

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA
RECHERCHE SCIENTIFIQUE
UNIVERSITE EL HADJ LAKHDAR BATNA



Faculté des Sciences
Département d'Informatique

Mémoire

Présenté en vue de l'obtention du diplôme de Magister en
Informatique

OPTION : Système Informatique de Communication

Présenté par :

KIMOUCHE Abdelkader

Thème

***Méta-heuristique pour la Résolution
des Problèmes de Transport :
Application pour le Transport des Patients***

Soutenu le : 17 / 12 / 2012

Devant le jury :

BILAMI Azzedine	Prof.	Univ.Batna	Président
BENMOHAMMED Mohammed	Prof.	Univ. Constantine	Rapporteur
ZIDANI Abdelmadjid	MCA	Univ. Batna	Examineur
CHIKHI Salim	Prof.	Univ. Constantine	Examineur
LEMOUARI Ali	MCB	Univ. Jijel	Invité

Année Universitaire 2011/2012

Je Dédie ce travail :

A mon père et ma mère.

A ma famille.

A tous ceux qui m'aiment et ceux que j'aime

Remerciements

Nous rendons grâce à Dieu qui nous a donné l'aide, la patience et le courage pour accomplir ce travail.

Je tiens à adresser mes plus vifs remerciements aux:

- Pr. BENMOHAMMED Mohamed pour m'avoir encadré et pour les recommandations qu'il m'a prodiguées et qui m'ont été d'un grand apport;
- Pr. BILAMI Azzedine qui m'a honoré par sa présence en qualité de président de jury;
- Membres de jury: Pr. CHIKHI Salim, Dr. ZIDANI Abdelmadjid pour avoir accepté de juger mon travail.

Je remercie aussi Dr. LEMOUARI Ali pour tous les conseils qu'il m'a donné.

Merci à tous ceux qui ont contribué à l'élaboration de ce travail de près ou de loin et qui méritent d'y trouver leurs noms.

L'auteur

Kimouche Abdelkader

Résumé

Ce mémoire porte sur les services de transport à la demande offerts à des personnes à mobilité réduite, typiquement des personnes âgées, malades ou handicapées...

Le problème de transport à la demande (DARP) est un problème d'optimisation qui consiste à déterminer les tournées et les horaires pour les véhicules qui effectuent le transport d'usagers à leur demande, d'une origine à une destination. C'est un cas particulier du problème de tournées de véhicule (VRP) qui appartient à la famille des problèmes NP-complets d'où la nécessité de l'utilisation des méthodes approchées ((méta-) heuristiques) pour résoudre ce type de problèmes.

Nous nous intéressons dans ce travail au transport des patients aux différentes unités de l'hémodialyse, sachant que le nombre de ces patients dans notre pays augmente d'une façon très rapide ces dernières années (14000 malades en 2011 ,15700 en mars 2012), cela pose un problème vraiment difficile que ce soit pour les opérateurs de transport sanitaire que pour les unités de l'hémodialyse.

Pour résoudre ce problème on a proposé un algorithme génétique, qui est testé sur des données générées aléatoirement et des données réelles d'un opérateur de transport sanitaire.

Mots-clés: Méta-heuristique, Problème de transport, Réseau de transport, NP-complets, VRPTW, DARP.

ملخص

هذه المذكرة تتناول خدمات النقل حسب الطلب المعروضة للأشخاص الذين لهم عجز في التنقل أمثال الكبار في السن، المرضى أو ذوي الاحتياجات الخاصة...

مشكل النقل حسب الطلب و الذي يرمز له بالانجليزية (DARP) هو أحد مشاكل التحسين و الذي يتمثل في تحديد مسارات وأوقات سيارات النقل التي تقوم بنقل الأشخاص حسب طلبهم، من منطقة ركوب إلى منطقة توصيل. هذا المشكل هو حالة خاصة من مشكل تحديد مسارات السيارات المعروف بالانجليزية باسم VRP و الذي ينتمي إلى مجموعة المشاكل المستعصية الحل و المعروفة بالانجليزية باسم NP-complete والتي تتطلب لحلها استعمال طرق تعطي نتائج جيدة و لكن ليس الحل الأمثل، هذه الطرق تسمى بالانجليزية ((meta-) heuristics approximate methods .

لقد اهتمنا في هذا العمل بمشكل نقل مرضى العجز الكلوي إلى مختلف وحدات تصفية الدم، بالعلم أن عدد هؤلاء المرضى في بلادنا يزداد بوتيرة سريعة جدا في الأعوام الأخيرة (14000 مريض في عام 2011 إلى 15700 في مارس 2012)، ما أدى إلى طرح مشكل صعب حقيقة ، سواء بالنسبة إلى عملاء النقل الصحي أو مراكز تصفية الدم.

لحل هذا المشكل، طرحنا خوارزمية يعتمد على مبدأ الوراثة (genetic algorithm) و الذي تم تجربته على معطيات تم تشكيلها عشوائيا و أخرى حقيقية حصلنا عليها من أحد عملاء النقل الصحي.

المفاتيح :

Méta-heuristic, Transportation problem, Transportation Network, NP-complete, DARP, VRPTW.

Abstract

This thesis focuses on transportation services available upon request to individuals with disabilities, typically the elderly, sick or disabled...

Dial-A-Ride Problem (DARP) is an optimization problem of determining the tours and times for vehicles that transport users at their request, from one origin to one destination. This is a special case of vehicle routing problem (VRP) which belongs to the family of NP-Hard problems, hence the need to use approximate methods ((meta-) heuristics) to solve such problems.

We are interested in this work to the transport of patients to different hemodialysis units, knowing that the number of patients in our country increases very rapidly in this recent years (14,000 patients in 2011, 15,700 in March 2012), that poses a really difficult problem that it is for the operators of medical transport that for the units of the hemodialysis.

To resolve this problem we proposed a genetic algorithm, which is tested on randomly generated data and real data of an operator of medical transport.

Key-words: Méta-heuristic, Transportation problem, Transportation Network, NP-complete, DARP, VRPTW.

Table des matières

Introduction générale	1
 Chapitre 1 : Les problèmes de tournées de véhicules	
1.1 Introduction	4
1.2 Problème d'optimisation	4
1.2.1 Variables de décision.....	5
1.2.2 Espace décisionnel / objectif	5
1.2.3 Contraintes.....	6
1.3 Le VRP et ses variantes	6
1.4 Modélisation mathématique de VRP	10
1.5 Les méthodes de résolution de VRP et de ses variantes.....	13
1.5.1 Les méthodes exactes	14
a. La méthode de Branch and Bound.....	14
b. La méthode de Branch and Cut	15
c. La programmation dynamique.....	15
d. Autres méthodes exactes	16
1.5.2 Les méthodes approchées	16
A. Les heuristiques	16
A.1 L'heuristique « groupe en premier, route en second »	17
A.2 L'heuristique « route en premier, groupe en second »	17
A.3 L'algorithme en pétale.....	17
B. Les métaheuristiques.....	17
B.1 Les métaheuristiques de recherche locale.....	18
B.1.1 Méthode de recuit simulé.....	18
B.1.2 Recherche tabou.....	19
B.1.3 La méthode de descente	20
B.2 Les métaheuristiques d'évaluation.....	21
B.2.1 Les algorithmes génétiques.....	21
B.2.2 Les colonies de fourmis	21
B.2.3 La recherche distribuée	22
B.2.4 Les algorithmes mimétiques	22
1.6 Conclusion.....	23
 Chapitre 2 : Le problème de transport à la demande	
2.1 Introduction	25

2.2	Présentation du problème de transport à la demande « Dial-A-Ride Problem ».....	25
2.3	Formulation mathématique.....	26
2.4	Classification des problèmes de transport à la demande	29
2.4.1	Selon le nombre des véhicules	29
2.4.2	Selon la disponibilité de l'information sur les requêtes	29
2.5	Etat de l'art des problèmes de transport à la demande « DARP »	30
2.5.1	DARP avec un seul véhicule	30
2.5.2	DARP avec plusieurs véhicules.....	31
2.6.	Conclusion	36

Chapitre 3 : Les algorithmes génétiques

3.1	Introduction	38
3.2	Terminologie	39
3.2.1	Gène.....	39
3.2.2	Chromosome.....	39
3.2.3	Individu.....	39
3.2.4	Fitness d'un individu	39
3.2.5	Population.....	39
3.3	C'est quoi un algorithme génétique ?	39
3.4	A quoi sert l'algorithme génétique ?	39
3.5	Principe des algorithmes génétiques	40
3.6	Pseudo code d'un algorithme génétique.....	41
3.7	Conception d'un algorithme génétique	41
3.8	Variantes.....	42
3.8.1	Codage.....	42
a.	Codage binaire	42
b.	Codage réel.....	43
3.8.2	Population initiale.....	43
3.8.3	La fonction d'adaptation.....	44
3.8.4	Sélection	44
a.	La sélection par roulette.....	44
b.	La sélection par rang.....	45
c.	La sélection par tournoi	45
3.8.5	Croisement.....	45
a.	Croisement binaire.....	46
a.1	Croisement en 1-point.....	46
a.2	Croisement en 2-points	46

a.3 Croisement en n-points	47
a.4 Croisement uniforme	47
a.5 Croisement avec trois parents	47
b. Croisement réel.....	48
b.1 L'opérateur de croisement PMX (<i>Partially Mapped Crossover</i>).....	48
b.2 L'opérateur de croisement CX (<i>Cycle Crossover</i>)	49
b.3 L'opérateur de Croisement OX (<i>Order Crossover</i>)	50
3.8.6 Mutation	51
a. Mutation en codage binaire	51
b. Mutation en codage réel	51
b.1 Mutation par inversion	51
b.2 Mutation par insertion	52
b.3 Mutation par déplacement	52
b.4 Mutation par permutation	53
3.9 Valeur des paramètres	53
3.10 Conclusion.....	54

Chapitre 4: Conception et architecture de l'application

4.1 Introduction	56
4.2 Description du problème	56
4.2.1 Les contraintes à respecter et l'objectif	57
4.3 Méthode de résolution	57
4.4 Architecture de l'application.....	69
4.5 Conclusion.....	77

Chapitre 5 : Résultats et discussion

5.1 Introduction	79
5.2 L'environnement de programmation	79
5.3 Choix du langage	79
5.4 Implémentation et Jeux de données.....	80
5.5 Présentation de l'interface	82
5.6 Paramètres d'algorithme	85
5.7 Paramètres de véhicule	86
5.8 Résultats	86
5.8.1 Problèmes P10D5	86
5.8.2 Problèmes P20D5	87
5.8.3 Problèmes P40D10	88

5.9	Analyse des résultats	89
5.9.1	Problèmes P10D5	89
5.9.2	Problèmes P20D5	92
5.10	Les données réelles.....	94
5.10.1	Résultats pour les données réelles	96
5.11	Conclusion.....	96
	Conclusion générale.....	97

Liste des figures

Chapitre 1 : Les problèmes de tournées de véhicules

Fig. 1.1	Les deux types de VRPB	8
Fig. 1.2	Les principales méthodes de résolution de VRP.....	14

Chapitre 2 : Le problème de transport à la demande

Fig. 2.1	Le DARP dynamique avec 4 clients connus et 1 client à insérer dynamiquement.....	30
-----------------	--	----

Chapitre 3 : Les algorithmes génétiques

Fig. 3.1	Principe générale des algorithmes génétiques	40
Fig. 3.2	Les phases de définition d'un codage	42
Fig. 3.3	Codage binaire	43
Fig. 3.4	Codage réel	43
Fig. 3.5	Croisement en un point de deux chromosomes	46
Fig. 3.6	Croisement en 2-points de deux chromosomes.....	46
Fig. 3.7	Croisement uniforme	47
Fig. 3.8	Croisement avec trois parents	48
Fig. 3.9	Opérateur de Croisement PMX.....	49
Fig. 3.10	Opérateur de Croisement CX.....	50
Fig. 3.11	Opérateur de Croisement OX.....	51
Fig. 3.12	Mutation par inversion	52
Fig. 3.13	Mutation par insertion.....	52
Fig. 3.14	Mutation par déplacement.....	52
Fig. 3.15	Mutation par permutation.....	53

Chapitre 4 : Conception et architecture de l'application

Fig. 4.1	codage de chromosome.....	58
Fig. 4.2	exemple d'un chromosome qui sera devisé par Split.....	59
Fig. 4.3	le graphe auxiliaire H	61

Fig. 4.4	résultat de la procédure <i>Split</i>	61
Fig. 4.5	l'application de l'opérateur de croisement sur un exemple.....	65
Fig. 4.6	l'application de l'heuristique d'insertion sur un exemple	66
Fig. 4.7	un individu avec les nœuds de chargement.....	67
Fig. 4.8	le package « info »	69
Fig. 4.9	diagramme de classe de l'application	70
Fig. 4.10	classe « IHM_For_Projet »	72
Fig. 4.11	classe « randome »	73
Fig. 4.12	classe « split »	73
Fig. 4.13	classe « insertion »	74
Fig. 4.14	classe « initialisation »	75
Fig. 4.15	classe « Crossover».....	76
Fig. 4.16	classe « Tour_selection»	77

Chapitre 5: Résultats et discussion

Fig. 5.1	l'organisation textuelle d'une instance d'un problème.....	81
Fig. 5.2	la représentation graphique d'une instance d'un problème.....	82
Fig. 5.3	l'interface de logiciel	82
Fig. 5.4	la fenêtre des paramètres de l'algorithme génétique.....	83
Fig. 5.5	la fenêtre des paramètres du véhicule	83
Fig. 5.6	l'interface après chargement de problème.....	84
Fig. 5.7	exemple d'une solution d'un problème	85
Fig. 5.8	la solution de l'instance P10D5_1 pour les demandes de dimanche	91
Fig. 5.9	la solution de l'instance P10D5_6 pour les demandes de mardi.....	91
Fig. 5.10	la solution de l'instance P20D5_1 pour les demandes de samedi	93
Fig. 5.11	la solution de l'instance P20D5_8 pour les demandes de lundi	94
Fig. 5.12	la solution appliquée par l'opérateur Ibn Sina pour jeudi.....	95

Liste des tableaux

Chapitre 5 : Résultats et discussion

Tab. 5.1	Résultats du cout total des tournées obtenus pour les problèmes de type P10D5	86
Tab. 5.2	Les résultats obtenus par l'AG sur les problèmes de P20D5	87
Tab. 5.3	Les résultats obtenus par l'heuristique basée essaim sur les problèmes de P20D5	87
Tab. 5.4	Les résultats obtenus par l'AG sur les problèmes de P40D10	88
Tab. 5.5	Les résultats obtenus par l'heuristique basée essaim sur les problèmes de P40D10	88
Tab. 5.6	Le pourcentage moyen du temps ajoutés au temps nécessaire pour servir les demandes pour P10D5	89
Tab. 5.7	Le temps moyen d'une tournée par jour pour la série des problèmes P10D5	90
Tab. 5.8	Le pourcentage moyen du temps ajoutés au temps nécessaire pour servir les demandes pour P20D5	92
Tab. 5.9	Le temps moyen d'une tournée par jour pour la série des problèmes P20D5	92
Tab. 5.10	Résultats du cout total des tournées obtenus pour les données réelles	96

Liste des algorithmes

Chapitre 1 : Les problèmes de tournées de véhicules

Algorithme 1.1 Recuit simulé	19
---	----

Algorithme 1.2 Colonies de fourmis	22
---	----

Chapitre 3 : Les algorithmes génétiques

Algorithme 3.1 Un algorithme génétique	41
---	----

Chapitre 4 : Conception et architecture de l'application

Algorithme 4.1 Procédure Split.....	59
--	----

Algorithme 4.2 : L'heuristique d'insertion aléatoire.....	62
--	----

Algorithme 4.3 : Le principe de la sélection	64
---	----

Algorithme 4.4 : Le principe de croisement.....	65
--	----

Algorithme 4.5 : L'heuristique d'insertion des nœuds de déchargement	67
---	----

Algorithme 4.6 : L'algorithme génétique pour le problème de transport de patient.....	68
--	----

Introduction Générale

La logistique est apparue dans le domaine militaire au moment où l'on a tenté de rationaliser l'expérience acquise au cours des campagnes napoléoniennes. Au 19^e siècle sont nés dans l'armée française les services de l'intendance et du train des équipages. Tous les états-majors comportaient désormais un bureau logistique. Les calculs de besoins, de délais de transport, d'espace de ramassage et de stocks faisaient apparaître une sorte de nouvelle science que l'on n'appelait pas encore recherche opérationnelle mais qui en avait déjà un peu l'esprit et que l'on appela *logistique* pour en consacrer le caractère *logico-mathématique*. L'expérience acquise dans l'armée a été ensuite transposée aux entreprises.

L'activité de transport est le cœur même de la logistique, c'est l'un de ses postes de coûts les plus importants de telle sorte que l'organisation logistique est souvent déterminée par l'optimisation des coûts de transports. Qui dit transport, dit organisation des tournées de véhicules (VRP : Vehicule Routing Problem). Le VRP est un problème classique qui consiste à construire des routes visitant tous les clients en minimisant le coût du transport, en satisfaisant les demandes de ces derniers et en respectant les différentes capacités des véhicules.

Une meilleure organisation des tournées de véhicules présente un potentiel d'économies majeur. C'est cette importance accrue des problèmes d'optimisation des tournées dans le secteur de transport qui a attiré de plus en plus les chercheurs et les gestionnaires d'entreprises.

Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure évaluation au regard du critère défini.

Plusieurs problèmes réels peuvent être modélisés par des VRP, à savoir la conception des systèmes de l'industrie, le traitement d'images, les problèmes rencontrés dans les réseaux industriels, la conception d'emplois du temps, le routage dans les réseaux informatiques etc. La majorité de ces problèmes sont qualifiés difficiles, car il n'est en général pas possible de fournir dans tous les cas une solution optimale dans un temps raisonnable.

Pour la résolution des problèmes d'optimisation, il existe de nombreuses méthodes génériques de résolution. Ces méthodes se classent en deux catégories bien distinctes. D'une part,

les méthodes exactes, cherchant à trouver de manière certaine la solution optimale en considérant l'ensemble des solutions possibles. D'autre part, les méthodes approchées, qui se contentent de rechercher une solution « de bonne qualité ». Dans le cadre de l'optimisation de problèmes NP-difficiles, les méthodes exactes ne résolvent pas les problèmes en temps polynomial, ce qui limite la taille des problèmes solubles par ce type d'approche. Cette limite est variable selon les problèmes, mais toujours présente. C'est pour la résolution approchée des problèmes de grande taille qu'ont été introduites les méta-heuristiques. Parfois on utilise également ces méthodes pour des applications temps réel qui ne permettent pas d'utiliser une méthode exacte, qui devient alors trop coûteuse en temps de calcul.

Les métaheuristiques sont des méthodes génériques de résolution approchée de problèmes d'optimisation. Elles permettent d'envisager une résolution approchée de nombreux problèmes d'optimisation différents, avec un minimum d'adaptation réalisée pour chaque problème. Parmi ces méthodes on trouve les algorithmes génétiques. Ces derniers sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique. À chaque génération, un nouvel ensemble de créatures artificielles (codées sous forme de chaînes de caractères) est construit à partir des meilleurs éléments de la génération précédente.

Dans le cadre de notre travail, nous nous intéressons à une variante de VRP, spécialement destiné au transport des personnes plus particulièrement le transport des patients, cette variante est appelée le transport à la demande ou « Dial-A-Ride Problem » (DARP). Notre but donc est de concevoir un outil de résolution de DARP, à base des algorithmes génétiques, permettra d'offrir à l'utilisateur un ensemble des solutions de meilleures qualités.

Ce mémoire, est organisée autour de cinq chapitres : le premier chapitre est consacré à une littérature décrivant un état de l'art succincte des problèmes de tournées de véhicules VRP, avec la présentation des différentes approches utilisées dans la résolution, et leurs variantes. Dans le deuxième chapitre, une description détaillée du problème DARP, la formulation mathématique ainsi qu'un état de l'art sont présentés. Par la suite, le chapitre 3, nous y détaillons le principe des algorithmes génétique, leurs caractéristiques, les opérateurs génétiques utilisés ainsi que les codages possibles à utiliser. Le chapitre 4, est consacré à l'adaptation des algorithmes génétiques au problème DARP considéré. Finalement, le dernier chapitre présente les résultats obtenus.

CHAPITRE 1

Les problèmes de tournées des véhicules

1.1 Introduction

Le problème de tournée de véhicule (VRP : vehicul routing problem) est l'un des plus beaux succès de la recherche opérationnelle. Suite à une collaboration très étroite entre les spécialités de la programmation mathématique et de l'optimisation combinatoire, d'une part, et les gestionnaires de transports, d'autre part, de nombreuses implantations de systèmes informatiques d'optimisation de tournées de véhicules ont vu le jour.

Parmi toutes les applications pratiques de ce problème, nous pouvons citer, entre autres, la distribution de journaux, le ramassage scolaire, la collecte d'ordures, la fourniture de combustible, la distribution de produits aux hypermarchés et magasins, la distribution de courrier, la gestion préventive d'inspection des routes,...

En toute généralité, le VRP 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.

Dans ce qui suit, nous donnons une définition d'un problème, dans le point de vue informatique, ensuite nous donnons quelques notions de base de l'optimisation, puis nous présentons les variantes de VRP. Enfin, nous présentons les méthodes de résolution du VRP les plus connues.

1.2 Problème d'optimisation

Dans ce qui suit, on se limitera à l'optimisation mono-objective.

Un problème d'optimisation consiste à trouver, parmi un ensemble de solutions potentielles, une solution optimale au regard d'un critère donné. De manière plus formelle, à chaque instance d'un tel problème, est associé un ensemble S des solutions potentielles et une fonction de coût f , qui associe à chaque solution potentielle $s \in S$ une valeur numérique $f(s)$. Résoudre l'instance (S, f) du problème d'optimisation consiste à trouver une solution $s^* \in S$ qui minimise (ou maximise) la valeur de la fonction de coût f . Une telle solution est appelée *solution optimale*, ou *optimum global*. Elle n'est pas forcément unique. Voici une définition, pour les cas de la minimisation [Papa et Stei 1982] :

Définition : Une instance d'un problème de minimisation est un couple (S, f) où S est un ensemble de solutions potentielles ou configurations, et f une fonction $f: S \rightarrow R$. Le but est de trouver $s^* \in S$ tel que $f(s^*) < f(s)$ pour tout élément $s \in S$.

1.2.1 Variables de décision

Dans les problèmes d'optimisation, les *variables de décision* sont des variables pour lesquelles des valeurs sont à choisir. Cet ensemble de variables est appelé *vecteur de décision*. Soit un problème d'optimisation avec n variables de décision. Le vecteur de décision est représenté par :

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Les différentes valeurs possibles prises par les variables de décision $x_i \in R$ ou N constituent l'ensemble des solutions envisageables.

Notons que l'on peut généralement distinguer deux branches de l'optimisation. Un problème d'optimisation est dit continu, si les domaines de définition des variables de décision sont continus (souvent dans R). Un problème d'optimisation est dit combinatoire si les domaines de définition des variables de décision sont discrets (souvent les variables sont binaires ou dans N).

1.2.2 Espace décisionnel / objectif

L'ensemble des *n-uplets* de valeurs réelles (ou entières, binaires ...) composant le vecteur de décision est un espace de dimension n . L'ensemble des valeurs pouvant être prises par le vecteur de décision constitue l'espace de recherche de la méthode d'optimisation. Deux espaces euclidiens sont à considérés en optimisation :

- L'espace *décisionnel*, de dimension n , n étant le nombre de variables de décision.
- L'espace *objectif*, l'ensemble de définition de la (des) fonction(s) objectif(s). Cet espace est défini dans R .

Dans le cadre des algorithmes évolutionnaires, faire varier les variables de décision peut être apparenté à faire varier les gènes d'un individu. Dans ce cas, l'espace décisionnel peut

s'appeler également espace *génotypique*. L'espace objectif peut également être appelé espace *phénotypique*. La valeur dans l'espace objectif d'une solution est généralement appelée *coût*, ou *fitness*.

1.2.3 Contraintes

Dans la plupart des problèmes d'optimisation, des restrictions sont imposées par les caractéristiques du problème, ou par les données d'une instance de problème. Ces restrictions doivent être satisfaites afin de considérer une solution comme acceptable. Cet ensemble de restrictions, ou *contraintes*, décrit les dépendances entre les variables de décision et les paramètres du problème. On formule usuellement ces contraintes c_i par un ensemble d'inégalités, ou d'égalités de la forme :

$$c_i(\bar{x}) \geq 0 \quad i = 1, \dots, m$$

1.3 Le VRP et ses variantes

Durant des années de recherches sur le VRP d'autres dérivées de ce problème sont apparues. Ces apparitions sont dues essentiellement aux activités des chercheurs qui travaillent de plus en plus sur les problèmes de transports et de distribution que rencontrent les sociétés. Dans ce qui suit nous allons présenter les principaux problèmes dérivés du VRP.

