REPUBLIQUE ALGEREINNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA

Faculté des Sciences

Département d'informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master
En Informatique (Système LMD)

Thème

LA DISTRIBUTION A GRANDE ECHELLE : NOUVELLE APPROCHE INTELLIGENTE POUR LE PROBLEME DE LA TOURNEE DES VEHICULES (VRP).

Présentés par :

TOUAIBIA Mohamed Zaki & ABDESSELAM Rachid

Promoteur : **Mr DJENOURI Youcef** . MA(B). Université SAAD DAHLEB. Blida. DZ.

Encadreur : **Mr TOUAIBIA Maamar Abderrahim** (Gérant de la SARL « HYDROTRIZ »). Blida. DZ.

Année Universitaire : 2015/2016

DEDICACES

Que ce travail témoigne de mes respects

A Mes parents :
Grâce à leurs tendres encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de mes études. Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes profonds sentiments envers eux.
Je prie le bon Dieu de les bénir.

Mon frère wail et aghilas :
Ils vont trouver ici l'expression de mes sentiments de respect et de reconnaissance pour le soutien qu'ils n'ont cessé de me porter.

A tous mes professeurs :
Leur générosité et leur soutien m'oblige de leurs témoigner mon profond respect et ma loyale considération.

A tous mes amis et mes collègues :
Ils vont trouver ici le témoignage d'une fidélité et d'une amitié infinie.

Rachid ...

DEDICACES

Je Dédie ce modeste travail

A

Ma très chère mère, symbole de soutien moral, de sacrifice, m'ayant témoigné toute l'aide dont j'avais besoin au cours de mon cursus universitaire

Que Dieu me la garde !

Mon cher père qui a décédé pendant cette période de préparation, il a toujours veillé à ma réussite et à mon succès

Que dieu ait son âme et l'accepte dans son vaste paradis !

A mon frère, mon beau-frère, ma sœur et ma belle-sœur à qui, je leur souhaite beaucoup de succès dans leur vie professionnelle.

A toute l'équipe de « Google Developer Group Blida »

A mes amis: Anis, Samir, youcef, ali, kamal, Karim, amir, et Yaniss

Et, Bien sûr, à toute ma grande famille

Et à toute personne qui m'aime et que j'aime.

Zaki ...

Résumé

L'objectif principal du travail réalisé est la distribution à grande échelle afin de satisfaire un maximum de clients, pour la livraison des produits et ce, en utilisant plusieurs véhicules constituant la flotte disponible au dépôt. Cette étude consiste à introduire une nouvelle approche intelligente d'optimisation de tournées de véhicules pour la livraison de produits depuis un dépôt central vers le client final en passant par les secteur.

Toute une stratégie a été mise en place ou en a fait une continuité des travaux de doctorat de Berghida (2015), pour résoudre ce problème afin de satisfaire les demandes de la direction commerciale « Sarl HYDROTRIZ ». Ainsi, nous avons implémenté le travail sur un cas réel sous forme d'une application WEB pour la gestion de la distribution basée sur «j2ee ». De même une autre application ANDROID est réalisée pour aider les chauffeurs ou les vendeurs à remettre la marchandise dans les délais et aux bons endroits sans de se tremper dans leur tournée. Ces applications ont donné satisfaction dans un cas réel.

Mots clés: Tournées de véhicules, Intelligence en essaim, Essaims d'abeilles.

Abstract

The main objective of the work is the large-scale distribution to satisfy as many customers use multiple vehicles constituting the fleet available to deposit in order to make deliveries of products. This study is to introduce a new smart approach to vehicle routing optimization for product delivery from a central repository veer the end customer go through a regional depot.

A strategy was put in place to solve the problem in order to meet the demands of commercial management « Sarl HydroTriz ». Thus, we implemented the work on a real case as a web application for the management of the distribution based on « j2ee » and another application ANDROID help drivers or vendors to deliver the goods on time and to the right this places without soak in the tours.

Keywords: Vehicle routing, Swarm Intelligence, Swarms of bee

ملخص

الهدف الرئيسي من هذا العمل هو توزيع على نطاق واسع لتلبية أكبر عدد ممكن من العملاء اللذين يستخدمون مركبات متعددة التشكل من أجل تسليم المنتجات. هذه الدراسة ناتجة عن عرض نهج الذكي الجديد لمركبة التوجيه و هذا من أجل تسليم المنتجات من مستودع مركزي إلى عملاء مرورا من مستودع الإقليمي.

ووضعنا استراتيجية من أجل حل هذه مشكلة, وذلك لتلبية مطالب الإدارة« Sarl HydroTriz » وهكذا نفذنا العمل على شكل تطبيق ويب من أجل التوزيع على أساس« j2ee », ولدينا تطبيق آخرANDROIDمن أجل مساعدة السائقين أو البائعين ويسمح هذا بتسليم البضائع في الوقت المناسب و مكان المناسب دون هذا لا يمكن تحديد الجولات.

وفي الحقيقة كانت كل هذه التطبيقات مرضية.

كلمات المفتاحية : توجيه السيارات, ذكاء أسراب, أسراب من النحل .

REMERCIEMENTS

Nous tenons à remercier particulièrement **Mr Youcef DJENOURI**, notre promoteur, d'avoir accepté de diriger ce travail, de même, nous le remercions pour son dévouement, ses précieux conseils.

Nos remerciements vont aussi au Gérant de la Sarl-HydroTriz en l'occurrence Mr M.A.Touaibia de nous avoir accepté dans sa SARL, de même pour toute la logistique qu'il nous a fourni et les conseils qu'il nous a prodigué.

Nos sincères remerciements vont au Prof B.Touaibia, enseignante à l'Ecole Nationale Supérieure d'Hydraulique de Blida pour ses orientations pédagogiques.

Nous remercions toute l'équipe TRIZ de la SARL-HydroTriz pour les précieux services qu'ils nous ont rendus et spécialement Y.Ferroukhi et A.Lakrib.

Nous tenons également à remercier le président et les membres du jury pour nous avoir fait l'honneur d'évaluer notre travail.

Que tous les enseignants ayant contribué à notre formation trouvent ici, l'expression de notre profonde reconnaissance.

Rachid ABDESSELAM et Zaki TOUAIBIA

Table des matières

Chapitre VI : IMPLEMENTATION

&&&&&

Liste des figures

&&&&&

Liste des tableaux

&&&&&

INTRODUCTION GENERALE

Introduction générale

Face à un défi de la diversification de l'économie hors hydrocarbures, l'état algérien se voit obliger de s'orienter vers l'industrie pour augmenter la productivité nationale. Cependant, le plus grand souci des industriels réside dans l'écoulement et la distribution des produits pour couvrir le marché national et assurer une force de vente.

En effet, les procédures classiques de distribution de produits présentent plusieurs inconvénients à savoir : délais de livraison très lents, coûts élevés de la chaine logistique (stockage, moyens de transports, etc.), coût de communication entre le fournisseur, les intermédiaires et le client final. En plus, les procédures classiques font intervenir plusieurs acteurs pour la réalisation d'une livraison et n'offrent pas une grande visibilité au fournisseur sur les ressources pour la distribution.

De plus, nous sommes actuellement dans une conjoncture où le transport routier se trouve de plus en plus mal. En effet, les récents débats sur la politique menés pour améliorer notre impact sur l'environnement, ainsi que l'augmentation du prix du carburant laissent à penser qu'il devient nécessaire voire obligatoire de minimiser les divers déplacements des véhicules.

Dans ce contexte, le bureau d'études de recherche et développement en informatique, HYDROTRIZ, propose de traiter un modèle de réseau de distribution à grande échelle pour un maximum de clients (magasins) avec plusieurs véhicules constituant la flotte disponible au dépôt pour effectuer les livraisons de produits. Cette étude consiste à introduire une nouvelle approche intelligente d'optimisation de tournées de véhicules (VRP), afin de maitriser les délais de livraison et la logistique.

Un progiciel de gestion dédié pour la grande distribution devra permettre d'une part, de suivre l'évolution d'une distribution depuis le déstockage jusqu'à la livraison au dernier client en respectant l'approche problématique de la tournée des véhicules et d'autre part, minimiser le coût et la distance parcourue par la flotte, en appliquant une métaheuristique coopérative combinante.

Dans notre travail, en vas faire une continuité des travaux de doctorat de **Berghida (2015),** afin, d'améliorées les procédures classiques de distribution de produits et de faire une approche intelligente pour la grande distribution.

Tenant compte de cette problématique, nous avons structuré ce mémoire en 4 chapitres, à savoir :

Dans le premier chapitre nous présentons un état de l'art sur les problèmes de la tournée de véhicules et les différentes approches qui existent ;

Dans le deuxième chapitre, nous détaillerons quelques algorithmes d'intelligence par essaims afin d'utiliser le comportement naturel pour le problème de VRP ;

Le troisième chapitre consiste à introduire notre contribution ainsi que la solution proposée pour résoudre le problème de distribution ;

Le quatrième chapitre est consacré à l'implémentation et la mise en œuvre de l'application.

Le mémoire se termine par une conclusion qui présente un bilan du travail réalisé et expose les perspectives d'avenir et les travaux futurs afin d'améliorer et de compléter le travail présenté.

&&&&&

Introduction

Le problème de la tournée des véhicules (VRP) est un nom générique donné à un ensemble de problèmes comprenant ensemble de routes pour une flotte de véhicules basée dans un ou plusieurs dépôts. L'objectif de la VRP est de former un trajet à faible coût pour servir tous les clients.

En 1959, Dantzig-Ramser a proposé la première formulation en utilisant la programmation algorithmique. Il a également appliqué le VRP avec une application dans le monde réel concernant la livraison de l'essence aux stations-service. D'autre modèles et algorithmes sont proposés pour une solution approximative des différentes versions du VRP. Dans ce chapitre on va mentionner l'architecture des modèles proposés et approfondir la version de base du VRP qui est le CVRP «capacitated vehicle routing problem » puis les différent travaux sur l'état de l'art.

I. Problème de la tournée des véhicules (VRP)

Le problème de la tournée des véhicules (VRP) est un des plus complexes problèmes d'optimisation combinatoire. C'est un problème NP-Difficile. Il peut être décrit brièvement comme suit :

Capacité d'un ou de plusieurs centres de dépôts avec une flotte de véhicules et un ensemble de clients ayant des besoins connus ou prévus.

L'objectif est de déterminer un ensemble de routes avec un circuit fermé combiné a d'autre exigences, comme la restriction du temps de voyage et l'itinéraire approprie.

Dans **Smaili (2012).** Le VRP est un problème de conception de routes. Il est définit par un graphe **(Fig.1.1)**.

Soit : G (V, A) où:

V= $\{\upsilon_0, \upsilon_1..., \upsilon_n.\}$ est l'ensemble des n + 1 noeuds (villes)

Avec : υ_0: le dépôt

$\{v_1..., \upsilon_n.\}$: Ensemble des clients.

A= $\{(i, j): i, j \in v, i \neq j\}$ Ensemble des arcs allant d'un point « i » à un point « j » selon le cas étudié ,chaque arc identifie un élément des matrices soit :

- De distance (où de coût) C = (c_{ij})
- De temps T = (t_{ij})

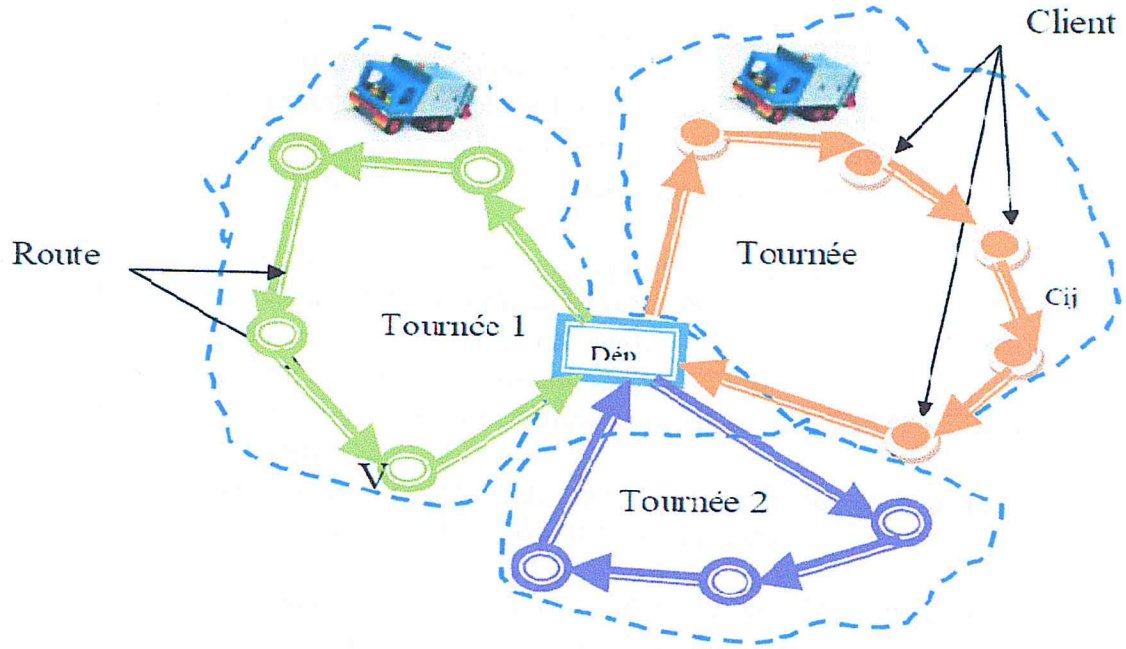


Figure 1.1 Présentation graphique du problème de la tournée des véhicules (VRP) (Smaili 2012).

II. Variantes du problème de la tournée des véhicules

Habituellement, dans le monde réel VRP, de nombreuses contraintes secondaires apparaissent. Certains des plus importants sont représentés en figure 1.2, **(Shodhganga 2015)**

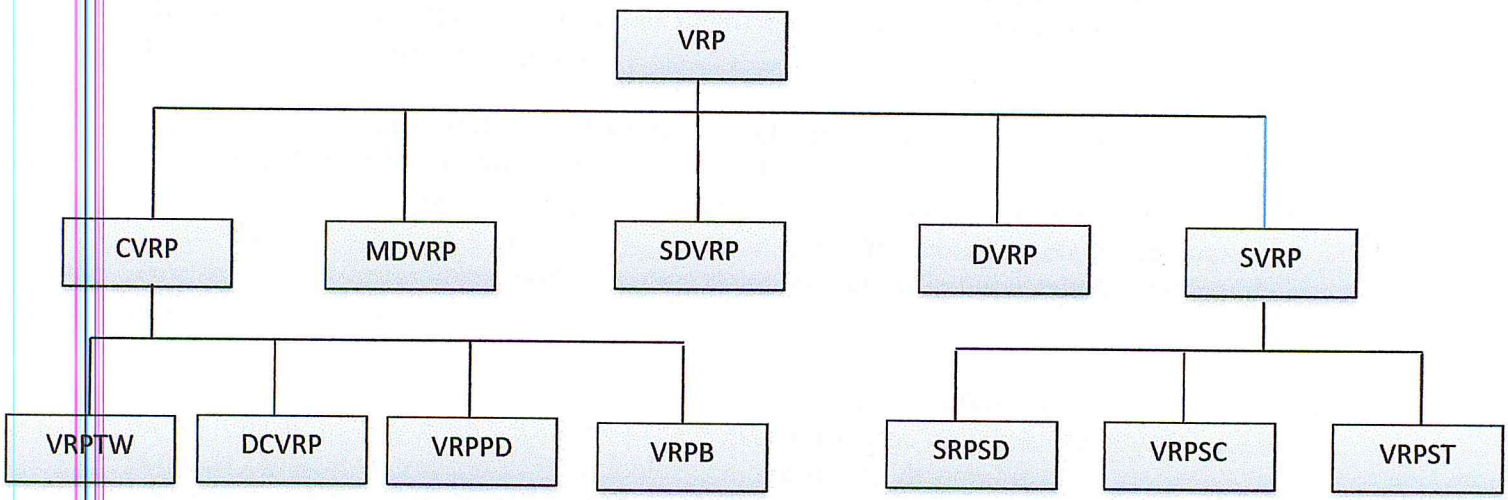


Figure 1.2 Variantes de base de VRP **(Shodhganga 2015)**

Le problème de base de la tournée des véhicules enchaine un certain nombre d'hypothèses, comme l'utilisation d'une flotte homogène, un seul dépôt, une route par véhicule...etc. Ces hypothèses peuvent être enrichies en introduisant des contraintes supplémentaires au problème, à savoir:

- La taille de la flotte disponible : un seul véhicule, plusieurs véhicules ;
- La composition de la flotte disponible : homogène (un seul type de véhicule), hétérogène (plusieurs types de véhicules) ;

Les constantes de données :

- n = nombre de sommets,
- m = nombre de véhicules,
- D_K = capacité du véhicule k,
- T_K = temps maximal de la tournée du véhicule k,
- d_i = demande du sommet i, ($d_1 = 0$),
- t_i^k= temps nécessaire au véhicule k pour charger ou décharger au sommet i,
- t^k_{ij}= temps nécessaire au véhicule k pour voyager du sommet i au sommet j,
- c_{ij} = coût ou distance du voyage du sommet i au sommet j.
- N = ensemble des sommets.
- X = matrice $X_{ij} = \sum_{k=1}^{M} x^k_{ij}$

Les variables de décision :

$$\mathbf{X^k_{ij}} = \begin{cases} 1 & \text{si le véhicule k voyage du sommet i au sommet j} \\ 0 & \text{sinon} \end{cases} \quad (1.1)$$

Avec $\mathbf{x^k=(x^k_{ij})}$.

La fonction à optimiser :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c^j_i \sum_{k=1}^{n} x^k_{ij} \quad (1.2)$$

Sous :

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x^k_{ij} = 1, \; j=2,\ldots\ldots,n \quad (1.3)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x^k_{ij} = 1, \; i=2,\ldots\ldots,n \quad (1.4)$$

$$\sum_{i=1}^{n} x^k_{ip} - \sum_{j=1}^{n} x^k_{pj} = 0, \; k=1,\ldots\ldots,m \; ; \; p=1,\ldots\ldots\ldots n \quad (1.5)$$

$$\sum_{i=1}^{n} d_i \left(\sum_{i=1}^{n} x_{ij}^k \right) \leq D_k \, , \; K=1,\ldots..,m \quad (1.6)$$

$$\sum_{i=1}^{n} t^k_i \sum_{j=1}^{n} x_{ij}^k + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij}^k \leq T_k \, , k=1,\ldots..,m \quad (1.7)$$

$$\sum_{j=2}^{n} x_{ij}^k \leq 1, \; k=1,\ldots..,m \quad (1.8)$$

$$\sum_{j=2}^{n} x_{i1}^k \leq 1, \; k=1,\ldots..,m \quad (1.9)$$

$$X^k \in S \quad (1.10)$$

Où S est donné par l'expression :

$$S=\left((x_{ij})^k \mid \sum_{i \epsilon Q} \sum_{jQ} x_{ij}^k \geq 1, \; \forall \, Q \subset N \, , |Q| \geq 2 \; ; \; k=1\ldots\ldots\ldots..m \right) \quad (1.11)$$

Où Q est un ensemble de sommets visités par un seul véhicule.

La fonction objectif (1.2) consiste à minimiser le coût total de transport.

Les équations (1.3) et (1.4) assurent que chaque sommet ne soit servi qu'une seule fois par un et un seul véhicule. L'équation (1.5) assure la continuité d'une tournée par un véhicule : le sommet visité doit impérativement être quitté l'équation (1.6) assure le respect de la contrainte de capacité du véhicule.

L'équation (1.7) assure le respect de la contrainte de la durée totale d'une tournée. Les équations (1.8) et (1.9) assurent le non dépassement de la disponibilité d'un véhicule.

Un véhicule ne sort du dépôt et n'y revient qu'une seule fois. Finalement les équations (1.10), (1.11) assurent le respect des contraintes d'élimination des sous tournées (tournées revenant au client et pas au dépôt). D'une façon générale, le problème CVRP consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité fixe.
Ce véhicule commence et termine sa tournée au dépôt.

2. Vehicle routing problem with time windows (VRPTW)

Le problème de tournées de véhicules avec contrainte de fenêtres de visite est une généralisation de VRP ou bien un prolongement de CVRP. Dans un VRPTW, de nouvelles contraintes temporelles sont ajoutées : chaque client doit être servi dans un intervalle de temps durant lequel il est disponible pour être visité.

Une fenêtre de visite est un intervalle de temps (a_i; b_i)associé à un client i, cet intervalle représente le temps pendant lequel une visite chez lui est possible.
Pour étendre le modèle proposé par Fisher et Jaikumar afin de prendre en considération les fenêtres de visite des différents sommets, introduisons les termes suivants :

- a_i : la borne inférieure de la fenêtre de visite du sommet i ;
- b_i : la borne supérieure de la fenêtre de visite du sommet i ;
- s_i : le temps de service du sommet i ;
- t_{ij} : le temps de transport entre les nœuds i et j ;
- u^k_i : l'instant de visite du sommet i par le véhicule k ;
- T : une grande valeur (T>>0).

Les contraintes de respect des fenêtres de visite peuvent ainsi s'écrire :

$$\forall i \in(1,\ldots\ldots,n),\ \forall k \in (1,\ldots\ldots..,M) a_i \leq u^k_i \leq b_i \qquad (1.12)$$

$$\forall i \in(1,\ldots\ldots,n),\ \forall j \in (1,\ldots\ldots..,n),\ \forall k \in(1,\ldots\ldots,M) u^k_i + s_i + t_{ij} - T(1-x^k_{ij}) \leq u^k_j \qquad (1.13)$$

La contrainte (1.12) permet de vérifier que l'instant de visite d'un client se trouve entre les bornes inférieure et supérieure de la fenêtre de visite du client concerné. La contrainte (1.13) vérifie la cohérence des instants de visite lorsque deux sites se suivent.

3. Vehicle routing problem with backhauls (VRPB)

Le VRP avec Backhauls VRPB est le prolongement du CVRP, dans lequel l'ensemble des clients V \{0} est partitionné en deux sous-ensembles.

Le premier sous-ensemble L, contient n clients à servir (Linehauls), chacun nécessitant une quantité donnée de produits à livrer.

Le deuxième sous-ensemble B, contient m clients Backhauls, où une quantité donnée de produits entrants doivent être ramassés. Les clients sont numérotées de sorte que :
L = 1 … n et B = n + 1…. n + m.

Dans le VRPB, une contrainte de précédence entre les clients de collecte (Backhauls) et de livraison existe : lorsqu'une route contient les deux types de clients, tous les clients de livraison doivent être servis avant tout client de collecte qui peut être servi. Une demande non négative d_i, pour être livrée ou collectée en fonction de son type, est associé à chaque client i, et le dépôt est associé à une demande fictive $d_0 = 0$. Lorsque la matrice des coûts est asymétrique, le problème est appelé Asymmetric VRP with Backhauls (AVRPB).

Le VRPB et AVRPB consistent à trouver exactement une collection de K circuits simples à un coût minimal, telle que :

1. Chaque circuit visite le sommet du dépôt ;
2. Chaque sommet de client est visité exactement par un circuit ;
3. La demande totale des deux types de clients visités par un circuit ne doit pas dépasser séparément, la capacité C du véhicule ;
4. Dans chaque circuit, tous les clients de livraison précédera la clientèle de collecte, le cas échéant. Les circuits contenant seulement les clients Backhauls généralement ne sont pas autorisés.

