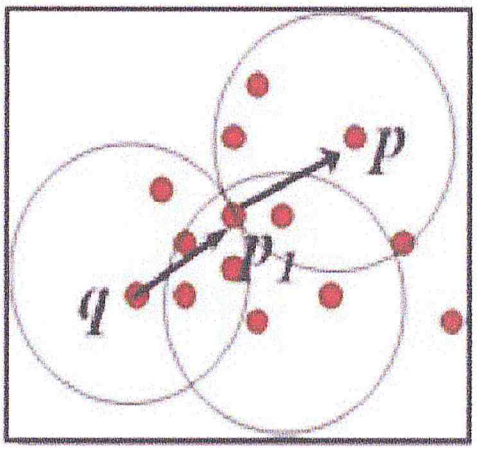

Figure 2.6 Illustration du concept "accessible" [25]

Un objet *p* est ''connecté'' à *q* s'il existe un objet *o* tel que *p* et *q* soient accessibles à partir de *o* (Figure7).

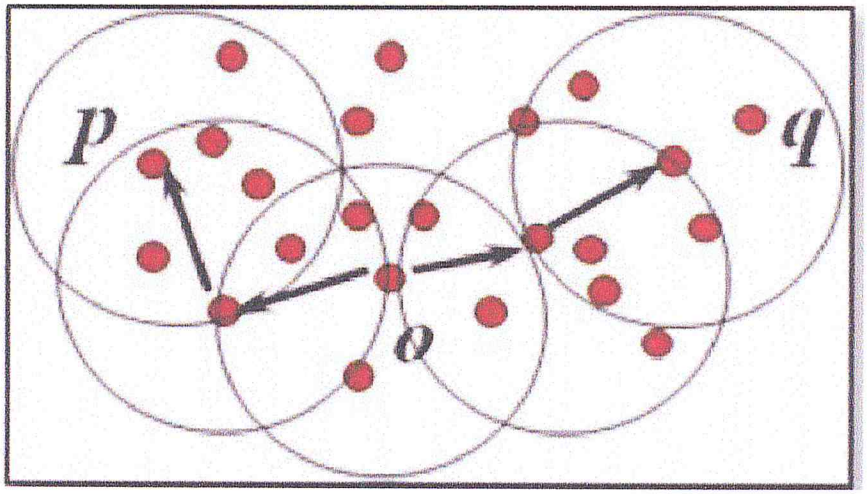

Figure 2.7 Illustration du concept "connecté" [25]

Une classe est l'ensemble maximal des objets connectés [25].

5.2.1 L'algorithme DBSCAN

DCAN (*Density Based Spatial Clustering of Application with Noise*) est introduit par Ester, Kriegel, Sander et Xu [21].

La démarche générale de DBSCAN se résume comme suit [25] :

1. Sélectionner aléatoirement un objet p de l'ensemble d'objets X.
2. Déterminer l'ensemble E des objets accessible depuis p dans X.
3. Si p est un noyau, alors E est une classe.
4. Sinon sélectionner un autre objet et aller en (2).
5. S'arrêter lorsque tous les objets ont été sélectionnés.

Comme avantages de DBSCAN, on cite [15] :

- ✓ Il peut chercher les classes de forme arbitraire ;
- ✓ Il est insensible à l'ordre d'arrivée des objets ;
- ✓ Il est incrémental car un nouvel objet qui arrive peut affecter certains voisinages.

Parmi les inconvénients de cet algorithme, on cite [15] :

- Il est difficile à préciser le rayon de voisinage ε ;
- La performance peut être diminuée pour les données de haute dimension.

5.2.2 L'algorithme OPTICS

L'algorithme OPTICS a été introduit par Ankerst, Breunig, Kriegel et Sandar pour pallier au premier inconvénient de l'algorithme DBSCAN. Il consiste à effectuer un clustering DBSCAN
pour plusieurs valeurs du rayon de voisinage de manière simultanée [21].
L'algorithme procède de la manière suivante :

1. Commencer par créer une liste ordonnée de différents rayons ;
2. Appliquer ensuite DBSCAN, en parallèle, à cette liste ;
3. Choisir la plus petite distance qui assure la plus forte densité.

5.3 Les algorithmes hiérarchiques

Les algorithmes hiérarchiques , Ces algorithmes construisent les classes graduellement sous une forme hiérarchique, autrement dit, un arbre des classes appelé'dendrogramme' (Figure 8).
Ils sont divisés en **2 types** : **Agglomératifs** (ascendants) et **divisifs** (descendants).
En agglomération, on commence par considérer chaque objet comme une classe et on essaye de fusionner deux ou plusieurs classes appropriées (selon une similarité) pour former une nouvelle classe. Le processus est répété jusqu'à atteindre un critère d'arrêt.
En division, tous les objets sont au début considérés comme une seule classe, on divise successivement les classes en classes plus raffinées selon une similarité. Le processus est répété jusqu'à atteindre un critère d'arrêt.
Le critère d'arrêt peut être le nombre de classes désiré, le nombre minimum (ou maximum) d'objets dans chaque classe, le nombre d'itérations, l'inertie...etc.

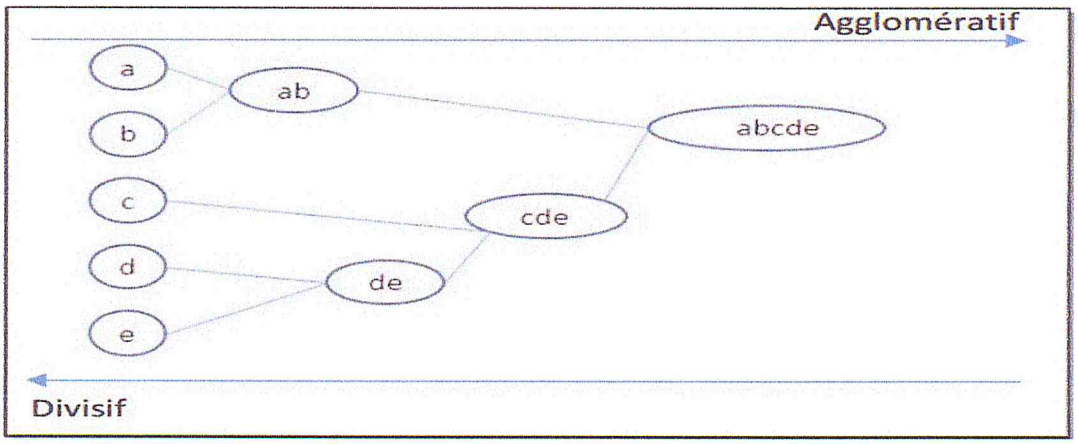

Figure 2.8 Dendrogramme [24].

5.3.1 La classification ascendante(Agglomératifs) hiérarchique

La classification ascendante hiérarchique, régulièrement mentionnée sous l'acronyme CAH, fait partie des trois familles d'outil d'analyse non supervisée les plus répandues avec les méthodes de partitionnement et les méthodes à noyaux de densité.

La classification ascendante hiérarchique est particulièrement efficace dans la construction de clusters/groupes d'observations d'une matrice de données à $P \geq 1$ variable(s) continue(s), $X = (X^1, \ldots\ldots, X^P)$, à partir de la structure même des données sans apport informatif d'une variable auxiliaire.

Le clustering, ou l'action de créer des groupes d'observations, ou clusters dans un argo-anglicisme bien souvent utilisé pour simplifier le propos, s'exécute via un algorithme ascendant en définissant deux paramètres,

– une fonction de distance (également appelée fonction de proximité) : distance euclidienne, de Manhattan, de Minkowski, etc.

– et une méthode spécifique de regroupement : distance maximale, méthode de Ward, méthode de Mc Quitty, etc, à partir desquelles la typologie des observations de **X** sera construite au travers de partitions emboîtées d'hétérogénéités croissantes d'où sa propriété hiérarchique. Il existe un très grand nombre de méthodes de regroupement et chacune d'elles dispose d'avantages et d'inconvénients.

A noter que la classification ascendante hiérarchique ne requiert aucune condition de validité, cependant l'application de méthodes de transformation des données peut avoir un effet plus ou moins accentué sur les clusters conçus en fin d'algorithme et sont même parfois recommandées avant application.

Enfin, le choix du nombre de clusters se fait en fonction de deux approches :
– soit nous définissons au préalable un nombre de groupes maximales à composer,

– soit en fonction de quatre outils décisionnels que nous présenterons: le R^2 , le $pseudo - T^2$, le $semi - partiel - R^2$ et le Cubic Clustering Criterion.

Le choix entre ces deux méthodes est purement subjectif et principalement lié au contexte car aucune n'est à considérer comme la meilleure.

5.3.1.1 *L'algorithme de classification ascendante hiérarchique*

La CAH regroupe un ensemble de techniques construites sur le même principe : le regroupement successif des points par ordre de proximité croissante [25]. L'Algorithme commun est le suivant :

- **Étape initiale :** Considérer autant de classes qu'il y a d'observations et regrouper les deux observations qui sont les plus proches selon la fonction de distance choisie.
- **Étapes récursives suivantes:** Tant qu'il reste une observation isolée (sans groupe) ou au moins deux groupes d'observations à fusionner, pour l'itération t,
- Calculer la matrice de distance entre les différents groupes d'observations et les observations isolées, selon la fonction de distance choisie et lorsque nous sommes en présence de groupe d'observations, selon la méthode de regroupement choisie.
- La distance minimale d_t au sein de la matrice de distance calculée à l'étape t indique le regroupement à faire. Il pourra alors s'agir aussi bien de la fusion de deux groupes d'observations comme d'un groupe observations et d'une observation isolée ou deux observations sans groupe. La distance d_t retenue sert enfin à construire l'arbre de classification ascendante hiérarchique.
- Fin de l'itération t, nous passons à l'itération t+1en reprenant les deux précédentes étapes.
- **Étape finale :** A ce stade nous sommes dans la situation où il nous reste soit deux groupes d'observations, soit un groupe d'observations et une observation isolée, nous les regroupons pour conclure l'algorithme. Une fois toutes les observations classées, nous disposons de l'arbre de classification ascendante hiérarchique présentant la structure du clustering et permettant de déterminer à quel niveau couper l'arbre pour visualiser les différentes classifications possibles en fonction des différentes classes emboîtées.

5.3.1.2 *Les fonctions distances*

Comme expliquer en introduction, il faut définir au préalable deux paramètres afin d'utiliser une CAH. Nous présentons le premier : la fonction de distance à utiliser pour le calcul de la distance entre deux observations décrites au travers de P variables. Six principales fonctions existent, les voici :

- La distance euclidienne: $d(X_{i1}, X_{i2}) = \sqrt{\sum_j^p (X_{i1}^j - X_{i2}^j)^2}$ [25].
- La distance maximale : $d(X_{i1}, X_{i2}) = max_{j\epsilon[1,P]} \left| X_{i1}^j - X_{i2}^j \right|$ [25].
- La distance de Manhattan: $d(X_{i1}, X_{i2}) = \sum_{j=1}^{P} \left| X_{i1}^j - X_{i2}^j \right|$ [25].
- La distance de Canberra: $d(X_{i1}, X_{i2}) = \sum_{j=1}^{P} \frac{\left| X_{i1}^j - X_{i2}^j \right|}{\left| X_{i1}^j \right| + \left| X_{i2}^j \right|}$ [25].

La distance de Minkowski (pour q, paramètre à fixer), qui est entre autre une généralisation de la distance euclidienne: $d(X_{i1}, X_{i2}) = \left(\sum_{j=1}^{P} \left(X_{i1}^j - X_{i2}^j \right)^q \right)^{\frac{1}{q}}$

5.3.1.3 *Les méthodes de regroupement*

Nous abordons maintenant le second paramètre à déterminer : la méthode de regroupement. Plusieurs méthodes existent pour déterminer la proximité entre deux clusters ou un cluster et une observation isolée, en voici les principales.

5.3.1.3.1 La distance moyenne

Selon RR.Sokal et CD.Michener, La distance moyenne se détermine en calculant toutes les distances possibles entre les observations respectives aux deux groupes étudiés et en déterminant la moyenne de ces distances. Les deux groupes dont la distance moyenne est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [26].