- **CVRP** (Capacitated Vehicle Routing Problem)

Dans le CVRP, toutes les demandes des clients sont statiques et elles correspondent aux demandes de livraison. Les véhicules sont identiques, un seul dépôt central est considéré, seul la restriction de capacité est imposée. L'objectif est de minimiser le coût total (c'est-à-dire, une fonction pondérée du nombre des routes et leur longueur) pour servir tous les clients [Ralphs et al. 2001].

- **DVRP** (Dynamic Vehicle Routing Problem)

Dans la version statique de VRP, des informations sur les demandes de service, l'annulation de service etc., appropriées à la planification des routes sont connues à l'avance et ne changent pas après la construction des routes. Dans la version dynamique du problème, on ne connaît pas à l'avance toutes les informations appropriées à la planification des routes, mais les routes sont construites progressivement avec le temps [Kilby et al. 1998], [Gendreau et al. 1998a].

- **VRPTW** (Vehicle Routing Problem with Time Windows)

Le VRPTW peut être défini comme une flotte homogène de véhicules avec des caractéristiques fixes (capacité, vitesse, etc.), qui commencent et terminent à un dépôt central, et transportent des marchandises à un ensemble de clients. Durant le transport, la capacité de chaque véhicule ne peut être dépassée. Chaque client ne peut être servi qu'une seule fois et dans un intervalle de temps prédéfini. À savoir, un véhicule peut ne pas arriver au site de destination dans la fenêtre d'horaire, et dans ce cas si le temps d'arriver est supérieur à la limite supérieure alors une violation de la contrainte de fenêtre de temps aura lieu, sinon si le temps d'arriver est inférieur à la limite inférieure alors le véhicule doit attendre. L'objectif de VRPTW est de réduire non seulement la distance totale de voyage, mais aussi le temps total de voyage [Cordeau et al. 2000].

- **DVRPTW** (Dynamic Vehicle Routing Problem with Time Windows)

Ce problème constitue la fusion des deux derniers. Outre les nouvelles contraintes temporelles de fenêtres de temps, le DVRPTW assume aussi la composante dynamique d'arrivées d'événements urgents (nouvelle demande, nouveau client, panne d'un véhicule,...) [Gendreau et al. 1999].

- **VRPB** (Vehicle Routing Problem with backhauls)

Le VRP avec Backhauls (VRPB) est l'extension du VRP dans lequel l'ensemble des clients est partitionné en deux sous-ensembles. Le premier sous-ensemble contient n fournisseurs, chacun ayant une quantité de produit à livré. Le deuxième sous-ensemble contient m clients, où une quantité de produit attendue doit être déchargé [Jacobs-Blecha et al. 1998]. Il existe deux types de VRPB.

Le premier type est appelé livré-premier et collecte-deuxième VRPB, qui exige que les clients doivent être servis après les fournisseurs dans chaque route pour réarranger les produits transportés dans le véhicule sur une tournées. Le deuxième type est appelé mixte collecte-livraison VRPB, qui ne nécessite pas cet ordre entre les clients et les fournisseurs dans chaque route. La figure 1.1 illustre les deux types.

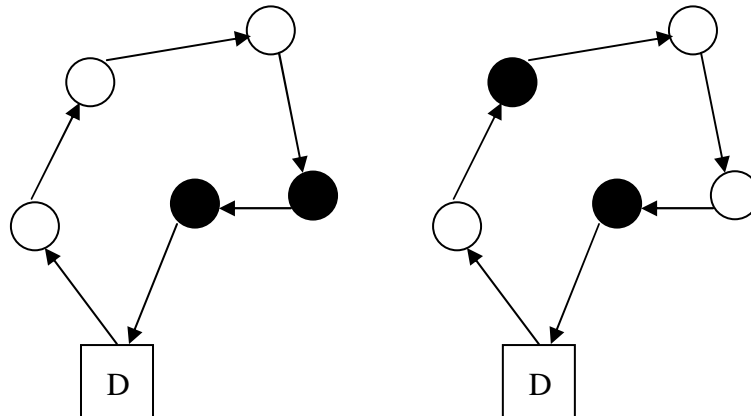
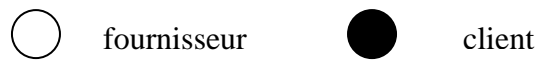


Fig.1.1 Les deux types de VRPB

- **PVRP** (Periodic Vehicle Routing Problem)

Le problème de tournées de véhicules multi périodique (PTVMP) consiste à livrer pour un ensemble de clients, la quantité demandée d'un ou de plusieurs produits sur un horizon de temps donné. Dans ce problème, la quantité de produits livrée à un client, permet à ce dernier de subvenir à ces besoins en attendant la prochaine visite du véhicule. Le but principale de ce problème se divise en deux parties : la première partie consiste à planifier les horaires de livraisons de chaque client sur un horizon de temps prédéterminé ; la deuxième consiste à organiser les tournées des véhicules afin d'effectuer les livraisons nécessaires tout en optimisant le coût total de transport [Witucki et al. 1997].

- **SVRP** (Stochastic Vehicle Routing Problem)

Un problème VRP est dit stochastique si au moins un de ses éléments est aléatoire, c'est-à-dire avec une certaine incertitude. Ces éléments peuvent être une ou plusieurs demandes des clients, les temps ou les coûts de transport etc., Le problème avec les demandes stochastiques est celui le plus étudié dans la littérature. La majorité des travaux de recherches supposent que les demandes aléatoires suivent une loi de distribution « normal » [Groth 2002].

- **TSP** (Travelling Saleman Problem)

Le problème du voyageur de commerce consiste, étant donné un ensemble de villes séparées par des distances données, à trouver le plus court chemin qui relie toutes les villes. C'est donc un cas particulier de VRP sans contrainte de capacité et avec un seul véhicule [Rego et al. 1994].

- **VRPPD** (Vehicle Routing Problem with Pick-up and Delivery)

Les problèmes de collecte et livraison (PCL) sont des problèmes d'optimisation des transports par tournées dans lesquels des passagers ou des marchandises doivent être transportés d'un point à un autre [Mechti 1995].

- **MDVRP** (Multi-Depot Vehicle Routing Problem)

Dans ce type de problème, plusieurs dépôts géographiquement distribués existent. Une tournée, dans ce genre de problème, est assurée par un véhicule qui part et revient au même dépôt initial [Fischetti et al. 1999].

- **VRPHF** (Vehicle Routing Problem with Heterogeneous Fleet)

La seule différence entre un VRPHF et un VRP est que la flotte de véhicule est hétérogène. Une flotte de véhicules hétérogènes est composée de véhicules qui sont de types différents. Les véhicules peuvent être différenciés par leurs coûts de transport, leurs capacités de transport, leurs vitesses, leurs tailles,... [Taillard 1999], [Prins 2002].

- **OVRP** (Open Vehicle Routing Problem)

La différence entre un OVRP et un VRP est que dans le premier les véhicules ne sont pas tenus de retourner au dépôt. Dans le cas où ils le sont, ils rebroussement chemin en revisitant les clients qui leur sont affectés dans l'ordre inverse. C'est pourquoi les parcours des véhicules sont tous des chemins ouverts [Fu et al. 2003].

- **SDVRP** (Split Delivery Vehicle Routing Problem)

Dans ce genre de problème, la demande d'un client peut être satisfaite sur plusieurs tournées ; pour cela un client peut être visité plusieurs fois si cela est nécessaire. Dans ce

type de problème, et contrairement aux autres, la demande d'un ou plusieurs clients peut excéder la capacité du véhicule [Archetti et al.2002].

- **PDPTW** (Pick-up and Delivery Problem with Time Windows)

Ce problème est une variante du VRPTW. Outre l'existence des contraintes de fenêtres de temps, ce problème possède un ensemble de clients et un ensemble de fournisseurs. A chacun de ces clients correspond un et un seul fournisseur. Les véhicules ne doivent alors passer par un client qu'après avoir visité son fournisseur. [Pasaraftis 1983a].

1.4 Modélisation mathématique de VRP

Soit $G = (V, A)$ un graphe où $V = \{1, \dots, n\}$ est un ensemble de sommets avec le sommet 1 pris comme dépôt, et $A = \{(i, j) \mid i, j \in V \text{ et } i \neq j\}$ est l'ensemble des arcs. $V' = V \setminus \{1\}$ est l'ensemble des villes ou clients.

À chaque arc est associé un coût non négatif c_{ij} . Celui-ci peut être interprété comme le coût de voyage ou le temps du voyage entre i et j .

Nous supposons, ici, que nous disposons de m véhicules identiques de capacité de transport D . Le VRP consiste à déterminer un ensemble de tournées de véhicules de coût minimal de telle façon que :

- Chaque ville de V' soit visitée une et une seule fois par un et un seul véhicule ;
- Toutes les routes commencent et se terminent au dépôt ;
- Certaines contraintes soient remplies.

Les contraintes prises en compte dans cette formulation :

- *Les restrictions de capacité* : à chaque ville i de V' est associé un poids d_i non négatif représentant la demande, la somme des poids d'une tournée ne dépassant pas la capacité du véhicule.
- *Les restrictions du temps total* : le temps total d'une tournée ne doit pas dépasser une borne T . Ce temps étant constitué des temps des voyages entre les villes et des temps d'arrêt à chaque ville sur la route.

Voici une formulation mathématique de CVRP proposée par Rego et Raucairol dans [Rego et al 1994], avec les notations suivantes :

Les Constantes :

n = nombre de sommets ;

m = nombre de véhicules ;

D = capacité d'un véhicule ;

T_k = temps maximal de parcours de véhicule k ;

d_i = demande du sommet i ($d_1=0$) ;

t_i^k = temps nécessaire au véhicule k pour décharger ou charger au sommet i ;

t_{ij}^k = temps nécessaire au véhicule k pour traverser l'arc (i, j) ;

c_{ij} = coût ou distance du sommet i au sommet j .

Variables de décision :

$$x_{ij}^k = \begin{cases} 1 & \text{Si le véhicule } k \text{ voyage de sommet } i \text{ au sommet } j \\ 0 & \text{Sinon} \end{cases}$$

Avec $X^k = (x_{ij}^k)$

La fonction à optimiser est :

$$\text{Minimiser } \sum_{i=1}^n \sum_{j=1}^n c_{ij} \sum_{k=1}^m x_{ij}^k \quad (1.1)$$

Sous contraintes

$$\sum_{i=1}^n \sum_{k=1}^m x_{ij}^k = 1 \quad j = 2, \dots, n \quad (1.2)$$

$$\sum_{j=1}^n \sum_{k=1}^m x_{ij}^k = 1 \quad i = 2, \dots, n \quad (1.3)$$

$$\sum_{i=1}^n x_{ip}^k - \sum_{j=1}^n x_{pj}^k = 0 \quad k = 1, \dots, m ; p = 1, \dots, n \quad (1.4)$$

$$\sum_{i=1}^n d_i \left(\sum_{j=1}^n x_{ij}^k \right) \leq D \quad k = 1, \dots, m \quad (1.5)$$

$$\sum_{i=1}^n t_i^k \sum_{j=1}^n x_{ij}^k + \sum_{i=1}^n \sum_{j=1}^n t_{ij}^k x_{ij}^k \leq T_k \quad k = 1, \dots, m \quad (1.6)$$

$$\sum_{j=2}^n x_{1j}^k \leq 1 \quad k = 1, \dots, m \quad (1.7)$$

$$\sum_{i=2}^n x_{i1}^k \leq 1 \quad k = 1, \dots, m \quad (1.8)$$

$$X^k \in S \quad (1.9)$$

Où S est proposé dans [Laporte 1992] de la façon suivante : Sachant que Q est un ensemble de sommets visités par un seul véhicule.

$$S = \left\{ \left(x_{ij}^k \right) \left| \sum_{i \in Q} \sum_{j \notin Q} x_{ij}^k \geq 1, \forall Q \subset N, |Q| \geq 2; k = 1, \dots, m \right. \right\} \quad (1.10)$$

$$S = \left\{ \left(x_{ij}^k \right) \left| \sum_{i \in Q} \sum_{j \in Q} x_{ij}^k \leq |Q| - 1, \forall Q \subset N, |Q| \geq 2; k = 1, \dots, m \right. \right\} \quad (1.11)$$

$$S = \left\{ \left(x_{ij}^k \right) \left| \begin{array}{l} y_i^k - y_j^k + (n-1)x_{ij}^k \leq n-2, i, j = 1, \dots, n; i \neq j; k = 1, \dots, m \\ 1 \leq y_i^k \leq n-1, i = 2, \dots, n \end{array} \right. \right\} \quad (1.12)$$

La fonction objective (1.1) consiste à minimiser le coût total de transport.

Les équations (1.2) et (1.3) assurent que chaque client soit servi par un et un seul véhicule.

La continuité d'une tournée est représentée par les équations (1.4) : un véhicule visitant un sommet doit en sortir.

Les équations (1.5) sont les contraintes de capacité d'un véhicule, celle (1.6), les contraintes de durée totale d'une tournée.

Les équations (1.7) et (1.8) assurant le non dépassement de la disponibilité d'un véhicule.

Finalement, les équations (1.9), (1.10), (1.11), et (1.12) représentent les contraintes d'élimination des sous-tours.

Le fait d'ajouter, de modifier ou de supprimer des contraintes peut nous faire passer d'un problème à un autre problème dérivé du VRP. Par exemple, la modélisation mathématique précédente est celle d'un CVRP ; en enlevant les contraintes (1.5) et (1.6) nous obtenons une modélisation d'un VRP.

1.5 Les méthodes de résolution de VRP et de ses variantes

Le VRP appartient à la famille des problèmes NP-difficiles [Savlesbergh 1995b], c'est-à-dire qu'on ne connaît aucun algorithme de temps polynomial pour résoudre le problème.

- Les méthodes exactes où la solution optimale du problème est obtenue ;
- Les méthodes approchées qui permettent d'obtenir de bonnes solutions sans toutefois pouvoir garantir leur optimalité.

La figure 1.2 résume la plupart de ces méthodes.

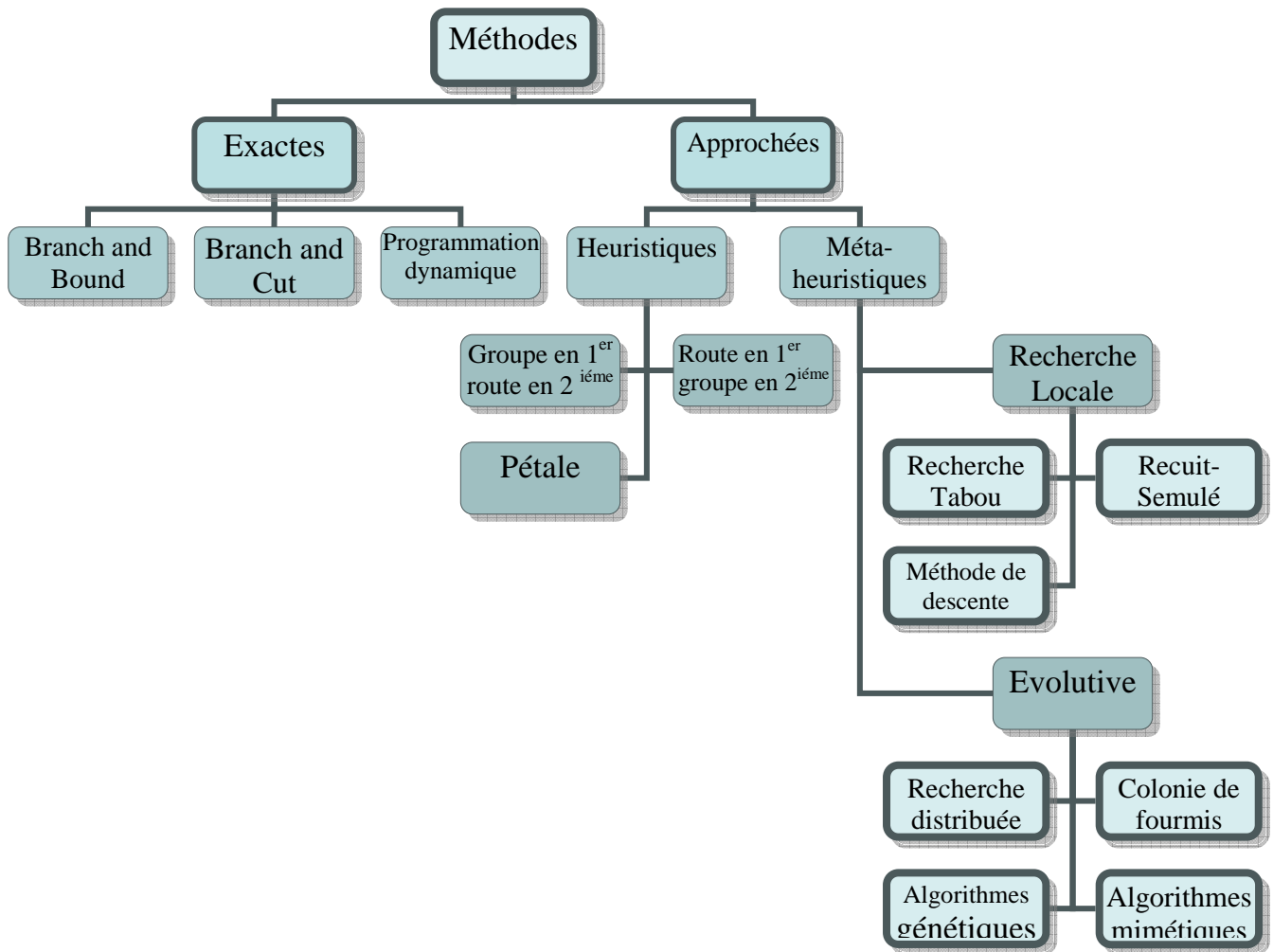


Fig.1.2 Les principales méthodes de résolution de VRP.

1.5.1 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. En 2001 une méthode résolvant un problème contenant 100 clients a été proposée dans [Ralphs 2002].

Parmi ces méthodes on peut citer :

a. 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. On représente l'exécution de la méthode à travers une arborescence, tel que la racine de cette arborescence représente l'ensemble de toutes les solutions du problème considéré.

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, devenant ainsi des enfants de la racine.

La méthode est ensuite appliquée récursivement à ces sous-problèmes, engendrant ainsi une arborescence. Si une solution optimale est trouvée pour un sous-problème, elle est réalisable, mais pas nécessairement optimale, pour le problème de départ, comme elle est réalisable, elle peut être utilisée pour éliminer toute sa descendance : si la borne inférieure d'un nœud dépasse la valeur d'une solution déjà connue, alors on peut affirmer que la solution optimale globale ne peut être contenue dans le sous-ensemble de solution représenté par ce nœud. La recherche continue jusqu'à ce que tous les nœuds soient soit explorés ou éliminés.

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.

b. 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.

c. 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. » [Borne et al 1990] Pour obtenir le chemin optimal du problème,

il suffit de construire les différents sous chemins optimaux. Cette méthode a été utilisée dans [Rego et al 1994] pour la résolution de problème allant de 10 à 25 clients.

d. 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, de l'algorithme \bar{A}^* ([CLR 1990]). D'autres approches exactes sont entièrement spécifiques au problème traité, comme l'algorithme de Johnson pour l'ordonnancement [Joh 1954]. Elles sont généralement appliquées aux problèmes peu difficiles, même si l'on en rencontre aussi pour la résolution de problèmes *NP-difficiles*.

1.5.2 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.

A. 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 VRP, nous citons :

- le groupe en premier, la route en second ;
- la route en premier, le groupe en second ;

- l'algorithme en Pétale.

A.1 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 [LeBouthillier 2000].

A.2 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 VRP [Bodin et al. 1979] [Golden et al. 1982].

A.3 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 [Ryan et al. 1993].

B. Les métaheuristiques

Les métaheuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile [Sait et Youssef 1999]. 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 métaheuristiques 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 métaheuristiques sont subdivisées en deux grandes familles :

1. les métaheuristiques de recherche locale :

- Recuit Simulé ;
- la recherche de tabou ;
- la méthode de la descente.

2. les métaheuristiques d'évolution :

- les algorithmes génétiques ;
- la recherche distribuée ;
- les colonies de fourmis.
- les Algorithmes mimétiques

B.1 Les métaheuristiques de recherche locale

Les métaheuristiques 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 tabou et la méthode de descente (recherche locale).

B.1.1 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 [Bono et Lutt 1988].

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

- On commence par choisir un point de départ au hasard ;
- On calcule un voisin de ce point ($\gamma = \text{Voisin}(x)$) ;
- On évalue ce point voisin et on calcule l'écart par rapport au point d'origine ($\Delta C = C(\gamma) - C(x)$);
- Si cet écart est négatif, on prend le point γ comme nouveau point de départ, s'il est positif on peut quand même accepter le point γ comme nouveau point de départ, mais avec une probabilité $e^{\frac{-\Delta C}{T}}$ (qui varie en sens inverse de la température T).
- Au fur et à mesure du déroulement de l'algorithme, on diminue la température T

$(T = \alpha(T))$, souvent par paliers ;

- On répète toutes ces étapes tant que le système n'est figé (par exemple, tant que la température n'a pas atteint un seuil minimal).

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. $r = \text{valeur aléatoire dans } [0,1]$;
 11. **Si** $r < p$ **Alors** $y = x$;
 12. $T = \alpha(T)$;
 13. **Jusqu'à** $(T > \Delta T)$
-

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.

B.1.2 Recherche Tabou

La recherche Tabou est une métaheuristique originalement développée par Glover [Glover 1989] [Glover 1990]. 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,....

Le principe général de la recherche de tabou est le suivant.

Soit une fonction objective f devant être minimisée dans un ensemble X de solutions admissibles. Un voisinage $N(s)$ est défini pour chaque solution s de X .

La recherche tabou est une méthode itérative qui, partant d'une solution initiale, tente d'atteindre la solution optimale, en exécutant, à chaque pas, des mouvements dans un graphe d'espace d'états G . En acceptant de détériorer la valeur de la solution courante, cela permet de s'éloigner d'un optimum local. Mais il y a alors un risque de faire un cycle qui peut être évité en utilisant une liste T tabou conservant en mémoire les derniers mouvements pendant un nombre limité d'itération.

Les différentes étapes de la recherche tabou sont :

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

$$s^* := s ;$$

$$T := \emptyset.$$

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 .

Pour la résolution du VRP, cette méthode a été utilisée dans [Gendreau et al. 1994], [Taillard 1993], [Xu et al. 1996], [Rego et al. 1996].

B.1.3 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 [Papa 1976], [PS82]. 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 : la descente déterministe (en général on choisit la solution apportant la plus grande amélioration), la descente stochastique

(choix aléatoire parmi les solutions améliorant la solution initiale) et la descente vers le premier meilleur.

B.2 Les métaheuristiques d'évolution

Le principe des métaheuristiques 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, la recherche distribuée et les algorithmes mimétiques.

B.2.1 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, ...etc. Cette méthode sera présentée en détail dans le chapitre deux.

B.2.2 Les colonies de fourmis

La méthode de la colonie 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, déposés 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.

Cette métaheuristique a été introduite pour la première fois [Colorni et al. 1992] et a été appliquée du voyageur de commerce. [Gambardella et al. 2003] ont appliqué cette métaheuristique pour le VRP.

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;
-

B.2.3 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 [Esqui 2001].

B.2.4 Les algorithmes mimétiques

Les algorithmes mimétiques ressemblent très fortement aux algorithmes génétiques 'classiques'. Ils ont été mis au point afin d'accélérer la convergence des algorithmes génétiques, réputée pour être lente. L'idée principale est d'inclure un mécanisme de recherche locale dans le déroulement de l'algorithme génétique, en remplaçant l'un de ses opérateurs génétiques. Pour cette raison, certains les classifient comme étant des algorithmes génétiques hybrides, faisant coopérer recherche locale et algorithme évolutionnaire. Ces algorithmes sont aussi parfois appelés 'recherche locale génétique' (genetic local search).

Le premier à avoir employé le terme d'algorithme mimétique est Moscato en 1989 [Mos 1989] pour un algorithme optimisant le problème de voyageur de commerce. Une bonne introduction aux algorithmes mimétiques. Ainsi que quelques applications, peuvent être trouvées dans [CDG 1999].

