الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire
Abd elhafid Boussouf Mila

Institut des sciences et de la technologie Département de Mathématiques et Informatiques

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication (STIC)

Conception et Réalisation d'un Outil d'Aide à la Décision pour une Agence de Transport à la Demande

Préparé par : - Benamor samira
- Chahdane nour el houda

Soutenue devant le jury :

Encadré par : Zekiouk Mounira M.A.A
Président : Deffas Zineb M.A.A
Examinateur : Bouchekouf Asma M.A.A

Année universitaire : 2015/2016

Remerciement

Nous ne serions pas arrivées jusque là si « ALLAH » le tout puissant ne nous avait donné la force et le courage pour continuer et ne nous avait aidé et éclairer le chemin.

Nous tenons à lui exprimer toute nos reconnaissances et gratitudes.

Nous remercions vivement Madame « Zekiouk Mounira », notre encadreur, de nous avoir pris sous son aile bienveillante en nous encadrant dans ce travail, et de nous avoir fait profités de sa grande expérience et expertise.

Nous tenons à lui exprimer nos grandes reconnaissances pour son orientation, son aide et sa gentillesse.

Qu'elle soit assurée de notre très grand respect et appréciation.

Nos vifs remerciements iront naturellement vers l'ensemble des membres du jury; madame « Deffas Zineb » et madame « Bouchekouf Asma », qui nous on fait l'honneur d' accepté de juger notre travail.

Nous tenons à remercier et à saluer L'ensemble de nos professeurs de Master 2 STIC,

un grand merci à tous nos amis pour leur aide et leur soutien qu'ils nous ont accordé, particulièrement « Ishak Boulfoul » pour leur encouragements et surtout pour leur

Assistance dans la réalisation de ce projet

Samira & Nour el houda

Merçi beaucoup

Dédicace

A...

La lumière de mes yeux Mon père miséricorde de Dieu et ma mère, qui ont été toujours là pour moi, me guider, m'inspirer et qui ont me donné un magnifique modèle de labeur et de persévérance.

Mon chère encadreur « Zekiouk Mounira »,

toute ma connaissance et mes amours…

mon adorable frère « Issam Edine »,

et ma chère sœur « zineb »,

Mes grands père « Mouloud », « Saadi »,

et Mes grande mère « cherifa », « merieme », et toute ma famille, Mon fiancé Ahmed

Mes très chères amies « Loubna »,

« Wahida » et « Mouna »,

Tous mes Collègues de travail particulièrement Monsieur «Allioua Ahcene » et à Tous mes Collègues de STIC 2

Promotion 2015/2016,

Tous ceux et celles qui me sont chères…

Je dédie ce mémoire.

Benamor Samira

Dédicace

À...

La lumière de mes yeux mes chers parents, qui ont été toujours là pour moi, me guider, m'inspirer et qui ont me donné un magnifique modèle de labeur et de persévérance.

mon chère encadreur « Zekiouk Mounira »,

toute ma connaissance et mes amours…

mon adorable frère « Youcef »

mes très chèrs amies« Marwa », « Imane », « Zina » , les amis et le directeur du travaille au ADE Oued elndja

et à Tous mes Collègues de STIC 2 Promotion 2015/2016.

Mon fiancé Mohammed avec qui j'espère vivre toute ma vie et qui donné du courage durant mes études

tous ceux et celles qui me sont chères…

Je dédie ce mémoire.

Chahdane Nour El Houda

Résumé

Ce mémoire expose notre travail de fin d'étude visant à réaliser un outil d'aide à la décision pour assister les gestionnaires (opérateurs) d'une agence de transport à la demande dans les tâches décisionnelles.

Notre outil d'aide à la décision assure deux fonctionnalités, à savoir la génération des tournées et la simulation de cas de coopération.

La réalisation des fonctionnalités de l'outil est assurée par des méta-heuristiques (ILS) pour la génération des tournées et les systèmes multi-agents(SMA) pour simuler la notion de coopération entre agences.

Concernant l'implémentation, nous avons utilisé JAVA comme langage de programmation, NetBeans comme IDE et JADE comme plateforme des SMA.

Les mots clés: transport a la demande, optimisation, coopération, outil d'aide a la décision, méta-heuristique, les système multi-agents

Abstract

This report exposes our work of the end of study to realize a decision-making tool to assist the administrators (operators) of an agency of demand responsive transport in the decision-making tasks.

Our decision-making tool assures two features, worth knowing the generation of the tours and the simulation of case of cooperation.

The realization of the features of the tool is assured by meta heuristics (ILS) for the generation of the tours and the multi-agents systems (SMA) to feign the notion of cooperation between agencies.

Concerning the implementation, we used JAVA as programming language, NetBeans as IDE and JADE as platform of the SMA.

Keywords: Transport on demand, decision-making tool, Meta-heuristic, multi agent system, cooperation, optimization.

Table de matière

Listes des figures

Introduction générale

De nos jours, le problème de transport occupe une place importante dans la vie économique des sociétés modernes. Les individus cherchent des moyens de transport plus souples, flexibles et proches de leurs attentes.

Le Transport à la demande permet de répondre à ces attentes. En effet, il est considéré comme un mode de transport collectif, individualisé et activé à la demande.
Dans ce mode de transport, les clients (voyageurs) envoient des demandes de transport à un opérateur comprenant le lieu de ramassage (origine), la destination finale et l'heure de passage désirée. Le problème de TAD consiste donc à élaborer des tournées pour répondre aux demandes de transport en minimisant les coûts engendrés par ces tournées (distance parcourue, temps de service, nombre de véhicules utilisés) et en fournissant une bonne qualité de service aux voyageurs (temps de voyage, nombre de stations visitées, etc..).

Comme la majorité des secteurs économiques le transport à la demande est basé sur des actions décisionnelles quotidiennes afin d'assurer une planification optimale des tâches opérationnelles. Alors, l'utilisation des différents outils d'aide à la décision est primordiale pour atteindre des bénéfices remarquables. Dans ce contexte le marché actuel propose des centaines de produits avec des fonctionnalités variées.

Dans le cadre de notre travail nous focalisons nos efforts sur la modélisation et la réalisation d'un outil d'aide à la décision qui assure deux fonctionnalités : Optimiseur et Simulateur des coopérations entre plusieurs agences de transport à la demande. Ces deux rôles sont assurés par un couplement entre des heuristiques d'optimisation (la recherche locale itérative (ILS)) et le paradigme multi-agents qui assure une simulation originale et naturelle de la réalité étudiée.

Précisément notre objectif concerne la proposition d'un moyen pour assister les gestionnaires d'une agence de transport à la demande. Alors notre outil d'aide à la décision est destiné à un usage collaboratif avec d'autres outils d'aide à la décision et non pas à la gestion globale d'une agence de transport à la demande.

Le reste du mémoire est organisé en 4 chapitres comme suit :

• **Chapitre 1 : Le Transport À la Demande.**

Ce premier chapitre est dédié à la présentation des différentes notions liées au domaine de transport à la demande.

• **Chapitre 2 : l'Aide à la Décision dans le Domaine de Transport À la Demande**

Dans ce chapitre nous introduisons dans un premier temps des généralités liées au domaine d'aide à la décision. Nous déterminons par la suite l'intérêt de l'aide à la décision et des outils d'aide à la décision dans le domaine de TAD.

• **Chapitre 3 : Outil d'Aide À la Décision pour une Agence de TAD.**

Le troisième chapitre est dédié à la modélisation de notre propre outil. Dans un premier temps nous présentons les limites fonctionnelles de l'outil. Ensuite, nous présentons en détailles les fonctionnalités de ce dernier.

• **Chapitre 4: Implémentation et Tests.**

Le dernier chapitre est consacré à la réalisation et la mise en œuvre de notre outil, nous allons présenter les outils de développement adoptés ; le langage de programmation Java, ainsi que l'environnement utilisé Netbeans, l'Environnement pour la programmation multi-agent JADE et enfin nous présentons les principales interfaces et fenêtres de l'outil.

Chapitre 01

Le Transport À la Demande

1. Introduction

Lors de ces dernières années, l'offre de transport public n'a cesse de se développer et de s'améliorer, et avec elle les demandes et les exigences des usages, rendant ces moyens de transport moins compétitifs surtout dans les milieux ruraux et périurbains.

L'utilisation de la voiture individuel reste assez répondu dans ces zones de part sa disponibilité et de son efficacité malgré les quelques inconvénients qu'elle possède en l'occurrence le coût d'utilisation (prix de l'essence) et les problèmes environnementaux qu'elle causse.

L'une des solutions qui peut résoudre ce problème et permet de profiter des avantages des deux modes de transport (en commun et individuel) et de réduire leurs inconvénients est le partage des voitures privées. Ce service de partage de voitures privées est traité dans la littérature sous le nom "problème de **T**ransport **À** la **D**emande (TAD)".

Ce problème fait partie de la famille des problèmes de tournées de véhicules présentés dans le présent chapitre.

2. Le problème de tournée des véhicules (VRP) :

2.1. Définition:

le VRP (Véhicule Routing Probelm) consiste à déterminer, en minimisant le coût, un ensemble de tournées, pour un nombre limité de véhicules, commençant et finissant à un dépôt, de telle façon que chaque client soit visité exactement une fois par un véhicule, et que la somme des demandes provenant des clients sur une tournée ne dépasse pas la capacité du véhicule qui dessert la route [3].

La figure 1 schématise un ensemble de quatre tournées qui commencent et terminent à un dépôt (point central) où chaque véhicule passe par plusieurs station distingues.

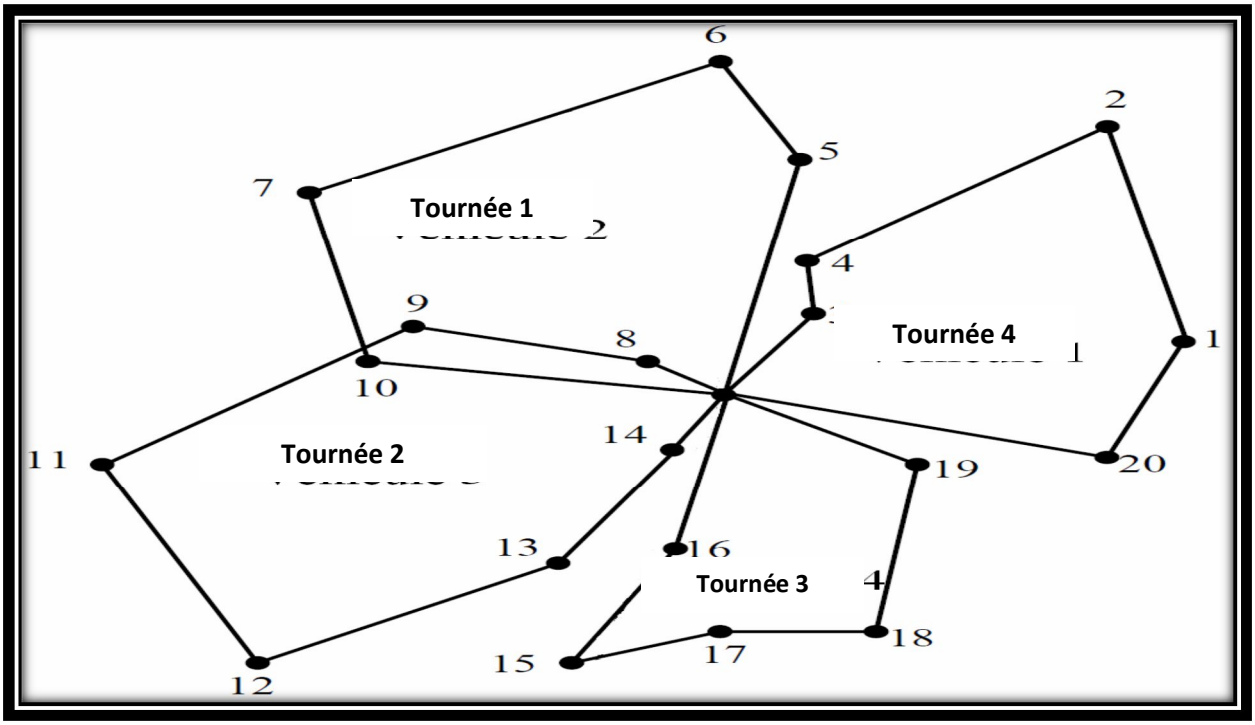

Figure 1.1 Un exemple de problème de VRP

2.2. Les variantes du VRP

La variation des paramètres du VRP, la suppression et/ou l'ajout ou bien la combinaison de contraintes du VRP classique permettent de définir un ensemble de variantes du VRP.

Dans ce qui suit, nous présentons une répartition des variantes du VRP selon le type de Contraintes [15].

2.2.1. VRP à Contraintes Liées à la Flotte de Véhicules

- VRP-C (Capacitated Vehicle Routing Problem)

C'est un problème de tournées de véhicules avec des contraintes de capacités.

Les véhicules ont une capacité d'emport limitée (quantité, volume, poids,. . .). Ceci se traduit par le fait que la somme des demandes des clients appartenant a une tournée ne doit pas dépasser la capacité du véhicule. En pratique tous les problèmes de tournées de véhicules sont à capacité limitée seulement le degré d'influence de cette contrainte varie.

Par exemple, les problèmes de transport de marchandise sont très sensibles à la capacité alors que le problème de distribution de courrier l'est moins [15].

- VRP-FL (Vehicle Routing Problem with Full Truckload)

C'est un VRP avec utilisation complète de la capacité du véhicule [15].

- VRP-HF (Vehicle Routing Problem with Heterogeneous Fleet)

Pour ce problème la flotte est composée de véhicules de types différents, qui se distinguent par la capacité, la puissance et le coût de transport.

- O-VRP (Open Vehicle Routing Problem)

Ce problème est identique au VRP, seulement le véhicule est libre de rejoindre ou pas le dépôt après la fin de la tournée. S'il choisit de reprendre le dépôt, il doit reprendre le parcours de la tournée dans le sens inverse [15].

- VRP-B (Vehicle Routing Problem Back)

Le VRP-B est un problème où le retour du véhicule au dépôt est exigé. Le véhicule doit rejoindre le dépôt aussitôt que le dernier client est servi sans reprendre le parcours de la tournée [15].

2.2.2. VRP à Contraintes Liées à la Demande des Clients:

- VRP à Demande Déterministe

C'est un problème fréquent pour les entreprises qui font des livraisons sur commande.

En effet, le livreur connaît avant son départ du dépôt la quantité à livrer à chacun de ses clients [15].

- VRP à Demande Stochastique

Contrairement au précédent, dans le VRP à demande stochastique, le livreur ne connaît pas la quantité à livrer au client il la découvre au moment de le servir.

Il estime approximativement la demande de chaque client par une fonction stochastique.

- S-VRP (Split Delivery Vehicle Routing Problem)

Ce problème consiste à visiter un client plusieurs fois afin de satisfaire entièrement sa demande. Exceptionnellement pour ce problème, la demande du client peut être supérieure à la capacité du véhicule [15].

- VRP-PD (Vehicle Routing Problem Pick-up and Deliveries)

C'est un problème de tournées de véhicules avec collecte et livraison. Avec ce genre de problème la durée du service est comptabilisée deux fois, car on doit effectuer une collecte et une livraison ou inversement [15].

2.2.3. VRP à Contraintes Liées aux Dépôts:

- VRP-MD (Multi-Dépôt Vehicle Routing Problem)

Les véhicules peuvent s'approvisionner de plusieurs dépôts.

- VRP-1D (Vehicle Routing Problem)

Les véhicules doivent s'approvisionner d'un seul dépôt.

On peut passer du VRP-MD au VRP-1D ou inversement par centralisation respectivement division du dépôt(s) [15].

2.2.4. VRP à Contraintes Liées aux Produits:

- MP-VRP (Problème de Tournées de Véhicule à Produits Multiples)

Une gamme de produits doit être livrée aux différents clients par chaque véhicule en une seule tournée [15].

- 1-VRP (Problème de Tournées de Véhicule à un Seul Produit)

Un seul produit doit être livré aux différents clients par chaque véhicule en une seule tournée.

2.2.5. VRP à Contraintes Liées au Temps:

- PVRP (Periodic Vehicle Routing Problem)

Dans le problème de tournées de véhicules périodique, chaque client est périodiquement visité selon une certaine planification prédéfinie.

- VRP a Temps de Service Déterministe

La durée du service est connue par le livreur avant d'entamer la tournée.

- VRP a Temps de Service Stochastique

Le livreur ne connait pas la durée du service des clients, il l'a découvre au moment de les servir. Il peut défini une fonction stochastique pour l'approximer.

- VRP-TW (Vehicle Routing Problem with Time Windows)

Le VRPTW est un problème de tournées de véhicules avec fenêtre de temps. Chaque client doit être servi dans un intervalle de temps défini, connu d'avance par le livreur et toute violation de cette contrainte peut engendrer une pénalité. Lorsque la contrainte de fenêtre de temps n'est pas satisfaite, soit on rejette la solution si on considère le cas rigide ou bien on

construit une fonction de pénalité qui sera rajoutée ou combinée avec la fonction objective pour le cas relâche. En réalité, c'est un problème très fréquent. La distribution des produits périssables (le lait, la viande . . .), de journaux, services ambulatoires sont des exemples pratiques du VRPTW.

Dans cette classe de problèmes, on distingue deux sous-classes :

- Le VRPTW rigide ou le service doit impérativement être effectue dans la fenêtre de temps
- Le VRPTW relâche ou le retard ou l'avance engendre uniquement une pénalité [15].

3. Le problème de transport à la demande (TAD) :

3.1.Positionnement :

Le problème de transport à la demande TAD, est un cas spécial du VRPPD dans lequel les « objets à servir » sont des personnes devant être transportés de leurs origines à leurs destinations. Ce service est destiné particulièrement aux transports des personnes âgées et aux personnes à mobilité réduite.

A l'origine, les TAD étaient appelés en France « Bus à la demande » ou « Minibus à la demande », mais rapidement l'expression « transport à la demande » s'est largement imposée dans la communauté des transporteurs et des chercheurs. Spécifiquement, l'abréviation TAD est utilisée pour désigner tout service de transport qui nécessite une réservation préalable.

À l'heure actuelle, les sigles "DRT"(demand responsive transport) et "DRAP"(Dial A Ride Problem) sert de référence dans les communications internationales pour désigner le transport à la demande. Alors, quelque soit la dénomination attribuée au problème de transport à la demande, ce dernier reste un problème de transport particulier, qui exige des contraintes de qualité de service à cause de la nature humain des objets transportés(les personnes âgés et à mobilité réduite).

3.2. Terminologie et contraintes

Dans la figure 1.2 nous montrons une présentation schématique du TAD. Cette présentation illustre les composants principaux du problème.

Ces composants sont essentiellement, l'ensemble des voyageurs, un nombre finis de stations (Points de départ et de destination) et une flotte de véhicules utilisés pour transporter les voyageurs.

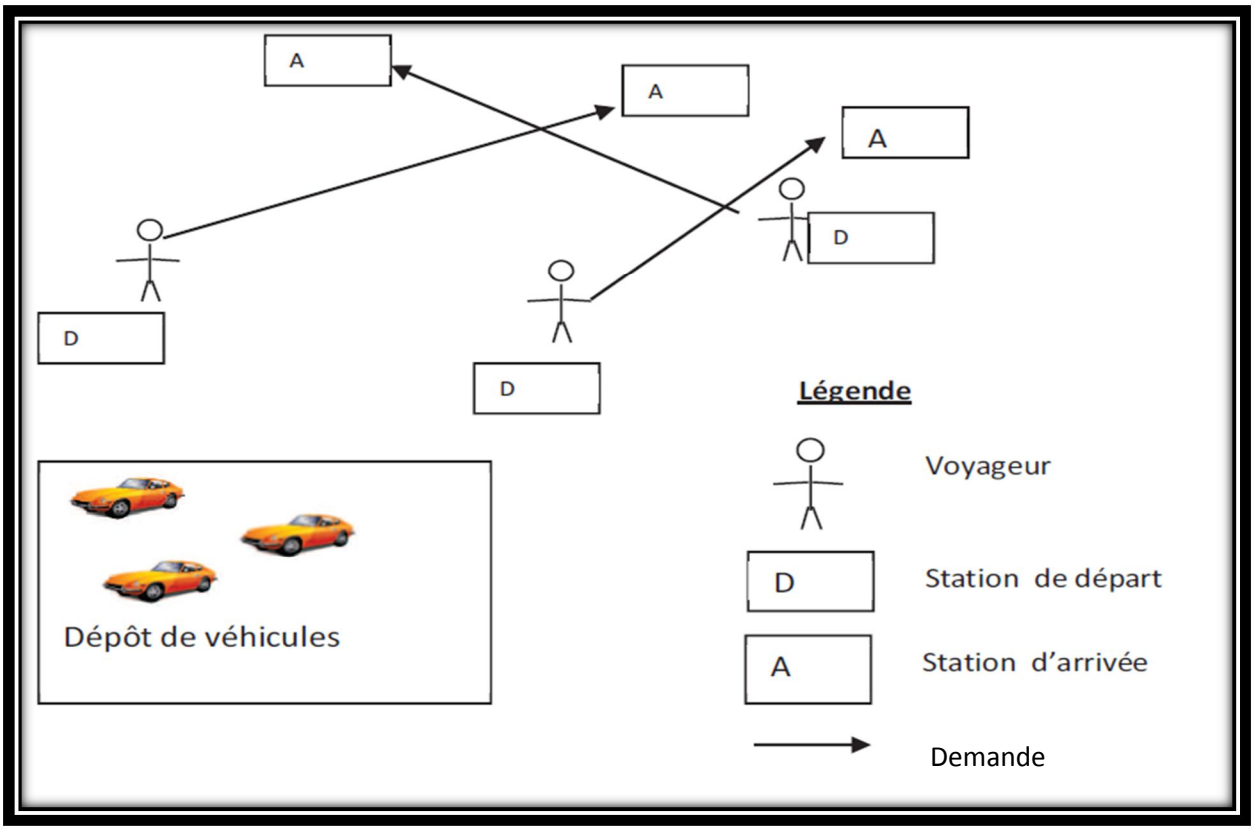

Figure 1.2 Présentation schématique du TAD.

- **Demande:** Une demande est décrite principalement par une station de départ (source) "S" (point de ramassage du voyageur) et une station d'arrivée (destination) "D".
- Contrairement à la majorité des problèmes de VRP dans le TAD le voyageur doit spécifier dans sa demande d'autres informations liées au temps de ramassage et de livraison souhaité. Généralement ces informations sont traduites par les fenêtres de temps $[e_o, l_o]$ pour le point de ramassage et $[e_d, l_d]$ pour le point de livraison. Ces information sont estimées selon les rendez vous et les distances entre les points de départs et d'arrivés. Alors, la forme générale d'une demande est la suivante:

Lieu de départ	**Lieu d'arrivée**	$[e_o, l_o]$	$[e_d, l_d]$

- **Tournée:** À partir d'un ensemble de requêtes plusieurs tournées sont planifiées afin d'assurer la transportation de ces dernières. Chaque tournée présente un chemin à suivre par un seul véhicule. Nous appelons l'ensemble de ces tournées une solution d'un problème de TAD. Dans le cadre de notre travail nous définissons une tournée par une liste de nœuds (départs et arrivés). Cette liste débute et termine par le nœud dépôt.

par exemple: pour trois requêtes :R1(S_1,D_1), R2(S_2,D_2), R3(S_3,D_3) et le dépôt D_{00}(début)/D_{01}(fin) nous définissons la tournée T:

D_{00}	S_1	S_2	D_2	D_1	S_3	D_3	D_{01}

- **Contraintes**: Comme la majorité de problème de tournée de véhicule le TAD exige pendant la génération des tournées le respect de plusieurs contraintes:
  - ❖ La contrainte de capacité du véhicule : est due à la capacité limitée de véhicule(Q).
  - ❖ La durée totale de la tournée : Le temps total d'une tournée ne doit pas dépasser une borne (*T*).
  - ❖ Les fenêtres de temps : chaque nœud a une fenêtre de temps le temps d'arrivé a ce nœud doit être dans cette fenêtre de temps.
  - ❖ Une autre contrainte particulière liée principalement au TAD est "le ride time" (L) c'est une borne sur le temps maximal passé par un client dans le véhicule.

  .

3.3. Ordonnancement et évaluation temporelle:

Étant donné une tournée T = (D_{00}, n1, n2,..ni…. D_{01}) constituée d'une séquence de nœuds avec ni= (sources(S_i) ,destinations (Di) ou bien le dépôt (D_{00}/D_{01}), le problème d'ordonnancement est de déterminer l'heure de départ du dépôt et la date à laquelle le service devrait commencer à chaque sommet ni, de tel sorte que les fenêtres de temps sont satisfaits et la durée de trajet est réduite au minimum. Ce problème est d'une importance critique car il est à la base de l'opération de calcule des coûts des solutions.