- ✓ Propriétés : La distance moyenne tend à créer des clusters de faible variance identique. De plus, elle est moins sensible au bruit.

5.3.1.3.2 La distance médiane

J C.Gower,La distance des médianes se détermine en calculant toutes les distances possibles entre les observations respectives aux deux groupes étudiés et en déterminant la valeur médiane de ces distances. Les deux groupes dont la distance médiane est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [27].

- ✓ Propriétés : La distance des médianes présente l'avantage d'être particulièrement robuste aux outliers s'ils sont trop nombreux. Pour le reste, elle demeure l'une des moins populaires et très rarement utilisée lors d'une CAH (classification ascendant hiérarchique) .

5.3.1.3.3 La distance minimale

K. Florek et al, La distance minimale consiste à calculer l'ensemble des distances possibles entre les observations respectives aux deux groupes considérés et à en retenir la plus petite. Les deux groupes dont la distance minimale est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [28].

Propriétés : La distance minimale est plus robuste que son homologue la distance maximale mais présente néanmoins une faiblesse (voir une force si c'est l'objectif visé). Soit la situation de deux classes éloignées et qui peuvent être reliées par un faible nombre d'observations proches les unes des autres et faisant la jonction entre les deux groupes. Dès lors, le regroupement basé sur la distance minimale ne considérera pas trois clusters (les deux groupes séparés et le lot d'observations entre eux) mais un unique cluster.

5.3.1.3.4 La distance maximale

Thorvald Julius Sørensen, La distance maximale consiste à calculer l'ensemble des distances possibles entre les observations respectives aux deux groupes considérés et à en retenir la plus grande. Les deux groupes dont la distance maximale est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [29].

- ✓ Propriétés : La distance maximale produit généralement des clusters de diamètres à peu près égaux, cependant elle est fortement influencée par la présence d'outliers. Logique par construction, étant donné le fait qu'elle se base sur la distance maximale

entredeux clusters. A l'instar de la distance médiane, la distance maximale est très peu utilisée.

5.3.1.3.5 La distance des barycentres

Selon sokal, La distance des barycentres se détermine en calculant les barycentres des différents groupes à partir de la moyenne des coordonnées des observations respectives à ces clusters. Les deux groupes dont la distance des barycentres est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [26].

- ✓ Propriétés : La distance des barycentres demeure plus robuste aux outliers que les autres méthodes, cependant elle est réputée comme étant moins performante que la méthode de Ward ou encore la distance moyenne.

5.3.1.3.6 La distance de Ward

Ward 1963, La distance de Ward se base sur la distance au carré entre les barycentres des deux groupes considérés et pondérés par leur effectif respectif. En notant Cl_{K1}, Cl_{K2} les deux clusters considérés et de barycentres respectifs, G_{K1}, G_{K2} la formule d'usage est alors,

$$D_{Cl_{K1},Cl_{K2}} = \frac{\|G_{K1} - G_{K2}\|^2}{\frac{1}{n_{K1}} + \frac{1}{n_{K2}}}$$

Les deux groupes dont la distance de Ward est la plus petite sont ceux qui seront fusionnés pour l'itération en cours [30].

- ✓ Propriétés : La distance de Ward est la méthode la plus souvent utilisée car elle a été conçue dans un objectif de maximisation de l'inertie inter-classe et de minimisation de l'inertie intra-classe. Elle aura tendance à produire des clusters de même taille et reste très sensible aux outliers. De plus, contrairement à la distance minimale, elle est très peu efficace face aux classes « étirées ».

5.3.1.4 La méthode du maximum de vraisemblance

Sarle,S 1983 , La méthode procède au regroupement d'observations par maximisation de la vraisemblance à chaque itération sous les hypothèses de mélange de lois multi-normales, d'égalité des matrices de variances-covariances et de probabilités d'échantillonnage inégales[31]. Le choix du regroupement est donc déterminé au travers de la formule

$$D_{Cl_{K1},Cl_{K2}} = n.P.log\left[1 + \frac{\sum K1 \cup K2 - \sum K1 - \sum K2}{\sum_{K+1}^{K} \sum K}\right] - \lambda\left[n_{K_1 \cup K_2} log(n_{K1 \cup K2}) - n_{K1} LOG(n_{K1}) - n_{K2} log(n_{K2})\right]$$

Où,

–n désigne le nombre d'observations,

- $\sum K1 \cup K2$la matrice de variances-covariances issues du regroupement des clusters,Cl_{K1}, Cl_{K2}
- $n_{K1 \cup k2}$ l'effectif obtenu lorsque que nous regroupons les clusters Cl_{K1}, Cl_{K2}

– le paramètre, fixé à 2 par défaut, représente la pénalité à appliquer aux matrices de variances-covariances afin de s'affranchir d'éventuel biais dû aux outliers.

Les deux groupes dont la distance calculée à partir de la méthode EML est la plus petite sont ceux qui seront fusionnés pour l'itération en cours.

Propriétés : La méthode EML est similaire à la méthode de Ward, réputée comme l'une des plus performantes méthodes de classification. Cependant, elle a tendance à créer des groupes de tailles disproportionnées.

5.3.1.5 La méthode du β-flexible

Selon Godfrey, W 1967, La méthode du β-flexible se base directement sur la maximisation de l'inertie inter-classe et la minimisation de celle intra-classe au travers de la formule suivante :

$$D_{Cl_{K1},Cl_{K2} \cup Cl_{K3}} = \left(D_{Cl_{K1},Cl_{K2}} + D_{Cl_{K1},Cl_{K3}}\right)\frac{1-\beta}{2} + \beta . D_{Cl_{K2},Cl_{K3}}$$

En général, le paramètre β est fixé à −0.25 par défaut. Les distances $D_{Cl_{K1},Cl_{K2}}$et $D_{Cl_{K1},Cl_{K3}}$ $D_{Cl_{K1}}, Cl_{K2}$ et $D_{Cl_{K1}}, Cl_{K3}$ correspondent à celles entre le groupe à fusionner aux deux autres groupes (maximisation de l'inertie inter-groupe) et la distance $D_{Cl_{k2}}, Cl_{K3}$ est celle entre les deux groupes déjà construits (minimisation de l'inertie intra-groupe). Les deux groupes dont la distance calculée à partir de la méthode du β-flexible est la plus petite sont ceux qui seront fusionnés pour l'itération en cours.

- ✓ Propriétés : La méthode du β-flexible est optimale dans une approche visant à obtenir les meilleurs critères de classification possibles. Néanmoins, il nécessite un choix sur le paramètre β dont l'incidence sur la classification finale reste majeure. Le problème est de taille puisqu'il n'y a pas vraiment d'approche pour déterminer le choix idéal de ce paramètre.

5.3.1.6 La distance de McQuitty

Louis L.Mc Quitty 1966[32], La distance de Mc Quitty consiste à calculer la moyenne des moyennes des moyennes des .... des moyennes des distances entre les observations des deux groupes à fusionner [32].

Par exemple, soit le cluster regroupant les observations 1,4 au niveau $t_1$de la classification puis l'observation 2 avec ce groupe au niveau t_2. Au niveau t_3, La distance entre une observation 5et le groupe des observations 1,2,4 sera la distance entre 5 et 2 additionnée à la somme divisée par deux des distances entre 5 et 1 et 5 et 4 , le tout divisé à son tour par deux.

La formule itérative et générale est alors,

$$D_{Cl_{K1},Cl_{K2}\cup Cl_{K3}} = \frac{D_{Cl_{K1},Cl_{K2}} + D_{Cl_{K1},Cl_{K3}}}{2}$$

- ✓ Propriétés : La distance de Mc Quitty se base sur une approche combinatoire des groupes plutôt que sur les observations isolées dans ces groupes.

5.4 Autres algorithmes

5.4.1 Les algorithmes basés sur la grille

Le principe de cette famille d'algorithmes est qu'on divise l'espace de données en cellules (fusion des régions jugées denses, selon la valeur d'un seuil fixé à priori, les régions jugées peu denses permettent d'établir des frontières.). Donc ce type d'algorithme est conçu pour des données spatiales. Une cellule peut être un cube, une région ou un hyper rectangle. En fait, elle est un produit cartésien de sous intervalles d'attributs de données.
Avec une telle représentation des données, au lieu de faire la classification dans l'espace de données, on la fait dans l'espace spatial en utilisant des informations statistiques des objets dans la cellule. Les algorithmes les plus connus dans cette catégorie sont STING, CLIQUE et WaveCluster [23] et [15].

5.4.2 Les algorithmes basés sur les statistiques

Ces algorithmes démarrent du principe que les données de départ ont été générées suivant une certaine loi de distribution (normale, uniforme, exponentielle,...)
Le principe général est d'utiliser le modèle statistique permettant d'approcher au mieux cette distribution pour réaliser le groupement.
Parmi les systèmes de classification basés sur des modèles statistiques, on peut trouver COBWEB (Fisher), CLASSIT (Gennari, Langley et Fisher) et AutoCass (Cheeseman et Stutz) [21].

6 Conclusion

Le choix d'un algorithme approprié dépend fortement du contexte de son application, la nature des données et les ressources disponibles. Une analyse attentive des données aide à bien choisir le meilleur algorithme à partir du moment qu'il n'existe pas un algorithme qui peut répondre à toutes les demandes.

Chapitre 3 : La conception

Introduction

Dans ce chapitre, nous commençons en premier lieu par une spécification des besoins auxquels doit répondre l'application, passant ensuite à l'analyse de ces besoins à travers l'introduction des acteurs et les diagrammes de cas d'utilisation relatifs à ces acteurs. Ensuit nous présentons l'architecteur de notre système.

Première partie : Analyse des besoin et spécification

Dans cette partie , nous présenterons les objectifs de notre application, ce qui nous amène à identifier les possibilités du système et les besoins des utilisateurs que nous essayerons de les projeter dans des diagramme de cas d'utilisations détaillés.

1 Analyse et spécification des besoins

L'analyse et la spécification des besoins représentent la première phase du cycle de développement d'un logiciel. Elle sert à identifier les acteurs réactifs du système et leur associer chacun l'ensemble d'actions avec lesquelles il intervient dans l'objectif de donner un résultat optimal et satisfaisant au client.

1.1 Spécification des besoins

1.1.1 Les besoins fonctionnels

Les besoins fonctionnels ou besoin métiers représentent les actions que le système doit exécuter, il ne devient opérationnel que s'il les satisfait.

Notre système doit couvrir principalement les besoins fonctionnels suivants :

Inscription à la plateforme : L'application offre aux les employés et les transporteur un espace pour leurs inscriptions.

Marquage et enregistrement de la position de lieux d'habitation et lieux de travail : L'application grâce à Google Maps api doit permettre à l'utilisateur d'enregistrer les coordonnées de la position d'habitation et la position de travail.

Consultation les employés et les transporteurs : L'application offre à l'administrateur une vue complet sur l'états des employés et des transporteurs

Représentation plus réaliste des données : grâce à Google Maps l'application fait une représentation des postions des employés et des lignes routières sur une carte géographique.

Recherche efficace sur les employés et les lignes routières :

L'application offre un algorithme qui a le but à trouve les groupes des employés et leurs trajets correspondants après elle représente ces données sur le map.

1.1.2 Les besoins non fonctionnels

Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application se résument dans les points suivants :

Ergonomie et souplesse : L'application développée doit offrir une consultation confortable même pour des supports différents.

L'utilisateur peut ainsi consulter la même application web à travers une large gamme d'appareils (moniteurs d'ordinateur, ordiphones (smartphones en anglais), tablettes,) avec le même confort visuel et sans avoir recours au défilement horizontal ou au zoom avant/arrière sur les appareils tactiles notamment.

Rapidité : L'application doit optimiser la durée d'exécution des traitements pour avoir un temps de génération de schéma raisonnable.

Efficacité : L'application doit être fonctionnelle indépendamment de toutes circonstances pouvant entourer l'utilisateur, elle répond à toutes les exigences des utilisateurs d'une manière optimale.

Maintenabilité et scalabilité : Le code de l'application doit être lisible et compréhensible afin d'assurer son état évolutif et extensible par rapport aux besoins du marché.

