Louvain School of Management (LSM)

Distribution en Belgique des médicaments issus du plasma sanguin : une étude de cas

Mémoire réalisé par
Michaël Dehan

Promoteur(s)
Daniele Catanzaro

Année académique 2017-2018
Master en Sciences de gestion

En vérité, le chemin importe peu,

La volonté d’arriver suffit à tout.

(Camus)

Remerciements

En premier lieu, je remercie mon promoteur, Daniele Catanzaro, pour ses conseils avisés et le temps qu'il a consacré à me guider sur le droit chemin. Parti de zéro dans la connaissance du champ d'application de ce mémoire, il a su m'insuffler l'envie de découvrir de nouvelles matières qui m'ont passionné lors de mes recherches.

Je ne peux évidemment pas oublier de remercier mes proches pour leur soutien indéfectible tout au long de ce parcours d'apprentissage. Plus particulièrement, Sandrine D. pour la relecture, Delphine L. pour ses précieux conseils, Anja V. pour la mise en page, Nathalie T. pour l'étincelle, mon père, mes camarades de classe et, bien entendu, mes fils Florian et Nathan ainsi que leur maman qui ont été à mes côtés et m'ont apporté l'énergie de relever ce challenge. Je remercie aussi tous ceux qui de près ou de loin ont été là. Ils se reconnaîtront

Résumé

Dans ce mémoire, nous investiguons un problème de Supply Chain Network Design (SCND) et un problème de Vehicle Routing Problem (VRP) au travers de l'étude de cas d'une entreprise pharmaceutique distribuant des médicaments issus du plasma sanguin en Belgique.

Dans une première partie, nous présentons le problème de localisation d'un dépôt en SCND et ses propriétés. Ensuite, nous appliquons la méthode de localisation de dépôt à notre étude de cas pour rechercher une localisation optimale.

En seconde partie, Nous présentons le VRP, un problème bien connu de type NP-complet où l'utilisation de méthodes exactes peut être difficile pour la résolution du problème en un temps raisonnable. Dès lors, nous passons en revue les méthodes métaheuristiques capables de trouver une solution réaliste et proche de la solution optimale en un temps acceptable. De plus, nous étudions plus en détails une classe de métaheuristique dénommée Ant Colony Optimization (ACO) et comment elle s'applique pour résoudre le VRP. Par la suite, nous implémentons un algorithme ACO pour résoudre le VRP de notre étude cas.

Finalement, nous sommes en mesure de faire des recommandations pour optimiser la distribution des produits par l'entreprise.

Table des matières

Liste des abréviations

SCND :	Supply Chain Network Design
VRP :	Vehicle Routing Problem
TSP :	Traveling Salesman Problem
MILP :	Mix Integer Linear Programing
STSP :	Symmetric Traveling Salesman Problem
ATSP :	Assymmetric Traveling Salesman Problem
NP :	Non Polynomial
CVRP :	Capacited Vehicle Routing Problem
VRPTW :	Vehicle Routing Problem Time Window
SA:	Simulated Annealings
TS:	Tabu Search
GRASP :	Greedy Randomized Adaptative Search Procedure
VNS:	Variable Neighborhood Search
VND:	Variable Neighborhood Descent
VNDS :	Variable Neighborhood Decomposition Search
GLS :	Guided Local Search
ILS :	Iterated Local Search
EC :	Evolutionary Computation
ACO :	Ant Colony Optimization
AS :	Ant System
EAS :	Elitist Ant System
RAS :	Rank Based Ant System
MMAS :	Max-Min Ant System
ACS :	Ant Colony System

Liste des tables

Liste des figures

Introduction

La vie n’est pas un long fleuve tranquille et c’est également le cas pour la vie des entreprises. Elles sont confrontées très régulièrement à des choix à poser, des décisions à prendre, des stratégies à dicter ou tout simplement prendre des mesures de bonne gestion en fonction des aléas internes ou externes.

Nous nous intéressons dans le cadre de ce mémoire plus particulièrement au cas pratique d’une entreprise de produits pharmaceutiques subissant la perte de l’approvisionnement de la principale matière première nécessaire à son activité. Ce changement soudain représente une diminution de 80% de son chiffre d’affaires sur deux ans.

La société est une société coopérative à responsabilité limitée belge spécialisée dans le fractionnement et la production de dérivés plasmatiques stables : facteurs de coagulation, immunoglobulines et solutions d’albumine. La plupart de ces produits sont essentiels au traitement de patients présentant une maladie chronique. Ces patients dépendent de ces produits tout au long de leur vie afin d’être en mesure de contrôler leur maladie. De cette façon, ces produits contribuent à une meilleure qualité de la vie des patients.

Pour la préparation de ces dérivés plasmatiques pour les soins de santé belges, on utilise le plasma prélevé auprès de donneurs volontaires non rémunérés. Pour répondre à la demande en composés sanguins, les services de transfusion sanguine de la Croix-Rouge de Belgique récoltent le sang. Etant donné que tout le plasma n’est pas nécessaire au traitement des patients dans les hôpitaux, une partie du plasma est transmise à la société pour la production de produits plasmatiques. La Croix-Rouge de Belgique reçoit une indemnité de la société pour les frais liés à la collecte de ce plasma. Dès lors, ce sang est fractionné en composants sanguins produits suivant les normes les plus strictes en matière de qualité.

La société propose une vaste gamme de produits de dérivés plasmatiques stables. Depuis de nombreuses années, la société est parvenue à fournir aux hôpitaux des médicaments importants, permettant souvent de sauver des vies humaines.

En 2017, le service public fédéral Santé publique de la chaîne alimentaire et environnement a publié un appel d’offres ayant pour but de désigner un adjudicataire chargé, pour une période de 4 ans, du traitement du plasma livré par les établissements de transfusion sanguine en

Belgique et de l'assurance, aux hôpitaux, d'une offre suffisante de dérivés plasmatiques stables (solutions d'albumine et immunoglobulines pour administration intraveineuse) fabriqués à partir de ce plasma. L'appel d'offres concerne l'adjudication de 100% du marché des solutions d'albumine et de 50% du marché des immunoglobulines.

Cet appel d'offres n'a pas été remporté par la société malgré des qualités évidentes dans sa capacité à gérer cette activité. En conséquence, les livraisons de plasma ont été stoppées et la société a disposé d'un délai pour produire et vendre les derniers lots issus du plasma déjà récolté. Au-delà, une grande partie de son portefeuille de produits ne peut plus être soutenue. Cette transition est évaluée à deux ans.

Cette forte diminution de produits a évidemment des conséquences importantes sur le chiffre d'affaires mais aussi sur la chaîne de distribution. La société s'attend à une diminution de 70% de son volume de colis à traiter vers les hôpitaux, sur la première année, et de 50% en moins sur la deuxième année. Ces transports sont effectués par la société à l'aide d'un prestataire de service externe.

Il est intuitif de prédire également une diminution des coûts de transport liés à la distribution des produits en Belgique. Bien que les coûts de transport soient corrélés au volume de l'activité, d'autres facteurs comme selon Watson (2013), les coûts minimum en transport, les différences régionales ou les profils de commandes des clients ont une influence sur le coût total.

Ceci nous amène à une réflexion qui est de répondre à deux questions liées à la distribution de produits. En premier, déterminer la localisation d'un dépôt et deuxièmement, déterminer comment servir les clients à travers un réseau de distribution au départ de ce dépôt.

Ce mémoire sera organisé comme suit. Dans une première partie, nous adresserons la question de la localisation d'un dépôt. Dans le chapitre 1, nous présenterons les méthodes issues du Supply Chain Network Design (SCND) pour localiser un dépôt de manière optimale. Le chapitre 2 sera consacré à l'application de ces méthodes à notre étude de cas. En seconde partie, nous nous intéresserons à présenter, dans le chapitre 3, la nature du problème lié au Vehicle Routing Problem (VRP) et ses variantes. Le chapitre 4 se focalisera sur l'approche métaheuristique pour résoudre ce problème. Dans le chapitre 5, nous présenterons plus en détail une classe de métaheuristique regroupée sous la dénomination de Ant Colony Optimization (ACO) ; Finalement, dans le chapitre 6, nous implémenterons un algorithme de

colonie de fourmis (ACO) pour démontrer l'opportunité éventuelle d'internaliser la distribution des produits.

Première partie

Chapitre 1

1. Localisation de dépôts

1.1. Supply Chain Network Design (SCND)

A cause de la complexité d'une chaîne d'approvisionnement et de la richesse des types de données quantitatives à traiter, l'utilisation de technologies mathématiques d'optimisation est un moyen efficace pour faire le tri parmi la grande variété des options, pour déterminer les meilleures localisations de dépôts et aider à la prise de décisions judicieuses. Nous détaillerons dans ce chapitre les étapes pour formuler un modèle logique de Supply Chain Network en suivant la structure du livre écrit par Watson (Supply Chain Network Design, 2013)

1.2. Optimisation d'un réseau

L'optimisation est une technologie complémentaire et non concurrente envers des systèmes de récoltes de données ; elle permet de déterminer les localisations optimales des entités de la chaîne d'approvisionnement. Comme ces problèmes ont une haute valeur stratégique pour une organisation sur bien des aspects de l'activité, il est aussi important de prendre en considération des aspects non quantifiables avant toute prise de décision définitive. Ceci étant dit, pour résoudre les aspects quantitatifs d'un problème avec l'optimisation mathématique et ainsi formuler un modèle logique de SCND, il est primordial de réfléchir à ces quatre éléments :

- L'objectif
- Les contraintes
- Les décisions
- Les données

L'objectif est le but de l'optimisation mais aussi le critère que nous pouvons utiliser pour comparer différentes solutions. Un des objectifs les plus courants en network design est de minimiser les coûts. Nous pouvons donc comparer deux solutions entre elles en se basant sur

la comparaison des coûts et de la sorte, nous pouvons juger laquelle satisfait le mieux. Il est dès lors important que l'objectif soit quantifiable.

1.3.La localisation de dépôts sur base de la distance

Localiser un dépôt par rapport à la demande des points de livraison est fait au mieux pour minimiser la distance moyenne pondérée par la demande. Nous pouvons réaliser cela en sélectionnant une solution au départ d'une liste de candidats. En sélectionnant un dépôt à la fois, il est possible de lister toutes les combinaisons et de choisir la meilleure d'entre elles.

Définition formelle du problème pour localiser *N* dépôts

Pour un ensemble de clients donnés et leur demande respective, trouver le meilleur *N* nombre de dépôts qui minimise le total de la distance pondérée du dépôt au client en assumant que chaque dépôt peut satisfaire la demande totale des clients et que chaque demande est toujours satisfaite. (Watson, 2013)

Trouver une meilleure localisation dans un ensemble équivalant à *N* possibilités est relativement simple. Le choix de la meilleure solution sera le résultat de la comparaison de toutes les combinaisons possibles dans un ensemble de *N* choix. Le problème peut devenir très vite beaucoup plus compliqué. Si nous sélectionnons les deux meilleures localisations, nous devons définir quels clients sont servis par quels entrepôts. Il est encore simple d'assigner chaque client à un dépôt, il suffit de simplement choisir celui des deux qui est le plus proche de chaque client. Néanmoins, la difficulté tient au fait du nombre de combinaisons potentielles nécessaires pour prouver cela. Si nous avons, par exemple, 18 dépôts différents et que nous souhaitons en sélectionner les 5 meilleurs, nous aurons 8568 combinaisons à évaluer. Le nombre de combinaisons différentes peut très vite augmenter si nous voulons considérer encore plus de choix possibles. Sur la base de cette information, il est évident que l'utilisation de techniques de programmation linéaire en nombres entiers est nécessaire pour rechercher à travers toutes ces combinaisons de manière systématique afin de trouver une solution optimale.

1.4.Formulation

Une formulation nous est donnée par Watson (2013). Nous avons une liste de clients à servir. Nous indexerons cette liste par la lettre majuscule *J*. Chaque client individuellement à l'intérieur de cette liste sera indexé avec la lettre minuscule *j*. La demande est connue pour

chaque client. Nous l'appellerons *d* et indiquerons la demande d'un client *j* par la dénomination d_j. Nous avons une liste de dépôts potentiels. Nous indexerons cette liste par la lettre majuscule *I*. Chaque dépôt de cette liste sera désigné par la lettre minuscule *i*. Nous pouvons représenter la matrice des distances d'un dépôt *i* à un client *j* par $dist_{ij.}$

Formulation 1 – (Watson, 2013)

$$\textbf{Minimize} \quad \sum_{i \in I} \sum_{j \in J} dist_{i,j} d_j Y_{i,j} \tag{1}$$

Subject to:

$$\sum_{i \in I} Y_{i,j} = 1; \forall j \in J \tag{2}$$

$$\sum_{i \in I} X_i = P \tag{3}$$

$$Y_{i,j} \leq X_i; \forall i \in I, \forall j \in J \tag{4}$$

$$Y_{i,j} \in \{0,1\}; \forall i \in I, \forall j \in J \tag{5}$$

$$X_i \in \{0,1\}; \forall i \in I \tag{6}$$

La fonction Objectif (1) détermine le fait qu'une solution est meilleure qu'une autre et, dans ce cas, vise à minimiser le total de la distance pondérée des dépôts aux points de livraisons. La contrainte (2) stipule que chaque client doit être complètement servi. La contrainte (3) stipule le nombre *P* de dépôts à localiser. La contrainte (4) stipule si un dépôt doit être considéré comme faisant partie de la solution. Les contraintes restantes définissent le domaine des variables.

Ce modèle proposé par Watson (2013), malgré une fonction objective assez restreinte qui minimalise uniquement la distance moyenne pondérée, permet de résoudre un grand nombre de problèmes de SCND. Etre aussi proche que possible de la demande est une considération importante à prendre en compte. Etant donné que la distance et les coûts de transport sont fortement corrélés, ce modèle est une bonne approximation pour minimiser le coût de transport vers les clients. Bien que nous ayons tendance, en optimisant, de vouloir apporter une seule réponse correcte, nous devons tenir compte du fait que la réalité est plus compliquée que cela. Il est donc important de se baser sur plusieurs scénarios.

1.5.Les coûts de transport

Les coûts de transport sont très souvent les coûts les plus importants dans une étude de SCND. En pratique, une large gamme d'études en network design ne considère que les coûts de transport quand on considère la recherche d'une localisation de dépôt optimale. Parfois, inclure les coûts de transport à un modèle ne change pas la solution trouvée ; ce qui peut être attribué à la forte corrélation entre la distance et le coût de transport. Dans la majorité des cas, la solution varie quand on applique des coûts de transport. Il y a plusieurs raisons comme par exemple un coût de transport minimum, des différences régionales dans le coût de transport ou des profils de clients différents. Dans ce dernier cas, Les clients qui, par exemple, commandent en plus grande quantité bénéficient d'un coût de transport inférieur. Ce qui incite l'optimisation à choisir des dépôts qui sont plus proches des clients ayant le prix unitaire le plus élevé.