Nous utilisons la notation suivante:

- t_{ij}: le temps de Voyage de ni vers nj;
- d_i: la durée du service au sommet ni;
- A_i: l'heure d'arrivée du véhicule à ni;
- B_i: le moment où le service commence à ni;
- D_i: l'heure de départ du sommet ni;
- W_i: le temps d'attente au sommet ni.

- y_i : la charge de véhicule dans le nœud i.
- L_i le temps passé par un client dans le véhicule.

pour évaluer les coûts des solutions dans un problème de TAD la fonction la plus utilisée dans la litérature est la suivante:

Objectif à minimisé F(s)=c(s)+q(s)+d(s)+w(s)+t(s)

- n: nombre de demandes.
- m:nombre de véhicules.
- 2n: nombre total de nœuds (sources et distinations)
- c(s): est la somme des couts C_{ij} associé à l'arce (i,j) traversé par touts les véhicules.
- q(s): la violation de la charge des véhicules calculée par $q(s) = \sum_{i=1}^{2n}(yi - Q)$
- d(s): la violation de la duré des tournées calculée par $d(s) = \sum_{k=1}^{m}(B_{D01}^{K} - B_{D00}^{K} - T)$
- w(s): la violation des fénetres temprelles calculée par $w(s)=\sum_{i=1}^{2n}(B_i - l_i)$
- t(s): la violation des ride times calculée par $t(s)=\sum_{i=1}^{n}(L_i - L)$

4. Classification des problèmes de transport à la demande

Les auteurs classent les problèmes de TAD selon plusieurs critères dans le cadre de notre travail nous citons les trois classifications suivantes[10]:

4.1. Classification selon le nombre de véhicules

Il existe deux types :

- **TAD à un seul véhicule :** dans ce genre de problèmes il existe un seul véhicule qui satisfait toute les demandes.
- **TAD à plusieurs véhicules :** c'est le cas où les demandes sont partagées entre plusieurs véhicules, et chaque demande sera satisfaite par un seul véhicule.

4.2. Classification selon la disponibilité de l'information sur les demandes

Il existe deux types :

- **TAD statique:** c'est le cas où toutes les demandes sont connues avant que les routes soient construites.

- **TAD dynamique:** dans ce cas, certaines demandes sont connues avant que les routes soient construites, et les autres demandes deviennent disponibles en temps réel pendant l'exécution de ces routes.

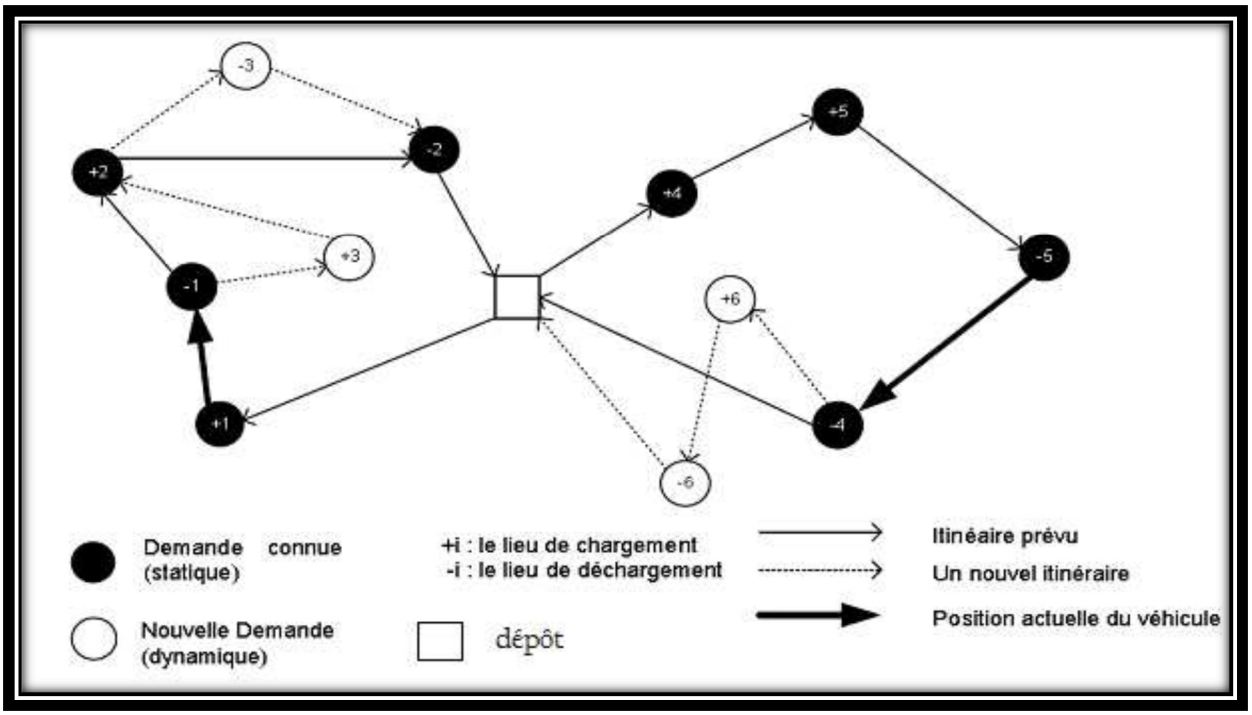

Figure 1.3 Le TAD Statique et dynamique

4.3. Classification selon le type des trajets des demandes

- **TAD avec transfert** : consiste à définir un ensemble de routes qui satisfont les demandes de transport des utilisateurs entre un ensemble de points de ramassage et un ensemble de points de livraison. Les utilisateurs peuvent changer de véhicule au cours de leur voyage. Ce changement de véhicule, appelé un transfert, est faite à des endroits spécifiques appelés points de transfert.
- **TAD sans transfert** : Dans ce type de transport les passagers ne changent pas les véhicules de la source vers la destination.

5. Transport à la demande en pratique

Plusieurs projets commerciaux on été réalisés dans le domaine de TAD à travers le monde. Dans ce qui suit nous présentons quelques uns dans l'Algérie et d'autres pays.

❖ **L'opérateur de transport Ibn sinaa:**

Ibn Sina assistance est un opérateur de transport sanitaire qui travail au niveau de la wilaya de Jijel, son principal service est de transporter les insuffisants rénale depuis ses domiciles vers différents centres de l'hémodialyse et assure leurs retour à la fin des séances.

Le nombre des patients servies par cet opérateur est 107 patients, chaque patient a de 1 à 3 séances par semaine dont le nombre total des séances est 273 séances, cela veut dire qu'il y a 273 demandes de transport par semaine, et puisque pour chaque demande de transport vers le centre de dialyse correspond une autre demande de retour, c'est-à-dire du centre de dialyse vers le domicile du patient. Alors, le nombre total de demande de transport est élevé à 546 demandes réparties sur six jours par semaine (du samedi au jeudi).

❖ **DIANEPHROS MILA :**

Est une clinique d'hémodialyse qui a deux tâches principales: la dialyse et le transport sanitaire des patients. Le nombre des patients servies par cette clinique est 90 patients chaque patient a de 1 à 3 séances par semaine dont 50 patient sont transportés par l'unité de transport de cette clinique qui contient 7 véhicules.

❖ **Taxitub à Saint-Brieuc:**

C'est un service de transport public à la demande mis en place en 1990 et développé par la CABRI (Communauté d'Agglomération de Saint-Brieuc) en partenariat avec les artisans-taxis de l'agglomération. Les réservations des clients se font sur simple appel, le coût de la course est équivalent au prix d'un ticket de bus. Le système est composé de :

· 57 lignes virtuelles fonctionnant en mode fixe avec des horaires préétablis ;

. 17 transporteurs pour 21 véhicules, uniquement des taxis [19].

❖ **Pti'bus à Poitiers**

Pti'bus est un service de transport à la demande mis en place en 1994, destiné à desservir les zones et quartiers où la densité de la population ne peut justifier le passage régulier d'un bus. Les abonnés réservent chaque trajet Pti'bus en appelant Allo'bus. La réservation s'effectue au plus tard une demi-heure avant le départ du véhicule. Si aucune réservation n'est enregistrée sur un trajet, celui-ci n'est pas assuré [19].

6. Le problème de transport à la demande et l'optimisation :

La littérature définit le problème de transport à la demande comme étant un problème d'optimisation combinatoire NP-difficile.

L'optimisation combinatoire est une discipline combinant diverses techniques de la recherche opérationnelles (mathématique) et de l'informatique afin de trouver la meilleure solution dans l'ensemble des solutions réalisables, noté par X.

En général, cet ensemble est fini mais compte un très grand nombre d'éléments qui doivent tous satisfaire un ensemble C de contraintes.

Si f est la fonction objectif qui permet d'évaluer chaque solution réalisable. Alors, un problème d'optimisation combinatoire vise à déterminer une solution $x^* \in X$ qui minimise f.

Donc, La résolution d'un problème de transport à la demande revient à la résolution d'un problème d'optimisation.

6.1. Méthodes et techniques de résolution du TAD :

Dans cette section nous proposons une vue sur les méthodes de résolution du problème de TAD. Notre but n'est pas de détailler le fonctionnement de ces méthodes, mais plutôt d'avoir un aperçu sur leurs principes de bases.

6.2. Classification générale:

Comme les autres problèmes d'optimisation combinatoire, le problème de transport à la demande a été étudié et résolu par des méthodes exactes, des heuristiques spécifiques ainsi que par des méta-heuristiques. Ces trois familles correspondent à la classification générale des méthodes de résolution illustrées dans la figure 1.4.

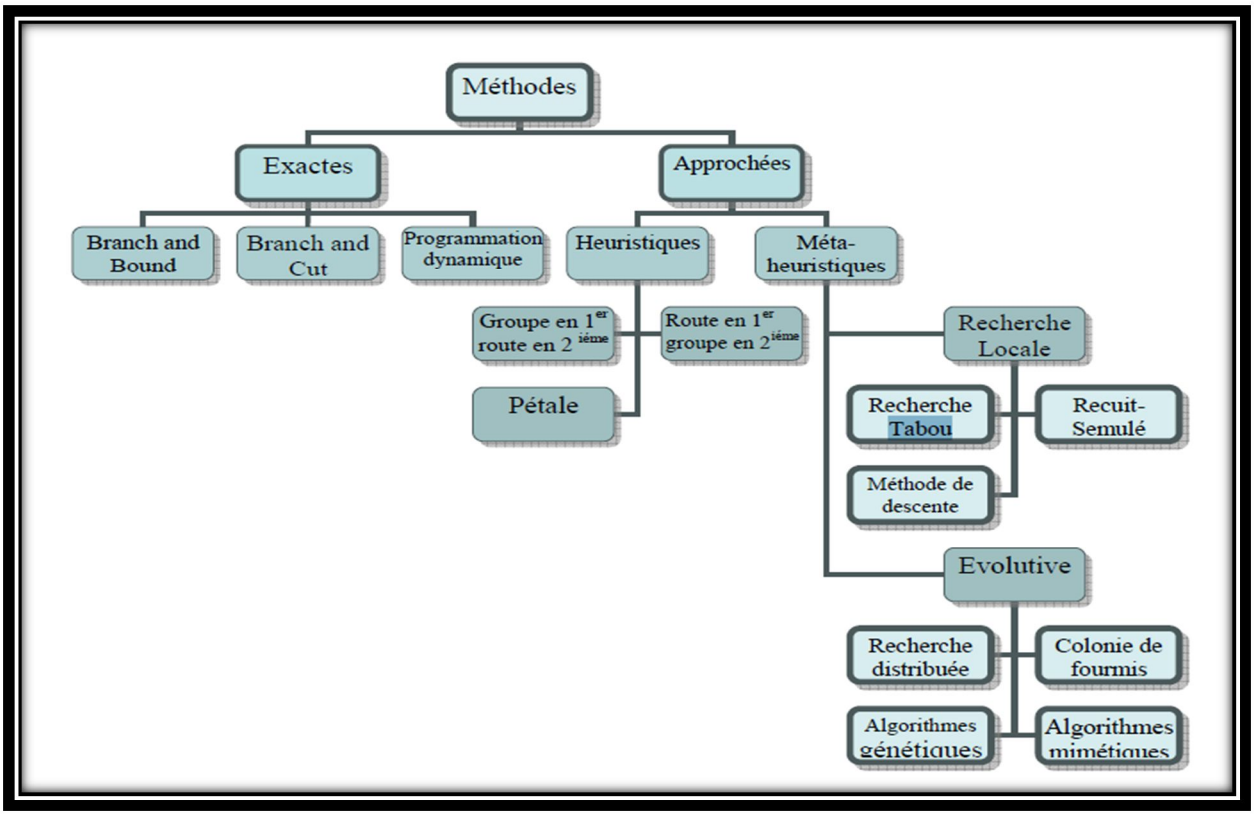

Figure 1.4 Les principales méthodes de résolution de TAD

6.2.1. Les méthodes exactes

Les méthodes exactes reposent sur l'utilisation d'algorithmes qui mènent de façon sûre vers la solution optimale. Le principe essentiel de ces méthodes est d'énumérer de manière implicite l'ensemble des solutions de l'espace de recherche. Malgré l'important temps de calcul que nécessitent généralement ces approches, plusieurs méthodes ont été développées.

Elles permettent de résoudre efficacement des problèmes allant jusqu'à 50 clients [16].

Parmi ces méthodes on peut citer :

6.2.1.1. La méthode de Branch and Bound

La méthode de branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer les solutions d'une manière intelligente, en ce sens que cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche.

La méthode commence par considérer le problème de départ avec son ensemble de solutions. Des procédures de bornes inférieures et supérieures sont appliquées à la racine, si ces deux bornes sont égaux, alors une solution optimale est trouvée, et on s'arrête là, sinon l'ensemble des solutions est divisé en deux ou plusieurs sous-problèmes, devenait ainsi des enfants de la racine.

La méthode est ensuite appliquée récursivement à ces sous-problèmes, Cette technique donne des bons résultats pour les problèmes d'ordonnancement de petites tailles, mais dans le cas contraire, elle risque de générer des branches très étendues [16].

6.2.1.2. La méthode de branch and cut

Elle est aussi appelée méthode de programmation en nombre entier. Comme toute méthode énumérative implicite, l'algorithme construit une arborescence nommée l'arbre du « Branch and cut », les sous-problèmes qui forment l'arbre sont appelés des nœuds. Il existe trois types de nœuds dans l'arbre de "branch and cut", le nœud courant qui est entrain d'être traité, les nœuds actifs qui sont dans la liste d'attente des problèmes et les nœuds inactifs qui ont été élagués au cours du déroulement de l'algorithme. Le principe est de partir d'une solution admissible entière du problème, et à l'aide du simplexe par exemple, vers une autre solution admissible entière jusqu'à l'optimum [16].

6.2.1.3. La programmation dynamique

Cette méthode se base sur le principe de Bellman : « Si C est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C.» Pour obtenir le chemin optimal du problème, il suffit de construire les différents sous chemins optimaux [16].

6.2.1.4. Autres méthodes exactes

En dehors des méthodes d'énumération (branch and bound et branch and cut) et la programmation dynamique décrites précédemment, de nombreuses approches exactes différentes existent, et utilisent les spécificités du problème traité pour résoudre le problème d'optimisation. C'est le cas de l'algorithme du simplexe et de l'algorithme À*[16].

6.2.2. Les méthodes approchées

Les méthodes de résolution approchée sont généralement utilisées là où les méthodes exactes échouent. En effet, une résolution exacte nécessite de parcourir l'ensemble de l'espace de recherche, ce qui devient irréalisable lorsque l'on veut résoudre de gros problèmes. Dans ce cas, une exécution partielle de l'algorithme exact permet rarement d'obtenir une solution de bonne qualité. Des méthodes de résolution approchée ont été mises au point afin de procurer rapidement des solutions de bonne qualité mais non optimales [16]

6.2.2.1.Les heuristiques

Une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile.

Vu que les méthodes exactes restreignent le nombre des clients envisageables dans les problèmes et impliquent, dans la plupart des cas, un temps de calcul important, l'élaboration et l'utilisation des heuristiques se sont avérées d'une grande utilité. Ces méthodes permettent de gérer des problèmes de grandes tailles avec des temps de résolution et des résultats acceptables.

Parmi les heuristiques qui traitent le TAD, nous citons :

A. L'heuristique « groupe en premier, route en second »

C'est une des heuristiques les plus connues. Elle se base sur l'aspect géométrique de problème. Elle consiste à grouper les nœuds qui sont géographiquement voisins, et chaque groupe est assigné à un véhicule, ensuite pour chaque véhicule le TSP correspondant est résolu [16].

B. L'heuristique « route en premier, groupe en second »

Le principe de cette heuristique est de construire des tournées comportant un grand nombre de clients, qui sont réellement non réalisables, puis de les subdiviser en de petites tournées pour obtenir des solutions acceptables pour le TAD [16].

C. L'algorithme en pétale

Cet algorithme de balayage génère plusieurs routes, appelées pétales, pour ensuite faire une sélection en résolvant un problème de partitionnement. Notons que si les routes correspondent à des secteurs continus des arcs, alors le problème peut être résolu en temps polynomial[16].

6.2.2.2. Les méta-heuristiques

Les méta-heuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie comme les algorithmes évolutionnaires.

Ces méthodes permettent de ne pas s'arrêter aux optimums locaux, qui sont des solutions réalisables qui peuvent être loin de l'optimum global.

Dans ce qui suit nous allons présenter les méta-heuristiques les plus prisées par les chercheurs. Leurs différentes utilisations et les améliorations qu'elles ont connues ont fait que les résultats établis ne cessent de s'améliorer.

Les méta-heuristiques sont subdivisées en deux grandes familles :

- les méta-heuristiques de recherche locale

- les méta-heuristiques d'évolution [16].

A. Les méta-heuristiques de recherche locale

Les méta-heuristiques de recherche locale sont basées sur un algorithme de recherche de Voisinage qui commence avec une solution initiale, puis l'améliore à pas en choisissant une nouvelle solution dans son voisinage. Les plus classiques sont le recuit simulé, la recherche taboue et la méthode de descente (recherche locale)[16].

A.1 Méthodes de recuit simulé

Cette méthode de recherche a été proposée par des chercheurs d'IBM qui étudiaient les verres de spin. Ici, on utilise un processus métallurgique (le recuit) pour trouver un minimum.

En effet, pour qu'un métal retrouve une structure proche du cristal parfait (l'état cristallin correspond au minimum d'énergie de la structure atomique du métal), on porte celui-ci à une température élevée, puis on le laisse refroidir lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement [16].

Le pseudo code du recuit simulé est représenté dans l'algorithme 1.1.

Algorithme 1.1 Recuit simulé.

1. **Init** T (température initiale) ;
2. **Init** x (point de départ) ;
3. **Init** ΔT (température) ;
4. **Répéter**
5. $Y = \text{Voisin}(x)$;
6. $\Delta C = \Delta(y) - \Delta(x)$;
7. **Si** $\Delta C < 0$ **Alors** $y = x$;
8. **Sinon**
9. $p = e^{\frac{-\Delta C}{T}}$;
10. r = valeur aléatoire dans [0,1] ;
11. **Si** $r < p$ **Alors** $y = x$;
12. $T = \alpha(T)$;
13. **Jusqu'a** $(T > \Delta T)$

Figure 1.5 Algorithme de Recuit simulé

Dans cet algorithme, le paramètre de contrôle est la température T. Si la température est élevée la probabilité p tend vers 1 et presque tous changements sont acceptés. Cette température diminue lentement au fur et à mesure du déroulement de l'algorithme pour simuler les processus de refroidissement des matériaux. Sa diminution est suffisamment lente pour que l'équilibre thermodynamique soit maintenu [16].

A.2 Recherche Tabou

La recherche Tabou est une méta-heuristique originalement développée par Glover [7]. Elle est basée sur des idées simples, mais reste néanmoins efficace.

Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes lui permettant de surmonter l'obstacle des extremums locaux, tout en évitant les problèmes de cycles. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire : problèmes de routage de véhicules, problèmes d'ordonnancement, problèmes de coloration de graphes [16].

Les différentes étapes de la recherche tabou sont :

1. Choisir une solution initiale $s \in X$;

   $s^* := s$;

   $T := \phi$.

2. Tant que le critère d'arrêt n'est pas vérifié faire

   - engendrer un échantillon $V^* \subseteq N(s) - T$;
   - rechercher $s' \in V^*$ telle que $f(s') = \min_{x \in V^*} f(x)$;
   - $s := s'$;
   - si $f(s') \leq f(s^*)$ alors $s^* := s'$;
   - mettre à jour la liste T.

Figure 1.6 les étapes de la recherche tabou

A.3 La méthode de descente

La méthode de descente, ou de recherche locale, est très ancienne et doit leur succès à leur rapidité et leur simplicité .A chaque pas de la recherche, ces méthodes progressent vers une solution voisine de meilleure qualité. La descente s'arrête quand tous les voisins candidats sont moins bons que la solution courante, c'est-à-dire lorsqu'un optimum local est atteint. On

distingue différents types de descentes en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage [16].

B. Les méta-heuristiques d'évolution

Le principe des méta-heuristiques évolutives est de faire évoluer un ensemble de solutions vers l'optimum cherché. Parmi ces méthodes, nous distinguons essentiellement les algorithmes génétiques, les colonies de fourmis et la recherche distribuée [16].

B. 1 Les algorithmes génétiques

Les algorithmes génétiques s'attachent à simuler le processus de sélection naturelle dans un environnement hostile lié au problème à résoudre. Ils utilisent un vocabulaire similaire à celui de la génétique naturelle, tout en rappelant que les principes sous-jacents liés à ces deux domaines sont beaucoup plus complexes dans le cadre naturel. On parlera ainsi d'individu dans une population et bien souvent l'individu sera résumé par un seul chromosome. Les chromosomes sont eux-mêmes constitués de gènes qui contiennent les caractères héréditaires de l'individu. On retrouvera aussi les principes de sélection, de croisement, de mutation [16].

B.2 Les colonies de fourmis

La méthode des colonies de fourmis simule le comportement de ces insectes qui, lorsqu'on pose un obstacle sur leur trajet, trouvent toujours le chemin le plus court pour contourner.

Leur technique repose sur la pose de marqueurs chimiques, les phéromones, déposées sur les trajets parcourus. Cela peut paraître surprenant au premier abord mais un chemin plus court reçoit plus de phéromones qu'un chemin plus long [16].

Le pseudo code de la colonie de fourmis est représenté dans l'algorithme 1.2.

Algorithme 1.2 Colonies de fourmis.

1. **Initialiser** les traces
2. **Tant qu'un** critère d'arrêt n'est pas satisfait
3. **Répéter** en parallèle pour des p fourmis :
4. Construire une nouvelle solution à l'aide des informations contenues dans les traces et
5. une fonction d'évaluation partielle;
6. Evaluer la qualité de la solution;
7. Mettre à jour les traces;

Figure 1.7 Colonies de fourmis

B.3 La recherche distribuée

Partant du principe des algorithmes génétiques, cette méthode assure l'évolution des solutions à l'aide d'un opérateur de combinaison sans imposer de règles de codage.

Enfin pour améliorer le résultat final, il est permis d'utiliser l'une des méthodes de la recherche locale [16].

7. Conclusion

Dans ce chapitre, nous avons présenté le problème de tournée de véhicules et ces variantes. Ensuite, nous avons détaillé le problème de transport à la demande. Enfin, nous avons présenté les principes des méthodes d'optimisation utilisées pour résoudre ce problème.

Chapitre 02

L'Aide À la Décision dans le TAD

1. Introduction

Les outils d'aide à la décision sont de plus en plus indispensables à la définition de la stratégie d'une entreprise et aux prises de décisions auxquelles elle est confrontée chaque jour. Ils sont conçus pour permettre des décisions rapides et motivées.

Dans ce deuxième chapitre nous introduisons dans un premier temps des généralités liées au domaine d'aide à la décision. Nous déterminons par la suite l'intérêt de l'aide à la décision et des outils d'aide à la décision dans le domaine de TAD.

2. Terminologie

2.1. La Décision

La décision peut être définie comme étant l'action de décider après délibération, et que l'acteur exerce un rôle important. Ce n'est donc pas un acte élémentaire et simple, mais c'est plutôt l'aboutissement de tout un processus de décision[9].