1.2 Les résultats attendus

Par notre application sont :

Se déroule convenablement, conserve ces fonctionnalités dans le système, et améliore s'il est possible les performances du système ainsi que les bases des données existantes.

1.3 Les diagrammes des cas d'utilisation

Montre les interactions fonctionnelles entre les acteurs et le système à l'étude

1.3.1 Identification des acteurs

Un acteur représente un rôle joué par une personne qui interagit avec le système. Par définition, les acteurs sont à l'extérieur du système. Les acteurs se recrutent parmi les utilisateurs du système et aussi parmi les responsables de sa configuration et de sa maintenance. D'où, les acteurs potentiels qui risquent d'interagir avec l'application sont :

Les clients (les employées, les transporteurs) : c'est le demandeur qui va s'inscrire et donner ces informations.

L'administrateur : C'est le gérant de l'application, il a une visibilité totale sur les bases de données. Il a pour tâches de gérer tout le système.

1.4 Les différents cas d'utilisation

L'étude de cas d'utilisation a pour objectif de déterminer ce que chaque utilisateur attend du système. La détermination du besoin est basée sur la représentation de l'interaction entre l'acteur et le système.

Nous avons regroupé l'ensemble de cas d'utilisation dans un seul diagramme qui figure ci-dessous :

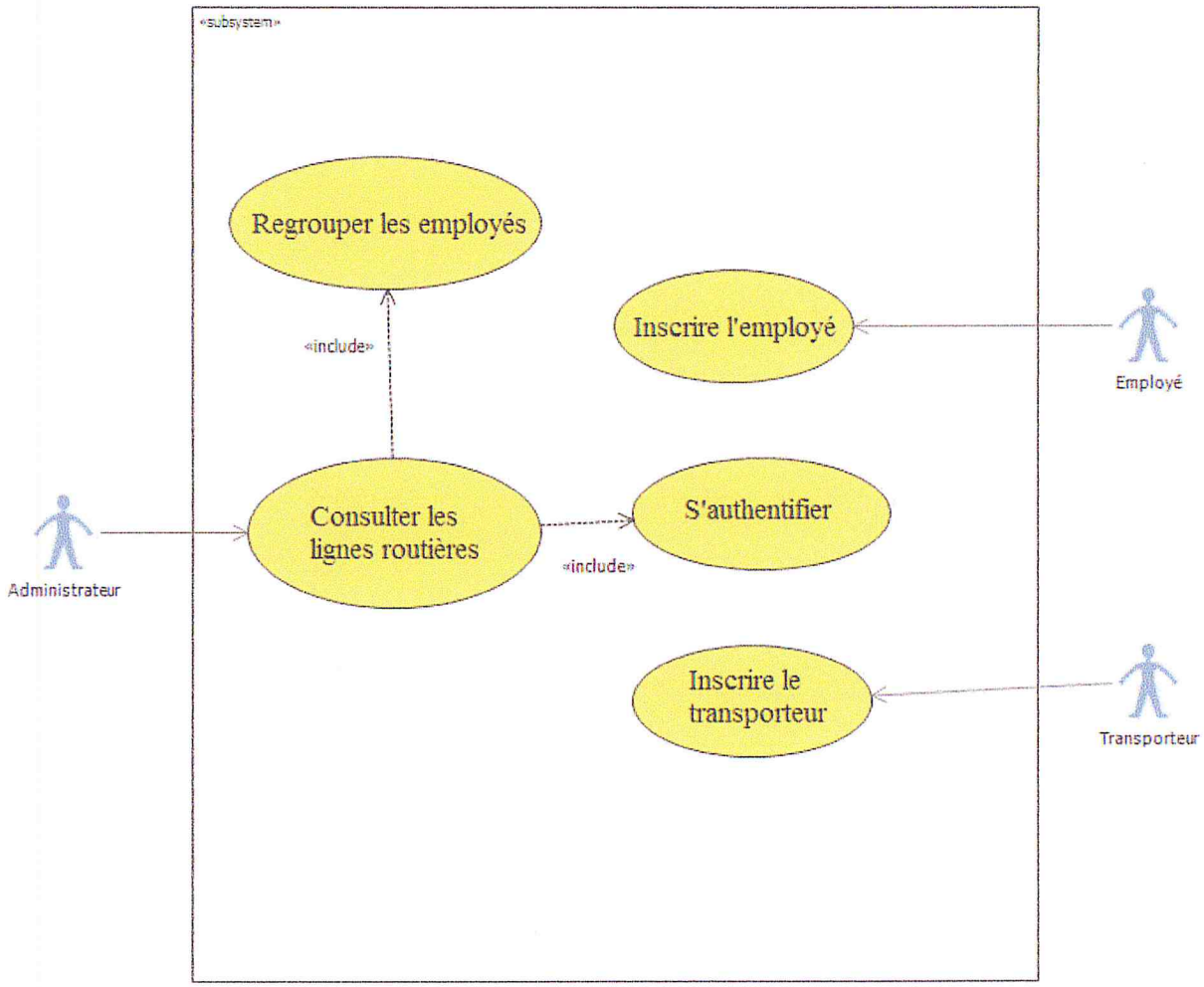

Figure 3.1 Diagramme de cas d'utilisation

Après avoir recensé l'ensemble des besoins, nous avons distingué les différents acteurs suivants :

1. **Les employés** : les salariés qui veulent assurer leurs déplacements quotidiens entre lieux de travail et de lieux d'habitation. Ils s'inscrivent à notre plate-forme et laissent leurs informations comme l'adresse d'habitation et de travail et leurs horaires de travail. Ainsi ils peuvent consulter leurs situations après l'authentification.

2. **Les transporteurs** : Ce sont les personnes qui veulent se transporter les travailleurs, ils utilisent notre plate-forme pour l'inscription. Ils permettent de consulter leurs situations après l'authentification.

3. **L'administrateur** : la personne qui a tous les droits sur l'application, il faut s'authentifier pour accéder à l'application, il permet de consulter les lignes de transport qui marchent actuellement ou le regroupement des employés, comme il peut activer ou désactiver des autres lignes.

Deuxième partie : L'architecture multi-agents dans notre approche

Dans cette partie on a déterminé l'architecteur de notre système et on détaille l'algorithme qui nous proposons

1 Notre approche

Dans le cadre de ce mémoire, le problème considéré est un problème multicritère. Nous avons proposé une approche non agrégative pour résoudre ce problème.
Cette approche est basée sur l'algorithme de CHA (classification ascendant hiérarchique) et un algorithme proposé par nous-mêmes.

L'approche basée sur un système composé de deux classes. Le premier est client qui envoie une requête à l'administrateur. Ce dernier exécute un processus basé sur les deux algorithmes

Ce processus est exécuté à chaque fois que le « administrateur » reçoit une demande du client.

Ce processus consiste à trouver une solution pour le PTD multicritère ayant les objectifs suivants : la durée du trajet, la distance parcourue et le nombre de stations visitées.

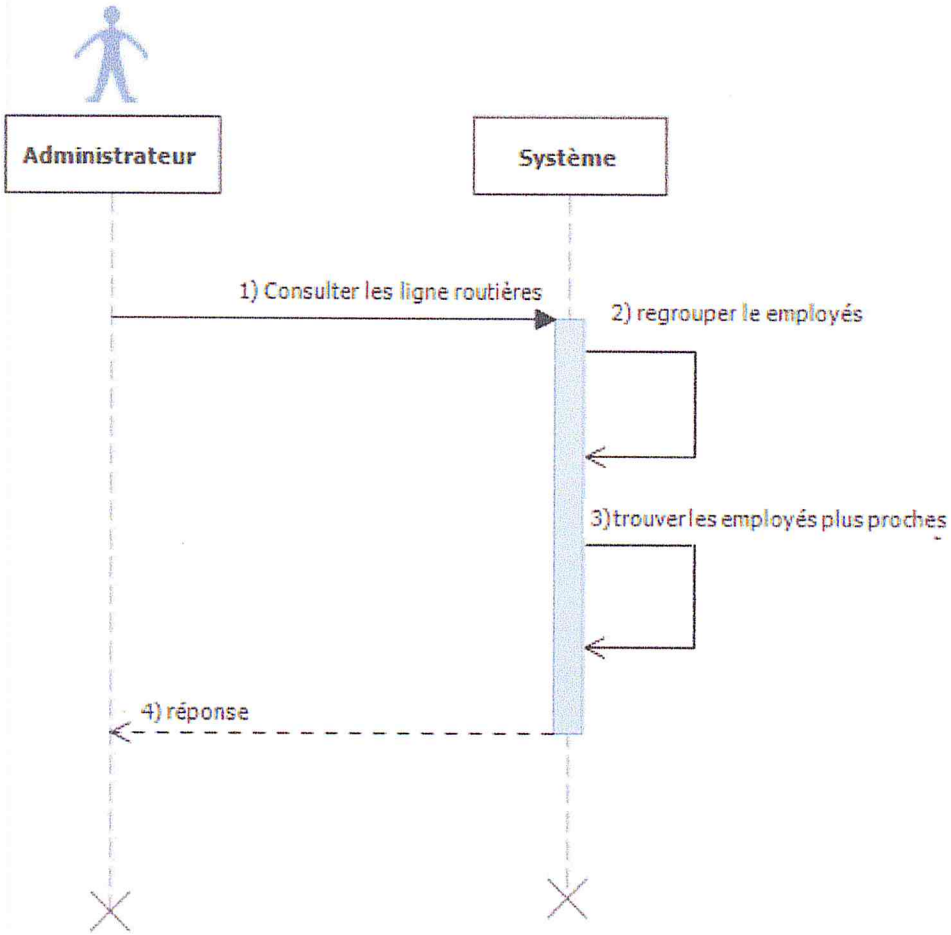

Figure 3.3 Le scénario de la consultation les lignes routières

Nous détaillons ici, les différentes étapes du scénario de la figure 13 présentant la consultation les lignes routières :

(1) L'étape où l'administrateur envoi une requête caractérisée par les distances de départ et d'arrivé et le maximum distance entre les employés et la route.

(2) Le système exécuter l'algorithme de CHA pour regrouper les employés en cluster par rapport aux distances de départ et d'arrivé .

(3) Le system exécuter l'algorithme proposé pour trouver les employés qui sont proche de nos routes.

(4) Le system envoyer une réponse à l'administrateur contient toutes les routes trouvées sur un maps.

1.1 Module administrateur

Dans la figure 3.5, nous présentons le comportement de l'Administrateur.

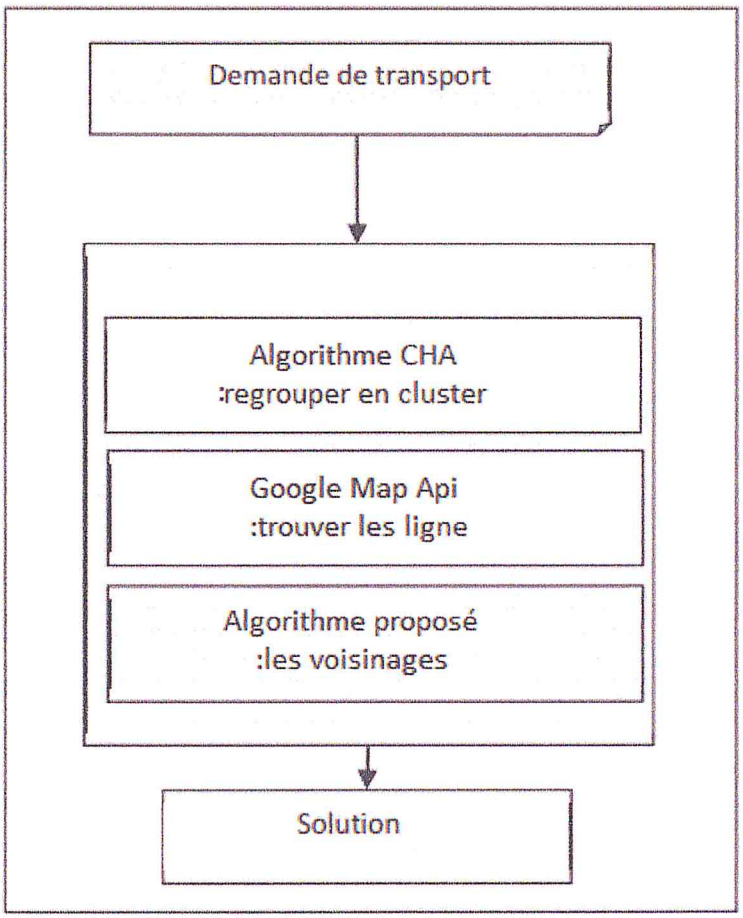

Figure 3.4 module administrateur

La construction d'une solution pour le PTD se fait de la manière suivante : A chaque fois qu'une demande de transport arrive, le module administrateur exécute les deux algorithmes.