Formulation 2 – (Watson, 2013)

Minimize $\sum_{i\in I}\sum_{j\in J} trans_{i,j} d_j Y_{i,j}$

Subject To:

$$\sum_{i\in I} Y_{i,j} = 1; \forall j \in J$$

$$\sum_{i\in I} X_i = P$$

$$Y_{i,j} \leq X_i; \forall i \in I, \forall j \in J$$

$$Y_{i,j} \in \{0,1\}; \forall i \in I, \forall j \in J$$

$$X_i \in \{0,1\}; \forall i \in I$$

À l'aide de cette formulation, nous minimisons le coût total de transport à dépenser en sélectionnant le meilleur nombre sélectionné de dépôts. La plus grande différence avec la formulation exprimée précédemment est que nous avons une matrice de coût au lieu d'une matrice de distance. La matrice $trans_{ij}$ représente le coût pour transporter une unité du dépôt i jusqu'au client j.

La demande d_j est exprimée en total, habituellement annuel et non transport par transport. Bien qu'ajouter chaque envoi individuellement donne une impression de précision accrue, ce n'est pas forcément le cas. Faire tourner un modèle avec une agrégation des différents envois en un seul point de demande est aussi précis qu'un modèle avec des envois individuels. (Watson, 2013)

Le coût de transport est converti en un coût à l'unité par la formule suivante :

$$trans_{i,j} = \frac{load_{i,j}}{avg_{i,j}}$$

Dans ce cas, *trans ij* est le coût à l'unité, *load ij* est le coût total pour transporter du point *i* au point *j* et *avg ij* est la taille moyenne du transport. Pour certains envois, déterminer avg ij est plutôt trivial. Pour les envois en colis par exemple, le coût par envoi est dépendant de la taille de l'envoi. En déterminant le nombre moyen de colis livrés par clients pour chaque envoi, nous obtenons une mesure précise de la taille d'un envoi.

1.6.Les coûts fixes et variables des dépôts

Les coûts fixes sont des coûts indépendants du volume. Il peut s'agir de coûts de construction d'un dépôt, de coûts d'agrandissement d'un dépôt, d'ajouts d'équipements ou d'engagement de personnel. Les coûts d'investissement sont dissociés des coûts des opérations car ils sont dépréciés dans le temps tandis que les coûts des opérations reflètent les coûts nécessaires au fonctionnement, indépendamment du volume à traiter.

Les coûts variables sont ceux qui dépendent du nombre d'unités qui sont traitées. Pour le déterminer, il faut multiplier le nombre d'unités traitées par le coût variable.

En pratique, il n'y a pas de réponse correcte pour déterminer quels coûts sont fixes et quels autres sont variables. Cependant, il est important de comprendre comment ils fonctionnent pour pouvoir les intégrer dans le modèle. De plus, nous souhaitons les utiliser d'une manière qui donne le plus de sens aux types de décisions à prendre avec le modèle.

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} (trans_{i,j} + facVar_i) d_j Y_{i,j} + \sum_{i \in I} \sum_{w \in W} facFix_{i,w} X_{i,w}$$

La lettre majuscule *W* représente la liste des options de dépôts et *w* le type d'options (par exemple, petit, moyen ou grand). Cet ensemble permet de modéliser les changements sur les frais fixes et variables pour chaque option. Le facteur *W* est ici appliqué uniquement sur les coûts fixes mais peut être étendu aux coûts variables également.

En ajoutant des coûts fixes et variables aux dépôts, cela donne à l'optimisation le choix de sélectionner le dépôt avec le coût le plus faible. Lors d'une optimisation basée sur le coût total, l'optimisation va sélectionner les dépôts avec les coûts fixes et variables les plus faibles pour autant que le gain apporté n'excède pas l'augmentation en coût de transport.

1.7.Les produits multiples

Il est typique pour les entreprises de disposer et de distribuer un grand nombre de produits différents. Il n'est pas rare que ces produits répondent à des catégories différentes qui correspondent à des caractéristiques différentes.

Bien que ces produits puissent être distribués dans le même style d'emballages avec plus ou moins la même densité. Les conditions de stockage et de transport peuvent être très différentes pour chaque catégorie. Les produits à température ambiante peuvent être transportés et stockés dans des conditions à température ambiante au contraire de produits nécessitant une réfrigération. Ces derniers requièrent un transport spécifique ainsi que des conditions de stockage adaptées. Typiquement, les coûts de transport sont 10% à 30% plus élevés que les coûts à température ambiante. En tenant compte de cela, il est nécessaire de refléter ces différences dans la modélisation pour tenir compte de la différence des coûts en fonction du type du produit.

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(transWC_{i,j,k} + whVar_{i,k})d_{j,k}Y_{i,j,k}$$

La formulation ci-dessus de la fonction objective permet de capturer les différences de produits ou de catégories de produits. La lettre majuscule *K* représente la famille de produits où la lettre minuscule *k* représente un produit spécifique de cette famille. Nous utilisons la représentation $d_{j,k}$ pour définir la demande d'un produit *k* pour le client *j*. Le coût pour opérer un produit *k* au dépôt *i* est représenté par le terme $whVar_{i,k}$. Les trois sommations $\sum$ indiquent que nous sommons les coûts de transport, les coûts de stockage pour chaque client, dépôt et produit.

1.8. Conclusion du chapitre

Nous avons introduit la modélisation en SCND et la notation qui sera utilisée dans ce mémoire. Nous avons présenté certaines propriétés appliquées à la modélisation en SCND :

1. Cela consiste à minimiser la distance moyenne pondérée entre des dépôts et des points de livraison.
2. Certains paramètres comme le coût de transport, les coûts fixes et variables des dépôts et les produits multiples sont incorporés pour valoriser le modèle.

Chapitre 2

2. Application SCND à l'étude de cas.

2.1.Problématique

Nous nous intéressons dans le cadre de cette étude de cas plus particulièrement au cas pratique d'une entreprise de produits pharmaceutiques subissant la perte de l'approvisionnement de la principale matière première nécessaire à son activité. Ce changement soudain représente une diminution de 80% de son chiffre d'affaires sur deux ans. En effet, elle ne pourra plus produire à terme la totalité des produits qu'elle distribue.

La société s'attend à une diminution de 70% de son volume de colis à traiter vers les hôpitaux sur la première année et d'encore 50% en moins sur la deuxième année. Ces transports sont effectués par la société à l'aide d'un prestataire de service externe.

Il est intuitif de prédire également une diminution des coûts de transport liés à la distribution des produits en Belgique. Bien que les coûts de transport soient corrélés au volume de l'activité, d'autres facteurs comme selon Watson (2013), les coûts minimum en transport, les différences régionales ou les profils de commandes des clients ont une influence sur le coût total.

Le problème nécessite une simulation des deux années suivantes affectées par la diminution du nombre de produits à livrer. Nous étudierons trois scénarios : l'année de référence (A) et les deux suivantes (A+1, A+2) extrapolées par la suppression des produits concernés.

Nous nous interrogeons, dans le cadre de cette étude, à la pertinence de l'emplacement du dépôt utilisé pour la distribution des produits. Nous souhaitons aussi évaluer l'impact de l'ouverture de un ou plusieurs dépôts supplémentaires. En général, situer les dépôts au plus proche des clients est souvent plus avantageux.

La question de recherche suivante se profile : **« Est-il opportun d'envisager une relocalisation du dépôt d'une entreprise soumis à une forte variation de son volume à transporter ? »**

Cela nous paraît cohérent dans le cadre d'une forte diminution de la demande de transport. La répartition géographique de la demande des clients peut varier également et par conséquent

les distances à parcourir pour satisfaire cette demande. Ce qui nous permet de formuler l'hypothèse suivante :

> **Hypothèse 1** : « Il existe une meilleure localisation d'un ou plusieurs dépôts en Belgique qui minimise la distance moyenne pondérée entre ce dépôt et les points de livraison. »

Si le coût du transport est tarifé au colis transporté plutôt qu'à la distance parcourue, c'est le profil de commande du client qui influence le coût de transport au lieu du nombre de kilomètres parcourus. Cela nous permet de formuler une deuxième hypothèse :

> **Hypothèse 2** : « Il existe une meilleur localisation d'un ou plusieurs dépôts en Belgique qui minimise le coût de transport moyen pondéré entre ce dépôt et les points de livraison. »

2.2. Collecte des données

Nous allons utiliser la base de données des livraisons d'une année d'une entreprise pharmaceutique pour la Belgique. Bien que les livraisons journalières puissent présenter de grands écarts, la demande annuelle reste relativement stable et équivalente d'année en année. L'entreprise livre les hôpitaux en Belgique qui sont les clients au départ d'un dépôt.

Les différents médicaments sont agrégés en deux catégories. Les produits qui nécessitent un stockage et un transport à une température comprise entre +15° et +25°C, nous les appellerons les produits « Room » et les produits qui nécessitent un stockage et un transport réfrigéré compris entre +2° et +8°C que nous appellerons « Cool ». Ces produits sont transportés dans des cartons de taille standard. Il est nécessaire de séparer les produits en deux catégories car notre objectif est de comparer les coûts de transport pour différentes solutions. Le coût de transport réfrigéré est plus cher que celui à température ambiante. Cela se comprend aisément par le matériel supplémentaire à mettre en œuvre pour garantir une température basse constante.

Les données sont issues de la facturation pour l'année 2017 du prestataire de transport. Elles reprennent des informations comme : le nom du client, l'adresse, la date d'envoi, le nombre de colis, le type de produits expédiés.

Cette base de données du transporteur a été vérifiée et réconciliée avec les données de facturation et de notes d'envois du système informatique de l'entreprise.

2.3. Traitement des données

Bien que la base de données récoltées soit complète et de bonne qualité, certaines données nécessitent d'être formatées pour permettre d'être utilisées pour la modélisation tandis que d'autres données sont manquantes et nécessitent d'être récoltées différemment voire même traitées antérieurement en vue de leur utilisation.

Nous avons également réalisé une extrapolation des données à partir de l'année de référence pour les deux prochaines années en retirant respectivement 100% des colis d'albumines et 50% des immunoglobulines aux commandes de l'année de référence pour la première année et les 50% restants des immunoglobulines pour la deuxième année.

2.4. Standardisation des données

Les adresses : Etant donné qu'il est nécessaire de géolocaliser les adresses des clients, une standardisation est nécessaire pour garantir une bonne reconnaissance de l'application Google Maps qui sera utilisée.

Les unités de mesure : Certains produits sont dans certains cas vendus par unité de principe actif plutôt que par flacons. Un ajustement a été effectué pour ne conserver que le nombre de flacons effectivement expédiés.

2.5. Création de base de données

2.5.1. Géolocalisation et distances

Pour évaluer la distance séparant les dépôts des clients mais aussi les distances séparant les clients entre eux. Pour ce faire, chaque adresse a été géolocalisée en reprenant les coordonnées de latitude et longitude récoltées via Google Maps. Ensuite, une matrice a été créée pour calculer la distance à vol d'oiseau entre chaque couple de coordonnées exprimées en degrés décimaux. Les dépôts sont indiqués en abscisse et les clients en ordonnée. La distance obtenue en kilomètres est obtenue grâce à la formule suivante :

Soit M la longueur de l'orthodromie exprimée en milles marins entre A (φ_A, λ_A) et B (φ_B, λ_B), où φ désigne la latitude et λ la longitude.

$$M = 60 \arccos\left[\sin(\varphi_A)\sin(\varphi_B) + \cos(\varphi_A)\cos(\varphi_B)\cos(\lambda_B - \lambda_A)\right]$$ [1]

[1] https://excelinsmartdata.fr/2016/08/17/distance-vol-oiseau-coordonnees-gps/

Cette formule a été convertie en une formule Excel (Version 2010) pour faciliter le calcul de chaque distance et arrondie sans décimales.

2.5.2. Le nombre de colis

Pour pouvoir comparer l'impact de la diminution de transport sur chaque année, nous avons simulé, en prenant l'année de référence comme base, sur les deux années suivantes la perte progressive du volume de transport suite à l'adjudication des produits concernés à un autre acteur du marché. Soit 100% des albumines et 50% des immunoglobulines sur la première année et la totalité des immunoglobulines sur la deuxième année.

Nbre de colis année A		Nbre de colis année A+1		Nbre de colis année A+2	
+ 15° --> + 25°	+ 2° --> + 8°	+ 15° --> + 25°	+ 2° --> + 8°	+ 15° --> + 25°	+ 2° --> + 8°
17.097	2.855	3.779	2.746	0	2.746

Table 1 : total des colis à livrer par année

2.5.3. Le coût de transport au kilomètre

Dans les données récoltées, le coût de transport est un coût forfaitaire par colis qui décroît en fonction du nombre de colis expédiés en même temps. Par exemple, il est plus avantageux de faire une livraison de trois colis plutôt que trois livraisons d'un colis.

Dès lors, le coût du transport est corrélé au profil de commande du client et non à la distance. Etant donné que nous voulons comparer des solutions en fonction de la distance parcourue, nous devons convertir le coût forfaitaire par colis en un coût équivalent au kilomètre.

Comme le coût de chaque expédition est pondéré par le nombre de colis expédiés lors de cet envoi, nous avons donc utilisé un calcul de moyenne pondérée par client pour obtenir le coût au kilomètre propre à chaque client en fonction de son profil de commande.

2.5.4. Le nombre d'envois

Les produits sont emballés pour expédition dans des cartons de taille standard. Chaque carton peut contenir un nombre standard de chaque forme de produit. En général, il n'est pas d'usage dans l'industrie pharmaceutique de regrouper des produits différents dans un même carton de groupage. Cela se comprend aisément pour des questions de sécurité dans le but de minimiser le risque d'erreur dans l'identification du produit livré.

Néanmoins, pour de très petites quantités de commande et pour éviter une multiplication du nombre de colis et donc du prix afférant, certains groupages sont acceptés. Comme nous

devons pouvoir supprimer le nombre de colis de certains produits, nous avons légèrement augmenté le nombre de colis en assignant un seul produit par colis. Cet ajustement représente moins de 1% du total des envois.

Le nombre d'envois pour chaque client est déterminé en divisant le nombre total de colis expédiés au client sur un an par le nombre moyen de colis commandés par envoi.

2.5.5. Les dépôts candidats

En plus du dépôt actuellement utilisé par l'entreprise que nous appellerons DEP1, nous avons localisé 17 autres dépôts disséminés sur le territoire de la Belgique. Les dix premiers ont été sélectionnés dans une liste de prestataires de services pouvant éventuellement fournir le même service que le prestataire actuel. Les 7 autres dépôts sont des localisations fictives mais ont été positionnées dans les régions qui ne disposaient pas de dépôts. Ceci afin de couvrir l'ensemble des possibilités et d'étendre l'espace de recherche.

2.5.6. Le nombre de clients

La suppression de certains produits a pour conséquence de supprimer certains clients. En effet, certains clients ne commandent que des produits qui font l'objet de la suppression. Le nombre total de clients passe donc respectivement de 170 pour l'année A, à 149 pour l'année A+1 et 127 pour l'année A+2.

2.6.Modèles SCND

2.6.1. Description du modèle basé sur la distance.

Le modèle proposé assume un ensemble de clients dont les adresses de livraison sont connues ainsi que la demande de chacun de ces clients et un ensemble de localisations de dépôts candidates en sus de la localisation du dépôt actuel. Il optimise la distance moyenne pondérée entre les dépôts et les clients en allouant les transports entre eux pour minimiser la distance totale parcourue.