1.6 Conclusion

Dans ce chapitre nous avons présenté le problème du VRP. Passant de sa formulation mathématique et ses variantes à son état de l'art, nous avons énuméré les différentes méthodes de résolution qui ont été utilisées pour le résoudre. Le chapitre suivant va détailler une variante de VRP qui est le problème de transport à la demande.

CHAPITRE 2

Le problème de transport à la demande

2.1 Introduction

Le problème que nous traitons dans ce travail est un cas particulier du problème de tournée de véhicule qui est le transport à la demande, il est destiné aux clients souhaitant être transportés depuis des lieux d'origine vers des lieux de destination.

Dans ce qui suit, on va donner une présentation de ce problème, puis sa formulation mathématique, et une classification de ses variantes, et en termine par un état de l'art de quelques travaux réalisés sur les problèmes transport à la demande.

2.2 Présentation du problème de transport à la demande « *Dial-A-Ride Problem* »

Le problème de collecte et livraison, ou VRPPD « *Vehicle Routing Problem with Pickup and Delivery* », est une variante du VRP, il constitue une classe importante de ces problèmes dans lesquels les objets ou les gens doivent être transportés entre les origines et leurs destinations.

Ces problèmes, qui ont été étudiés pendant plus de 30 années, surgissent dans beaucoup de contextes tels que la logistique, les services ambulatoires, et la robotique. Nous pouvons aussi rencontrer ce problème dans notre vie quotidienne comme par exemple les camions de transfert de fonds, le ramassage scolaire, le transport de personnel, la poste, le ramassage des déchets ...

Le problème de transport à la demande, ou DARP « *Dial-A-Ride Problem* » est un cas spécial du VRPPD dans lequel les « objets » sont des clients devant être transportés de leurs origines à leurs destinations. Le DARP peut avoir plusieurs contraintes concernant des fenêtres de temps, une borne sur le temps maximal de trajet pour chaque objet O à transporter (Riding time), noté L_0 , et la qualité de service [Cordeau et al. 2003]. L'application principale du DARP est le service « porte-à-porte » de transport offert aux personnes âgées et aux personnes à mobilité réduite dans de nombreuses villes. Pour le DARP, les objets transportés sont un groupe de personnes.

Ayant un ensemble de véhicules de même capacité ou non, il s'agit de trouver une tournée (ordonnancement) réalisable et de qualité qui permet de satisfaire toutes les demandes des clients en respectant les différentes contraintes de capacité, du temps et de précédences qui peuvent être les suivantes :

- *Les contraintes de capacité* : sont dues aux capacités limitées des véhicules, ces capacités peuvent être impossibles à dépasser et la contrainte de capacité sera une

contrainte rigide, comme elles peuvent être dépassées tout en ayant, dans ce cas, une pénalité à assumer.

- *Une contrainte de temps total* : Le temps total d'une tournée ne doit pas dépasser une borne T . Ce temps est la somme des temps des voyages entre les sommets et les temps d'arrêt à ces derniers.
- *Les contraintes de précédences* : sont présentes pour garantir qu'un site de destination ne soit pas visité avant le site d'origine d'une même demande.

Vu que le VRP est un problème NP-difficile, le VRPPD ainsi que le DARP étant des extensions de ce dernier à laquelle sont additionnées les différentes contraintes sont aussi des problèmes NP-difficile.

2.3 Formulation mathématique

Nous présentons ici une formulation mathématique du DARP, nous supposons alors que :

- Une demande est représentée par un nœud d'origine et un nœud de destination.
- Un nœud n'est servi que par un seul véhicule et en une seule fois.
- Il y a un seul dépôt.
- Les contraintes de capacité doivent être respectées.
- Chaque véhicule commence le trajet du dépôt et y retourne à la fin.
- Le véhicule reste à l'arrêt à un nœud le temps nécessaire pour le traitement de la demande.
- Le temps total d'une tournée ne doit pas dépasser la borne T .

Une formulation générale du DARP est donnée par [Savlesbergh 1995a] comme suit :

Soient les variables suivantes :

N = L'ensemble des nœuds d'origine, destination et dépôt.

N' = L'ensemble des nœuds d'origine, destination.

N^+ = L'ensemble des nœuds d'origine.

N^- = L'ensemble des nœuds de destination.

K = Le nombre des véhicules.

d_{ij} = la distance euclidienne entre le nœud i et le nœud j , si $d_{ij} = \infty$ alors le chemin entre i et j n'existe pas (impasse, rue piétonne).

t_{ij}^k = le temps mis par le véhicule k pour aller du nœud i au nœud j .

q_i = la quantité traitée au nœud i , si $q_i > 0$ alors le nœud i est un nœud d'origine, si $q_i < 0$ le nœud est un nœud de destination.

Q_k = la capacité du véhicule k .

$i = 0..N$: indice des nœuds prédécesseurs.

$j = 0..N$: indice des nœuds successeurs.

$k = 0..K$: indice des véhicules.

Les variables de décision

$$x_{ij}^k = \begin{cases} 1 & \text{Si le véhicule } k \text{ voyage du nœud } i \text{ au nœud } j \\ 0 & \text{Sinon} \end{cases}$$

D_i : Le temps de départ du nœud i .

y_i^k : La quantité présente dans le véhicule k visitant le nœud i .

La fonction à optimiser

Les fonctions à optimiser diffèrent d'un problème à un autre, le critère le plus utilisé dans ce genre de problème est la minimisation de la distance totale parcourue par un ensemble minimum de véhicules.

$$\text{Minimiser} \left(\alpha_1 K + \alpha_2 \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij}^k \right) \quad (2.1)$$

Où α_1 et α_2 sont des coefficients de pondération et de mise à l'échelle.

Sous contraintes

$$\sum_{i=1}^n \sum_{k=1}^m x_{ij}^k = 1 \quad j = 2, \dots, n \quad (2.2)$$

$$\sum_{j=1}^n \sum_{k=1}^m x_{ij}^k = 1 \quad i = 2, \dots, n \quad (2.3)$$

$$\sum_{i \in N} x_{i0}^k = 1 \quad \forall k \in K; \quad (2.4)$$

$$\sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K \quad (2.5)$$

$$\sum_{i \in N} x_{iu}^k - \sum_{j \in N} x_{uj}^k = 0 \quad \forall k \in K; \forall u \in N \quad (2.6)$$

$$x_{ij}^k = 1 \Rightarrow y_i^k = y_i^k + q_i \quad \forall i, j \in N; \forall k \in K \quad (2.7)$$

$$y_0^k = 0 \quad \forall k \in K; \quad (2.8)$$

$$Q_k \geq y_i^k \geq 0 \quad \forall i \in N; \forall k \in K \quad (2.9)$$

$$x_{ij}^k = 1 \Rightarrow D_i + t_{ij}^k \leq D_j \quad \forall i, j \in N; \forall k \in K \quad (2.10)$$

$$D_w \leq D_v \quad \forall i \in N; w = N_i^+, v = N_i^- \quad (2.11)$$

$$D_0 = 0 \quad (2.12)$$

La fonction objective (2.1) consiste à minimiser le coût total de transport.

Les équations (2.2) et (2.3) assurent que chaque sommet ne soit servi qu'une seule fois par un et un seul véhicule.

Les équations (2.4) et (2.5) assurent le non dépassement de la disponibilité d'un véhicule. Un véhicule ne sort de dépôt et n'y revient qu'une seule fois.

L'équation (2.6) assure la continuité d'une tournée par un véhicule : le sommet visité doit impérativement être quitté.

Les équations (2.7), (2.8), (2.9) et (2.10) assurent le non dépassement de la capacité de transport d'un véhicule.

Les équations (2.11) et (2.12) assurent le respect de précédences.

2.4 Classification des problèmes de transport à la demande

Les auteurs classent ces problèmes selon deux critères [Savelsberg 1995 b], qui sont :

- Le nombre des véhicules.
- La disponibilité de l'information sur les demandes.

2.4.1 Selon le nombre des véhicules

Il existe deux types :

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

2.4.2 Selon la disponibilité de l'information sur les demandes

Il existe deux types :

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

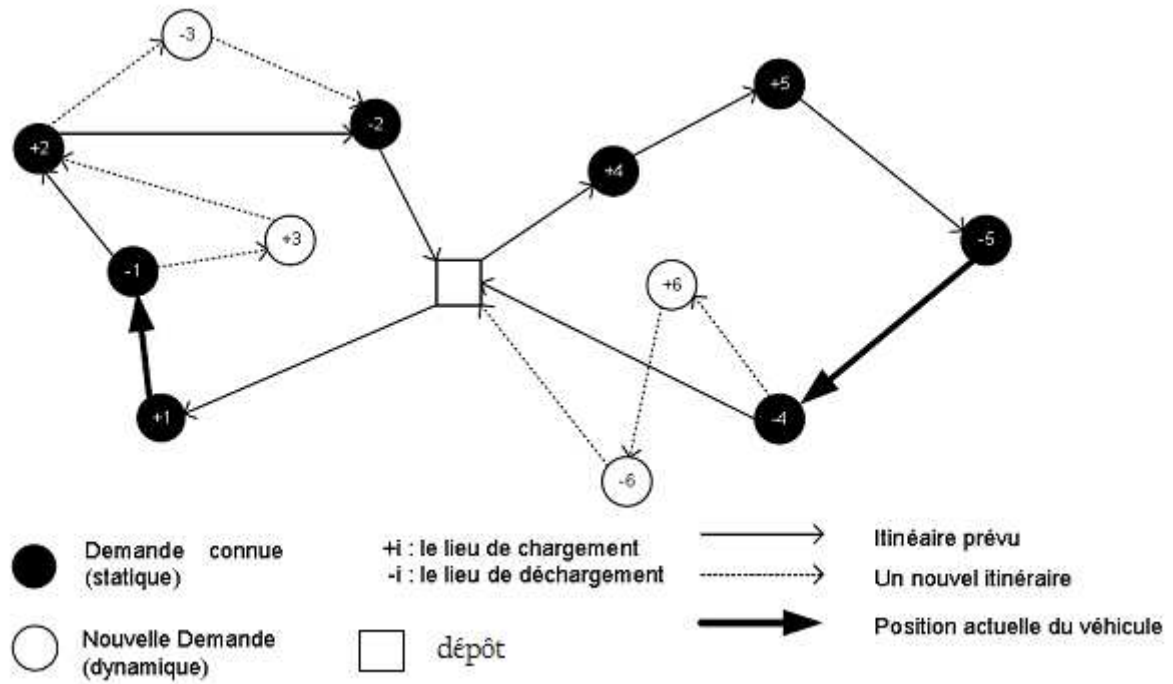


Fig.2.1 Le DARP dynamique avec 4 clients connus et 1 client à insérer dynamiquement.

Remarque : certains auteurs considèrent la fenêtre de temps comme un critère de classification pour certains types cités ci-dessus.

2.5 Etat de l'art des problèmes de transport à la demande « DARP »

2.5.1 DARP avec un seul véhicule

Dans le cas d'un seul véhicule, la plus part des travaux que j'ai trouvé sont destinés au problème de collecte et livraison VRPPD, car l'utilisation d'un seul véhicule est plus adapté au transport de marchandises que de personnes.

Une des premières études sur le DARP a été effectuée par Psaraftis [Psaraftis 1980] qui a considéré le cas avec un seul véhicule. Dans ce problème, les clients souhaitent être desservis aussitôt que possible et l'objectif est de minimiser une combinaison pondérée du temps total de service et de l'insatisfaction des clients. D'abord, un algorithme exact de programmation dynamique ($O(n^2 3^n)$) a été présenté pour le cas statique. Pour le cas dynamique, quand une nouvelle demande arrive, l'instance statique est mis à jour et ré-optimisée en fixant la tournée partielle déjà construite. En raison de la complexité de calcul de l'algorithme statique, seules les petites instances peuvent être résolues.

Sexton et Bodin [Sexton et al. 1985 a] [Sexton et al. 1985 b] ont proposé un algorithme heuristique basé sur la décomposition de Benders pour le DARP avec un seul véhicule. Dans leur problème, les clients spécifient une date souhaitée de déchargement et le retard n'est pas permis. L'objectif est de minimiser les insatisfactions des clients.

2.5.2 DARP avec plusieurs véhicules

De nombreux problèmes appartenant à la famille des DARP avec plusieurs véhicules ont été étudiés. Dans l'article de Cordeau et Laporte [Cordeau et al. 2003], les auteurs considèrent ce type de problème avec des fenêtres de temps pour les demandes, des capacités propres à chaque véhicule, et des durées de trajets maximales aussi bien pour les personnes que pour les véhicules. Ils proposent une recherche Tabou pour résoudre ce problème. Durant la recherche de solutions, des solutions irréalisables (qui violent la contrainte des capacités du véhicule ou des fenêtres de temps) peuvent être explorées. L'évaluation des solutions est une combinaison linéaire entre la somme des coûts de chaque route et de fonctions évaluant le niveau des contraintes violées. Les coefficients de la combinaison linéaire affectés à ces fonctions augmentent avec le nombre d'itérations de manière à obtenir une solution réalisable à la fin de l'algorithme. Sachant qu'une solution est représentée par un ensemble de couples : numéro de véhicule et numéro de demande de transport, l'opérateur de voisinage consiste à changer le numéro de véhicule d'un couple. Cette opération revient à supprimer un sommet d'une route et à l'ajouter dans une autre route selon une procédure basée sur le calcul du "forward time slack" d'un sommet [Savelsbergh 1992]. Tout le voisinage d'une solution n'est pas exploré entièrement, une règle s'appuyant sur des sommets critiques permet de diminuer l'ensemble des solutions à explorer. Enfin, en plus du critère d'aspiration (rendre possible l'exploration de solution tabou si elle améliore la meilleure solution trouvée), la recherche tabou possède une phase de diversification (forcer la recherche de solutions vers d'autres voisinages pas encore ou peu explorés).

Une autre étude se basant sur les mêmes contraintes du problème précédent, excepté le temps de trajet maximal, a été menée par Rekiek et al. [Rekiek et al. 2006] dans un contexte de transport de personnes handicapées. La problématique est la suivante : étant donné plusieurs dépôts avec pour chacun une zone de couverture des demandes, l'objectif est de satisfaire toutes les demandes en minimisant le nombre de véhicules. Les auteurs proposent un algorithme génétique pour résoudre ce problème. Chaque solution est décomposée en plusieurs gènes caractérisés par un véhicule et les sommets des demandes qu'il dessert. L'opérateur de croisement consiste à prendre aléatoirement une sous séquence de gènes d'un

parent afin de l'insérer aléatoirement dans la séquence d'un autre parent, puis d'appliquer un opérateur de réparation. Ce dernier élimine les sommets doublons et réinsère les sommets manquants. L'opérateur de mutation permet soit de créer aléatoirement des nouveaux groupes de véhicules, soit d'en éliminer, soit d'échanger les sommets entre gènes (véhicules). Une procédure regroupant les gènes de bonne qualité en les déplaçant est aussi implémentée afin d'augmenter la chance de les diffuser dans les individus de la génération suivante. La fonction objectif de ce problème est de maximiser la moyenne au carré de "l'efficacité des véhicules", calculée comme un ratio entre la durée réelle de la route d'un véhicule sans compter les attentes et la durée maximum de la route.

Parmi les méthodes de résolutions performantes, il existe les méthodes à deux phases comme par exemple celle proposée par Dumas et al. [Dumas et al. 1989]. La première phase consiste à créer des groupes de personnes dans la même région à servir approximativement au même moment. Ces groupes sont ensuite combinés pour former le parcours des véhicules en se basant sur une technique de génération de colonnes. Enfin, la deuxième phase réordonne chaque route des véhicules en utilisant des algorithmes de la littérature avec un unique véhicule. Les auteurs ont réussi à résoudre facilement des instances avec 200 personnes à transporter, cependant les grandes instances exigent l'utilisation d'une technique de décomposition spatiale et temporelle des données pour éviter un temps de résolution trop important. La phase de formation des groupes a ensuite été reprise et améliorée par Desrosiers et al. [Desrosiers et al. 1991].

Borndörfer et al. [Borndörfer et al. 1997] s'intéressent à ce même type de problème avec un contexte de transports de personnes handicapées (nommé Telebus Berlin). Ils proposent une heuristique basée sur une approche de partition d'ensembles en deux étapes. La première étape consiste à regrouper les demandes de transport tel que le véhicule associé au traitement de ces demandes contienne toujours au moins une personne lors de son trajet. L'objectif de cette étape est de réduire la taille des instances du problème en utilisant au maximum les capacités des véhicules. Ces ensembles de demandes sont construits en minimisant les distances des trajets des véhicules. Ce problème de partitionnement d'ensembles est résolu de manière optimale. La seconde étape consiste tout d'abord à construire un chaînage de chaque demande d'un même groupe de manière qu'un véhicule puisse assurer cette route en respectant toutes les contraintes. Ensuite, un autre problème de partitionnement d'ensembles, résolu aussi de manière optimale, intervient. Celui-ci a pour but de trouver le meilleur sous-ensemble de routes parmi toutes les routes réalisables énumérées

par combinaison des groupes, en répondant à toutes les demandes et en minimisant les distances parcourues par les véhicules. Les deux problèmes de partitionnement d'ensembles sont résolus à l'aide d'une procédure par séparation et évaluation.

Dans un article de Xiang et al. [Xiang 2006], une autre approche est présentée pour la résolution d'un DARP avec fenêtres de temps, durées de trajet maximales pour les véhicules et les personnes, et différents types de véhicules (chaque type de véhicule ne peut desservir qu'un ensemble de demandes). En plus, les auteurs considèrent encore d'autres contraintes concernant les chauffeurs : temps maximal de travail dans une journée, temps de pause obligatoire, et compétence à pouvoir conduire un type de véhicule. Les auteurs définissent un temps d'attente pour chaque personne en fonction de l'arrivée du véhicule et de la fenêtre de temps au sommet départ de la demande. Leur fonction objectif est de minimiser la somme des coûts suivants : les coûts des trajets, les coûts kilométriques, les coûts d'attentes des personnes, et les coûts de temps des transports. Ces coûts prennent en compte les différents types de véhicules et le temps de travail des conducteurs. L'heuristique mise au point par les auteurs pour résoudre ce problème est basée sur le principe d'une recherche locale. Elle est traditionnellement constituée des étapes suivantes : prétraitement, construction d'une solution initiale, recherche de la meilleure solution dans un voisinage de la solution courante avec diversification puis intensification, et si la solution trouvée peut être encore améliorée alors on retourne à l'étape de recherche dans le voisinage. L'étape de prétraitement consiste à réduire l'espace de solutions en déduisant des circuits impossibles en fonction des fenêtres de temps, des distances, des types de véhicules, etc. Puis une première solution est construite en deux phases. La première phase ordonnance tous les sommets en fonction des fenêtres de temps. Et la seconde phase groupe ces sommets en différents trajets de manière à respecter les contraintes des fenêtres de temps dans un premier temps, puis des autres contraintes dans un deuxième temps (comme ceux des temps de travail des conducteurs). L'étape suivante est une recherche locale avec une diversification pour sortir des optima locaux. La recherche locale est basée sur les opérateurs suivants : suppression d'une ou deux demandes de transports dans un même trajet et insertion de la ou les demandes dans un autre trajet, ou échange de deux demandes entre deux trajets. Ces opérateurs sont appliqués successivement jusqu'à ce qu'ils ne puissent plus améliorer la solution courante. La phase de diversification change le mécanisme de recherche en fonction d'une deuxième fonction objectif s'appuyant sur les temps d'attentes des personnes uniquement. Enfin, la dernière étape consiste en une intensification de la recherche en échangeant des sous-ensembles de trajets entre véhicule.

Wong et al. [Wong et al. 2006] abordent le même type de problème. L'objectif est de minimiser une combinaison linéaire entre le temps total de l'opération, le temps de trajet pour chaque personne, et des coûts fixes par demande de transport non assurée. La raison pour laquelle les véhicules, ne sont pas tous identiques, provient de la possibilité d'accueillir des fauteuils roulants dans certains véhicules. Les auteurs proposent une procédure d'insertion parallélisée pour résoudre ce problème. Dans un premier temps, les demandes de transports sont toutes classées en fonction d'un indice. Cet indice mesure pour chaque demande les difficultés d'insertion dans la route d'un véhicule et les désagréments causés aux autres demandes assurées par ce véhicule si l'insertion est effectuée. Dans un deuxième temps, ces insertions dans les routes de chaque véhicule sont réalisées en fonction des indices de telle sorte que les demandes les plus difficiles à placer soient les premières à être insérées. Puis, une phase de post optimisation a été implémentée de manière à améliorer la solution en échangeant des demandes entre différentes routes. Les auteurs ont montré que la méthode proposée est efficace par rapport aux algorithmes classiques de la littérature.

Parmi les méthodes souvent utilisées pour résoudre ce type de problème, l'algorithme génétique connaît un grand succès. Par exemple Cubillos et al. [Cubillos et al. 2007] présentent plusieurs algorithmes génétiques pour le DARP. Leur fonction objectif est de minimiser le coût engendré par les véhicules (temps des trajets, kilomètres parcourus, etc.) et la qualité du service de transport aux personnes (temps d'attente, temps supplémentaire du transport, etc.). Le codage de l'individu, ou solution, est une liste ordonnée de passagers pour chaque véhicule sachant qu'un passager est composé de deux attributs : le sommet origine et le sommet de destination. Pour générer la population de départ, deux initialisations sont proposées : aléatoirement et par insertions successives. La sélection pour les générations suivantes s'effectue par tournois. Les opérateurs de croisement proposés sont soit le croisement en un point soit le "Partial Match Crossover" (similaire à un croisement en deux points). Ils utilisent deux opérateurs de mutation couramment utilisés dans la littérature : "bit level" et "2-opt operator". Pour terminer, les auteurs exposent les résultats de différents algorithmes génétiques construits à partir de combinaisons entre les opérateurs proposés et les initialisations possibles.

Madsen et al. [Madsen et al. 1995] ont présenté un algorithme pour un cas réel du DARP dynamique multi-véhicules qui concerne jusqu'à 300 demandes par jour pour le transport des personnes âgées et handicapées, à Copenhague. Le problème a beaucoup de contraintes telles que des fenêtres de temps, des contraintes de capacité multidimensionnelle,

des priorités sur les clients et une flotte de véhicules hétérogènes. Beaucoup de critères de qualité, tenant compte de la satisfaction des clients et des coûts de service ont été considérés et mélangés par l'utilisation de paramètres de poids. Quand une nouvelle demande arrive, elle est insérée dans l'itinéraire courant en utilisant un algorithme efficace d'insertion basé sur celui de [Jaw et al. 1986]. Les résultats sur des instances réels avec un maximum de 300 demandes et 24 véhicules ont prouvé que l'algorithme est assez rapide pour être utilisé dans un contexte dynamique et qu'il est capable de produire des solutions de bonne qualité.

Horn [Horn 2002a] a développé un logiciel pour des services de transports à la demande tels que les taxis et les autobus à tournées variables. Son problème inclut des contraintes de fenêtres de temps pour les demandes dynamiques, des contraintes de capacité et des annulations de réservation. L'insertion de nouvelles demandes est faite par un schéma d'insertion à coût minimal qui est complété par une procédure de recherche locale exécutée périodiquement. L'auteur a développé une heuristique pour déplacer stratégiquement les véhicules à vide en tenant compte des tendances des futures demandes. Des expérimentations ont été effectuées et basées sur des données d'une société de taxis, en Australie.

Un algorithme parallèle pour le DARP dynamique a été développé par Attanasio et al. [Attanasio et al. 2004]. Au début de l'horizon de planification, une solution statique est construite à partir des demandes connues en utilisant un algorithme de recherche taboue. Quand une nouvelle demande arrive, chacun des threads parallèles insère la demande de façon aléatoire dans la solution courante et exécute la recherche taboue pour l'obtention d'une solution réalisable. Si une solution faisable est trouvée, la demande est acceptée et une phase de post-optimisation est exécutée. Des résultats expérimentaux correspondant à différentes stratégies de parallélisations sont décrits pour des exemples comprenant jusqu'à 144 demandes.

Un algorithme pour un problème de type DARP dynamique a été développé par Coslovich et al. [Coslovich et al. 2006]. Dans ce problème, un conducteur peut recevoir une demande de voyage d'une personne située à un arrêt et doit décider rapidement s'il l'accepte ou pas. L'algorithme maintient un stock de solutions réalisables qui sont compatibles avec les tournées partielles déjà effectués. Quand une nouvelle demande arrive, un algorithme efficace d'insertion s'efforce d'insérer cette demande dans au moins une des solutions. La demande acceptée seulement si l'algorithme d'insertion réussit. Une fois la nouvelle demande est acceptée, le stock des solutions faisables est mis à jour. Ce procédé est exécuté tandis que le véhicule se déplace entre deux endroits. Les temps de réponse ne sont pas pris en compte, ni

les contraintes de capacité. Des résultats sur des instances artificielles comprenant jusqu'à 50 demandes sont décrits.