2.2. Aide à la décision

L'aide à la décision est l'activité de celui qui, prenant appui sur des modèle, clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponse aux questions que se pose un acteur dans un processus de décision. Ces éléments concourant à éclairer la décision et normalement à recommander, ou simplement à favoriser, un comportement de nature à accroître la cohérence entre l'évolution du processus d'une part, les objectifs et les systèmes de valeurs au service desquels ces acteurs se trouve placé d'autre part » [9].

2.3. Outil d'aide à la décision

Un outil d'aide à la décision est un moyen (informatique ou non) qui aide les décideurs à dégager des informations utiles à partir de données brutes, de documents, de connaissances personnelles et de modèles métier, afin d'identifier et résoudre des incidents et prendre des décisions[9].

2.4. Outil d'aide à la décision interactif

Un outil d'aide à la décision interactif est un système informatique qui doit aider un décideur à suivre un processus d'aide à la décision. C'est grâce à l'interactivité qui est à la base de la coopération des deux partenaires (l'outil et l'utilisateur) que l'outil accomplira le rôle pour lequel il a été conçu[9].

3. Classification des outils d'aide à la décision interactifs

Quelque soit l'activité et la taille des entreprises, l'utilisation des outils d'aide à la décision est une obligation pour assurer les bénéfices attendu. Le marché des outils d'aide à la décision est un marché en perpétuelle évolution, il propose des outils variés du simple calculateur jusqu'aux outils complexes d'inférence et de simulation.

3.1. Outils simples

L'outil d'aide à la décision le plus simple qui soit, présent dans beaucoup d'entreprises est en fait un système de visualisation de données. La feuille Excel faisant l'affaire, dans bien des cas, c'est le premier des « logiciels d'aide à la décision ». Quand l'entreprise croît, la feuille Excel laisse sa place à des « cubes » agrégeant les indicateurs de performance, et facilitant des comparaisons entre produits, entre segments clients, entre business unit, et incitant à la prise de décision [32].

3.2. Outils évolués

Avec l'évolution des entreprises l'utilisation des outils simples est devenue inefficace. Alors, l'adoption de nouveaux outils évolués est une nécessité primordiale. Les outils évolués d'aide à la décision proposent des fonctionnalités de calcule et d'analyse très performants. Ces outils assistent les décideurs dans les différentes phases de la réalisation des projets, de la planification à la l'analyse des résultats. La majorité des outils évolués reposent sur des algorithmes d'optimisation et de planification et il propose même des plateformes de simulations interactives.

3.2.1. L'aide à la décision par optimisation

De nos jours, l'environnement des entreprises se transforme. La concurrence mondiale, les développements technologiques font que l'accent est de plus en plus mis sur la productivité, la qualité et la satisfaction de la clientèle. Pour arriver à de bons résultats, il est devenu vital de prendre rapidement de bonnes décisions. Cependant, la complexité des problèmes industriels, le nombre sans cesse croissant d'objectifs à optimiser simultanément et la rapidité des changements de l'environnement raccourcissent considérablement les délais de prise de décision tout en rendant cette tâche plus difficile pour les gestionnaires. Dans un tel contexte, des outils informatiques d'optimisation s'avèrent d'une grande utilité pour le décideur car ils lui permettent d'évaluer les situations et les diverses alternatives et leurs impacts éventuels dans un temps raisonnable [11].

La majorité des outils d'aide à la décision par optimisation proposent des fonctionnalités de calcule rapide et efficace basé sur les différentes heuristiques et méthodes d'optimisation (voir chapitre1). Actuellement des centaines d'outils d'optimisation standards sont disponibles où chaque outil propose des services compétitifs.

3.2.2. L'aide à la décision par simulation

3.2.2.1.Définition

La simulation informatique désigne l'exécution d'un programme informatique sur un ordinateur ou réseau en vue de simuler un phénomène réel et complexe. Elle serve à étudier le fonctionnement et les propriétés d'un système modélisé ainsi qu'à en prédire son évolution. Les interfaces graphiques permettent la visualisation des résultats des calculs par des images de synthèse [32].

ce type de simulation est rapidement devenus incontournable pour la modélisation des systèmes naturels en physique, chimie et biologie, mais également des systèmes humains en économie et en science sociale. Elle permet de limiter le risque et d'éviter le coût d'une série d'épreuves réelles (ex: essais de véhicules). Elles peuvent offrir un aperçu sur le développement d'un système trop complexe pour simuler avec de simples formules mathématiques [34].

La simulation est un outil d'aide à la décision puissant pour l'homme plus particulièrement l'industriel. Faute de pouvoir mettre l'usine en équation (les contraintes sont multiples et les acteurs décideurs sont de plus en plus nombreux), la simulation permet de reproduire son fonctionnement sur ordinateur et comparer des scénarios d'exploitation. On peut de cette manière, tester l'impact d'un investissement, d'une modification d'un paramètre, du lancement d'un nouveau produit sur une ligne de production ou d'une autre règle d'ordonnancement. Cette technique peut aider à mieux cerner les conséquences de choix potentiels d'actions, afin de mieux les maîtriser et d'améliorer ainsi le pilotage de l'atelier [32].

3.2.2.2. Catégories de simulation informatique

On peut distinguer trois catégories de simulations[34] :

- **La simulation continue** : où le système se présente sous la forme d'équations différentielles à résoudre. Elle permet de suppléer à la résolution analytique quand celle-ci est impossible. Effectuée au départ sur des calculateurs analogiques, elle s'est effectuée aussi sur des ordinateurs ainsi que des *machines hybrides*, et un troisième type de calculateurs qui n'a pas eu de lendemain, les calculateurs stochastiques.

- **La simulation discrète** : dans laquelle le système est soumis à une succession d'événements qui le modifient. Ces simulations ont vocation à appliquer des principes simples à des systèmes de grande taille. La simulation discrète se divise en deux grandes catégories :

  - asynchrone ou *time-slicing* : on simule à chaque fois le passage d'une unité de temps sur tout le système. Ce terme n'est généralement plus utilisé dans le domaine professionnel depuis l'apparition croissante des nouvelles technologies.

  - synchrone ou *event-sequencing* : on calcule l'arrivée du prochain événement, et on ne simule qu'événement par événement, ce qui permet souvent des simulations rapides, bien qu'un peu plus complexes à programmer.

- **La simulation par agents** : où la simulation est segmentée en différentes entités qui interagissent entre elles. Elle est surtout utilisée dans les simulations économiques et sociales, où chaque agent représente un individu ou un groupe d'individus. Dans le cadre de notre travail nous nous intéressons à la simulation par le paradigme muti-agent détaillé dans la section suivante.

4. le paradigme multi-agents

La technologie des systèmes basés sur les agents a généré ces dernières années beaucoup d'attention. Ceci est du au fait que ces systèmes ont donné naissance à un nouveau paradigme appelé Multi-Agents (SMA). Ce paradigme permet en effet la compréhension, la conception et la simulation de la façon la plus naturelle possible des différentes entités qui sont manipulées par des systèmes complexes.

4.1.L'Agent

4.1.1. Définitions

Plusieurs définitions attribuées à la notion d'agent existent dans la littérature, donc il est encore difficile de donner une définition commune ou une bonne vision de ce concept car cette diversité a engendré des définitions aussi riches que variées. On peut présenter quelques définitions importantes :

- ***Selon Jacques Ferber [20]*** : Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-

agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

- ***Une définition plus concise est donnée par Wooldridge and Jennings[21]*** **:** Un agent est un système informatique qui est situé dans un environnement et qui est capable d'appliquer des actions autonomes dans le but de satisfaire ses buts.
- ***Et depuis Florez [22]*** *:* Un agent est une entité interactive qui existe en tant que partie d'un environnement partagé par d'autres agents. C'est une entité conceptuelle, qui perçoit et agit avec initiative ou en réaction, dans un environnement où d'autres agents existent et interagissent les uns avec les autres, sur la base de connaissances partagées de communication et de représentation.

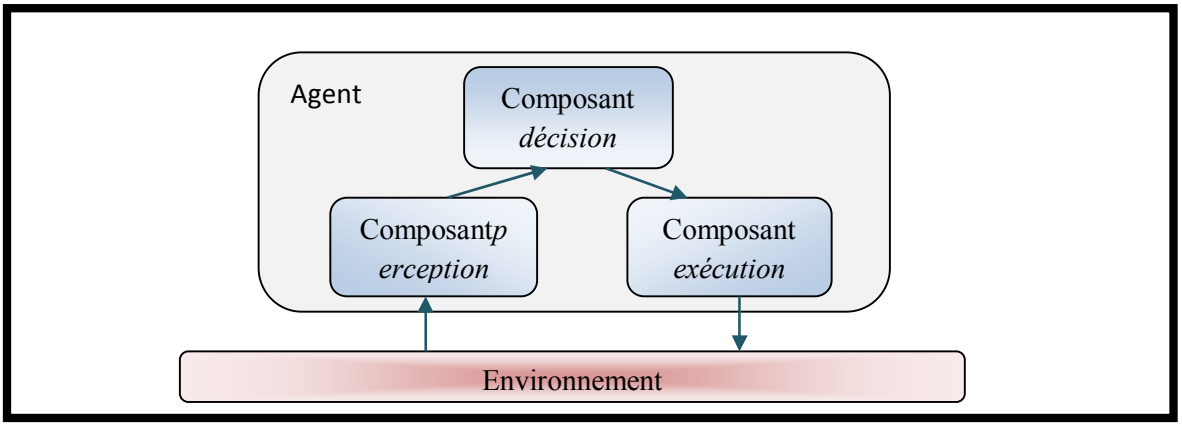

Figure 2.1 : Structure générale d'un agent

4.1.2. Caractéristiques d'agent

Il ressort des définitions précédentes des caractéristiques clés, comme l'autonomie, la réactivité, la pro-activité et la sociabilité [22] :

- ***L'autonomie*** : L'agent est capable de prendre des initiatives et d'agir sans l'intervention d'autres agents et notamment de l'être humain. Et doit également avoir un certain degré de contrôle sur ses actions et sur ses états internes.
- ***La réactivité*** : C'est la capacité de l'agent de percevoir son environnement et de réagir automatiquement en fonction des changements de son environnement.
- ***La pro-activité*** : Un agent ne devrait pas simplement réagir à des événements dans son environnement, mais également pouvoir initier des actions en concordance avec ses objectifs.

- ***La sociabilité*** : L'agent est capable d'interagir avec d'autres agents ou avec un humain quand la situation l'exige afin de satisfaire ses objectifs ou aider les autres à satisfaire les leurs.

4.1.3. Classification des agents

Faut-il concevoir les agents comme des entités intelligentes, capables de résoudre les problèmes par eux-mêmes, ou bien faut-il les assimiler à des entités simples, réagissant directement aux variations de l'environnement, La réponse à cette question a donné naissance à trois grandes catégories d'agents [22] :

- ***Agent réactif :*** Un agent réactif n'a pas de représentation de son environnement. Il agit avec un comportement de stimulus/ réponse et réagit à l'état présent de l'environnement dans lequel il est situé. Ce genre d'agents ne possède pas le moyen de mémorisation car il ne tient pas compte du passé et ne planifie pas le futur. A travers des interactions simples avec les autres, le comportement global complexe peut apparaître. Les propriétés principales de ces systèmes sont la robustesse et la tolérance aux fautes. En effet, un groupe d'agents peut compléter une tâche quand l'un d'eux échoue. Par contre, les agents réactifs ont un comportement myope, puisqu'ils ne prévoient pas l'effet des décisions locales sur le comportement global du système.

- ***Agent cognitif :*** L'agent cognitif possède une représentation explicite de son environnement et des autres agents. Il est aussi doté de capacités de perception-raisonnement-action. Ainsi, les agents cognitifs disposent d'une base de connaissances et de plans explicites leur permettant d'atteindre leurs buts.

- ***Agent hybride :*** Les agents hybrides sont des agents ayant des capacités cognitives et réactives. Ils conjuguent en effet la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs .

4.2. Les systèmes multi-agents

4.2.1. Définition

Un Système Multi-agent (SMA) est un ensemble d'agents qui interagissent dans un environnement pour résoudre des problèmes qui dépassent les capacités ou les connaissances individuelles de chaque agent [20]. L'interaction peut aussi bien être une coopération qu'une compétition et l'environnement représente un espace commun d'interaction pour tous les

agents. Cet environnement dépend de la nature de l'application considérée, il peut être par exemple le monde physique, les données d'un problème [22].

Donc, un SMA comme est définit par J.Ferber consiste d'un *environnement*, d'un ensemble d'*objets passifs* et d'un ensemble d'*agents actifs*. Les objets sont reliés entre eux par un ensemble de *relations* définies et sont manipulés par les agents qui les perçoivent [24].

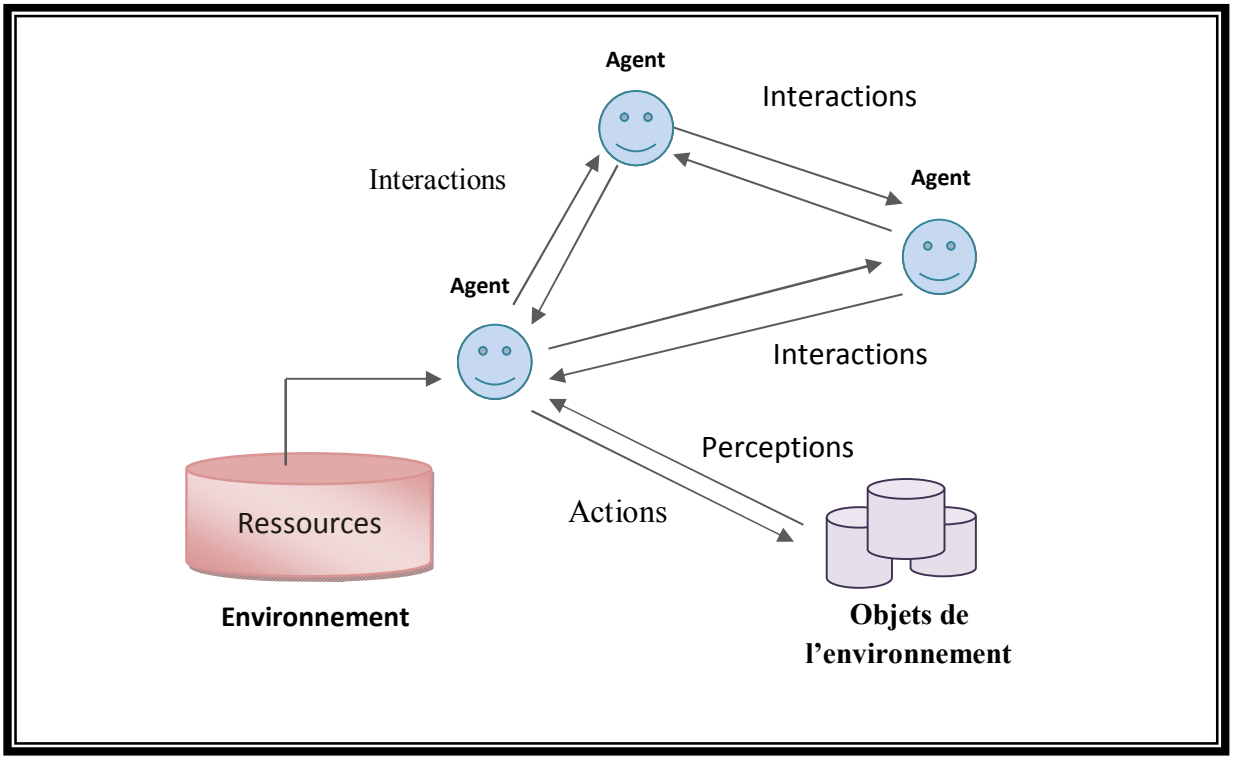

Figure 2.2 : Architecture d'un système multi-agent

4.2.2. Communication dans un SMA

La communication entre les agents repose explicitement sur les mécanismes d'envoie de messages, de réception et de synchronisation. Les agents doivent disposer d'un langage de communication pour l'échange d'information afin de pouvoir coopérer pour la résolution d'un problème. Ce langage appelé mécanisme de communication inter-processus est un ensemble de primitives connues par chaque agent et dont l'ensemble constitue un protocole de communication. Dans la littérature, deux modes de communication sont connus :

- La communication par envoie de messages dans laquelle les agents envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire. Les systèmes fondés sur ce mode relèvent d'une

distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème [25].

- La communication par partage d'information où les agents ne sont pas en liaison directe mais communiquent via une structure de données partagée dans laquelle se trouvent les connaissances relatives à la résolution qui évolue durant le processus d'exécution [20].

4.2.3. Coopération dans un SMA

La coopération est une caractéristique très importante dans les SMA. Une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. Un agent doit Savoir mettre à jour le modèle du monde environnant, intégrer les informations émanant d'autre agents, interrompre pour aider d'autres agents et déléguer une tâche qu'il ne sait pas résoudre à un agent dont il connaît les compétences. Ces caractéristiques constituent les qualités essentielles d'un agent coopératif [22].

4.2.4. Coordination dans un SMA

Dans un système multi-agents le travail des agents doit être temporellement coordonné. Ceci permet aux agents de prendre en compte toutes les tâches en évitant la duplication du travail. Dans les systèmes IAD, on distingue deux schémas principaux de coordination : une coordination au moyen d'un système capable de déterminer et de planifier les actions des différents agents, ou capable de donner une totale autonomie aux agents qui, à leur tour, identifient les conflits pour les résoudre localement [25].

4.2.5. Négociation dans un SMA

Les activités des agents dans un système distribué sont souvent interdépendantes et entraînent des conflits. Pour les résoudre, il faut considérer les points de vue des agents, les négocier et utiliser des mécanismes de décision concernant les buts ciblés. Le processus de négociation ne consiste pas forcément à trouver un compromis mais peut s'étendre à la modification des croyances d'autres agents pour faire prévaloir un point de vue. Pour mener à bien ce type de processus, il est nécessaire de suivre un protocole qui facilite la convergence vers une solution [25].

5. L'aide à la décision dans le domaine de transport à la demande

Dans le secteur de transport à la demande, la décision est considérée comme étant le centre de l'acte du transport. Le processus de la décision dans le TAD consiste entre autres à construire des solutions par des heuristiques d'optimisation afin d'assurer une planification efficace des demandes des clients même dans les cas de perturbation où la prise d'une décision rapide et efficace est important.

Ainsi, de très nombreux logiciel d'aide à la décision ont été développés dans ce domaine. Ces logiciels sont destinés à soutenir l'opérateur de transport dans sa prise de décision. La majorité des logiciels disponible sont des logiciels d'optimisation ou bien de simulation des cas de perturbation (panne de véhicule, coupeur de chemin, changement de demande). Dans les sections suivantes nous présentons quelques logiciels.

5.1.Le logiciel Smartour

"Smartour "optimise automatiquement les tournées tout en intégrant les contraintes des opérateurs de transport et celles des clients. Ainsi, le logiciel est entièrement personnalisé et s'adapte aux exigences des processus internes, au parc informatique ou aux tâches complexes de planification [26].

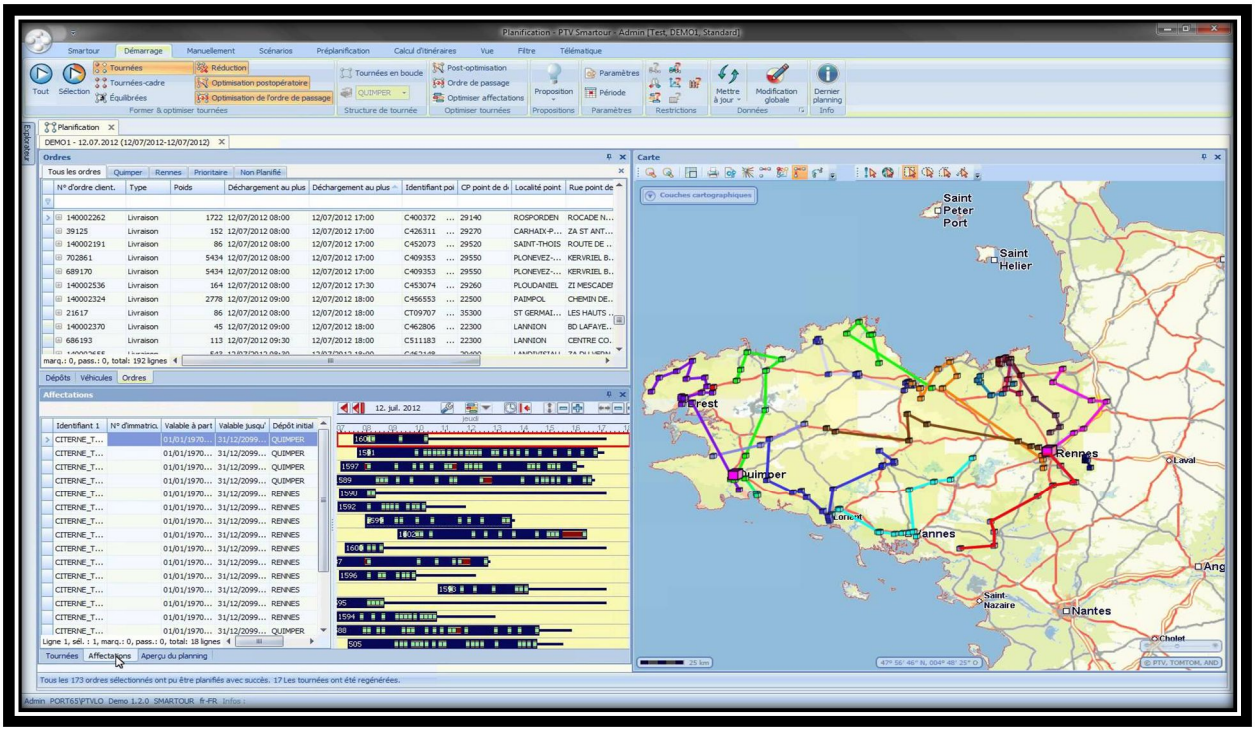

Figure 2.3 : Interface du logiciel Smartour

5.2. Toursolver

Toursolver est un logiciel pour automatiser et optimiser la gestion des tournées, tout en respectant les contraintes liées aux ressources et les besoins des clients. Ce logiciel optimise les itinéraires des tournées à réaliser pour répartir au mieux les clients à visiter entre les ressources mobiles (techniciens, livreurs, commerciaux...) et générer les plans de tournées optimaux. Il permet aussi de réduire les coûts de déplacement, d'optimiser les plannings et d'améliorer le service client.

Toursolver génère l'organisation et les itinéraires les plus rentables, tout en prenant en compte les contraintes de vos clients, les contraintes liées à vos ressources humaines et les spécificités de chaque véhicule [27].

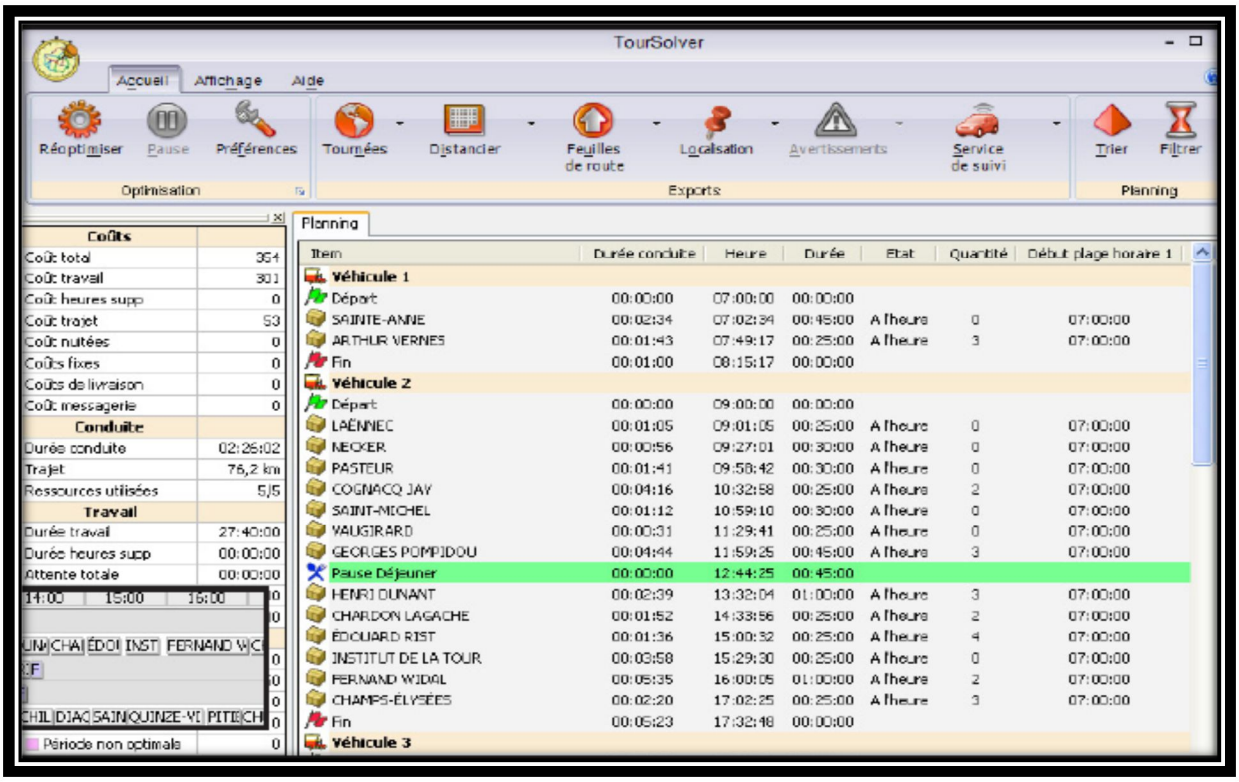

Figure 2.4 : Interface de Toursolver

5.3. Transept ROUTEUR

Transept ROUTEUR est un logiciel d'optimisation de tournées. Il permet de planifier les missions à partir de la sélection d'un ensemble d'ordres de transport, en fonction des ressources disponibles et de différentes contraintes relatives aux conducteurs (temps de conduite, fréquence de retour base...), aux conditions de transport (vitesse, capacités...) et de livraison (horaires, type de véhicules...) [28].