2.6.2. Suppositions et limitations du modèle

1. Les localisations des dépôts et des clients sont connues
2. La demande des clients est connue pour la période.
3. La demande est multi-produits, réfrigérés et température ambiante
4. Tous les dépôts ont la capacité de servir l'ensemble de la demande

5. Les distances entre les dépôts et les clients sont à vol d'oiseau et arrondies à un nombre entier.
6. Les produits réfrigérés et à température ambiante sont transportés séparément
7. Les transports spéciaux ou express ne sont pas pris en compte.
8. Les transports exceptionnels vers des destinations hors de Belgique ne sont pas comptabilisés.

2.6.3. Formulation du modèle

Le modèle basé sur la distance qui est utilisé est la formulation 1 de Watson (2013) énoncée dans le chapitre précédent. Nous y ajoutons l'index k pour prendre en compte la gestion du type de produits : Ambient ou Cool. La formulation de la fonction objective devient donc :

$$Min \sum_{d \in D} \sum_{c \in C} \sum_{k \in K} distance_{dc} \; demand_{ck} \; y_{dck}$$

Le modèle de coût remplace dans la fonction objective le paramètre distance par le paramètre coût qui est la multiplication de la distance par le coût au kilomètre calculé pour chaque client.

Le modèle implique les différents ensembles, paramètres et variables suivants :

Ensembles :

Clients : le nombre potentiel de clients indexé par *c*.
Dépôts : le nombre potentiel de dépôts indexé par *d*.
Produits : le nombre potentiel de types de produits indexé par *k*

Paramètres :

Distance : la distance entre le dépôt d et le client c
Demand : la demande du client c pour le produit ambient
Demand2 : la demande du client c pour le produit cool
P : le nombre de dépôts à ouvrir

Variables de décision :

X_d : variable binaire égale à 1 si l'entrepôt *d* est ouvert et 0 sinon
Y_{dc} : variable binaire égale à 1 si l'entrepôt *d* livre le client *c* et 0 sinon

Le problème est formulé comme un Mix Integer Linear Programming (MILP). Le modèle est résolu en utilisant Xpress Fico software qui utilise le langage de programmation Mosel. La programmation a été réalisée en s'inspirant d'un modèle proposé par Catanzaro (2018)

2.7.Résultats

Modèle distance

A					
Distance Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total KM	366.206	Total KM	266.443	Total KM	215.440
Open Depot	DEP (7)	Open Depots	DEP (1) (14)	Open Depots	DEP (7) (10) (14)
		From CASE 1	-99.763	From CASE 2	-51.003
			-27%		-19%
				From CASE 1	-150.766
					-41%

Table 2 : Résultats modèle distance pour l'année A

La localisation optimum pour un dépôt localise ce dépôt à DEP7.Celà signifie un changement par rapport à la localisation actuelle situé à DEP1.

Pour les autres scénarios, respectivement avec deux dépôts ouverts ; l'optimum associe DEP1et DEP14 avec un gain de 27% en moins de kilomètres parcourus et avec trois dépôts, l'optimum est DEP7, DEP10 et DEP14 avec un gain additionnel de 19%.

A + 1					
Distance Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total KM	183.137	Total KM	129.365	Total KM	103.217
Open Depot	DEP (7)	Open Depots	DEP (1) (14)	Open Depots	DEP (7) (10) (14)
		From CASE 1	-53.772	From CASE 2	-26.148
			-29%		-20%
				From CASE 1	-79.920
					-44%

Table 3 : Résultats modèle distance pour l'année A + 1

Pour l'année A + 1, on conserve les mêmes résultats pour la localisation des dépôts. On observe une diminution du total de kilomètres parcourus justifiée par la diminution du nombre de colis à transporter et donc du nombre d'envois. Le nombre de clients à servir a également diminué. Il y a moins de clients à servir d'où moins de kilomètres à parcourir pour satisfaire la demande.

A + 2					
Distance Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total KM	100.741	Total KM	72.834	Total KM	57.401
Open Depot	DEP (7)	Open Depots	DEP (1) (14)	Open Depots	DEP (7) (10) (14)
		From CASE 1	-27.907	From CASE 2	-15.433
			-28%		-21%
				From CASE 1	-43.340
					-43%

Table 4 : Résultats modèle distance pour l'année A + 2

Ici encore, on observe les mêmes résultats pour la localisation des dépôts. La diminution des kilomètres à parcourir est encore accentuée par la diminution des colis et du nombre de clients sur l'année A + 2.

Modèle Coût transport

A					
Cost Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total €	99.001	Total €	77.227	Total €	66.078
Open Depot	DEP (1)	Open Depots	DEP (1) (7)	Open Depots	DEP (1) (7) (14)
		From CASE 1	-21.774	From CASE 2	-11.149
			-22%		-14%
				From CASE 1	-32.923
					-33%

Table 5 : Résultats Modèle coût pour l'année A

La localisation optimum place le dépôt en DEP1. Ce qui correspond à la configuration de la situation actuelle de l'entreprise étudiée.

Les deux autres scénarios voient également leur solution optimum se modifier par rapport au modèle basé sur la distance. Respectivement en DEP1 et DEP7 pour deux dépôts ouverts et DEP1, DEP7, et DEP14 pour trois dépôts ouverts.

A + 1					
Cost Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total €	57.430	Total €	44.925	Total €	37.335
Open Depot	DEP (1)	Open Depots	DEP (1) (7)	Open Depots	DEP (1) (7) (14)
		From CASE 1	-12.505	From CASE 2	-7.590
			-22%		-17%
				From CASE 1	-20.095
					-35%

Table 6 : Résultats modèle coût pour l'année A + 1

La localisation des dépôts optimums reste constante pour l'année A + 1 par rapport à l'année A. On observe également une diminution logique du coût total vu la diminution de transport et de clients à fournir.

A + 2					
Cost Model					
1 Open Depot		2 Open Depots		3 Open Depots	
CASE 1 : Optimal Solution		CASE 2 : Optimal Solution		CASE 3 : Optimal Solution	
Total €	42.314	Total €	32.726	Total €	27.511
Open Depot	DEP (1)	Open Depots	DEP (1) (7)	Open Depots	DEP (1) (7) (14)
		From CASE 1	-9.588	From CASE 2	-5.215
			-23%		-16%
				From CASE 1	-14.803
					-35%

Table 7 : Résultats modèle coûts pour l'année A + 2

La diminution du coût total est observée et pas de changement de localisation optimum pour l'année A + 2 par rapport à l'année précédente.

2.8.Limitations

Bien que les modélisations et leurs résultats nous donnent une certaine indication pour trouver une bonne solution, le monde réel présente souvent des réalités plus complexes. La pertinence de la technique développée par nos modèles dépend de l'évaluation faite de ces réalités.

Premièrement, l'évaluation de la distance à parcourir entre les dépôts et les clients. Nos modèles évaluent la distance à parcourir par un calcul de distance entre deux points à vol d'oiseau. L'avantage est que le problème devient symétrique ou $d_{i,j} = d_{j,i.}$ Cela ne tient dès lors pas compte du réseau routier et des éventuels obstacles naturels qui peuvent imposer un détour plus long. L'optimisation proposée a tendance à localiser les dépôts au plus près de la demande la plus forte. Un hôpital avec une forte demande mais situé dans une zone nécessitant un parcours moins direct peut avoir une influence sur nos résultats actuels.

Deuxièmement, le coût du transport appliqué aux autres dépôts n'est pas forcément identique au tarif pratiqué au départ du dépôt actuel. Si un dépôt se trouve fortement éloigné de la grande majorité des clients à forte demande et si le coût de transport est corrélé à la distance à parcourir même si ce coût est incorporé dans un tarif forfaitaire, il est intuitif de penser que le tarif proposé par un prestataire de services éloigné des clients proposera un tarif de prestations de ses services plus élevé.

Troisièmement, les frais fixes et variables d'ouverture d'un nouveau dépôt sont à prendre en considération avant toute décision d'ouverture. En fonction de la stratégie à court, moyen ou long terme de l'entreprise, les frais fixes comme les coûts de déménagement, d'infrastructures informatiques, de formation du personnel, etc. peuvent être amortis sur plusieurs années, tandis que les coûts doivent être comptabilisés année par année. Quelle que soit la méthode utilisée et sans tenir compte de facteurs autres que financiers, la décision d'ouverture ne doit être prise que si le gain apporté en coût de transport est au moins égal ou supérieur aux coûts fixes et variables que l'ouverture d'un nouveau dépôt implique.

Quatrièmement, les modèles étudiés ne représentent pas une réalité de transport efficient. Un retour au dépôt après chaque livraison ne rentabilise pas le temps consacré aux livraisons. En général, les livraisons s'organisent en tournées où plusieurs clients sont livrés successivement en fonction de critères prédéfinis, comme par exemple, la capacité du véhicule. Nous analyserons la gestion de tournées plus en détail dans le prochain chapitre.

2.9. Conclusion du chapitre

Nous avons présenté dans ce chapitre deux modèles issus de techniques de SCND. L'objectif était de déterminer la localisation optimale d'un dépôt pour servir au meilleur coût les clients de l'entreprise. La réponse à notre première hypothèse est positive. Il existe bel et bien une meilleure localisation de dépôt par rapport à la situation actuelle. En ce qui concerne notre seconde hypothèse, la réponse est négative. Il n'existe pas de meilleure localisation qui minimise le coût de transport que la situation actuelle.

Nous pouvons dès lors répondre à notre question de recherche par la négative. Il n'est pas opportun de choisir une localisation qui engendre un coût supérieur à la solution courante. Notre recommandation est de conserver le dépôt actuel pour servir la demande des produits pour la Belgique.

Deuxième Partie

Chapitre 3

3. Optimisation de tournées

3.1. Traveling Salesman Problem

Initialement, les premiers problèmes d'optimisation de tournées ont été posés comme résoudre le Traveling Salesman Problem (TSP) ou problème du voyageur de commerce. C'est un problème d'optimisation combinatoire qui peut être décrit comme un problème visant à trouver la distance minimum d'une tournée entre *n* villes en commençant et en finissant la tournée dans la même ville et en visitant chaque ville exactement une fois.

Dans un graphe $G = (V, A)$ où $V = \{v1, \ldots, v\ n\}$ est un set de sommets (ou nœuds) et $A = \{(v\ i, v\ j) \mid v\ i\ , v\ j \in v, i \neq j\}$est un set d'arêtes, ensemble avec un coût non-négatif (ou distance) matrice $C = (C\ ij)$ associé à *A*. Le problème est considéré comme symétrique (STSP) si $C\ ij = C\ ji$ pour tous les $(v\ i, v\ j) \in A$ et asymétrique (ATSP) sinon.

Malgré son énoncé relativement simple, le TSP est un problème d'optimisation combinatoire qui a inspiré l'élaboration de nombreux algorithmes pour le résoudre de manière de plus en plus efficace. Le TSP est un problème de type NP-complet c'est-à-dire qu'il n'existe pas d'algorithme connu capable de résoudre toutes les instances du problème en temps polynomial.

3.2. Vehicle Routing Problem

Le Vehicle Routing Problem (VRP) selon Kumar et Panneerselvam (2012) est utilisé pour définir un trajet optimal pour une flotte de véhicules dans le but de servir un certain nombre de clients sous certaines contraintes. Le VRP est un problème qui a été abondamment étudié pour ses applications multiples et pour l'importance de réussir à déterminer des stratégies efficaces pour réduire les coûts opérationnels dans les réseaux de distribution. Ce problème est de type NP-Difficile. Les meilleures approches exactes ne peuvent résoudre des énoncés qu'entre 50 et 100 instances pour garantir l'optimalité selon les variantes et le temps de calcul nécessaire. En conséquence, l'approche préférée est d'utiliser des algorithmes approximatifs

qui sont capables de trouver des solutions de grande qualité endéans un temps limité. Cette approche heuristique propose de trouver une solution raisonnable sans garantie qu'il s'agisse de la solution optimale.

Le VRP classique est défini comme suit :

Pour $G = (V,A)$ où G est un graphe dirigé où $V = \{0, \dots, n\}$ est l'ensemble de sommets et $A = \{(i, j) : i, j \in V, i \neq j\}$ est l'ensemble des arêtes. Le sommet 0 correspond au dépôt là où les autres sommets correspondent aux clients. Une flotte de m véhicules identiques de capacité Q est basée au dépôt. La taille de la flotte est donnée à priori ou est une variable de décision. Chaque client i a une demande positive q_i.

3.2.1. Variantes du VRP

Nous pouvons distinguer certaines variantes comme le VRP sous contrainte de capacité (CVRP) ou encore le VRP sous contrainte horaire (VRPTW). Ils visent à servir un certain nombre de clients avec des contraintes horaires de livraison en minimisant le coût (en termes de distance parcourue) sans violer la capacité du véhicule.

3.2.2. Capacited VRP (CVRP)

Le problème se résume à une flotte de véhicules identiques, localisés à un dépôt central, qui doivent être dirigés de manière optimale pour servir un ensemble de clients qui ont chacun une demande connue.

Le VRP sous contrainte de capacité (CVRP) est décrit comme un problème de la théorie des graphes: Soit G = (V, E) un graphe complet et non orienté où V = {0, ⋯, n} est l'ensemble des sommets et E l'ensemble des arêtes. L'ensemble des sommets Vc = {1, ⋯, n} correspond à n clients, tandis que le sommet 0 correspond au dépôt.

3.2.3. Time Windows VRP (VRPTW)

L'objectif du VRPTW est de servir un nombre de clients endéans des plages horaires prédéfinies à un coût minimum (en termes de distance parcourue), sans pour autant violer les contraintes de capacité et de temps de parcours total de chaque véhicule. Les problèmes d'optimisation combinatoire de ce type sont non polynomiaux (NP-durs).

Les contraintes du problème de tournée des véhicules avec des plages horaires (VRPTW) consistent en un ensemble de véhicules identiques, un nœud de dépôt central, un ensemble de

nœuds clients et un réseau reliant le dépôt et les clients. Il y a $N + 1$ clients et K véhicules. Le nœud du dépôt est désigné comme client 0. Chaque arc du réseau représente une connexion entre deux nœuds et indique également la direction dans laquelle il se déplace. Chaque itinéraire commence à partir du dépôt. Le nombre de routes dans le réseau est égal au nombre de véhicules utilisés. Un véhicule est dédié à un itinéraire. Un coût c_{ij} et un temps de parcours t_{ij} sont associés à chaque arc du réseau.

Solomon (1987) propose un modèle pour un problème de VRPTW. Toutes les distances sont représentées par la distance euclidienne, et la vitesse de tous les véhicules est supposée être unitaire. Autrement dit, il faut une unité de temps pour parcourir une unité de distance. Cette hypothèse rend le problème plus simple car, numériquement, le coût de transport c_{ij}, le temps de parcours t_{ij} et la distance euclidienne entre les nœuds clients sont égaux.

Chaque client ne peut être visité qu'une seule fois par l'un des véhicules. Chaque véhicule a la même capacité q_k et chaque client a une demande variable m_i. La capacité du véhicule q_k doit être supérieure ou égale à la somme de toutes les demandes sur la route parcourue par le véhicule k, ce qui signifie qu'aucun véhicule ne peut être surchargé. La contrainte de plage horaire est indiquée par un intervalle de temps prédéfini, compte tenu de l'heure d'arrivée la plus proche et de la dernière heure d'arrivée. Les véhicules doivent arriver chez les clients au plus tard à la dernière heure d'arrivée. Si les véhicules arrivent plus tôt que l'heure d'arrivée la plus proche, le véhicule est mis en attente. Chaque client impose également un temps de service à l'itinéraire, en tenant compte du temps de chargement / déchargement des marchandises. Dans les instances de Solomon (1987), le temps de service est supposé unique indépendamment de la quantité de charge à traiter. Les véhicules sont également supposés compléter leurs itinéraires individuels dans un temps total de parcours, qui est essentiellement la plage horaire du dépôt.