Beaudry et al. [Beaudry et al. 2008] ont développé un algorithme en deux phases pour résoudre un DARP dynamique complexe qui se présente dans le cas du transport des patients dans les hôpitaux. Ce problème comporte plusieurs contraintes portant sur la prise en compte de demandes avec différents degrés d'urgence et des besoins en équipement, des modes multiples de transport des patients, et des fenêtres de temps. De nouvelles demandes sont insérées en utilisant un schéma d'insertion rapide, capable de répondre aux exigences en temps réel. La deuxième phase consiste en une recherche taboue qui essaye d'améliorer la solution courante. L'algorithme a réussi à réduire les délais d'attente pour les patients tout en utilisant moins de véhicules sur les données réelles d'un hôpital allemand.

Xiang et al. [Xiang 2008] ont étudié un DARP dynamique sophistiqué dans lequel les temps de transport et de service ont un composant stochastique. En outre, il peut y avoir des non-présentations ou des annulations et les véhicules peuvent être bloqués dans des embouteillages ou des pannes. De nouvelles demandes sont insérées dans les tournées établies au moyen d'une procédure de recherche locale basée sur des mouvements simples entre tournées. La recherche locale a été complétée avec une stratégie de diversification qui utilise un objectif secondaire en mesurant la période des véhicules à vide. L'algorithme a été évalué par simulation sur plusieurs instances impliquant jusqu'à 610 demandes.

Cordeau et al. [Cordeau et al. 2007] ont présenté le problème « Dial-a-Flight » dans lequel l'objectif est de modéliser et d'optimiser les services d'accès à des vols sans réservation qui sont proposés par près de 3000 sociétés dans les États-Unis. Dans ce problème, les demandes se composent généralement de chargements multiples et de paires de déchargements sur lesquels des règles strictes concernant le service sont imposées. Les auteurs discutent des caractéristiques du problème et présentent quelques approches de résolution.

2.6. Conclusion

Dans ce chapitre nous avons présenté le problème de transport à la demande « Dial-A-Ride problem », après une description du problème, on a présenté sa formulation mathématique, puis une classification de ses variantes, et on a terminé par un état de l'art des travaux effectués dans le domaine.

CHAPITRE 3

Les Algorithmes génétiques

3.1 Introduction

Les premiers travaux sur les algorithmes génétiques (AG) ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland [Hol 1975], sur la base des travaux précédents, développe les principes fondamentaux des AG dans le cadre de l'optimisation mathématique. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation de la méthode sur des problèmes réels de grande taille. La parution de l'ouvrage de référence écrit par Goldberg [Gold 1989b] décrivant l'utilisation des AG dans le cadre de la résolution des problèmes a permis de marquer le début d'une nouvelle approche de résolution intéressante.

Les AG s'attachent à simuler le processus de sélection naturelle dans un environnement hostile lié au problème à résoudre [Dav 1991]. 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, etc.

Dans le cadre de l'optimisation mono-objectif, chaque individu représente un point dans l'espace d'état auquel on associe la valeur du critère à optimiser. On génère ensuite une population d'individus aléatoirement, pour laquelle l'algorithme génétique s'attache à sélectionner les meilleurs individus tout en assurant une exploration efficace de l'espace d'état.

Les applications des AG sont diverses et variées; c'est ainsi qu'ils ont donné de bons résultats dans des problèmes d'ordonnancement [KBH 1994], de contrôle adaptatif [FW 1993], dans le problème du voyageur de commerce [HSL 1993], dans les problèmes de transport [YG 1991], dans la synthèse de forme [WO 1993], dans les réseaux de neurones [SRD 1993], dans la synthèse moléculaire [UM 1993], dans le domaine médical [FSJP 1993] etc.

Dans ce chapitre, nous introduisons les principes de base des AG, leur terminologie, les opérateurs participants à l'exploration de l'espace de recherche tout en mettant l'accent sur leur utilité, leur conception et leur mode de fonctionnement.

3.2 Terminologie

Nous allons maintenant donner un certain nombre de définitions de vocabulaire nécessaires pour une bonne compréhension. En effet, la communauté scientifique utilisant les algorithmes génétiques (AG) utilise un nombre d'analogies issues du monde biologique pour illustrer des idées.

3.2.1 Gène : Un gène est une suite de bases qui contient le code d'une protéine donnée. On appellera *gène* la suite de symboles qui codent la valeur d'une variable. Dans le cas général, un *gène* correspond à un seul symbole (0 ou 1 dans le cas binaire).

3.2.2 Chromosome : Un *chromosome* est constitué d'une séquence finie de *gènes* qui peuvent prendre des valeurs appelées *allèles* qui sont prises dans un alphabet qui doit être judicieusement choisi pour convenir du problème étudié.

3.2.3 Individu : On appellera *individu* une des solutions potentielles. Dans la plupart des cas, un *individu* sera représenté par un seul *chromosome*, dans ce cas, par abus de langage, on utilisera indifféremment *individu* et *chromosome*.

3.2.4 Fitness d'un individu : Nous appelons *fitness* d'un *individu* toute valeur positive que nous noterons $f(i)$, où f est typiquement appelée fonction de *fitness*. Cette fonction est déterminée en fonction du problème à résoudre pour mesurer la qualité d'un individu.

3.2.5 Population : On appellera *population* l'ensemble des solutions potentielles qu'utilise l'AG.

3.3 C'est quoi un algorithme génétique ?

L'algorithme génétique est une méthode de résolution des problèmes qui utilise la génétique comme un modèle de résolution de problème. C'est une technique de recherche pour trouver des solutions approximatives pour les problèmes d'optimisation.

3.4 A quoi sert l'algorithme génétique ?

L'algorithme génétique résout des problèmes dont la solution exacte, si elle est connue, est trop compliquée pour être calculée en un temps raisonnable. Ceci dit, face à un problème pour lequel il existe pour ainsi dire une infinité de solutions, plutôt que d'essayer naïvement toutes les solutions une à une pour trouver la meilleure, on va explorer l'espace des solutions en se laissant guider par les principes des algorithmes génétiques.

3.5 Principes des algorithmes génétiques

Les opérations successives utilisées dans les algorithmes génétiques sont décrites sur la figure 3.1.

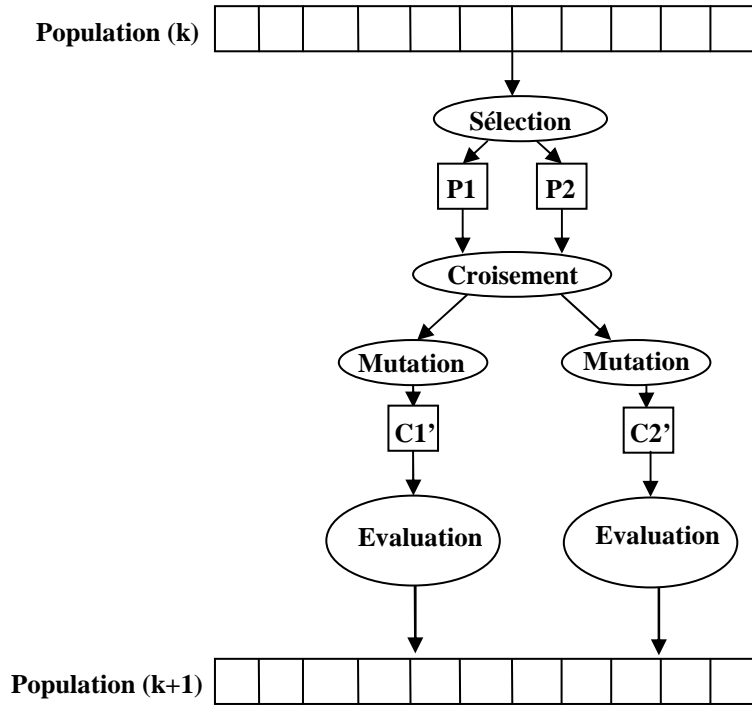


Fig 3.1 Principe générale des algorithmes génétiques

1. On commence par générer une population d'individus d'une façon aléatoire.
2. Deux parents sont ensuite sélectionnés (P1 et P2) en fonction de leurs adaptations. On applique aléatoirement l'opérateur de croisement avec une probabilité P_c qui génère deux enfants C1 et C2. Si $P_c = 0.6$, dans 60 pour cent des cas on appliquera l'opérateur de croisement sur les individus sélectionnés P1 et P2. On modifie ensuite certains gènes de C1 et C2 en appliquant l'opérateur de mutation avec la probabilité P_m , ce qui produit deux nouveaux individus C'_1 et C'_2 pour lesquels on évalue le niveau d'adaptation avant de les insérer dans la nouvelle population. On réitère les opérations de sélection, de croisement et de mutation afin de compléter la nouvelle population, ceci termine le processus d'élaboration d'une génération.

On reboucle ensuite N fois sur l'étape 2 (N, qui est un paramètre, représente le nombre totale de générations).

3.6 Pseudo code d'un Algorithme génétique

Le pseudo code de l'algorithme génétique est donné dans l'algorithme 3.1 :

Algorithme 3.1 Un algorithme génétique

1. **Entrée** : Instance du problème
 2. **Sortie** : Une solution
 3. Initialisation de la Population ;
 4. Evaluation de la population;
 5. **Tant que** (Condition d'arrêt non satisfaite)
 6. Sélection ;
 7. Croisement;
 8. Mutation ;
 9. Remplacement ;
 10. **Fin tan que**
-

3.7 Conception d'un algorithme génétique

La simplicité de mise en œuvre et l'efficacité constituent deux des caractéristiques les plus attrayantes de l'approche proposée par les AGs. La mise en œuvre d'un algorithme génétique sollicite la disponibilité :

- d'une *représentation génétique* du problème, c'est-à-dire un codage approprié des solutions sous la forme de chromosomes. Cette étape associe à chacun des points de l'espace de recherche une même structure de données qui synthétise toutes les informations liées à ces derniers. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques (voir la figure 3.2) ;
- d'un *mécanisme de génération* de la population initiale. Ce mécanisme doit être capable de produire une population non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut prendre plus ou moins rapidement la convergence vers l'optimum globale. Dans le cas où l'on ne connaît rien sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche ;

- d'une *fonction d'évaluation* pour mesurer la force de chaque chromosome ;
- d'un *mode de sélection* des chromosomes à reproduire ;
- des *opérateurs* permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace de recherche ;
- des *valeurs* pour les *paramètres* qu'utilise l'algorithme : taille de la population, nombre totale de générations ou critère d'arrêt, probabilités de croisement et de mutation.

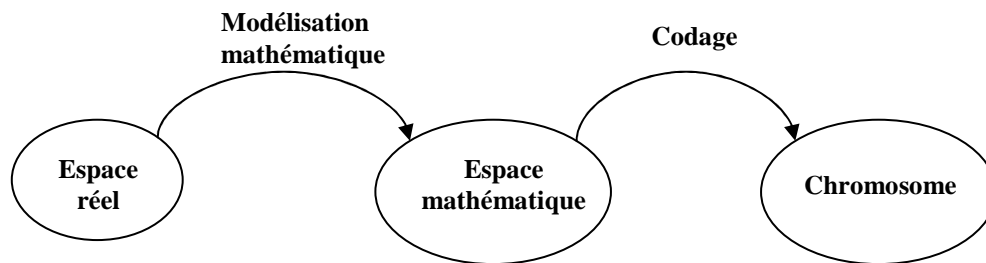


Fig. 3.2 Les phases de définition d'un codage

3.8 Variantes

Il existe beaucoup de variantes possibles suivant la représentation choisie, les opérateurs de croisement, de mutation et de sélection. La section suivante représente les choix les plus courants qui définissent les variantes.

3.8.1 Codage

Le codage est un processus de représentation des gènes. Le processus peut être effectué par utilisation des : bits, nombres, arbres, tableaux, listes ou tous autres objets. La littérature définit deux types de codage : binaire et réel.

a. Codage binaire

La manière la plus commune du codage est une chaîne binaire (voire la figure 3.3), dont chaque bit dans la chaîne peut représenter certaines caractéristiques de la solution. Chaque

chaîne binaire est donc une solution mais pas nécessairement la meilleure. Une autre possibilité est que la chaîne entière peut représenter un nombre.

Chromosome1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome2	1 0 0 1 1 0 0 1 1 0 1 1

Fig. 3.3 Codage binaire

b. Codage réel

Pour certain problème d'optimisation, il est plus pratique d'utiliser un codage réel des chromosomes. Un gène est ainsi représenté par un nombre réel au lieu d'avoir à coder les réels en binaire puis de les décoder pour les transformer en solutions effectives. Le codage réel permet d'augmenter l'efficacité de l'algorithme génétique et d'éviter des opérations de décodage supplémentaires. En effet, un chromosome codé en réels est plus court que celui codé en binaire. La figure 3.4 illustre ce type de codage.

Chromosome1	1 5 3 2 6 4 7 9 8
Chromosome2	8 5 6 7 2 3 1 4 9

Fig. 3.4 Codage réel

3.8.2 Population initiale

Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature. Le choix de l'initialisation se fera en fonction des connaissances que l'utilisateur a sur le problème. S'il n'a pas d'informations particulières, alors une initialisation aléatoire, la plus uniforme possible afin de favoriser une exploration de l'espace de recherche maximum, sera la plus adaptée. Par ailleurs, cette étape présente un problème principal qui est celui de choix de la taille de la population. En effet une population trop grande augmente le temps de

calcul et demande un espace mémoire considérable, alors qu'une population petite conduit à l'obtention d'un optimum local.

3.8.3 La fonction d'adaptation

La fonction d'adaptation, ou *fitness*, associe une valeur pour chaque individu. Cette valeur a pour but d'évaluer si un individu est mieux adapté qu'un autre à son environnement. Ce qui signifie qu'elle quantifie la réponse fournie au problème pour une solution potentielle donnée. Ainsi les individus peuvent être comparés entre eux.

Cette fonction, propre au problème, est souvent simple à formuler lorsqu'il ya peu de paramètres. Au contraire, lorsqu'il ya beaucoup de paramètres ou lorsqu'ils sont corrélés, elle est plus difficile à définir. Dans ce cas, la fonction devient une somme pondérée de plusieurs fonctions. Un ajustement des coefficients est nécessaire.

3.8.4 Sélection

La prochaine étape est de décider comment effectuer la sélection c'est-à-dire, comment choisir des individus à partir de la population qui créera les enfants pour la génération suivante.

La sélection est une méthode qui sélectionne aléatoirement des chromosomes à partir de la population en fonction des valeurs de la fonction d'adaptation. Ce procédé permet de donner aux meilleures chaînes, une probabilité de contribuer à la génération suivante. Cet opérateur est bien entendu une version artificielle de la sélection naturelle qui est la survie darwinienne des chaînes les plus adaptées.

Il existe de nombreuses techniques de sélection, les plus courantes seront évoquées dans la section suivante.

a. La sélection par roulette

La sélection par roulette consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné.

Si la population d'individus est de taille égale à N , alors la probabilité de sélection d'un individu x_i notée $p(x_i)$ est égale à : $p(x_i) = \frac{f(x_i)}{\sum_{k=1}^N f(x_k)}$ En pratique, on calcul pour chaque

individu x_i sa probabilité cumulée $q_i = \sum_{j=1}^i p(x_j)$ et on choisie aléatoirement un nombre r compris entre 0 et 1.

L'individu retenu est x_1 si $q_1 \geq r$ ou x_i ($2 \leq i \leq N$) si $q_{i-1} < r \leq q_i$. Ce processus est répété N fois. Avec une telle sélection, un individu fort peut être choisi plusieurs fois. Par contre, un individu faible a moins de chance d'être sélectionné.

b. La sélection par rang

La sélection par rang trie d'abord la population par fitness. Chaque chromosome se voit associé un rang en fonction de sa position. Le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). La sélection par rang d'un chromosome est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais.

c. La sélection par tournoi

Elle consiste à choisir aléatoirement deux ou plusieurs individus et à sélectionner le plus fort. Ce processus est répété plusieurs fois jusqu'à l'obtention de N individus. L'avantage d'une telle sélection est d'éviter qu'un individu très fort soit sélectionné plusieurs fois.

3.8.5 Croisement

La naissance d'un nouvel individu, nécessite la prise aléatoire d'une partie des gènes de chacun des deux parents. Ce phénomène, issu de la nature est appelé *croisement* (crossover).

Le croisement est le processus de prendre deux parents et de produire à partir d'elles des enfants. Il s'agit d'un processus essentiel pour explorer l'espace des solutions possibles.

La littérature définit plusieurs opérateurs de croisement. Ils diffèrent selon le type de codage adapté et la nature du problème traité.

a. Croisement binaire

Ce croisement peut avoir recourt à plusieurs types en occurrence :

a.1 Croisement en 1-point

C'est le croisement le plus simple et le plus connu dans la littérature. Il consiste à choisir au hasard un point de croisement pour chaque couple de chromosomes. Les sous-chaînes situées après ce point sont par la suite inter-changées pour former les fils (voire la figure 3.5).

Parent1	1 1 0 1 0 0 0 1 1 0 1 0
Parent2	1 0 0 1 1 0 0 1 1 0 1 1
Fils 1	1 1 0 1 0 0 0 1 1 0 1 1
Fils 2	1 0 0 1 1 0 0 1 1 0 1 0

Fig 3.5 Croisement en un point de deux chromosomes

a.2 Croisement en 2-points

Dans ce type de croisement, deux points de coupure sont choisis au hasard et le contenu entre ces points est inter-changé pour former les fils (voire la figure 3.6).

Parent1	1 1 0 1 0 0 0 1 1 0 1 0
Parent2	1 0 0 1 1 0 0 1 1 0 1 1
Fils 1	1 1 0 1 1 0 0 1 1 0 1 0
Fils 2	1 0 0 1 0 0 0 1 1 0 1 1

Fig 3.6 Croisement en 2-points de deux chromosomes

a.3 Croisement en n-points

Ce type de croisement s'énonce par un choix aléatoirement de n-points de coupure pour dissocier chaque parent en n+1 fragments. Pour former un fils, il suffit de concaténer alternativement n+1 sous chaînes à partir des deux parents.

a.4 Croisement uniforme

Il existe des versions distinctes de ce croisement. La plus connue consiste à définir de manière aléatoirement un masque, c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel parent le premier fils devra hériter du gène s'y trouvant; si face à un locus le masque présente un 1, le fils héritera le gène s'y trouvant du parent1, s'il présente un 0 il en héritera du parent2.

La création du fils2 se fait de manière symétrique: si pour un gène donné le masque indique que le fils1 devra recevoir celui-ci du parent1 alors le fils2 le recevra du parent2, et si le fils1 le reçoit du parent2 alors le fils 2 le recevra du parent1 (voire la figure 3.7).

Parent1	1 1 0 1 1 0 0 1 1 0 1 0
Parent2	0 1 0 0 1 1 1 1 0 0 1 1
Masque	1 1 0 0 1 1 0 0 1 0 0 0

Fils 1	1 1 0 0 1 0 1 1 1 0 1 1
Fils 2	0 1 0 1 1 1 0 1 0 0 1 0

Fig. 3.7 Croisement uniforme

a.5 Croisement avec trois parents

Dans cette technique, trois parents sont aléatoirement choisis. Chaque bit du premier parent est comparé avec le bit du deuxième parent. Si tous les deux sont les mêmes le bit est pris pour le résultat, sinon le bit du troisième parent est pris pour le résultat (voire figure 3.8).

Parent1	1 1 0 1 1 0 0 1 1 0 1 0
Parent2	0 1 0 0 1 1 1 1 0 0 1 1
Parent3	1 1 0 0 1 1 0 0 1 0 0 0
Fils	1 1 0 0 1 1 0 1 1 0 1 1

Fig. 3.8 Croisement avec trois parents

b. Croisement réel

Le codage réel requiert des opérateurs génétiques spécifiques pour la manipulation des chromosomes. Il est de plusieurs types :

b.1 L'opérateur de Croisement PMX (*Partially Mapped Crossover*)

PMX est un opérateur de croisement à deux points de coupure qui définit un segment de même longueur dans chacun des parents P1 et P2. Les segments sont indiqués avec une trame foncée dans la partie (a) de la Figure 3.9. Ces segments sont copiés chez les enfants E1 et E2. E1 a hérité du segment de P2 et E2 de P1, voire la Figure 3.9 partie (b).

A partir de l'un de ces segments, on établit à chaque gène une liste de correspondance. Cette liste va servir à placer les gènes redondants et elle est formée de la manière suivante: pour chaque position du segment on note x le gène qui s'y trouve et y celui de l'autre enfant dans la même position. Tant que y est retrouvé ailleurs dans le segment de départ, on note y' son correspondant dans l'autre enfant et on remplace y par y' . Par exemple, le gène correspondant à "1" de E1 est "6" mais ce gène existe aussi dans E1 et son correspondant est "3". Ainsi dans la liste de correspondance, on note que "1" a pour correspondant "3". La liste complètement formée se trouve à la marge de la partie (b) du schéma.

On procède ensuite au placement des gènes hors segment en les copiant des parents respectifs. Par exemple, copier "1" de P1 dans E1 provoque un conflit puisque ce gène existe déjà. On utilise alors son correspondant dans la liste qui est "3". En procédant de manière itérative, on arrive à former les enfants E1 et E2 illustrés à la partie (c) de la Figure 3.9.

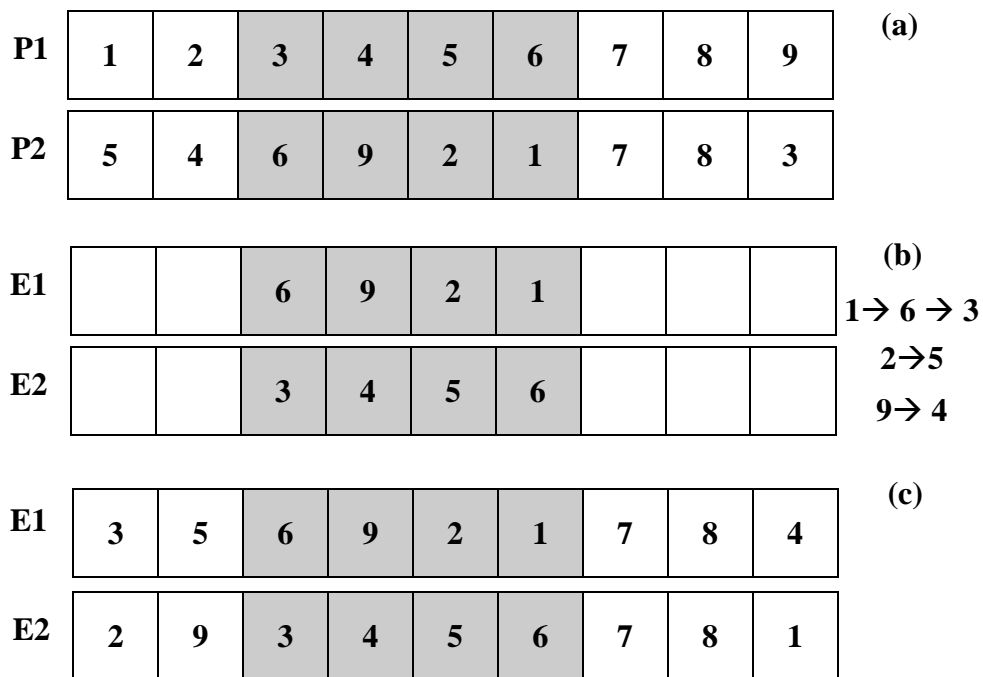


Fig 3.9 Opérateur de Croisement PMX

b.2 L'opérateur de Croisement CX (*Cycle Crossover*)

CX est un opérateur qui satisfait la condition suivante: chaque gène d'un enfant provient de l'un des parents à la même position. Les enfants sont donc formés en copiant un gène d'un parent et en éliminant l'autre à la même position puisqu'il va appartenir au deuxième enfant. Une fois que les positions occupées sont copiées par élimination, on a complété un cycle. Les places restantes des deux enfants sont complétées par les parents opposés.