Figure 2.5 : Interface de transept ROUTEUR

6. Conclusion

L'utilisation des outils d'aide à la décision assure l'amélioration de la qualité de service et de résoudre des problèmes de planification au niveau stratégique, tactique et opérationnel dans les entreprises. Spécifiquement dans le domaine de transport ces outils proposent dans nos jours des fonctionnalités magiques dans l'augmentation des rendements.

Dans le chapitre suivant nous détaillons les fonctionnalités de notre outil d'aide à la décision.

Chapitre 03

Outil d'Aide À la Décision pour une Agence de TAD

1. Introduction

Ce troisième chapitre est dédié à la modélisation de notre propre outil. Dans un premier temps nous présentons les limites fonctionnelles de l'outil. Ensuite, nous citons en détailles les fonctionnalités de ce dernier. Dans un second temps nous présentons les différents modèles et heuristiques utilisées dans la conception.

2. Limites fonctionnelles de notre outil d'aide à la décision.

Notre objectif est de proposer un moyen pour assister les gestionnaires d'une agence de transport à la demande dans les tâches quotidiennes. Alors notre outil d'aide à la décision est destiné à un usage collaboratif avec d'autres outils d'aide à la décision, des opérateurs humain et même des systèmes d'information (bases de données de l'agence).

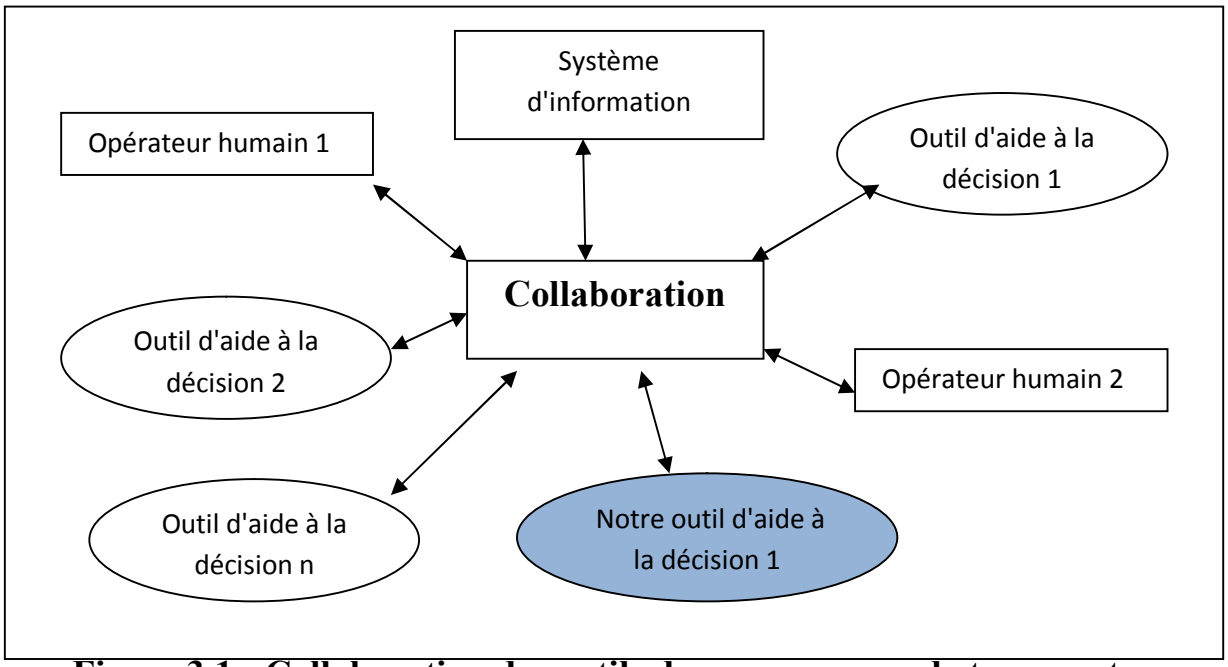

Figure 3.1 : Collaboration des outils dans une agence de transport.

3. Description détaillée des fonctionnalités de l'outil :

L'aide à la décision dans le domaine de transport à la demande prend plusieurs formes. Dans le cadre de notre travail nous nous limitons à la réalisation d'un outil qui assure la fonctionnalité de base d'un tableau de bord pour la génération des tournées et un simulateur de cas de coopération.

3.1. Un tableau de bord pour la génération de tournées

La génération de tournées est la fonctionnalité principale ou bien minimale assurée par n'importe quel outil d'aide à la décision. Dans le cadre de notre travail, nous proposons cette fonctionnalité sous la forme d'un tableau bord interactif où l'utilisateur peut lancer la génération des tournées, voir les résultats sous plusieurs formes (texte, graphe) et même changer les paramètres de calcul. Un tableau de bord est un outil de gestion qui présente synthétiquement les activités et les résultats sous forme d'indicateurs qui permettent de contrôler la réalisation des objectifs fixés et de prendre des décisions nécessaires dans un

délai limité. Pour assurer cette fonctionnalité nous proposons l'utilisation de l'algorithme de "***recherche locale itérative"*** comme heuristique d'optimisation (voir section 4).

3.2. Simulation de cas de coopération

La deuxième fonctionnalité assurée par notre outil d'aide à la décision est liée principalement à la simulation de cas de coopération entre plusieurs agences distribuées géographiquement et gérées par un seul opérateur de transport. Cette simulation donne à un opérateur de transport une vision préalable sur la possibilité et la rentabilité de la coopération entre plusieurs agences par rapport à une activité isolée des agences.

Dans le cadre de notre travail nous limitons la forme de coopération dans la possibilité de partager les requêtes entre plusieurs agences (dans le cas de requêtes qui dépassent la zone d'activité d'une seule agence et qui nécessite la coopération de plusieurs agences pour assurer la transportation du client concerné par la requête). Ce type de transport à la demande et appelé transport avec transfert.

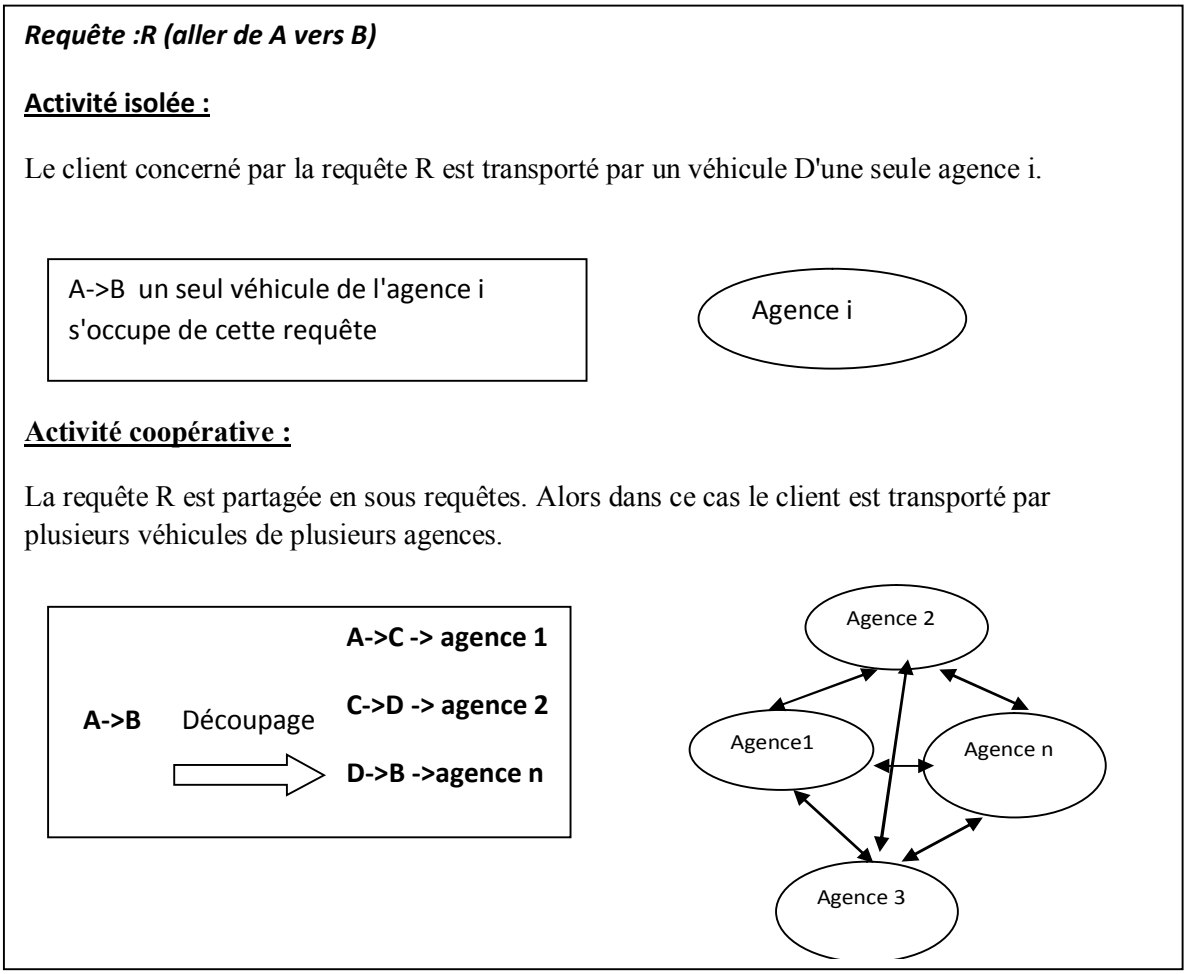

Figure 3.2 : Transportation coopérative des requêtes

La réalisation de cette fonctionnalité est assurée par le paradigme Multi-agents.

4. La méthode de modélisation utilisée

Pour modéliser notre SMA nous avons utilisé la méthodologie O-MaSE (voir annexe)

Dans les paragraphes suivants nous présentons les principaux diagrammes, à savoir le diagramme de buts, le diagramme de rôle, le diagramme d'agents et le diagramme de protocole

4.1. Le diagramme de but.

D'après notre deuxième fonctionnalité (simuler la coopération entre agences), le but principal est de "**servir un ensemble de demandes en coopération"**. Cet objectif représente le but globale, noté **But0.**

- Le **But0** dépend de la réalisation de deux buts fils : gérer les demandes **(But1)** et construire les solutions **(But2)**
- Le **But1** dépend à son tour de la réalisation de deux buts : vérifier la validité des demandes **(But1.1)** et stocker les demandes dans la base de demandes ou un fichier texte **(But1.2).**
- Le **But 2** dépend à son tour de la réalisation de deux buts : distribuer les demandes aux agences (**But2.1**) et optimiser la solution(**But2.2**).
- le But 2.1 nécessite deux buts fils : obtenir les demandes locales **(But2.1.1)** et obtenir les demandes globales**. (But2.1.2).**
- En effet, pour optimiser une solution (**But2.2**)., on doit faire une optimisation centralisée **(But 2.2.1)** et une optimisation coopérative **(But 2.2.2)**

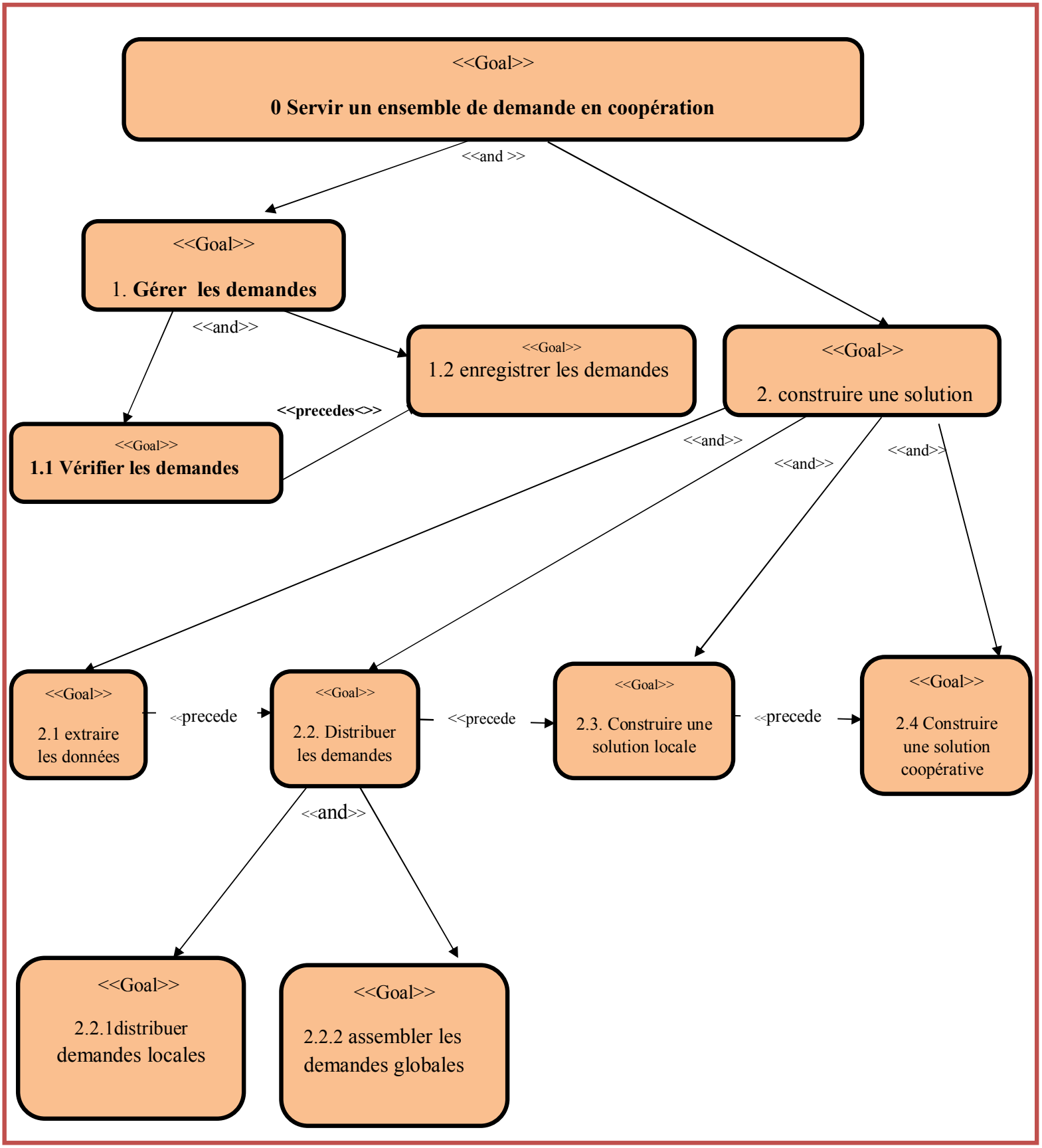

Figure 3.3 : Diagramme de but

4.2. Diagramme de rôle

Pour chaque sous-but (feuille) identifié précédemment, nous devrons créer un rôle permettant de le réaliser. Afin de réaliser un but, un rôle doit avoir à sa disposition une (ou plusieurs) «Capacités».

- Pour le But1.1 «**vérifier la validité des demandes** », nous créons le rôle «**vérificateur de validité** » requiert comme capacité « **règles de validation des demandes** ».
- Le But 1.2 «**enregistrer les demandes** » est réalisé par le rôle «**enregistreur des demandes**» qui requiert comme capacité «**les règles de manipulation d'une base de donnée ou des fichiers texte** ».
- Pour le But 2.1 «**extraire les données** », nous avons identifié le rôle «**extracteur des données »** qui requiert comme capacité «**les règles de manipulation d'une base de donnée ou des fichiers texte»**.
- Pour le But 2.2.1. «**Distribuer les demandes locales**», nous avons identifié le rôle «**distributeur des demandes**» qui requiert comme capacité «**les règles de communication avec les différentes agences ».**
- Pour le But 2.2.2. «**Assembler les demandes globales**», nous avons identifié le rôle «**assembleur des demandes**» qui requiert comme capacité «**les règles de communication avec les différentes agences ».**
- Le But.2.3 «**construire une solution locale** » est réalisé par le rôle «**optimiseur local »** qui dépend de la capacité «**algorithme d'optimisation locale**»
- Le But.2.4 «**construire une solution coopérative**» est réalisé par le rôle «**optimiseur coopératif**» qui dépend de la capacité «**algorithme d'optimisation coopérative** »

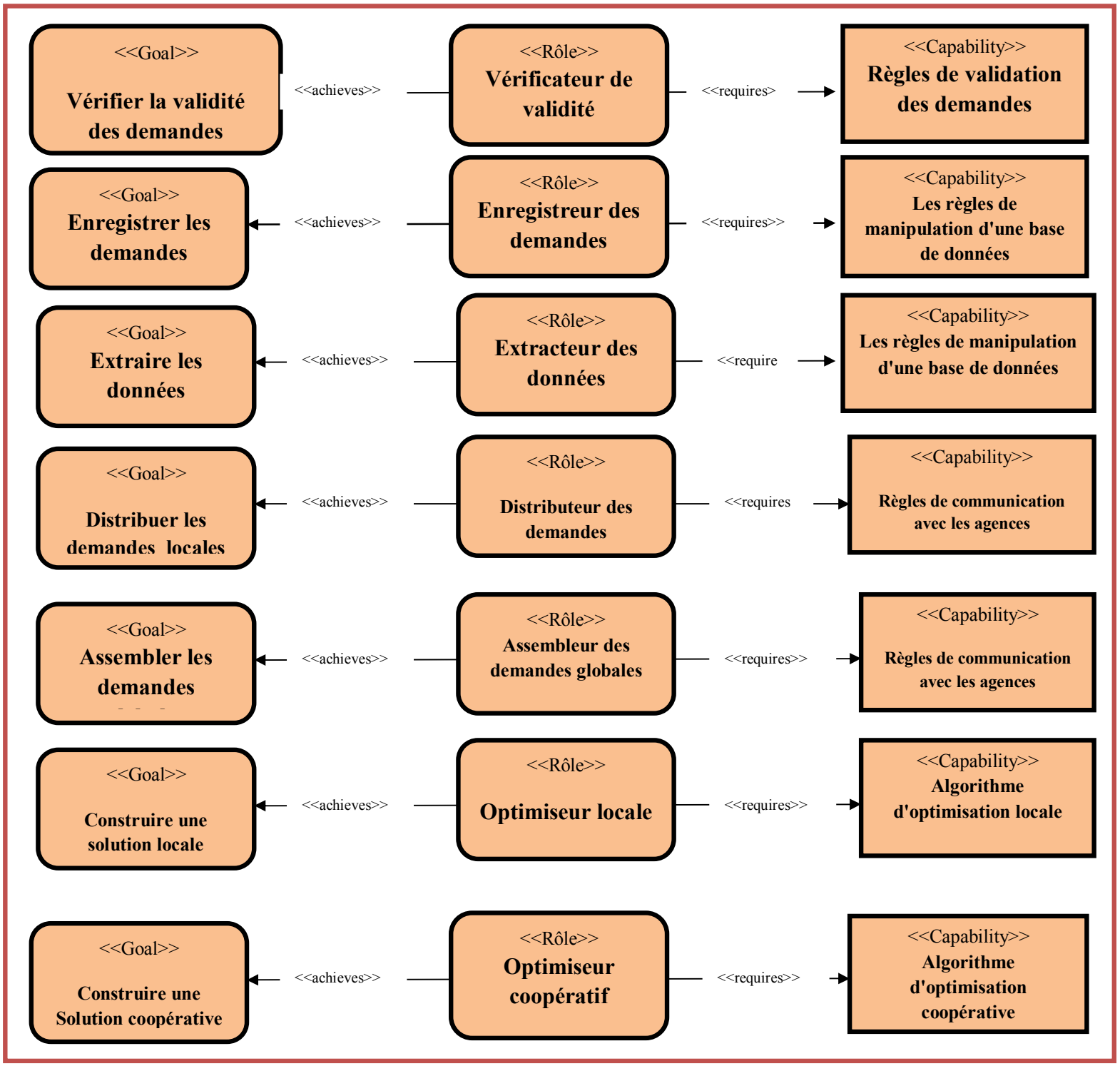

Figure 3.4 Diagramme de rôle

4.3. Diagramme d'agent

Le diagramme d'agents est créé pour qu'il y ait au moins une catégorie d'agent possédant toutes les capacités nécessaires pour jouer chaque rôle.

Chaque classe représente un modèle pour un type d'agent qui pourra être générer plusieurs fois selon les besoins du système. La philosophie de la méthodologie O-MaSe fournit une grande flexibilité de conception. En effet, il y aura un nombre minimal d'agent qui assurent le bon fonctionnement du système et changeront les rôles joués selon la situation. Ainsi, on assure une optimisation de l'utilisation des ressources mémoires. Par ailleurs, il est plus intéressant de séparer les tâches de calcul dans des rôles distincts.
Il est possible ainsi de paralléliser les calculs sur des agents différents. Il s'agit donc de trouver le meilleur compromis entre un nombre minimal d'agents et une bonne parallélisassions.

Pour le fonctionnement de notre système, il nous faut trois types d'agents qui fonctionnent simultanément (Figure 3.3):

1. Un premier agent nommé «**Interface** » joue les rôles :
- ❖ Vérificateur de validité,
- ❖ Enregistreur des demandes.

2. L'agent «**Responsable**» joue les rôles :
- ❖ Extracteur des données,
- ❖ Distributeur des demandes,
- ❖ Assembleur des demandes globales.
- ❖ Optimiseur coopératif.

3. Le troisième agent utilisé est l'agent «**Agence**», ayant comme rôles :
- ❖ **Optimiseur locale**

A partir du diagramme de la Figure 3.3, nous commençons à pouvoir visualiser la structure globale du SMA et ses interactions avec l'environnement extérieur.