Variables de décisions principales

Il existe trois types de variables de décision principales dans un VRPTW. La variable de décision principale x_{ijk} $(i, j \in \{0, 1, 2, \cdots, N\}; k \in \{1, 2, ..., K\}; i \neq j)$ est 1 si le véhicule k se déplace du client i au client j, et 0 sinon. La variable de décision T_i indique le moment où un véhicule arrive chez le client, et w_i indique le temps d'attente au client i. L'objectif est de concevoir un réseau qui satisfasse à toutes les contraintes tout en minimisant le coût total du voyage.

Paramètres

K nombre total de véhicules
N nombre total de clients
c_{ij} coût sur l'arête entre le client i et le client j
t_{ij} temps de transport entre le client i et le client j
m_i demande du client i
q_k capacité du véhicule k
e_i temps d'arrivée au plus tôt au client i
l_i temps d'arrivée au plus tard au client i
f_i temps de déchargement au client i
r_k Temps de transport maximum octroyé au véhicule k

La contrainte (1) assure que le nombre maximum de véhicules disponibles soit respecté. La contrainte (2) assure que pour chaque véhicule il y a exactement une arête sortant du dépôt qui est sélectionné. En parallèle, la contrainte (3) assure que, pour chaque véhicule, il y a une arête entrant vers le nœud par rapport au dépôt (I=0). Ces deux contraintes mises ensemble (2 et 3) assurent qu'il y a une tournée complète pour chaque véhicule. La contrainte (4) assure que pour chaque client i, seulement une arête émane de ce nœud par véhicule. La contrainte (5) assure que pour chaque client j, seulement une arête par véhicule y entre. L'ensemble des contraintes, (4) et (5), assure que chaque véhicule visite chaque client une et une seule fois. La contrainte (6) assure que pour chaque véhicule, la demande totale allouée à chaque véhicule est inférieure ou égale à la capacité du véhicule. La contrainte (7) assure que le temps total de transport pour la tournée de chaque véhicule est inférieur ou égal au temps maximal de transport alloué à ce véhicule. La contrainte (8) définit que le temps d'arrivée, le temps d'attente et le temps de déchargement pour chaque véhicule au dépôt est égal à 0. La contrainte (9) garantit que l'heure d'arrivée de chaque véhicule au client j est inférieure à l'heure d'arrivée spécifiée pour ce client. La contrainte (10) garantit que la somme des heures d'arrivée et des temps d'attente de chaque véhicule à chaque client i est plus grande que la plus petite durée d'arrivée à ce client et inférieure ou égale à la dernière heure d'arrivée chez ce client i, $i = 1, 2, 3, ..., N$. L'ensemble des contraintes (8), (9) et(10) définissent les plages horaires. Ces formulations spécifient complètement les solutions réalisables pour le VRPTW.

Formulation Solomon (1987)

Minimiser $\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{j\neq i,k=1}^{K} c_{ij}x_{ijk}$

La fonction objective minimise le coût total de transport de tous les véhicules pour leurs tournées respectives.

Sous contraintes :

$$\sum_{k=1}^{K}\sum_{j=1}^{N} x_{ijk} \leq K, \quad \text{for } i = 0 \qquad (1)$$

$$\sum_{j=1}^{N} x_{ijk} = 1 \quad \text{for } i = 0 \text{ and } k \in \{1,\cdots,K\} \qquad (2)$$

$$\sum_{J=1}^{N} x_{jik} = 1 \quad \text{for } i = 0 \text{ and } k \in \{1,\cdots,K\} \qquad (3)$$

$$\sum_{k=1}^{K}\sum_{j=0,j\neq i}^{N} x_{ijk} = 1, \quad \text{for } i \in \{1,\cdots,N\} \qquad (4)$$

$$\sum_{k=1}^{K}\sum_{i=0,i\neq j}^{N} x_{ijk} = 1, \quad \text{for } j \in \{1,\cdots,N\} \qquad (5)$$

$$\sum_{i=1}^{N} m_i \sum_{j=0,j\neq i}^{N} x_{ijk} \leq q_k, \quad \text{for } k \in \{1,\cdots,K\} \qquad (6)$$

$$\sum_{i=0}^{N}\sum_{j=0,j\neq i}^{N} x_{ijk}\left(t_{ij} + f_i + w_i\right) \leq r_k, \quad \text{for } k \in \{1,\cdots,K\} \qquad (7)$$

$$T_0 = w_0 = f_0 = 0 \qquad (8)$$

$$\sum_{k=1}^{K}\sum_{i=0,i\neq j}^{N} x_{ijk}\left(T_i + t_{ij} + f_i + w_i\right) \leq T_j, \quad \text{for } i \in \{1,\cdots,K\} \qquad (9)$$

$$e_i \leq \left(T_i + w_i\right) \leq l_i \quad \text{for } i \in \{1,\cdots,N\} \qquad (10)$$

3.3. Conclusion du chapitre

Nous avons présenté dans ce chapitre le problème d'organisation de tournées dénommé dans ce mémoire par VRP. Nous avons également présenté deux de ces variantes dont le VRPTW. Le VRP est un problème combinatoire NP-complet. Il n'existe pas d'algorithme en temps polynomial (pour $P = NP$) capable de résoudre toutes les instances d'un problème pour de grandes instances. Bien que des approches exactes soient réalisables, elles deviennent impossibles à résoudre dans un temps raisonnable à mesure que le nombre d'instances augmente. Une large gamme d'heuristiques ont été développées pour trouver une solution raisonnable avec l'inconvénient de trouver une seule solution optimale dans un immense espace de solutions. Les métaheuristiques sont des stratégies plus sophistiquées qui guident des procédures heuristiques pour créer une procédure capable de s'échapper d'un optimum local pour explorer d'autres régions du champ des solutions (Gendreau & Potvin, 2010). Nous consacrons le prochain chapitre pour présenter les plus importantes métaheuristiques décrites dans la littérature.

Chapitre 4

4. Approches métaheuristiques

"Une métaheuristique peut être définie comme un processus de génération itérative qui guide une heuristique subordonnée en combinant des concepts intelligents différents pour explorer et exploiter l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer les informations afin de trouver efficacement des solutions quasi-optimales.» (Osman et Laporte 1996). Les stratégies de recherche des différentes métaheuristiques sont fortement dépendantes de la métaheuristique en elle-même.

Selon Blum et Roli (2003), Il y a plusieurs façons de classifier et de décrire des métaheuristiques. En fonction des caractéristiques choisies qui les différencient, plusieurs classifications sont possibles et chacune d'elles découle d'un point de vue spécifique. Une caractéristique qui peut être utilisée pour la classification des métaheuristiques est le nombre de solutions utilisées au même moment. L'algorithme fonctionne-t-il sur une population ou sur une simple solution à tout moment? Les algorithmes travaillant sur des solutions uniques sont appelés méthodes de trajectoire et englobent les métaheuristiques basées sur la recherche locale, comme la recherche Tabou (Tabu Search (TS)), la Recherche Locale Itérative (Iterated Local Search (ILS)) et la recherche à voisinage variable (Variable Neighborhood Search (VNS)). Elles partagent toutes la propriété de décrire une trajectoire dans l'espace de recherche pendant le processus de recherche. Les métaheuristiques basées sur la population comme le calcul évolutif (Evolutionary Computation (EC)) et les colonies de fourmis (Ant Colony Optimization (ACO)), au contraire, effectuent un processus de recherche qui décrit l'évolution d'un ensemble de points dans l'espace de recherche. Dans ce chapitre, nous décrirons les métaheuristiques les plus importantes sur base de ces deux classes pour résoudre des problèmes d'optimisation combinatoire.

4.1. Méthodes par trajectoire

Ce terme est utilisé parce que le processus de recherche utilisé par ces méthodes est caractérisé par une trajectoire dans l'espace de recherche. Ce processus de recherche peut être vu comme l'évolution dans le temps d'un système dynamique discret (Bar-Yam 1997 ; Devaney 1989). L'algorithme débute au départ d'un état initial et décrit une trajectoire dans l'espace de recherche. La dynamique de système dépend de la stratégie utilisée. Les

algorithmes les plus simples génèrent une trajectoire en deux phases : une phase transitoire suivie d'un attracteur (un point fixe, une boucle ou un attracteur complexe) alors que les algorithmes plus complexes ne peuvent être subdivisés dans ces deux phases. La dynamique est le résultat de la combinaison de l'algorithme, de la représentation du problème et des instances du problème. En fait, la représentation du problème combinée avec la structure du voisinage définissent l'espace de recherche. L'algorithme décrit la stratégie utilisée pour parcourir l'espace de recherche qui est lui-même défini par les caractéristiques du problème à résoudre.

4.1.1. Recherche locale basique : amélioration itérative

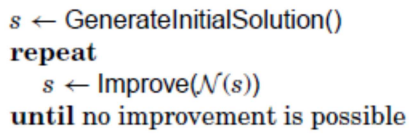

Figure 1 : Algorithme Amélioration Itérative

Un algorithme de recherche locale est aussi appelé amélioration itérative à partir du moment où chaque perturbation, le choix d'une solution *s'* dans le voisinage *(N(s))* par rapport à une solution *s*, est réalisé si la solution trouvée est meilleure que la solution initiale. L'algorithme s'arrête dès qu'il trouve un minima local. Par conséquent, la performance dépend fortement de la définition de *S*, *f* et *N*. certaines méthodes ont été ajoutées pour permettre à l'algorithme de s'évader du piège d'un minima local. Cela comprend la définition d'un maximum de temps de calcul, d'un nombre maximum d'itérations à accomplir, une solution *s* où *f(s)* est inférieure à une certaine valeur prédéfinie ou encore un nombre maximum d'itérations sans qu'une meilleure solution soit trouvée. Les résultats obtenus par cette méthode sont généralement insatisfaisants pour les problèmes d'optimisation combinatoire.

4.1.2. Recuit simulé

Le Recuit Simulé ou Simulated Annealing (SA) est communément l'un des premiers algorithmes qui avait une stratégie explicite pour échapper aux minimas locaux. Les origines de l'algorithme proviennent de la mécanique statistique et s'inspire des alternances de refroidissement et de recuit en métallurgie. Il fut le premier présenté comme un algorithme de recherche sur des problèmes d'optimisation combinatoire (Kirkpatrick et al. ,1983 ; Cerny 1985). L'idée fondamentale est de permettre des mouvements entraînant des solutions de moins bonnes qualités que la solution actuelle (mouvements ascendants) afin d'échapper aux

minima locaux. La probabilité de faire un tel mouvement est diminuée graduellement au cours de la recherche.

```
s ← GenerateInitialSolution()
T ← T_0
while termination conditions not met do
  s' ← PickAtRandom(N(s))
  if (f(s') < f(s)) then
     s ← s'            % s' replaces s
  else
     Accept s' as new solution with probability p(T, s', s)
  endif
  Update(T)
endwhile
```

Figure 2 : Algorithme Simulated Annealings (SA)

L'algorithme commence en générant une solution initiale et en initialisant le paramètre de température T. Ensuite, à chaque itération, une solution $s' \in N(s)$ est choisie et acceptée en fonction de $f(s)$, $f(s')$ et T. s' remplace s si $f(s') < f(s)$ ou, dans le cas où $f(s') \geq f(s)$ avec une probabilité qui est une fonction de T et de $f(s') - f(s)$. La probabilité est généralement calculée suivant une distribution de Boltzmann $exp\left(-\frac{f(s')-f(s)}{T}\right)$. (Blum et Roli, 2003)

La température T est diminuée pendant le processus de recherche. Ce qui implique que, au début de la recherche, la probabilité d'accepter des mouvements ascendants est plus importante et celle-ci diminue graduellement pour aboutir à un simple algorithme d'amélioration itérative. La probabilité d'accepter des mouvements ascendants est contrôlée par deux facteurs : la différence entre les deux fonctions $f(s')$ et $f(s)$ et la température T. le choix du plan de décroissance de T est crucial pour la performance de cet algorithme. Le plan de décroissance de T définit la valeur de T à chaque itération k, $T_{k+1} = Q(T_k, k)$ où $Q(T_k, k)$ est une fonction de la température et du nombre d'itérations. La règle de décroissance peut varier durant la recherche. Par exemple, au début de la recherche, T peut être constant ou linéairement diminué dans le but d'échantillonner l'espace de recherche et puis y appliquer une loi géométrique $T_{k+1} = \alpha\, T_k$ où $\alpha \in (0,1)$ pour converger vers un minimum local à la fin de la recherche. Le recuit simulé a été appliqué à quelques problèmes d'optimisation combinatoire comme l'optimisation quadratique (Quadratic Assignment Problem (QAP)) (Connoly 1990) et l'ordonnancement des tâches (Job Shop Scheduling Problem (JSS)) (Van Laarhoven et al. 1992). Bien que la méthode soit aussi utilisée pour solutionner le VRP (Afifi 2013), le recuit simulé est généralement utilisé comme un composante des métaheuristiques plutôt qu'en algorithme de recherche autonome.

4.1.3. La Recherche Tabou

La Recherche Tabou ou Tabu Search (TS) est une des métaheuristiques les plus utilisées pour les problèmes d'optimisation combinatoire. L'idée de base fut introduite par Glover (1986) et décrite ensuite par Glover et Laguna (1997). L'algorithme simple de TS utilise une mémoire à court terme comme ingrédient de base pour appliquer une amélioration de la recherche locale et permettre de s'échapper d'un minima local et éviter les cycles.

```
s ← GenerateInitialSolution()
TabuList ← ∅
while termination conditions not met do
  s ← ChooseBestOf(N(s) \ TabuList)
  Update(TabuList)
endwhile
```

Figure 3 : Algorithme simple Tabu Search (TS)

La mémoire à court terme est implémentée comme une liste tabou qui conserve les solutions les plus récentes trouvées et interdit à l'algorithme d'à nouveau les explorer. Le voisinage de la solution courante est donc restreint aux solutions qui n'appartiennent pas à la liste tabou. Nous l'appellerons l'ensemble autorisé. A chaque itération, la meilleure solution de cet ensemble autorisé est choisie comme solution courante. De plus, cette solution est ajoutée à la liste tabou et une des solutions déjà présente dans la liste est retirée. A cause de cette restriction de voisinage, la recherche tabou peut être considérée comme une technique de recherche de voisinage dynamique (Stützle 1999b). L'algorithme s'arrête quand une condition d'arrêt est atteinte ou si l'ensemble autorisé est vide c'est-à-dire que toutes les solutions de *N(s)* sont interdites par la liste tabou.

La longueur *l* (Tabu tenure) de la liste tabou contrôle la mémoire nécessaire de la procédure de recherche. Avec une petite valeur de *l*, la recherche se concentre sur de petites zones du voisinage de recherche. A l'opposé, une grande valeur permet l'exploration de larges zones. La variation de cette valeur durant la recherche conduit vers un algorithme plus robuste en réinitialisant périodiquement au hasard la valeur de *l* à partir de l'intervalle *[l min, l max]*. D'autres moyens plus élaborés pour créer la tabu tenure ont été décrits par Glover (1990).