4. Vehicle routing problem with pickup and delivery (VRPPD)

Dans la version de base de VRPPD, chaque client i est associé avec deux quantités d_i et p_i, représentant la demande des produits homogènes à livrer et à collecter au client i, respectivement. Dans certains cas, seulement une quantité de demande $d'_i = d_i - p_i$ est utilisée pour chaque client i, indiquant la différence nette entre les demandes de livraison et de collecte (éventuellement négative). Pour chaque client i, O_i dénote le sommet qui est l'origine de la demande de livraison, et D_i dénote le sommet qui est la destination de la demande de collecte.

On suppose que, à chaque emplacement d'un client, la livraison est exécutée avant la collecte, donc le chargement courant du véhicule avant son arrivée à un emplacement donné est défini par la charge initiale moins toutes les demandes déjà livrer plus toutes les demandes déjà collectées.
Le VRPPD consiste à trouver exactement une collection de K circuits simples avec un coût minimal tel que :

1. Chaque circuit visite le sommet de dépôt.
2. Chaque sommet de client est visité exactement par un circuit.
3. La charge courante du véhicule tout au long de circuit ne doit pas être négative et ne doit jamais dépasser la capacité C du véhicule.
4. Pour chaque client i, le client O_i doit être servi dans le même circuit et avant le client i ;
5. Pour chaque client i, le client D_i doit être servi dans le même circuit et après le client i.

Le VRPPD est une généralisation du VRP classique, qui appartient aussi à une large famille de problèmes de livraison et Collecte (PDPs). On peut distingué trois fameux types de problèmes de collecte et livraison qui ont été étudié dans la littérature. Un est Problème de livraison et collecte à un seul produit (single-commodity PDP) dans lequel un seul type de biens est soit collecté ou bien livré à chaque nœud. C'est le cas par exemple quand une véhicule blindée transporte de l'argent entre des branches de bureaux d'une banque.

Une autre variante est le problème de collecte et de livraison à deux produits (two-commodity PDP) où deux types de biens sont considérés et chaque nœud peut agir comme un nœud de livraison et de collecte au même temps. Ce problème survient dans la livraison des boissons quand les véhicules livrent les bouteilles pleines et collectent celles qui sont vides. Une variante de ce problème est le VRP with Backhauls où toutes les livraisons doivent être exécutées avant toute collecte. Finalement, le PDP à n-produits se produit quand chaque produit est associé à un seul nœud de collecte et un seul nœud de livraison. C'est le cas où des voyageurs ou des biens doivent être transportés d'une origine à une destination.

D'habitude, ce problème se ramène au VRPPD. Puisque la plupart des applications pratiques de VRPPD incluent des restrictions sur le temps dans lequel chaque emplacement peut être visité par un véhicule, il est pratique de présenter une variante générale du problème, nommée VRPPD avec fenêtres de temps (VRPPDTW).

Notons par n le nombre de requêtes à satisfaire. Supposons que tous les véhicules ont un seul sommet de dépôt. Le VRPPDTW peut être défini par un graphe direct G = (N,A) où N = P ∨ D ∨ {0,2n+1}, p={1,.....,n} et D = {n + 1,.....2n}. Les sous-ensembles P et D contiennent les nœuds de collecte et de livraison, respectivement, où les nœuds 0 et 2n + 1 présentent le dépôt d'origine et de destination. Avec chaque requête i est ainsi associé un nœud d'origine i et un nœud de destination n + i. Supposons K est l'ensemble des véhicules et m = |K|. À chaque véhicule $k \in K$ est associée une capacité Q_k et sa durée totale à ne pas dépasser en route T_k . Avec chaque nœud $i \in N$ est associé une charge q_i et une durée de service non négative d_i tel que $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ (i = 1,.....,n) et $d_0 = d_{2n+1} = 0$.

Une fenêtre de temps $[e_i , l_i]$ est aussi associée avec chaque nœud $i \in N$ où e_i et l_i représentent le temps le plus tôt et le plus tard, respectivement, dans lequel le service peut démarrer au nœud i. A chaque arc (i; j) 2 A est associé un coût de routage c_{ij} et un temps de voyage t_{ij} .

Pour chaque arc $(i; j) \in A$ et chaque véhicule $k \in K$; $x_{ij}^k=1$ si et si seulement si le véhicule k voyage du noeud i jusqu'au noeud j. Pour chaque noeud $i \in N$ et chaque véhicule $k \in K$. Soit B_{ik} le temps dans lequel un véhicule k commence le service au nœud i, et Q_{ik} est la charge du véhicule k après qu'il visite le nœud i. Le VRPPDTW peut être formulé comme suit:

$$\text{Min}\sum_{k\in K}\ \sum_{i\in N}\sum_{j\in N}\ c_{ij}^k\, x_{ij}^k \qquad (1.14)$$

sous:

$$\sum_{k\in K}\ \sum_{j\in N}\ x_{ij}^k=1,(i \in p) \qquad (1.15)$$

$$\sum_{j\in N}\ x_{ij}^k - \sum_{j\in N}\ x^k_{n+i,j}=0,(i \in p , k \in K) \qquad (1.16)$$

$$\sum_{j\in N}\ x_{0J}^k=1, (k \in K) \qquad (1.17)$$

$$\sum_{j\in N}\ x_{ji}^k-\sum_{j\in N}\ x_{ij}^k=0, (i \in P \cup D , k \in K) \qquad (1.18)$$

$$\sum_{i\in N}\ x_{i,2n+1}^k =1, (k \in K) \qquad (1.19)$$

$$B_j^k \geq (B_i^k+d_i + t_{ij})x_{ij}^k,(i \in N, j \in N , k \in K) \qquad (1.20)$$

$$Q_j^k \geq (Q_i^k+q_j)x_{ij}^k,(i \in N, j \in N , k \in K) \qquad (1.21)$$

$$B_i^k + d_i + t_{i,n+1} \leq B^k_{n+1} ,(i \in p, k \in K) \qquad (1.22)$$

$$B^k_{2n+1} - B_0^k \leq (T_k,(, k \in K)) \qquad (1.23)$$

$$e_i \leq B_i^k \leq l_i,(i \in N, k \in K) \qquad (1.24)$$

$$\max\{0,q_i\} \leq Q_i^k \leq \min\{Q_k,Q_k+q_i\}, (,(i \in N, k \in K) \qquad (1.25)$$

$$x_{ij}^k \in \{0,1\},(i \in N , j \in N , k \in K) \qquad (1.26)$$

La fonction objectif minimise le coût total de routage. Les contraintes (1.14) et (1.15) assurent que chaque requête est servie exactement une fois et que les noeuds de livraison et de collecte associés sont visités par le même véhicule. Les contraintes (1.16)-(1.19) garantissent que la route de chaque véhicule k commence à l'origine dépôt et termine au dépôt destination.

La cohérente des variables de temps et de charge est assurée par les contraintes (1.20) et (1.21). La contrainte (1.22) force les véhicules à visiter le noeud de collecte d'une demande avant son noeud de livraison. Enfin, l'inégalité (1.23) limite la durée de chaque route alors que (1.24) et (1.25) impose la fenêtre de temps et les contraintes de capacité, respectivement.

5. Dynamic vehicle routing problem (DVRP)

La première référence à un problème de routage dynamique du véhicule est due à Wilson et Colvin. Ils ont étudié le problème DARP à un seul véhicule, dans lequel les demandes des clients sont des voyages à partir d'une origine vers une destination qui apparait dynamiquement. Leur approche utilise des heuristiques capables d'obtenir de bons résultats avec un temps de calcul raisonnable. Plus tard, Parafais a introduit le concept de demande immédiate : un client qui demande le service veut toujours être desservi le plus tôt possible, ce qui nécessite une re-planification immédiate de la route actuelle du véhicule.

Le problème de tournées de véhicule dynamique (DVRP) est fortement liée à la VRP statique, car il peut être décrit comme un VRP dans lequel des informations sur le problème peuvent changer au cours du processus d'optimisation.

DVRP est un problème dynamique en temps discret, et peut être considéré comme une série d'instances de P ; chaque cas est un problème statique, qui commence à l'instant t et doit être résolu dans un délai spécifique de t.

Nous le résumons comme suit :

$$p=\{(p_i,t_i, \Delta_t), i=0,1,\dots ,i_{max}\} \qquad (1.27)$$

Avec cette information, la durée de l'instance i est $t_{(i+1)} - t_i$. Le nombre maximum d'instances i_{max} peut-être infini si le problème est ouvert. Une nouvelle instance est générée par l'action de modification d'environnement p_i sur l'instance i. Cela se traduit par $P_{(i+1)} = p_i + Pi$. Ce changement dans l'environnement peut être dû à plusieurs facteurs, par exemple, les temps de déplacement peuvent prendre du temps ou à cause des problèmes liés au trafic, les commandes peuvent être annulées ou modifiées certains clients peuvent être inconnus lorsque l'exécution commence etc.

III. Méthodes de résolution de vrp et ses variantes

1. Travaux sur l'état de l'art

Dans **Laporte (1992)**, quelques résultats principaux connus relatifs au VRP sont examinés. **Toth, Vigo (2001)** ont revues les méthodes exactes les plus efficaces dans la littérature jusqu'en 2002.

Une autre revue sur le problème VRPPD s'est trouvée sur deux parties dans **Parragh et als (2008).** Ces dernières années ont vu une attention accrue sur le VRP intégré avec des contraintes de chargement supplémentaires, connus sous le nom de 2L-CVRP ou 3L-CVRP.

Dans **Wang et als (2009)**, un état de l'art sur 2LCVRP et 3L-CVRP est présenté. Dans **Pillac et als (2013)**, les auteurs classent les problèmes de routage de point de vue qualité de l'information et évolution, après avoir présenté une description générale du routage dynamique, ils introduisent la notion de degré de dynamisme et présentent une revue complète d'application et méthodes de résolution des problèmes de tournées de véhicules dynamique.

Le document **Baldacci et als (2012)** présente une revue des développements récents qui ont eu un impact majeur sur l'état de l'art des algorithmes exacts actuels résolvant le VRP. Les auteurs passent en revue les formulations mathématiques, relaxations et méthodes exactes récentes de deux des variantes les plus connus de VRP : CVRP et VRPTW. Une revue de littérature sur les développements et les publications récentes portant sur le VRP est effectuée dans **Panneerselvam et als (2012).**

Compte tenu de la sensibilité environnementale des problèmes de tournées de véhicule, une revue approfondie de la littérature des problèmes de tournées de véhicules verts (Green VRP) est présenté dans **Lin et als (2014)** par **Berghida (2015).**

2 Méthodes de résolution exactes

Nous présentons d'abord quelques méthodes de la classe des algorithmes complets ou exacts, ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité. Les méthodes exactes peuvent être divisées en 3 grandes catégories :

2.1. Méthode séparation et évaluation (Branch and Bound)

Dans **Douiri et als (2010),** l'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (B&B). **Land, Doig (1960),** repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des nœuds et des feuilles. Le branch-and-bound est basé sur trois axes principaux :

– L'évaluation,
– La séparation,
– La stratégie de parcours.

- L'évaluation

L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale.

L'objectif est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence. Le branch-and-bound utilise une élimination de branches dans l'arborescence de recherche de la manière suivante :

la recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration et à comparer le coût de chaque nœud parcouru à

celui de la meilleure solution. Si le coût du nœud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et toutes les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée.

- Séparation

La séparation consiste à diviser le problème en sous-problèmes. Ainsi, en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions. L'ensemble de nœuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des nœuds actifs.

- Stratégie de parcours

La largeur d'abord : Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées,

La profondeur d'abord : Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

Le meilleur d'abord : Cette stratégie consiste à explorer des sous problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

- Algorithme du Branch & Bound :

L'algorithme maintient la valeur M de la meilleure solution trouvée jusqu'à présent et la liste NA des nœuds actifs, susceptibles de contenir de meilleures solutions que M. (Tableau 1.1)

Tableau 1.1 Algorithme Branch and Bound (**Douiri et als 2010**)

```
NA ←{ racine de l'arbre des solutions }
tant que NA <> ∅ faire
        prendre un noeud actif n dans NA ;
        diviser n ;
        pour chaque fils f de n
            évaluer f et en fonction du résultat, mettre à jour M et
            transformer f en noeud actif (le mettre dans NA) ou l'élaguer ;
        fin pour
fin tant que
```

Exemple du problème du sac à dos

Le problème de sac à dos consiste à remplir un sac à dos de capacité b avec des produits $x_1, x_2,..., x_n$, qui ont un poids $b_1, b_2,..., b_n$ et rapportent un coût $c_1, c_2,..., c_n$ par unité, de façon à maximiser le profit. On considère l'exemple suivant :

$$\text{Max } z = 4x_1 + 5x_2 + 6x_3 + 2x_4$$
$$33x_1 + 49x_2 + 60x_3 + 32x_4 \leq 130$$
$$x_i \in \mathbb{N}$$

Pour ne pas violer la contrainte du problème, chaque variable peut prendre les valeurs suivantes $x_1 \in \{0, 1, 2, 3\}$, $x_2 \in \{0, 1, 2\}$, $x3 \in \{0, 1, 2\}$ et $x_4 \in \{0, 1, 2, 3, 4\}$, **(Fig.1.3)**.

1. Le nœud $x_1 = 3$ avec ($x_2 = 0$, $x_3 = 0$, $x_4 = 0$), dans ce cas, on a mis un seul article dans le sac et prendre comme évaluation du nœud 3 * 4 = 12 : c'est une évaluation exacte qui correspond à une solution. Cela permet d'initialiser la valeur de la meilleure solution courante à 12.

2. Le nœud $x_1 = 2$, On fixe l'article qui donne le meilleur rapport coût/poids, cela est vérifié pour l'article 2. L'évaluation du nœud est donc calculée par : 4*2 + 5* (64/49) = 14, 53. Puisque 14, 53 > 12, on divise ce nœud.

3. Le nœud $x_1 = 2$, $x_2 = 1$ avec ($x_3 = 0$, $x_4 = 0$) on obtient une évaluation exacte égale à 13. La solution correspondante devient la nouvelle meilleure solution courante et la meilleure valeur est égale à 13.

4. Le nœud x1 = 2, x2 = 0 a comme évaluation 2*4+ 6* (64/60) = 14, 4 (x3 = 64/60, x4 = 0). Puisque 14, 4 > 13, on divise ce nœud.

5. Le nœud x1 = 2, x2 = 0, x3 = 1, a comme évaluation est exacte égale à 14. La solution correspondante devient la nouvelle meilleure solution courante et la meilleure valeur est égale à 14.

6. Le nœud x1 = 1, a comme évaluation égale à 4*1 + 5*(97/49) = 13, 89. On passe au dernier nœud x1 = 0, a comme évaluation égale à 5*(130/49) = 13, 26 (x2 = 130/49).

La valeur de la solution optimale égale à 14. Elle consiste à prendre 2 unités du produit 1 et une du produit 3 (x1 = 2, x2 = 0, x3 = 1, x4 = 0), **(Fig.1.4)**.

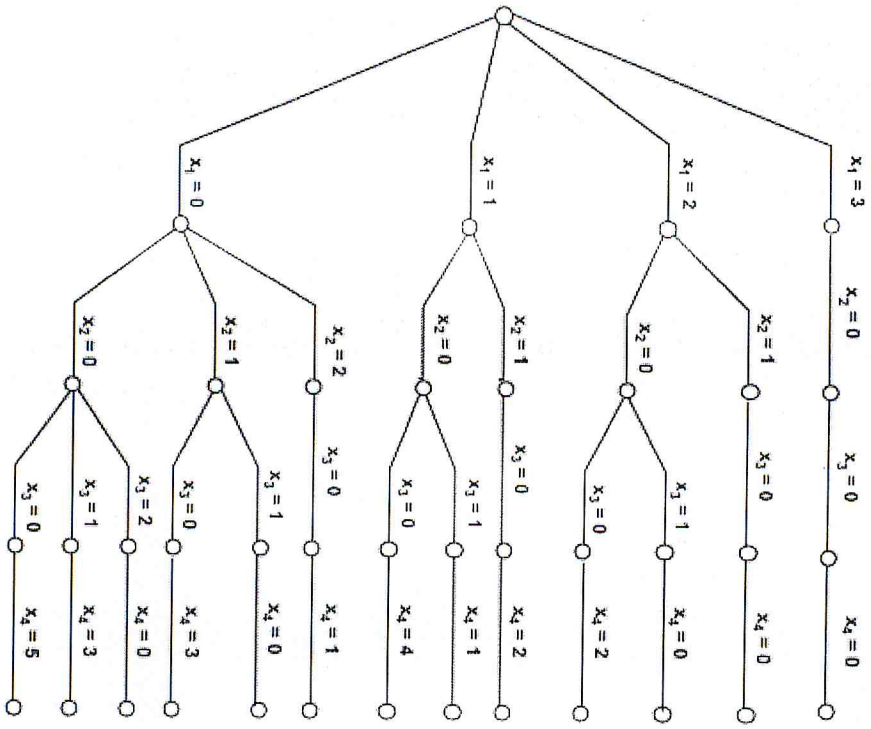


Figure 1.3 : Solutions possibles de l'exemple de sac à dos.

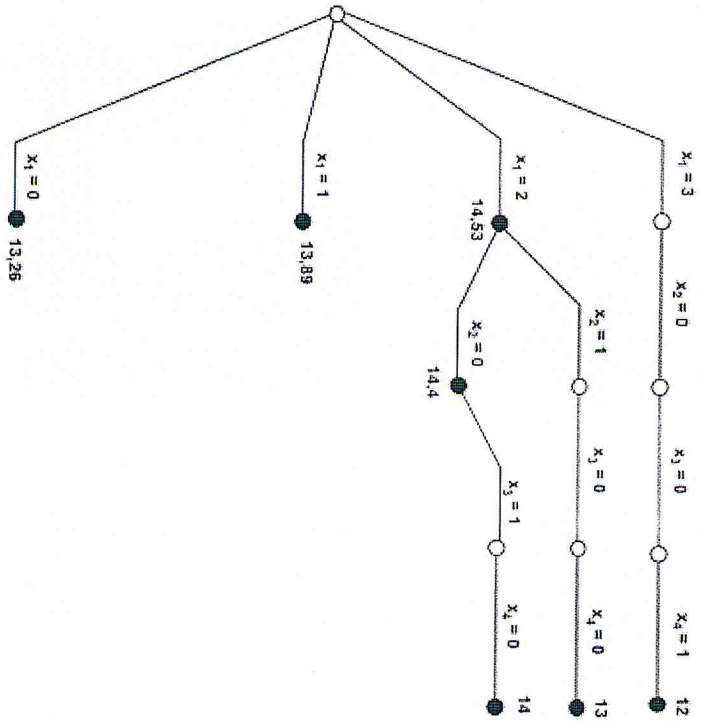


Figure 1.4. Résolution de l'exemple de sac à dos par Branch & Bound.

2.2. Procédure de coupes et de séparation: la méthode (Branch and Cut)

On vient de voir qu'on ne peut souvent pas résoudre efficacement des problèmes N P-difficiles par la méthode des coupes polyèdres. De même, bien que l'algorithme du "Branch and Bound" puisse être très performant pour une certaine classe de problèmes, par exemple le problème de sac à dos, il reste très limité quand il est difficile de calculer une borne inférieure du problème d'optimisation. C'est le cas des problèmes possédant un nombre exponentiel d'inéquations dans leur formulation en un programme linéaire en nombres entiers.

Dans **Ekbal (2004)**, l'algorithme du "Branch and Cut" est une méthode qui conjugue les efforts de l'algorithme du "Branch and Bound" et de la méthode des coupes polyèdres. Ainsi, pour résoudre un programme linéaire en nombres entiers, le "Branch and Cut" commence par résoudre une relaxation du problème puis il applique la méthode des coupes polyèdres sur la solution trouvée. Si celle-ci n'arrive pas à obtenir une solution entière alors le problème est divisé en plusieurs sous-problèmes qui seront résolus de la même façon. L'algorithme de base est décrit dans **tableau.1.2**.

Tableau.1.2 Algorithme Branch and Cut

Debut
Liste des problèmes = Ø;
Initialiser le programme linéaire par le sous problème de contraintes
(A_1, b_1) avec $A_1 \in R^{m1xn}$ et $b_1 \in R^{m1}$) avec $m_1 << m$;
Etapes d'évaluation d'un sous problème
Calculer la solution optimale $\bar{x}$ du programme linéaire $c^t \bar{x} = \min(c^t x : A_1 x \geq b_1, x \in R^n)$;
Solution courante= Appliquer la méthode des coupes polyédrales() ;
Fin étapes d'évaluation
Si Solution courante est réalisable alors
$x^* = \bar{x}$ est la solution optimale de $\min(c^t x : Ax \geq b, x \in R^n)$;
Sinon
Ajouter le problème dans Liste des sous problèmes ;
Fin Si
Tantque Liste des sous problèmes ≠ Ø
Sélectionner un sous problème ;
Brancher le problème ;
Appliquer les étapes d'évaluation ;
Fin_Tantque
Fin.

Une telle approche a été utilisée pour la première fois pour le problème de "linear ordering" par **Grotschel et als (1984)**. Le terme de "Branch and Cut" était introduit par **Padberg , Rinaldi (1991 et 1987)** pour un algorithme de résolution du problème du voyageur de commerce. Ces derniers ont établi le premier état de l'art de cet algorithme, en introduisant des nouvelles procédures telles que la génération de colonnes, des procédures de séparation sophistiquées et une utilisation efficace du solveur du programme linéaire.

On veut résoudre le problème d'optimisation ($\min c^t x : Ax \geq b; x \in \mathbf{R}^n$) avec $A \in \mathbf{R}^{mxn}$ et $b \in \mathbf{R}^m$.

2.3 Programmation dynamique

La programmation dynamique est un paradigme de conception qu'il est possible de voir comme une amélioration ou une adaptation de la méthode « *diviser et régner* ». Ce concept a été introduit par Bellman, dans les années 50, pour résoudre typiquement des problèmes d'optimisation. La programmation dynamique repose sur le principe d'optimalité « *toute politique optimale est composée de sous-politique optimales* ». Elle a été appliquée pour la première fois au VRP pare **Eilon, al (1971).** La programmation dynamique peut être appliquée dès lors qu'une solution optimale peut être représentée comme une combinaison de solutions optimales de sous-problèmes.

Alors, la programmation dynamique résout chaque sous-problème une seule fois et stocke les solutions de tous les sous-problèmes rencontrés dans une matrice, pour éviter d'avoir à les recalculer par la suite. Enfin, elle cherche une relation de récurrence entre les sous-problèmes de sorte à résoudre le problème principale, **tableau.1.3**.

Cette méthode a été utilisée pour des problèmes de VRP de très petites tailles comme dans **Rego, Roucairol (1994)** et **Haj_Rachid (2010)** pour la résolution de problèmes allant de 10 à 25 clients.

Tableau.1.3 Algorithme de Bellman **(Haj_Rachid 2010)**

Debut

P(y) désigne l'ensemble des prédécesseurs du sommet y au sens large

y ϵP(y)=> Fk est toujours décroissante.

N : nb de sommets

x_0 : racine du graphe

P : ens. des prédécesseurs (sens large)

l(z,y) : longueur de l'arête (z,y)

$F_0(x0)=0$, $F0(y) = \infty$ pour $y \neq x0$

Pour k de 1 à N-1

Pour tout sommet y

F1(y)= min(F0(z) + l(z,y); z ϵ P(y)) ;

Fin_pour

$F_0=F_1$;

Fin_pour

Fin.

Exemple :
Par programmation dynamique, déterminer dans la **(Fig.1.5)**, le plus court chemin de A à F.