1.1.1 Algorithme CHA

Le premier problème et de trouver un algorithme permettant de parcourir toutes les coordonnées géographiques des utilisateurs, l'algorithme prendre en entrée des coordonnées géographiques, et donne en sortie un ensemble des points.

Pour résoudre ce problème on applique l'algorithme de CHA (Classification Hiérarchique Ascendante) de data miging, L'algorithme est présenté en détails dans le chapitre 2.

L'algorithme :

Initialement :
- Chaque employée est considérée comme un cluster
- Calculer la matrice de distance M

Tantque taille(M) > 0 :
- Sélectionner les deux clusters les plus proche C1,C2
- Si Distance (C1,C2) <= 2 (Departeur_barry_center - Arrival_barry_center):
  - Alors
    * Fusioner C1 et C 2 par un cluster générale C3
  - Sinon
    * Retirer le cluster qui contient maximums des employée
- Mise à jour de la matrice de distance

Nous avons adopté l'algorithme CHA selon nos besoins, comme suit :

1. Calculer la distance :

   On utilise la formule de vincenty pour calculer la distance géographique entre deux points.

   Distance entre A, B = distance d'arrive entre A, B + distance de départ entre A, B

```
# comment: méthode pour calculet la distance d'arrivée entre deux cluster
def arrival_distance(self, clusterA, clusterB):
    return vincenty((clusterA.arrive_coordinate_x, clusterA.arrive_coordinate_y),
                    (clusterB.arrive_coordinate_x, clusterB.arrive_coordinate_y)).km
# comment: méthode pour calculet la distance de départ entre deux cluster
def departure_distance(self, clusterA, clusterB):
    return vincenty((clusterA.depart_coordinate_x, clusterA.depart_coordinate_y),
                    (clusterB.depart_coordinate_x, clusterB.depart_coordinate_y)).km
```

2. Rapprochement des clusters :

   On utilise le lien moyen pour calculer la distance entre deux clusters.

   Distance (C1, C2) = distance (M1, M2) tel que M1, M2 est le centroide de C1, C2 respectivement

3. Les contraints :

   La distance entre deux points (employées) ne dépasse pas certaine valeur

   Par exemple

   - La distance de départe entre deux employée doit être inferieur ou égale 2 km
   - La distance d'arrivée entre deux employée doit être inferieur ou égale à 1km

   Pour cela on fixe un barré centre de départ et barré center d'arrivée pour chaque cluster

   Donc on met une condition pour vérification

   Distance (C1, C2) <= 2 (départeur_barry_center + arrival_barry_center)

1.1.2 Algorithme 2 « proposer »

1.1.2.1 Présentation générale

Le but de notre algorithme est de trouver les meilleures routes qui rassemblent le maximum des employées.

Apres l'exécution de l'algorithme précédent (CHA) et trouvé les clusters qui contient les employés, nous utilisons cette résultat pour trouve les meilleures routes.

L'aidé de cet algorithme est de trouver la route d'un cluster précis à partir de Google Maps, ensuite nous utilisons Google Maps également pour trouver les points qui construit la route.

A partir de ces résultats, nous construisons des cercles de diamètre donnée.

Nous pouvons dire que les employés qui appartiennent à ces cercles sont confédérés comme des bons employés et on peut les prendre.

Contraintes :

- Les employés doivent être à distance précis d'une route pour cela nous avons fixe un paramètre 'max_distance'

Algorithme: 2 "proposé"

- On trie les clusters descendent para port à la distance de départ à l'arrivée.

For cluster in clusters :

- On utilise api de Google Map pour trouver la bonne route de « cluster »

- On utilise api de Google Map pour décoder cette route et obtenir les points de cette route

- A partir de ces résultats on construire des cercles de diamètre 'max_distance',
  on trouve des cercles qui a des centres appartient de cette route
- A partir de la fusion de ces cercles on construirt un polygone global

For cluster_voisin in clusters :

§ For employé in cluster_voisin.employees :

- Si employé appartient à polygone

Alors :

On teste la direction de cette employée pour assurer que l'employé est dans le sens correct

Trouve = faux

For cercle in cercles :

§ Si employé .depart_coordinate appartient à cercle

Alors :

Trouve =vrais

§ Si emplyee.arrive_coordinate appartient à cercle et Trouve vrais :

Alors :

- Cette employé est dans le sens correct donc on rend cette employé
- On exclue cette emplyé de cluster_voisin

1.1.2.2 La présentation détaille de l'algorithme proposé

D'après le résultat de l'algorithme CHA, on prend un cluster comme échantillon,

Le cluster contient des employés qui ont le même point de départ et d'arrivé.

L'algorithme proposé sert à rassembler tous les autres employés qui sont proche de la route. Et qui ne sont pas appartiennent du cluster précédant.

Chaque employé a un point de départ et un point d'arrivé, on met une direction pour chaque employé de point de départ à point d'arrivé

Etape 1 :

Pour trouver la meilleure route de cette cluster on utilise Google Maps api comme suit :

Route = gmaps.directions(origin= point de départ,

Destination = point d'arrivé,

Mode = driving)

On obtient le diagramme suivant :

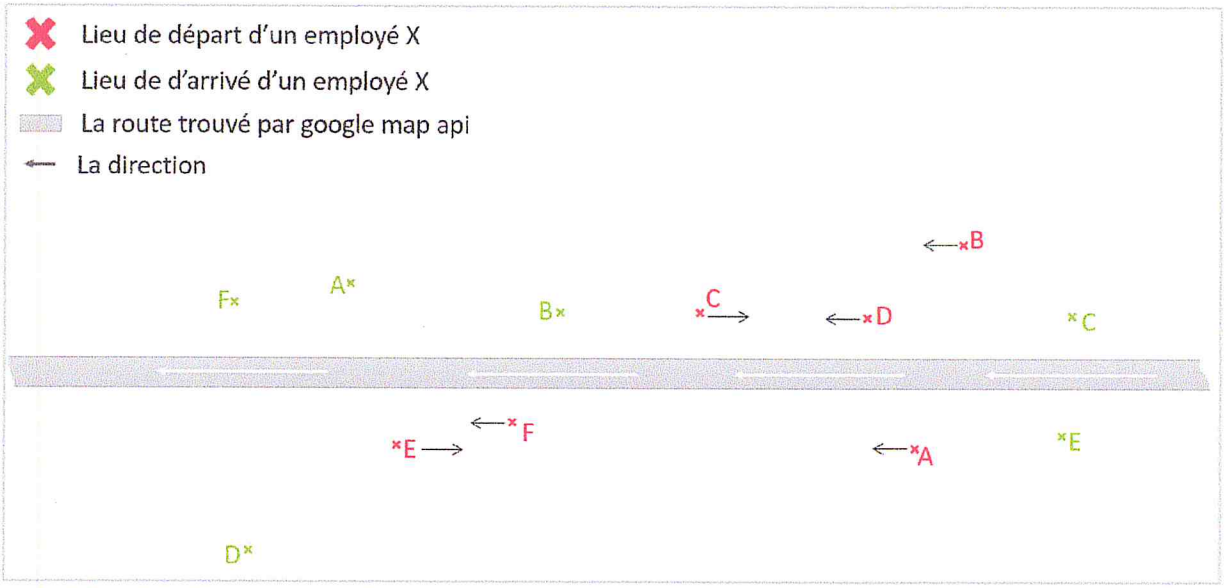

Figure 3.5résultat d'étape 1

Etape 2 :

A partir de Google Maps api on décode cette route et on obtient des points comme nous montre la figure suivant :

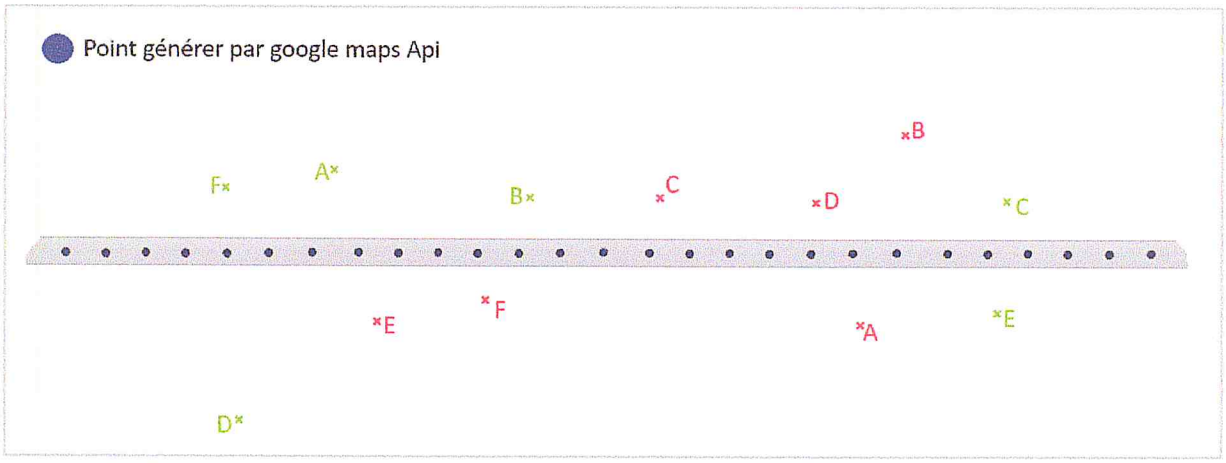

Figure 3.6 résultat d'étape 2

Etape 3 :

Pour chaque point On construit des cercles de 1 km de diamètre

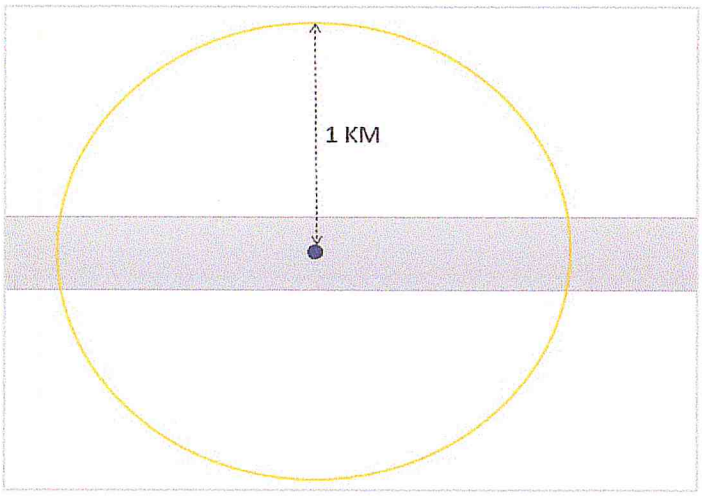

Figure 3.7 résultat 1 d'étape 3

Le résultat final comme suit :