Cependant, l'implémentation d'une mémoire à court terme comme une liste qui contient des solutions complètes n'est pas pratique parce que gérer une liste des solutions n'est pas efficient. Dès lors, au lieu des solutions en elles-mêmes, on conserve des attributs de solution. Des attributs sont habituellement des parties de solution, mouvements ou différences entre

deux solutions. Conserver des attributs est plus efficient mais génère une perte d'informations. Et comme interdire un attribut implique d'assigner un statut tabou à probablement plus que une solution, il est possible que des solutions ignorées de bonne qualité soient exclues de l'ensemble autorisé des solutions. Pour pallier à ce problème, un critère d'aspiration est défini qui permet d'inclure une solution à l'ensemble autorisé bien que cette solution soit interdite par une liste tabou. Le critère d'aspiration le plus utilisé est celui qui sélectionne une meilleure solution que la meilleure solution courante.

$s \leftarrow$ GenerateInitialSolution()
InitializeTabuLists($TL_1, \ldots, TL_r$)
$k \leftarrow 0$
while termination conditions not met **do**
  $AllowedSet(s,k) \leftarrow \{s' \in \mathcal{N}(s) \mid s$ does not violate a tabu condition,
  or it satisfies at least one aspiration condition$\}$
  $s \leftarrow$ ChooseBestOf($AllowedSet(s,k)$)
  UpdateTabuListsAndAspirationConditions()
  $k \leftarrow k+1$
endwhile

Figure 4 : Algorithme Tabu Search (TS)

Les informations collectées durant la procédure de recherche sont particulièrement utiles pour guider la stratégie appliquée à l'algorithme. Ce type de mémoire à long terme est appliqué à la recherche tabou selon quatre principes : récence, fréquence, influence et qualité. Ce dernier principe nous intéressera plus particulièrement car il réfère à l'accumulation et à l'extraction d'information de l'espace de recherche dans le but d'identifier de bons composés de solutions. D'autres métaheuristiques comme les colonies de fourmi (ACO) utilisent ce principe.

La recherche tabou a été appliquée à une grande variété de problèmes d'optimisation combinatoire dont entre autres le VRP (Gendreau et al. 2001).

4.2.Méthodes de recherche locale explorative

4.2.1. The Greedy Randomized Adaptative Search Procedure (GRASP)

Le GRASP (Feo et Resende 1995) est une simple métaheuristique qui combine une construction heuristique et une recherche locale (Pitsoulis et Resende 1995).

Le GRASP est une procédure itérative composée de deux phases : construction de solution et amélioration de la solution. La meilleure solution trouvée est proposée à la fin de la recherche.

Le mécanisme de construction de la solution est caractérisé par deux ingrédients principaux : une heuristique constructive dynamique et une randomisation.

```
while termination conditions not met do
  s ← ConstructGreedyRandomizedSolution()
  ApplyLocalSearch(s)
  MemorizeBestFoundSolution()
endwhile
```

Figure 5 : Algorithme Greedy Randomized Adaptative Search Procedure

En assumant qu'une solution *s* consiste en un sous-ensemble d'un ensemble d'éléments, la solution est construite en intégrant étape par étape un nouvel élément à la fois. Le choix du prochain élément est fait au hasard à partir d'une liste candidate. Les éléments sont classés par critère heuristique qui leur donne un score en fonction du bénéfice s'ils sont insérés dans la solution. La liste des candidats est composée des meilleurs α éléments. Les valeurs sont adaptées à chaque étape donc le score des éléments change durant la phase de construction en fonction des différents choix. Ce type d'heuristique constructive est de type dynamique au contraire d'une construction statique qui alloue un score aux différents éléments uniquement au début de la construction. Par exemple, une heuristique statique pour le TSP, un sous-problème du VRP, le coût de l'arc le plus faible reçoit le meilleur score.

La longueur α de la liste de candidats détermine la robustesse du biais heuristique. Dès lors, α est un paramètre critique qui influence l'échantillonnage de l'espace de recherche. Selon Pitsoulis et Resende (2002), les thèmes les plus importants pour définir α sont listés. Le plus simple étant de conserver α constant. Il peut être aussi adapté au hasard ou à l'aide d'une méthode adaptative à chaque itération.

La seconde phase de l'algorithme est un procédé de recherche locale. Il peut s'agir d'un algorithme simple d'amélioration itérative ou plus une technique plus avancée comme SA ou TS. Le GRASP est dès lors effectif si deux conditions sont satisfaites :

1) Le mécanisme de construction de la solution échantillonne les régions les plus prometteuses de l'espace de recherche.
2) Les solutions élaborées par l'heuristique constructive appartiennent à des bassins d'attraction de solutions de différents minimas locaux.

La description du GRASP comme donné indique qu'un GRASP basique n'utilise pas l'historique de la recherche. Le seul besoin en mémoire est pour enregistrer les instances du

problème et la meilleure solution courante. C'est une des raisons pour laquelle le GRASP est moins efficient que certaines autres métaheuristiques. Néanmoins, en raison de sa simplicité, il est capable de trouver de plus ou moins bonnes solutions très rapidement et facilement intégrées à d'autres techniques de recherche.

4.2.2. La recherche à voisinage variable

La recherche à voisinage variable ou Variable Neighborhood Search (VNS) est une métaheuristique (Hansen and Mladenovic 1999,2001) qui applique une stratégie basée sur un changement dynamique des structures du voisinage. Lors de la phase d'initialisation, un ensemble de structures de voisinages doit être défini. Ces structures peuvent être choisies aléatoirement mais le plus souvent une séquence $|N1| < |N2| < \ldots |N_{kmax}|$ où la croissance de la cardinalité est définie. Alors une solution initiale est générée, l'index de voisinage est initialisé et l'algorithme itère jusqu'à ce qu'une condition d'arrêt soit rencontrée.

```
Select a set of neighborhood structures N_k, k = 1, ..., k_max
s ← GenerateInitialSolution()
while termination conditions not met do
  k ← 1
  while k < k_max do            % Inner loop
    s' ← PickAtRandom(N_k(s))          % Shaking phase
    s'' ← LocalSearch(s')
    if (f(s'') < f(s)) then
      s ← s''
      k ← 1
    else
      k ← k + 1
    endif
  endwhile
endwhile
```

Figure 6 : Algorithme Variable Neighborhood Search

Le cycle principal est composé de trois phases : perturbation, recherche locale et mouvement.

Dans la phase de perturbation, une solution *s'* dans le voisinage *k* de la solution courante *s* est aléatoirement sélectionnée. C'est alors que *s'* devient le point de départ de la recherche locale. La recherche locale peut utiliser n'importe quelle structure de voisinage de l'ensemble des structures *Nk, k = 1, …, k_{max}*. A la fin de la recherche locale, la nouvelle solution *s''* est comparée avec *s*. Si c'est meilleur alors *s''* remplace *s* et l'algorithme recommence avec *k = 1* Sinon, k est incrémenté et une nouvelle phase de perturbation est lancée en utilisant un voisinage différent. L'objectif de cette phase de perturbation est de proposer un bon point de départ à la recherche locale. Sans être trop éloigné de *s*, l'efficacité de cette stratégie peut être expliquée par le fait qu'une mauvaise place dans l'espace de recherche défini par un certain

voisinage peut être une bonne place pour un autre voisinage. Dans lequel une bonne place dans l'espace de recherche est une zone où un bon minimum local peut être trouvé. Selon Jones (1995a, 1995b), la structure du voisinage détermine les propriétés topologiques de l'espace de recherche et les propriétés d'un voisinage sont en général différentes de celles des autres voisinages. Dès lors, une stratégie de recherche aura un impact différent sur eux.

Le choix de la structure de voisinage est le point critique de cette métaheuristique. Certaines variantes ont dès lors été développées. La propriété est directement exploitée par une recherche locale appelée Variable Neighborhood Descent (VND) qui applique une recherche locale d'amélioration et si un minima local est trouvé, la recherche passe à une autre structure du voisinage. Une autre variante est obtenue par décomposition du problème, le Variable Neighborhood Decomposition Search (VNDS) s'apparente à un VNS classique sauf que les structures de voisinage sont définies comme des sous-problèmes. Pour chaque solution, tous les attributs sont conservés fixes excepté un nombre k d'entre eux. Pour chaque k, une structure de voisinage N_k est définie. La recherche locale ne tient compte que des variables définies pour le sous-problème.

La décision de réaliser un mouvement peut varier également. Le critère d'acceptance fondé sur une amélioration est profondément orienté vers une descente des solutions possibles et n'est pas efficient pour explorer tout l'espace de recherche. Si un minima local est borné, le VNS peut trouver très rapidement l'optimum endéans ces bornes mais il ne dispose d'aucune règle pour s'échapper de ces bornes et d'en trouver d'autres. Skewed VNS (SVNS) utilise un critère d'acceptation plus large qui prend aussi en compte la distance à partir de la solution courante. Une mesure de distance entre les solutions est à définir formellement.

4.2.3. La recherche locale guidée

La recherche locale guidée ou Guided Local Search (GLS) utilise une autre approche qui vise à changer dynamiquement la fonction objective au lieu de changer de manière dynamique les voisinages comme le font TS et VNS. Selon Voudouris et Tsang (2003) le principe de base du GLS est de graduellement modifier l'espace de recherche pour s'échapper d'un minimum local. L'ensemble des solutions et les structures de voisinage sont fixes alors que la fonction f est modifiée dynamiquement dans le but de rendre le minimum local moins désirable. La méthode est définie par l'utilisation de tous types de caractéristiques ou propriétés pour discriminer entre les différentes solutions. Par exemple, les arcs entre les paires de villes pour le TSP.

Un indicateur de fonction $I_i\,(s)$ indique si la caractéristique i est présente dans la solution s. $I_i\,(s) = 1$ si i est présent pour s sinon est égal à 0. La fonction objective f est modifiée pour atteindre une fonction f' en ajoutant un terme dépendant des m caractéristiques :

$$f'(s) = f(s) + \lambda \sum_{i=1}^{m} pi \,.\, I_i(s)$$

où p_i est un paramètre de pénalité et λ est le paramètre de régularisation. Le paramètre de pénalité donne de l'importance à la caractéristique. Au plus p_i est grand, au plus est grande l'importance d'avoir la caractéristique i dans la solution. Pendant que le paramètre de régularisation rétablit la pertinence des caractéristiques par rapport à la fonction objective.

```
s ← GenerateInitialSolution()
while termination conditions not met do
  s ← LocalSearch(s, f')
  for all feature i with maximum utility Util(s, i) do
    p_i ← p_i + 1
  endfor
  Update(f', p)          % p is the penalty vector
endwhile
```

Figure 7 : Algorithme Guided Local Search (GLS)

L'algorithme commence au départ d'une solution initiale et applique une recherche locale jusqu'à ce qu'un minimum local soit atteint. Le champ des pénalités $p = (p_1, \ldots, p_m)$ est mis à jour en incrémentant certaines des pénalités et la recherche locale est relancée. Les caractéristiques qui sont pénalisées sont celles qui ont une utilité maximum :

$$Util(s, i) = I_i(s) \,.\, \frac{ci}{1+pi}$$

Où c_i sont des coûts ajoutés à chaque caractéristique i donnant une évaluation heuristique de l'importance relative des caractéristiques par rapport aux autres. Néanmoins, le coût est équilibré par le paramètre de pénalité pour empêcher l'algorithme d'être entièrement biaisé par le coût et de le rendre ainsi sensible pour l'historique de recherche. La mise à jour des pénalités peut être modifiée en y appliquant une règle multiplicative au lieu d'une simple règle d'incrémentation. Elle est appliquée avec une fréquence moindre que la règle d'incrémentation pour prévenir que l'espace de recherche devienne trop robuste rapidement. Les pénalités sont souvent très sensibles aux instances du problème.

GLS a été implémenté avec succès au problème VR (Kilby et al. 1999) et au TSP (Voudouris et Tsang 1999)

4.2.4. La recherche locale itérée

La recherche locale itérée ou Iterated Local Search (ILS) est une simple mais puissante métaheuristique (Stüzle 1999a, 1999b ; Lourenço et al. 2001, 2002 ; Martin et al. 1991) qui applique une recherche locale à une solution initiale jusqu'à trouver une minimum local et puis qui perturbe cette solution avant de relancer une recherche locale. L'importance de la perturbation est à prendre en considération. Trop proche, cela risque d'empêcher le système de s'échapper du bassin d'attraction de l'optimum local juste trouvé et si trop éloigné alors l'algorithme se transforme en une recherche locale aléatoire.

Une recherche locale est efficiente si elle permet de trouver un bon minimum local c'est-à-dire trouver les états du bassin d'attraction de ce minimum. Une recherche efficace peut être décrite comme une trajectoire dans l'ensemble d'un minimum local $\hat{S}$ à la place d'un ensemble S de tous les états. Il n'est néanmoins pas possible dans la majorité des cas d'assigner une structure de voisinage pour $\hat{S}$. Dès lors, une trajectoire le long du minimum local $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_t$ est réalisé sans explicitement introduire une structure de voisinage.

$s_0 \leftarrow$ GenerateInitialSolution()
$\hat{s} \leftarrow$ LocalSearch(s_0)
while termination conditions not met **do**
  $s' \leftarrow$ Perturbation($\hat{s}$, *history*)
  $\hat{s}' \leftarrow$ LocalSearch(s')
  $\hat{s} \leftarrow$ ApplyAcceptanceCriterion($\hat{s}, \hat{s}'$, *history*)
endwhile

Figure 8 : Algorithme Iterated Local Search

L'algorithme exécute une recherche locale d'un état initial *s* jusqu'à ce qu'un minimum local $\hat{s}$ soit trouvé, perturbe $\hat{s}$ et obtient *s'*. Ensuite il exécute une recherche locale à partir de *s'* jusqu'à ce qu'un minimum local $\hat{s}$' soit atteint et finalement sur base d'un critère d'acceptation de décider de remplacer $\hat{s} \leftarrow \hat{s}$' avant de perturber à nouveau. Le critère d'acceptation agit comme une contrebalance et donne un retour sur l'action de perturbation en fonction des caractéristiques du nouveau minimum local.

En premier, la construction d'une solution initiale peut être très rapide. Le moyen le plus rapide étant de la sélectionner au hasard. Une méthode de construction guidée par heuristique peut également être adoptée. En second, la perturbation est en général non déterministe pour éviter un effet de boucle. Sa caractéristique principale est la force définie par le nombre de changement effectué sur la solution courante. Cette force peut être fixe ou variable bien qu'une force variable soit plus efficace. Au plus la taille du problème est importante, au plus

la force doit être grande également. La force peut également être adaptative dans certains cas en alternant intensification ou diversification quand cela semble préférable. Une dernière approche est d'utiliser un mécanisme de perturbation différent pour la recherche locale par rapport à celle utilisée dans l'algorithme principal. Troisièmement, le critère d'acceptation où deux exemples extrêmes consistent à accepter le nouveau minimum local en cas d'amélioration seulement ou dans tous les cas peu importe si il y a une amélioration. Entre les deux existent plusieurs possibilités. Entre autres l'utilisation d'un recuit simulé pour approuver les solutions qui n'améliorent pas la solution courante par l'utilisation d'une fonction de température *T* et la différence des valeurs de la fonction objective. Le schéma de refroidissement peut-être fixe ou variable. A partir du moment où l'intensification ne semble plus adaptée, une période de diversification est nécessaire où la température est augmentée.