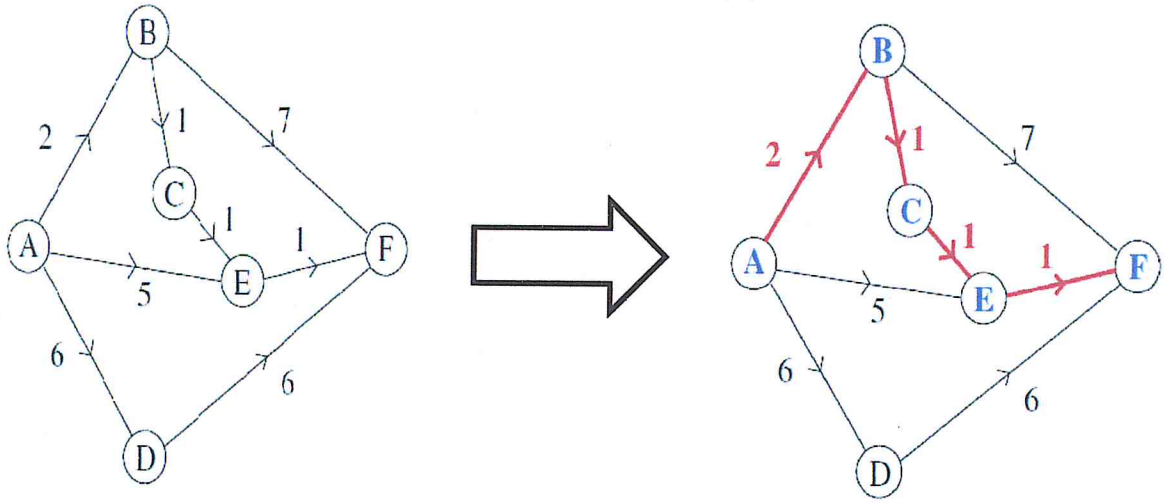


Figure 1.5 Exemple du plus court chemin

Donc chemin optimal (A,B, C,E, F) de longueur minimale 5

3 Méthodes de résolution approchées

Un problème d'optimisation combinatoire NP-difficile est difficile à résoudre de manière exacte pour les grandes instances. Les méthodes exactes nécessitent un grand temps de calcul qui croit exponentiellement avec la taille des instances du problème. C'est pour cela qu'on fait appel aux méthodes approchées (heuristiques et métaheuristique).

Dans ce qui suit, nous citons les principaux travaux qui traitent le problème du VRP et ses variantes par les méthodes approchées entre autre, ceux de **Berghida (2015)**.

En 1999, un algorithme heuristique a été développé par **Toth, Vigo (1999)** pour résoudre le problème de VRPB symétrique et asymétrique. Pour la même classe (VRPB), une heuristique unifiée a été proposée aussi par **Ropke, Pisinger (2006).** Dans **Nagy, Salhi (2005),** les auteurs proposent une méthode qui trouve une solution au problème de VRP et modifie cette solution pour qu'elle soit possible pour le VRPPD. Dans **Geem et als (2005),** l'algorithme de recherche d'harmonie est appliqué sur le problème VRP. En 2006, un algorithme mémétique **Saremi et als (2006)** utilisant différents algorithmes de recherche locale est proposé pour résoudre le VRPB.

De nombreuses façons de modélisation des contraintes de collecte ont été proposées dans la littérature, imposant chacune des restrictions différentes concernant le traitement des clients de collecte. Une étude de ces modèles est présentée et un modèle unifié est développé dans **Ropke, Pisinger (2006).** Ce modèle est capable de gérer la plupart des variantes du problème dans la littérature. Un nouvel algorithme évolutionnaire quantique (Quantum Inspired Evolutionary Algorithm QIEA) est présenté dans **Hu, Wu (2009)** pour le VRPSPD (Vehicle Routing Problem with Simultaneous Pickup and Delivery).

En 2009, un système d'aide à la décision DSS est développé **Yazg-Tütüncü (2009)** afin de résoudre le VRPB, VRPB mixte et le VRPB restreint (le VRPB restreint est un problème compromis entre VRPB classique et le VRPB mixte), et un nouveau critère qui

considère la capacité restante des véhicules est propos é pour la création des solutions pour le VRPB mixte et restreint.

Un algorithme quantique d'optimisation par essaims particulaires est proposé dans **Zhengchu et als (2010)** pour résoudre le problème de CVRP. De même, pour résoudre la même variante (CVRP), un algorithme de colonies de fourmis modifié est proposé dans **Chen et als (2010).** Il consiste à minimiser la distance totale parcourue par chaque véhicule ainsi que le temps total de service sur les noeuds clients. Le CVRP étudié est traité en deux phases : phase d'affectation client/véhicule et phase de minimisation du temps d'exécution.

Le problème de tournées de véhicule avec collecte et livraison sélectif SPDP est une nouvelle variante de VRPPD. Ce problème relaxe la contrainte que tous les nœuds de collecte doivent être visités le long du chemin. Plus précisément, le SPDP a pour but de trouver le chemin le plus court qui peut servir les nœuds de livraison avec les produits nécessaires à partir de quelques nœuds de collecte sélectionnés. Dans l'étude de **Liao, Ting (2012),** ils proposent une mutation adaptative qui se concentre sur la sélection de noeuds de collecte appropriés pour le SPDP. Deux algorithmes évolutionnaires (algorithme génétique et algorithme mémétique) sont développés.

Dans l'article de **Naji-Azimi, Salari (2013),** un programme linéaire basé sur une approche heuristique est proposé. Ce programme peut être utilisé comme un outil complémentaire pour améliorer la performance des méthodes existantes qui résolvent ce problème. En 2012, les auteurs dans **Hong (2012)** étudient le problème de tournées de véhicule dynamique avec fenêtres de temps (DVRPTW).

Dans la même année, une nouvelle version modifiée de GRASP nommée (MPNSGRASP) pour résoudre le CVRP est proposée dans **Marinakis (2012).** En 2013, l'article **Franceschelli et als (2013)** aborde une nouvelle classe des problèmes de routage de véhicules multiples hétérogènes. Les auteurs proposent deux algorithmes distribués basés sur la communication potinée (Gossip communication).

Le premier algorithme est basé sur une optimisation locale exacte et le deuxième est basé sur une heuristique gloutonne approximative locale. Ran Liu et al. **Liu, Xie (2013)** traitent dans leur article un problème d'ordonnancement de véhicules rencontré dans la logistique des soins à domicile. Il concerne la fourniture des médicaments et de dispositifs médicaux de la pharmacie de l'entreprise des soins à domicile aux maisons des patients, livraison des médicaments spéciaux de l'hôpital aux patients et collecte d'échantillons biologiques, des médicaments non utilisés et des dispositifs médicaux de patient. Le problème peut être considéré comme un VRP spécial avec livraison et collecte simultanée et fenêtres de temps avec quatre types de demandes : livraison de dépôt au patient, livraison de l'hôpital à un patient, collecte de patient au dépôt et collecte de patient au laboratoire médicale.

Deux modèles de programmation entière mixte sont proposés. Ils proposent ensuite un algorithme génétique et une méthode taboue. Un algorithme (max-min Ant System) pour résoudre le SDWVRP (Split Delivery Weighted Vehicle Routing Problem, qui consiste à construire des routes optimales en respectant les contraintes de capacité de véhicule et le poids de chargement pour servir un ensemble de client à coût minimal) est proposé dans **Tang et als (2013).**

Dans **Cao et als (2014),** les auteurs étudient l'OVRP avec des demandes incertaines, où les véhicules ne reviennent pas nécessairement à leur emplacement d'origine après la

livraison des marchandises aux clients. Ils proposent quatre stratégies robustes pour faire face à l'incertitude de la demande et une amélioration de l'algorithme d'évolution dfférentielle IDE pour résoudre le modèle robuste d'optimisation.

Dans **Gulic, Jakobovic (2013),** la programmation génétique est utilisée pour faire évoluer une heuristique qui crée des solutions initiales pour différents objectifs et différentes classes de VRP.

Conclusion

Le problème de routage de véhicules. VRP est un problème d'optimisation combinatoire et de recherche opérationnelle. Il fait partie de la catégorie des problèmes de transport, tout comme le problème du voyageur de commerce.

Dans ces problèmes relevant du domaine de la logistique, un ou plusieurs véhicules doivent couvrir un réseau de transport pour livrer des marchandises à des clients ou couvrir les routes de ce réseau.

Dans notre cas, On vas faire une continuité des travaux de doctorat de **Berghida (2015),** afin, d'améliorées les procédures classiques de distribution de produits et de faire une approche intelligente pour la grande distribution. Car les procédures classiques, présentent plusieurs inconvénients à savoir : délais de livraison très lents, coûts élevés de la chaine logistique (stockage, moyens de transports, etc.), coût de communication entre le fournisseur, les intermédiaires et le client final.

En plus, les procédures classiques font intervenir plusieurs acteurs pour la réalisation d'une livraison et n'offrent pas une grande visibilité au fournisseur sur les ressources pour la distribution.

&&&&&

Chapitre II

Intelligence en essaim

Introduction

À ses débuts, l'Intelligence Artificielle a puisé son inspiration dans le comportement individuel de l'être humain, en cherchant à reproduire son raisonnement. Elle s'est donc focalisée sur la manière de représenter les connaissances d'un expert et de modéliser son processus de décision, pour construire des systèmes dont le résultat pouvait être qualifié d'« intelligent ».

Mais dans la nature, on observe bien des formes d'intelligence. Pourquoi ne pas prendre en compte une intelligence qui ne serait plus individuelle, mais collective ? Les chercheurs ont ainsi conçu des systèmes « multi-agents » en s'inspirant du fonctionnement d'un groupe d'experts humains.

Une autre source d'inspiration est celle des sociétés d'insectes. Ainsi est née «l'intelligence en essaim », qui trouve des applications dans le domaine de la simulation et au-delà, par exemple en robotique collective ou pour les réseaux.

Ainsi, dans ce chapitre, les différentes approches basées sur l'intelligence par essaims, en particulier, les essaims d'abeilles seront étudiées. On analyse d'abord le comportement des abeilles dans la nature, pour étudier ensuite les différentes implémentations des abeilles artificielles proposées dans la littérature.

I. Intelligence par essaims

Un groupe d'experts humains qui doit résoudre un problème, ou encore une société d'insectes, peuvent devenir des sources d'inspiration pour concevoir un système informatique dont « l'intelligence » provient d'un ensemble d'entités - des « agents » - en interaction.

Dans le premier cas, lorsqu'on emploie la métaphore d'un collectif d'humains, chaque entité est supposée douée «d'intelligence ». Les agents communiquent directement en s'envoyant des messages, Ils possèdent des représentations du problème à résoudre et sont capables de raisonnements élaborés. La complexité des modèles utilisés autorise à faire appel à des facultés cognitives avancées : représentation explicite et raisonnement sur les autres, raisonnement sur l'avancement de la résolution, persistance d'intentions, notion d'engagement, etc... .

Dans le second cas, au contraire, la métaphore est d'ordre biologique : les agents sont dotés de capacités restreintes de représentation et de raisonnement ; ils sont situés dans un environnement au travers duquel ils interagissent de manière indirecte en y déposant des marques. Alors que dans le premier cas, l'environnement est souvent absent, ici il joue un rôle primordial puisqu'il sert de support aux interactions entre les individus.

1. Intelligence collective

On a longtemps cru (à tort) que les insectes sociaux étaient plus intelligents que les insectes solitaires, au vu des tâches complexes qu'ils accomplissaient. En effet, bien que ne pouvant être individuellement qualifiés d'intelligents, les membres de ces sociétés sont collectivement capables de réaliser des constructions sophistiquées, de s'adapter à des environnements changeants et de trouver le plus court chemin à une source de nourriture.

Autant d'activités collectives dont la sophistication va bien au-delà des simples capacités de chacun des individus : on parle alors d'intelligence collective.

2. Intelligence en essaim artificielle

En Intelligence Artificielle, l'intelligence en essaim fait référence à cette intelligence collective des sociétés d'insectes. Elle a fait l'objet de nombreux travaux de recherche.

L'intelligence en essaim consiste à étudier et à construire des sociétés d'individus artificiels simples qui sont capables collectivement de fournir une réponse complexe. Dans un tel système multi-agents, chaque agent n'a qu'une vue limitée du système, mais il décide de manière autonome de ses actions.

De ce fait, le système est caractérisé par un fonctionnement décentralisé : aucun agent ne décide, ni ne coordonne les actions des autres.

Chaque agent est simple : il ne fait appel à aucune représentation ni mécanisme de raisonnement sophistiqué. Ainsi la résolution est le fait des interactions et de la dynamique du système : l'intelligence naît de façon collective. Le résultat global du système est donc émergent, constitué d'une succession de comportements de type « réflexes ».

Par exemple, imaginons un système multi-agent dont l'objectif serait que chaque agent se dispose de manière équidistante sur un cercle de centre O et de rayon r. Cet objectif peut être atteint en dotant chaque agent de deux comportements élémentaires. Le premier comportement s'énonce ainsi : « se diriger jusqu'à une distance r du point O (perçu comme un stimulus de l'environnement) » et le second : « une fois à distance r de O, s'éloigner au maximum des autres ». Elle montre la capacité qu'ont des agents, placés aléatoirement au départ, à s'organiser seuls pour former un cercle, en se répartissant régulièrement autour de celui-ci. Les agents, rouges à l'origine, deviendront jaunes quand ils seront dans une situation relativement stable, et verts dans une situation très stable. L'applet montre aussi l'importance du réglage des paramètres : en effet, des paramètres mal adaptés ne permettent pas d'arriver à une situation stable.

3. Avantage du collectif

De tels systèmes se caractérisent par leur adaptabilité et leur robustesse. En effet, du fait d'un contrôle décentralisé, chaque agent réagit en fonction de ses propres perceptions aux modifications de son contexte et il est capable de s'adapter continuellement aux variations de celui-ci. De plus, le nombre des agents, leur caractère interchangeable, l'absence d'entité centralisatrice rendent un tel système tolérant à la défaillance d'un de ses membres. Dans le cas de l'exemple du cercle, le nombre d'agents composant le système n'a pas besoin d'être précisé pour que le système fonctionne (on peut même ajouter ou supprimer des agents pendant qu'il fonctionne).

Ces deux propriétés permettent à un tel système de changer de comportement en cours de fonctionnement pour qu'il s'adapte aux évolutions de son environnement et d'avoir une dégradation progressive du fonctionnement collectif plutôt qu'un effondrement brutal.

Pour concevoir de tels systèmes, la difficulté principale qui se pose est de déterminer les comportements individuels, leur environnement et la dynamique qui va régir le fonctionnement du système afin qu'il produise la réponse collective souhaitée (**Snterstices, 2016**)

II. Simulation des comportements naturels

Cette approche sert tout d'abord à représenter des comportements naturels observés, en biologie du comportement animal, en agronomie, en sociologie, et on présentera deux algorithmes,

- (ACO) « **A**nt **C**olony **O**ptimisation » basés sur le comportement des colonies de fourmis.
- (PSO) « **P**article **S**warm **O**ptimization » basés sur le comportement des oiseaux.

Ces derniers ont été appliqués dans plusieurs problèmes d'optimisation. Pour les deux prochaines sections, on détaillera sur les essaims d'abeilles et leur comportement dans la nature, qui ont été très utilisé dans les années 2000.

1. Algorithme des colonies de fourmis

L'algorithme de colonies de fourmis (ACO) provient du comportement des fourmis. Elle se base sur la première expérience qui a permis de mettre en évidence l'intelligence collective développée par les fourmis en quête de nourriture. C'est cette expérience, menée dans les années 1980 par Deneubourg, professeur à l'Université Libre de Bruxelles, qui est à l'origine du développement du domaine de recherche de l'intelligence en essaim.

Deux chemins de longueurs différentes permettent aux fourmis d'accéder de leur nid à une source de nourriture. Une fourmi seule prendra indifféremment l'un ou l'autre chemin, alors qu'une colonie de fourmis choisira de manière privilégiée le chemin le plus court.

L'explication biologique de ce phénomène est assez simple. En effet, en se déplaçant, chaque fourmi marque son chemin en déposant une certaine quantité de phéromones. Au départ, les fourmis empruntent au hasard l'un des chemins. Mais celles qui ont pris le chemin le plus court retournent plus rapidement au nid, déposant donc plus rapidement une nouvelle quantité de phéromones sur le chemin du retour. Les autres fourmis sont ensuite guidées par les traces de phéromones les plus importantes. On observe donc qu'après un temps de convergence, le plus court chemin est emprunté collectivement par une majorité d'individus.
Elle illustre la découverte du plus court chemin entre une fourmilière et une source de nourriture.

Des fourmis sortant à gauche vont essayer d'atteindre la nourriture à droite, en passant par l'un des deux chemins, puis chacune devra rentrer déposer la nourriture acquise. Après un certain nombre de passages des fourmis, l'une des branches deviendra très fréquentée : la plus courte entre la fourmilière et la nourriture. Cet algorithme est présenté en tableau 2.1.

Tableau 2.1. Algorithme « Ant Colony Optimisation » (ACO)

```
Debut
        Initialisation des pistes de phéromone ;
        Tanque critère d'arrêt non atteint faire
                Construire les solutions composant par composant ;
                Mise à jour des pistes de phéromone ;
        fin_tanque
Fin
```

2 Algorithme des essaims de particules

En 1995, Russel Eberhart, ingénieur en électricité et James Kennedy, socio-psychologue, s'inspirent du monde du vivant pour mettre en place une méta-heuristique : l'optimisation par essaim particulaire. Cette méthode se base sur la collaboration des individus entre eux : chaque particule se déplace et à chaque itération, la plus proche de l'optimum communique aux autres sa position pour qu'elles puissent modifier leur trajectoire. Cette idée veut qu'un groupe d'individus peu intelligents puisse posséder une organisation globale complexe.

De nombreuses recherches récentes sont faites sur la P.S.O, mais la plus efficace jusqu'à lors est l'élargissement au cadre de l'optimisation combinatoire. En effet, en 2000, Maurice Clerc, un chercheur de France Telecom met en place la D.P.S.O (Discrete Particle Swarm Optimization), en remplaçant les points par des ordonnancements et les fonctions continues par des fonctions d'évaluation.

Nous avons donc, afin de comprendre l'efficacité et les limites de cette approche, appliqué cette méthode au Problème du Voyageur de Commerce, un problème d'optimisation combinatoire, qui, à priori, est très mauvais pour ce genre d'heuristique d'optimisation.

Pour de nombreux problèmes, il n'existe pas de solution déterministe qui donne le résultat en un temps raisonnable, et ceci malgré la création d'ordinateurs de plus en plus performants. Pour pallier à ce problème, on a recours à des méthodes dites heuristiques, c'est-à-dire des méthodes qui fournissent une solution approchée. Toutefois, il faut reproduire le processus sur plusieurs itérations pour tendre vers une solution acceptable.

On retrouve parmi ces heuristiques, certains algorithmes qui possèdent un principe générique adaptable et qui s'applique donc à plusieurs problèmes d'optimisation. On les appelle des métaheuristiques.

La plus courante est la descente stochastique : on part d'une solution initiale, on la compare à tous ses voisins en conservant à chaque fois le meilleur résultat.

L'optimisation par essaim particulaire, qui dérive de la descente stochastique, entre dans cette famille d'algorithmes. Elle s'inspire fortement des relations grégaires des oiseaux migrateurs qui doivent parcourir des longues distances et qui doivent donc optimiser leurs

déplacements en termes d'énergie dépensée, comme par exemple la formation en « V », **Maxime, Abdoulaye (2011),** Cet algorithme est présenté en **tableau 2.2.**

Le principe de l'algorithme peut être plus facilement visualisé en figure 2.1.

Tableau 2.2. Algorithme « Particle Swarm Optimization » *(***Maxime, Abdoulaye 2011)**

On note **g** la meilleure position connue de l'essaim et f(x) la fonction qui calcule le critère de **x.**
Debut
Pour chaque particule :
On initialise sa position ;
On initialise sa meilleure position p connue comme étant sa position initiale ;
Si $f(p) < f(g)$ **alors**
On met à jour la meilleure position de l'essaim
On initialise la vitesse de la particule.
Tant que l'on n'a pas atteint l'itération maximum ou une certaine valeur du critère **faire**
Pour chaque particule i **faire**
On tire aléatoire c_2 et c_3 ;
On met à jour la vitesse de la particule suivant la formule vue précédemment ;
On met à jour la position x_i ;
Si $f(x_i) < f(p_i)$ **alors**
On met à jour la meilleure position de la particule ;
Si $f(p_i) < f(g)$ **alors**
On met à jour la meilleure position de l'essaim ;
Fin si
Fin si
Fin pour
Fin tant que
g est l'optimum.
Fin.

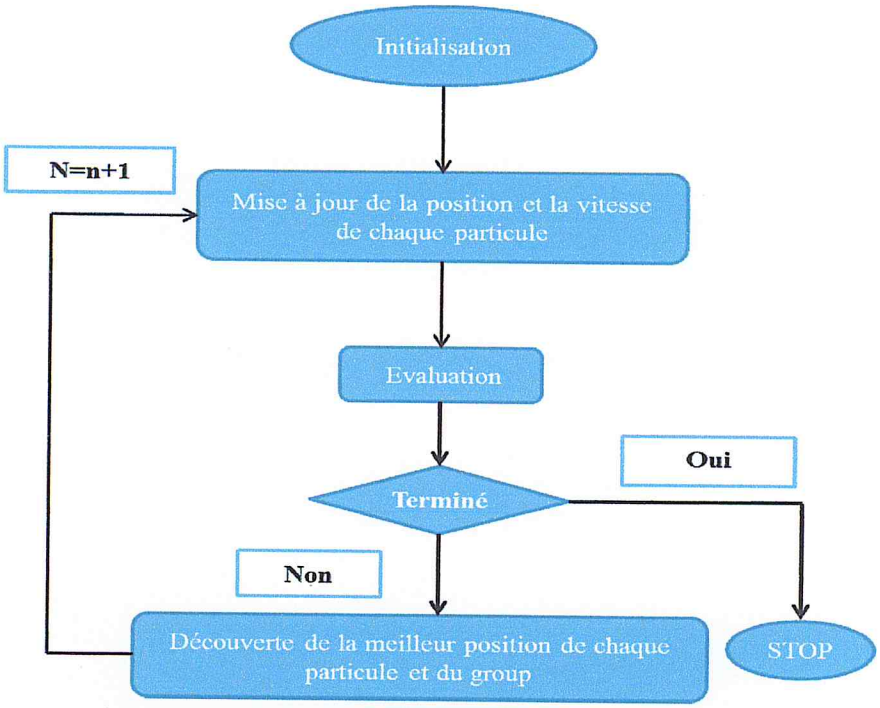


Figure 2.1: Organigramme de la méthode des essaims particulaires

III Les essaims d'abeilles

Les abeilles ont une mémoire photographique, des antennes et des systèmes de navigation, qui permettent, d'une part, de détecter de nouvelles ruches, et d'une autre part, d'accomplir leurs différentes tâches biologiques telles que : la recherche de nourriture et la reproduction.

Plusieurs chercheurs se sont inspirés du comportement naturelle collectif des abeilles naturelles afin de créer des différents modèles et d'algorithmes résolvant des problèmes d'optimisation combinatoire.

Les essaims d'abeilles se composent de 3 types :

- la reine
- les ouvrières
- les mâles.

1 Types d'abeilles

La reine : Elle est la seule femelle fertile de la communauté, Elle est donc la mère de tous les individus composant cette communauté (les faux bourdons, les ouvrières et les futures reines). Sa capacité à pondre est très importante, sa production journalière dépasse souvent 1500 oeufs.

Sa nourriture quasi exclusive est une secrétion, appelée gelée royale, produite par les glandes situées dans la tête des ouvrières.