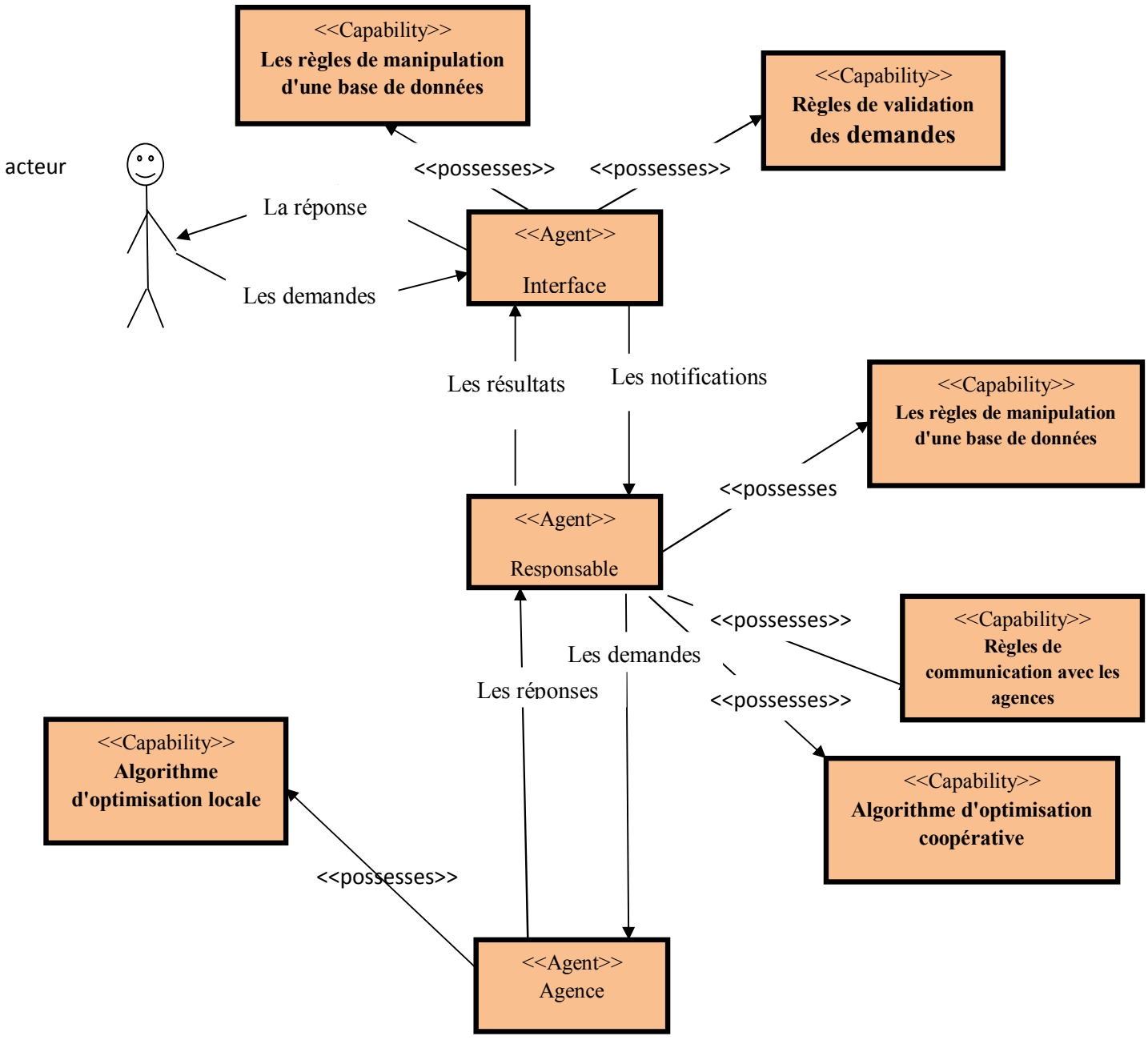

Figure 3.5 : Diagramme d'agent

4.4. Diagramme de protocole.

Dans ce dernier diagramme nous présentons les différentes interactions entre les agents de notre système multi-agents.

- l'agent interface débute l'interaction par l'envoi d'une notification à l'agent responsable pour informer ce dernier sur l'existence d'un nombre de requêtes à planifier.

- ❖ l'agent responsable récupère les demandes déjà stockées et il diffuse ces demandes pour attribuer chaque requête à une agence et en même temps il assemble les requêtes qui nécessitent une planification coopérative (partage d'une requête entre agences).
- ❖ après distribution des requêtes locales, chaque agent agence lance une optimisation locale (heuristique ILS).
- ❖ à la fin des optimisations locales chaque agence envoie son plan à l'agent responsable qui s'occupe de l'insertion des requêtes globales déjà assemblées en lançant une optimisation coopérative (heuristique de transfert).

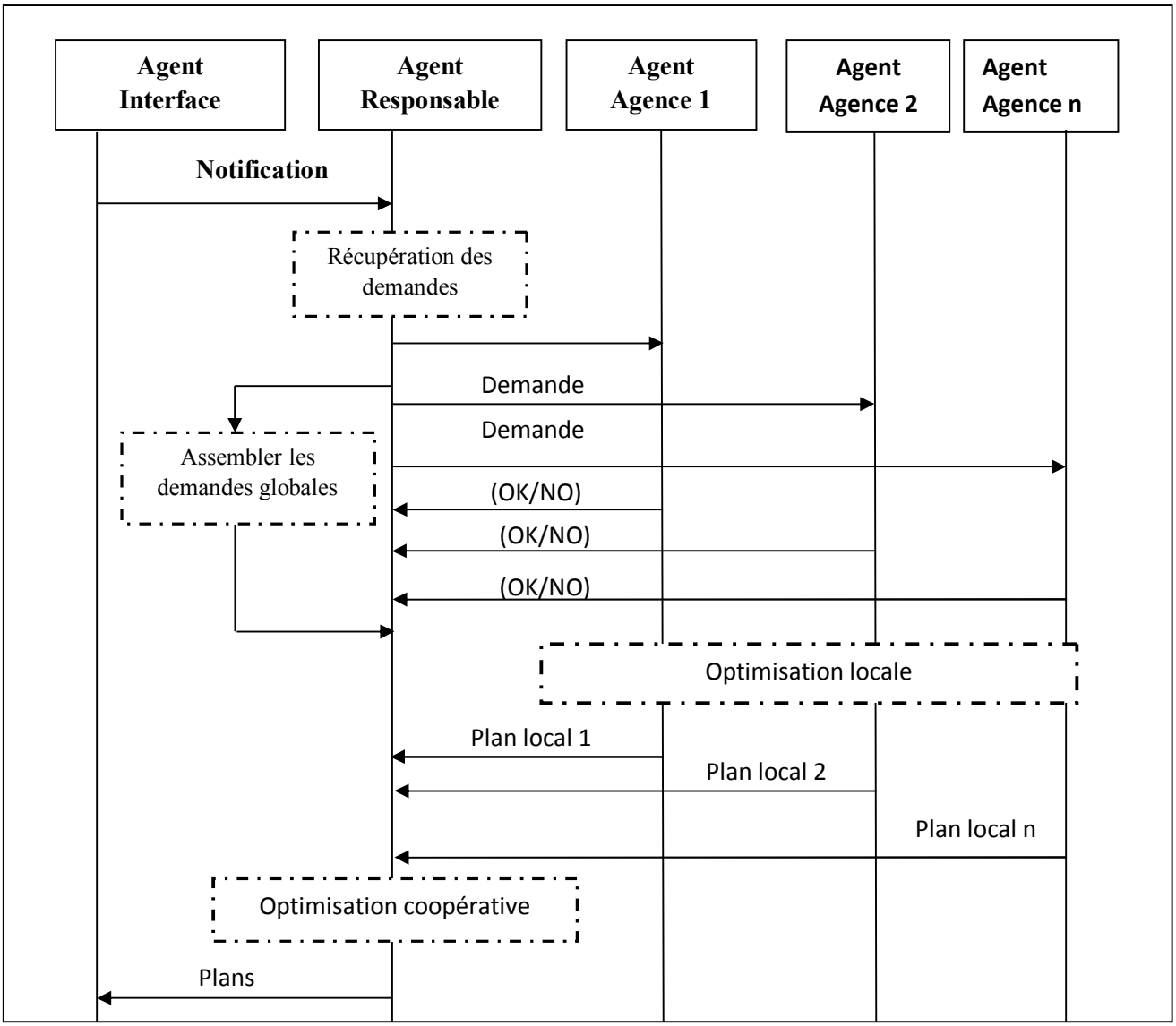

Figure 3.6 : Diagramme de protocole

5. Description des algorithmes utilisés

Le fonctionnement général de notre outil d'aide à la décision est assuré par deux heuristiques d'optimisation, dans la section suivante nous détaillons les principes et les étapes de ces deux dernières.

5.1. Heuristique d'optimisation locale

5.1.1. Principe :

Pour la construction des solutions locales chaque agent agence applique la méta-heuristique ILS comme méthodes d'optimisation afin de planifier ces propres demandes et générer les tournées. La recherche locale itérative ou encore Iterated local search (ILS) est une méta-heuristique simple. Son principe consiste à alterner une recherche locale afin d'obtenir un minimum local, et une procédure de perturbation afin de s'échapper de ce minimum. Lors de la recherche, la meilleure solution est mémorisée et sert de point de départ pour la procédure de perturbation.

5.1.2. Etapes et notation

1- la première étape de l'heuristique concerne la construction d'une solution initiale s_0 via un algorithme d'insertion (voir heuristique d'insertion).

2- la solution initiale s_0 est traitée par une recherche locale pour avoir un premier optimum local $s*$.

3- le reste des étapes concerne une boucle itérative conditionnée par un nombre d'itérations, dans cette boucle trois étapes sont alternées : une perturbation, une recherche locale et une instruction conditionnelle pour vérifier l'obtention d'une nouvelle solution améliorante.

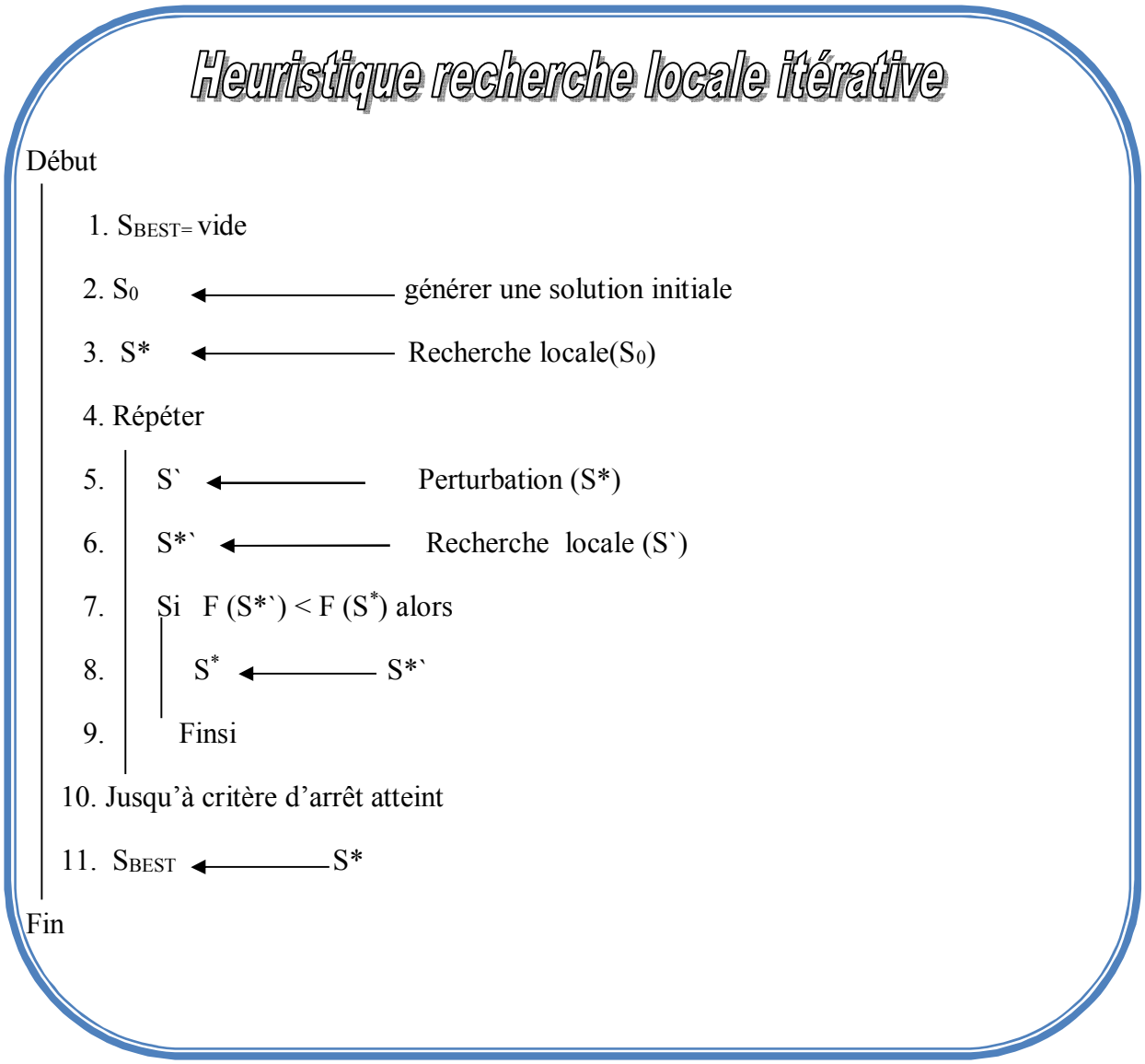

Figure 3.7 Heuristique recherche locale itérative

5.1.3. Construction de la solution initiale.

Une heuristique d'insertion est utilisée dans notre problème pour générer la solution initiale. Le principe général de cette heuristique est le suivant : nous créons premièrement des tournées vides à partir du nombre de véhicule disponible ensuite, à partir de la liste des demandes nous remplissons ces tournées aléatoirement.

Heuristique d'insertion

LD : liste de demandes,

S_0 : solution initial =<vide>

NV: nombre de véhicules disponible

Début

Pour i allant de 1 à NV

Créer une tournée contenant seulement les dépôts (tournée vide)

Finpour

Pour i allant de 1 à |LD|

Choisir aléatoirement une tournée j

Insérer la requête i dans la dernière position de la

tournée j de la solution S_0.

Finpour

Retourner S_0

Fin

Figure 3.8 : Heuristique d'insertion

5.1.4. Voisinage :

La recherche locale dans notre algorithme est basée sur une exploration du voisinage de la manière suivante : pour chaque tournée j et pour chaque demande i de cette tournée, la demande i est supprimée de sa position initiale et réinsérée dans les différentes positions dans sa propre tournée et dans les autres tournées afin de trouver une nouvelle position qui améliore le coût général.

Heuristique de Voisinage

```
S*: solution (initial ou après perturbation)
Début
    Pour i allant de 1 à |nombre de tournées dans S* |
        Pour j allant de 1 à |nombre de demandes dans la tournée i|
            Supprimer la demande j
            Chercher la première position améliorante (insertion de la demande j dans
            La même tournée de sa position initiale ou bien dans les autres tournées)
            si( position  améliorante trouvée )alors
                Garder la demande dans sa nouvelle place
            sinon
                Réinsérer la requête j dans sa position initiale
            Finsi
        Finpour
    Finpour
    Retourner  S*(après modification)
Fin
```

Figure 3.9 : Heuristique de voisinage

5.1.5. Perturbation :

Pour échapper des optimums locaux notre algorithme manipule une technique de perturbation simple :

- dans chaque itération nous choisissons aléatoirement deux tournées,
- dans ces deux tournées nous choisissons aussi aléatoirement deux demandes chacune d'une tournée.
- les deux requêtes sont permutées entre les deux tournées dans des positions aléatoires.

Heuristique de Perturbation

S^* : solution avant perturbation

Début

Choisir deux tournées aléatoirement i et j de S*

Choisir deux requêtes aléatoirement dans les deux tournées R_i et R_j

Retire les deux requêtes R_i et R_j

Insérer la requête R_i dans la tournée j dans une position aléatoire
Insérer la requête R_j dans la tournée i dans une position aléatoire

Retournée S*`

Fin

Figure 3.10 : Heuristique de perturbation

5.2. Heuristique de transfert

Pour simuler les coopérations entre les différentes agences nous utilisons une heuristique de transfert. Cette heuristique assure le découpage des requêtes globales (assemblées par l'agent responsable) qui nécessitent plusieurs agences (plusieurs véhicules) pour être planifier.

LDG: liste de demandes globales,

$SL=\{S_1,S_2,S_3,.....,S_n\}$ liste de solutions locales de chaque agence.(plan local)

Début

Pour i allant de 1 à |LDG|

- Trouver la liste d'agences qui peuvent coopérer dans la planification de la requête globale i.
- Découper la requête i en sous requêtes
- Insérer les sous requêtes dans les plans locaux des agences concernées par la coopération

Finpour

Retourné SL (après insertion des requêtes globales)

Fin

Figure 3.11 : Heuristique de transfert

6. Conclusion

Dans ce dernier chapitre nous avons présenté une description de notre outil d'aide à la décision et les méthodes utilisés pour la conception de ce dernier.

Dans le chapitre suivant, nous allons présenter l'environnement et les outils de développement utilisés ainsi que les interfaces graphiques de notre outil.

Chapitre 04

Implémentation et Teste

1. Introduction

Ce chapitre est consacré à la réalisation et la mise en œuvre de notre outil d'aide à la décision, nous allons présenter premièrement les outils de développement adoptés; à savoir le langage de programmation **Java**, l'environnement de développement **Netbeans** et la plateforme multi-agent **JADE**. Enfin nous montrons les principales interfaces et fenêtres de l'outil.

2. Langages et outils de développement

2.1. Le langage de programmation java

Java est un langage de programmation très utilisé, notamment par un grand nombre de développeurs professionnels, ce qui en fait un langage incontournable actuellement. Voici les caractéristiques de Java en quelques mots [12]

- Java est un langage de programmation moderne développé par Sun Microsystems aujourd'hui racheté par Oracle.
- Il ne faut surtout pas le confondre avec JavaScript (langage de script utilisé sur les sites Web), car ils n'ont rien à voir.
- Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc.

2.1.1. NetBeans

Pour la sélection de l'environnement de développement, notre choix s'est porté vers l'EDI Netbeans.

C'est un environnement de développement intégré (EDI). En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres (comme Python ou Ruby). Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, éditeur graphique d'interfaces et de pages Web, etc.)

2.2. Plateformes de développement des SMA :

A fin de réaliser une opérationnalisation plus accessible des systèmes multi-agents, des travaux ont tenté de réutiliser des architectures et des langages existants pour construire des environnements de développement de ces systèmes. Les environnements de développement ou les plateformes multi-agents sont nécessaires pour renforcer le succès de la technologie multi-agents.

Les plates-formes multi-agents permettent aux développeurs de concevoir et de réaliser leurs applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi-agents. Parmi les plateformes les plus connues on peut citer Swarm, MadKit et JADE, etc.

2.2.1. La plateforme SWARM

S W ARM est une plateforme multi-agents avec agents réactifs.

L'inspiration du modèle d'agent utilisé vient de la vie artificielle. SWARM est l'outil privilégié de la communauté américaine et des chercheurs en vie artificielle.

L'environnement offre un ensemble de bibliothèques qui permettent l'implémentation des systèmes multi-agents avec un grand nombre d'agents simples qui interagissent dans le même environnement. De nombreuses applications ont été développées à partir de SWARM qui existe aujourd'hui implémenter en plusieurs langages (Java, Objective-C).

2.2.2. La plateforme MADKIT

MADKIT (Madkit, 2003) est une plate-forme développée par le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. MADKIT est libre pour l'utilisation dans l'éducation. Il est écrit en Java et est fondé sur le modèle organisationnel Alaadin. Il utilise un moteur d'exécution où chaque agent est construit en partant d'un micronoyau. Chaque agent a un rôle et peut appartenir à un groupe. Il y a un environnement de développement graphique qui permet facilement la construction des applications.

2.2.3. La plate-forme JADE

JADE (Java Agent Development Framework - Bellifemine, Poggi, Rimassa, 1999) est une plateforme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (FIPA, 1997).

JADE comprend deux composantes de base : une plateforme agents compatible FIPA et un paquet logiciel pour le développement des agents Java.

Le but de JADE est de simplifier le développement des systèmes multi-agents en conformité avec la norme FIPA pour réaliser des systèmes multi-agents interopérables.

2.2.3.1. Les principaux composants de Jade.

- **Agent RMA :** (*Remote Management Agent) :*

Le **RMA** permet de contrôler le cycle de vie de la plate-forme e. L'architecture répartie de JADE permet le contrôle à distance d'une autre plateforme. Plusieurs RMA peuvent être lancés sur la même plateforme du moment qu'ils ont des noms distincts.

- **Agent Dummy.**

L'outil Dummy Agent permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

- **Agent Directory Facilitator.**

L'interface du DF peut être lancée à partir du menu du RMA .Cette action est en fait implantée par l'envoi d'un message ACL au DF lui demandant de charger son interface graphique. L'interface peut être juste vue sur l'hôte où la plate-forme est exécutée. En utilisant cette interface, l'utilisateur peut interagir avec le DF.

- **Agent Sniffer.**

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du RMA ou de la ligne de commande suivante : *Java jade.Boot sniffer:jade.tools.sniffer.sniffer.*

- **Agent Inspector :**

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et la file de ses messages envoyés et reçus.

2.2.3.2.Structure et déclaration des agents dans Jade

Un agent dans la plateforme Jade est une classe qui hérite de « jade.core.Agent ». Cette classe doit nécessairement contenir une méthode spéciale « *setup ( )* » qui constitue la partie initialisation de l'agent. Le fragment de code java suivant est un exemple de déclaration d'un agent sur plateforme jade [14].

```
import jade.core.Agent ;
public class monAgent extends Agent {
protected void setup()
  {
  // initialization de l'agent
    System.out.println( "MonAgent est initialisé"+getAID().getName()+" is ready.") ;
  }
}
```

Figure 4.1 Déclaration d'un agent dans JADE

- **La méthode « getAID ( ) »**

Est une méthode pré-implémentée dans jade.core. Elle permet d'obtenir l'identifiant de l'agent qui est une instance de la classe jade.core.AID. Cet identifiant contient le nom unique qui référence l'agent.

Ce sont les « Behaviour » (comportements) qui permettent aux agents d'agir dans une plateforme. Un comportement est une tâche qu'un agent doit réaliser à un moment précis et selon des contraintes. « Behaviour » sont implémentés comme des instances d'une classe qui hérite de «jade.core.behaviours.Behaviour» [14].

Chaque «Behaviour» contient au moins deux méthodes :

- ✓ La méthode « *action ( )* » : qui définit les actions à réaliser quand ce «Behaviour» est invoqué.
- ✓ La méthode « *done( )* » : retourne un booléen qui indique si le «Behaviour» a terminé son exécution ou non.

Il existe deux principaux types de « Behaviours »

- **One-shot behaviours**

Les actions de ce « Behaviour » s'exécutent une seule fois. Ce « Behaviour » se termine immédiatement après l'appel de sa méthode « *action( )* ». La méthode « *done( )* » est déjà implémentée dans le jade.core.behaviours.OneShotBehaviour.

Le fragment de code suivant est un exemple d'implémentation d'un « One-shot Behaviour »

```
import jade.core.behaviours.OneShotBehaviour
public class MyOneShotBehaviour extends OneShotBehaviour
        {
         public void action()
         {
           // Instructions à réaliser qu'une seul fois pour ce behaviour
         }
        }
```

Figure 4.2 Un exemple de code d'implémentation du « OneShotBehaviour »

- **Cyclic behaviours**

Ce « Behaviour » ne se termine jamais, il continue à s'exécuter en boucle jusqu'à ce que l'agent meurt. Ces « comportements » sont généralement utilisés de la manière suivante : un « CyclicBehaviour » qui joue le rôle d'un serveur. Il boucle en vérifiant les messages reçus, vérifie leurs formes et leurs destinataires. S'ils répondent aux critères, cela signifie qu'il est bien compétent à les gérer. Il les traite donc et renvoie une réponse à leur expéditeur. De l'autre côté, un « OneShotBehaviour » s'exécute sous certaines conditions en envoyant un message vers un « CyclicBehaviour » en attente.

Il faut aussi signaler que si un agent n'a pas de comportement à exécuter, le thread qui gère l'agent en question s'endort afin de ne pas consommer du temps du processeur. Il est réveillé dès qu'il y a un « Behaviour » dans sa liste d'attente [14].

```
import jade.core.behaviours.CyclicBehaviour
public class MyCyclicBehaviour extends CyclicBehaviour
 {
  public void action()
  {
    //Tant que l'agent est vivant, les instructions qui se trouvent ici seront exécutées
   }
}
```

Figure 4.3 Un exemple de code d'implémentation du « CyclicBehaviour »

3. Présentation d'outil développée

3.1. Choix techniques

Notre choix s'est porté sur la plateforme de développement des SMA : JADE pour les raisons suivantes :

- Il est simple de créer des agents avec jade.
- Jade gère la communication entre les agents et offre des interfaces de gestion des agents.
- Le mode de communication direct (avec des messages) est supporté par la plateforme JADE en utilisant le langage FIPA ACL.
- La plateforme Jade répond à plusieurs fonctionnalités et offre une large gamme de bibliothèques.
- Les agents développés en JADE sont écrits totalement en JAVA qui est un langage facile et basé sur la notion d'objet.
- JAVA a été choisie pour ses riches outils facilitant son utilisation.