La recherche locale itérée ou ILS a été appliqué avec succès au TSP (Johnson et McGeoch 1997)

4.3.Méthodes de population

Les méthodes basées sur une population travaillent à chaque itération de l'algorithme sur un ensemble de solutions plutôt qu'une solution unique. Ces algorithmes proposent un moyen naturel d'explorer l'espace de recherche où la performance finale dépend fortement de la manipulation de cette population. Les méthodes les plus étudiées sont le calcul évolutif ou Evolutionary Computation (EC) et l'optimisation par colonies de fourmis ou Ant Colony Optimization (ACO). Pour les algorithmes EC, une population d'individus est modifiée par des opérateurs recombinants et mutants. Pour l'ACO, une colonie de fourmis artificielle est utilisée pour construire des solutions guidées par des dépôts de phéromones et par une information heuristique.

4.3.1. Calcul évolutif

Le calcul évolutif ou Evolutionary Computation (EC) sont des algorithmes inspirés par les capacités de la nature à s'adapter à un environnement changeant. Ils peuvent être succinctement caractérisés par des modèles informatiques de processus naturels évolutifs. Ils utilisent deux types d'opérateurs, les combinaisons et les mutations. La principale force de ces algorithmes est la sélection des individus basée sur leur aptitude. Cela peut être une valeur de la fonction objective, le résultat d'une simulation ou tous types de mesures de qualité. Les

individus ayant un plus haut score d'aptitude ont une probabilité plus importante d'être choisi pour faire partie de la population à la prochaine itération.

P représente la population des individus. Une population *A* de descendants est générée en appliquant les opérateurs combinants et mutants et les individus de la prochaine population sont sélectionnés à partir de l'union de l'ancienne population et de la population de descendants.

```
P ← GenerateInitialPopulation()
Evaluate(P)
while termination conditions not met do
  P′ ← Recombine(P)
  P″ ← Mutate(P′)
  Evaluate(P″)
  P ← Select(P″ ∪ P)
endwhile
```

Figure 9 : Algorithme Evolutionary Computation (EC)

Description des individus

Les algorithmes d'EC gèrent des populations d'individus. Ces individus ne sont pas nécessairement des solutions du problème considéré mais ils peuvent être des solutions partielles, un ensemble de solutions ou tous types d'objets qui peuvent être transformés en une ou plusieurs solutions de manière structurée. Le choix d'une représentation appropriée est crucial pour le succès de ce type d'algorithme car il faut compter sur la différence entre représentation des solutions et les solutions en elles-mêmes.

Processus d'évolution

A chaque itération, il est décidé quels individus incorporeront la population de la prochaine itération. C'est réalisé via un schéma de sélection. Si on ne sélectionne que des descendants, on parle de replacement générationnel et si on incorpore également des individus appartenant à la population courante, on parle de processus d'évolution stable. La plupart des algorithmes fonctionnent avec des populations fixes qui conservent toujours le meilleur individu dans la population courante. Au contraire, si la population est réduite à chaque itération aboutissant à la situation où un seul individu compose la population, cela peut devenir une condition d'arrêt de l'algorithme.

Structures de voisinage

Une fonction de voisinage $N_{EC} : I \rightarrow 2^I$ sur l'ensemble des individus I assigne $i \in I$ un ensemble d'individus $N_{EC}(i)$ I qui a la permission d'agir comme partenaires combinants pour créer une descendance. On parle de population non-structurée si chaque individu peut être combiné avec n'importe quel autre sinon on parle de population structurée.

Sources d'informations et infaisabilité des individus

La source d'informations la plus utilisée pour créer de nouveaux individus est un couple de parents mais selon Eiben et al. (1994) il existe également des opérateurs capables d'opérer sur plus que deux individus pour créer un nouvel individu. En recombinant des individus, il est possible que les descendants soient infaisables. L'action la plus simple est dès lors de rejeter les individus infaisables bien qu'il peut être difficile de détecter les individus infaisables, une stratégie de pénalité des individus par la fonction qui mesure la qualité des individus peut sembler plus appropriée. Une dernière possibilité est de tenter de réparer les solutions infaisables (Eiben et Ruttkay 1997)

Stratégie d'intensification

Pour beaucoup d'applications, il est bénéfique d'instaurer des mécanismes d'intensification pour augmenter l'aptitude des individus. Les algorithmes mémétiques (Moscato 1989, 1999) applique un algorithme de recherche locale à chaque individu d'une population pour aider à rapidement identifier de bonnes zones de l'espace de recherche. Une autre stratégie d'intensification consiste d'utiliser des opérateurs recombinants en essayant de combiner les bonnes propriétés des individus pour guider la recherche vers des zones ayant les bonnes propriétés également.

Stratégie de diversification

Une des principales difficultés des Algorithmes EC est la convergence prématurée vers des solutions sous-optimales. Dès lors, le mécanisme le plus simple pour diversifier la recherche est d'utiliser des opérateurs de mutation. La forme la plus simple est de réaliser une petite perturbation aléatoire sur un individu. D'autres moyens sont utilisés pour maintenir une diversité de la population. Comme par exemple, le partage des aptitudes (Goldberg et Richardson 1987) où l'aptitude reproductive allouée à un individu dans une population est

réduite proportionnellement au nombre d'individus qui partage la même région de l'espace de recherche.

Les algorithmes EC ont été appliqués avec succès à de nombreux problèmes d'optimisation combinatoire comme l'optimisation multi-objectifs (Coello Coello 2000).

4.4. Conclusion du chapitre

Nous avons présenté dans ce chapitre, plusieurs métaheuristiques présentant des options pour la résolution de problèmes liés au VRP. Bien que ces métaheuristiques soient différentes, dans le sens où, certaines sont basées sur une population (EC) et d'autres sur une trajectoire dans l'espace de recherche (SA, TS, ILS, VNS, GRASP), les méthodes par population sont meilleures dans l'identification de régions intéressantes de l'espace de recherche alors que les méthodes par trajectoire sont plus efficaces dans l'exploration des régions intéressantes de l'espace de recherche. Dans le prochain chapitre, nous présentons plus en détails les algorithmes de colonies de fourmis ou Ant Colony Optimization (ACO)

Chapitre 5

5. Algorithmes de colonies de fourmis

Selon Blum (2005), Les algorithmes de colonies de fourmis ou Ant Colony Optimization (ACO) sont une des plus récentes techniques d'optimisation approximative. Ils tirent leur inspiration de l'observation des techniques développées par de réelles colonies de fourmis pour trouver et exploiter des sources de nourriture. Le premier algorithme fut développé par Marco Dorigo (1992) et s'inspire de la méthode utilisée par ces insectes sociaux pour trouver le chemin le plus court entre les sources de nourriture et le nid. Lors de la collecte de nourriture, les fourmis explorent la région entourant leur nid de manière aléatoire. En se déplaçant, les fourmis laissent une piste de phéromones sur le sol que les autres fourmis peuvent détecter. Dès lors qu'une nouvelle fourmi quitte le nid pour collecter de la nourriture, elle est incitée à choisir le chemin où la concentration de phéromones laissées par les autres fourmis est la plus intense. Cette communication indirecte entre les fourmis via les pistes de phéromones leur permet de trouver le chemin le plus court entre la nourriture et le nid. Cette piste de phéromones constitue une mémoire globale pour l'algorithme et favorise la convergence vers de bonnes solutions tout en permettant aux fourmis d'explorer l'espace de recherche.

Les algorithmes ACO sont basés sur un modèle probabilistique paramétré. On utilise un modèle phéromone pour modéliser les pistes chimiques de phéromones. Les fourmis artificielles construisent des solutions en ajoutant des composés de solution définis à une solution partielle sous considération. Pour ce faire, les fourmis parcourent aléatoirement un graphe totalement connecté $G = (C, L)$ où les sommets C sont les composés de solution et l'ensemble L sont les connections. Quand un problème d'optimisation combinatoire contraint est considéré, les contraintes du problème Ω sont construites lors de la procédure de construction de façon qu'à chaque étape du processus uniquement les composés de solution possibles sont ajoutés à la solution partielle.

5.1. Ant System pour le TSP

Comme évoqué dans le chapitre 3, le problème du voyageur de commerce ou Traveling Salesman Problem (TSP) peut être décrit comme un problème visant à trouver la distance

minimum d'une tournée entre *n* villes en commençant et en finissant à la même ville tout en visitant chaque autre ville exactement une fois.

Concernant l'approche avec un algorithme AS, les arcs du graphe pour le TSP peuvent être considérés comme des composés de solutions où pour chaque $e_{i,j}$ est assignée une valeur de phéromone $\tau_{i,j}$. La tâche de chaque fourmi consiste en la construction d'une solution réalisable en une tournée. Chaque fourmi construit une solution comme suit. Premièrement, un des nœuds du graphe est choisi aléatoirement comme point de départ. Puis, la fourmi construit un tour en se déplaçant depuis le nœud où elle se situe jusqu'à un prochain nœud qui n'a pas encore été visité. A chaque étape, la valeur de l'arc traversé est ajoutée à la solution en construction. Dès qu'il n'y a plus de nœuds non visités, la fourmi termine la tournée en se déplaçant du nœud où elle se situe vers le nœud de départ. Cette manière de construction d'une solution implique que la fourmi a une mémoire *T* pour conserver les nœuds déjà visités. Chaque étape de la construction d'une solution est réalisé en assumant qu'une fourmi se trouvant au nœud v_i choisira le prochain nœud suivant la probabilité :

$$\mathbf{p}(e_{i,j}) = \frac{\tau_{i,j}}{\sum_{\{k \in \{1,\ldots,|V|\} | v_k \notin T\}} \tau_{i,k}}, \quad \forall j \in \{1,\ldots,|V|\},\ v_j \notin T.$$

Une fois que toutes les fourmis de la colonie ont réalisé leur solution, on applique un procédé d'évaporation des phéromones comme suit :

$$\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j}, \quad \forall \tau_{i,j} \in \mathcal{T},$$

Où $\mathcal{T}$ est l'ensemble de toutes les valeurs de phéromones. Puis les fourmis réalisent leur chemin de retour. Une fourmi ayant construit une solution s, réalise pour chaque ei,j ϵ s un dépôt de phéromones comme suit :

$$\tau_{i,j} \leftarrow \tau_{i,j} + \frac{Q}{f(s)},$$

Où Q est une constante positive et *f(s)* est la valeur de la fonction objective de la solution *s*. Le système est itéré jusqu'à ce qu'une condition d'arrêt soit satisfaite. Par exemple, une limite de temps.

Bien que l'algorithme AS a prouvé sa capacité à résoudre le problème d'optimisation discrète. Il obtient des résultats inférieurs par rapport aux meilleurs algorithmes de la discipline. Dès lors, plusieurs variantes ont vu le jour que nous détaillons dans la prochaine section. Elles sont toutes regroupées sous la dénomination de Ant Colony optimization (ACO) métaheuristique.

5.2.La métaheuristique ACO

Selon Blum (2005), pour un problème donné d'optimisation combinatoire, l'approche ACO vise à résoudre le problème en itérant les deux étapes suivantes :

- Les solutions candidates sont construites en utilisant un modèle de phéromones qui est une distribution paramétrée de probabilité au travers de l'espace de solutions
- les solutions candidates sont utilisées pour modifier les valeurs des phéromones d'une manière qui est supposée biaiser les recherches futures vers des solutions de haute qualité.

La mise à jour des phéromones vise à concentrer la recherche sur des régions de l'espace de recherche contenant des solutions de haute qualité. Plus particulièrement, le renforcement de composés de solution dépendant de la qualité de la solution est un ingrédient fondamental des algorithmes ACO. Il est, dès lors, assumer implicitement qu'une bonne solution est composée de bons éléments de solution. Identifier quels composants d'une solution contribuent à une bonne solution peut aider à les assembler en une solution meilleure.

L'algorithme AS initial a été amélioré pour obtenir un algorithme baptisé Ant Colony Optimization (ACO) (Dorigo 1999). Il consiste en trois activités qui sont planifiées et synchronisées.

```
while termination conditions not met do
  ScheduleActivities
    AntBasedSolutionConstruction()
    PheromoneUpdate()
    DaemonActions()          % optional
  end ScheduleActivities
endwhile
```

Figure 10 : Algorithme Ant System (AS)

AntBasedSolutionConstruction : Une fourmi construit une solution au problème en se déplaçant de nœuds en nœuds sur le graphe. Elles se déplacent en appliquant une règle de

décision locale stochastique qui utilise une valeur de phéromones et une valeur heuristique sur les composés de solutions et/ou les connections du graphe. En se déplaçant, la fourmi garde en mémoire la solution partielle qu'elle construit.

$s = \langle\rangle$
Determine $\mathcal{N}(s)$
while $\mathcal{N}(s) \neq \emptyset$ **do**
  $c \leftarrow$ ChooseFrom($\mathcal{N}(s)$)
  $s \leftarrow$ extend s by appending solution component c
  Determine $\mathcal{N}(s)$
end while

Figure 11 : Algorithme AntBasedSolutionConstruction

Les fourmis artificielles assemblent une solution comme une séquence de composés de solutions. L'ensemble fini de l'ensemble des composés de solution *C = {c1,…,cn}* est dérivé du problème d'optimisation à prendre en considération. Chaque construction de solution commence par une séquence vide $s = \langle\rangle$. Ensuite, la séquence est à chaque étape de la construction étendue en ajoutant un nouveau composé de solution à partir de l'ensemble des composés de solutions réalisables. Le choix d'un composé de solution à chaque itération est réalisé par le calcul d'une probabilité par rapport au modèle phéromone. Dans la plupart des algorithmes ACO, ces probabilités sont appelées probabilités de transition et sont définies comme suit :

$$\mathbf{p}(c_i \mid s) = \frac{[\tau_i]^\alpha \cdot [\eta(c_i)]^\beta}{\sum_{c_j \in \mathcal{N}(s)} [\tau_j]^\alpha \cdot [\eta(c_j)]^\beta}, \quad \forall c_i \in \mathcal{N}(s),$$

Où η est une fonction de balance qui dépendamment de la séquence courante assigne à chaque étape de la construction une valeur heuristique. Les exposants α et β sont des paramètres positifs qui sont des valeurs qui déterminent la relation entre l'information phéromone et l'information heuristique. Le bon paramétrage de ces deux paramètres est crucial. Une valeur trop haute de α et l'algorithme converge rapidement vers des solutions proches de solutions aléatoires. Au contraire une valeur trop faible transforme les fourmis en chercheuses aléatoires qui ne tiendraient compte que de l'information heuristique pour décider du prochain nœud à visiter.

PheromoneUpdate : En ajoutant des composés à la solution partielle, une fourmi peut mettre à jour la piste de phéromones. Une fois que la fourmi a construit une solution, elle peut, en

utilisant la mémoire, retracer à l'envers ce chemin et mettre à jour la piste de phéromones sur les composés et/ou les connections parcourues en fonction de la qualité de la solution construite.

Premièrement, l'évaporation des phéromones est le procédé par lequel l'intensité de la piste de phéromones sur les composants décroit avec le temps. L'évaporation est nécessaire pour éviter une trop rapide convergence de l'algorithme vers une région moins optimale. Ça implémente une sorte de procédure d'oubli des moins bonnes solutions en favorisant l'exploration de nouvelles régions de l'espace de recherche.