L'ouvrière : Les ouvrières sont toujours beaucoup plus nombreuses que les mâles, sont compris entre 8 000 et 15 000 individus au printemps, mais peut dépasser 80 000 au début de l'été. Les ouvrières sont incapables de s'accoupler et donc de se reproduire. Elles secrètent la cire, construisent les alvéoles, récoltent le nectar, le pollen et l'eau, transforment le nectar en miel, nettoient la ruche et la défendent contre les prédateurs.

Le mâle : Il ne travaille pas, il est incapable de se nourrir lui-même et ne peut même pas défendre la colonie, puisqu'il n'a pas de dard, Le nombre de mâle est compris entre 300 et 3000 dans la colonie, Sa seule fonction est de s'accoupler avec la reine. Quand celle-ci quitte la ruche, attirés par son l'odeur de cette dernière. Après l'accouplement, il meurt de faim rapidement.

2. Comportement des abeilles dans la nature

Les abeilles vivent en colonie. Elles forment une société très organisée, un peu comme une grande entreprise. Autour de la reine, dont la tâche unique est de pondre et pondre encore, jusqu'à 80 000 ouvrières qui s'activent avec ardeur. Dans la ruche, seules les quelques centaines de faux-bourdons paressent ! Durant leur existence, les abeilles exercent jusqu'à sept fonctions différentes : nettoyeuse, nourrice, architecte, manutentionnaire, ventileuse, gardienne et butineuse. Mais toutes les abeilles ne suivent pas le même « parcours professionnel », certaines brûlent les étapes pour devenir butineuses, alors que d'autres n'accèdent jamais à ce statut.

La nettoyeuse : Au premier jour de sa vie, l'abeille est préposée au ménage. Elle commence par nettoyer les cellules. Le nettoyage général du fond de la ruche est effectué par des abeilles plus âgées, entre 10 et 15 jours.

La nourrice : Quand elle atteint 5 à 6 jours, l'abeille est capable de sécréter de la nourriture pour les larves, elle devient alors nourrice et le reste jusqu'à l'âge de 15 jours. Les nourrices prodiguent des soins attentifs aux larves qui sont alimentées individuellement plus de 1 000 fois et reçoivent 7 000 visites de contrôle.

L'architecte : La construction des rayons est un travail collectif qui demande une grande coordination. Ils sont fabriqués par une chaîne d'abeilles qui sécrètent des écailles de cire. Un ouvrage délicat et épuisant entrepris par des maçonnes qualifiées ayant en général entre 5 et 20 jours, âge où la capacité de production des glandes cirières est optimale.

La ventileuse : L'âge moyen des ventileuses est estimé à 18 jours, mais cette fonction est assurée par des ouvrières de tout âge. La ventilation consiste à battre des ailes pour aérer la ruche et contrôler ainsi sa température, son taux d'humidité et son taux de gaz carbonique. Elle sert aussi à assécher le nectar.

La gardienne : C'est une vigile postée à l'entrée de la ruche. La gardienne protège la colonie de ses ennemis. Elle contrôle l'identité des abeilles qui entrent dans la ruche en vérifiant leur odeur, pour s'assurer qu'il ne s'agit pas d'individus d'autres colonies venus piller leurs réserves. Les gardiennes ont entre 12 à 25 jours.

La butineuse : Vers l'âge de trois semaines, l'ouvrière peut devenir butineuse et s'envole enfin hors de la ruche à la recherche de nectar, de pollen et d'eau, indispensables à la colonie. Une butineuse effectue une dizaine à une centaine de voyages par jour selon la

proximité des fleurs. A ce train d'enfer, elle s'épuise vite et, au bout de quatre à cinq jours, elle meurt.

Le rôle principal de l'abeille qui cherche la nourriture (La butineuse) est de trouver une source très riche de nourriture pour obtenir une quantité maximale de nectar. Et pour celà, il y a de différents types de ces abeilles :

- **Abeille en chômage:** il y a des abeilles n'ayant aucune connaissance sur la source de nourriture et aucune vue de source de nourriture à exploiter. Donc, ces abeilles restent dans la ruche et attendent la danse exercée par les autres abeilles pour établir la source de nourriture.

- **Abeille employée:** est associée avec une source de nourriture particulière. Elle a une connaissance suffisante pour connaître la source de nourriture, Elle trouve et exploite la source, mémorise la location et charge une partie de nectar et la décharge dans la ruche.

La danse est la phase oscillante. Pendant cette phase l'abeille aligne son corps de telle sorte que l'angle de déviation de la verticale est similaire à l'angle de l'objectif de l'azimut actuel du soleil la figure 2.2 montre cette notion sur la direction.

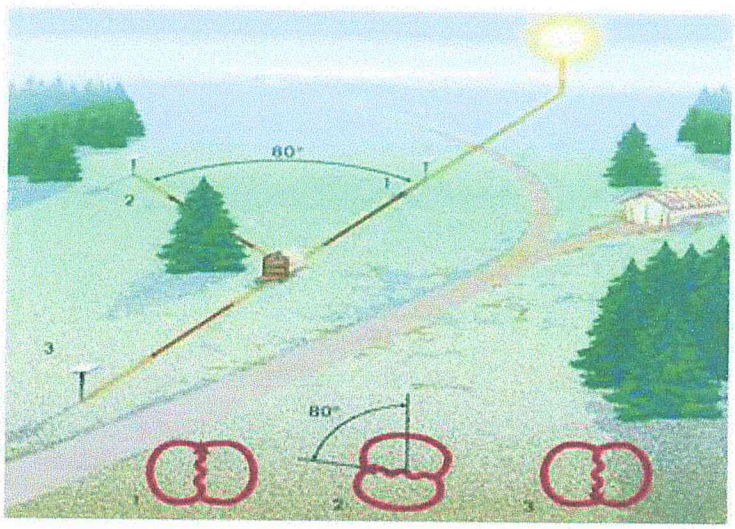


Figure 2.2. Direction du fourage chez les abeilles

L'information de distance est codée dans la durée de la phase oscillante. Danses pour les cibles proches ont des phases courtes frétillantes tandis que des danses pour les cibles éloignées ont des phases frétillantes prolongées. La figure 2.3 montre relation entre nombre du tour et distance

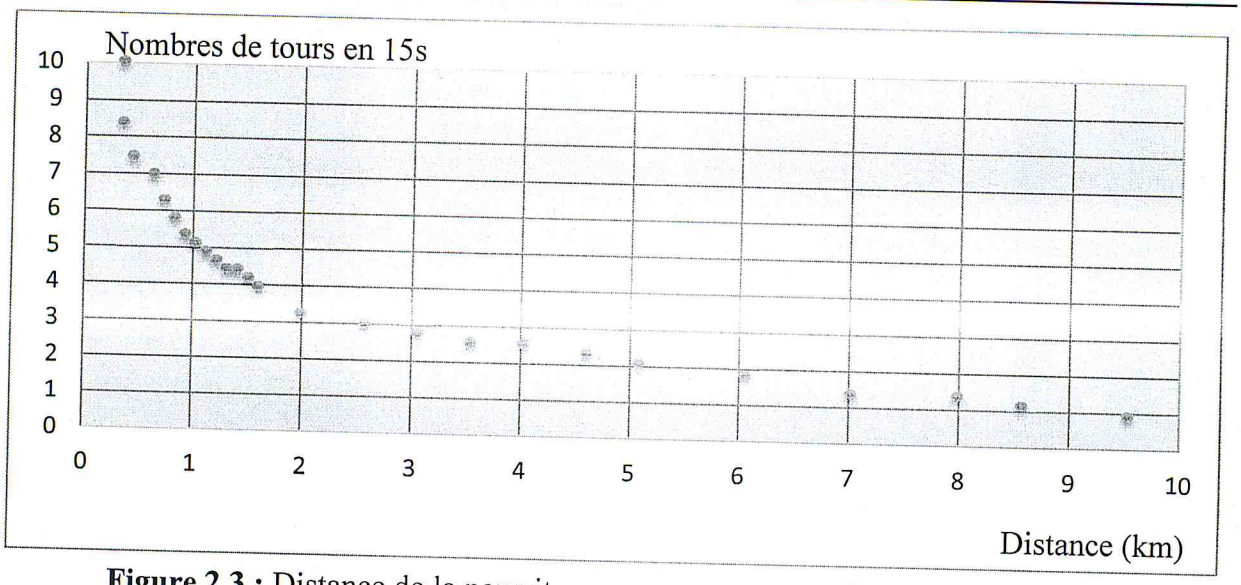


Figure 2.3 : Distance de la nourriture par rapport au nombre de tours de danse

Les abeilles mellifères modulent leur danse frétillante en fonction de la rentabilité de la source de nourriture trouvée. En conséquence, les abeilles dansent pour les sites très rentables à attirer des disciples de danse plus que ceux que la danse pour les sites médiocres. Le langage de la danse permet à une colonie d'abeilles de suivre les conditions sans cesse changeantes de recherche de nourriture (**Sentinelle, 2016).**

IV Abeilles artificielles

Comme il est indiqué, les approches actuelles d'optimisation d'inspiration des abeilles sont basées sur l'un des deux comportements trouvés chez les abeilles, butinages ou d'accouplement. Un aperçu est donné sur les Algorithmes utilisés par certains chercheurs qui serviront dans les prochains chapitres (Tableau 2.3).

Tableau 2.3. Algorithmes utilisés par certains chercheurs basés sur le comportement du butinages

Année	Auteurs	Algorithmes	Problème étudié
1997	Sato, Hagiwara	Bee System (BS)	Amélioration Algorithme Génétique
2001	Lucic, Teodorovic	BCO	Problème de Voyageur de commerce
2001	Abbas	MBO	Problèmes de satisfiabilité propositionnelle
2004	Wedde, Farooq, Zhang	BeeHive	Protocoles de routage
2005	Karaboga	ABC	optimisation numérique
2005	Yang	VBA	Optimisations de fonction avec l'application dans les problèmes d'ingénierie
2005	Benatchba, Admane, Koudil	MBO	Max-Sat problème
2006	Basturk and Karaboga	ABC	Optimisation de la fonction numérique

2006	Navrat	Bee Hive	recherche Web
2007	Koudil, Benatchba, Tarabetand, El Batou	MBO	Partitionnement et des difficultés de programmation
2007	Quijano, Passino	Honey Bee Social	Résoudre les problèmes d'allocation des ressources optimales
2007	Markovic, Teodorovic, Acimovic-Raspopovic	BCO	Routage et affectation de longueur d'onde dans les réseaux entièrement optiques
2007	Karaboga,Basturk	ABC	Tests ABC algorithme sur un ensemble de problèmes multidimensionnels d'optimisation Numérique
2007	Karaboga, Akay, Ozturk	ABC	Feed-forward de formation réseaux de neurones
2011	BOMBRUN, SENE	PSO	L'optimisation par essaim particulaire pour des problèmes d'ordonnancement
2012	N.Hanane	ABC	Conception d'un Classifieur Foul Utilisant Colonie D'abeille Pour Diagnostic Médical
2015	Y.Djenouri	BSO	La fouille et la reduction des règles d'association

1 Algorithme d'optimisation de colonie d'abeilles artificielle (ABC)

L'algorithme ABC (Artificiel Bee Colony) est développé par **Karaboga, Basturk (2007a et 2007b)**, en inspectant les comportements des abeilles réelles pour trouver la source de Nourriture, qui s'appelle le nectar, et partager l'information des sources de nourriture aux autres abeilles dans le nid.

Dans cet algorithme, les abeilles artificielle sont définies et classifiées en trois groupes : abeilles employeuses (abeilles qui recherche la nourriture), spectatrices (abeilles d'observation) et scouts (éclaireuses) sont chargées de trouver de nouvelles nourritures, (le nectar de nouvelles source).

Pour chaque source de nourriture, il y a seulement une abeille employeuse. C'est-à-dire, le nombre d'abeilles employeuses est égal au nombre de sources de nourriture.

Si l'abeille employeuse d'un site ne réussit pas à trouver la source de nourriture, elle doit être forcément devenir un scout pour rechercher aléatoirement de nouvelles sources de nourriture. Les abeilles employeuses partagent l'information avec les abeilles spectatrices dans une ruche de sorte que ces dernières puissent choisir une source de nourriture pour l'explorer. Le processus de l'algorithme ABC est présenté comme suit:

Etape 1: Initialisation:

On commence par sélectionner Fe pourcentage de population de façon aléatoire dans l'espace de recherche en utilisant l'équation suivante:

$$U_j = U_j^{min} + n_j * (U_j^{max} - U_j^{min}) \qquad n_j \in [0,1]$$

Sachant que chaque abeille porte un vecteur '*U*' de 'n' solution. A titre d'exemple le cas qu'on va simuler est expliqué comme suit:

$$U_{controle} = (P_i, V_i, T_i) \Rightarrow \left\{ \begin{array}{l} abeille_1 = (P_1, V_1, T_1) \\ abeille_2 = (P_2, V_2, T_2) \\ abeille_3 = (P_3, V_3, T_3) \\ \vdots \quad = \quad \vdots \\ abeille_1 = (P_i, V_i, T_i) \end{array} \right\}$$

Pi : puissance générée au jeu de barres *i*;
Vi : tension générée au jeu de barres *i*;
Ti : prises en charge des transformateurs;

Puis, on les évalue dans la fonction objective (équation de coût), ensuite on calcule leurs valeurs Fitness, appelées la quantité de nectar par l'équation suivante:

$$Fitness = \frac{1}{F_{objective}}$$

Fe représente le rapport des abeilles dans la population totale. Une fois que ces populations sont placées dans l'espace de recherche, elles prennent le nom : les abeilles employeuses.

Etape 2 : Déplacement des abeilles employeuses: Calculer la probabilité de choisir une source de nourriture par l'équation.

$$P_i = \frac{0.9 * Fitness_i}{\max(Fitness_i)} + 0.1$$

puis sélectionner une source de nourriture et ensuite déterminer ses quantités de nectar. L'équation de mouvement des abeilles observatrices est donnée ci-dessous:

$$m_{ij}(t+1) = x_{kj} + y\,(x_{ij}(t) - x_{kj}(t)) \qquad y \in [0.1]$$

Tel que *mij* est la *ieme* position de l'abeille spectatrice, *t* est le nombre d'itération, *xij* est l'abeille utilisée choisie aléatoirement, *'j'* représente la dimension du vecteur de solution qui produit une série de variables aléatoires dans la gamme [-1,1]; où *k*∈ 1,2,3……..,*N* et *j*∈ 1,2,……..,*D* sont choisis aléatoirement;
'*D*' est le nombre de paramètre à optimiser. '*K*' est aussi choisi aléatoirement mais doit être différent de l'indice '*i* '

Etape 3 : Déplacer les scouts : Si les valeurs de Fitness des abeilles employeuses ne sont pas améliorées par un nombre d'itérations prédéterminé, appelé "max-cycle", ces sources de nourriture sont abandonnées, et l'abeille trouvée dans cet emplacement passera aléatoirement pour explorer d'autres nouveaux emplacements. (Abeilles employeuses deviennent des Scouts). Cette explication est traduite mathématiquement par l'équation :

$$V_{ij} = V_{ij}^{min} + q_{ij} * (V_{ij}^{max} - V_{ij}^{min}) \qquad q_{ij} \in [0,1]$$

Étape 4 : Mettre à jour la meilleure source de nourriture trouvée jusqu'ici : Apprendre la meilleure valeur de Fitness et la position, qui sont trouvées par les abeilles, et les mémoriser.

Étape 5 : Critère d'arrêt Vérifier le processus de calcule jusqu'à ce que le nombre d'itérations atteigne la valeur maximale prédéfinie ou qu'une solution de la fonction objective acceptable soit trouvée.

2. Algorithme d'optimisation par colonie d'abeilles (BCO)

Dans **Mouassa (2012),** (Bee Colony Optimization) est introduit afin de trouver la solution optimale pour un problème d'optimisation combinatoire difficile donné, comme : le problème de voyageur de commerce, le problème de p-Médiane, problème de routage dans les réseaux optiques.

Chaque abeille génère une solution au problème. Il existe deux phases alternatives (le pas en avant et le pas en arrière) construisant une seule étape dans l'algorithme BCO.

Dans chaque pas en avant, chaque abeille artificielle visite NC solutions, crée une solution partielle, et ensuite retourne vers la ruche. Les abeilles se réunissent dans la ruche et commencent le pas en-arrière. Lorsque toutes les solutions sont complétées, la meilleure parmi elles est déterminée, et elle est utilisée pour mettre à jour la meilleure solution globale et comme ça une itération de BCO est accomplie. A ce point, toutes les solutions sont supprimées, et une nouvelle itération prend naissance. Soit '*B*' le nombre des abeilles dans la ruche, et 'NC' le nombre des déplacements constructifs en-avant. Au début de la recherche, toutes les abeilles sont dans la ruche. Le pseudo-code de l'algorithme BCO peut être décrit dans le **tableau 2.4**

Tableau 2.4. Algorithme BCO **(Mouassa 2012),**

Debut
Initialisation : une solution vide est assignée à chaque abeille ;
Tanque le critère d'arrêt n'est pas vérifié **faire**
Tanque les solutions ne sont pas complètes **faire**
Pour chaque abeille **faire** // (pas en-avant)
a. k = 1 ; // (compter les déplacements constructives en-avant)
Tanque k ≤NC
Evaluer tous les pas possibles;
Choisir un pas ;
k = k + 1 ;
Fin_tanque
Retour de toutes les abeilles à la ruche ; // (pas en-arrière)
pour chaque abeille évaluer la valeur de la fonction objective ;
Chaque abeille décide aléatoirement soit de continuer sa propre exploration et devenir une recruteuse, ou de devenir l'abeille qui fait larécolte ;

Pour chaque suiveur, choisir une nouvelle solution à partir des recruteuses.
Fin_tanque
Evaluer toutes les solutions et trouver la meilleure parmi elles.
Fin_tanque
Afficher la meilleure solution trouvée.
Fin.

3 Algorithme d'abeille basé sur d'autres comportements (MBO)

En plus du comportement de la recherche de la nourriture, dans **Mouassa (2012),** un autre algorithme est inspiré du processus biologique de reproduction des abeilles appelé MBO (Marriage in Honey Bees Optimization). L'optimisation par mariage d'abeilles est apparue en 2001. Pour initialiser l'algorithme MBO, cinq paramètres sont fixés:

1. nombre de reines,
2. nombre d'ouvrières,
3. nombre de couvées,
4. nombre de vols nuptiaux,
5. taille de la spermathèque de la reine.

Dans le processus de recherche, les reines représentent des solutions, tandis que les ouvrières représentent l'heuristique employée pour la recherche locale (amélioration). La taille de spermathèque représente le nombre d'accouplements par reine. Au début de l'algorithme, les ouvrières sont initialisées avec certaines heuristiques. Un ensemble de reines s'est produit aléatoirement et leurs génotypes se sont améliorés en utilisant une heuristique (ouvrière) pour préserver seulement les meilleures reines. Un ensemble de vols nuptiaux sont alors entrepris sachant que la vitesse et l'énergie de chaque reine sont initialisées aléatoirement pour s'assurer qu'elle volera pendant un certain nombre de fois. Les transitions effectuées par chaque reine sont en fonction de cette vitesse et cette énergie.

A chaque itération d'un vol, la reine s'accouple avec un bourdon rencontré durant sa trajectoire selon une probabilité. Si l'accouplement est réussi, le sperme du bourdon (génotype) est ajouté à la spermathèque de la reine. Quand toutes les reines ont terminé leurs vols, le procédé de création de couvée commence. Pour créer une nouvelle couvée, une reine est choisie selon le coût de son génotype, et le sperme est choisi aléatoirement à partir de la spermathèque de la reine, ensuite, on croise le sperme avec le génotype de la reine. La mutation est alors appliquée à la nouvelle couvée pour l'améliorer, en utilisant les ouvrières.

Les nouvelles couvées améliorées sont alors triées selon leur forme physique (fitness) et elles remplacent les reines de mauvaises qualités jusqu'à ce qu'il n'y ait aucune couvée meilleure que n'importe quelle reine. Les couvées restantes sont alors détruites et un nouveau vol nuptial est entrepris. Ceci est répété jusqu'à ce que tous les vols nuptiaux soient générés ou un critère d'arrêt soit vérifié (**tableau 2.5**).

Tableau 2.5. Algorithme MBO

```
Debut
    initialisation aléatoire de la reine améliorer la reine par les ouvrières en faisant une recherche
    local à partir de cette reine ;
    Tanque le nombre d'accouplement n'est pas atteint faire
        initialiser énergie et rapidité de la reine ;
        Tanque énergie de la reine > 0 faire
            La reine se déplace et choisie les mâles ;
            Si le male est selectionné alors
                Ajouter son spèrme dans l'ovaire de la reine ;
                Modifier l'énergie et la rapidité de la reine ;
            fin_si
        fin_tanque
        Produire les larves avec les mutations et les croisements ;
        Utiliser les ouvrieres pour l'occupation des larves en faisant une recherche locale à partir
        de chaque larve ;
        Modifier la fitness des ouvrières ;
        Si la meilleure larve adéquate la reine alors
            Remplacer les chromosomes de la reine avec les chromosomes de la meilleure
            larve ;
        fin_si
        Tuer toutes les larves ;
    fin_tanque
Fin.
```

4 Optimisation par colonie d'abeille (BSO)

Dans **Djenouri (2015),** La métaheuristique BSO (Bee Swarm Optimization) est proposée par **Drias et als (2005)** simule le comportement collectif des abeilles dans la recherche de la nourritue.

Dans un premier temps, une abeille appellée InitBee essaye de trouver une solution appelée Sref qui représente de bonnes caractéristiques. A partir de cette solution, on détermine

un ensemble de points de l'espace de recherche dont lesquelles les abeilles réalisent une recherche approfondie afin de trouver d'autres solutions plus efficaces que Sref.

Cet ensemble de solutions est appelé l'espace des régions. Il est calculé de telle sorte que les différents points sont très éloignés entre eux. Chaque abeille considère un point de l'espace des régions où elle effectue une recherche locale à partir de ce point. Après la terminaison d'exploitation des régions pour chaque abeille, les abeilles se communiquent entre elles dans le but de trouver la meilleure solution de toute la colonie, cette opération est effectuée grâce à la table Dance.

Par la suite, la meilleure solution est considérée comme la solution de référence Sref pour la prochaine itération. Afin d'éviter des cycles, d'autres terme éviter que les abeilles retournent à une solution déjà exploitée, dans chaque itération, la solution référence est sauvegardée dans une structure appelée taboo list. Cependant, si après un nombre d'itérations donné, les abeilles observent qu'il y'a aucune amélioration, elles introduisent la qualité de diversification. Ce dernier consiste à sélectionner à partir de taboo list, la solution la plus éloignée que la solution courante. L'algorithme se termine quand la solution optimale est trouvée ou bien un nombre maximal d'itérations est atteint, **tableau 2.6**.

Tableau 2.6. Algorithme général de (BSO) (**Djenouri 2015),**

```
Debut
    Sref=la solution trouvée par InitBee ;
    Tanque le critére d'arrêt non atteint faire
        insérer Sref dans taboo list ;
        espace-regions(Sref) ;
        affecter une solution de l'espace des régions pour chaque abeille ;
        Pour chaque abeille a faire
            recherche-locale(a) ;
            sauvegarder le résultat dans la table Dance ;
        fin_pour
        choisir la nouvelle solution référence Sref ;
    fin_tanque
Fin.
```

5 Algorithme BeeHive

Dans la nature, on distingue deux types d'abeilles, les abeilles à petite distance, il s'agit des abeilles qui cherchent leurs nourritures près de la ruche. D'autres abeilles appelées abeilles à longue distance, explorent d'autres régions loin de la ruche. L'algorithme BeeHive s'est inspiré de cette idée. En effet, l'ensemble des abeilles est divisé en deux sous-ensembles. Le premier sous ensemble applique une recherche locale près de la ruche, tandis que le deuxième applique une opération de diversification en explorant d'autres zones de recherche. A la fin de chaque itération, on sauvegarde les meilleures solutions trouvées par les deux types d'abeilles, tableau 2.7.