3.2. fichier de demande :

Pour tester l'efficacité de notre outil d'aide à la décision nous avons utilisés un fichier de demande standard qui a la structure suivante :

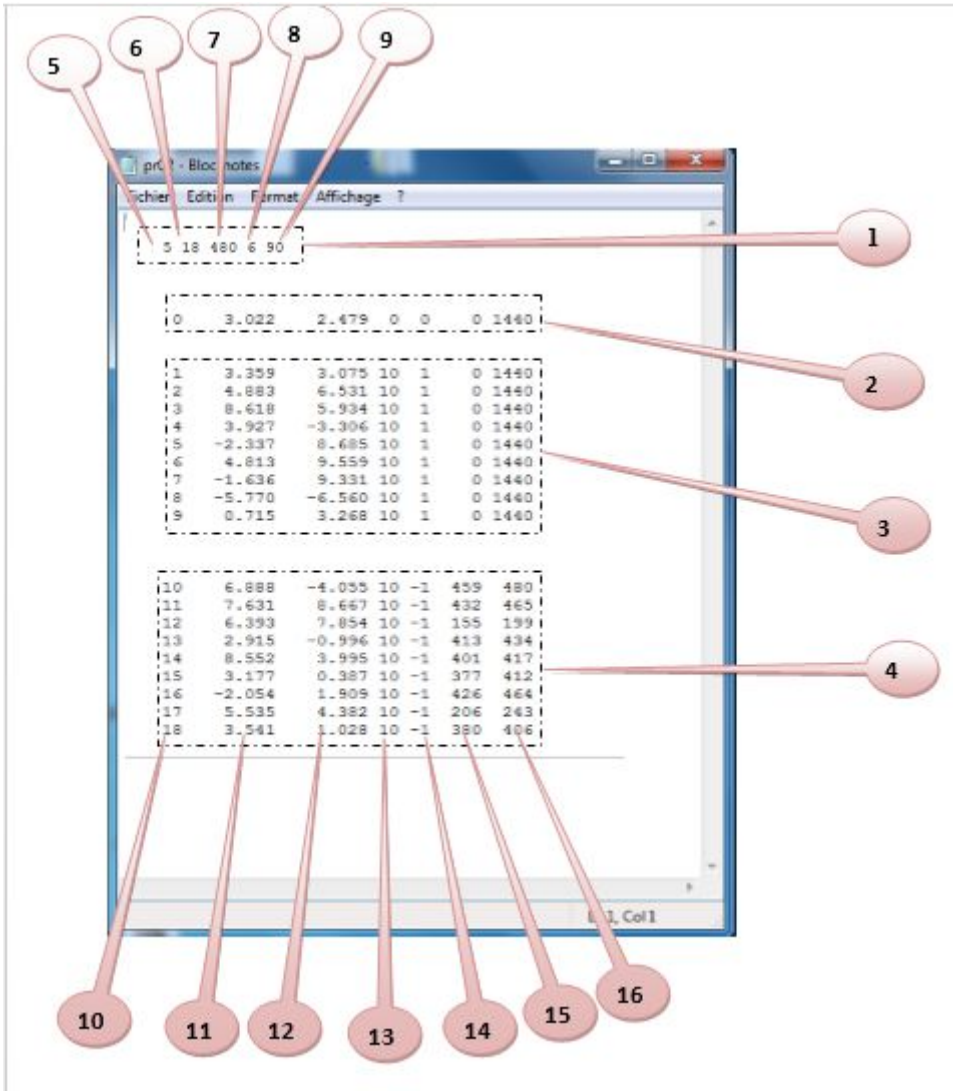

Figure 4.4 la structure du fichier de demande

Ce fichier est divisé en quatre parties : 1, 2,3 et 4 :

1 ⟶ La première partie compose d'une seule ligne et cinq colonnes 5, 6, 7,8et 9 :

- 5⟶ Le nombre de véhicules d'un opérateur de transport à la demande.
- 6⟶ Le nombre de nœuds visités.
- 7⟶ La durée maximale de chaque tournée.
- 8⟶ Le nombre de place dans chaque véhicule.
- 9⟶ Le temps maximal pour transporter un client de la source à la destination.

2 ⟶ La deuxième partie contient les informations du dépôt.

3 ⟶ La troisiéme partie contenant les informations nécessaires des nœuds sources de tous les clients qui ont demandé s le service de transport .

4 ⟶ La quatriéme partie contenant les informations nécessaires des nœuds destinations de tous les clients qui ont demandés le service de transport .

- Les trois derniéres parties ayant la même structure ,ellle sont composées de sept colonnes :

  10 ⟶ l'identité de chaque nœud :

  - Commence par 0.
  - Termine par le nombre de nœuds déclarés dans la première partie.

  11 et 12⟶ sont les coordonnées de chaque nœud : X et Y

  13 ⟶ Le temps de service de chaque nœud.

  14 ⟶ L'état de chaque nœud : on distingue 03 cas 0,1 et -1 :

  - Si 0 c'est le dépôt.
  - Si 1 c'est la source.
  - Sinon c'est la destination.

  15 et 16⟶ la fenêtre de temps de chaque nœud ou :

  - 15⟶ le temps au plus tôt.
  - 16⟶ le temps au plus tard.

- Remarque :

La troisiéme et La quatriéme partie ayons le même nombre de lignes : chaque source doit avoir une destination .

3.3. Les interfaces de notre outil :

Dans cette partie nous allons présenter les interfaces principales de notre outil

- **L'interface d'Accueil**

Cette interface permet au gestionnaire d'accéder à l'outil pour l'exploitation des services offerts.

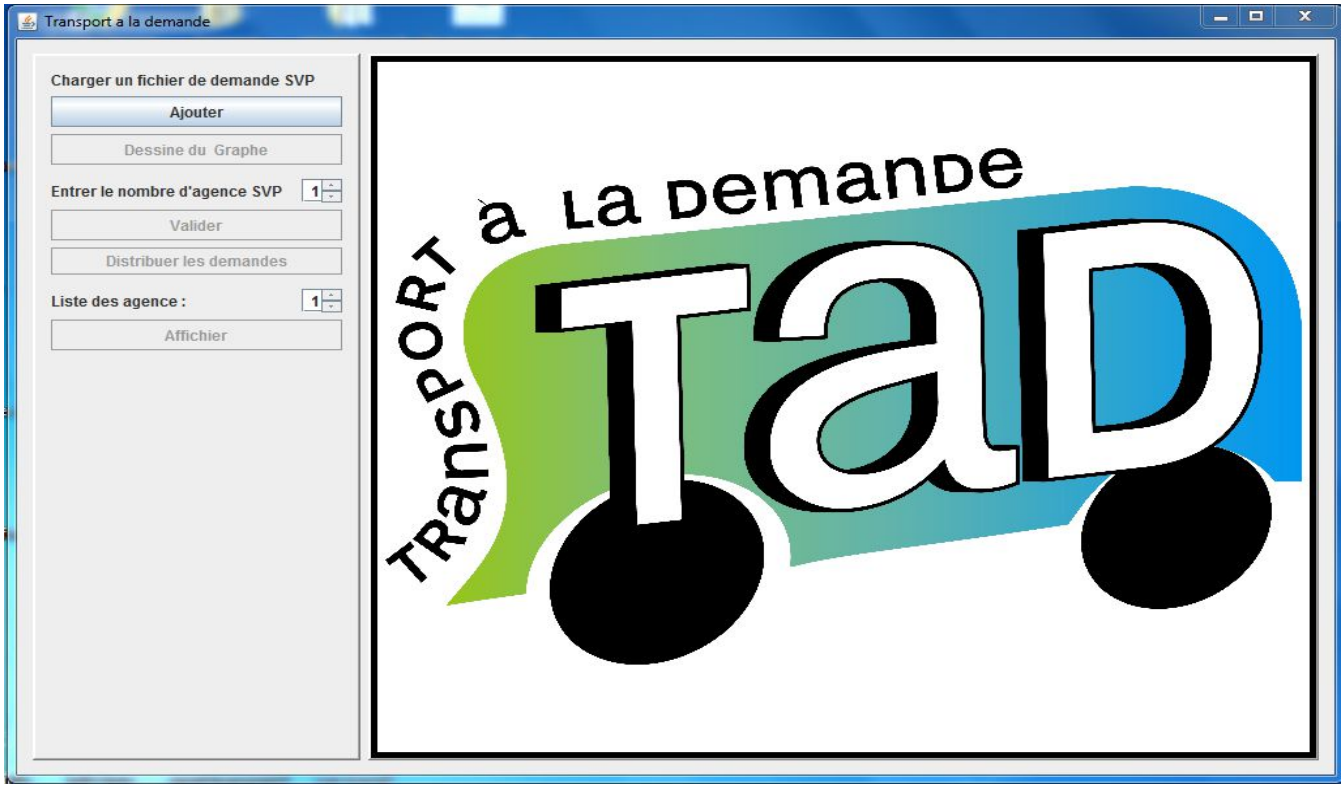

Figure 4.5 Fenêtre « Accueil »

Cette fenêtre contient les fonctionnalités suivantes :

- **Charger un fichier de demande** :

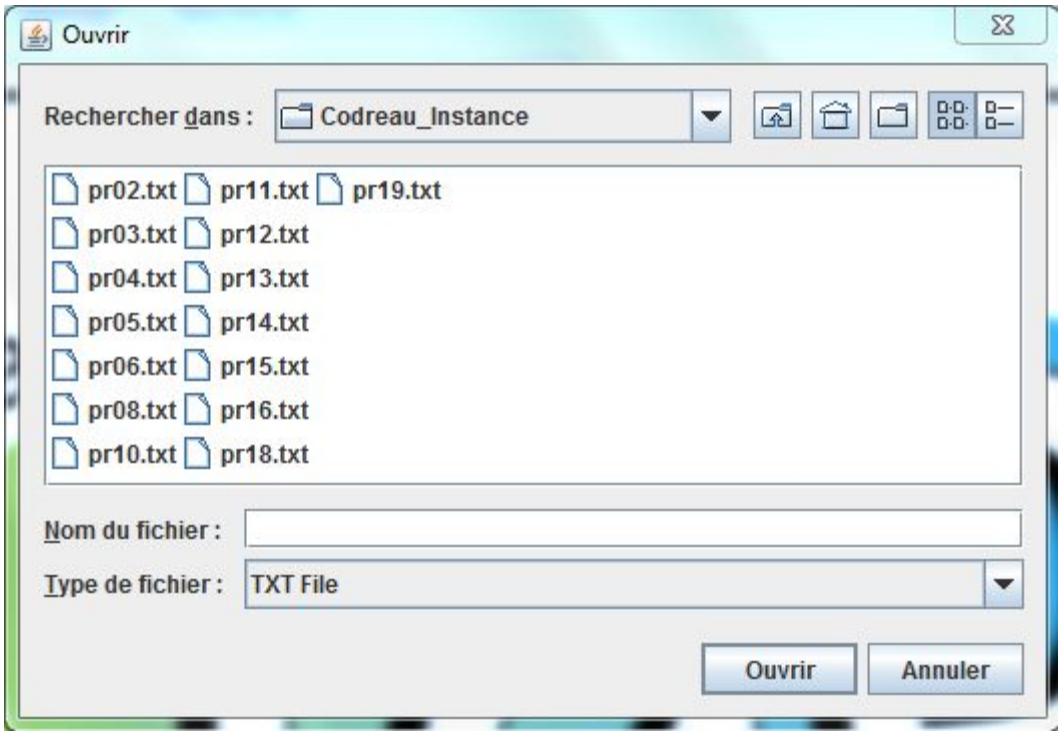

Figure 4.6 Fenêtre « Chargement d'un fichier de demande »

- **Dessiner le graphe** :

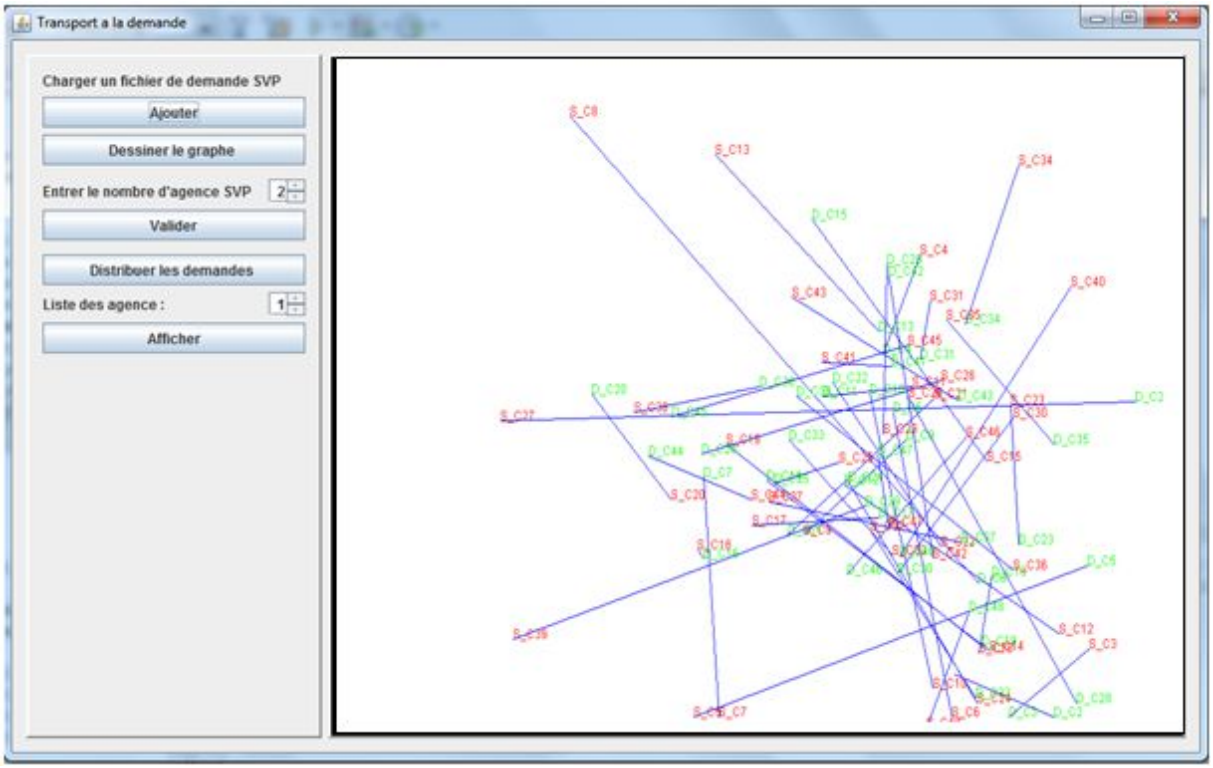

Figure 4.7 Fenêtre « dessine du graphe »

- **Afficher les interfaces des agences :**

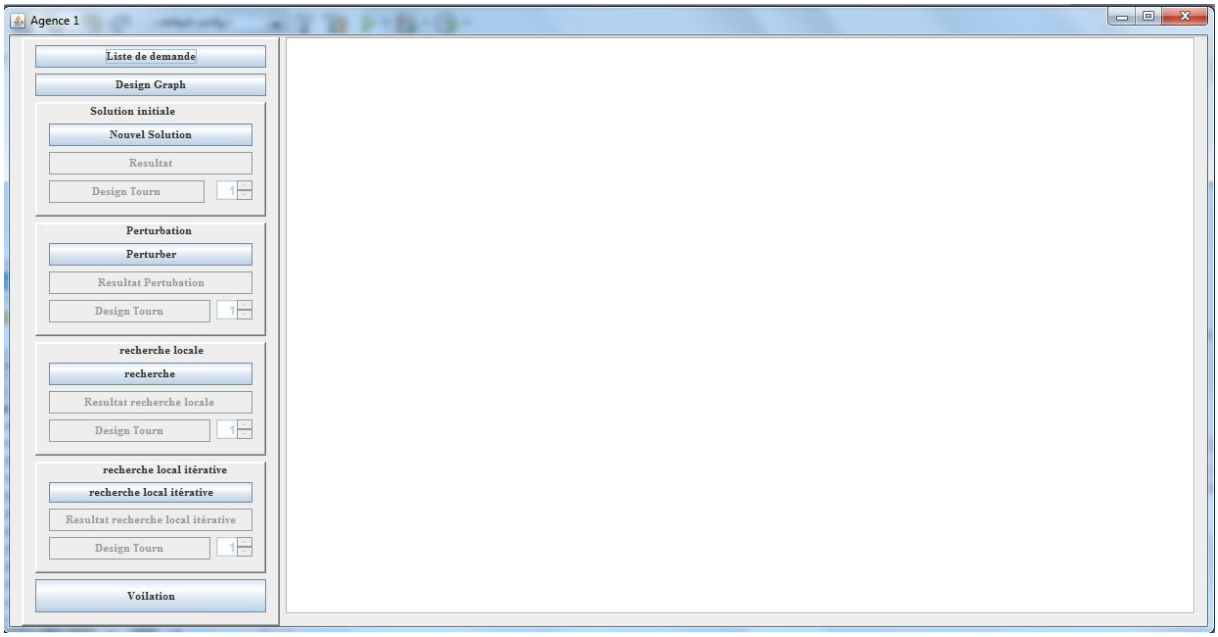

Figure 4.8 Fenêtre « interfaces des agences»

- **L'interface d'agence**

Cette interface contient les fonctionnalités suivantes :

- **Dessiner le graphe des demandes affectées à une agence**

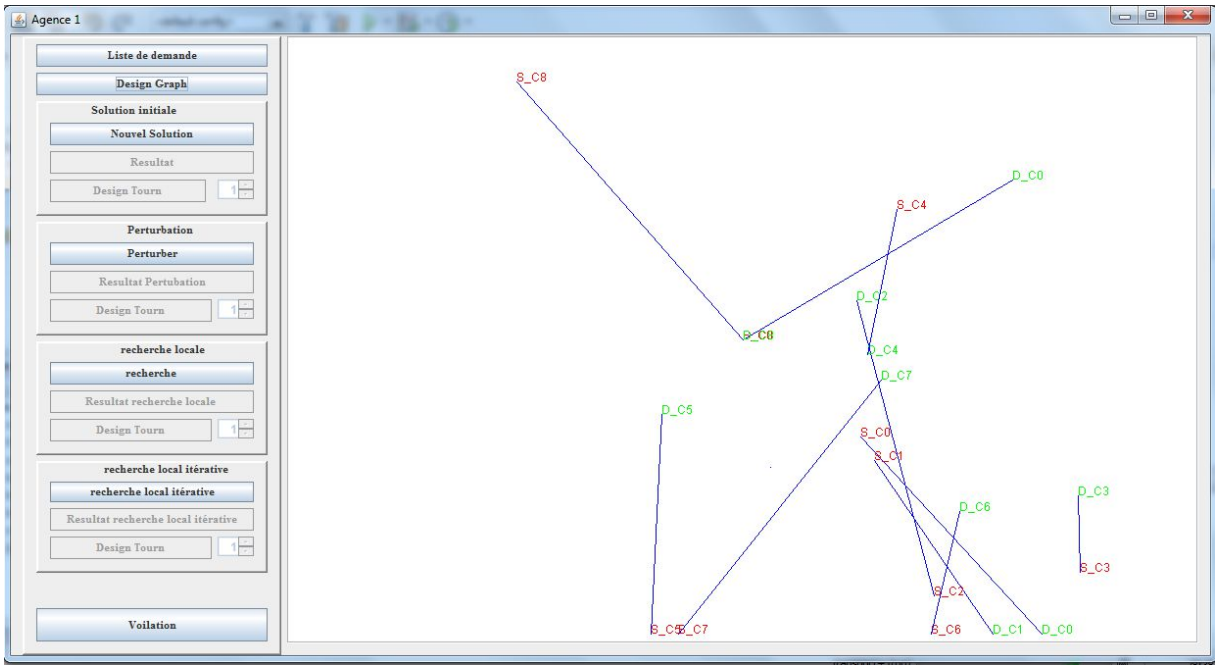

Figure 4.9 Fenêtre « dessin du graphe »

- **Générer une solution initiale :**

Cette fenêtre permet d'afficher la solution initiale

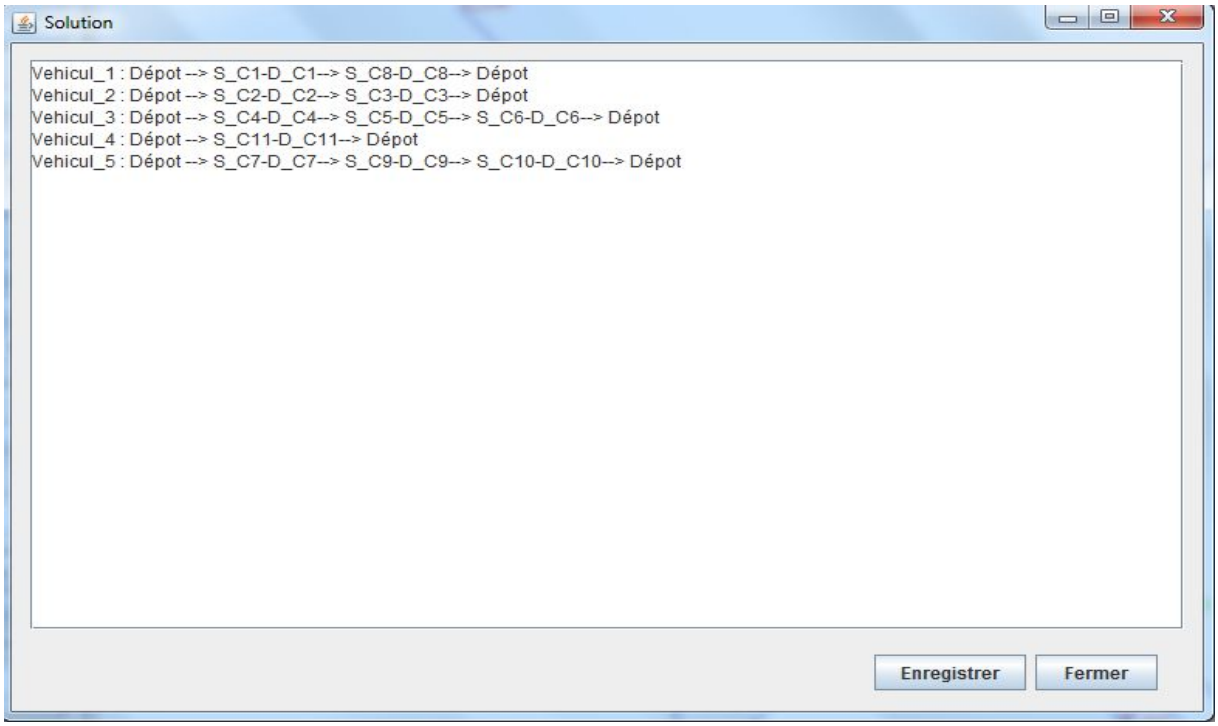

Figure 4.10 Fenêtre « génération d'une solution initiale »

- **Dessiner le graphe de chaque tournée de la solution initiale :**

Cette fenêtre permet de choisir une tournée pour dessiner son graphe

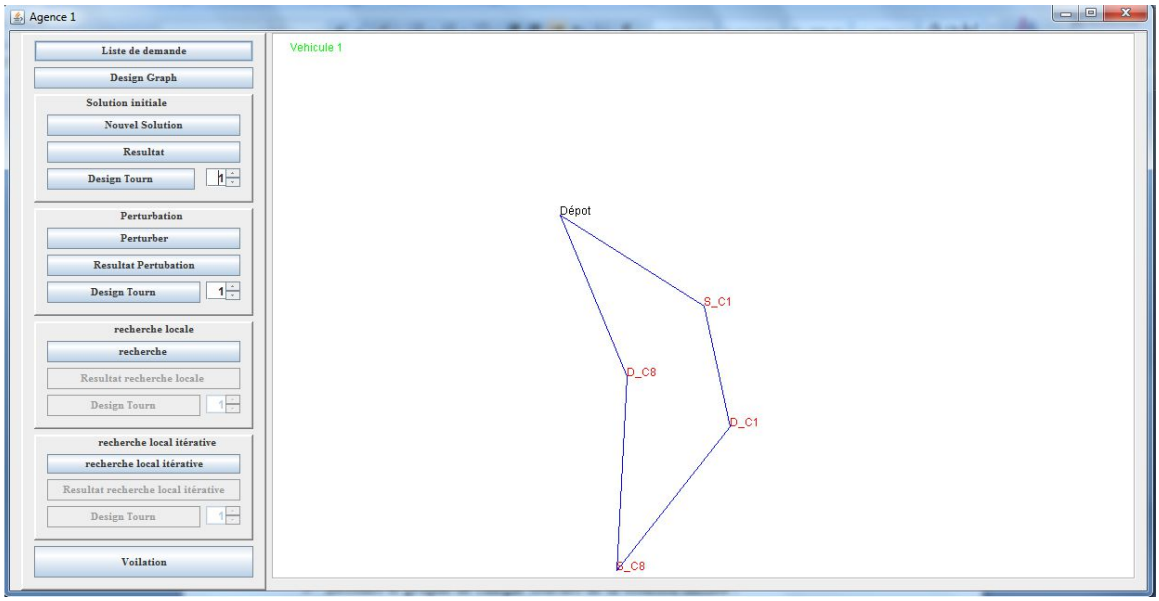

Figure 4.11 Fenêtre « le dessin du graphe de chaque tournée de la solution initiale »

- **Appliquer une perturbation :**

❖ **Le choix :**

Cette fenêtre permet au gestionnaire de choisir le fichier ou la solution qui existe déjà pour appliquer la perturbation.