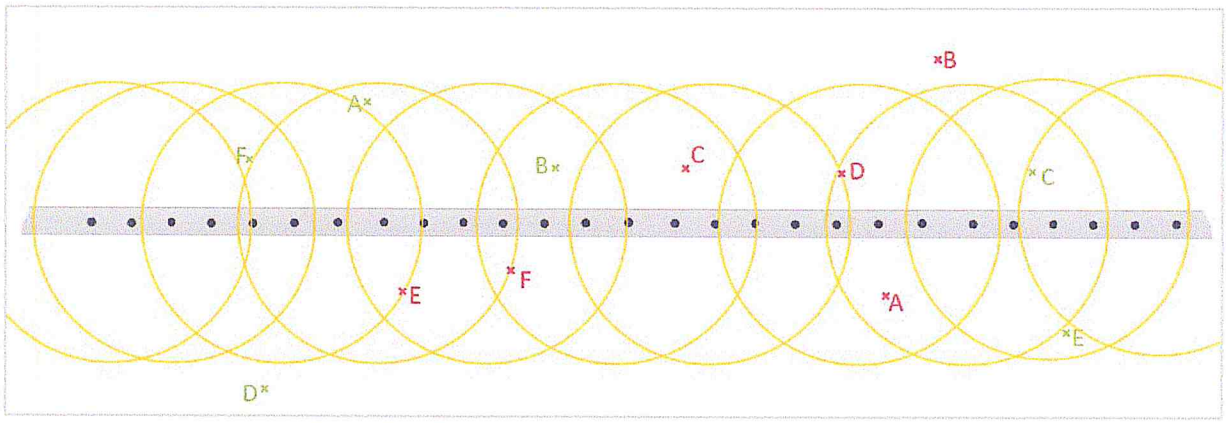

Figure 3.8 résultat final d'étape 3

Etape 4 :

L'unir de ces cercles donne un rectangle de largeur presque de 2 km

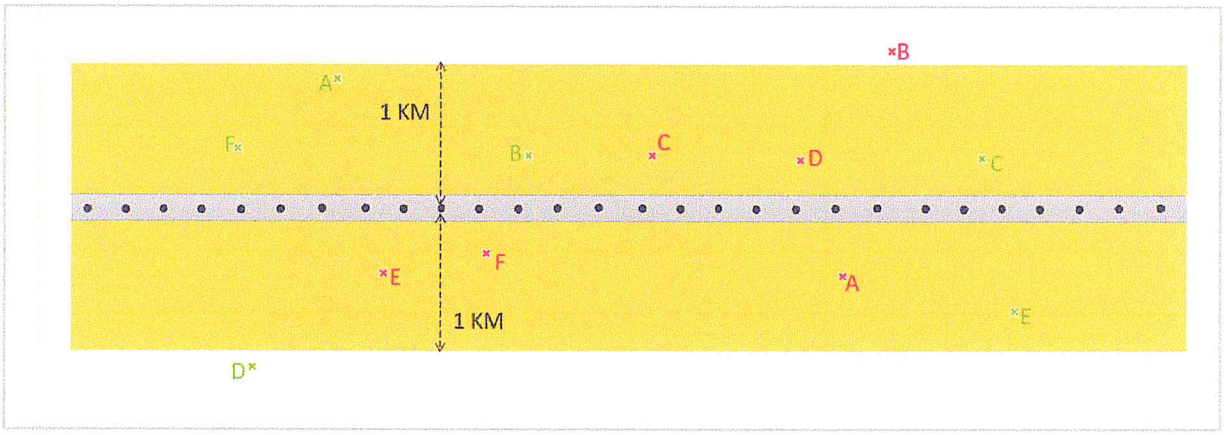

Figure 3.9 résultatd'étape 4

Etape 5 :

1- Les points qui n'appartiennent pas à ce rectangle ont une distance supérieure à 1 km de la route, dans nous n'acceptons pas ces employés. Comme les employés B, D.
2- Les employés qui ont une direction différente de la direction de la route nous les n'acceptons pas.
Comme les employés E, C

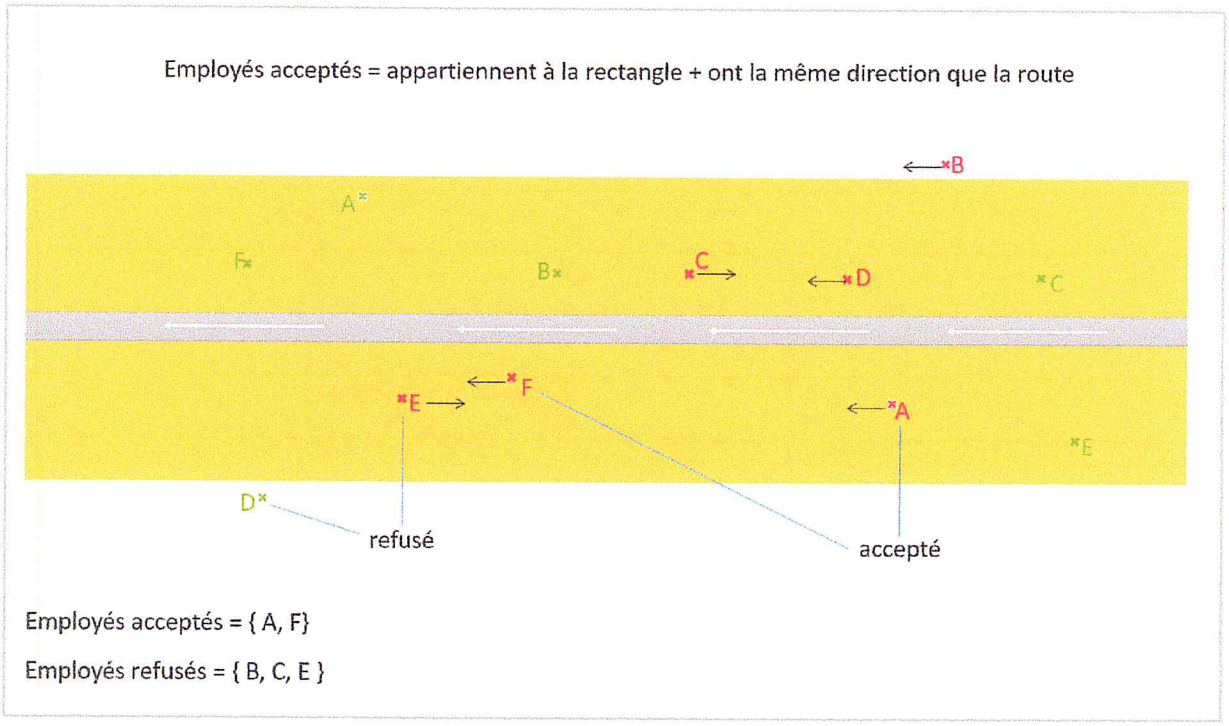

Figure 3.10 résultat d'étape 5

2 Conclusion

Dans ce chapitre, nous avons présenté la conception en deux parties, la première partie été consacré pour la spécification des besoins fonctionnels et non fonctionnels du système résultant, et la deuxième partie, nous avons présenté notre approche basée sur l'algorithme de CHA et l'algorithme propose pour la résolution du PTD.

Chapitre 4 : La réalisation

1 Introduction

Dans ce chapitre nous allons exposer le travail achevé. Nous présenterons en premier lieu l'environnement matériel et logiciel dans les quels notre application a été développée en indiquant les technologies utilisées et les ensembles qui le composent, nous allons présenter dans ce chapitre la réalisation de l'application Linkibus.

2 Environnement de travail

2.1 Environnement matériel

Nous avons élaboré ce travail sur un PC dont la configuration est la suivante :

- ✓ **Processeur** : Intel(R) Core (TM) i5-4005U CPU @ 1.70GHz Cadencé à 1.70 GHz
- ✓ **Disque dur** : 500 GO
- ✓ **RAM**:4 GO

2.2 Environnement logiciel

- ✓ System exploitation **:** Linux Mint 18 Sarah, Architecture : 64 bit
- ✓ Pycharm 2017.1.4 IDE, il est développé par l'entreprise tchèque JetBrains.
- ✓ Pour développer le côté serveur (backend) **:** il supporte python 2.7 et flask framework
- ✓ Visual Studio Code 1.13.1est un IDE développé par Microsoft.
- ✓ Pour développer le côté client (fontend) **:** il supporte Javascript ES6 et angularjs framework.

Qu'est-ce que Web Framework ?

Cadre d'application Web ou simplement Web Framework représente une collection de bibliothèques et de modules qui permet à un développeur d'applications web pour écrire des applications sans avoir à se soucier des détails de bas niveau tels que les protocoles, la gestion des threads, etc. [35].

3 Mise en place de la partie Front-end

Le JavaScript est utilisé depuis la nuit des temps, souvent sous une de ses formes les plus connues : les librairies (ex : jQuery).

Aujourd'hui le JavaScript a lui aussi le droit d'avoir sa palette de framework, les plus populaires étant : Backbone.js, Ember.js, Knockout.js et AngularJS.

Pour réaliser notre application web de côté client en JavaScript devient de plus en plus complexe par leur taille, par leur fonctionnement et par leur coût de développement si pour ça nous avons choisi le Framework Angularjs.

L'architecture de la partie client d'une application est souvent délaissée voire non prise en compte, alors que l'architecture côté serveur attire toute l'attention. Utiliser AngularJS, c'est un moyen d'avoir une architecture de qualité pour un faible coût. Avec lui, vous gagnerez en maintenabilité et en productivité, le tout sans négliger l'expérience utilisateur.

3.1 Définition

AngularJS est un framework JavaScript libre et open-source développé par Google. Il a pour but d'ajouter une couche d'abstraction de JavaScript, et de combler les faiblesses de ce dernier en lui ajoutant de nouvelles fonctionnalités. AngularJS est fondé sur la philosophie de simplifier la programmation. Le framework adapte et étend le HTML traditionnel pour servir le contenu dynamique de façon améliorée grâce à un data-binding bidirectionnel permettant la synchronisation automatique des modèles et des vues. En conséquence, AngularJS minore l'importance des manipulations DOM et améliore la maintenance / testabilité du code. AngularJs apporte aux applications web côté client les services traditionnellement apportés côté serveur, comme les contrôleurs de vues. En conséquence, une bonne partie du fardeau supporté par le back-end est supprimée, ce qui conduit à des applications web beaucoup plus légères.

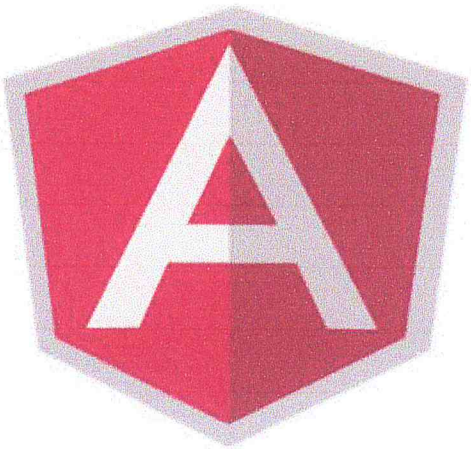

Figure 4.1 logo de AngularJs framework

3.2 Objectifs de conception du framework :

- Découpler les manipulations du DOM de la logique métier. Cela améliore la testabilité du code.
- Considérer le test d'une application aussi important que l'écriture de l'application elle-même. La difficulté de la phase de test est considérablement impactée par la façon dont le code est structuré.
- Découpler les côtés client et serveur d'une application. Cela permet au développement logiciel des côtés client et serveur de progresser en parallèle, et permet la réutilisabilité de chacun des côtés.
- Fournir un cadre afin de guider les développeurs pendant toute la durée de la construction d'une application : de la conception de l'interface utilisateur, en passant par l'écriture de la logique métier, jusqu'au test de l'application.
- Simplifier la manipulation des objets métiers dans le code HTML.

3.3 Pattern MVVMETANGULARJS

Le Modèle-Vue-VueModèle (en abrégé MVVM), est une architecture et une méthode de conception utilisée dans le génie logiciel. MVVM est originaire de Microsoft et adapté pour le développement des applications basées sur les technologies Windows Presentation Foundation et Silverlight via l'outil MVVM Light par exemple. Cette méthode permet, tel le modèle MVC (Modèle-Vue-Contrôleur), de séparer la vue de la logique et de l'accès aux données en accentuant les principes de binding et d'événement [36].
La Vue reçoit toujours les actions de l'utilisateur et interagit seulement avec le ViewModel. Le Modèle communique avec le serveur et notifie le ViewModel de son changement [37].
Le ViewModel s'occupe de :

- Présenter les données du Model à la Vue.
- Recevoir les changements de données de la Vue.
- Demander au Model de se modifier.