Tableau 2.7. Algorithme BeeHive **(Mouassa 2012)**

Debut
Diviser la population initiale en deux sous-ensembles (abeilles à petite distance et abeilles à longue distance) ;
Tanque le critére d'arrêt non atteint **faire**
Chaque abeille à petit distance applique une recherche locale à sa solution ;
Détermine la meilleure solution des abeilles à petite distance ;
Chaque abeille à longue distance explore sa région localement ;
Détermine la meilleure solution des abeilles à longue distance ;
Sauvegarder la meilleure solution de ces sous-ensembles ;
Fin_tanque
Fin.

Conclusion

Ces divers exemples montrent que l'approche d'intelligence en essaim est adaptée pour la modélisation de systèmes faisant intervenir un grand nombre d'entités différentes, en interaction, situées dans un environnement dynamique et fonctionnant de manière décentralisée.

C'est pourquoi, elle s'applique bien à la simulation de phénomènes collectifs dont les propriétés globales ne découlent pas directement des propriétés de ses composants, en y apportant une dimension explicative.

L'intelligence en essaim ne se limite pas à la représentation de comportements naturels, car elle est aussi appliquée à la gestion de trafics routiers urbains, à la simulation du mouvement de foule, à la distribution et la logistique ...

&&&&&

Chapitre III
Contribution

Introduction

L'optimisation des tournées de véhicules concerne l'optimisation des parcours d'un ou plusieurs véhicules destinés à rendre un service.

Dans notre cas, nous nous intéressons au problème de la grande distribution. Celui-ci est un problème réel que les vendeurs et les chauffeurs de camion rencontrent lorsqu'ils doivent livrer leur marchandise depuis le dépôt central jusqu'aux différents clients. Pour faire ces déplacements, on doit utiliser le meilleur chemin afin d'optimiser le parcours.

Le problème consiste à déterminer pour un ensemble donné de commandes de marchandise, dans quelle tournée du camion seront elles satisfaites, à quel moment dans la tournée et dans quel type de camion doit se faire la livraison ?

Notre travail consistera à proposer une solution au problème de tournée de véhicules en se basant sur la méthode d'optimisation par essaims d'abeilles et la méthode split pour le VRP « **V**ehicle **R**outing **P**roblem ».Ainsi, pour le faire, trois sections sont développées dans la suite de ce travail.

Premièrement, une présentation du problème lié aux contraintes, suivi de la méthode de résolution à base des essaimes d'abeille (Algorithme BSO) « **B**ee **S**warm **O**ptimization » qui est utilisée pour résoudre une partie de ce problème,on terminer avec les algorithmes VRP pour résoudre ce qui reste du problème d'optimisation.

I. Problématique

Le problème de livraisons général GDP « **G**eneral **D**elivery **P**roblem », est un problème d'optimisation consistant à trouver un ensemble de tournées, pouvant satisfaire un ensemble de clients demandant une certaine quantité de marchandises. Pour réaliser ces tournées, une flotte de différents types de véhicules (camion) est disponible.

Chaque demande de livraison (Command) se fait via un appel téléphonique ou une commande en ligne ou toutes les commandes sont centralisées dans un seul centre d'appel.

Chaque client est affilié à un secteur le plus proche, donc après chaque commande faite au niveau du centre d'appel, on peut connaître dans quel secteur et associer ce client, en fin de journée, toutes les commandes sont classées selon les secteurs et livrées la marchandise.

Pour ce faire, chaque demande de livraison (une commande) est caractérisée par une charge et un volume. Ce dernier possède un site d'origine (dépôt central) et une seule destination ou de livraison (client).

Lorsque plusieurs demandes de livraison sont prises en charge, elles peuvent être effectuées comme suit :

- Classement de toutes les commandes prises au centre d'appel ;
- Calcul de la totalité des marchandises de chaque région ;
- Sélection d'un camion pour envoyer l'ensemble de la marchandise;

- Classement des commandes par secteur et camion ;
- Livraison de chaque commande depuis le dépôt vers le client.

La figure 3.1 représente les différentes étapes du processus de commande.

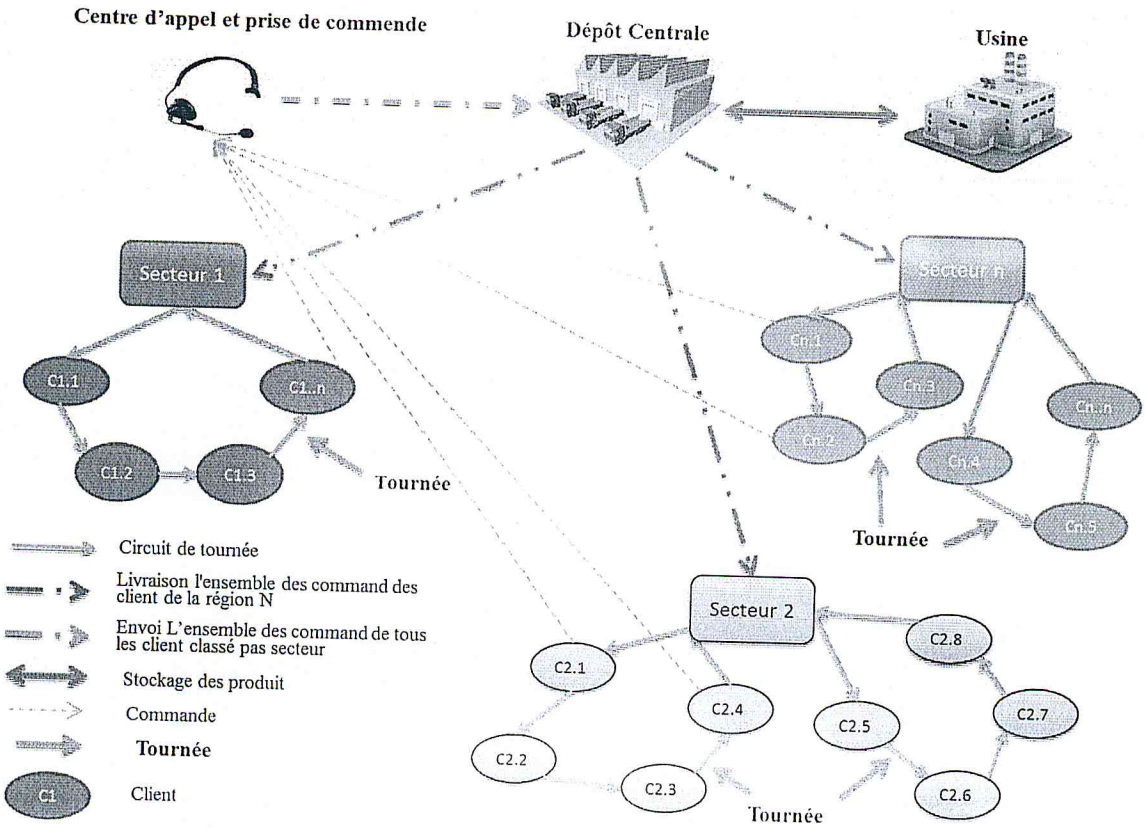


Figure 3.1. Processus de commande et de livraison à grande échelle

Ces problèmes ont de nombreuses applications parmi lesquelles nous trouvons, par exemple, la distribution de colis, le transport de personnes handicapées et le transport du personnel.

Le problème de livraisons général **(GDP)** est défini de la façon suivante :

- Soit n le nombre de demandes de marchandise ou de livraison (client).
- Chaque demande de marchandise i est caractérisée par une charge q_i à transporter de d'un point d'origines N_i^+ vers un ensemble des destinations N_i^-.

On suppose que cette charge est répartie sur les origines et les destinations de la façon suivante :

$$Q = \sum_{i \in n} q_i = \sum_{j \in N_i^+} q_j$$

Q : représente la charge totale.

De cette façon, le site de départ a une charge positive et les sites de livraison ont une charge négative.

Soit :

- $N^+ = \cup_{i \in n} N_i^+$ l'ensemble de toutes les origines, vue qu'on a une seule origine qui est le dépôt alors on mentionne juste N^+ car **n =1** ;
- $N^- = \cup_{i \in n} N_i^-$ l'ensemble de toutes les destinations ;
- $N = N^+ \cup N^-$.

Soit M l'ensemble des véhicules disponibles. Chaque véhicule $m \in M$ a une capacité maximale Q_m. Soit :

- $M^+ = \{m^+ \mid m \in M\}$ l'ensemble des véhicules du site de départ (dépôt) ;
- $M^- = \{m^- \mid m \in M\}$ l'ensemble des véhicules des sites d'arrivée (client) ;
- $M = M^+ \cup M^-$.

II. Déscription et contraintes liées au problème trait

Notre travail consiste à l'application de la méthode d'optimisation par essaims d'abeilles, au problème de tournée de véhicules, et plus particulièrement aux problèmes de la distribution.

Chaque jour des marchandises doivent être transportées du dépôt au client. Ces transports sont faits par des camions ayant une capacité limitée. L'objectif des distributeurs est de déterminer les tournées de façon à minimiser les coûts de transport par camion, tout en satisfaisant les demandes de livraison.

Nous disposons d'un cas réel de données sur une carte de l'Algérie d'un distributeur de boisson dans la wilaya de Boumerdès. On dispose de plusieurs types de camion qui sont différents au niveau de leur charge et chaque camion a une charge à respecter Q_i et chaque route possède une charge maximale qui peut la supportée Q_{max}.

Dans le tableau 3.1, nous représente un aperçu de quelque fiche technique de camions de référence.

Tableau 3.1 : Fiche technique de quelques camions

Type	Volume m^3	Capacité Kg	Surface m^2
Semi-remorque	27	33500	19.2
Camion Benne	2.9	3500	5.88
Camion Frigorifique	15	6000	11.23

III. Méthodes de résolution

Soit n le nombre de demandes de livraison à satisfaire, chaque commande k concerne un seul client qui a une marchandise a transporté d'un site d'origine qui est le dépôt k_p, à un site de destination k_d. Pour satisfaire les n commandes, plusieurs camions sont disponibles, la livraison se passe en deux étapes :

- Du dépôt central au secteur, la méthode de résolution des essaime d'abeille BSO « **B**ee **S**warm **O**ptimization» est utilisée.

- Du secteur au client final, la méthode « **SPLIT** » est utilisée afin d'optimiser le circuit de livraison pour le problème d'optimisation **VRP** « **V**ehicle **R**outing **P**roblem».

1. Première étape: du dépôt central au secteur

Dans le deuxième chapitre de ce mémoire, on a détaillé les algorithmes d'optimisation des essaims d'abeilles, l'algorithme BSO Bee Swarm Optimization) et proposé pour simuler le comportement collectif des abeilles dans la recherche de la nourriture.

Dans notre cas de la distribution, nous proposons d'adapter le comportement collectif pour résoudre le problème d'optimisation lors de la livraison et la distribution à grand échelle.

1.1. Algorithme général de BSO

L'algorithme est présenté dans le **tableau 3.2**.

Tableau 3.2. Algorithme général de BSO

Paramètres d'entrée/sortie :

S_{erf} : ensemble de solution de référence ;

Debut

S_{ref}=la solution trouvée par InitBee ;

Tanque le critère d'arrêt non atteint **faire**

insérer S_{ref} dans taboo list ;

espace-regions(S_{ref}) ;

Affecter une solution de l'espace des régions pour chaque abeille ;

Pour chaque abeille à **faire**

recherche-locale(a) ;

Sauvegarder le résultat dans la table Dance ;

fin_pour

Choisir la nouvelle solution référence S_{ref} ;

fin_tanque

Fin.

1.2. Amélioration de l'algorithme BSO pour la distribution

Dans un premier temps, on essaye de trouver une solution appelée S_{ref} qui représente de bonnes caractéristiques avec un vecteur S de n éléments. A partir de cette solution, on détermine un ensemble de trajets de l'espace de recherche dont lequel, chaque camion réalise une recherche approfondie durant son trajet afin de trouver d'autres solutions plus efficaces que S_{ref} Initiale.

Cet ensemble de solutions est appelé l'espace des régions, il est calculé de telle sorte que les différents chemins sont très éloignés entre eux. Chaque camion considère un point de l'espace des régions où il effectue une recherche locale à partir de ce point.

Après la fin d'exploitation des régions pour chaque camion, la meilleure solution est considérée comme la solution de référence $\mathbf{S_{ref}}$ pour la prochaine itération. Afin d'éviter des cycles, et d'éviter que les camions retournent à une solution déjà exploitée, dans chaque itération, la solution référence est sauvegardée dans une structure appelée tab_list.

Cependant, si après un nombre d'itérations donné, on n'observe aucune amélioration, on introduit la qualité de diversification. Le critère de diversification consiste de sélectionner à partir de tab_list, la solution la plus éloignée que la solution courante. L'algorithme se termine quand la solution optimale est trouvée ou bien un nombre maximal d'itérations est atteint.

En considère $\mathbf{t_i}$, un trajet à destination **d** tel que le vecteur **S** représente un ensemble de solutions de trajet référence.

Exemple :

Soit **S** vecteur de référence de **n** éléments, pour chaque autre élément ***i*** dans *S*, si :
S[*i*]=*j* tel que *j* > 0 si l'item *j* apparaît dans la i_{me} position de *S*.
S[*i*]= 0 s'il n'aura pas un item dans la i_{me} position de la solution *S*.

Considérons T={t1,t2,...,t10} est un ensemble de trajets :
S1 = {2; 4; 6; 3; 0; 0; 6; 7; 8} représente la solution S: t2; t4; t6 ; t3; t6; t7; t8.
S2 = { 0; 0; 5; 3; 0; 0; 0; 1} représente la solution S : t5; t3; t1.

1.2.1. Calcul de voisinage

Le calcul de voisinage est déterminé à partir de chaque solution ***S*** en ajoutant 1 ou bien en soustrayant 1 à chaque item de la solution **S**.
Considérons la solution suivante de l'exemple précèdent S = {2; 4; 6; 3; 0; 0; 6; 7; 8}.

- En ajoutant la valeur 1 du premier item de la solution S, on aura S1 = {3; 4; 6; 3; 0; 0; 6; 7; 8}.
- En soustrayant la valeur 1 du premier item de la solution S, on aura : S2 = {2; 4; 6; 3; 0; 0; 6; 7; 8}.
- En ajoutant la valeur 1 du second item de la solution S, on obtient : S3 = {2; 5; 6; 3; 0; 0; 6; 7; 8}.

1.2.2. Détermination des secteurs

On assume $\mathbf{S_{ref}}$ comme la solution référence trouvée par l'abeille initiale (Initbee). Dans notre cas de distribution, la détermination des secteurs se fait en deux étapes par rapport au trajet et la distance ainsi que la route que le camion doit traverser.

- **Premièrement**: un chemin confié à prendre qui sera l'initialisation de la solution.
- **Deuxièmement**: après chaque recherche locale qui se fait durant le trajet, si on trouve un chemin meilleur alors il sera mis à jour dans le vecteur S_{ref}.

1.3. Organigramme BSO pour la distribution

L'organigramme donne le déroulement général de la procédure BSO, (Fig 3.2).

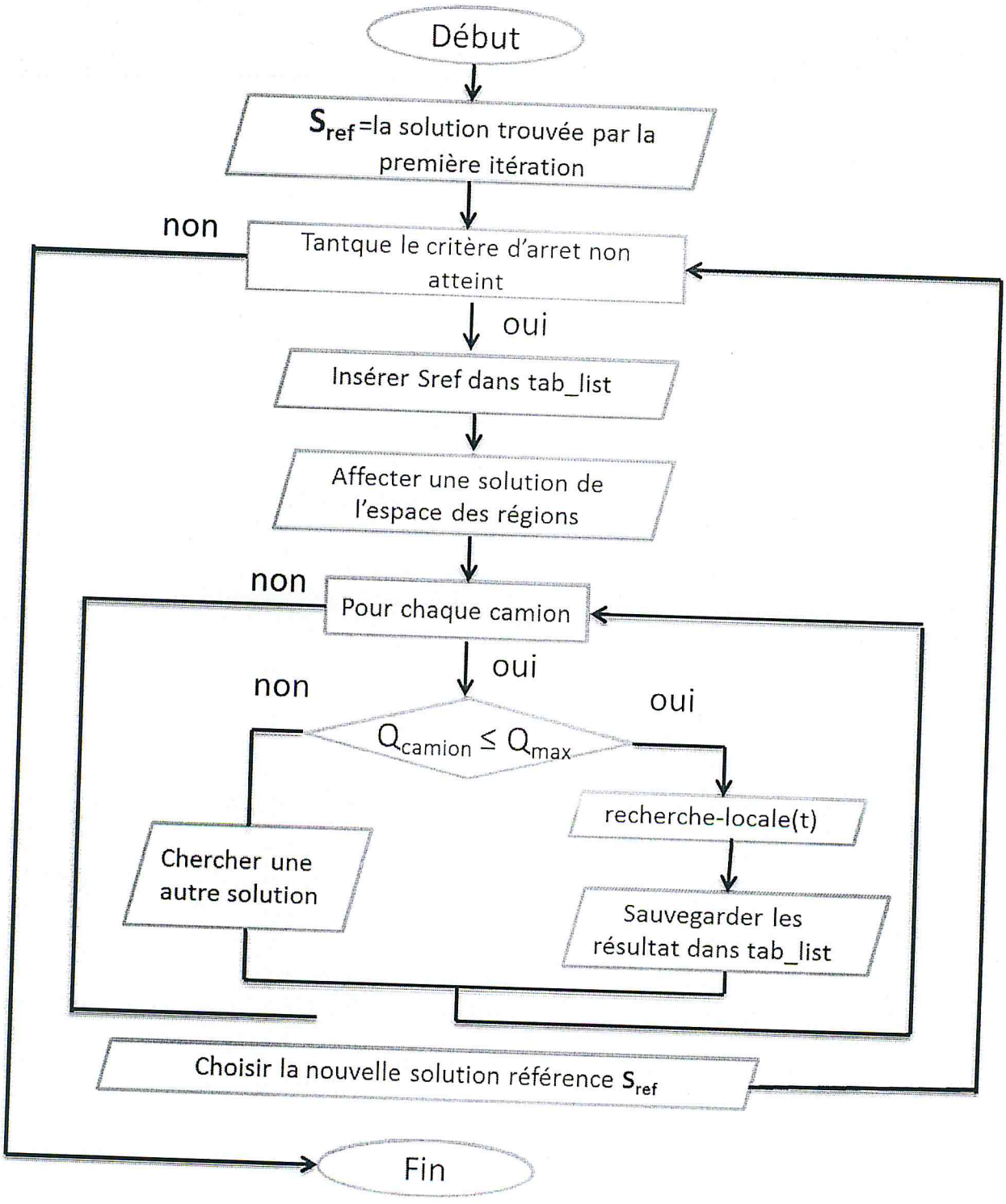


Figure 3.2. Organigramme de BSO

1.4. Algorithme BSO pour la distribution

Cet algorithme est présenté dans le tableau 3.3.

Tableau 3.3. Algorithme BSO pour la distribution

Paramètres d'entrée :

T : Trajet

Q_{max} : Ta charge maximale qu'en peut transporter sure une route X ;

Q_i : La charge transporté dans un camion **i** ;

D : Distance **;**

S_{erf} : Solution de référence.

paramètres de sortie :

tab_list : Ensemble de solution de référence.

Debut

S_{ref}=la solution trouvée par la première itération en considère que la charge et la distance sont pris en compte;

Tanque le critère d'arrêt non atteint **faire**

Insérer Sref dans tab_list ;

espace-regions(S_{erf}) ;

Affecter une solution de l'espace des régions pour chaque camion ;

Pour chaque camion à faire

Si $Q_{Camion} \leq Q_{max}$ **Alors**

recherche-locale($_{Camion}$) ;

Sauvegarder le résultat dans tab_liste;

Sinon

Chercher une autre solution ;

Fin si

fin_pour

Choisir la nouvelle solution référence S_{ref} ;

fin_tanque

return tab_list ;

Fin.

2. Deuxième étape : du secteur au client final (recherche locale intensive)

Dans cette étape, nous présentons la résolution de problèmes de type VRP qui consiste à définir une tournée pour chaque véhicule de manière à servir les clients. L'objectif est de minimiser la distance totale parcourue.

Nous allons proposer une heuristique SPLIT pour les VRP dans le cadre de la distribution.

Une telle approche s'inscrit pleinement dans une démarche de recherche scientifique où dernièrement une partie de ce problème a été déjà traité par (Phan et als 2009), (Brandão 2006) et (Ropke, Pisinger 2006).

2.1 Définition de la méthode SPLIT

La recherche s'effectue dans deux espaces: l'espace des tournées et l'espace des tours géants (obtenue par concaténation des tournées). L'heuristique donne une solution initiale sous forme de tournées. Elle est concaténée pour donner un tour géant afin d'appliquer les permutations. Le passage ensuite dans l'espace des tours géants vers l'espace des tournées s'effectue grâce à la méthode SPLIT **(Fig 3.3).**

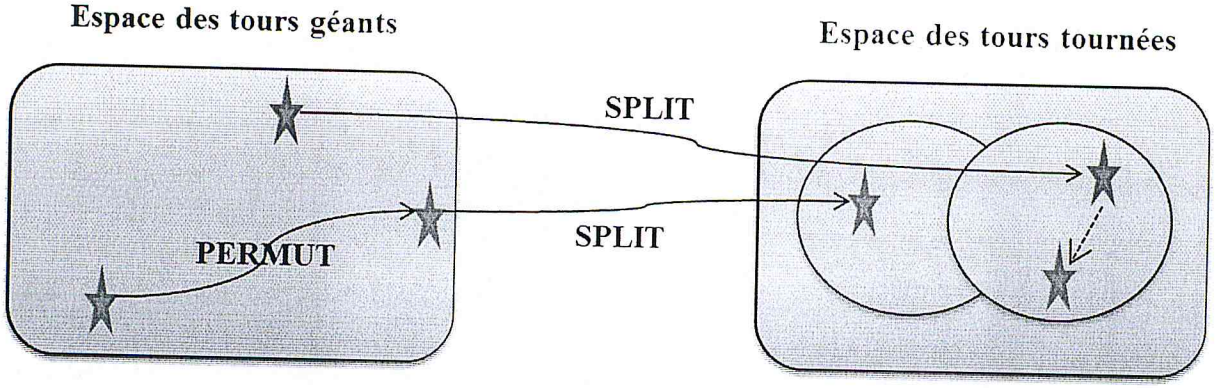


Figure 3.3. Schéma des deux espaces de travail

L'idée de "découper" un tour géant en un ensemble de tournées a été introduite par **Beasley (1983)** sous la forme de la méthode du route-first and cluster-second.
Ce principe a été repris par **Lacomme et als (2004)** pour un problème de tournées de véhicules sur arcs.

SPLIT est un algorithme de la programmation dynamique qui, à partir d'une séquence de clients, trouve un découpage optimal en tournées. Par exemple, lorsque le tour géant est de la forme 1 2 3 4 5 6, il est possible d'avoir un découpage en tournées : (1 2 3) et (4 5 6) mais pas en tournées : (1 2 3) et (5 4 6) puisque 5 et 4 ont été permutés.

2.2. Mise en œuvre de la méthode SPLIT

Considérons le tour géant : $\lambda = (0, \lambda_1, \ldots, \lambda_n)$ où λ_i représente le client en position **i** et **0** le dépôt. Le graphe $H \in (S', A', \lambda)$ se compose de **n+1** sommets (numérotés de 0 à n), et un

arc entre un sommet **i** et **j** représente la tournée composée des clients $\lambda_{i+1}, \ldots, \lambda_j$. Seuls les arcs correspondants à des tournées réalisables sont considérés comme un graphe orienté.