Deuxièmement, une ou plusieurs solutions de la solution courante et/ou des itérations précédentes sont utilisées pour augmenter les paramètres de la piste de phéromones sur les composés de solutions qui font partie de ces solutions.

$$\tau_i \leftarrow (1-\rho)\cdot\tau_i + \rho\cdot \sum_{\{s\in\mathcal{S}_{\text{upd}}|c_i\in s\}} w_s\cdot F(s),$$

Pour $i = 1, \ldots, n.$, S_{upd} représente l'ensemble des solutions qui sont utilisées pour la mise à jour. ρ est le paramètre donnant le taux d'évaporation. Dans bien des cas, S_{upd} est composé des solutions générées lors d'une itération S_{iter} et de la meilleure solution trouvée depuis le début de l'algorithme S_{bs}. En pratique, les algorithmes ACO utilisant une règle de mise à jour des phéromones basée sur la meilleure solution trouvée depuis le début et des mécanismes destinés à éviter une trop rapide convergence obtiennent les meilleurs résultats.

DaemonActions : elles peuvent être utilisées pour implémenter des actions qui ne peuvent être réalisées par les fourmis. Comme l'utilisation d'une procédure de recherche locale appliquée aux solutions construites par les fourmis ou la collecte d'informations qui peuvent être utilisées pour décider s'il faut ou non déposer des phéromones additionnelles pour biaiser le processus de recherche dans une perspective non-locale. Par exemple, on peut observer chaque solution construite par chaque fourmi et décider de déposer plus de phéromones sur les composants trouvés par la fourmi avec la meilleure solution. Cette procédure de dépôt de phéromone est dite hors ligne.

5.3. Variantes ACO

5.3.1. Elitist Ant System

L'idée de l'Elitist Ant System (EAS) est d'augmenter l'exploitation de la meilleure solution trouvée en introduisant un biais important sur les composés contenus dans la solution. Alors que les composés de solutions de l'itération courante sont mis à jour par une même valeur, la mise à jour des composés de la meilleure solution est supérieure à la valeur octroyée aux autres composés de l'itération.

5.3.2. Rank Based Ant System

Proposé par Bullnheimer, Hartl et Strauss (1999), cette amélioration de l'algorithme AS est sensiblement plus performante qu'EAS. La valeur des phéromones que chaque fourmi a la permission de déposer décroît en fonction du rang attribué à chaque fourmi pour la même itération. Chaque fourmi est classée par valeur de solution trouvée et un rang r est attribué correspondant à la position dans le classement. La mise à jour des phéromones est obtenue en permettant uniquement aux meilleures fourmis de déposer des phéromones en remplissant S_{upd} avec les meilleures $m - 1$ solutions où $m - 1 \leq n_a$ de S_{iter} et en ajoutant la meilleure solution trouvée S_{bs} à S_{upd}. Le poids des solutions devient $w_s = m - rs$ pour tous $s \epsilon S_{upd} \backslash \{sbs\}$ où r_s est le rang de la solution s. A la fin, le poids w_{sbs} de la solution s_{bs} devient la valeur de m. Cela revient à dire que, à chaque itération, la meilleur solution a une plus grande influence sur la mise à jour des phéromones en même temps qu'une sélection de solutions influence la mise à jour en fonction de leur rang.

5.3.3. MAX-MIN Ant System

L'une des variantes les plus performantes d'ACO est le Max-Min Ant System (MMAS) proposé par Stützle (1999) et Stützle et Hoos (1997,2000). Premièrement, la mise à jour des phéromones est uniquement réalisée sur les arcs utilisés par la meilleure solution de l'itération *IB-update* et la meilleure solution trouvée depuis le début de l'algorithme *BS-update*. Au début de l'algorithme *l'IB-update* est privilégié au début de l'algorithme et graduellement la fréquence d'utilisation de *BS-update* est augmentée.

Deuxièmement, les valeurs des phéromones sont restreintes à un intervalle $[\tau_{min}, \tau_{max}]$. L'algorithme utilise une borne explicite $\tau_{min} > 0$ qui empêche que la probabilité de construire une solution descende en dessous d'une certaine valeur supérieure à 0. Ceci implique que la

chance de trouver un optimum global ne s'évapore jamais lors du déroulement de l'algorithme.

Troisièmement, on utilise une borne supérieure $F(S_{bs})/\rho$ pour la valeur des phéromones. Cette valeur est mise à jour chaque fois que l'algorithme trouve une meilleure solution parmi toutes les autres. En cas de stagnation, MMAS peut également réinitialiser la meilleure solution trouvée pour éviter de se bloquer sur une même solution.

5.3.4. Ant Colony System

Ant Colony System (ACS) (Dorigo et Gambardella, 1997) diffère de l'algorithme AS original sur bien plus d'aspects que la mise à jour des phéromones.

Premièrement, ACS utilise une règle différente pour décider vers quel nœud suivant une fourmi va se déplacer. Cette règle est appelée la règle pseudo aléatoire proportionnelle. Ce mécanisme a pour but d'augmenter la diversification de la recherche. Au lieu de n'utiliser que la règle de probabilité générale d'ACO pour décider du prochain composé de solution à ajouter, une fourmi artificielle choisit, avec une probabilité q_0, le composé de solution qui maximise $[\tau_i]^{\alpha} \cdot [\eta(c_i)]^{\beta}$ ou, avec une probabilité $1 - q_0$, une construction de probabilité suivant la règle générale.

Deuxièmement ACS utilise une règle de mise à jour de la valeur des phéromones en appliquant un mécanisme d'évaporation uniquement sur les arcs qui appartiennent à la meilleure solution trouvée depuis le début. Cette règle favorise l'émergence d'autres solutions que la meilleure solution trouvée depuis le début.

Troisièmement, après chaque étape de construction, une mise à jour additionnelle est appliquée sur les valeurs de phéromones τi pour les composés de solution correspondant à la solution en cours de construction.

$$\tau_i \leftarrow (1 - \xi) \cdot \tau_i + \xi \cdot \tau_0,$$

Où τ_0 est une constante positive pour que $F_{min} \geq \tau_0 \geq c$, $F_{min} \leftarrow \{F(s) \mid s \in S\}$ et c est la valeur initiale de phéromone. En pratique, l'effet de cette mise à jour est de faire décroître la valeur de phéromones sur les arcs visités les rendant ainsi moins désirable pour les fourmis suivantes. Cela implique une augmentation de l'exploration de l'espace de recherche à chaque itération.

5.3.5. Hyper-Cube Framework

L'Hyper-cube Framework a été proposé par Blum, Roli et Dorigo (2001). En pratique, il gère la mise à l'échelle des valeurs des phéromones pour être limitées dans un intervalle [0,1]. Il s'agit dès lors d'un hyper-cube puisque si un problème est représenté par des variables de décision binaires alors la solution est un coin d'un hyper-cube de *n* dimensions. L'ensemble des solutions à mettre à jour peut être composé de n'importe quelle manière ce qui est en fait plutôt qu'une variante d'ACO, un cadre pour l'implémentation de variantes ACO comme AS, ACS ou MMAS. La mise à jour des phéromones peut dès lors être interprétée comme un mouvement dans un hyper-cube.

5.4. Conclusion du chapitre

Dans ce chapitre, nous avons présenté les variantes les plus courantes d'ACO et les principales caractéristiques qui les différencient selon Blum et Roli (2003). Un comparatif des performances de résolution a été proposé par Dorigo et Stützle (2004) mettant en exergue que les variantes MMAS et ACS sont les plus performantes. ACS est très agressif et trouve de bonnes solutions dans un temps très court alors que MMAS est moins efficace qu'AS dans les premières itérations mais éventuellement produit de meilleures solutions sur le long terme.

Chapitre 6

6. Application ACO à l'étude de cas

6.1. Problématique

Trouver des tournées efficientes pour des véhicules est un problème important de la logistique. Quand une entreprise est capable de réduire la longueur des tournées ou est capable de réduire le nombre de véhicules, elle est capable de fournir un meilleur service à ses clients, d'opérer de manière plus efficiente et d'éventuellement augmenter ses parts de marché.

Un VRP typique inclut simultanément de déterminer une ou des tournées pour un ou plusieurs véhicules à partir d'un dépôt central vers un certain nombre de clients et de revenir au dépôt sans dépasser les contraintes de capacité de chaque véhicule. Ce problème est d'une grande importance économique en raison du temps et du coût alloué à une flotte de véhicules pour transporter des produits à un ensemble de clients géographiquement dispersés.

Le problème implique de trouver le coût minimum pour une combinaison de tournées pour un nombre de véhicules pour faciliter les livraisons vers les clients. Etant donné que le coût est associé à la distance, une entreprise cherche à trouver la distance minimum parcourue par un nombre de véhicules pour satisfaire la demande des clients. De ce fait, l'entreprise essaye de minimiser ses coûts tout en augmentant ou, au moins, en maintenant un certain niveau de service au client.

Dans le cadre de notre étude de cas, l'entreprise est confrontée à une forte diminution des colis à transporter. Sur l'année A + 2, les prévisions ne montrent plus que des livraisons de produits « Cool » soit 2746 colis de demande pour servir 127 clients. Cela correspond selon nos estimations à 1879 livraisons sur un an. Le changement conséquent entre les différentes années implique une évaluation par l'entreprise de son domaine d'activités qui tend vers la distribution de produits achetés à des partenaires plutôt qu'à la fabrication de ces produits avec ses propres matières premières.

Notre question de recherche est : **« Faut-il externaliser ou internaliser la distribution des produits pour la Belgique ? »**

Nous pensons que la question mérite d'être posée. En effet, un tel bouleversement peut présenter des changements dans le réseau de distribution comme, par exemple, de nouvelles concentrations de clients où la demande est plus forte.

En internalisant le transport, une entreprise ajoute les coûts fixes et variables de la gestion d'une flotte de véhicules. Minimiser ces coûts est un critère important d'efficacité. Dès lors, assurer les livraisons à l'aide d'un seul véhicule est le minimum en termes de coûts. Nous formulons l'hypothèse suivante :

> **Hypothèse 3** : Il est possible de réaliser les tournées de l'année avec un véhicule de transport tout en satisfaisant toute la demande des clients.

Bien que la décision d'internaliser le transport soit une décision stratégique qui s'évalue sur le long terme. Une comparaison des coûts entre les deux options sur le court terme est intéressante pour déterminer la rentabilité de l'opération. Ceci nous amène à l'hypothèse :

> **Hypothèse 4** : « La distribution des produits en internalisant le transport est moins onéreux que le groupage par un transporteur externe. »

6.2. Collecte des données

L'étude est basée sur les données collectées pour une année de référence où tous les produits de l'entreprise ont été distribués normalement. Nous souhaitons dans le cas qui nous préoccupe, analyser la distribution des produits pour l'année A + 2 où seuls les produits « cool » subsistent.

Nous avons supprimé des données de l'année de référence tous les colis d'albumines et d'immunoglobulines qui ne seront plus distribués.

Les données de distance générées par géolocalisation sont conservées et sont calculées au départ du dépôt DEP1 qui correspond au dépôt de la situation de distribution actuelle. Nous avions déterminés dans le chapitre 2 que ce dépôt est celui qui minimise le coût de distribution.

6.3. Création de base de données

Nous avons évoqué dans le chapitre 2, la création des données manquantes nécessaire à l'analyse. Les données créées comme les distances sont d'application pour cette étude mais de nouvelles données sont nécessaires.

6.3.1. Le coût de transport au kilomètre

Nous voulons dans cette étude pouvoir comparer le coût total de transport aussi bien dans le cas où le transport est externalisé que le cas où, au contraire le transport est internalisé.

Nous devons incorporer les coûts de gestion d'un véhicule et d'un chauffeur-livreur pour assurer les tournées. Nous nous sommes aidés d'un simulateur de coût de revient [2] pour facilement calculer le coût au km en incorporant les coûts fixes : financement du véhicule, assurances, taxes et les coûts variables : carburant, entretien. Le coût du personnel de conduite est également simulé et incorporé.

6.3.2. La capacité du véhicule

Les capacités d'un véhicule bien qu'elles puissent être importantes ne sont pas illimitées. Nous allons utiliser un véhicule pour lequel une certaine capacité doit être définie. Dans notre étude de cas, le nombre de colis est relativement limité et ne constitue pas une contrainte pour la majorité des véhicules de livraison que l'on peut trouver sur le marché. Le volume à transporter sera en général inférieur à la capacité chargeable du véhicule.

Nous nous sommes dès lors basés sur le temps maximum de travail alloué au chauffeur. Nous avons pris une valeur de 8 heures maximum par jour. En considérant la vitesse de déplacement à 50km/h de moyenne entre chaque localisation, chargement et déchargement compris, un véhicule peut en théorie visiter 8 localisations. Il s'agit de la capacité que nous avons imposée à notre véhicule.

6.3.3. Le temps de travail

Le temps de travail est le temps nécessaire à un véhicule pour effectuer une tournée en partant et en revenant au dépôt tout en visitant tous les clients en fonction de la capacité du véhicule ou du nombre restant de clients à servir.

[2] http://www.cnr.fr/Outils-simulation2/Simulateurs

Nous avons choisi de déterminer ce temps en divisant le nombre de kilomètres total de la tournée par une vitesse moyenne estimée de 50 km/h. Les temps de déplacement et les arrêts pour les livraisons sont compris.

6.3.4. Le niveau de service

Si nous tenons compte du niveau de service actuellement rendu par l'entreprise. Une commande passée le jour *J* est livrée le jour *J + 1*. Optimiser une tournée sur base journalière revient à solutionner un TSP. Il existe des méthodes exactes capable de résoudre un problème de TSP de moins de 25 instances et ce, endéans un temps raisonnable.

Si la demande est stochastique, il est fortement probable de pouvoir résoudre le problème de manière efficace mais pas de manière efficiente. En effet, il est possible que les localisations soient très éloignées les unes des autres.

Un algorithme comme ACO capable d'explorer un grand nombre de possibilités dans un vaste espace de recherche est moins pertinent à utiliser. Nous avons donc regroupé les commandes journalières par semaine pour favoriser l'optimisation en regroupant des localisations proches. Le niveau de service devient une livraison endéans les 5 jours.

6.4.Modèle ACO

Nous utilisons un modèle ACO proposé par Bell et McMullen (2004). La génération des solutions ACO a été faite en langage Mosel (Catanzaro, 2018) sur un Dell i5-6300U CPU @ 2.40 Ghz. Le modèle est résolu à l'aide de Fico Xpress Software.

6.4.1. Construction de tournée

Une fourmi individuelle simule un véhicule et sa tournée est construite en sélectionnant chaque client jusqu'à ce que tous les clients aient été visités. Initialement chaque fourmi débute au dépôt et l'ensemble des clients de la tournée est vide. La fourmi choisit un prochain client à visiter sur base de la liste des localisations possibles et la capacité du véhicule est mis à jour avant qu'un nouveau client soit sélectionné. La fourmi revient au dépôt quand la capacité maximale du véhicule est atteinte ou si tous les clients ont été visités. La distance totale est enregistrée comme valeur de la fonction objective pour la tournée totale de la fourmi. L'algorithme construit une tournée complète pour la première fourmi avant qu'une prochaine fourmi commence son propre tour. La procédure se répète jusqu'à ce qu'un nombre prédéterminé de *m* fourmis construise chacune une tournée réalisable.