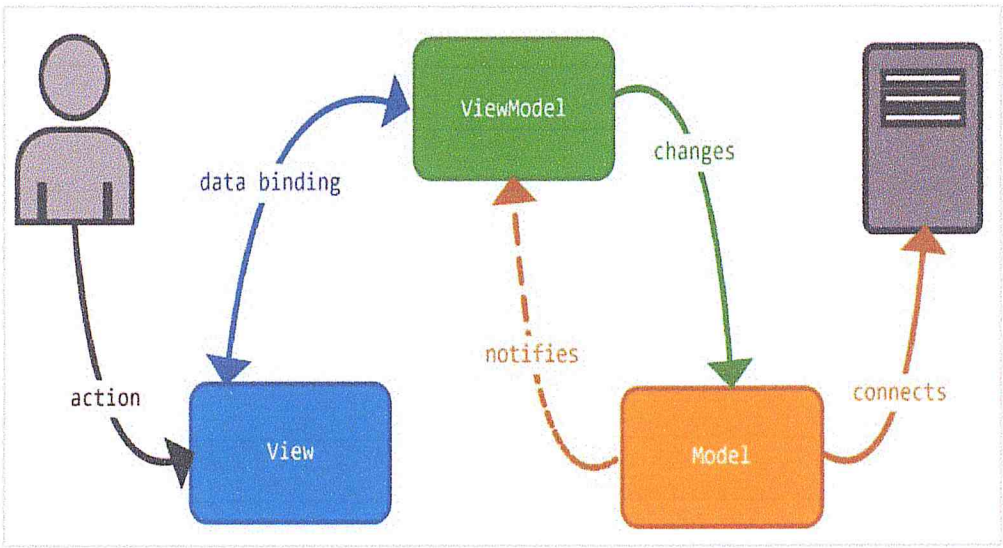

Figure 4.2 L'architecteur MVVM [37]

3.4 Data Binding

La Vue n'a donc plus aucun lien avec le Model. Ainsi le ViewModel s'occupe entièrement du cycle de modification de ce dernier. Il réalise à la fois la réception et l'envoi des données à la Vue. On parle alors de « data binding ». Les informations affichées sont liées entre deux entités et mises à jour en temps réel.
Ce dernier mécanisme est la clef du pattern MVVM. Il nous permet de découpler les différentes parties de notre application en étant capable de la faire évoluer de manière modulaire [37].

3.5 Mise en application

Sur notre projet, nous avons mis en place de multiples services REST qui communiquent par défaut au format JSON. Ces services permettent d'obtenir diverses données de référentiels, mais aussi de manipuler des objets métiers via des opérations (CRUD).

Ces services pourront être appelé par différents projets et en particulier par notre interface Web écrite en AngularJS. Nous avons aussi défini un modèle de données côté serveur qui permet à toutes les applications appelant les services de parler le même « langage ».

Sur le projet, AngularJS nous a permis de gagner en productivité car il y a beaucoup moins de code à écrire par rapport à du Javascript standard ou jQuery. Voici quelques exemples :

- Les formulaires sont liés au modèle de données. Une réponse POST est directement interprétée par le service adéquate en récupérant l'objet métier directement sans avoir besoin d'utiliser des méthodes GET ou encore en récupérant un ID dans la réponse et en rechargeant l'objet métier via une requête en base de données. AngularJS étant bidirectionnel ceci est vrai dans le sens inverse. Un formulaire de modification pourra être peuplé automatiquement.

- Le code est directement inséré dans la page HTML. C'est possible avec Javascript standard ou encore jQuery, mais avec des bibliothèques immenses et difficilement maintenable par la suite. Avec AngularJS, vous déclarez simplement vos objets (tirés du modèle de données) dans le code HTML et c'est fini !

- Le code de gestion des appels des Web Services est simple, court et efficace. Il n'y a plus besoin de gérer les multiples « browsers » et les fonctions de retour.

4 Mise en place de la partie Back-end

4.1 Qu'est-ce que Flask ?

Flask est un cadre d'application web écrit en Python. Il est développé par Armin Ronacher, qui dirige un groupe international d'amateurs Python nommé Pocco. Flask est basé sur la boîte à outils Werkzeug WSGI et moteur de template Jinja2. Les deux sont des projets Pocco.

Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates. Il est distribué sous licence BSD2[38].

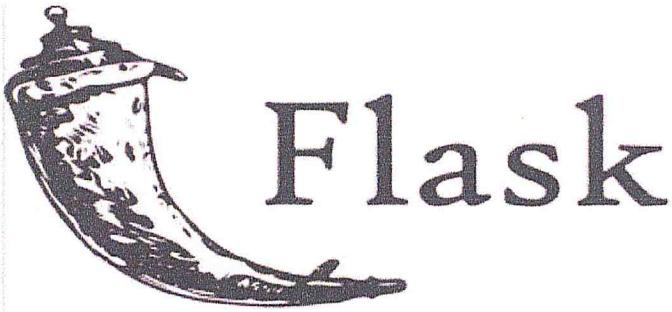

Figure 4.3 logo de la framework Flask

4.2 Fonctionnalités du flask

Falsk est un framework très riche, il a les spécifications suivant :

- Contient un serveur de développement et un débuggeur
- Supporte les tests unitaires
- Utilise le moteur de template Jinja2
- Supporte les cookies sécurisés (session)
- Entièrement compatible avec WSGI 1.0
- Se base sur l'Unicode
- Dispose d'une documentation complète
- Compatible avec Google App Engine
- Il est possible de créer des extensions

4.3 WSGI

Web Server Gateway Interface (de WSGI) a été adopté comme un standard pour le développement d'applications web Python. WSGI est une spécification d'une interface universelle entre le serveur Web et les applications web [35].

4.4 Werkzeug

Il est une boîte à outils de WSGI, qui met en œuvre des demandes, des objets de réponse, et d'autres fonctions d'utilité. Cela permet la construction d'un cadre sur le dessus de celui-web. Le cadre Flasque utilise Werkzeug comme une de ses bases [35].

4.5 Jinja2

Jinja2 est un moteur de template populaire pour Python. Un système web de texturation combine un modèle avec une source de données certaine pour rendre les pages Web dynamiques [35].

5 Google Maps Api :

Google Maps est un service gratuit de carte géographique et de plan en ligne. Le service a été créé par Google. Il s'agit d'une forme de géo portail.

Cette api permet de localiser tout type de données sur une carte (routière, satellite, mixte) à partir de son adresse postale. Cet api s'avère très utile pour proposer aux internautes une vision globale et géographique de données (membre d'une communauté, restaurants d'un quartier...).
Il est également possible de calculer un itinéraire (piéton ou voiture) depuis un point de départ (par exemple, votre résidence) jusqu'à un point d'arrivée (votre recherche).

Pour notre application nous utilisons GoogleMaps api v3 qui permet de :

- Affichage d'une carte Google Maps
- Création d'overlays (marqueurs, polygones et polylignes)
- Gestion des évènements souris (click, rightclick, drag, ...etc.)

6 Présentation de l'application

6.1 Page d'accueil :

La page d'accueil de l'application i contient une représentation de Linkibus et des liens pour l'inscription et la connexion.

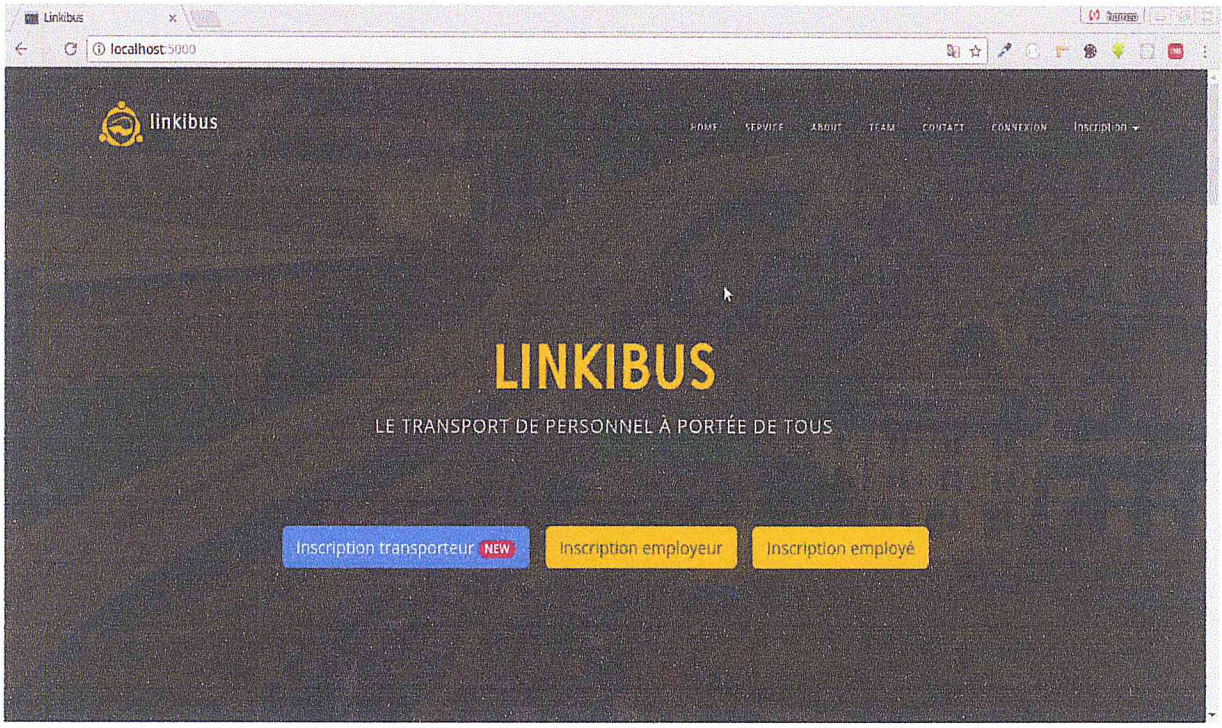

Figure 4.4 Page d'accueil

6.2 Page de connexion

Avec cette page l'admin, les employés, les transporteurs peuvent accéder à notre application.

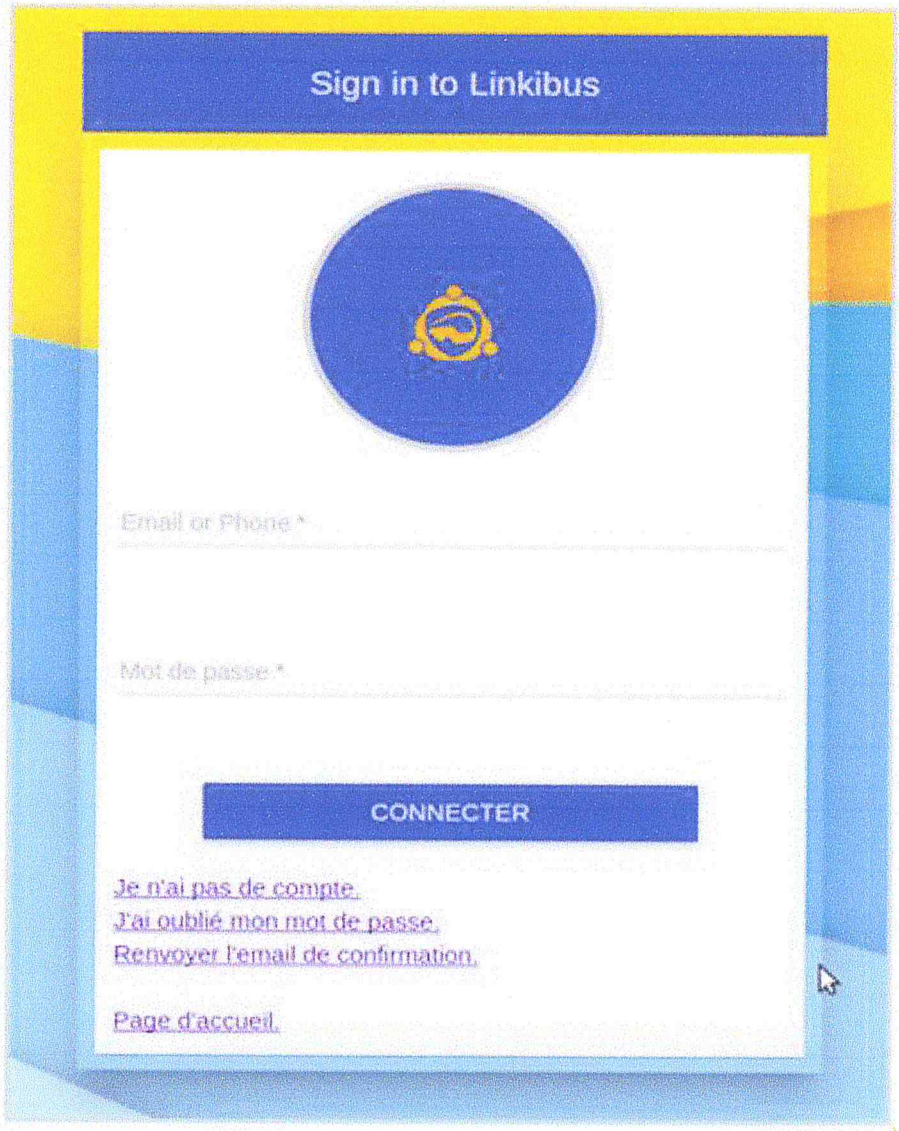

Figure 4.5 page de la connexion

6.3 Inscription des employés

Avec cette page les employés peuvent donner leurs informations.