Dans le cas du VRP (flotte homogène sans limitation sur le nombre de véhicules disponibles), chaque nœud **i** du graphe **H** ne porte qu'un seul et unique label $\mathbf{L_i}$ qui représente le label au coût le plus faible pour traiter les clients λ_1 à λ_i. Le label $\mathbf{L_i}$ se définit donc par : $\mathbf{L_i.Cout}$, et par $\mathbf{L_i.(k, p)}$ qui donne le label père ici le label **k** du nœud **p**.

Remarquons que l'insertion d'un label sur un nœud ne se fait qu'en prenant en compte le coût du label de sorte que **L** domine **P** si et seulement si **L.Cout < P.Cout**. Dans les cas plus complexes (Heterogeonous VRP, Location Routing Problem etc...), un label comprend non seulement le coût, mais aussi les disponibilités des ressources utilisées.

Nous avons introduit une règle de dominance à 2 variables : le nombre de tournées et la distance totale. La relation d'ordre entre les labels n'est que partielle. Sur chaque nœud du graphe split **H**, les labels forment un front de Pareto, illustrés en figure 3.4.

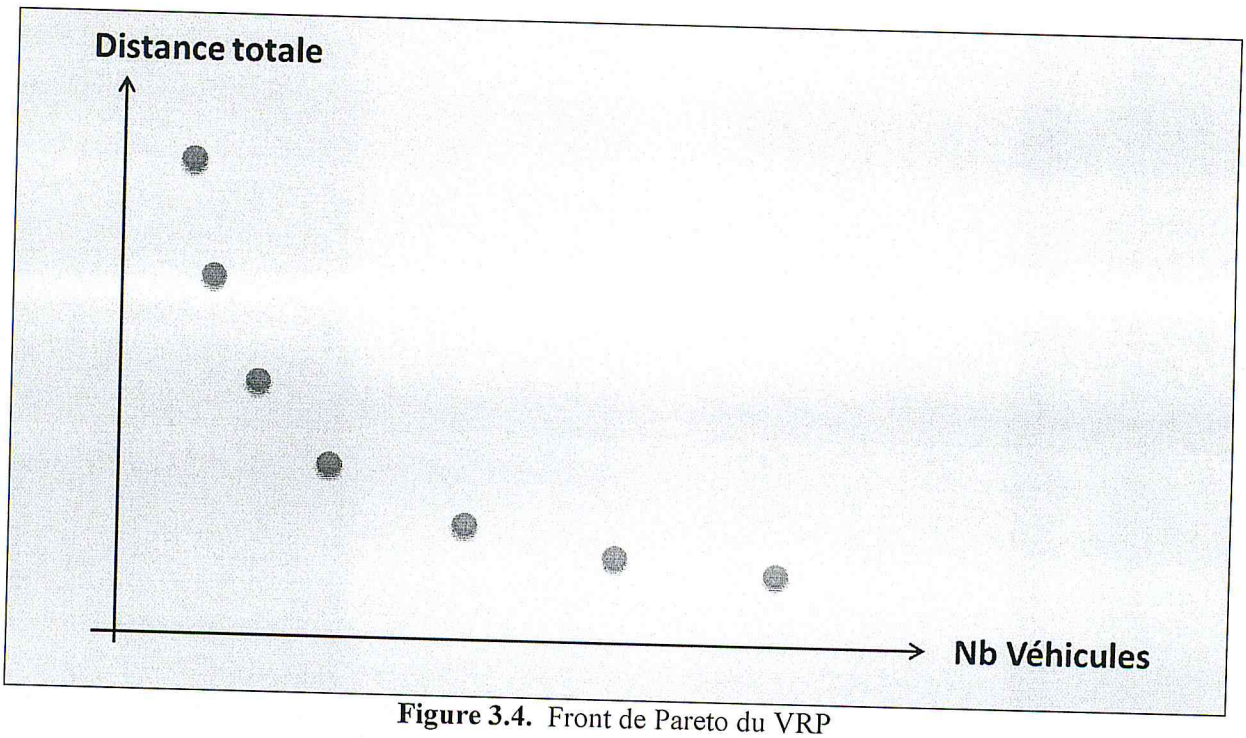


Figure 3.4. Front de Pareto du VRP

2.3. Organigramme de la procédure SPLIT

L'organigramme donne le déroulement général de la procédure SPLIT, (Fig 3.5).

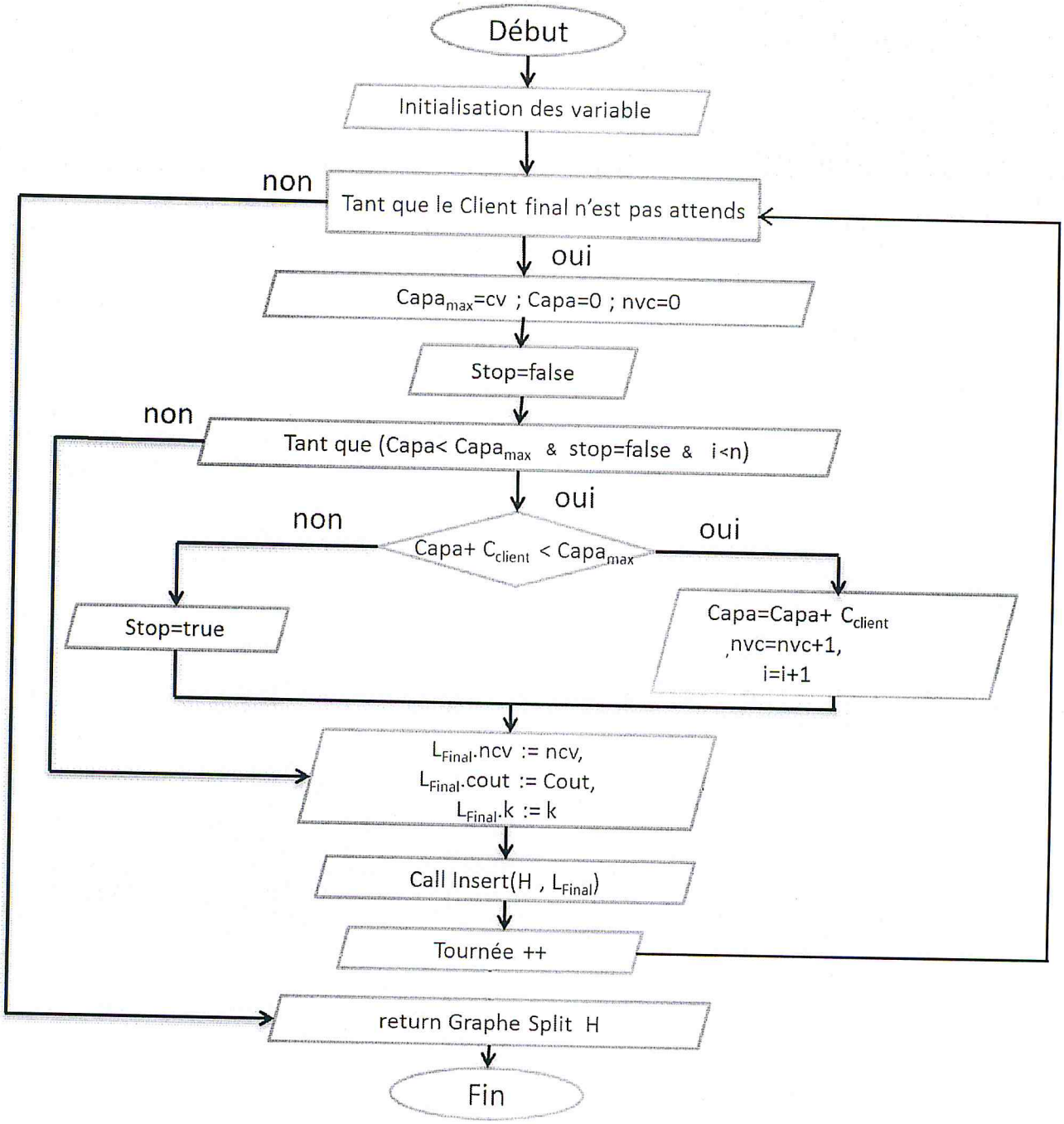


Figure 3.5. Organigramme de split

2.4. Algorithme de la procédure SPLIT

L'algorithme suivant, donne le déroulement général de la procédure SPLIT. (Tableau 3.4)

Tableau 3.4 Algorithme de la procédure SPLIT

Paramètres d'entrée :
λ: Tour géant
n : nombre de client
ncv : nombre de client pare véhicule ;
k : tournée ;
Paramètres de sortie :
H : Graphe split
Début
i = 0 ;// position du client
k = 0 ;
Tant que (i <= n)
// début d'une nouvelle tournée
$Capa_{max}$ = cv ;// capacité résiduelle = capacité du véhicule.
Capa= 0; // cout initialisé à 0
ncv = 0;
stop = false ;
Tant que (Capa < $Capa_{max}$,et stop = false && i < n)
Si (Capa+ $C_{\lambda.i}$ < $Capa_{max}$)
Capa= Capa+ $C_{\lambda.i}$; // capacité du client
ncv=ncv + 1 ;
i = i+ 1 ;
sinon
Stop := true ; // non respect de la capacité du véhicule
fin si
fin tant que
L_{Final}.ncv := ncv ;
L_{Final}.cout := Cout;
L_{Final}.k := k;
Call Insert(H , L_{Final}) ;
k=k+1;
fin tant que
return H
Fin.

Conclusion

Dans ce chapitre, nous avons défini les problèmes de base de la grande distribution et ses contraintes. Une étude approfondie, afin de résoudre les problèmes d'optimisation par les essaims d'abeilles et les VRP, a été entreprise. Toute une stratégie a été faite pour aboutir à un résultat fiable grâce à l'implémentation de la méthode SPLIT.

&&&&&

Chapitre VI
Implémentation

Introduction

Partant des résultats de la contribution qui nous a éclairés sur les différents axes de notre recherche, l'étape d'implémentation permet de définir la mise en œuvre finale du système.

Dans ce chapitre, on s'intéresse aux outils et à l'environnement du travail choisis pour l'implémentation et aussi à la réalisation et la mise en œuvre de l'application et la démonstration pour atteindre l'objectif du bon déroulement de l'entreprise.

I. Outils de réalisation de l'application de distribution

1. Base de données

Au niveau de la direction commerciale de la Sarl HYDROTRIZ, ils ont gèrent plusieurs distributeurs au niveau national. Pour cela notre choix a porté sur SQL Serveur. Ce dernier est conseillé pour des données qui sont très volumineuses.

➢ **SQl Server**

SQL Server est un système de gestion de base de données relationnelle **(SGBDR),** ce qui lui confère une très grande capacité à gérer les données tout en conservant leur intégrité et leur cohérence. SQL Server est chargé de stocker les données, vérifier les contraintes d'intégrité définies, garantir la cohérence des données qu'il stocke, même en cas de panne (arrêt brutal) du système, assurer les relations entre les données définies par les utilisateurs.

- **Avantages** SQL Server représente une très bonne solution à adopter par des applications de taille modérée, qu'elles soient professionnelles ou non. En effet, SQL Server offre une certitude d'héberger les données de façon optimum. De plus, il est possible de migrer vers une autre édition de SQL Server et ceci de façon transparente pour les applications travaillant avec les données. (Books Google 2016)

2. Application serveur

La gestion d'une distribution se fait selon un processus de développement lourd qui assure les différents outils d'exploitation et de manipulation des données aussi il offre une architecture qui permet aux développeurs de faire des opérations sur le code de l'application sans la reprendre du début.

Ainsi, nous avons choisi de travailler avec J2EE « Java Entreprise Edition ». L'impression des documents et leur mise en page est conçu grâce à « Jasper Ireport » qui supporte le langage « JAVA »

➢ **Java Entreprise Edition (J2EE)**

Dans les travaux de (Ferroukhi, Benali 2015), le terme « Java » fait bien évidemment référence à un langage, mais également à une plate-forme : son nom complet est « Java SE ». Celle-ci est constituée de nombreuses bibliothèques, ou API : citons par exemple java.lang, java.io,java.math, java.util, … etc.

Toutes ces bibliothèques contiennent un nombre conséquent de classes et de méthodes prêtes à l'emploi pour effectuer toutes sortes de tâches.

Le terme « Java EE » signifie Java Enterprise Edition, et était anciennement raccourci en « J2EE ». Il fait, quant à lui référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plate-forme Java SE, et elle y ajoute un grand nombre de bibliothèques remplissant tout un tas de fonctionnalités que la plate-forme standard ne remplit pas d'origine.

L'objectif majeur de Java EE est de faciliter le développement d'applications WEB robustes et distribuées, déployées et exécutées sur un serveur d'applications.

J2EE est particulièrement destiné aux applications d'entreprise. Ces applications sont considérées dans une approche multi-niveaux.

Le J2EE utilise une architecture multi couches basé sur le modèle ou l'architecture «MVC » qui est définit dans la figue 4.1.

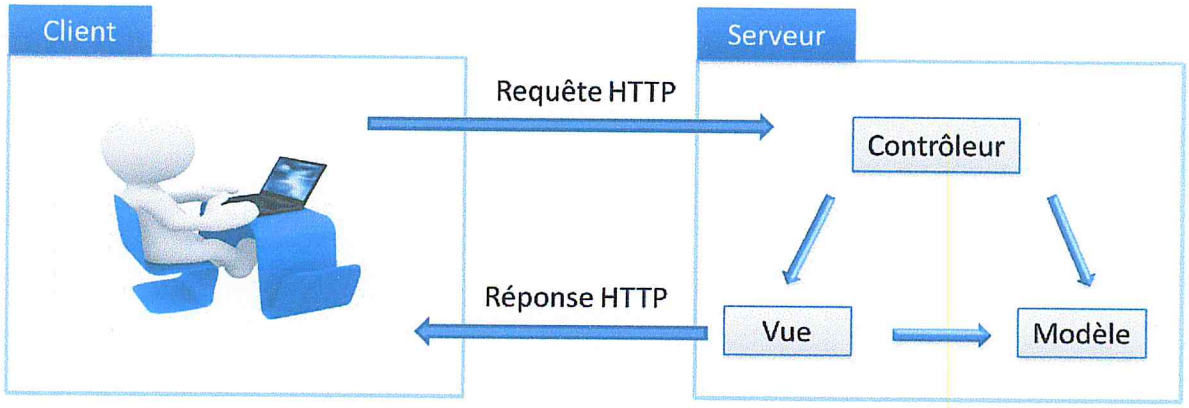


Figure 4.1 : Architecture MVC

L'architecture du modèle « MVC » définit dans la figure.4.1 explique qu'une fois une requête HTTP est envoyée par le client au serveur, le contrôleur accueille cette requête et il la traite après qu'il utilise des traitements qui sont faits au niveau du modèle, et il leur retourne les résultats de ces traitements. Le contrôleur inclut ces résultats en vue qu'il répondre aux besoins de la requête du client qui correspondant.

Le J2EE utilise des framework MVC tel que JSP (Java Server pages), JSF (Java server faces), Spring. Sont des framework J2EE les plus connus et les plus utilisés.
Au niveau de l'étude, Le travail est réalisé suivant la technologie JSF

➢ **Java server faces « JSF »**

Le design pattern ou l'architecture de notre framework est toujours MVC où les couches de notre architecture MVC sont comme suit :

✓ **Modèle : des traitements et des données**

Dans le modèle, on trouve à la fois les données et les traitements à appliquer à ces données. Ce bloc contient donc des objets Java d'une part, qui peuvent contenir des attributs (données) et des méthodes (traitements) qui leur sont propres, et un système capable de stocker des données d'autre part. Rien de bien transcendant ici, la complexité du code dépendra bien évidemment de la complexité des traitements à effectuer par votre application.

✓ **Vue : des pages faces**

Une page face est destinée à la vue. Elle est exécutée côté serveur et permet l'écriture de gabarits (pages en langage "client" comme HTML, CSS, Javascript, XML, etc.). Elle permet au concepteur de la page d'appeler de manière transparente des portions de code Java, via des balises et expressions ressemblant fortement aux balises de présentation HTML.

✓ **Contrôleur : des servlet et implémentation spring security**

Une servlet est un objet qui permet d'intercepter les requêtes faites par un client, et qui peut personnaliser une réponse en conséquence. Il fournit pour cela des méthodes permettant de scruter les requêtes HTTP. Cet objet n'agit jamais directement sur les données, il faut le voir comme un simple aiguilleur : il intercepte une requête issue d'un client, appelée éventuellement des traitements effectués par le modèle, et ordonne en retour à la vue d'afficher le résultat au client.

Implementation Spring Security offre des services de sécurité complets pour les applications logicielles d'entreprise basées sur J2EE. Il y a un accent particulier sur le soutien de projets construits en utilisant le cadre de printemps, qui est la solution leader de J2EE pour le développement de logiciels d'entreprise, **(Spring, 2016)**.

- **Avantage du spring security**
  - Il est très largement utilisé dans le monde Java.
  - Très bonne intégration avec des frameworks open source (Struts, Hibernate, ...) ou des standards de Java (Servlets, JMS, JDO, ...).

➢ **Maven**

Maven est un outil de construction de projets (build) développé par la fondation Apache, il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java, **(Maven, 2016).**

- **Avantage :**
  - Il existe des plugins sur les plateformes : eclipse, netbeans.
  - Un répertoire lib riche, qui contenait toutes les librairies.
  - Open source.

3. Serveur « Apache Tomcat »

Pour faire fonctionner une application web Java EE, nous avons besoin de mettre en place un serveur d'applications. Il existe beaucoup de serveurs d'application J2EE tel qu'Apache TOMCAT, GLASSFISH ...
Dans notre projet, on a utilisé « apache tomcat »

- **Avantage** :
  - Apache Tomcat est une implémentation open source d'un conteneur web, qui permet d'exécuter des applications web reposant sur les technologies servlets.

4. Impression des documents « Jasper Ireport »

Pour la mise en œuvre des documents imprimés, Jasper Reports est un outil de reporting open source, offert sous forme d'une bibliothèque qui peut être embarquée dans tout type d'applications Java.

Jasper Reports se base sur des fichiers « XML» (dont l'extension est en général .jrxml) pour la présentation des états. Il peut être couplé à iReport ou Jasper Studio (plugin Eclipse équivalent) pour faciliter sa mise en œuvre dans une application Java, classique ou orientée web (Jasper-ireport 2016).

5. Application mobile

Pour faciliter le trajet et la distribution, une application mobile Android a été faite afin d'aider les chauffeurs ou les vendeurs de remettre la marchandise dans les délais et aux bons endroits sans de se tremper dans la tournée.

La plate-forme ANDROID est un système d'exploitation OPEN SOURCE pour terminaux mobiles, Smartphones et PDA, conçu pas la startup du même nom. Ce dernier est basé sur le noyau linux et utilise la plateforme java pour ses applications.

- **Avantages:**
  - Bibliothèque innovante.
  - Facilités de développement.
  - Exécution rapide.

6. Communication socket

La communication entre l'application mobile et l'application serveur de distribution est fait par des socket. Ce dernier est une structure de données abstraite qui est utilisée pour établir un canal de communication permettant l'envoi et la réception d'informations entre des processus qui s'exécutent dans un environnement distribué, (Stephane ,2016).

- **Avantages :**
  - Consomme peu de ressources.
  - Très rapide.
  - Ne nécessite pas de rafraîchissement de page.
  - Ne nécessite pas de base de données.

7. Géolocalisation, APIs « Googles maps »

Dans **Touaibia (2014)**, API Googles Maps permet de localiser tout type de données sur une carte (routière, satellite, mixte) à partir de son adresse postale ou ses coordonnées (latitude et longitude). 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...).

Les résultats sur la carte apparaissent sous la forme d'une petite icône cliquable. En cliquant sur cet icône, une fenêtre s'ouvre dans laquelle se trouve toutes les informations concernant la donnée géolocalisée.

Il est également possible de calculer un itinéraire (piéton ou voiture) depuis un point de départ jusqu'à un point d'arrivée.

II. Démonstration de l'application de distribution

1. Modèle physique de données

Dans la figure **4.2,** nous présentons le modèle physique de données de l'application.

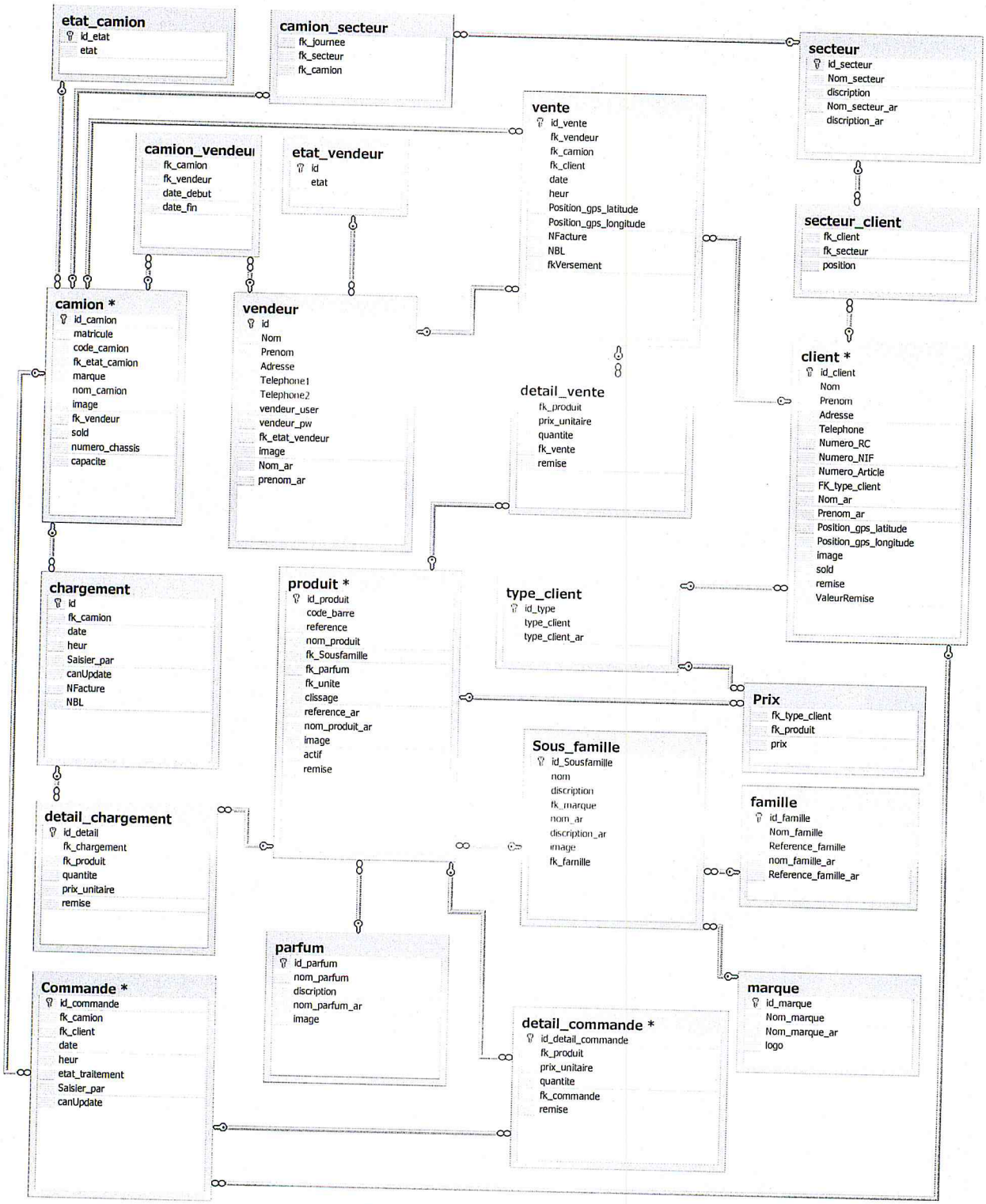


Figure 4.2. Modèle physique de données

2. Implémentations des algorithmes

Dans cette partie nous figurent les Pseudos code développés pour l'optimisation des trajets et l'affectation des circuits.