Dans l'exemple présenté à la Figure 3.10 partie (a), la première position de E1 est attribuée à "1" provenant de P1. Le gène de P2 situé à la même position est "4", il est recherché dans P1 et retrouvé à la quatrième position. Le gène "4" est copié dans E1 et son correspondant "8" est recherché dans P1 et retrouvé à la huitième position. "8" est copié dans E1 à cette même position et son correspondant "3" est recherché dans P1 et retrouvé à la troisième position. "3" est copié dans E1 et son correspondant "2" est recherché dans P1 et retrouvé à la deuxième position. "2" est copié et son correspondant "1" est retrouvé dans P1 à la première position où cette position est déjà occupée dans E1 donc le cycle est terminé. Le cycle de E1 est "1, 4, 8, 3, 2" et de manière analogue, le cycle "4, 1, 2, 3, 8" est formé pour E2.

En dernier lieu, il faut combler les positions vacantes à partir des parents opposés. Par exemple, les gènes "7 ,6 ,9" de E1 proviennent de P2. On obtient ainsi les enfants illustrés dans la partie (b) de la Figure 3.10.

P1	1	2	3	4	5	6	7	8	9	(a)
P2	4	1	2	8	7	6	9	3	5	
E1	1	2	3	4				8		(b)
E2	4	1	2	8				3		1→4→8→3→2 4→1→2→3→8
E1	1	2	3	4	7	6	9	8	5	(c)
E2	4	1	2	8	5	6	7	3	9	

Fig 3.10 Opérateur de Croisement CX

b.3 L'opérateur de Croisement OX (*Order Crossover*)

OX est un opérateur avec deux points de coupure qui copie le segment formé par ces deux points de coupure dans les deux enfants. Par la suite, il recopie, à partir du deuxième point de coupure ce qui reste des gènes dans l'ordre du parent opposé en évitant les doublons.

La Figure 3.11 présente un exemple du déroulement du croisement avec l'opérateur OX. Dans la partie (a), un segment est formé par les points de coupure dans les deux parents et ces segments sont copiés tels quels dans les enfants E1 et E2 comme le montre la partie (b).

Enfin, on procède à la copie des gènes situés hors du segment copié. Pour cela, on se place à partir du deuxième point de coupure et on choisit les gènes non redondants provenant du parent opposé. Par exemple, dans la partie (c) de la Figure 3.11, on essaie de placer le gène "6" de P2 après le deuxième point de coupure dans E1 mais ce gène existe déjà à l'intérieur du segment. Il est donc ignoré et on passe au suivant. Le gène "2" ne présente pas de conflit, il

est donc copié et ainsi de suite jusqu'à former les deux enfants E1 et E2 tel qu'illustré à la Figure 3.11 partie (c).

P1	1	2	3	4	5	6	7	8	9	(a)
P2	5	7	4	9	1	3	6	2	8	
E1			3	4	5	6				(b)
E2			4	9	1	3				
E1	9	1	3	4	5	6	2	8	7	(c)
E2	5	6	4	9	1	3	7	8	2	

Fig. 3.11 Opérateur de Croisement OX

3.8.6 Mutation

C'est un processus où un changement mineur de code génétique est appliqué à un individu pour introduire de la diversité et ainsi d'éviter de tomber dans des optimums locaux. Différentes manières de mutation d'un chromosome sont aussi définies dans la littérature.

a. Mutation en codage binaire

Dans un algorithme génétique simple, la mutation en codage binaire est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un caractère de la chaîne.

b. Mutation en codage réel

Pour le codage réel, les opérateurs de mutation les plus utilisés sont les suivants :

b.1 Mutation par inversion

Deux positions sont sélectionnées au hasard et tous les gènes situés entre ces positions sont inversés. La Figure 3.12 montre un exemple de mutation par inversion. L'individu i avant

mutation est représenté dans la partie (a) avec le segment formé par les deux positions sélectionnées. L'inversion est effectuée à l'étape (b) afin de produire l'individu i' .

i	1	2	3	4	5	6	7	8	9	(a)
i'	1	2	3	7	6	5	4	8	9	(b)

Fig. 3.12 Mutation par inversion

b.2 Mutation par insertion

Deux positions sont sélectionnées au hasard et le gène appartenant à l'une est inséré à l'autre position. Dans la Figure 3.13 partie (a), les gènes "3" et "6" de l'individu i sont sélectionnés. La deuxième étape, illustrée par la partie (b), montre l'insertion de "6" avant "3" et le décalage de tous les autres gènes.

i	1	2	3	4	5	6	7	8	9	(a)
i'	1	2	6	3	4	5	7	8	9	(b)

Fig. 3.13 Mutation par insertion

b.3 Mutation par déplacement

Une séquence est sélectionnée au hasard et déplacée vers une position elle-même tirée au hasard. Un exemple de mutation par déplacement est illustré à la Figure 3.14. Dans la partie (a), la séquence "3-4-5-6" est sélectionnée et déplacée à la première position pour former l'individu i' représenté dans la partie (b).

i	1	2	3	4	5	6	7	8	9	(a)
i'	3	4	5	6	1	2	7	8	9	(b)

Fig. 3.14 Mutation par déplacement

b.4 Mutation par permutation

Deux positions sont sélectionnées au hasard et les gènes situés dans ces positions sont permutés. Comme illustré à la Figure 3.15, les éléments "3" et "6" sont sélectionnés dans i (partie (a)) et permutés dans i' (partie (b)).

i	1	2	3	4	5	6	7	8	9	(a)
i'	1	2	6	4	5	3	7	8	9	(b)

Fig. 3.15 Mutation par permutation

3.9 Valeurs des paramètres

Les paramètres qui conditionnent la convergence d'un algorithme génétique sont :

- la taille de la population d'individus ;
- le nombre maximal de génération ;
- la probabilité de croisement ;
- la probabilité de mutation.

Les valeurs de tels paramètres dépendent fortement de la problématique étudiée. Ainsi il n'existe pas de paramètres qui soient adaptés à la résolution de tous les problèmes qui peuvent être posés à un algorithme génétique. Cependant certaines valeurs sont souvent utilisées (définies dans la littérature) et peuvent être de bons points de départ pour démarrer une recherche de solutions à l'aide d'un AG.

- La probabilité de croisement est choisie dans l'intervalle $[0.7, 0.99]$;
- La probabilité de mutation est choisie dans l'intervalle $[0.05, 0.1]$.

3.10 Conclusion

Ce chapitre a établi les fondations nécessaires à la compréhension des algorithmes génétiques, de leurs mécanismes et de leur puissance. Ces algorithmes classés parmi les méthodes stochastiques, s'inspirent de l'évolution génétique des espèces, plus précisément du principe de la sélection naturelle.

CHAPITRE 4

Conception et architecture de l'application

4.1 Introduction

Nous nous intéressons au problème de transport de patients en hémodialyse. L'objectif est de minimiser la durée totale des tournées en respectant les contraintes du problème et en favorisant la qualité de services. L'approche de résolution fait appel aux algorithmes génétiques, développé dans le chapitre précédent.

Nous proposons une heuristique d'insertion utilisée pour générer la population, puis on a proposé une nouvelle technique de croisement pour éviter les problèmes de violation des contraintes, cette technique nous a conduites à proposer une heuristique d'insertion pour la reconstruction des individus.

4.2 Description du problème

Le problème qu'on a traité dans ce travail est le transport des patients hémodialisés, ce transport se fait en général à l'aide d'un opérateur de transport sanitaire qui doit transporter le patient de son lieu d'habitation vers le centre de dialyse, et après la séance de dialyse, il doit le retourner du centre de dialyse vers son lieu d'habitation, sous quelques contraintes :

- Le patient peut faire de une à trois séances de dialyse au maximum par semaine, ces séances doivent être faites dans des jours non consécutifs.
- L'opérateur ne peut pas transporter plus de deux patients par véhicule.

Le problème est défini sur un graphe orienté $G = (X, A, W)$ où :

$X = \{0\} \cup N^+ \cup N^-$ tel que 0 représente la base du véhicule, N^+ est l'ensemble des nœuds de départ et N^- l'ensemble des nœuds d'arrivée. A est l'ensemble de tous les arcs, et W est une application de A dans R qui affecte à chaque arc (i, j) un poids $w_{i,j}$ égale à la distance entre i et j .

Chaque nœud i a un temps de service s_i (pour embarquer les passagers et pour les déposer). Le temps de trajet t_{ij} entre deux nœuds i et j est calculé à partir de la distance et la vitesse du véhicule. Ce dernier est caractérisé par une capacité limitée, une vitesse moyenne, une base à laquelle il est rattaché, et une durée maximale T au bout de laquelle il doit retourner à la base.

Chaque demande de transport k concerne un patient qui doit être transporté d'un site de départ (pick-up) p_k à un site d'arrivée (delivery) d_k .

4.2.1 Les contraintes à respecter et l'objectif

Une solution réalisable doit satisfaire les contraintes suivantes :

- **Pairing** : les sites de départ p_k et d'arrivée d_k d'une demande k doivent être visités dans une même tournée.
- **Précédence** : le site de départ p_k de chaque demande de transport k doit être visité avant son site de d'arrivée d_k .
- **Fenêtre de temps** : c'est un intervalle de temps $[e_i, l_i]$ dont le quel le service doit commencer au plutôt à la date e_i , et au plus tard à la date l_i pour chaque site i , et si le véhicule arrive avant e_i , il doit attendre jusqu'à e_i pour commencer le service.
- **Capacité du véhicule** : à un instant donnée, le nombre de patients dans un véhicule ne doit pas dépassé un nombre maximum qui représente la capacité de ce véhicule.
- **Durée maximale des tournées** : la durée d'une tournée ne doit pas dépasser T , le véhicule devant alors retourner à la base pour se ravitailler en carburant.
- **Temps de transport** : c'est une contrainte liée à la qualité de service, elle consiste au temps passé par le patient dans le véhicule, il est en relation avec la distance entre le site de départ et le site d'arrivée.

Donc, l'objectif est de satisfaire les demandes des clients sur une journée, de façon à minimiser la durée totale des tournées en respectant les contraintes précédentes. Il s'agit donc d'un problème mono-objectif.

4.3 Méthode de résolution

La méta-heuristique que nous utilisons est un algorithme génétique. La section suivante décrit les caractéristiques principales d'AG qu'on a utilisé : le codage de chromosome, l'évaluation de

chromosome, la population initiale, la procédure de sélection, l'opérateur de croisement, l'opérateur de mutation, reconstruction de la population, et la condition d'arrêt.

➤ Le codage de Chromosome

Un chromosome est une séquence S de nœuds, sans délimiteurs de tournées (sans la base de véhicule). Ce codage peut être interprété comme l'ordre dans lequel un véhicule doit visiter tous les nœuds, si toutes les demandes sont satisfaites dans la même tournée. Dans ce chromosome, chaque demande k apparaît deux fois : une première fois en tant que $k+$ pour indiquer le nœud de départ (pick-up) et une deuxième fois en tant que $k-$ pour le nœud d'arrivée (delivery). La figure 4.1 représente un exemple de chromosome satisfaisant quatre demandes.

1+	1-	2+	2-	3+	4+	4-	3-
----	----	----	----	----	----	----	----

Fig.4.1 Codage de chromosome

Afin de découper un chromosome en tournées (solution de problème de transport), nous utilisons une procédure *Split* expliquée dans ce qui suit.

• La procédure Split

La procédure *Split* permet de découper un chromosome (voire la figure 4.2) en tournées réalisables, de façon à minimiser la durée totale des tournées.

Soit S une séquence de $2n$ nœuds, où n est le nombre de demandes de transport. L'objectif de *Split* est de trouver un plus court chemin dans un graphe auxiliaire $H = (X, A, Z)$ où, X contient $2n + 1$ nœuds (le nœud 0 correspondant à la base de véhicule, et 2 nœuds associés à chaque demande) et A contient un arc (i, j) si la tournée visitant les nœuds S_{i+1} à S_j est réalisable. Une telle tournée est réalisable si les contraintes de pairing et la durée maximale d'une tournée sont respectées. Enfin le poids $z(i, j)$ sur chaque arc est la durée de la tournée.

Une solution optimale pour un chromosome associé à une séquence S de nœuds correspond à un chemin de coût minimal μ de 0 à $2n$ dans ce graphe H . μ peut être calculé en $O(n^2)$ en utilisant l'algorithme de Bellman [Cormen et al. 1990].

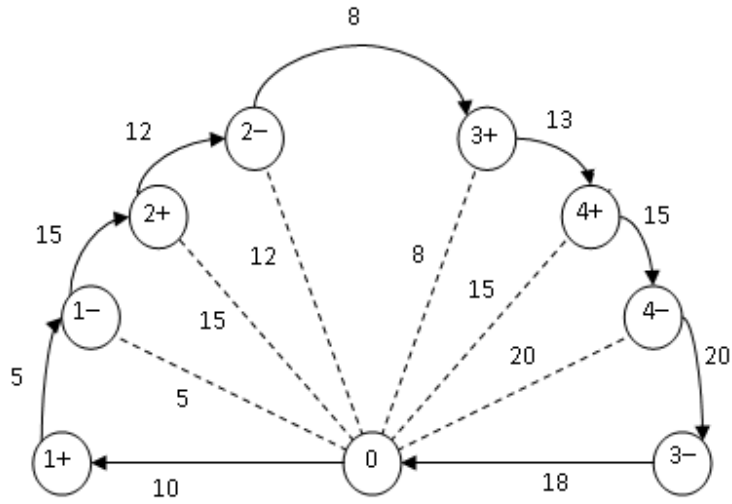


Fig.4.2 Exemple d'un chromosome qui sera divisé par *Split*

La procédure *Split* est représentée dans l'algorithme 4.1 suivant :

Algorithme 4.1 Procédure *Split*

```

1 :  $V[0] = 0$ ;
2 : Pour  $k := 1$  à  $2n$  faire
3 :    $V[i_k] = +\infty$  ;
4 : Fin pour
5 : Pour  $k = 1$  to  $2n$  faire
6 :    $D = 0$ ; // La durée de la route.
7 :    $V = \emptyset$  ; //Ensemble des requêtes dont le nœud de ramassage a été visité. Mais le nœud
8 :     de déchargement n'a pas été visité.
9 :    $l = k$  ;
10:  Répété
11:    // La mise à jour de  $V$ 
12:    Si  $i_l \in N^+$  Alors
13:       $V = V \cup \{r(i_l)\}$ ; //  $r(i_l)$  : Demande associée au nœud  $i_l$ .
1 2 3

```

```

14: 1 2 3
15:   Sinon
16:   |
17:   |  $V = V \setminus \{r(i_l)\};$ 
18:   |
19:   | Fin si
20:   |
21:   | // Mettre à jour la durée de la route
22:   |
23:   | Si  $l = k$  Alors
24:   | |
25:   | |  $D = D + t_{0,i_l} + t_{i_l} + t_{i_l,0} + s_{i_l};$ 
26:   | |
27:   | | Sinon
28:   | | |
29:   | | |  $D = D + t_{i_l-1,i_l} + t_{i_l,0} - t_{i_l-1,0} + s_{i_l};$ 
30:   | | |
31:   | | | Fin si
32:   | |
33:   | | // Si tous les contraintes sont vérifiées
34:   | |
35:   | | Si  $(D \leq T)$  et  $(V = \emptyset)$  Alors
36:   | | |
37:   | | | Si  $V[i_k - 1] + D < V[k_l]$  Alors
38:   | | | |
39:   | | | |  $V[i_l] = V[i_k - 1] + D;$ 
40:   | | | |
41:   | | | |  $pred[i_l] = k - 1;$  //  $pred[i_l]$  : Prédécesseur de  $i_l$  dans le plus court chemin.
42:   | | | |
43:   | | | | Fin si
44:   | | |
45:   | | | Fin si
46:   | |
47:   | |  $l = l + 1;$ 
48:   |
49:   | Jusqu'à  $(j > 2n)$  ou  $(D > T)$ 
50:   |
51:   | Fin pour

```

L'algorithme *Split* permet de calculer pour chaque nœud $i_k \in X$ une étiquette $V[i_k]$ qui représente la durée du plus court chemin, de 0 vers i_k . La boucle principale (la boucle « Répété ») énumère tous les arcs possible ou sous-séquences i_k, \dots, i_{k+nr} et mettre à jour la valeur $V[i_k]$. L'algorithme ne génère pas le graphe H (graphe auxiliaire) explicitement.

Exemple

Soit le chromosome de la figure 4.2. Nous supposons que le temps de service pour chaque nœud n_i est 5 min et que la durée maximale d'une route est 100 min.

Dans cet exemple la procédure *Split* génère implicitement le graphe auxiliaire H (voire la figure 4.3), qui contient, par exemple, l'arc $(0,2-)$ qui représente la route $(0,1+,1-,2+,2-,0)$ avec une durée de 74 min. Le chemin de coût minimum μ contient deux arcs gras et son coût est 168.

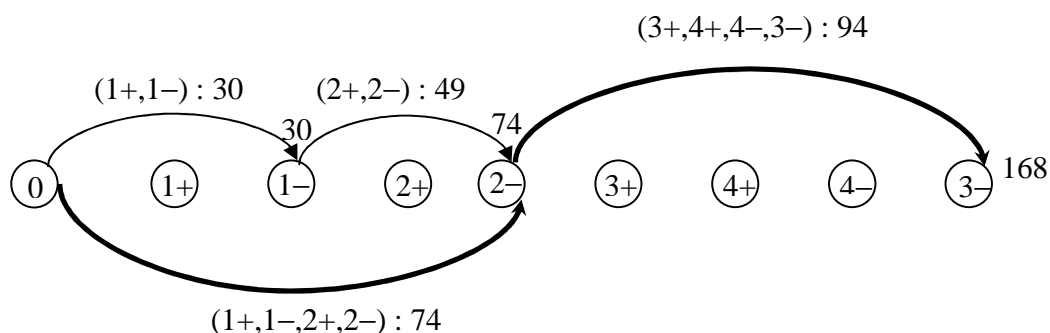


Fig.4.3 Le graphe auxiliaire H

La figure 4.4 représente le résultat de la procédure *Split*. Elle est composée de deux routes. La route 1 est $(0,1+,1-,2+,2-,0)$ avec une durée de 74 min, et la route 2 est $(0,4+,3+,3-,4-,0)$ avec une durée de 94 min.

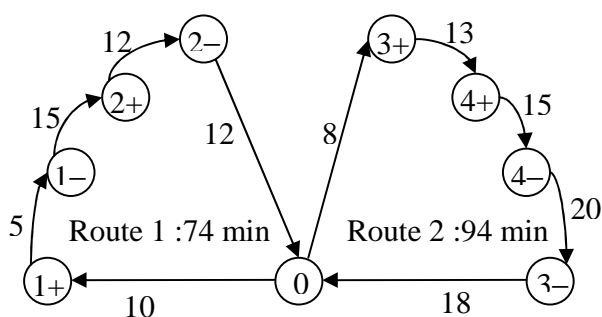


Fig.4.4 Résultat de la procédure *Split*

➤ La population initiale

La population initiale est générée par des méthodes d'insertion aléatoires, en cherchant à favoriser la diversité de la population pour mieux explorer l'espace des solutions.

Ces approches posent deux problèmes essentiels ; le premier concerne la génération des individus invalides, vu que les solutions doivent satisfaire les contraintes du problème, à savoir la précedence et la satisfaction de toutes les demandes, leurs fenêtres de temps, et le temps de

transport. Le deuxième concerne le cas où les solutions générées sont valides mais irréalisables à cause de la limitation du temps total d'une route.

Ces problèmes peuvent mettre l'exécution du programme dans un état de stagnation en sorte qu'il est possible de ne pas générer une population complète par des individus valides et réalisables, ou de la générer dans un temps irraisonnable.

Afin de faire en sorte que la génération des individus de la population initiale soit guidée, nous avons élaboré une heuristique qui respecte les caractéristiques temporelles du problème en vue de générer la population initiale dans un temps raisonnable. Le principe de cette heuristique est de générer aléatoirement les nœuds de chaque individu et en même temps vérifier leur validation et si le nœud généré est un nœud de chargement il faut réserver le temps nécessaire pour délivrer cette demande dans le temps total de la route et en même temps déclencher un compteur qui calcule le temps de transport de cette demande pour ne pas dépasser un temps maximal qui égale au double du temps nécessaire pour passer directement du site de départ au site d'arrivée de cette demande.

Si à un moment donné l'ajout d'un nœud de chargement d'une demande viole le temps total de la route, ce nœud sera éliminé et l'ensemble des demandes qui sont collectées et pas encore délivrées sera délivrées immédiatement, puis on continue jusqu'à la fin de la génération d'un individu.

En suite on va appliquer l'algorithme de *Split* décrit ci-dessus sur cet individu pour créer les routes de façon optimale. Si le résultat obtenu existe déjà dans la population initiale, il faut régénérer cet individu. Cette procédure est répétée jusqu'à l'obtention du nombre désiré d'individus à placer dans la population initiale. A la fin de l'exécution de cette heuristique, on obtient une population initiale de taille fixée par l'utilisateur qui contient des individus valides et réalisables et non répétés. L'algorithme 4.2 illustre le principe de cette heuristique.

Algorithme 4.2 : L'heuristique d'insertion aléatoire

```

1: Début
2:   Pour (i=1 à la taille de la population) Faire // chargement de la population initiale
3:     Pour (j=1 à la taille de l'individu) Faire // génération aléatoire d'un individu
4:       Tant que (les contraintes ne sont pas vérifiées) Faire
5:         1 2 3 4 Générer un site aléatoirement ;

```

	1	2	3	4
6:				Tester la contrainte de la précédente ;
7 :				Tester la répétition ;
8:				Si (les contraintes sont vérifiées) alors
9:				Si (la fenêtre de temps est satisfait) alors
10 :				Ajouter le site à l'individu ;
11 :				Fin si
12:				Vérifier les fenêtres de temps et le temps de transport pour toutes les sites générés;
13:				Vérifier la longueur totale de la route ;
14 :				Fin si
15:				Fin tant que
16:				Fin pour
17:				Appliquer la procédure <i>Split</i> sur l'individu généré
18:				Si (l'individu généré n'existe pas dans la population) alors
19:				Ajouter l'individu à la population initiale ;
20 :				Sinon
21 :				Régénérer l'individu ;
22:				Fin si
23:				Fin pour
24:				Fin.

➤ La sélection

La méthode de sélection élaborée est une combinaison de deux techniques de sélection, la première consiste simplement à sélectionner les meilleurs individus dans la population, la deuxième technique est la sélection par tournoi exposé au chapitre précédant. Notre méthode permet de sélectionner 1/4 de la population par la première technique et 1/4 par la deuxième technique.

Cette combinaison assure que les meilleurs individus de la population courante seront apparus dans la prochaine population et donne aux autres individus une possibilité d'être aussi dans la population. Le pseudo code de la sélection est donné dans l'algorithme 4.3.

Algorithme 4.3 : Le principe de la sélection

```
1 : Début
2 :   Sélectionner les 1/4 meilleurs individus de la population ;
3 :   // Sélectionner un quart des individus parmi le reste des individus par la technique
4 :   du tournoi
5 :   Tant que (le nombre des individus sélectionnés < taille de population /4) Faire
6 :     Choisir deux individus de la population aléatoirement ;
7 :     Générer une probabilité ;
8 :     Si (la probabilité <= paramètre de sélection) alors
9 :       Choisir le mauvais des deux individus pour le croisement ;
10 :    Sinon
11 :      Choisir le meilleur des deux individus pour le croisement
12 :    Fin si
13 :  Fin tant que
14 : Fin.
```

➤ **L'opérateur de croisement**

• **Le principe**

Pour le croisement, nous utilisons un croisement circulaire à deux points de coupure (OX). Après avoir sélectionné deux parents P1 et P2, cet opérateur de croisement consiste à sélectionner aléatoirement deux points de coupure i, j , à condition que le rang de i est inférieur au rang de j . La sous séquence P1 (i)::P1 (j) du parent P1 est tout d'abord copiée dans le fils C1. Puis P2 est parcouru de façon circulaire à partir de $j+1$ pour compléter C1. Un deuxième fils C2 est construit de façon similaire en échangeant les rôles de P1 et P2 et les deux points de coupure en gardant la même distance entre eux.

Le problème posé dans ce cas est le risque de violer les contraintes de précédence et du temps total de chaque route ainsi que le temps de transport d'une demande (la distance entre le site $+i$ et le site $-i$ de la demande i), et donc les fils seront non valides. Ce problème est compliqué surtout

dans le cas de grand nombre de demandes. Pour éviter ce problème, on a considéré seulement les nœuds de chargement des demandes dans les parents qui seront croisés.

Exemple : la figure 4.5 illustre l'exécution de l'opérateur de croisement sur deux individus.