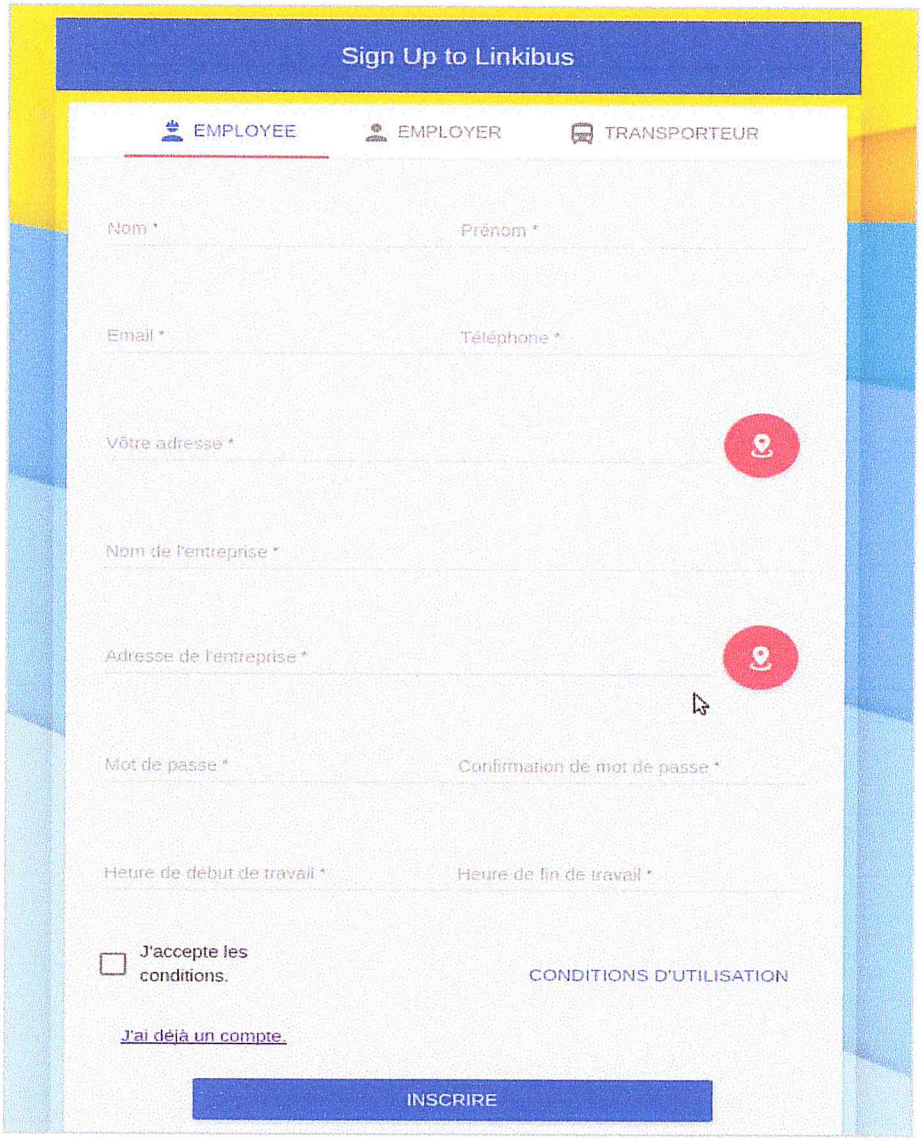

Figure 4.6 Page d'inscription des employés

6.4 Inscription des transporteurs

Avec cette fenêtre les transporteurs peuvent laisser leurs informations.

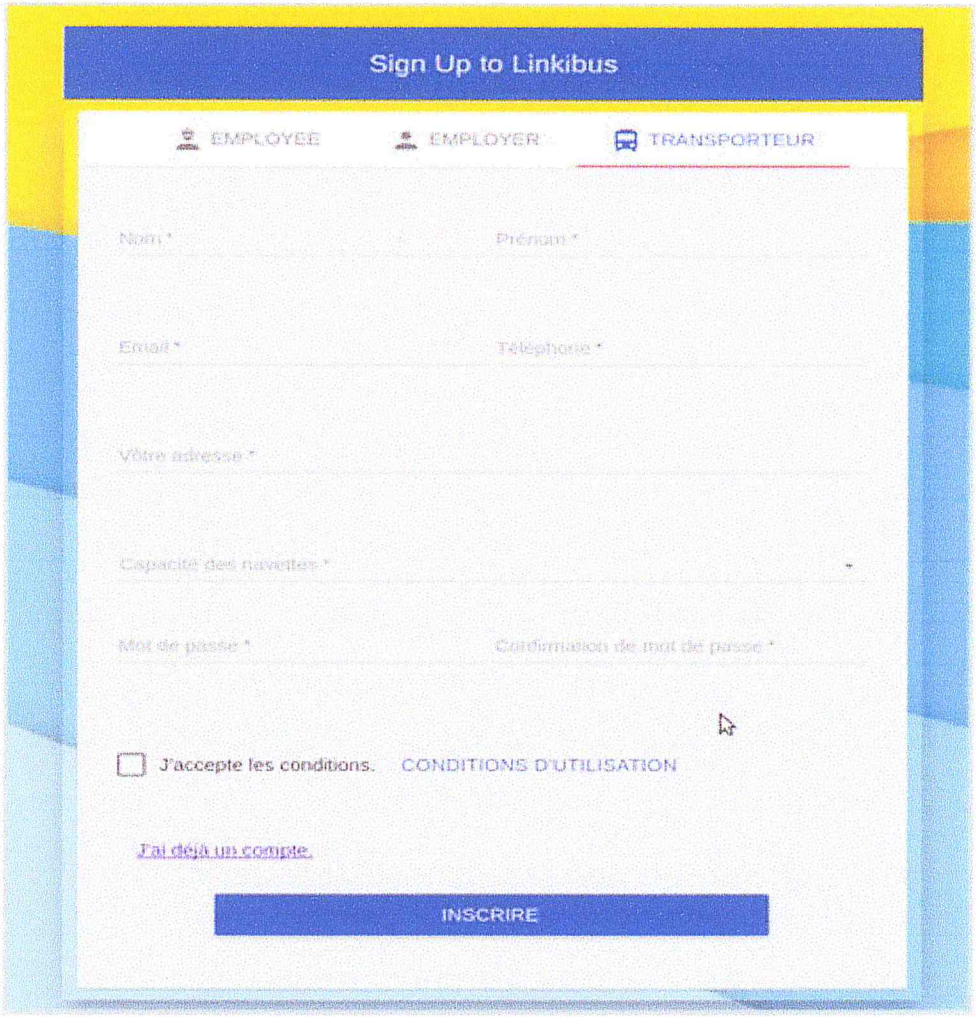

Figure 4.7 l'inscription des transporteurs

6.5 Consulter le regroupement des employés

Cette page représente les employés regroupés en clusters en deux positions départ ou d'arrivé, il a deux paramètres d'entrés

- La distance de départ : par default 1km
- La distance d'arrivé : par default 1km

Avec cette fenêtre admin peut chercher à des clusters à n'importe quelles distances ainsi il peut faire des filtres.

Cette fenêtre est une représentation de l'exécution de l'algorithme CHA.
Par exemple nous prenons :

- 2 km pour la distance de départ.
- 2 km pour la distance d'arrivé.
- 1460 employé qui s'inscrire sur notre application.

Et voilà le résultat au-dessus :

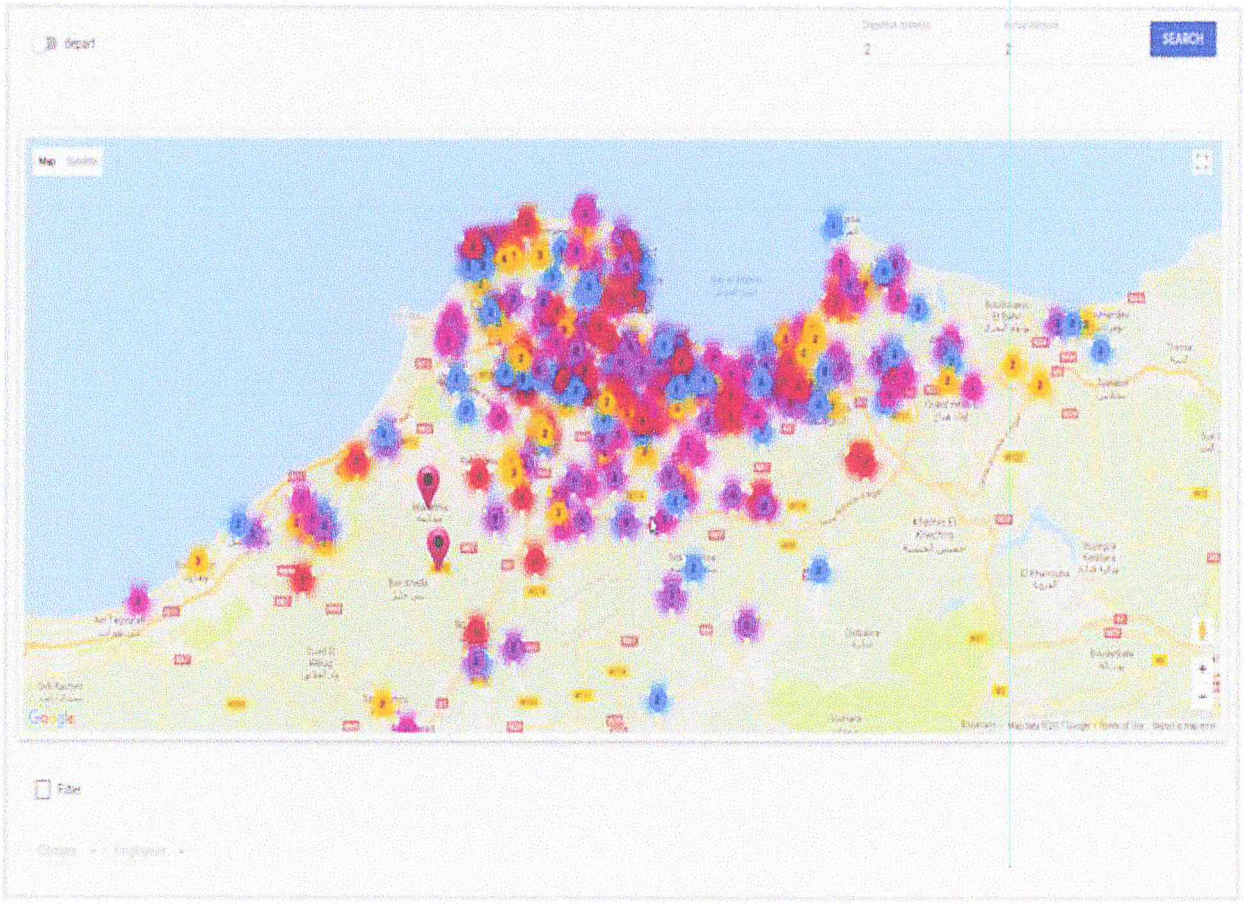

Figure 4.8 consulter les regroupements des employés

6.6 Consulter les lignes routières

Cette page affiche les routes qui on a trouvé l'aide à l'algorithme proposé.