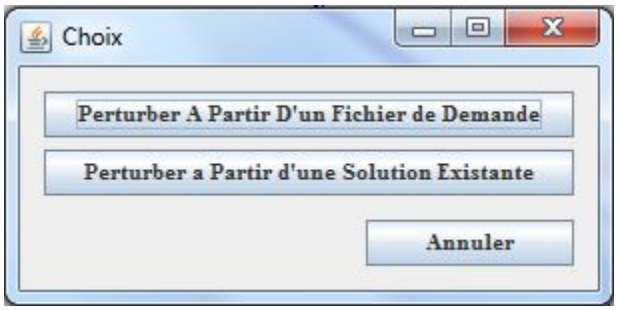

Figure 4.12 Fenêtre « Choix »

❖ **Le résultat de la perturbation :**

Cette fenêtre affiche le résultat de la perturbation

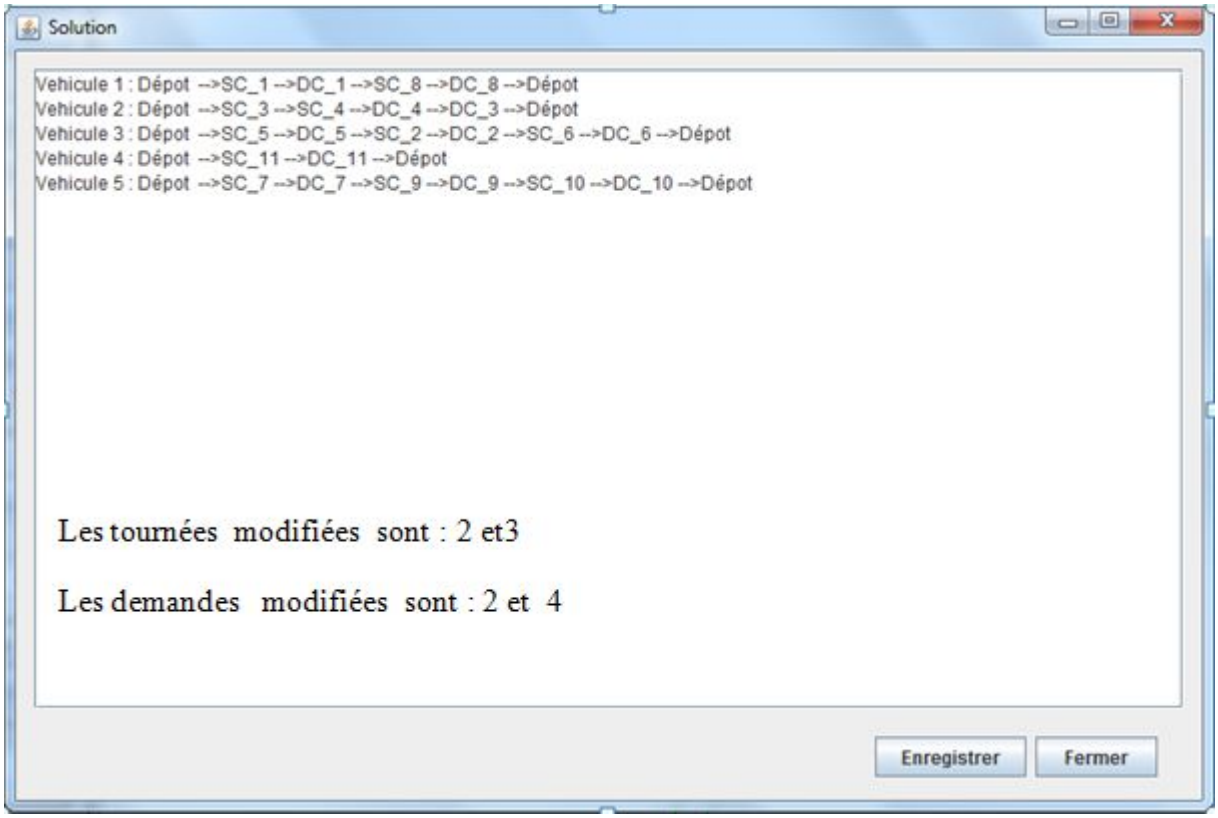

Figure 4.13 Fenêtre « résultat de la perturbation »

➢ **Dessiner le graphe de chaque tournée de la solution perturbée**

Cette fenêtre permet de choisir une tournée pour dessiner son graphe

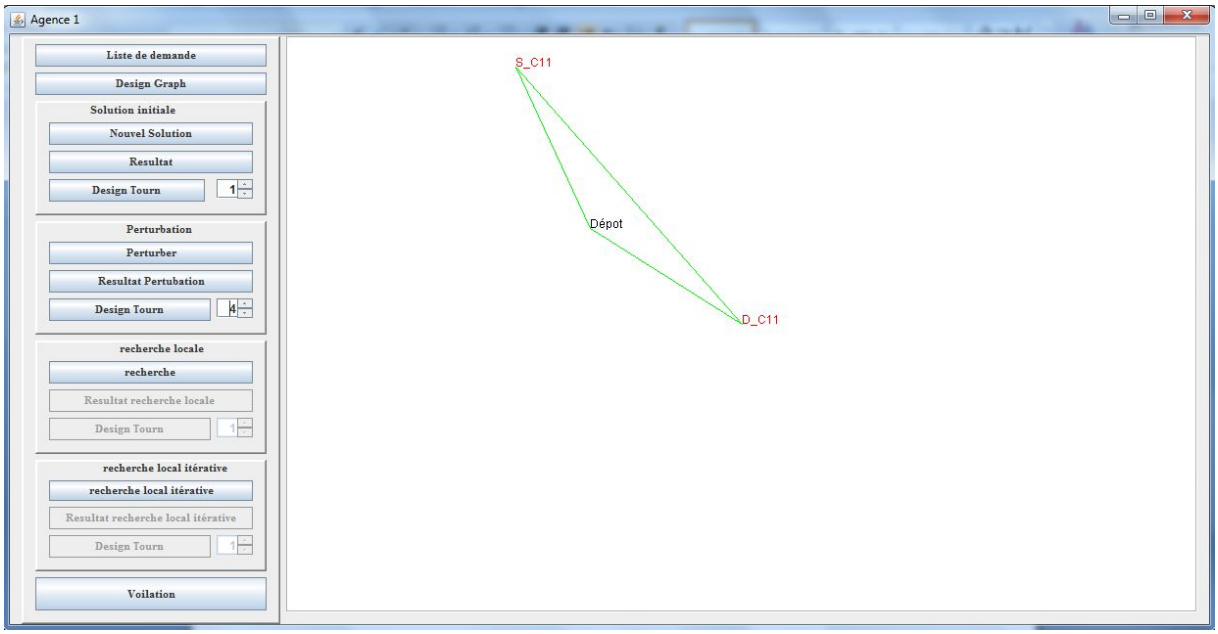

Figure 4.14 Fenêtre « dessin du graphe de chaque tournée de la solution perturbée»

- **Appliquer une recherche locale:**
  - **Le choix :**

Cette fenêtre permet au gestionnaire de choisir le fichier ou la solution qui existe déjà pour appliquer la recherche locale.

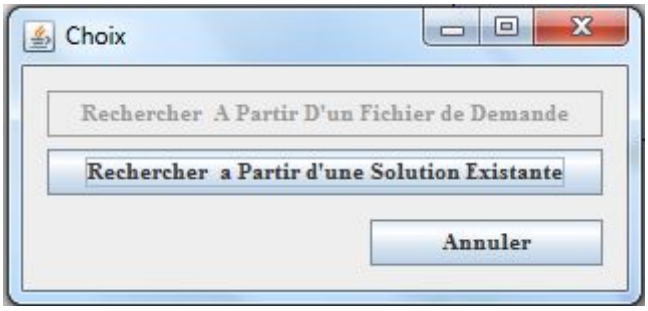

Figure 4.15 Fenêtre « Choix »

- **Le résultat de la recherche locale :**

Cette fenêtre affiche le résultat de recherche locale

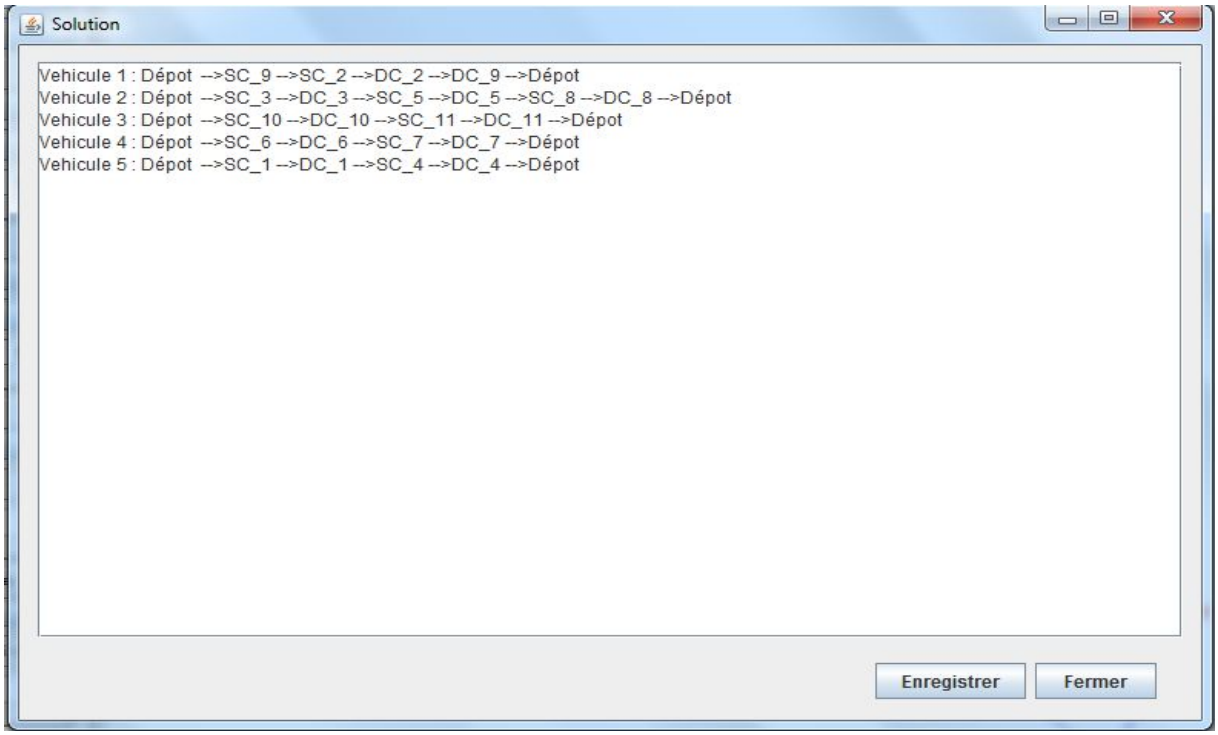

Figure 4.16 Fenêtre « résultat de la recherche locale »

- **Dessiner le graphe de chaque tournée de la recherche locale**

Cette fenêtre permet de choisir une tournée pour dessiner son graphe

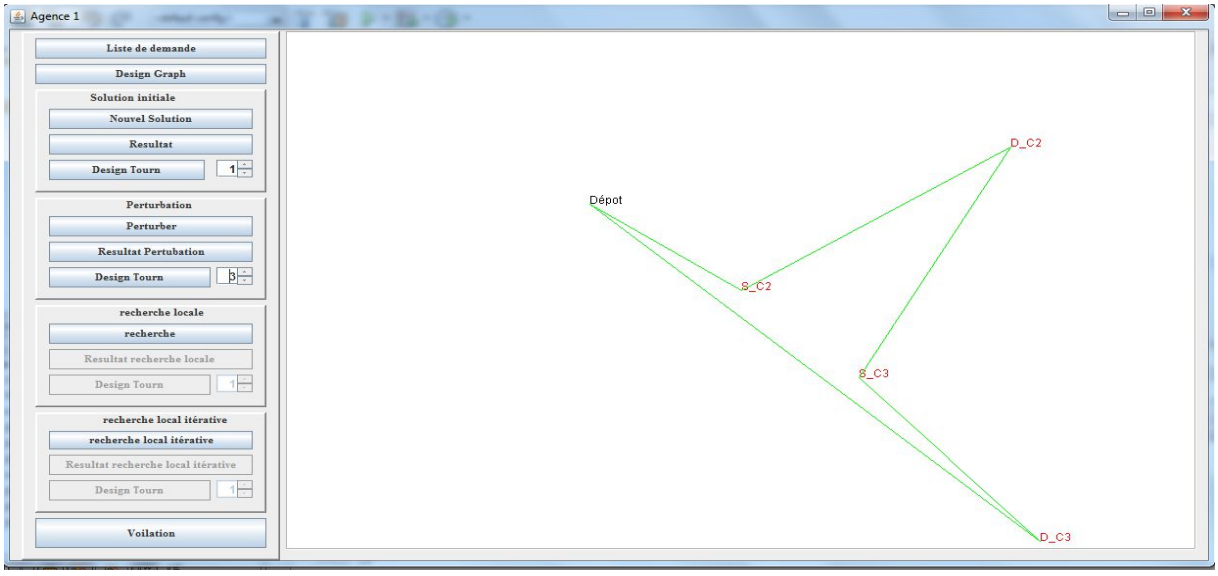

Figure 4.17 Fenêtre « le graphe de chaque tournée de la solution locale »

- **Appliquer une recherche locale itérative :**
- **Le résultat de la recherche locale itérative :**

Cette fenêtre affiche le résultat de recherche locale

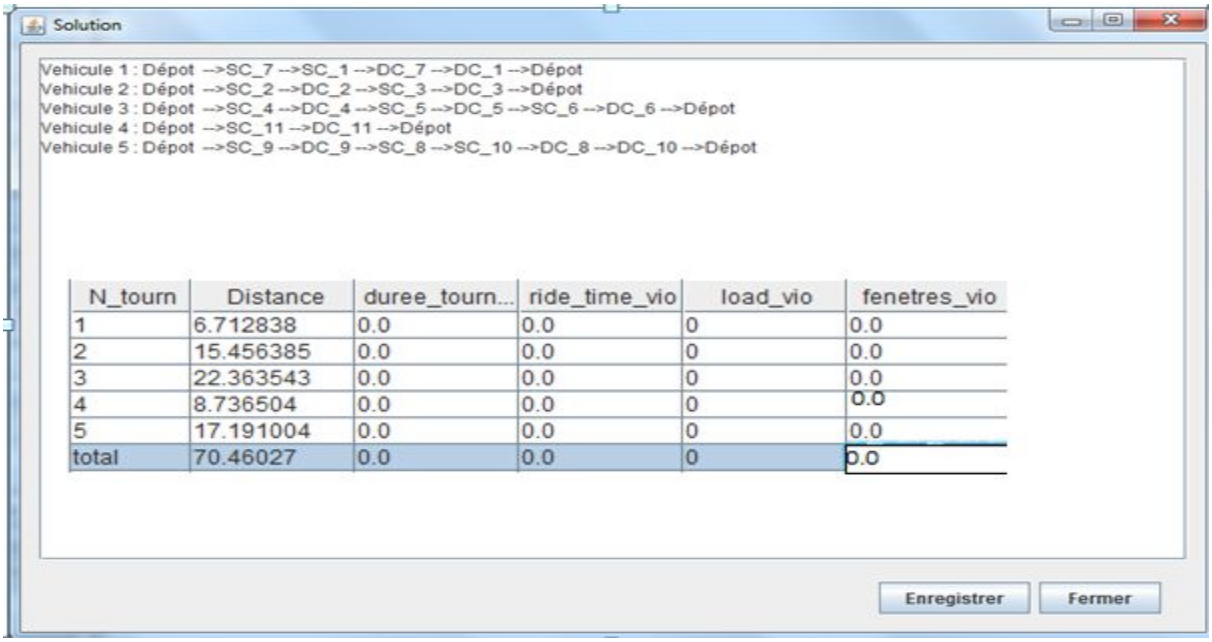

Figure 4.18 Fenêtre « le résultat de la recherche locale itérative »

- **Dessiner le graphe de chaque tournée de la recherche locale itérative**

Cette fenêtre permet de choisir une tournée pour dessiner son graphe

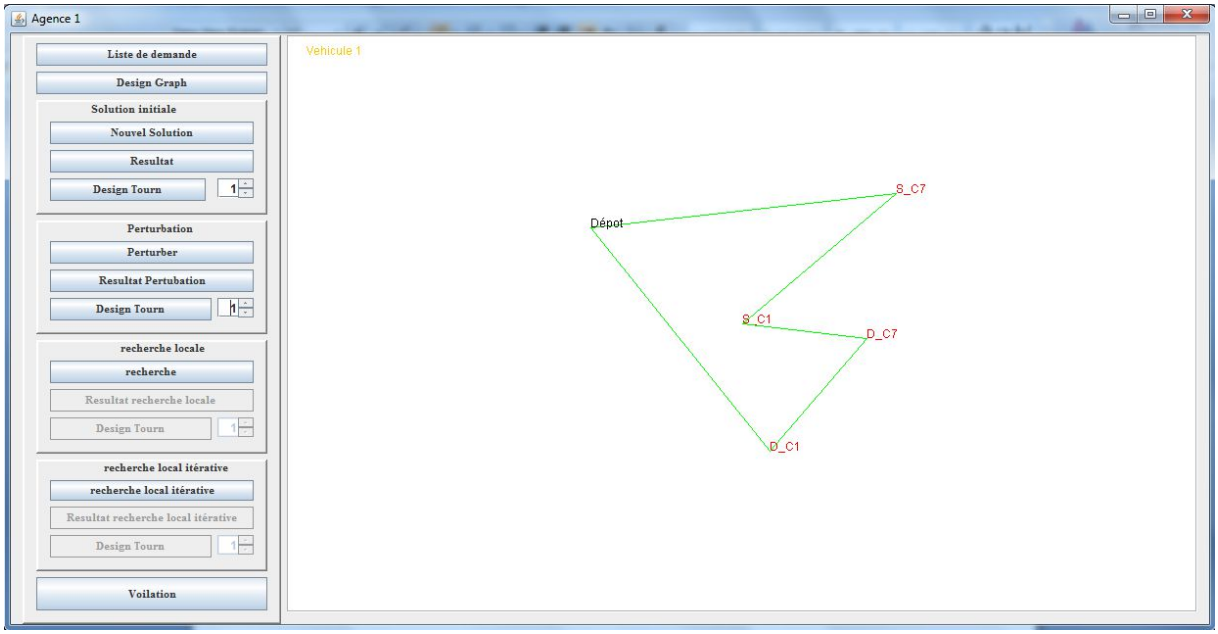

Figure 4.19 Fenêtre « le dessin du graphe de chaque tournée de la recherche locale itérative»

- **Afficher les violations :**

Cette fenêtre permet de calculer les violations d'une solution donnée.

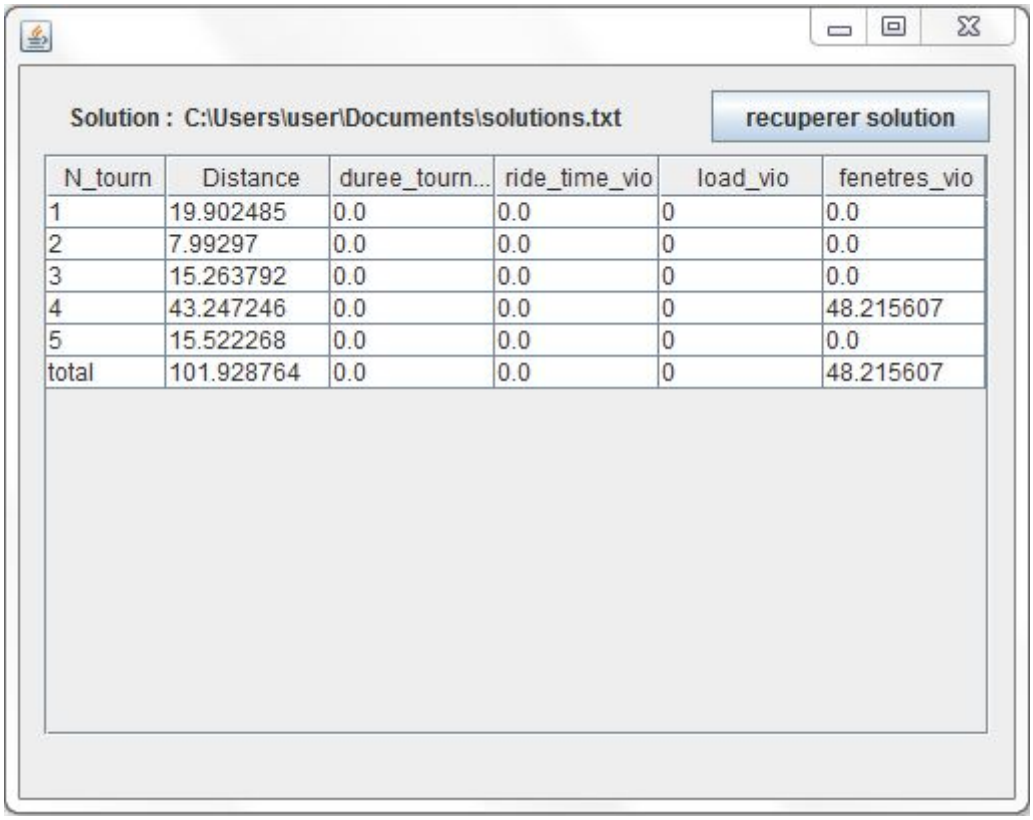

Figure 4.20 Fenêtre « d'affichage des violations »

4. Conclusion

Dans ce dernier chapitre nous avons présenté tout ce qui est lié au développement de notre outil. Nous avons présenté une brève description des différentes plateformes utilisées dans le

domaine de développement des SMA avec une justification de notre choix concernant la plateforme JADE. Le chapitre est clôturé par une description détaillée des interfaces graphiques de notre outil d'aide à la décision.

Conclusion générale

Les outils d'aides à la décision sont devenus de nos jours un moyen incontournable et même Indispensable pour améliorer les rendements dans les entreprises. Ils sont rapides, pratiques et ils répondent parfaitement aux différents besoins des individus décideurs dans différents domaines.

L'objectif de notre travail était de réaliser un outil d'aide à la décision pour assister les gestionnaires d'une agence de transport à la demande.

Avant d'aborder la réalisation de notre outil, nous avons présenté une étude bibliographique relative au domaine du TAD. Nous avons présenté aussi Les principes des différentes méthodes d'optimisation existantes et les principes liées au domaine d'aide à la décision Ensuite, nous avons présenté notre outil d'aide à la décision qui assure deux fonctionnalités: (1) l'optimisation des tournées et (2) la simulation d'un cas de coopération original entre plusieurs agence de transport à la demande

Pour l'optimisation des tournées nous avons utilisé l'heuristique de la recherche locale itérative (ILS) caractérisée par la simplicité.

Pour la simulation des cas de coopération nous avons adopté le paradigme multi-agent car ce dernier facilite la modélisation du problème étudié d'une façon naturelle et claire.

Pour la réalisation, nous avons utilisé JAVA comme langage de programmation et JADE comme plateforme d'implémentation des différents agents.

Le plus grand bénéfice tiré de cette expérience est que nous avons eu la chance d'enrichir nos connaissances dans des domaines variés comme : le Transport A la Demande, l'Aide A la Décision, la méthode O-MASE, le langage JAVA et la plate forme JADE …

Pour l'outil réalisé et bien que nous avons pu couvrir les objective initial de notre projet, il y a beaucoup de fonctionnalités que nous aimons intégrer mais les délais nous ont empêchés de s'approfondir plus dans l'amélioration de notre outil.

Annexe

O-MaSE : Une approche pour la conception des systèmes multi-agents complexes.

1. Introduction

Aujourd'hui l'industrie du logiciel est chargée de la construction des applications logicielles de plus en plus complexe. Bien que les méthodes de développement de logiciels ont fait de grands progrès au cours des trente dernières années aujourd'hui les entreprises exigent des applications qui fonctionnent de façon autonome, adapté à des environnements dynamiques et capable d'interagir avec d'autres applications distribuées afin de fournir des solutions de grande envergure. La technologie des systèmes Multiagents (MAS) est une approche prometteuse capable de répondre à ces nouvelles demandes.