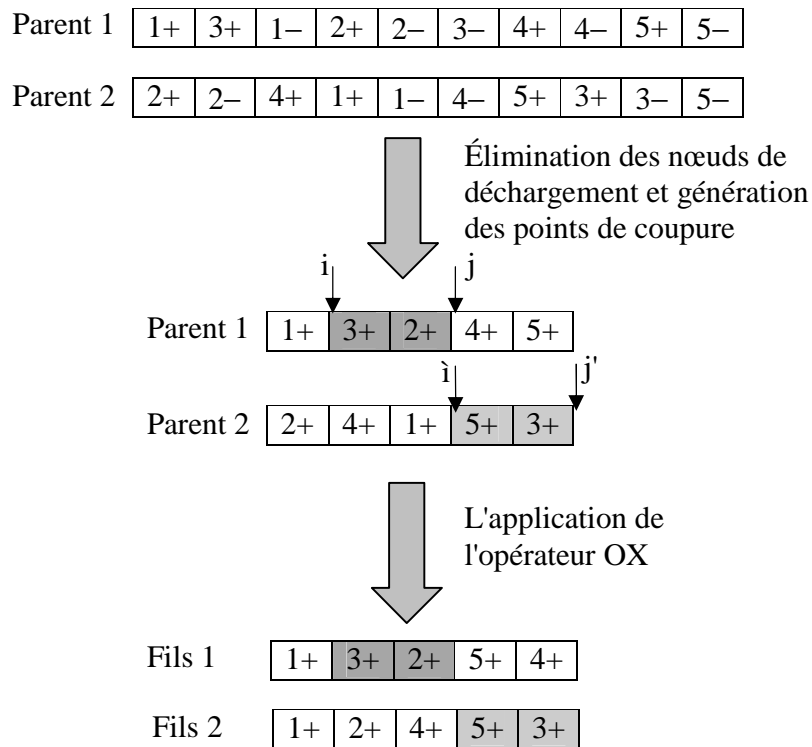


Fig.4.5 L'application de l'opérateur de croisement sur un exemple

L'algorithme 4.4 illustre le principe de croisement

Algorithme 4.4 : Le principe de croisement

- 1 : **Début**
 - 2 : Générer aléatoirement deux points de coupure pour l'individu 1
 - 3 : Calculer la distance entre ces deux points
 - 4 : Générer le premier point de coupure pour l'individu 2
 - 5 : Calculer le deuxième point par l'utilisation de la distance qu'on a calculée
 - 6 : Faire le croisement OX
 - 7 : **Fin.**
-

Après le croisement on applique sur les fils une heuristique qui permet d'insérer les nœuds de déchargement dans les bonnes positions de façon à minimiser le temps total de satisfaction des demandes sans violation des contraintes du problème.

- **L'heuristique d'insertion des nœuds de déchargement**

Cette procédure prend en entrée un individu qui contient uniquement les nœuds de chargement des demandes et elle donne en sortie un individu complet ; contient tous les nœuds de chargement avec leurs nœuds de déchargement correspondant.

Son principe est de parcourir l'individu à partir de sa fin jusqu'au début et à chaque nœud de chargement elle insert son nœud de déchargement dans la bonne position de façon à respecter les contraintes et à minimiser le temps total de satisfaction des demandes.

Exemple : la figure 4.6 illustre l'exécution de cette heuristique sur un individu qui contient les nœuds de chargement de trois demandes.

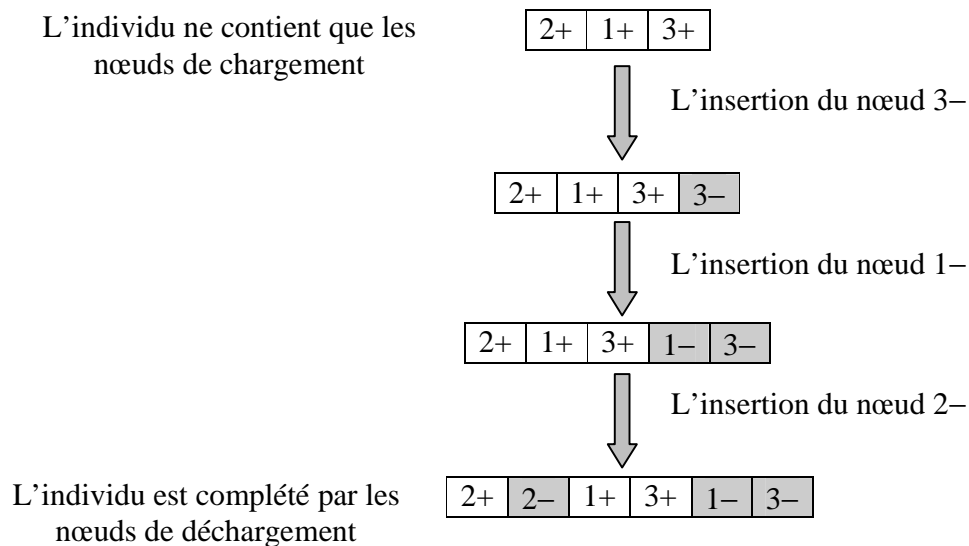


Fig.4.6 L'application de l'heuristique d'insertion sur un exemple

L'algorithme 4.5 illustre le principe de cette heuristique :

Algorithme 4.5 : L'heuristique d'insertion des nœuds de déchargement

```

1 : Début
2 :   Tant que (il y a un nœud de déchargement pas encore inséré) Faire
3 :     Insérer le nœud de déchargement à la fin de l'individu
4 :     Tant que (le nœud de chargement précède le nœud de déchargement) Faire
5 :       Si (les contraintes du problème sont vérifiées) alors
6 :         Si (le coût total de l'individu est meilleur que la valeur sauvegardée) alors
7 :           Remplacer l'ancienne valeur du coût par la nouvelle
8 :           Sauvegarder la position du nœud de déchargement
9 :         Fin si
10 :      Fin si
11 :      Décrémenter la position du nœud de déchargement
12 :    Fin tant que
13 :    Insérer le nœud de déchargement à la meilleure position dans l'individu
14 :  Fin tant que
16 : Fin.

```

Enfin on applique la procédure de Split pour rendre l'individu réalisable.

➤ **L'opérateur de mutation**

Après l'étape de croisement, l'opérateur de mutation est appliqué sur les individus de la nouvelle population avec une probabilité p_m choisie généralement dans l'intervalle [0.05, 0.1] (c'est-à-dire on peut appliquer la mutation sur l'individu i avec une probabilité $p_m(i)$).

Notre opérateur de mutation est de type « mutation par permutation » exposé au chapitre précédent qui sera appliqué sur un individu qui contient seulement les nœuds de chargement (voire figure 4.7) (on élimine les nœuds de déchargement).

1+	2+	3+	4+
----	----	----	----

Fig.4.7 Un individu avec les nœuds de chargement

Le résultat de mutation est réinséré dans la population après une étape de réinsertion des nœuds de déchargement pour obtenir une solution réalisable qui sera utilisé dans la génération suivante. Ce dernier processus est fait par un appel à l'heuristique d'insertion.

➤ **La reconstruction de la population**

Cet opérateur est simple. Dans notre algorithme on remplace la totalité de la population P (ancienne population) par la population P' (nouvelle population) qui a été obtenue par application successive des opérateurs de sélection, de croisement et de mutation.

➤ **La condition d'arrêt**

Dans notre connaissance, il n'existe pas de conditions d'arrêt universellement acceptées pour les algorithmes génétiques mono-objectif. Nous arrêtons simplement notre algorithme au bout d'un nombre fixe d'itération.

➤ **L'algorithme génétique pour le problème de transport de patient**

L'algorithme 4.7 illustre notre algorithme génétique pour résoudre le problème de transport de patient qui contient toutes les techniques décrites ci-dessus :

Algorithme 4.6 : L'algorithme génétique pour le problème de transport de patient

```

1 : Début
2 :   Génération de la population initiale ;
3 :   Tant que (le nombre des générations inférieur au nombre des itérations choisies) Faire
4 :     Sélectionner un demi des éléments de l'ancienne population par la méthode de sélection
5 :     Ajouter ces éléments à la nouvelle population ;
6 :     Tant que (il reste des individus sélectionnés qui n'ont pas participé dans le croisement) Faire
7 :       Choisir deux individus aléatoirement ;
8 :       Croiser ces deux individus ;
9 :       Ajouter les deux individus fils à la population ;
10 :   Fin tant que
1 2

```

```

11 : 1 2 Si (la population n'est pas complète) alors
12 :     Compléter la population à partir du reste des éléments de l'ancienne population qui ne sont
13 :     pas sélectionnés ;
14 :     Fin si
15 :     Pour (chaque élément de la population) Faire
16 :         Appliquer la mutation selon une probabilité générée ;
17 :     Fin pour
18 :     Remplacer l'ancienne population par la nouvelle ;
19 : Fin tant que
20 : Fin.

```

4.4 Architecture de l'application

La section suivante présente une description détaillée de l'architecture de l'application. Dans notre application nous avons développé (07) classes, dont les (06) premières classes sont groupées dans un package qui a comme nom « info » (voire figure la 4.8). La dernière est la classe principale nommée « IHM_For_Projet ».

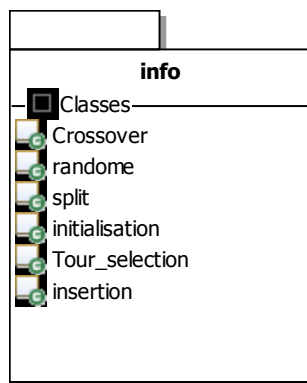


Fig. 4.8 le package « info »

➤ Diagramme de classes de l'application

Afin de voir les relations entre les classes de l'application, nous allons les représenter dans un diagramme de classe présenté dans la figure la 4.9 suivante :

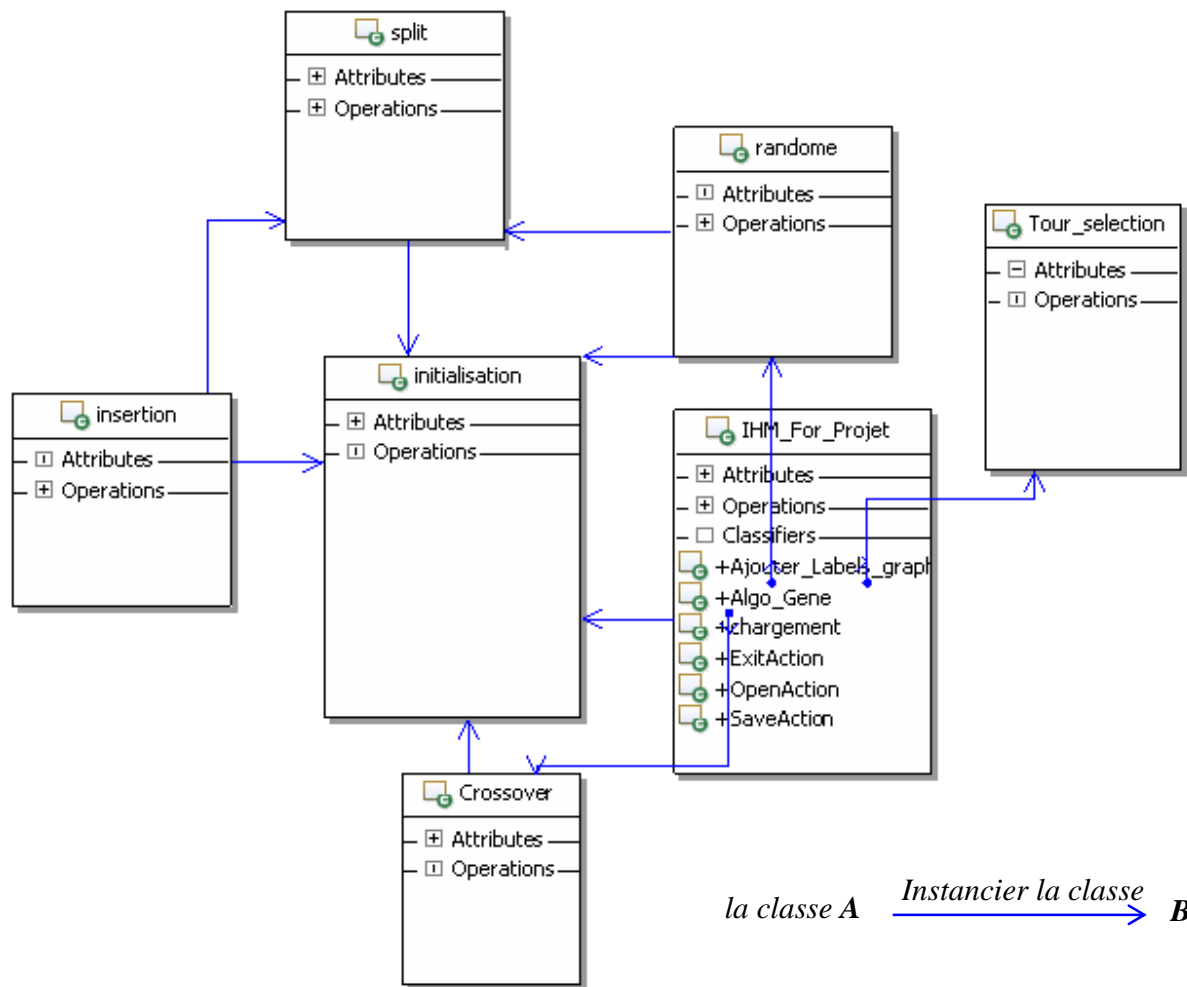


Fig. 4.9 Diagramme de classes de l'application

Dans la figure 4.10, les flèches bleues représentent des associations entre les classes signifiant qu'une classe instancie une autre classe. Par exemple la classe « Algo_Gene », qui appartient à la classe « IHM_For_Projet », instancie la classe « initialisation » pour initialiser les attributs de cette dernière.

Dans ce qui suit nous allons présenter chaque classe à part afin de bien expliquer le rôle de chacune d'elle ainsi que leurs méthodes.

➤ La classe « IHM_For_Projet »

La classe « IHM_For_Projet » est la classe principale de projet. Elle permet de construire l'interface graphique. Elle est composée par un nombre des méthodes et classes (voire la figure 4.10):

- **Les méthodes**

Parmi les méthodes de cette classe on peut citer :

+hashDefaultActions : cette méthode permet de construire une table d'hachage de deux colonnes, dont la première contient les noms d'action de composant `JTextArea` (exp : paste, copy...), et la deuxième contient l'adresse mémoire de chaque action.

+getHashedAction : pour obtenir l'adresse d'action à partir de la table d'hachage.

+makeActionsPretty : permet d'associer pour chaque action une icône et un nom qui seront par la suite affichés à l'utilisateur.

+updateKeymap : permet la mise à jour de Keymap de composant `JTextArea`.

Les (04) méthodes précédentes permettent de créer un mini éditeur de texte avec les fonctions de base : copier, coller, couper, sélectionner.

+dessiner_graphe_Données : cette méthode permet de dessiner un graphe à partir d'un document et le mettre dans un panel pour afficher par la suite.

+dessiner_graphe_résultat : elle permet de dessiner le graphe de résultat et le mettre dans un panel.

+creation_arcs_Données et +creation_arcs_résultat : permettent de créer les arcs d'un graphe.

+creation_noeuds_Données et +creation_noeuds_résultat : permettent de créer les nœuds d'un graphe.

- **Les classes**

+chargement : elle permet de charger la matrice de distance, la matrice de requête et les variables `nbr_site`, `nbr_req` et `base` à partir d'un fichier texte.

+algo_gene : elle contient le programme principal de l'algorithme génétique.

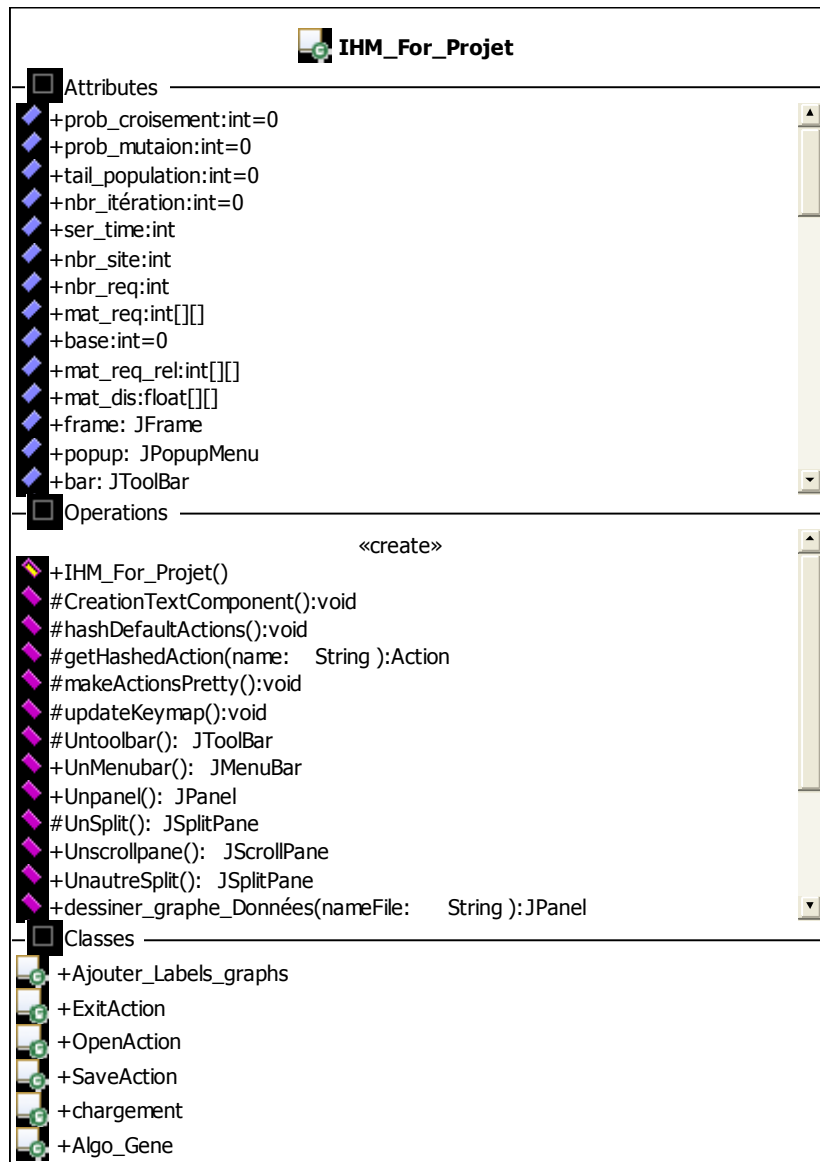


Fig. 4.10 la classe « IHM_For_Projet »

➤ La classe « randome »

Cette classe permet de construire une population initiale contient un nombre d'éléments réalisables défini par l'utilisateur pour l'algorithme génétique (voir la figure 4.11), ces méthodes sont :

+randome : constructeur.

+decr: assure la non violation des contraintes du problème.

+repeat: tester la répétition de l'individu dans la population.

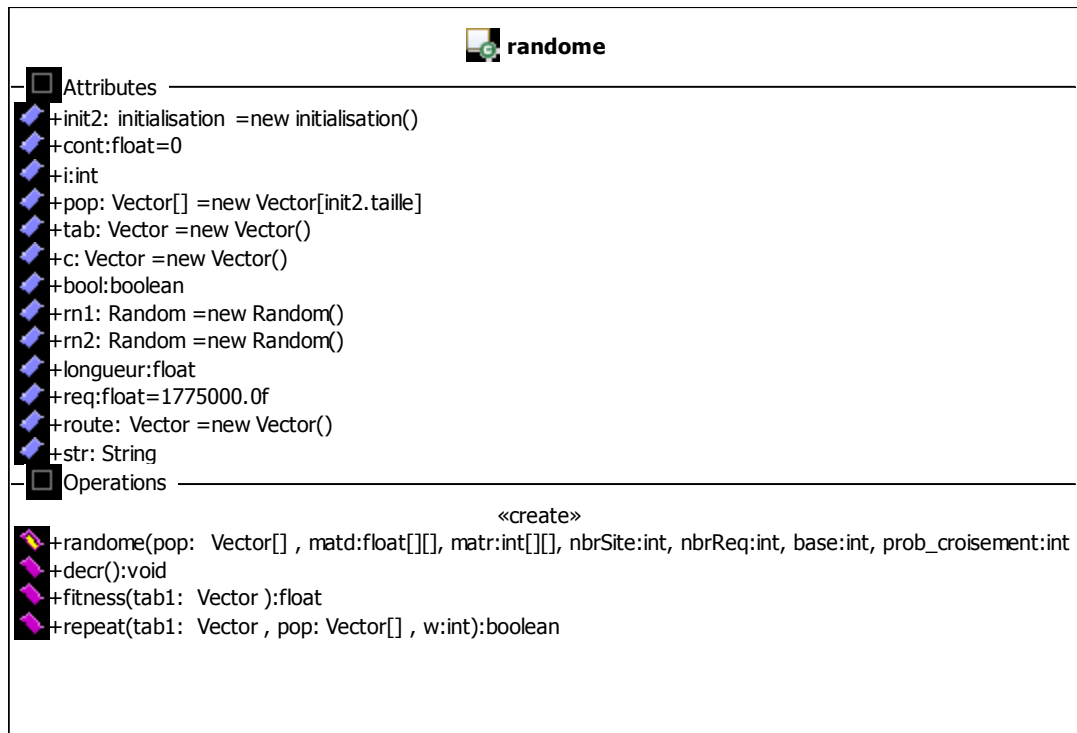


Fig. 4.11 la classe « randome »

➤ La classe « split »

La classe Split permet de décomposer un individu viable en routes réalisables de façon à optimiser le temps total de réalisation des demandes (voire la figure 4.12). Elle contient les méthodes suivantes :

+split: le traitement principal de la classe Split.

+fitness: calculer le coût total de l'individu.

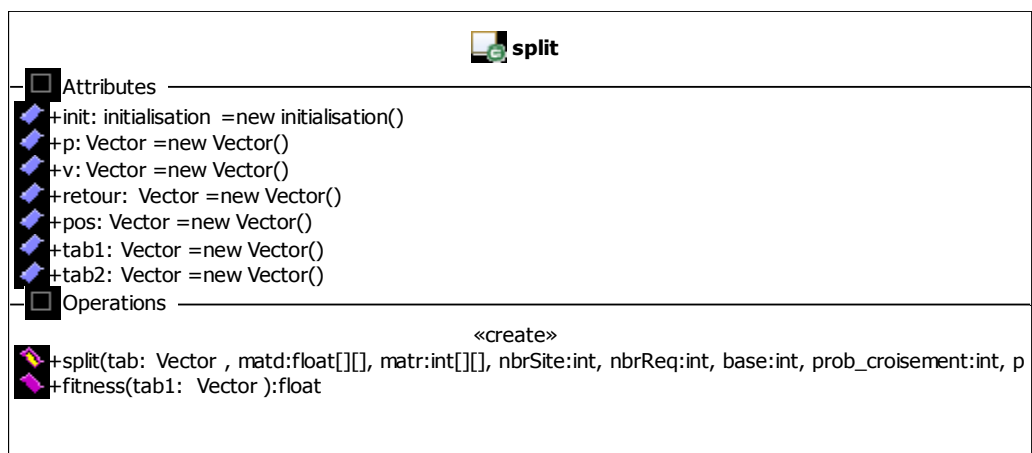


Fig.4.12 la classe « split »

➤ La classe « insertion »

C'est la classe qui permet d'insérer les sites de déchargement aux individus qui ne contiennent que les sites de chargement après le processus de croisement et de mutation (voire la figure 4.13), elle contient les méthodes suivantes:

+insert: c'est la méthode qui applique le processus d'insertion.

+Controle: vérifie la contrainte du temps maximal des tournées.

+Controle_fenetre: c'est la méthode qui vérifie les fenêtres du temps des demandes.

+tps_trans_control: c'est la méthode qui vérifie les temps de transport des demandes.

+fitness: calcule le coût de l'individu.