2.1. Implémentation de l'Algorithme BSO

Tableau 4.1. Pseudo code BSO en JAVA

```
public Trajet BSO(List<Trajet> trajets) {
    Trajet new_trajet = recherche_locale();
    if (new_trajet.getQ() < new_trajet.getQmax()) {
       trajets.add(new_trajet);
    }
    return Serf(trajets, new_trajet.getQ());
  }
```

Tableau 4.2. Fonction Serf qui cherche la solution de référence

```
public Trajet Serf(List<Trajet> trajets, double charge) {
    if (trajets != null) {
       serf = trajets.get(0);
       dis_min = Calcule_dist(trajets.get(0));
       for (Trajet trj : trajets) {
          if (charge < trj.getQmax()) {
             double dist_trj = Calcule_dist(trj);
             if (dist_trj < dis_min) {
                serf = trj;
                dis_min = dist_trj;
       } } }
       return serf;
    }
    return null;
  }
```

Tableau 4.3. Fonction qui calcule la distance d'un trajet

```
public double Calcule_dist(Trajet trajet) {
    Double distance = 0.0;
    if (trajet.getLatLng() != null) {
       LatLng init_pos_latlng = trajet.getLatLng().get(0);// intialiser par la premier position
GPS
       for (LatLng latlng : trajet.getLatLng()) {
          distance= distance + Math.acos(Math.sin(init_pos_latlng.getLat()) *
Math.sin(latlng.getLat()) + Math.cos(init_pos_latlng.getLat()) * Math.cos(latlng.getLat()) *
Math.cos(init_pos_latlng.getLng() - latlng.getLng()));
          init_pos_latlng = latlng;
       }
    }
    return distance;
  }
```

2.2. Implémentation de l'Algorithme Split

Tableau 4.4. Pseudo code Split en JAVA

```
public Graphe Split(List< Client> clients, List<Camion> camions) {
      Graphe list_tournee = null;
      Camion camion = null;
      Client client;
      if (clients != null & camions != null) {
         int id_tournee = 0;
         int id_camion = 0;
         int i = 0;
         while (i < clients.size()) {
            Tournee new_tournee = new Tournee();
            new_tournee.setId(id_tournee);
            if (id_camion < camions.size()) {
               camion = camions.get(id_camion);
               boolean stop = false;
               while (new_tournee.getCapa() < camion.getCapa() && stop == false && i <
clients.size()) {
                  client = clients.get(i);
                  if (new_tournee.getCapa() + client.getCapa() < camion.getCapa()) {
                     new_tournee.clients.add(client);
                     new_tournee.setCapa(new_tournee.getCapa() + client.getCapa());
                     new_tournee.addNbr_client();
                     i++;
                  } else {
                     stop = true;
                  }
               }
               new_tournee.setCamion(camion);
               id_camion++;
               list_tournee.addTours(new_tournee);
               id_tournee++;
            } else {
               return list_tournee;
            }
         }
      }
      return list_tournee;
   }
```

3. Application web de distribution

Quand l'utilisateur dans de site, l'application sera redirigée automatiquement vers la page de l'authentification. La figure 4.3 représente la page d'authentification de l'application afin que l'utilisateur entre son nom et le mot de passe.

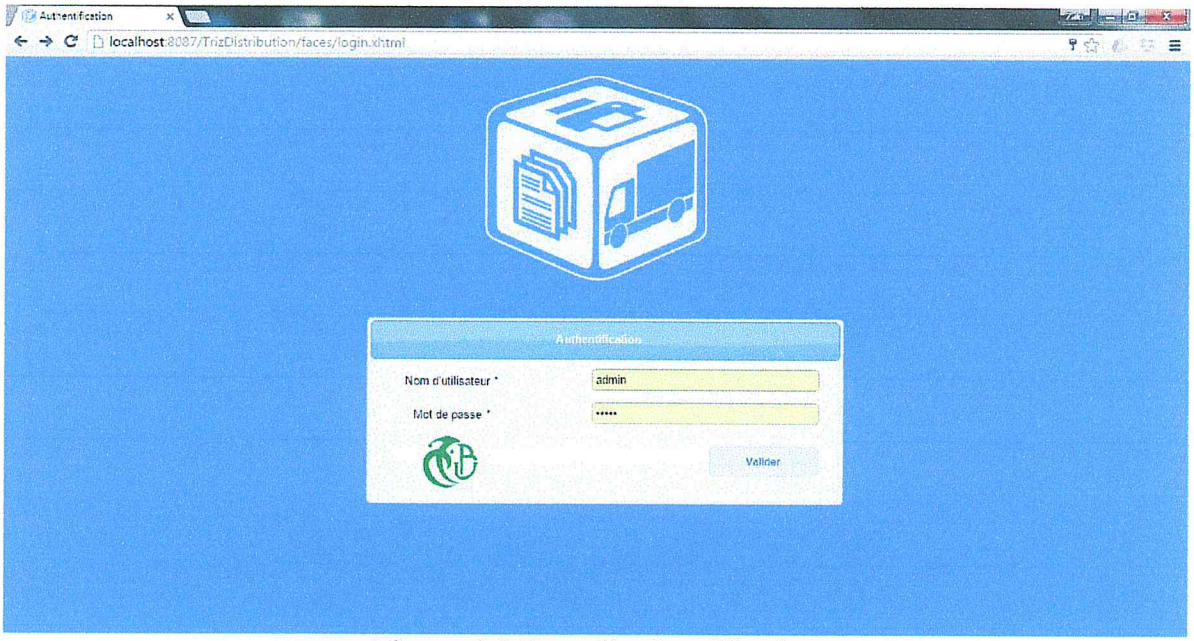


Figure 4.3. Page d'authentification

Après l'authentification de l'utilisateur, une page d'accueil sera affichée selon les permissions autorisées, L'administrateur à la possibilité de voir toutes les sections de l'application. La figure 4.4 représente la page d'accueil.

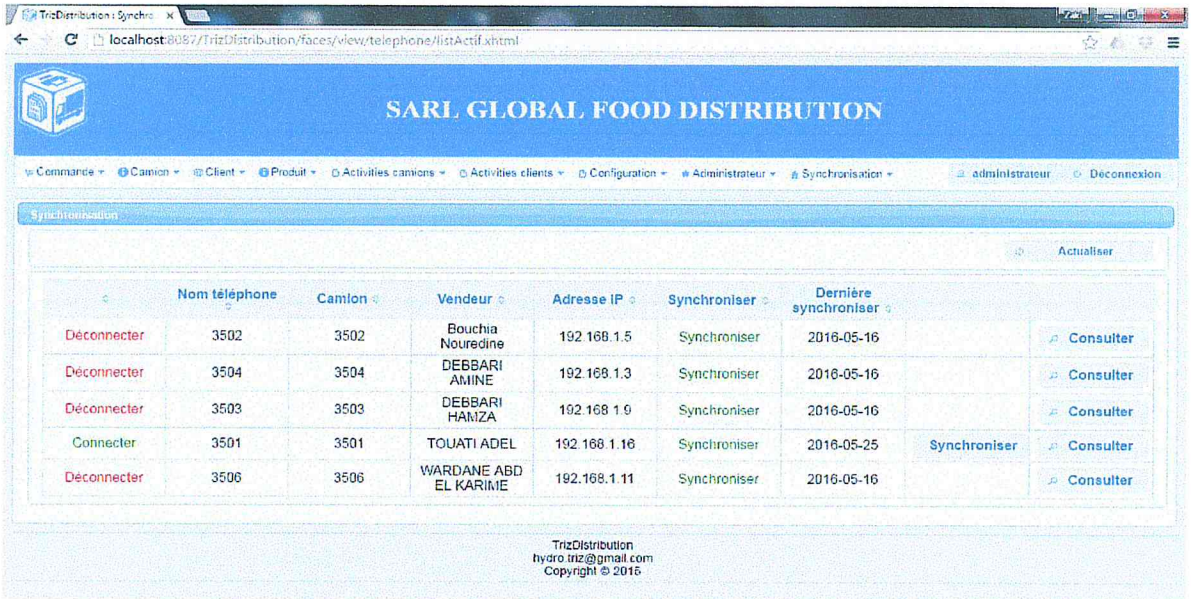


Figure 4.4 Page d'accueil de l'application.

Dans la page d'accueil, on a un tableau sur tous les vendeurs/chaufeurs. Cela nous permis de voir la synchronisation avec l'application mobile. Chaque vendeur/chaufeur possède une application mobile pour qu'il puisse suivre la tournée et faire sa livraison au bon endroit.

Dans la page d'accueil, on trouve une barre d''accueil. Cela nous permet de faciliter la navigation dans les fonctionnalités de l'application (Fig.4.5).

Figure 4.5. Barre d''accueil.

Ce qui suit un aperçu d'une prise de commande est donné **(Fig 4.6).**

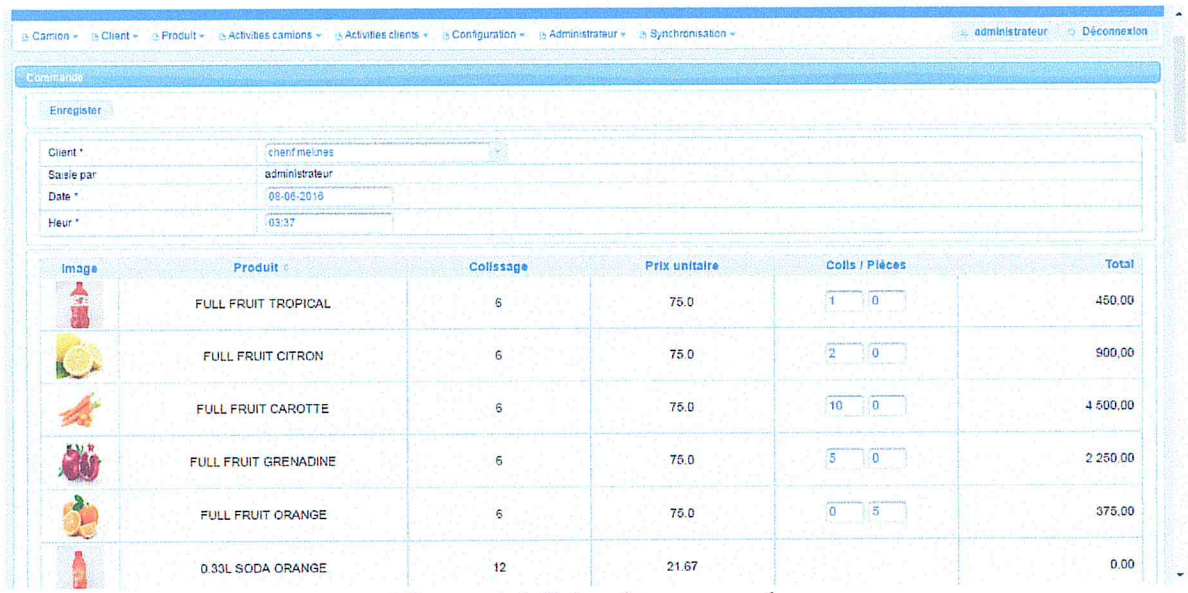


Figure 4.6. Prise de commande.

Dans la figure 4.7, on peut voir la liste de tous les clients.

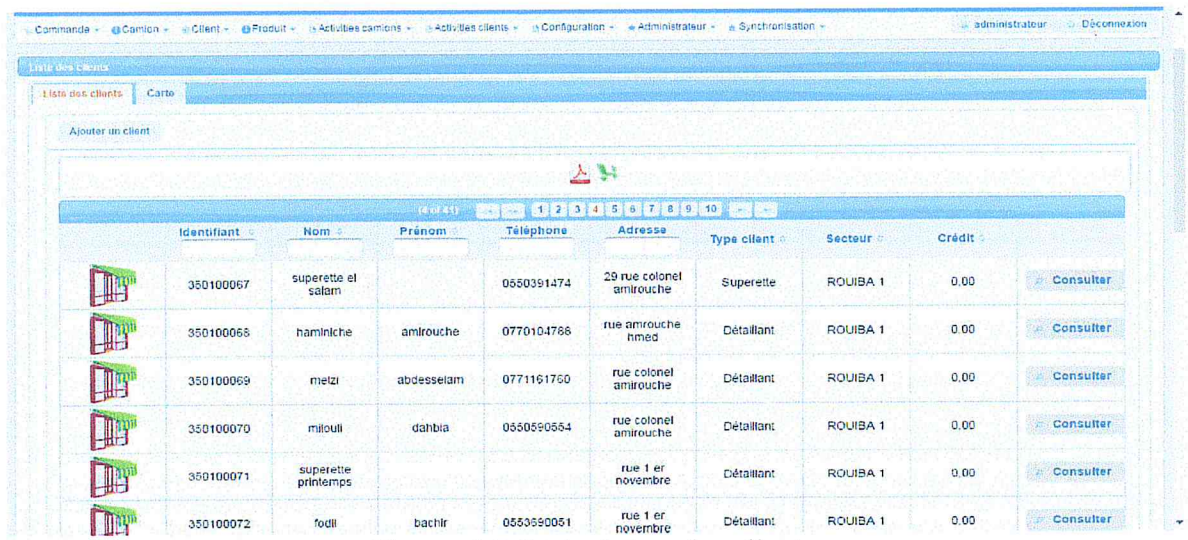


Figure 4.7. Liste de tous les clients.

Pour résoudre le problème de VRP, on fait pour chaque client une position GPS. Dans la figure 4.8, on peut voir chaque client dans la carte et déduire les secteurs.

Figure 4.8. Géolocalisation des clients.

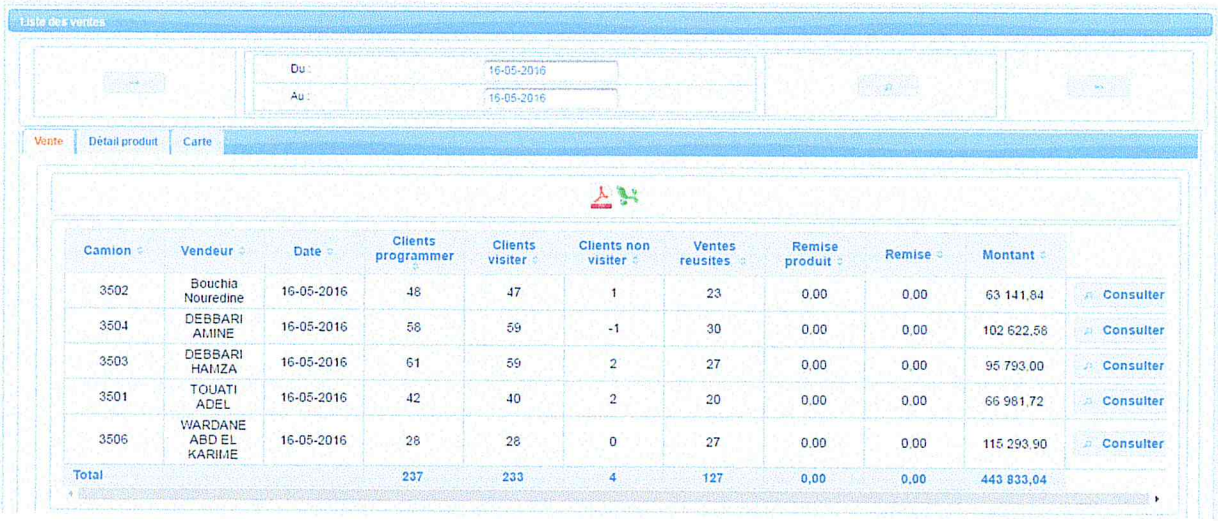
A partir de cela, on peut déterminer la tournée qu'on doit faire, la figure **4.9** illustre les clients classés pare tournée par différentes couleurs.

Figure 4.9. Client classé pare tournée.

Chaque client est positionné dans la tournée et on peut ajouter aussi que chaque client est affecté à un secteur afin que l'algorithme de détermination de régions puisse être appliqué.

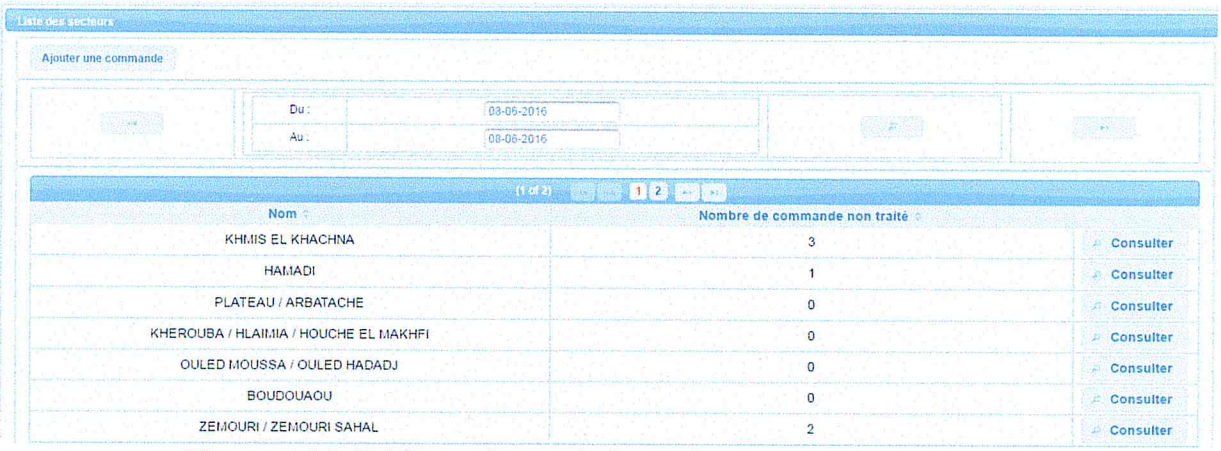
La figure 4.10 illustre les détails du client et sa position dans la tournée. Cette position peut servir au chauffeur grâce à l'application mobile, de connaitre le prochain client qu'il doit leur livrer la marchandise.

Aussi dans la figure 4.11 en peut voir les statistiques des commandes non traité



Camion	Vendeur	Date	Clients programmer	Clients visiter	Clients non visiter	Ventes reusites	Remise produit	Remise	Montant	
3502	Bouchia Nouredine	16-05-2016	48	47	1	23	0,00	0,00	63 141,84	Consulter
3504	DEBBARI AMINE	16-05-2016	58	59	-1	30	0,00	0,00	102 622,56	Consulter
3503	DEBBARI HAMZA	16-05-2016	61	59	2	27	0,00	0,00	95 793,00	Consulter
3501	TOUATI ADEL	16-05-2016	42	40	2	20	0,00	0,00	66 961,72	Consulter
3506	WARDANE ABD EL KARIME	16-05-2016	28	28	0	27	0,00	0,00	115 293,90	Consulter
Total			237	233	4	127	0,00	0,00	443 833,04	

Figure 4.10 Tableaux des statistiques de tournée.



Nom	Nombre de commande non traité	
KHMIS EL KHACHNA	3	Consulter
HAMADI	1	Consulter
PLATEAU / ARBATACHE	0	Consulter
KHEROUBA / HLAIMIA / HOUCHE EL MAKHFI	0	Consulter
OULED MOUSSA / OULED HADADJ	0	Consulter
BOUDOUAOU	0	Consulter
ZEMOURI / ZEMOURI SAHAL	2	Consulter

Figure 4.11. Tableaux des statistiques des commandes non traité.

4. Application mobile de distribution

L'application mobile permet de suivre le chauffeur/vendeur durent toute sa tournée. En figure 4.12, une page d'authentification de l'application mobile afin que l'utilisateur peut entrer dans son compte s'affiche. L'application est bilingue (français & arabe) afin que l'utilisateur peut surfer dans l'application sans aucune difficulté.

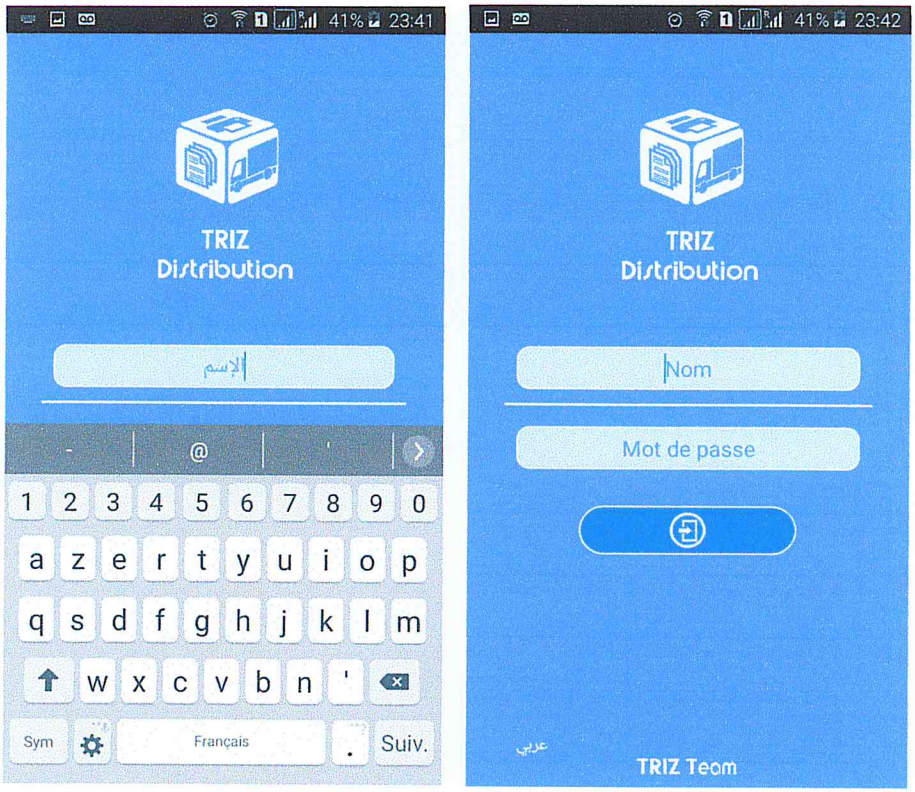


Figure 4.12. Page d'authentification de l'application mobile

L'application possède un menu simple facile à manipuler (Fig 4.13).

Figure 4.13. Menu de l'application mobile.

La figure 4.14 illustre le classement des clients dans la tournée, les clients sont aussi classés par date de tournée.

Figure 4.14. Classement des clients dans la tournée

Les clients sont synchronisés avec l'application Web, et l'utilisateur peut voir leur localisation GPS dans la Carte. L'application permet de voir la carte en deux modes, soit en mode satellite ou en mode plan (normal), (Fig.4.15).