Pour l'analyse et la conception des applications à base d'agents plusieurs paradigmes et méthodes sont développés afin d'assurer une conception juste et efficace, parmi ces méthodes nous présentons dans cette annexe la méthode O-MASE [30].

2. O-MASE : Organization-based Multiagent Software Engineering

C'est une méthodologie qui intègre un ensemble de technologies concrètes visant à faciliter l'acceptation industrielle. Plus précisément, O-MaSE est une méthodologie orientée agent personnalisable basée sur des concepts bien définis.[30].

O-MaSE a ses racines dans la méthodologie (MaSE) Multiagent Systems Engineering.
Tandis que MaSE a fourni un bon point de départ pour le développement de systèmes multi-agents, il eu plusieurs problèmes. Tout d'abord, MaSE produit des systèmes multi-agents avec une organisation fixe. Agents développés en MaSE ont joué un nombre limité de rôles et ont une capacité limitée de changer les rôles, quelle que soit leur capacité individuelle. En outre, MaSE ne comprend pas la notion de sous-équipes et avaient aucun mécanisme pour modéliser les interactions avec l'Environnement [30].

2.1. Meta-model O-MaSE:

Le méta-modèle O-MaSE définit les principaux concepts et les relations utilisées pour définir les systèmes multi-agents. Le méta-modèle O-MaSE est basé sur une approche organisationnelle et comprend les notions qui permettent la décomposition hiérarchique des systèmes.
Le méta-modèle O-MaSE a été dérivé de Modèle (OMACs) Organization Model for Adaptive Computational Systems[31].

OMACs capture les connaissances requises d'un système de l'organisation comme la structure et les capacités pour lui permettre d'organiser et réorganiser sont fonctionnement. Une organisation est composée de cinq entités: Buts, rôles, agents, Domain, et Protocoles [30].

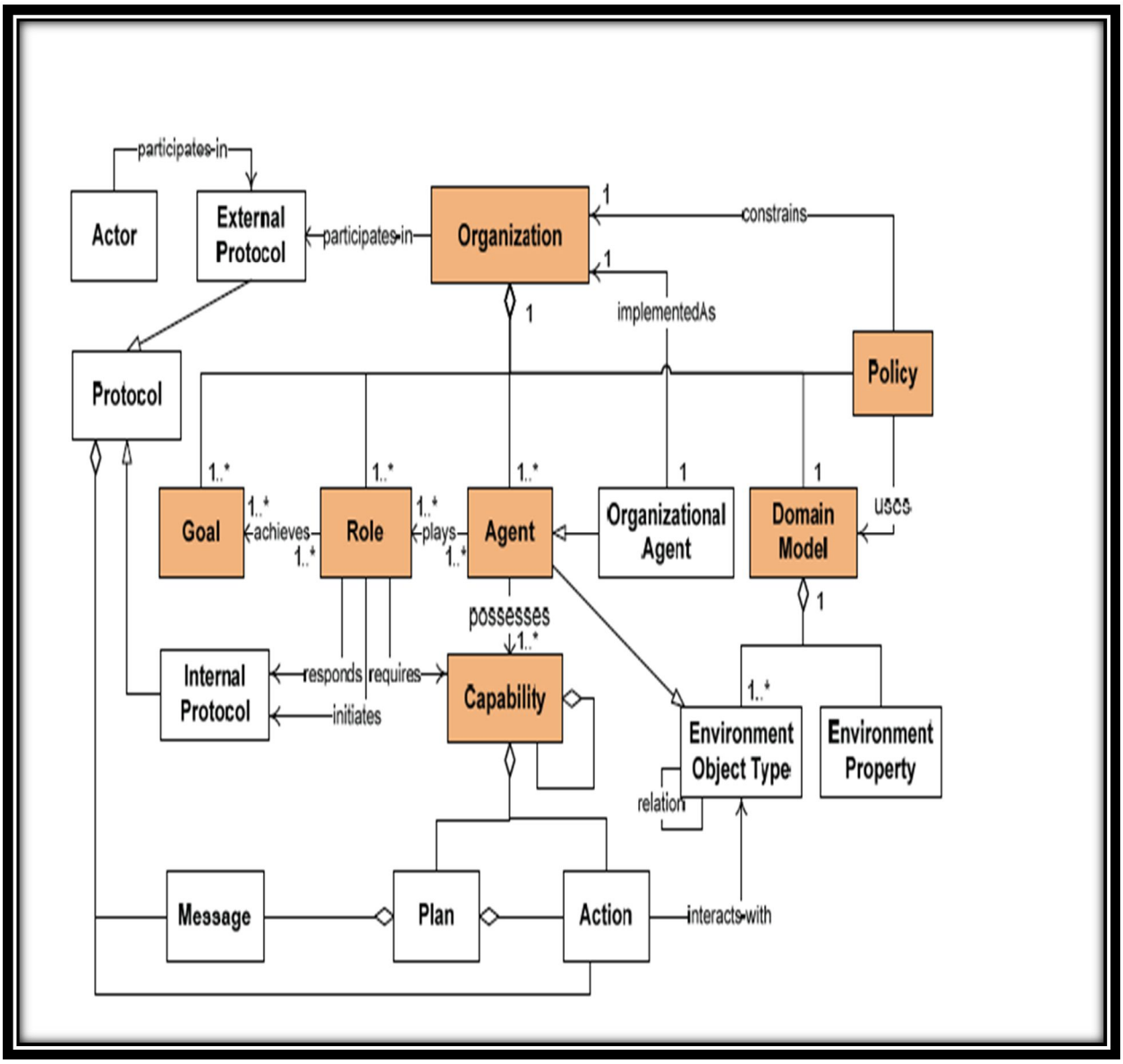

Figure 1: O-MaSE Meta-model

Comme toute autre méthode de modélisation la methode O-MaSE fonctione selon un processus simple composé de trois étapes : le tableau suivant résume les étapes et les modéles développés dans chaque phases ainsi que l'acteur (modélisateur) responsible de la production des modéles et même les language et les notations utilisées dans chaque étapes.

Work Units					
Activity	**Task**	**Technique**	**Work Products**	**Producer**	**Language**
Requirement Engineering	Model Goals	AND/OR Decomposition	AND/OR Goal Tree	Goal Modeler	Natural languages, for textual documents UML, for specific models Agent-UML O-MaSE specific notation Formal Language, for formal specification of properties of the system.
	Goal Refinement	Attribute-Precede-Trigger Analysis	Refined GMoDS		
Analysis	Model Organizational Interfaces	Organizational Modeling	Organization Model	Organizational Modeler	
	Model Roles	Role Modeling	Role Model	Role Modeler	
	Define Roles	Role Description	Role Description Document		
	Model Domain	Traditional UML notation	Domain Model	Domain Expert	
Design	Model Agent Classes	Agent Modeling	Agent Class Model	Agent Class Modeler	
	Model Protocol	Protocol Modeling	Protocol Model	Protocol Modeler	
	Model Plan	Plan Specification	Agent Plan Model	Plan Modeler	
	Model Policies	Policy Specification	Policy Model	Policy Modeler	
	Model Capabilities	Capability Modeling	Capabilities Model	Capabilities Modeler	
	Model Actions	Action Modeling	Action Model	Action Modeler	
	Model Service	Service Modeling	Service Model	Service Modeler	

Dans le cadre de notre travail nous avons utilisé quelques modèles pour présenter les capacités fonctionnelles de notre système multiagent et non pas la totalité du processus présenté dans le tableau précédant.

Dans ce qui suit nous présentons les modèles utilisés dans notre conception:

2.1.1. Le diagramme de buts :

L'objectif du diagramme de but est de transformer les exigences du système initial en un ensemble de buts structurés. Les modèles de buts se sont répandue dans de nombreuses méthodologies orientées agents [30].

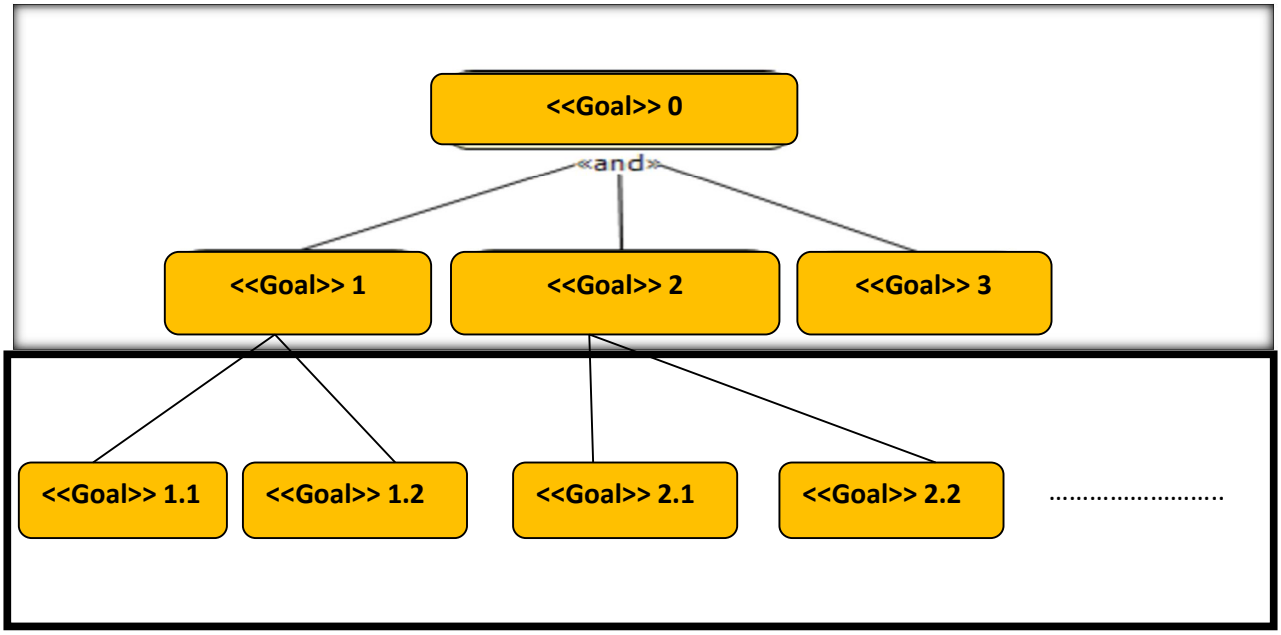

Figure 2: Diagramme de buts

L'approche typique de modélisation des buts est la décomposition AND / OR .L'objectif de cette approche est d'affiner le but global du système en un ensemble de sous-buts. Si tous les sous-but doit être atteint dans l'objectif d'atteindre le but parent la décomposition utilisée est AND, tandis que si les sous-buts représentent d'autres moyens pour atteindre les Buts parents, la décomposition utilisée est OR.

2.1.2. Le diagramme de Rôle

Le diagramme de rôle identifie tous les rôles dans l'organisation ainsi que leurs interactions les uns avec les autres et avec les acteurs externes. L'objectif de la modélisation du rôle est d'attribuer à chaque but un rôle spécifique.

En tant que première coupe, un seul rôle est souvent créé pour chaque but. Cependant, il est parfois bénéfique pour permettre à un seul rôle pour parvenir à plusieurs types de buts. Le concepteur doit également identifier les interactions entre les rôles et avec les acteurs extérieurs. Les interactions avec les acteurs externes peuvent être dérivées directement à partir du diagramme d'organisation si elle est fournie [30].

pour chaque but il faut définir un rôle qui va réaliser ce but et à chaque rôle il faut attribuer les capacités necessaires pour la rélisation du but.

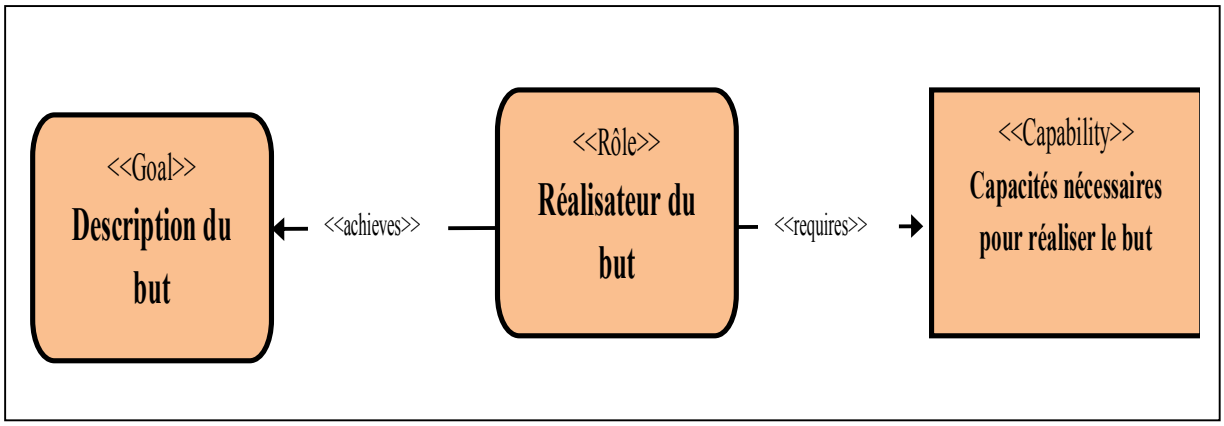

Figure 5 : Diagramme de rôle

2.1.3. Diagramme d'agent:

Le diagramme d'agent est une décomposition du système étudié en un ensemble d'unités actives où chaque unité doit jouer un ou plusieurs rôles selon la nature du système. Par exemple pour les tâches séquentielles on peut créer un seul agent et pour les tâches parallèles On peut créer plusieurs agents de type différents. Un agent doit posséder toutes les capacités pour jouer un rôle.

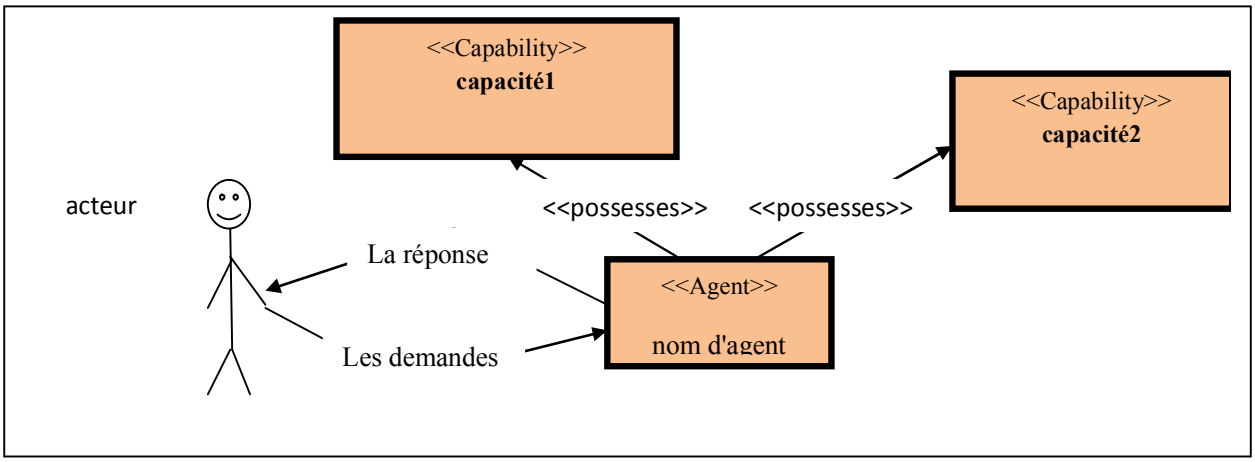

Figure 5 : Diagramme d'agent

2.1.3. Le diagramme de Protocoles

Le but de ce diagramme est de définir les détails des interactions entre les différents agents du système pour la réalisation du but global [30]. Il résume les messages qui gèrent le fonctionnement global du système

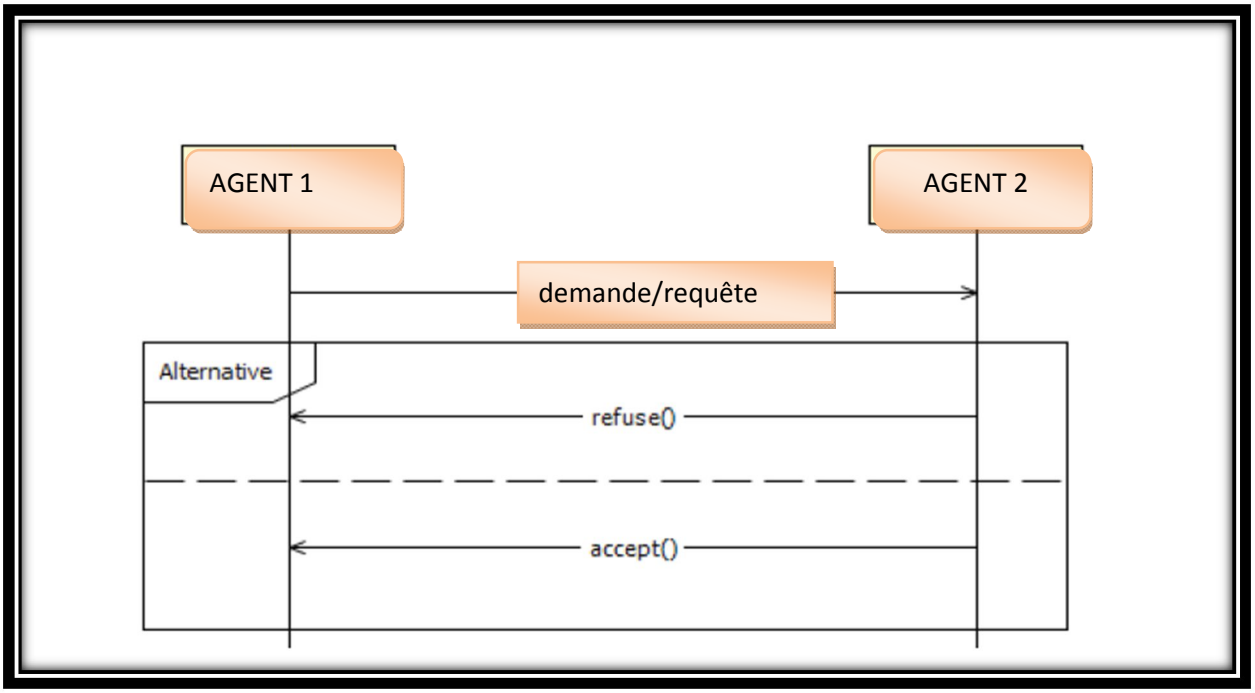

Figure 6 : Diagramme de protocole

Glossaire

Liste des abréviations, acronymes et sigles

CABRI	**Communauté d'Agglomération de Saint-Brieuc**
DRAP	**Dial A Ride Problem**
DRT	**Demand responsive transport**
MaSE	***MULTIAGENT SOFTWARE ENGINEERING***
MP-VRP	**Problème de Tournées de Véhicule a Produits Multiples**
O-MASE	**Organization-based Multi-agent Software Engeneering**
O-VRP	**Open Vehicle Routing Problem**
PVRP	**Periodic Vehicle Routing Problem**
S-VRP	**Split Delivery Vehicle Routing Problem**
SMA	**Système Multi-Agents**
IAD	**Intelligence Artificielle distribuée**
ILS	**Recherche locale itérative**
TAD	**Transport À la Demande**
1-VRP	**Problème de Tournées de Véhicule a un Seul Produit**
VRP	**Véhicule Routing Probelm**
VRP-1P	**Vehicle Routing Problem**
VRP-B	**Vehicle Routing Problem Back**
VRP-C	**Capacitated Vehicle Routing Problem**
VRP-FL	**Vehicle Routing Problem with Full Truckload**
VRP-HF	**Vehicle Routing Problem with Heterogeneous Fleet**
VRP-MD	**Multi-Dépôt Vehicle Routing Problem**

VRP-PD	**Vehicle Routing Problem Pick-up and Deliveries**
VRP-TW	**Vehicle Routing Problem with Time Windows**

Bibliographies

[1]:T.K. Ralphs, “Parallel branch and cut for capacitated vehicle routing”. Paralle Computing archive, Volume 29, Issue 5, 2 December 2002

[2] :E.Castex ; “Le Transport A la Demande (TAD) en France : de l'état des lieux à l'anticipation. Modélisation des caractéristiques fonctionnelles des TAD pour développer les modes flexibles de demain”

[3]:T. H. Cormen, C. E. Leiserson, and R. L. Rivest; “Introduction to algorithms”, chapitre 15, pages 350—355. The MIT Press, Cambridge, Massachuchetts, 1990

[4] :LeBouthilier ; “Modélisation UIVIL pour une architecture coopérative appliquée au problème de tournées de véhicules avec fenêtres de temps”.

[5]: L. Bodin and L. Berman; “Routing and scheduling of school buses by computer”

[6] : E. Bonomi et J.-L. Lutton ; “le recuit simulé pour la science

[7]: F. Glover; “Tabu Search, Part II”

[8]:C.H. Papadimitriou.; “The complexity of combinatorial optimization problerns. PhD thesis, Princeton University, New Jersey, USA, 1976.

[9] :B. Nachet ; “Modèle multi-agents pour la conception de systèmes d’aide à la décision collective”

[10] :I.Zidi ; “ Modélisation et Optimisation d'un Système de Transport a la Demande Multicritère et Dynamique”. Other. Ecole Centrale de Lille; Ecole nationale des sciences de l'informatique(Tunis), 2012. French.

[11] :N.BOUHA ; “Modélisation, approche multi agent et outils d’aide a la décision applique sur les systèmes du trafic routier urbain”

[12] :C.Herby ; “Apprenez à programmer en JAVA”, 2011.

[13] :M. FEKI ; “Optimisation distribuée pour la recherche des itinéraires multi-opérateurs dans un réseau de transport Co-modal”.

[15] : M. AKLI “problème de tournées de véhicules avec contraintes et Fenêtre de temps” Thèse de magister

[16] R. Kammarti ; “ approches évolutionnistes pour la résolution du 1-PDPTW statique et dynamique” Thèse de magister

[17]:Cordeau J-F; “A Branch-and-cut algorithm for the dial-a-ride problem”

[18]:Cordeau J-FLaporte G, Savelsbergh MWP, Vigo D; “Vehicle Routing”, In: C. Barnhart and G. Laporte (eds.), Transportation, Amsterdam: Elsevier

[19] : C. Angèle “Projet de Transport à la Demande”,Mai-Octobre 2003

[20] :J.Ferber; “les systèmes Multi-agents: Vers une Intelligence Collective”. Techniques et science informatiques, vol. 16, n°8, 1997, pp.979-1012.

[21] :M.Wooldridge, and J.R. Jennings; ‘’Agent Theories, architectures, and languages”: a Surevy. In M. Wooldridge and J. R. Jennnings, editors, intelligent Agent, LNAI890, Springer Verlag, 1995, pp.1-39.

[22] :N. Zarour ; ‘’contribution à la modélisation de la coopération des systèmes d’information distribués et hétérogènes : le système DAArCHE ‘’. Thèse de doctorat de l’Université Mentouri de Constantine. Le 18 septembre 2004.

[23]:L.Gasser; “Social Conceptions of Knowledge and action”. Rapport technique ACT-AI-355-90, MCC, Octobre 1990.

[24]:http://www.memoireonline.com/07/08/1413/m_modelisation-systeme-multi agentshypermedia-educatif22.html.ht

[25] :S. Bouzidi ; ‘’Un modèle de Négociation basé sur la Formation de Coalition pour l’Allocation des Taches dans les Systèmes d’Information Coopératifs‘’. Mémoire de magister.2005.

[26] :http://smartour.ptvgroup.com/fr/notre-solution/

[27]:TourSolver_Psheet-FR.pdf

[28]:http://www.transept.net/optimisation-chaine-logistique/optimisation-des-tournees.

[29]:http://macr.cis.ksu.edu/projects/agentTool/agentool.htm.

[30]:S. Deloach.,M. Wood.,C. Sparkman.; " Multiagent Systems Engineering ", International Journal of Software Engineering and Knowledge Engineering, vol.11, n° 3,p. 231-258, World Scientific, 2001.

[31]:S.DeLoach.; "Engineering Organization-Based Multiagent Systems", SELMAS, p.109-125, 2005.

[32]: O. CARDIN "Apport de la simulation en ligne dans l'aide à la décision pour le pilotage des systèmes de production : Application à un système flexible de production". Thèse de Doctorat de l'Université de Nantes.2007

[33]: http://www.eurodecision.com/decouvrez-notre-metier-suite

[34]: F. Mebrek "Outils d'aide à la décision bas´es sur la simulation pour la logistique hospitalière, application à un nouvel hôpital". Thèse de Doctorat École Doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand.2008.