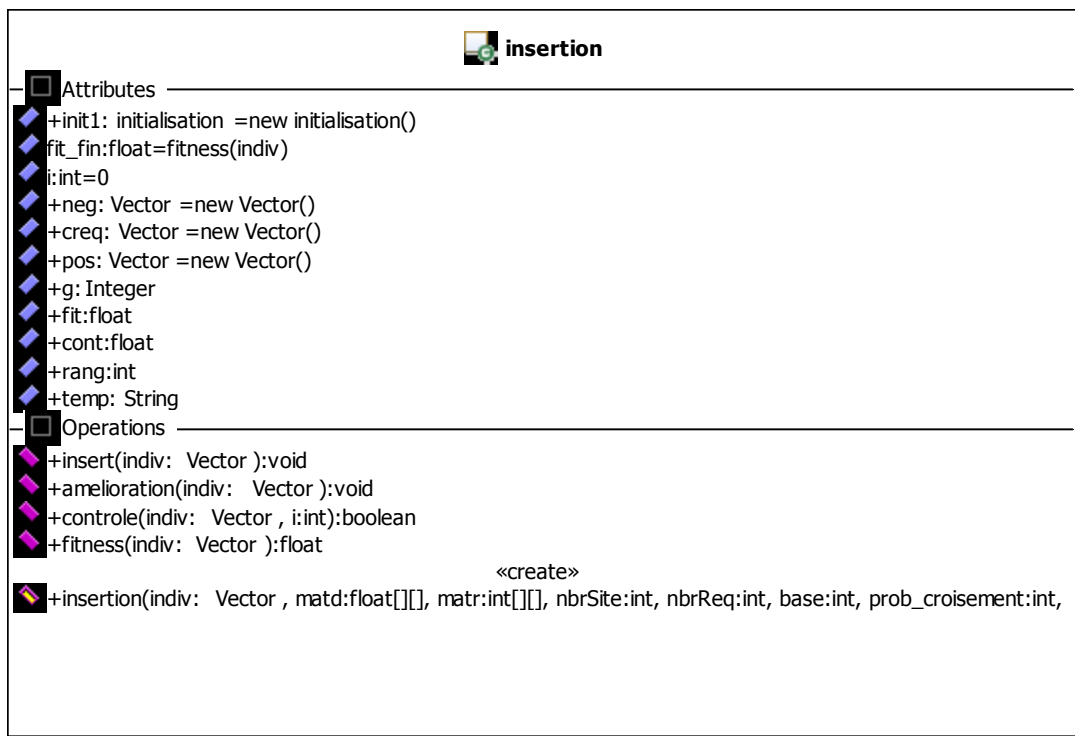


Fig. 4.13 la classe « insertion »

➤ La classe « initialisation »

C'est la classe qui contient les paramètres de l'algorithme génétique, et qui permet de charger les données, sur les sites et les requêtes, nécessaire pour démarrer l'algorithme génétique (voir la figure 4.14), elle contient les méthodes suivantes :

+chrg_table-ref : c'est la méthode qui charge le tableau de référence entre les sites réels du problème et les sites vertueux du programme.

+chrg_table_dis: c'est la méthode qui charge la matrice des distances entres les sites.

+chrg_table_dis_base: c'est la méthode qui charge la matrice des distances entres les sites et la base.

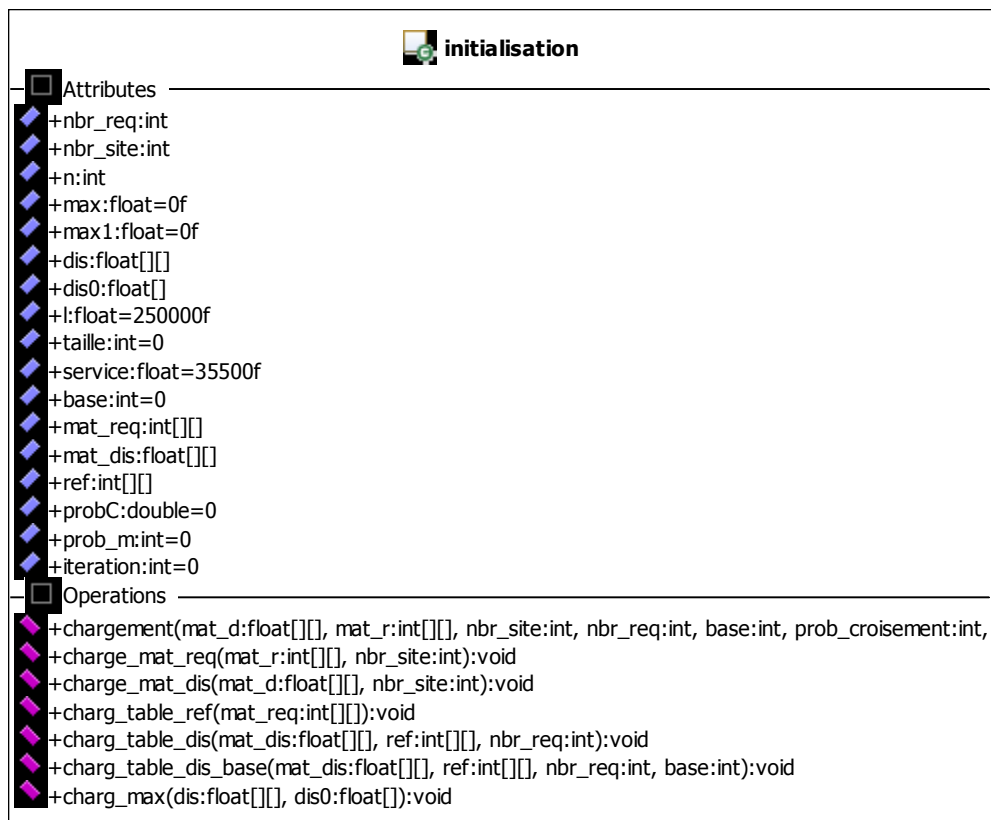


Fig. 4.14 la classe « initialisation »

➤ La classe « Crossover »

C'est la classe qui fait le croisement entre deux individus ou la mutation d'un individu (voire la figure 4.15), elle contient les méthodes suivantes :

+Selectcross : elle sélectionne deux individus aléatoirement pour le croisement, puis elle ajoute les fils à la population.

+Supp (vector individu) : elle supprime les sites de déchargement dans l'individu pour faire le croisement.

+Mutationwithpermutation: elle fait la mutation d'un individu.

+CrossOx: c'est la méthode qui fait le croisement entre deux individus.

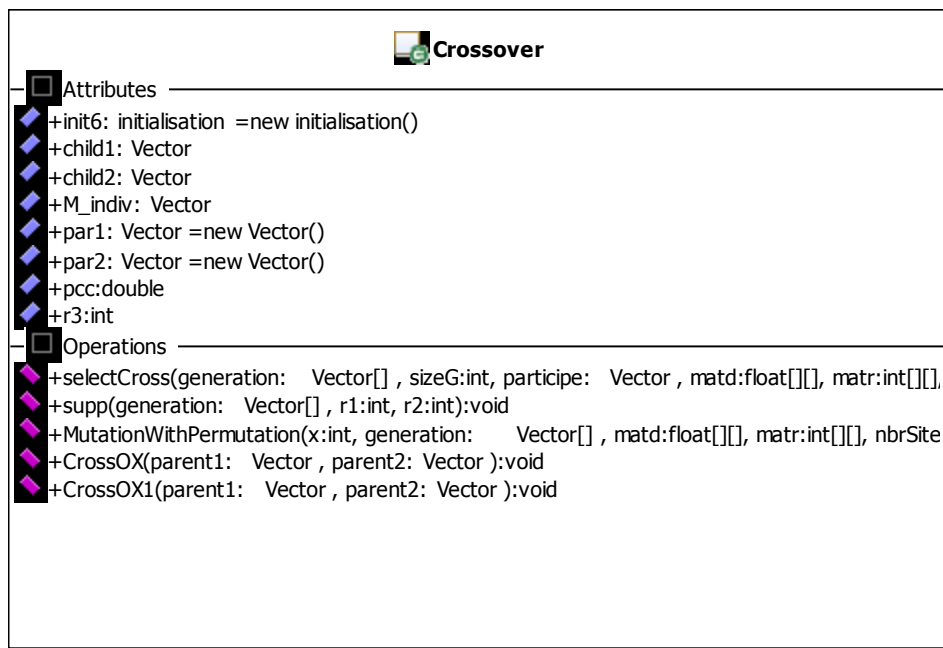


Fig. 4.15 la classe « Crossover»

➤ La classe « Tour_selection » :

C'est la classe qui fait la sélection des individus qui participent dans le croisement et qui seront ajoutés à la prochaine population (voire la figure 4.16).

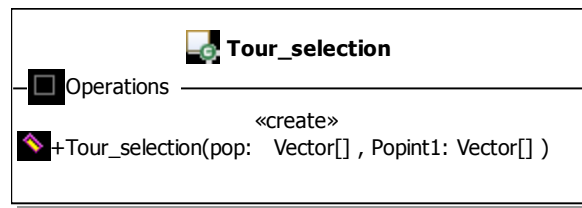


Fig.4.16 la classe « Tour_selection »

4.5 Conclusion

Dans ce chapitre nous avons présenté la conception de notre algorithme génétique. Après une description du problème, nous avons détaillé les techniques utilisées dans chaque étape de l'algorithme ainsi que les heuristiques proposées, et on a terminé par la présentation de l'architecture générale de l'application sous forme d'un diagramme de classe ainsi que les principales méthodes qui construisent ces classes.

CHAPITRE 5

Résultats et discussion

5.1 Introduction

Dans ce chapitre nous présentons tout ce qui concerne notre expérimentation, on débute par la description des jeux de données utilisés ensuite la présentation de l'interface de l'application puis les paramètres de l'algorithme génétique. Notre algorithme génétique est testé sur le jeu de données aléatoire, reflétant le transport à la demande des patients, et afin de mesurer l'efficacité de l'AG nous avons utilisé une heuristique basée essaim qui utilise deux voisinages (un voisinage basé sur l'expérience et un autre voisinage local).

5.2 L'environnement de programmation

Nous avons développé l'application en utilisant l'environnement de programmation : JCreator Pro v 4.5, sous système d'exploitation Windows 7.

5.3 Choix du langage

Le langage Java (café en argot anglais) a été créé en 1991, par la firme Sun Microsystems, qui souhaitait créer un environnement indépendant du hardware et pouvant permettre de programmer des appareils. [Labosun]

Définition de SUN : « Java est un langage simple, orienté objet, distribué, robuste, sûr, indépendant des architectures matérielles, portable, de haute performance, multithread et dynamique ».

- **Simple** : inspiré du C++, Fortran, Lisp, Smalltalk. Pas de pointeur ; pas de surcharge d'opérateur ; pas d'héritage multiple et présence d'un « garbage collecteur ».
- **Orienté objet** : la programmation objet modélise des objets ayant un état (ensemble de variables) et des méthodes (fonctions) qui leur sont propres. L'unité de base en Java est la classe. Un des intérêts de Java est de disposer de nombreuses classes déjà faites. Un objet créé à partir d'une classe est une *instance*.
- **Distribué** : les fonctions d'accès au réseau et les protocoles Internet les plus courants sont intégrées.
- **Robuste** : typage des données très strict, pas de pointeur.
- **Sûr** : Java n'est pas compilé à destination d'un processeur particulier mais en « byte code » qui pourra ensuite interpréter sur une machine virtuelle Java (JVM= Java Virtual Machine), Le « byte code » généré est vérifié par les interpréteurs Java avant

l'exécution, un débordement de tableau déclenchera automatiquement une exception et l'absence d'arithmétique de pointeur évite les malversations.

- **Portable** : les types de données sont indépendants de la plateforme (par exemple les types numérique sont définis indépendamment du type de plateforme sur laquelle de type code sera interprétée).
- **Haute performance** : discutable car Java est un langage pseudo interprété et les techniques de « Just in time » (Jit) peuvent améliorer ces performances.
- **Multithread** : une application peut être décomposée en unité d'exécution fonctionnant simultanément.
- **Dynamique** : les classes Java peuvent être modifiées sans avoir à modifier le programme qui les utilise.
- **Politique** : Java actuellement totalement contrôlé par SUN.

5.4 Implémentation et Jeux de données

Tous les tests ont été réalisés sur un ordinateur Pentium 2.0 GHz CORE 2 Duo et 2.0 Go de RAM sur Windows 7. Les développements ont été faits en java et testés sur un jeu de données qui contient des données générées aléatoirement. Le jeu de données est composé de trois séries de 10 problèmes chacune :

- la première série (P10D5) est composée de problèmes contenant 5 sites et 10 patients, et comme chaque patient ne peut faire l'opération de dialyse que trois fois au maximum par semaine, donc le nombre de requêtes dans chaque problème de cette série varie entre 10 et 30 requêtes.
- La deuxième série est (P20D5), c.-à-d. 20 patients et 5 sites, donc le nombre de requêtes est entre 20 et 60 requêtes par semaine.
- La troisième série est (P40D10), dans cette série le nombre de requêtes peut atteindre jusqu'à 120 requêtes.

Puisque chaque requête représente une demande de transport d'un patient depuis son site d'habitation vers le centre de dialyse, cela nécessite une autre requête inverse qui consiste à transporter ce patient de son centre de dialyse vers son site d'habitation après la séance de dialyse, donc le nombre de requêtes de chaque problème va être le double.

Prenant par exemple un problème de (P10-D5), ce problème doit être représenté dans un fichier texte et organisé comme suit (voire la figure 5.1)

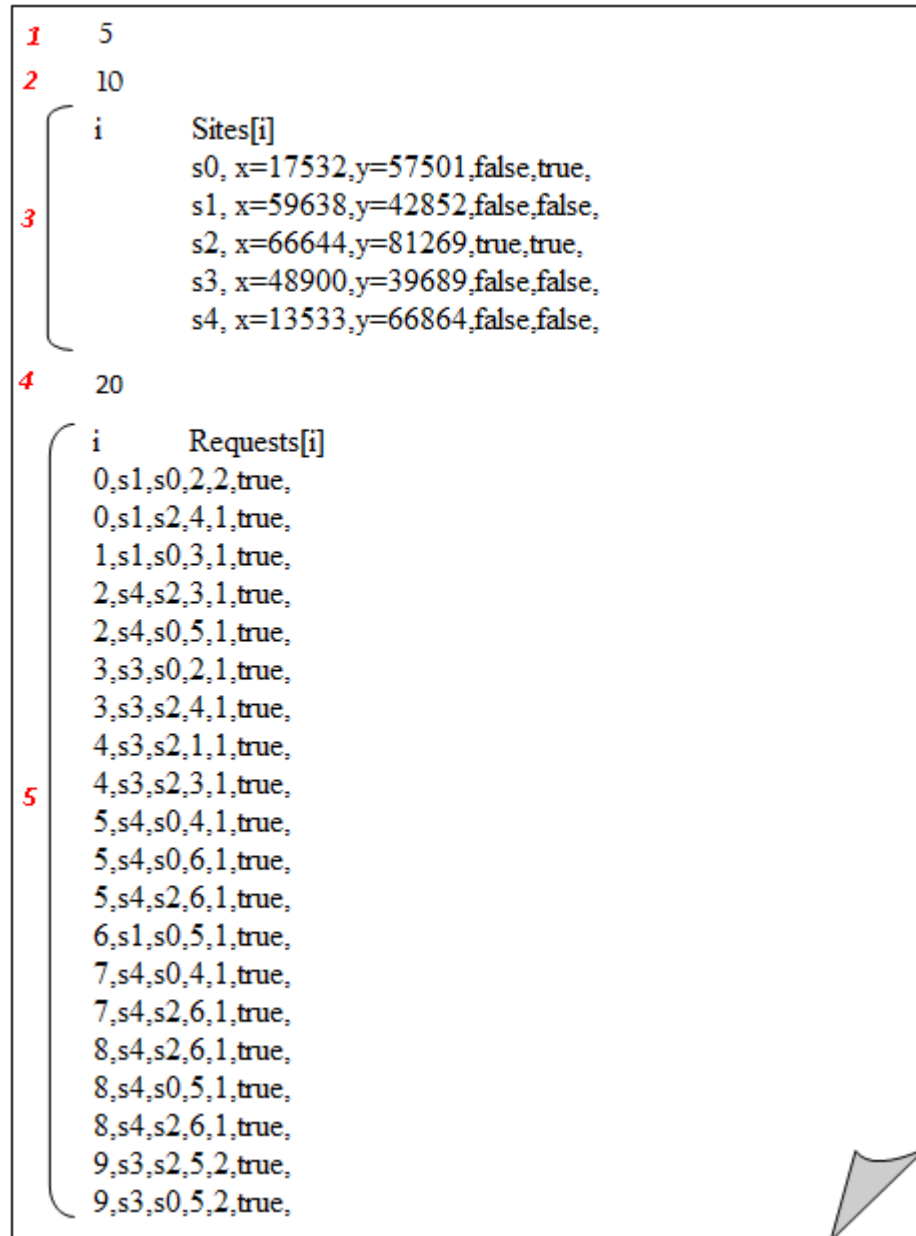


Fig. 5.1 l'organisation textuelle d'une instance d'un problème

Tel que :

1 : représente le nombre de sites.

2 : représente le nombre de patients.

3 : représente les caractéristiques de chaque site ; chaque site est caractérisé par :

numéro du site, sa position (x, y), une base (true/false), a un centre de dialyse (true / false),

Remarque : dans l'exemple de la figure 4.1, le site qui représente la base est le site s2, et les sites qui contiennent des centres de dialyse sont s0 et s2.

4 : c'est le nombre de requêtes.

5 : représente la suite des requêtes, chacune est de la forme :

Numéro du patient, site d'habitation, site de dialyse, jour de dialyse, séance de dialyse, état.

Une requête est représentée graphiquement par une flèche qui relie le site de chargement par son site de déchargement. Les requêtes de l'exemple précédent sont représentées dans le graphe orienté et connexe illustré dans la figure 5.2.

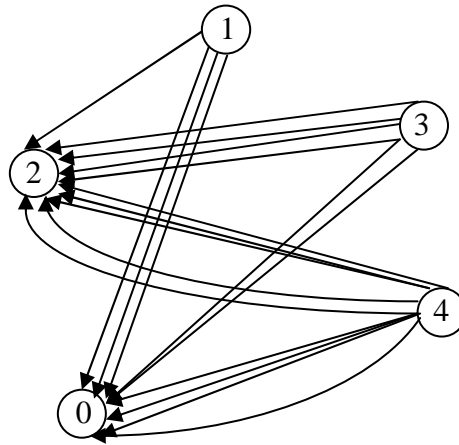


Fig. 5.2 la représentation graphique d'une instance d'un problème

5.5 Présentation de l'interface

Au lancement de l'exécution de l'application la fenêtre suivante est apparue :

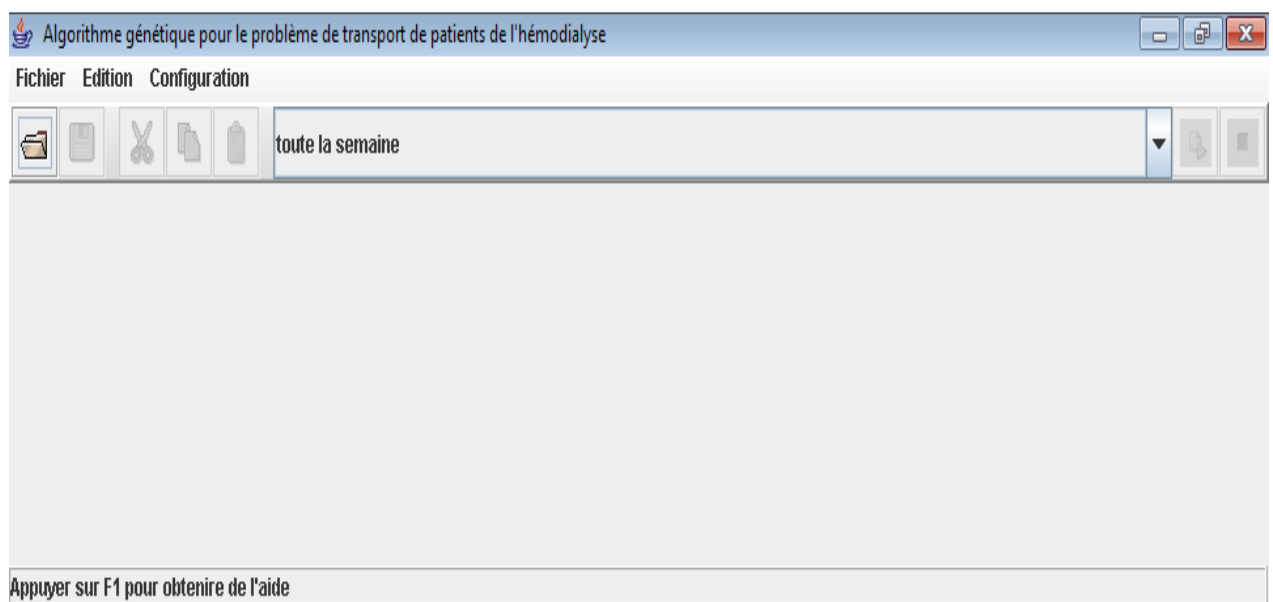
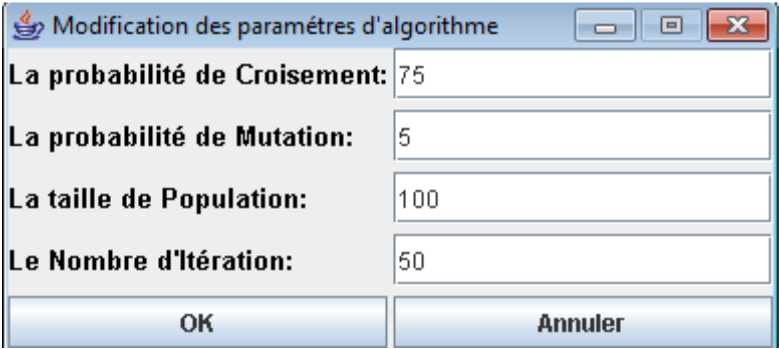


Fig. 5.3 l'interface de logiciel

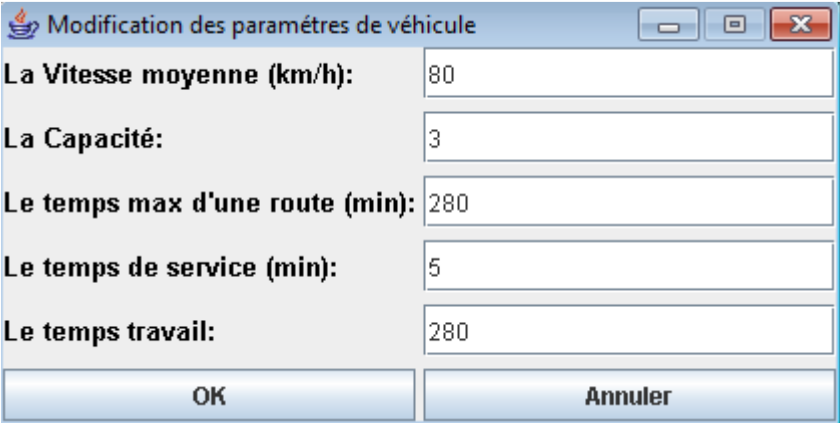
Les composants principaux de cette interface sont :

- *Fichier* : elle contient les opérations suivantes :
 - *Ouvrir* : permet de charger les données d'un problème situées dans n'importe quel endroit sur l'ordinateur.
 - *Enregistrer* : permet de mettre à jour les données modifiées d'un problème chargé sur l'application.
- *Edition* : permet d'éditer (couper, copier, coller, supprimer, supprimer tout) les données d'un problème chargé sur l'application.
- *Configuration* : permet d'afficher les paramètres de l'algorithme génétique dans une petite fenêtre (la figure 5.4), ou les paramètres du véhicule dans une autre fenêtre (la figure 5.5), pour les confirmer ou les modifier.



Modification des paramètres d'algorithme	
La probabilité de Croisement:	75
La probabilité de Mutation:	5
La taille de Population:	100
Le Nombre d'itération:	50
<div>OK Annuler</div>	

Fig. 5.4 la fenêtre des paramètres de l'algorithme génétique



Modification des paramètres de véhicule	
La Vitesse moyenne (km/h):	80
La Capacité:	3
Le temps max d'une route (min):	280
Le temps de service (min):	5
Le temps travail:	280
<div>OK Annuler</div>	

Fig. 5.5 la fenêtre des paramètres du véhicule

Dans la barre d'outils, le combo box représente le choix du jour, il est initialisé à la valeur « toute la semaine », donc il va afficher les demandes de toute la semaine, avant l'exécution il faut spécifier un jour de la semaine pour traiter seulement les demandes de ce jour.

Les deux dernières icônes concernant l'exécution de l'algorithme génétique sur le problème sélectionné, la première icône pour lancer l'exécution et la deuxième pour l'interrompre.