L'admin peut consulter les routes avec leurs employés au même temps, ainsi il peut faire des recherches et des filtres.

Par exemple pour 1460 employés qui s'inscrire à l'application, et grâce à l'algorithme proposé nous obtenons une route de Hemmadi vers Alger centre et affiche les positions de départ des employés en rouge et l'arrivé en vert.

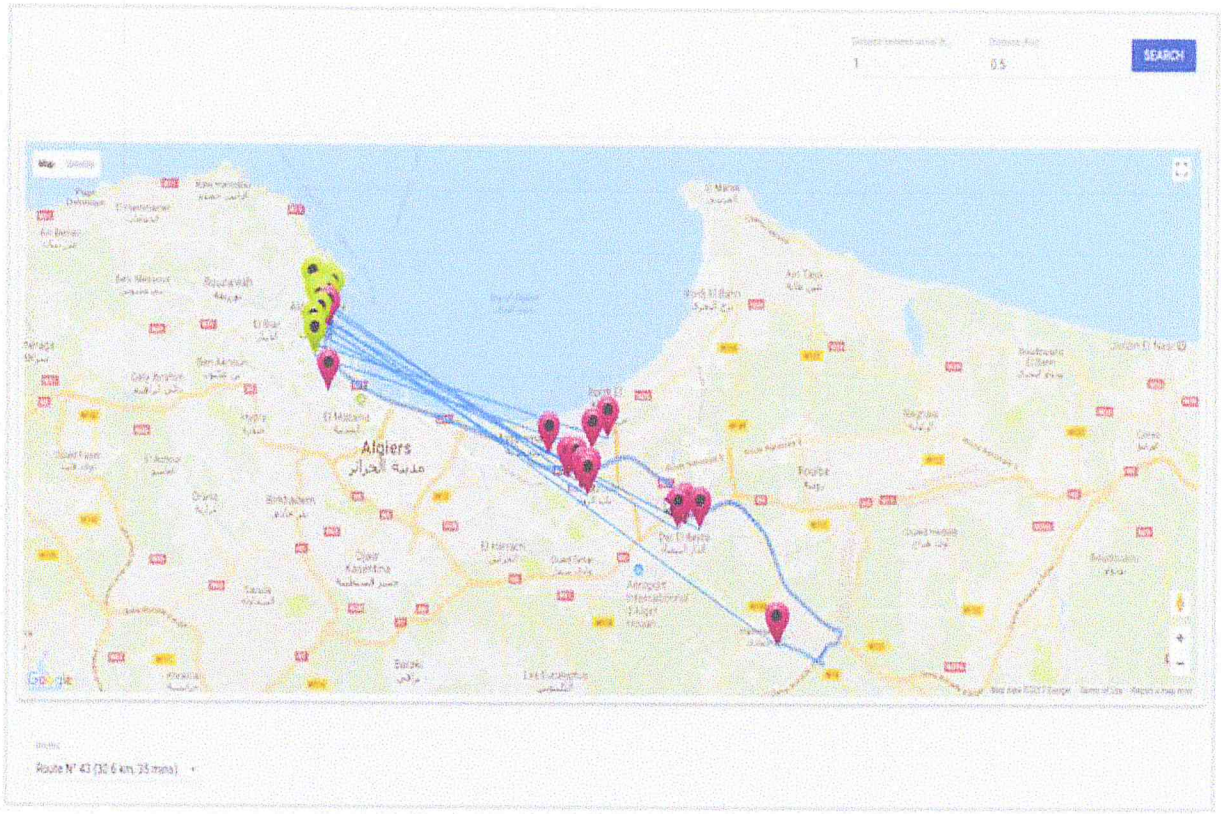

Figure 4.9 consulter les lignes routières

7 Conclusion

Dans ce chapitre nous avons présenté l'environnement matériel et logiciel utilisé dans la réalisation. Puis nous avons fourni quelques captures d'écrans décrivant des interfaces de notre application Linkibus.

Conclusion

Dans le cadre de ce mémoire, nous nous sommes intéressé aux problèmes de transport à la demande (TAD). Ces problèmes consistent à déterminer des tournées pour satisfaire les demandes de clients souhaitant être transportés depuis des lieux d'origine vers des lieux de destination. Une des difficultés de ce problème est de prendre en compte différentes contraintes, comme les fenêtres de temps et les temps de transport des clients.

Pour implanter des systèmes de transport à la demande, plusieurs projets ont été lancés partout dans le monde pour la conception de ce dernier.

Nous avons pu constater que les systèmes de transport à la demande existants ne respectent pas tous les objectifs fixés par la compagnie de transport et les voyageurs.

L'objectif de notre projet de fin d'étude était d'améliorer un algorithme pour la création dynamique de ligne de transport de personnel avec des contraint.

Le point de départ de la réalisation de ce projet était élaboration d'un état de l'art sur le problème de transport à la demande(TAD) et en détaillant la problématique du système de TAD, Ensuite nous rappelons les différents éléments qui composent ce problème, ainsi que les contraintes à satisfaire et les objectifs , Nous présentons également une étude bibliographique concernant les problèmes de TAD dans les cas statique et dynamique , dans notre travaille ont basé sur le cas dynamique où les demandes de transport arrivent peu à peu en temps réel.et nous présentons en générale les différentes techniques et différent algorithmes de la classification de data mining qui sont utilisés , notamment, pour la classification automatique de données(non supervisée).

En suit, cette partie décrit la partie conceptuelle. Nous commencerons par comprendre le contexte du système, déterminer les principaux cas d'utilisation, déterminer les besoins fonctionnels et les besoins non fonctionnels. Puis en présentant l'architecteur de notre système.

Le dernier volet de notre projet était la partie réalisation qui a été consacré aux tests et résultats où les algorithmes sont testés avec des données réels.

Références

[1] : Issam Zidi, « Modélisation et optimisation d'un système de transport a la demande »

Mémoire d'ingénieur, Ecole centrale de lille ,2012.

[2]: Psaraftis HN (1980) A dynamic programming approach to the single-vehicle, many-tomany immediate request dial-a-ride problem, Transportation Science 14:130–154

[3] : Psaraftis HN (1983) An exact algorithm for the single-vehicle many-to-many dial-a-ride problem with time windows, Transportation Science 17:351–357

[4] : Sexton T (1979) The single vehicle many-to-many routing and scheduling problem, Ph.D. dissertation, SUNY at Stony Brook

[5] Sexton T, Bodin LD (1985a) Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling, Transportation Science 19:378–410

[6] Sexton T, Bodin LD (1985b) Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing, Transportation Science 19:411–435

[7] Brotcorne L, Laporte G, Semet F (2003) Ambulance location and relocation models, European Journal of Operational Research 147:451–468

[8] Aldaihani M, Dessouky MM (2003) Hybrid scheduling methods for paratransit operations, Computers & Industrial Engineering 45:75–96

[9] Rekiek B, Delchambre A, Saleh HA (2006) Handicapped person transportation: An application of the grouping genetic algorithm, Engineering Application of Artificial Intelligence 19:511–520

[10] Xiang Z, Chu C, Chen H (2006) A fast heuristic for solving a large-scale static dial-aride problem under complex constraints, European Journal of Operational Research 174:1117 1139

[11] Claudio Cubillos, Nibaldo Rodríguez, Enrique Urra (2009), Application of Genetic Algorithms for the DARPTW Problem. International Journal of Computers, Communications & Control, June 2009

[12] Colorni A, Righini G (2001) Modeling and optimizing dynamic dial-a-ride problems, International Transactions in Operational Research 8:155–166

[13] Coslovich L, Pesenti R, UkovichW(2006) A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem, European Journal of Operational Research 175:1605–1615

[14]. Nicola Beck. ''*Application de méthodes de clustering traditionnelles et extension au cadre multicritère*''. Mémoire d'ingénieur. Université Libre de Bruxelles, faculté des sciences appliquées. 2006.

[15]. Le Anh Tuan, Ifi Hanoi. ''*Réduction de base de données par la classification automatique*''. Rapport de stage. Institut de la francophonie pour l'informatique (IFI). 2004.

[16]. Chareles Christophe. ''*Une méthode d'aide à la navigation fondée sur k-means, algorithme de classification non supervisée*''. 2004.

[17]. Michel J. A. Berry, Gordon Linoff. *''Data mining, techniques appliquées au marketing, à la vente et aux services clients''*. Ouvrage. Editions InterEditions. 1997.

[18]. Jiawei Han, Micheline Kamber. ''*DM Book*''. Ouvrage. Editions Elsevier Inc. 2006.

[19]. Clark F. Olson. ''*Parallel algorithms for hierarchical clustering. Parallel Computing*''. Ouvrage. Edition Elsevier Science Publishers. 1995.

[20]. Nicolas Pasquier, ''*Data Mining, Clustering*''. Cours en ligne, Université de Nice - Sophia-Antipolis. (Consulté le 21.04.2017). Disponible sur : http://www.i3s.unice.fr/~pasquier/web/?download=m2ntdp.dm.cours.clustering.pdf

[21]. Jiawei Han, Micheline Kamber. *''Data Mining, concepts and techniques''*. Ouvrage. Edition Morgan Kaufmann Publishers. 2000.

[22]. Pierre Emmanuel Jouve. ''*Apprentissage Non Supervisé et Extraction de Connaissance à partir de Données*''. Thèse de Doctorat en Informatique. Université Lumière Lyon 2. 2003.

[23]. Pavel Berkhin. "*Survey of clustering data mining techniques*". Article en ligne. Accrue Software Inc. 2002. (Consulté le 03.05.2017). Disponible sur : http://www.ee.ucr.edu/~barth/EE242/clustering_survey.pdf

[24]. E-G Talbi. ''*Fouille de données (Data Mining) - Un tour d'horizon*''. Présentation en ligne. Laboratoire d'informatiue fondamentale de Lille (LIFL). (Consulté le 02.05.2017). Disponible sur : http://www2.lifl.fr/~talbi/Cours-Data-Mining.pdf

[25]. Sofian Maabout. ''*Regroupement (Clustering)*''. Présentation en ligne. Laboratoire Bordelais de Recherche en Informatique (LaBRI). (Consulté le 02.02.2017). Disponible sur : http://www.labri.fr/perso/maabout/dess_isc/clustering.ppt

[26].Robert Reuven Sokal, C. D. (1958). Statistical Method for Evaluating Systematic Relationships. University of Kansas, USA.

[27].Gower, J. (1968).document "A Comparison of Some Methods of Cluster Analysis. *Biometrics*" , 23(4):623-37.

[28]. K. Florek, J. L. (1951-1957). Document « CHA du saut minimum ».

[29].Sørensen, T. J. (1948). *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons.* denmark.

[30].ward, j. h. (1963). Document " Ward's minimum-variance methode" .

[31].Sarle, S. (1983). Document "*Cubic Clustering Criterion de Warren*".

[32].Quitty, L. L. (1966). Elementary Linkage Analysis for Isolating Orthogonal and Oblique Types and Typal Relevancies.

[33] Sycara K.P. (1998). Multi-Agent Systems. American Association for Artificial Intelligence, AI Magazine, pages 79-92, 1998.

[34] Labrie M.A (2004), Langage de communication agent basé sur les engagements par l'entremise des jeux de dialogue, Mémoire présenté `a la Faculté des études supérieures de l'Université Laval pour l'obtention du grade de maître des sciences, Janvier 2004

[35]Flask Guide rapide, http://www.w3ii.com/fr/flask/flask_quick_guide.html,

(dernière visite 15/08/2017).

[36]Un modèle d'architecture angularjs, https://www.technologies-ebusiness.com/enjeux-et-tendances/modele-darchitecture-angularjs , (dernière visite 15/08/2017).

[37] Architecturer ses applications js à l'aide du pattern mvvm by etienne monier, https://www.docdoku.com/blog/2015/02/17/architecturer-ses-applications-js-pattern-mvvm/, (dernière visite 18/08/2017).

[38] Flask (framework), https://fr.wikipedia.org/wiki/Flask_(framework), édité le 12/03/2017 (dernière visite 20/08/2017).