Chaque fourmi construit une tournée qui visite chaque client. Pour sélectionner le prochain client j, la fourmi utilise la formule suivante :

$$j = \underset{\text{otherwise } S}{\arg\max}\{(\tau_{iu})(\eta_{iu})^{\beta}\} \quad \text{for } u \notin M_k, \quad \text{if } q \leq q_0, \tag{1}$$

Où τ_{iu} est la valeur de phéromones sur l'arête entre la localisation courante i et les prochaines localisations possibles u. La valeur η_{iu} est définie comme l'inverse de la distance entre deux localisations et le paramètre β correspond à l'importance de la distance en fonction de la valeur des phéromones associée. Les localisations déjà visitées par une fourmi sont enregistrées dans une mémoire M_k et ne sont plus prises en compte pour une sélection. La valeur q est une variable uniforme aléatoire [0,1] et la valeur q_0 est un paramètre. A chaque décision de sélection, une fourmi choisit l'arc avec la plus grande valeur provenant de l'équation (1) sauf si $q \geq q_0$. Dans ce cas, une fourmi sélectionne une variable aléatoire (S) pour être le prochain client à visiter sur de la distribution de probabilité de p_{ij} qui favorise les trajets les plus courts avec une grande valeur de phéromones.

$$p_{ij} = \frac{(\tau_{ij})(\eta_{ij})^{\beta}}{\sum_{u \notin M_k}(\tau_{iu})(\eta_{iu})^{\beta}} \quad \text{if } j \notin M_k \text{ otherwise } 0 \tag{2}$$

La mise en œuvre des formules (1) et (2) permet à une fourmi, soit de suivre le chemin le plus favorable déjà établi, soit de sélectionner un autre chemin aléatoirement basé sur la distribution de probabilité construite par la distance et l'accumulation de phéromones. Si la contrainte de capacité du véhicule est atteinte, la fourmi retourne au dépôt sans sélectionner un nouveau client. Ce processus continue jusqu'à ce que chaque client soit visité et que la tournée soit finie.

6.4.2. Mise à jour des phéromones

La mise à jour des phéromones inclut une mise à jour locale des arêtes utilisées après que les solutions individuelles ont été générées et une mise à jour globale qui renforce les arêtes de la meilleure solution trouvée comme dans la variante ACS.

Premièrement, une mise à jour est générée pour réduire la valeur des phéromones sur tous les arcs visités pour simuler l'évaporation des phéromones, et ce afin d'assurer qu'aucun arc ne devienne trop prédominant. Ceci est réalisé par l'équation suivante :

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + (\alpha)\tau_0$$

Où α est un paramètre qui contrôle la vitesse d'évaporation et T_0 est égal à un niveau prédéterminé de phéromones sur tous les arcs.

Deuxièmement, une fois que les *m* fourmis ont construit une solution, une mise à jour globale est réalisée en renforçant les arcs de la meilleure solution trouvée

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + \alpha(L)^{-1}$$

Cette mise à jour encourage l'utilisation des chemins les plus courts et augmente la probabilité que les fourmis utilisent les arcs de ces meilleures solutions. Le processus est répété jusqu'à ce qu'une condition d'arrêt soit rencontrée et la meilleure solution est présentée comme une bonne estimation de la solution optimale du problème.

6.4.3. Paramètres du modèle

Pour toutes les solutions, les paramètres suivants ont été utilisés :

m = 10	nombre de fourmis construisant une solution à chaque itération
Cap = 8	Capacité maximale de livraisons du véhicule
N = n	Nombre de clients à livrer en fonction du problème
β = 2	Importance de l'information heuristique
Dépôt = 1	Nombre de dépôts ouverts
Véhicule = 1	Nombre de véhicules utilisés
PheInit = 0.5	Valeur initiale des phéromones
PheR = 0.02	Valeur de renforcement des phéromones
PheE = 0.01	Valeur d'évaporation des phéromones
Time = 30s	Temps maximum de calcul en secondes avant proposition de solution

6.5.Résultats

Pour une facilité de lecture des résultats, nous avons regroupés les semaines en fonction du nombre de jours qu'elles comptaient.

Temps de travail

			Temps disponible	Temps utilisé par ACO
	Jours par semaine	Nbre de semaine	(heures)	(heures)
	3	1	24	14
	4	9	288	195
	5	42	1680	960
Total	249	52	1992	1169

Table 8 : Comparaison du temps total disponible et du temps utilisé par les tournées optimisées par ACO

L'ensemble des tournées optimisées par l'algorithme ACO utilise 69% du temps total disponible pour les livraisons.

Coût total

			Transport Externe	Transport Interne
	Jours par semaine	Nbre de semaine	(€)	(€)
	3	1	635	912
	4	9	6.729	11.834
	5	42	35.218	61.770
Total	249	52	42.582	74.516

Table 9 : Comparaison entre le coût de la solution de transport de groupage par un transporteur externe et la solution des tournées optimisées par ACO en interne.

La solution en transport interne est 75% plus onéreuse que la solution en transport externe.

6.6.Limitations

Les résultats obtenus sont cohérents mais il est difficile de prendre position par rapport à la robustesse du modèle appliqué.

Le modèle utilisé a démontré sa force en minimisant la distance à parcourir ainsi que par corrélation le coût du transport. Cependant la littérature montre que les variantes ACS et MMAS sont parmi les plus performantes (Blum et Roli, 2003). Particulièrement, ACS avec son comportement agressif dans la construction des solutions qui exploite au mieux l'expérience accumulée par les fourmis précédentes. Nous ne pouvons pas garantir que la solution fournie soit effectivement la plus performante.

Nous avons attribué une capacité de 8 livraisons maximum et une vitesse moyenne de 50km/h. Le modèle est fortement contraint par ces paramètres. Nous pouvons justifier ce choix par le faible nombre d'instances considérées mais une augmentation du nombre de livraison dans le futur nécessiterait un ajustement. En outre, utiliser une contrainte de temps maximum plutôt qu'un nombre de livraisons permettrait de favoriser un plus grand nombre de solutions où les localisations sont proches les unes des autres et par la même occasion améliorerait le résultat final.

Nous avons estimé les livraisons et le nombre de colis en évinçant les produits qui ne seront plus livrés et en conservant les commandes des produits restants. Il est néanmoins difficile de prédire le comportement des clients et le profil de commande qu'ils adopteront. Nous avons vu que c'est le profil de commande qui impacte le coût de transport réalisé par le transporteur externe. Si les clients ont tendance à commander souvent de petites quantités, le coût global peut augmenter significativement.

Le nombre de fourmis déployées a un effet sur la qualité de la solution trouvée. A chaque itération, une fourmi construit une solution. Le nombre de solutions explorées dépend donc non seulement du nombre de fourmis mais aussi du temps alloué avant que la condition d'arrêt soit rencontrée. Dans notre cas, nous avons alloué 10 fourmis et 30 secondes de calcul mais sans garantie que cette configuration donne le meilleur résultat.

Nous avons décidé de modifier le niveau de service actuel qui consiste à livrer le lendemain une commande réceptionnée le jour même par une livraison endéans les 5 jours. Bien que cela se justifie opérationnellement par notre volonté d'allouer au modèle un plus grand nombre de possibilités d'optimisation pour obtenir une solution performante qui améliore l'utilisation des ressources allouées au transport, un changement de service nécessite une évaluation du point de vue marketing et médical pour évaluer les avantages et inconvénients d'un telle décision sur le sentiment de satisfaction des clients.

6.7. Conclusion du chapitre

Nous avons présenté dans ce chapitre l'application d'un modèle ACO à un problème défini de VRP. Nous avions comme objectif d'optimiser la distribution des produits de l'entreprise en comparant la solution de transport actuelle avec une solution de transport internalisée. La réponse à notre troisième hypothèse est positive. Nous pouvons raisonnablement penser qu'un seul véhicule puisse satisfaire toute la demande étant donné qu'il n'est utilisé qu'à 69% de sa

capacité de temps. La réponse à notre quatrième hypothèse est négative. La solution de transport en interne trouvée est nettement moins compétitive que la solution de transport en externe.

Sur base de ces résultats, nous pouvons répondre par la négative à la question d'internaliser le transport. Néanmoins, une augmentation du nombre de colis à livrer par l'introduction de nouveaux produits, une augmentation des ventes couplé à un ajustement des paramètres du modèle permettrait une nouvelle évaluation du seuil de rentabilité entre les deux solutions.

Conclusion et directions pour recherches futures

Dans ce mémoire, Nous avons investigué le cas d'une entreprise confrontée à des changements pour la distribution de ses produits vers ses clients. Nous avons adressé le problème en deux parties.

En première partie, nous avons présenté certaines propriétés des techniques de Supply Chain Network Design pour résoudre le problème de localisation optimale d'un dépôt. La mise en application de ces modèles à notre étude de cas, nous a permis de valider la localisation du dépôt actuel de l'entreprise comme optimale.

En seconde partie, Nous avons abordé le problème du transport vers les clients ce qui correspond à résoudre un Vehicle Routing Problem (VRP). Après avoir présenté les propriétés du problème et décrit sa complexité à être solutionné par des méthodes exactes classiques, nous avons vu les différentes méthodes de la classe des métaheuristiques qui sont des stratégies permettant de trouver une solution acceptable au problème dans un temps raisonnable. Nous nous sommes plus particulièrement intéressés à une méthode appelée Ant Colony Optimization (ACO), qui s'inspire du comportement collaboratif des fourmis à l'aide de phéromones pour trouver le chemin le plus court vers les sources de nourriture. Nous avons implémenté un modèle ACO qui a démontré sa capacité à optimiser les tournées de distribution de notre étude de cas tout en valorisant le coût de la solution trouvée.

Les résultats obtenus sont encourageants et nous permettent de faire les recommandations suivantes pour l'entreprise :

- Conserver la situation actuelle du dépôt
- Conserver la solution externe de transport

Nous sommes bien conscients des limites de nos modèles et pour une recherche future, nous suggérerions les aspects suivants. Premièrement, les différente variantes d'ACO ont des résultats différents et se comporte plus ou moins efficacement en fonction des caractéristiques du problème. Etablir un comparatif des résultats des différentes versions permettrait de sélectionner la plus performante pour notre cas. Deuxièmement, appliquer une recherche locale par une méthode exacte sur les solutions trouvées par métaheuristique donnerait la

possibilité d’encore affiner les résultats. Finalement, un ajustement systématique des paramètres du modèle permettrait de valider la meilleure configuration.

Bibliographie

Afifi, S., Dang, D., & Moukrim, A. (2013). A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. Paper presented at the , 7997 259-265. doi:10.1007/978-3-642-44973-4_27

Bar-Yam, Y. (1997). Dynamics of complex systems (Vol. 213). Reading, MA: Addison-Wesley.

Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics, 18*(1), 41-48. doi:10.1016/j.aei.2004.07.001

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews, 2*(4), 353-373. doi:10.1016/j.plrev.2005.10.001

Blum, C., Roli, A., and Dorigo, M. 2001. HC–ACO: The hyper-cube framework for ant colony optimization. In Proceedings of MIC'2001—Meta–heuristics International Conference. Vol. 2. Porto, Portugal, 399–403.

Blum, C., & Roli, A. (2003, September). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM computing surveys, 35(3), 268-308. doi:10.1145/937503.937505

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 89, 319-328.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, 45(1), 41-51. doi:10.1007/BF00940812

Coello Coello, C. A. 2000. An updated survey of GA-based multiobjective optimization techniques. ACM Comput. Surv. 32, 2, 109–143.

Devaney, R. L. (1989). An introduction to chaotic dynamical systems / robert L. devaney (2nd ed.). Redwood City: Addison-Wesley.

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. Theoretical Computer Science, 344(2-3), 243-278. doi:10.1016/j.tcs.2005.05.020

Dorigo, M. and Di Caro, G. 1999. The ant colony optimization meta-heuristic. In New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw-Hill, 11–32.

Dorigo, M. and Stützle, T. 2002. The ant colony optimization metaheuristic: Algorithms, applications and advances. In Handbook of Metaheuristics, F. Glover and G. Kochenberger, Eds. International Series in Operations Research & Management Science, vol. 57. Kluwer Academic Publishers, Norwell, MA, 251–285.

Dorigo, M., and Stützle, T. (2004). Ant colony Optimization. Cambridge: MIT Press.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. Biosystems, 43(2), 73-81. doi:10.1016/S0303-2647(97)01708-5

Eiben, A & Raué, Paul-Erik & Ruttkay, Zsófia. (1994). Genetic algorithms with multi-parent recombination. 78-87. 10.1007/3-540-58484-6_252.

Eiben, A. E. and Ruttkay, Z. 1997. Constraint satisfaction problems. In Handbook of Evolutionary Computation, T. B¨ack, D. Fogel, and M. Michalewicz, Eds. Institute of Physics Publishing Ltd, Bristol, UK.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization, 6(2), 109-133. doi:10.1007/BF01096763

Gendreau, M., Laporte, G., and Potvin, J.-Y. 2001. Metaheuristics for the vehicle routing problem. In The Vehicle Routing Problem, P. Toth and D. Vigo, Eds. SIAM Series on Discrete Mathematics and Applications, vol. 9. 129–154.

Gendreau, M., & Potvin, J. (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research, 140*(1), 189-213. doi:10.1007/s10479-005-3971-7

Glover, F., & Laguna, M. (1997). Tabu search. Boston: Kluwer Academic Publishers

Glover, F. (1990). tabu search—part ii. ORSA Journal on Computing, 2(1), 4-32. doi:10.1287/ijoc.2.1.4

Goldberg, D. E. and Richardson, J. 1987. Genetic algorithms with sharing for multimodal function optimization. In Genetic Algorithms and their Applications, J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, Hillsdale, NJ, 41–49.

Hansen P., Mladenović N. (1999) An Introduction to Variable Neighborhood Search. In: Voß S., Martello S., Osman I.H., Roucairol C. (eds) Meta-Heuristics. Springer, Boston, MA

Johnson, David & A. McGeoch, Lyle. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. Local Search in Combinatorial Optimization. 1.

Kilby P., Prosser P., Shaw P. (1999) Guided Local Search for the Vehicle Routing Problem with Time Windows. In: Voß S., Martello S., Osman I.H., Roucairol C. (eds) Meta-Heuristics. Springer, Boston, MA

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220(4598), 671-680. doi:10.1126/science.220.4598.671

Kumar, S. N., & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management, 4*(3), 66-74. doi:10.4236/iim.2012.43010

Lourenco, H. R., Martin, O. C., & Stutzle, T. (2001). Iterated local search.

Moscato, P. 1999. Memetic algorithms: A short introduction. In New Ideas in Optimization, F. G. D. Corne and M. Dorigo, Eds. McGraw-Hill.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research, 63*(5), 511-623. doi:10.1007/BF02125421

Solomon, M. “Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints,” Operations Research, Vol. 35, No. 2, 1987, pp. 254-265.

Stützle, T. (1998). Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, *20*.

Stützle, T. and Hoos, H. H. 2000. MAX-MIN Ant System. Fut. Gen. Comput. Syst. 16, 8, 889–914.

Voudouris C., Tsang E.P.K. (2003) Guided Local Search. In: Glover F., Kochenberger G.A. (eds) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol 57. Springer, Boston, MA

Watson, M. (2013). Supply chain network design: Applying optimization and analytics to the global supply chain. Upper Saddle River, N.J: FT Press.