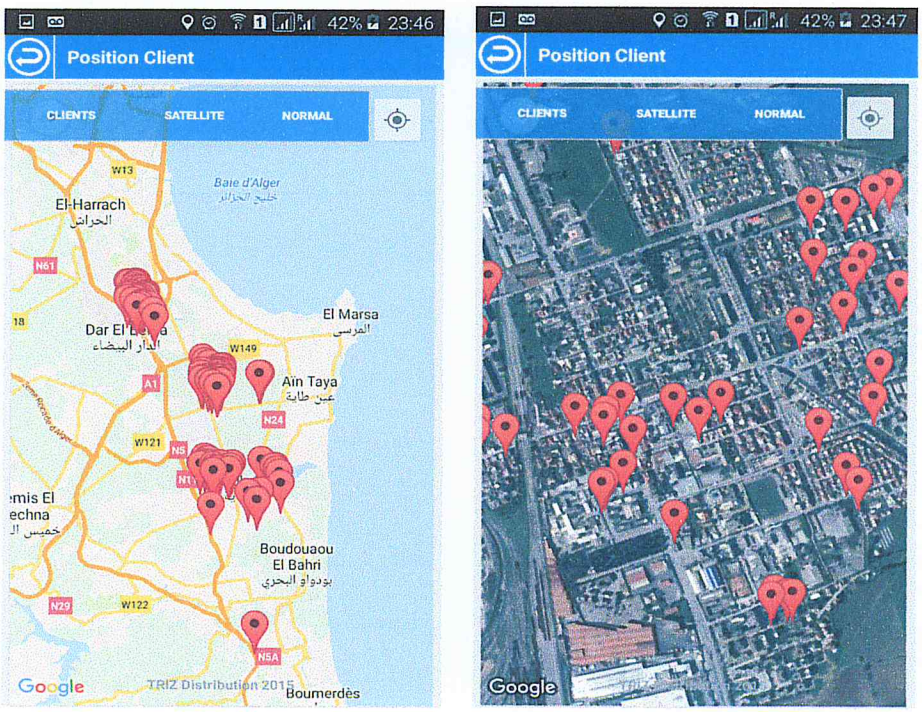


Figure 4.15. Position GPG des clients sur la carte.

On peut aussi voir les positions GPS des clients par secteur **(Fig4.16).**

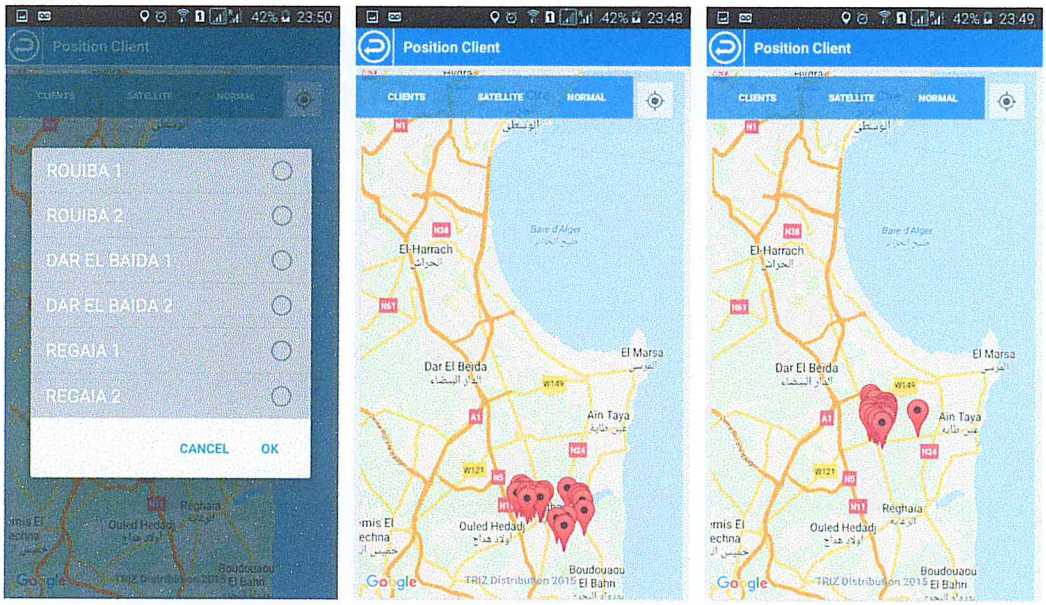


Figure 4.16. Position GPG des clients par secteur.

L'application mobile offre la possibilité de voir la route à prendre pour arriver au prochain client (Fig.4.16). La carte peut être affichée en deux modes aussi.

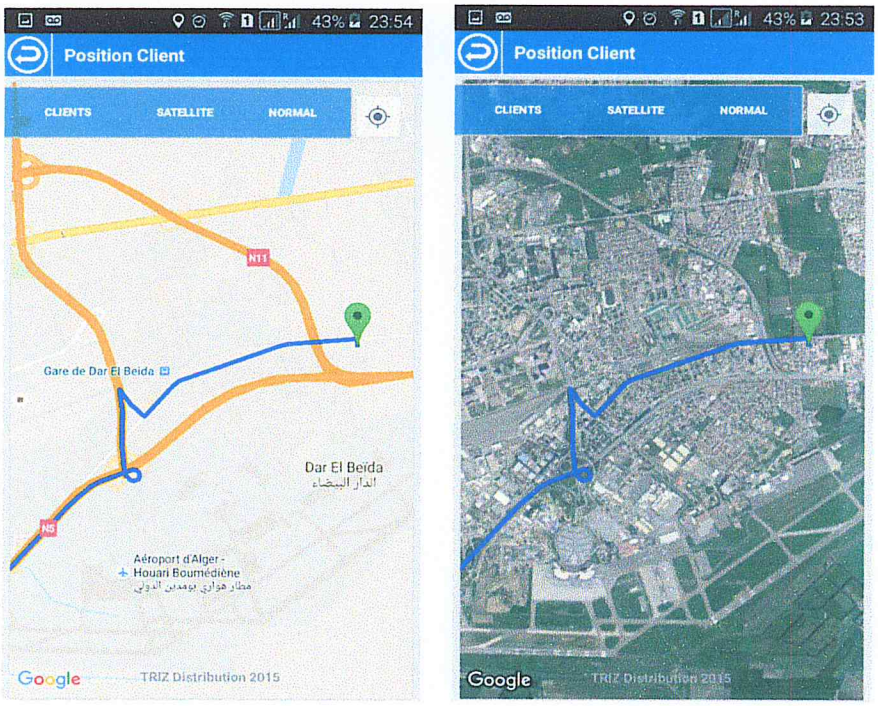


Figure 4.17. Itinéraire à prendre pour arriver au client.

5. Validation de l'application

Plusieurs expérimentations ont été élaborées pour valider l'efficacité des algorithmes (BSP et SPLIT) aussi la fonction de recherche du plus proche client pare rapport à un point et la fonction de calcul de la distance d'un trajet.

L'algorithme est implémenté en JAVA dans un Intel Core i5 2.50 GHz, 8GO de RAM.

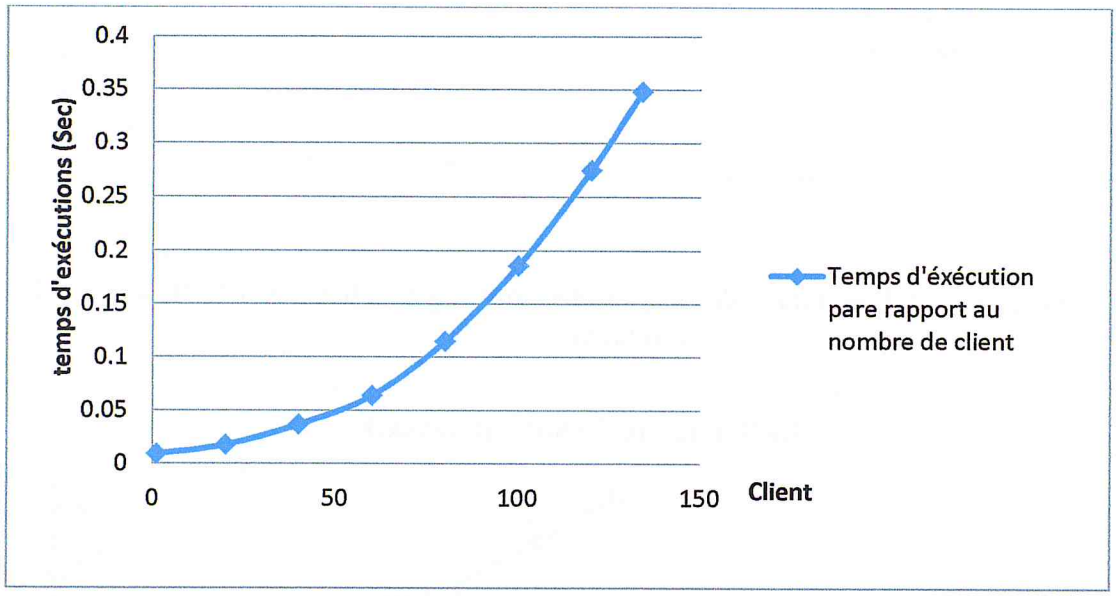


Figure 4.18. Exécution d'un algorithme de recherche du plus proche client

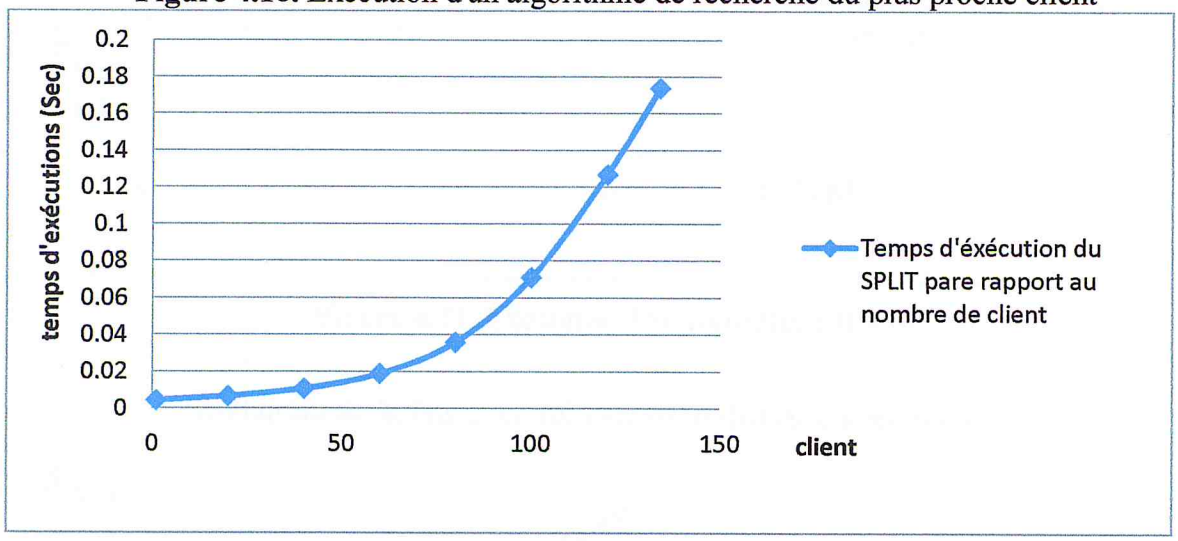


Figure 4.19. Exécution d'un algorithme SPLIT

Conclusion

Dans ce chapitre, nous avons vu les différents outils de la réalisation des deux parties du travail à faire. En commençant par l'application de la distribution WEB où, on a utilisé le SGBD « SQL Serveur » pour la base de donnée, l'application a été faite par « Java entreprise édition » car ce dernier est spécialement dédié pour les entreprises. Dans la 2ème partie du travail, on a vu aussi la réalisation de l'application Mobile sous la Plateforme ANDROÏD où, elle est conçu d'une manier simple a manipulation et elle possède une API « Google Maps » afin de faciliter la tournée.

&&&&&

CONCLUSION GENERALE

Conclusion générale

En fin de notre cursus en master informatique, nous avons été chargés de réaliser un projet de fin d'année au sein de l'entreprise HYDROTRIZ. Notre travail consiste à faire une nouvelle approche intelligente d'optimisation de tournées de véhicules pour la grande distribution. Ceci nous a permis d'approcher et de bien connaître l'environnement de la distribution. Ce dernier nous a amené à découvrir toutes les recherches existantes et aussi de comprendre le comportement naturel des abeilles qui a enrichi notre savoir et notre expérience.

Notre projet porte d'une part, sur une recherche approfondie du problème de la tournées des véhicules pour la grande distribution, et d'autre part, l'implémentation de tout un système de suivi de la distribution qui est constitué en deux parties (Application Web et Mobile).

Pour assurer cette recherche, nous avons fait un état de l'art sur les VRP, ensuite nous nous sommes intéressés à l'intelligence des essaims d'abeilles et adopter leur comportement naturel afin de résoudre et d'améliorer le problème de la distribution.

L'implémentation de notre projet a été effectuée sur deux environnements de développements:

- Environnement de développement « J2EE », utilisé pour la création des applications web, avec l'architecture multi couche qui est basée sur le modèle « MVC » où on peut décomposer chaque couche en sous couches pour accéder à la programmation modulaire ;
- Environnement de développement « JAVA, SDK Android » utilisé pour la création des applications Mobile.

L'application Web a un rôle de suivre la distribution, de l'affectation des trajets, de la détermination des tournées avec classement des clients dans la tournée et leur statistique. Chaque client est positionné sur un GPS qui nous a permis d'adopté l'application mobile afin d'arriver au client directement et suivre le camion jusqu'à la fin de son parcours.

Les deux applications peuvent communiquer entre elles par socket afin d'assurer une bonne synchronisation et une rapidité de transfert.

En perspective, afin de compléter le travail présenté dans ce mémoire, plusieurs travaux peuvent être envisagés notamment :

- Introduire des modules réservés à l'import/export ;
- Suivre les produits jusqu'à leur date d'expiration ou péremption ;
- Agrandir la distribution a l'échelle mondiale.

Pour conclure, nous affirmons que ce travail a permis de répondre à la problématique et les besoins posés au sein de l'entreprise.

&&&&&

Références bibliographiques

Références bibliographiques

Baldacci R, Aristide M & Roberto R. « *Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints*», European Journal of Operational Research, vol1, N°1, pp 1-6, 2012.

Beasley J E. *Route-first cluster-second methods for vehicle routing*, 1983,403p.

Berghida M, *Stratégies adaptatives et coopératives pour la résolution de problèmes de tournées de véhicules avec collecte et livraison*, 2015, 105p.

Books google (01 mai 2016), Avantages SQL, [en ligne]. URL : https://books.google.dz/

Brandão J. «*A new tabu search algorithm for the vehicle routing problem with backhauls*», European Journal of Operational Research, pp 540–555, 2006.

Cao E, Mingyong L & Hongming Y. «*Open vehicle routing problem with demand uncertainty and its robust strategies*», Expert Systems with Applications, vol1, pp 3569-3575,2014.

Chen J, Liu Q, Huang J & Hou Y . *An Adaptive Genetic Algorithm Based on Multi-population Parallel Evolutionary and Variable Population Size*,vol1, 2010, 262 p.

Djenouri Y . *Parallel BSO Algorithm for Association Rules Mining using Master/Workers Paradigm* , 2015, 193 p.

Douiri S, Elbernoussi S & Lakhbab H. *Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques*, (Module cryptographie et sécurité de l'information), 2010, 36p.

Drias H, Sadeg S & Yahi S. «Cooperative bees swarm for solving the maximum weighted satis_ability problem », In Computational Intelligence and Bioinspired Systems, pp 318-325, 2005.

Duhamel C. *Heuristiques pour la resolution du problème de tournées de Véhicules avec livraision et collecte et fenetre de temps*, Université Blaise Pascal, Clermont-Ferrand, 1995 .

Eilon S & Watson-Gandy C D T. *An N hristofide DistrilHilioi Management: Mathematical Modelling and Practical Analysis Griffin* , London, vol 1, 1971 .

Ekbal B M . *Parallélisation de la méthode du "Branch and Cut" pour résoudre le problème du voyageur de commerce*, 2004, vol 1, 141 p .

Eyskens S (10 mais 2016). *Travailler avec les sockets*, [en ligne]. Adresse URL : http://stephaneey.developpez.com/tutoriel/php/sockets/

Ferroukhi Y & Benali A. *Conception et réalisation d'un système d'information d'aide à la décision pour une direction commerciale de TTA spa*, 2015, 97 p .

Franceschelli M, Rosa D, Seatzu C & Bullo F. «*Gossip algorithms for heterogeneous multi-vehicle routing problems*». *Nonlinear Analysis*: Hybrid Systems, vol. 10 , pp 156-174, 2013.

Geem Z W, Lee K S & Park Y J. «*Application of Harmony Search to Vehicle Routing*». American Journal of Applied Sciences, vol 2, N°12, pp 1552-1557, 2005.

Golden B L & Wong R T. *Capacitated arc routing problems*, 1981, 305 p.

Grotchel M, Junger M & Reinelt G. «*A cutting plane algorithm for the linear ordering problem*», Operations Research, N°32, pp 1195–1220, 1984.

Gulic M & Jakobovic D. «*Evolution of vehicle routing problem heuristics with genetic programming*»,In Information Communication Technology Electronics Microelectronics (MIPRO), 2013 36th International Convention on, pp 988-992, 2013.

Haj_Rachid M. *Problème de tournées de véhicules en planification industrielle: classification et comparaison d'opérateur évolutionnaires*, 2010, 277 p.

Hong L. «*An improved {LNS} algorithm for real-time vehicle routing problem with time windows*». Computers & Operations Research, vol 39, N°2, pp 151-163, 2012.

Hu F & Wu B. «*Quantum evolutionary algorithm for vehicle routing problem with simultaneous delivery and pickup*». *In Decision and Control*, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, pp 5097-5101, 2009.

Interstices (20 février 2016), *interstices.info*, [en ligne].Adresse URL: www.interstices.info

Jasper-ireport (10 mais 2016), *Choix d'un outil de reporting*, [en ligne]. Adresse URL:https://adullact.net/docman/view.php/208/266/choix_outil_reporting.pdf

Karaboga D & Basturk B. «*Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*», Springer-Verlag Berlin Heidelberg , pp 789–798, 2007.

Karaboga D & Basturk B. «*On the performance of artificial bee colony (ABC) algorithm*», *Science Direct Applied Soft Computing*, vol 8, N°1, pp 687–697, 2007.

Lacomme P, Prins C & Ramdane-Chérif w. «*Competitive memetic algorithms for arc routing problems*», Annals of Operations Research, pp 85-159, 2004.

Land A H & Doig A G. «*An automatic method for solving discrete programming problems*», *Econometrica*, pp 497–520, 1960.

Laporte G. «*The vehicle routing problem: An overview of exact and approximate algorithms*», European Journal of Operational Research, vol 59, N°3, pp 345-358, 1992.

Liao X & Ting C. «*Evolutionary algorithms using adaptive mutation for the selective pickup and delivery problem*», In Evolutionary Computation (CEC), 2012 IEEE Congress on, pp 1-8, 2012.

Lin C, Choy K L, Ho G T S, Chung S H & Lam H Y. «*Survey of Green Vehicle Routing Problem : Past and future trends*», Expert Systems with Applications, vol 41, N°4, pp 1118-1138,2014.

Liu R, Xie X, Augusto V & Rodríguez-Verjan C. «*Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care*», *European Journal of Operational Research*, vol 230, N°3, pp 475-486, 2013.

Marinakis Y. «*Multiple Phase Neighborhood Search-GRASP for the Capacitated Vehicle Routing Problem*». Expert Systems with Applications, vol 39, N°8, pp 6807-6815, 2012.

Maxime B. *Abdoulaye S, L'optimisation par essaim particulaire pour des problèmes d'ordonnancement*, 2011, 28 p.

Maven (01 mais 2016) ,*What is Maven*, [en ligne]. Adresse URL:http://maven.apache.org

Mouassa S. «*Optimisation de l'écoulement de puissance par une méthode métaheuristique (technique des abeilles) en présence d'une source renouvelable (éolienne) et des dispositifs facts*», vol 1, pp 25-30, 2012.

Nagy G & Salhi S. «*Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries*", European Journal of Operational Research, vol 162, N°1, pp 126-141, 2005.

Naji-Azimi Z & Salari M. «*A complementary tool to enhance the effectiveness of existing methods for heterogeneous fixed fleet vehicle routing problem*», Applied Mathematical Modelling, vol 37,N°6, pp 4316-4324, 2013.

Padberg M & Rinaldi G. «*Optimization of a 532 city symmetric traveling salesman problem by branch-and-cut*», *Operations Research Letters*, pp 1–7, 1987.

Padberg W & Rinaldi G. «*A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*», SIAM Review, pp 60–100, 1991.

Panneerselvam R & Kumar S N. «*A Survey on the Vehicle Routing Problem and Its Variants*», Intelligent Information Management, vol4, N°3, pp 66-74, 2012.

Parragh P, Doerner K & Hartl R. «*A survey on pickup and delivery problems*», *Journal für Betriebswirtschaft*, vol 58 ,N°2, pp 81-117, 2008.

Phan R, Lacomme P & Duhamel C, *Une stratégie hybride pour le problème de tournées de véhicules avec livraisons et collectes, Rapport d'ingénieur*, 2009, 121 p .

Pillac V, Gendreau M, Guéret C & Medaglia A L. «*A review of dynamic vehicle routing problems*», European Journal of Operational Research, vol 225, N°1, pp 1-11, 2013.

Rego C & Roucairol C. *Le Problème de Tournées de véhicule Etude et Résolution Approchée, Technical Report, inria*, 1994, 40 p.

S. Ropke & D. Pisinger. «*A uniffed heuristic for a large class of vehicle routing problems with backhauls*», European Journal of Operational Research, vol 2004, pp 750-775, 2006.

Saremi A R, Tavakkoli-Moghaddam R & Ziaee M. «*A memetic algorithm for a vehicle routing problem with backhauls*», Applied Mathematics and Computation, vol 181,N° 2, pp 1049-1060, 2006.

Sentinelle, (21 février 2016), *Sentinelle et Les essaims d'abeilles*, [en ligne].Adresse URL: http://www.abeillesentinelle.net/

Smaili Y. *Optimisation des flux dans un système de distribution application à une entreprise*, 2012, 100 p.

Shodhganga (25 Janvier 2015), *shodhganga.inflibnet*, [en ligne]. Adresse URL: http://shodhganga.inflibnet.ac.in/bitstream/10603/15491/6/06_hapter%201.pdf.

Spring (10 mais 2016), *What is springsecurity*, [en ligne]. Adresse URL: http://docs.spring.io/springsecurity/site/docs/3.0.x/reference/springsecuritysingle.html

Tang J, Ma Y, Guan J & Yan G. «*A Max-Min Ant System for the split delivery weighted vehicle routing problem*», Expert Systems with Applications, vol 40, N°18, pp 7468-7477, 2013.

TotalProg(01 mais 2016), *avantages et inconveignents* , TotalProg ,[en ligne].Adresse URL: http://totalprog.blogspot.com/2009/11/maven-avantages-et-inconveignents.html

Toth P & Vigo D. «*A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls*», European Journal of Operational Research, vol 113, N°3, pp 528-543, 1999.

Toth P & Vigo D. «editeurs. *The vehicle routing problem.Society for Industrial and Applied Mathematics*», Philadelphia,PA, USA, 2001.

Touaibia Z, *Conception et Realisation d'une application android de facturation pour la distribution directe des produits laitiers*, 2014, 50 p.

Wang F, Tao Y & Shi N. *A Survey on Vehicle Routing Problem with Loading Constraints, In Lean Yu, Kin Keung Lai et S K Mishra, editeurs, CSO (2)*, IEEE Computer Society, 2009. 606 p.

Yazg-Tütüncü G, Carreto C A & Baker B M. «*A visual interactive approach to classical and mixed vehicle routing problems with backhauls*», Omega, vol 37, N°1, pp 138-154, 2009.

Zhengchu W, Muxun Z, Xiufeng L, Chun F & Feixiang J. «*A quantum particle swarm optimization for solving the capacitated vehicle routing problem*», *In Intelligent Control and Automation (WCICA)*, 2010 8th World Congress on, pp 3281-3285,2010.

&&&&&