Après le chargement des données d'un problème la fenêtre sera divisée en trois zones comme suit :

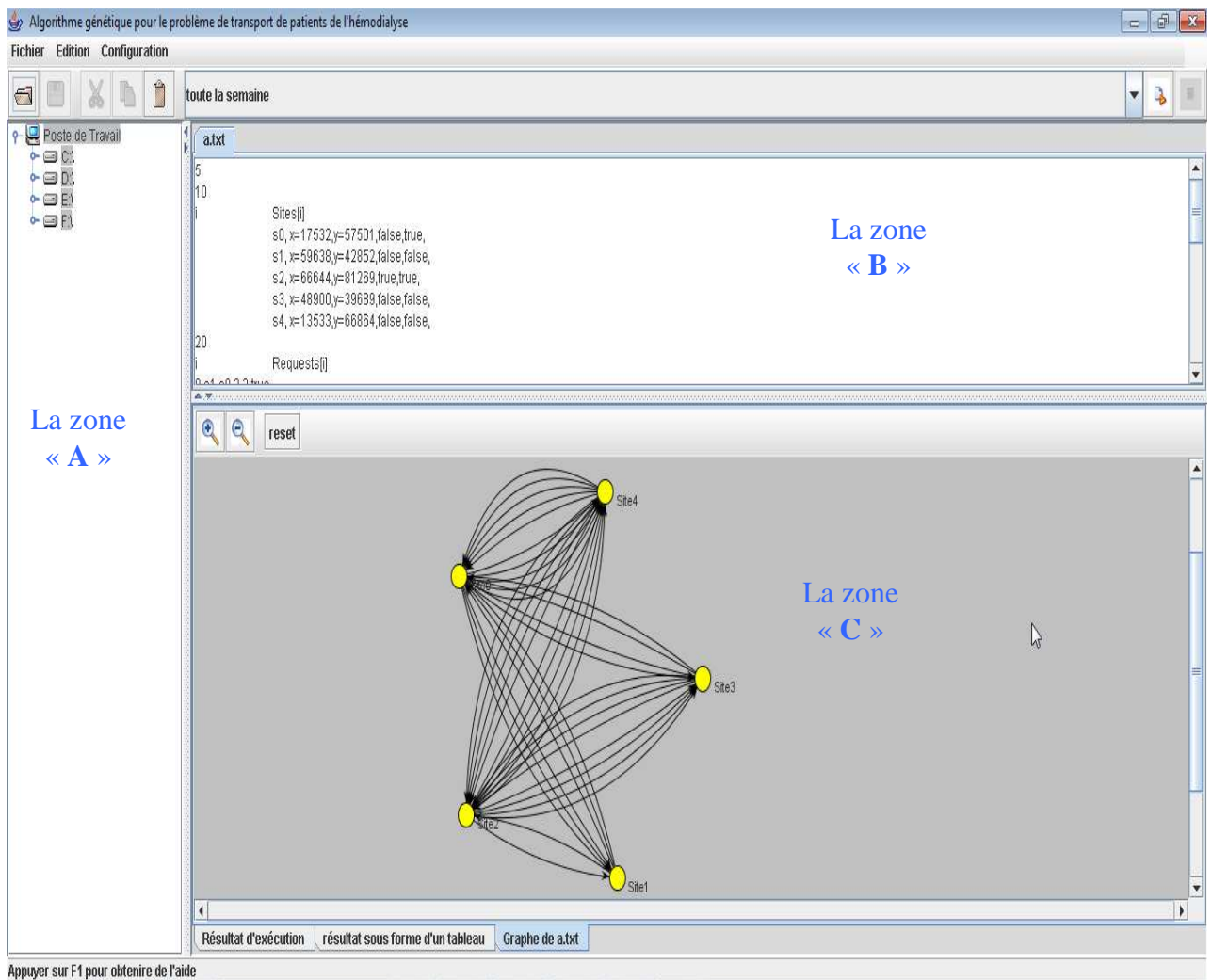


Fig. 5.6 l'interface après chargement d'un problème

La zone « A » concerne l'exploration du disque dur de l'ordinateur pour charger les données des problèmes.

La zone « **B** » concerne l’affichage textuel des données des problèmes chargés dans l’application, elle permet à l’utilisateur d’éditer ces données puis de les sauvegardées.

La zone « **C** » concerne l’affichage graphique des sites géographiques par des nœuds et les requêtes par des arcs entre ces nœuds.

Au lancement de l’exécution d’un problème chargé dans l’application, un autre anglet nommé « Résultat d’exécution » est apparu dans la zone « **C** », cet anglet contient la meilleur solution obtenue jusqu’à la génération en cours ainsi que le temps de réalisation de cette solution (fitness), cet résultat sera mettre à jour à chaque fois la solution est améliorée.

Si on maximise la zone « **C** » on obtient la fenêtre suivante :

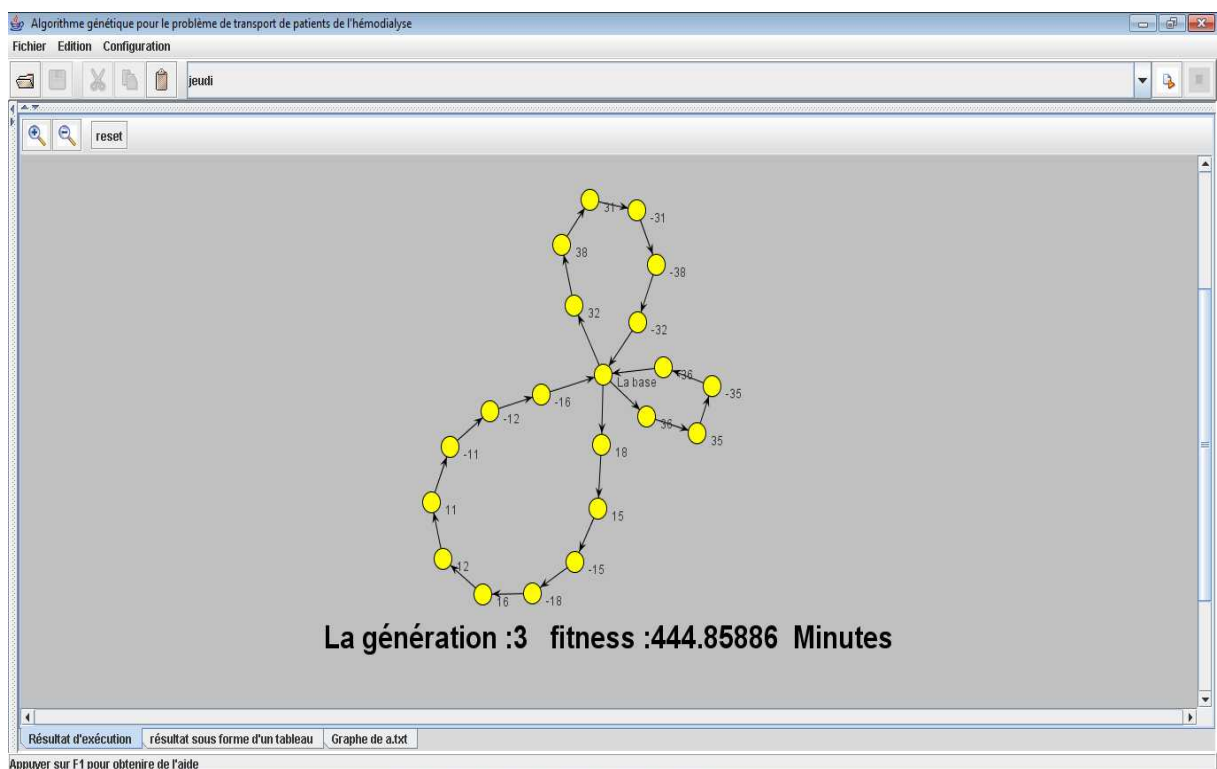


Fig. 5.7 exemple d'une solution d'un problème

5.6 Paramètres d'algorithme

Dans la littérature, il est souvent conseillé de prendre une grande probabilité de croisement $P_c \in [0.75, 0.95]$, et une petite probabilité de mutation $P_m \in [0.05, 0.1]$. Dans notre cas, et après avoir fait plusieurs simulations préliminaires, nous avons fixé P_c à 0.75 et P_m à 0.05. Afin d’augmenter les chances d’obtention de solution de qualité acceptable en un temps minimum, nous avons fixé la taille de population à 100 individus et le nombre d’itérations à 300.

5.7 Paramètres de véhicule

Concernant le véhicule, on a fixé sa vitesse moyen à 80 km/h, et sa capacité à 3 personnes (type VSL : Véhicule Sanitaire Léger). Pour le temps maximal d'une tournée et le temps de travail du véhicule on a fixé les deux à 280 minutes et le temps de service à 5 minutes.

5.8 Résultats

Dans cette partie, nous exposons les résultats obtenus pour le jeu de données utilisé. Ces résultats ont été obtenus par l'application de l'algorithme génétique développé, et par l'heuristique basée essaim. Nous signalons, pour une meilleure lecture des résultats, que tous les résultats sont exprimés en minutes.

5.8.1 Problèmes P10D5

Le tableau 5.1 présente les résultats du cout total des tournées obtenus pour les problèmes de type P10D5 par l'application de l'algorithme génétique développé.

Jours problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P10D5_1	386,52	410,48	479,82	358,41	446,61	302,27
P10D5_2	679,95	676,90	623,60	592,76	446,96	666,46
P10D5_3	833,62	355,48	831,46	638,41	839,67	501,37
P10D5_4	526,40	0	298,86	521,37	403,70	442,49
P10D5_5	427,87	74,95	355,53	487,89	291,74	291,27
P10D5_6	185,16	771,53	185,16	758,37	185,16	764,70
P10D5_7	938,54	910,67	557,94	911,17	674,66	431,34
P10D5_8	493,24	191,02	625,24	186,92	562,48	186,92
P10D5_9	421,93	331,21	358,69	294,31	377,53	228,36
P10D5_10	802,50	321,46	387,84	543,33	463,01	0

Tab. 5.1 Résultats du cout total des tournées obtenus par l'algorithme génétique pour les problèmes de type P10D5

Remarque : on a obtenu exactement les mêmes résultats par l'application de l'heuristique par essaim.

5.8.2 Problèmes de la série P20D5

Le tableau suivant illustre les résultats obtenus par l'application de l'algorithme génétique.

Jours Problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P20D5_1	753,42	727,85	513,35	786,38	607,36	857,17
P20D5_2	618,76	420,85	612,58	397,90	521,02	442,89
P20D5_3	791,83	614,03	606,84	569,00	651,55	615,07
P20D5_4	650,87	489,29	415,48	499,11	487,59	649,29
P20D5_5	671,66	684,33	424,27	779,21	607,35	940,27
P20D5_6	785,44	388,12	755,68	474,50	452,12	1140,54
P20D5_7	633,14	496,75	491,94	434,59	705,41	621,54
P20D5_8	869,80	288,48	796,41	197,72	592,74	868,59
P20D5_9	1047,33	482,46	899,07	489,00	655,76	616,27
P20D5_10	784,73	784,42	779,50	1007,17	482,22	1084,84

Tab. 5.2 Les résultats obtenus par l'AG sur les problèmes de P20D5

L'heuristique basée essaim a donné les résultats présentés dans le tableau 5.3 :

Jours Problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P20D5_1	753,42	727,85	513,35	795,94	608,27	863,24
P20D5_2	622,62	420,85	612,58	397,90	521,02	442,89
P20D5_3	791,83	614,03	606,84	569,00	651,55	615,07
P20D5_4	650,87	489,29	415,48	499,11	487,59	649,29
P20D5_5	671,66	684,33	424,27	801,43	607,35	941,04
P20D5_6	785,44	388,12	755,68	474,50	452,12	1224,06
P20D5_7	633,14	496,75	491,94	434,59	705,41	621,54
P20D5_8	871,28	288,48	796,41	197,72	592,74	869,35
P20D5_9	1050,64	482,46	899,07	489,00	655,76	616,27
P20D5_10	784,73	784,42	779,50	1007,17	482,22	1084,84

Tab. 5.3 Les résultats obtenus par l'heuristique basée essaim sur les problèmes de P20D5

5.8.3 Problèmes de la série P40D10

Le tableau suivant illustre les résultats obtenus par l'application de l'algorithme génétique.

Jours Problèmes	Samedi	Dimanche	Lundi	Mardi	Mercredi	Jeudi
P40D10_1	1726,26	1925,71	1525,81	2346,04	1220,68	2036,51
P40D10_2	1211,51	1835,96	1546,03	1813,69	1277,16	1641,99
P40D10_3	1750,57	937,71	1434,53	764,68	1126,76	866,81
P40D10_4	1916,48	1188,72	1701,13	1515,44	1287,09	1711,58
P40D10_5	814,25	1599,03	776,42	1223,24	804,64	1442,63
P40D10_6	1529,91	1554,79	1348,07	2063,4	1960,95	1835,63
P40D10_7	985,88	1147,28	1128,43	993,43	959,20	850,28
P40D10_8	1638,35	2351,12	1073,49	1996,27	1650,19	1432,90
P40D10_9	1296,32	1090,13	1188,15	1205,65	759,68	1204,92
P40D10_10	1718,47	1530,71	1309,23	1122,05	1334,92	1578,52

Tab. 5.4 Les résultats obtenus par l'AG sur les problèmes de P40D10

Concernant l'heuristique basée essaim, les résultats obtenu sont présentés dans le tableau 5.5

Jours Problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P40D10_1	1915,57	1978,31	1536,05	2548,82	1220,68	2070,13
P40D10_2	1211,51	2003,1912	1548,16	1995,91	1266,71	1656,22
P40D10_3	1861,39	1276,23	1543,35	764,68	1150,83	866,81
P40D10_4	1887,93	814,25	1701,16	1536,32	1287,09	1775,77
P40D10_5	814,25	1753,77	776,42	1287,04	804,64	1485,37
P40D10_6	1613,51	1667,99	1379,48	2292,87	1977,90	1912,93
P40D10_7	1006,52	1196,96	1128,43	1005,36	987,47	863,95
P40D10_8	1642,70	2277,27	1073,49	2001,08	1774,157	1431,15
P40D10_9	1356,70	1141,20	1185,21	1192,00	756,00	1261,82
P40D10_10	1732,53	1690,53	1353,47	1164,88	1377,85	1742,59

Tab. 5.5 Les résultats obtenus par l'heuristique basée essaim sur les problèmes de P40D10

A partir des résultats présentés précédemment, on remarque que l'algorithme génétique développé et l'heuristique basée essaim ont donné les mêmes résultats concernant les instances de la série des problèmes P10D5.

Pour les séries de P20D5 et P40D10, l'algorithme génétique est plus efficace que l'heuristique basée essaim et surtout pour les instances ayant un grand nombre de requêtes.

5.9 Analyse des résultats

5.9.1 Problèmes P10D5

Le tableau 5.6 illustre les pourcentages moyens du temps ajoutés au temps de transport entre les deux sites de chaque demande ; par exemple si la distance entre deux sites d'une demande nécessite 100 minutes, mais dans notre solution elle a pris 110 minutes, donc le pourcentage du temps ajouté est 10%.

Jours problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P10D5_1	43,32	24,48	22,76	16,37	10,05	0
P10D5_2	6,04	23,32	17,02	29,00	0,0	21,72
P10D5_3	19,95	0	24,70	0	11,15	0,0
P10D5_4	23,40	0	3,48	1,84	3,23	42,50
P10D5_5	0	0	17,98	18,33	0	20,87
P10D5_6	0	19,84	0	18,27	0	9,56
P10D5_7	4,59	21,07	9,50	20,66	1,26	25,53
P10D5_8	29,89	0	32,82	0	18,43	0
P10D5_9	40,37	14,24	15,84	20,81	32,91	0
P10D5_10	12,45	0	20,61	4,87	1,90	0

Tab. 5.6 Le pourcentage moyen du temps ajoutés au temps nécessaire pour servir les demandes pour P10D5

Le tableau 5.7 illustre le temps moyen des tournées effectuées dans chaque jour.

Jours problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P10D5_1	128,84	136,82	119,95	119,47	148,87	100,75
P10D5_2	226,65	225,63	207,86	197,58	223,48	222,15
P10D5_3	208,405	177,74	207,86	212,80	209,918	167,12
P10D5_4	131,60	0	99,62	173,79	134,56	147,49
P10D5_5	142,62	37,47	177,76	162,63	145,87	145,63
P10D5_6	92,58	192,88	92,58	151,67	92,582	191,17
P10D5_7	234,63	182,13	278,97	151,86	224,88	143,78
P10D5_8	164,41	95,513	156,31	93,46	187,49	93,46
P10D5_9	140,64	110,40	179,34	98,10	125,84	114,18
P10D5_10	200,62	160,73	129,28	181,11	154,33	0
Moyen par jour	167,10	146,59	164,95	160,18	154,24	147,30

Tab. 5.7 Le temps moyen d'une tournée par jour pour la série des problèmes P10D5

Les données du tableau 4.6 montrent que les résultats obtenus sont acceptables vis-à-vis du critère de temps de transport des requêtes, vu que le pourcentage moyen du temps ajouté au temps de transport ne dépasse pas 25% dans la plus part des problèmes (le moyen général est 17%), ce qui montre une bonne qualité de service offert aux clients.

Nous remarquons dans le tableau 4.7 que le temps de la route dans tous les problèmes est en moyen de 156 minutes, c'est-à-dire que les véhicules n'exploitent que 55% du temps maximale de la route, ceci est dû à la taille du problème qui est petite ou à l'utilisation de la fonction *Split* qui prend comme objectif de la décomposition le meilleur coût sans tenir compte le nombre des routes.

Afin de voir l'enchaînement de réalisation de ces solutions sous forme des routes, on donne des captures des solutions de quelques instances qui sont les suivants :

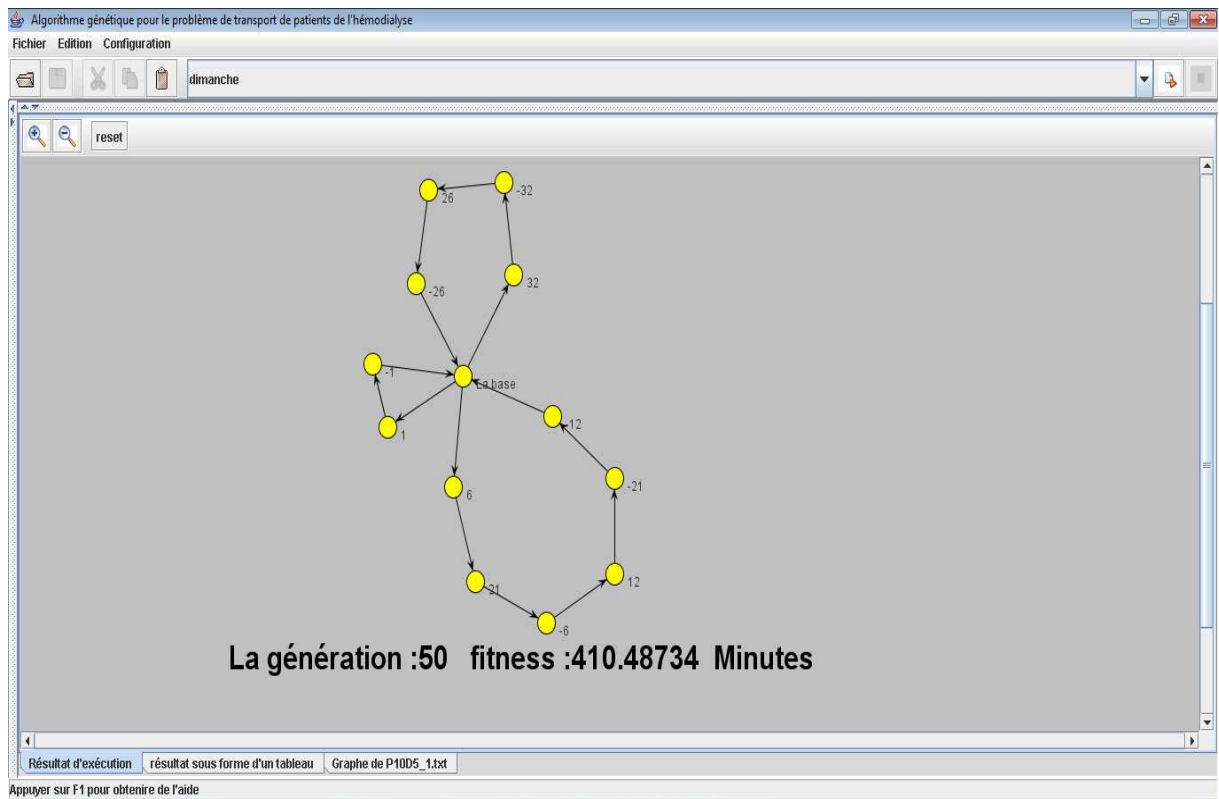


Fig. 5.8 la solution de l'instance P10D5_1 pour les demandes de dimanche

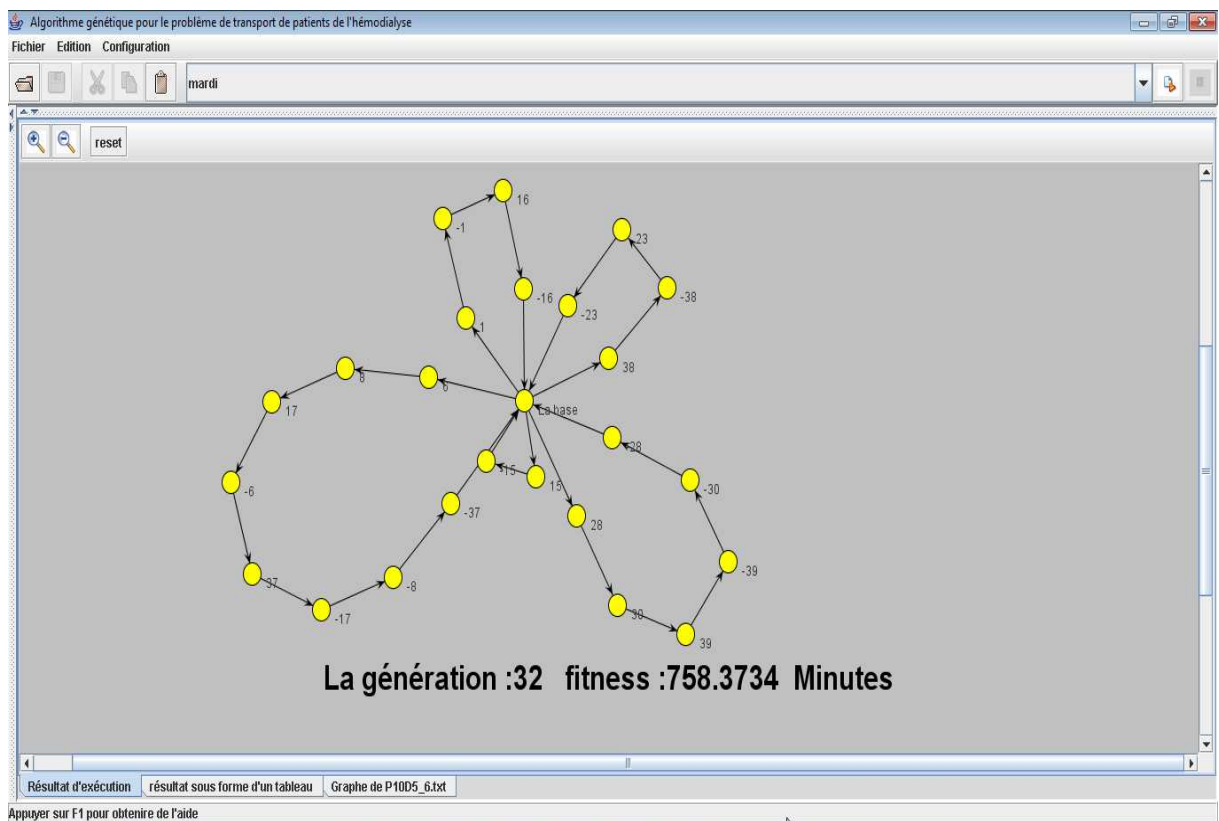


Fig. 5.9 la solution de l'instance P10D5_6 pour les demandes de mardi

5.9.2 Problèmes P20D5

Jours Problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P20_D5_1	27,53	28,76	16,87	39,31	62,26	26,42
P20_D5_2	34,46	16,92	36,25	28,22	25,60	20,63
P20_D5_3	33,24	41,25	34,84	29,97	18,53	23,17
P20_D5_4	21,57	29,69	25,73	35,18	22,93	8,68
P20_D5_5	31,75	24,31	30,47	14,26	15,32	12,00
P20_D5_6	16,59	53,79	24,43	40,55	22,06	17,30
P20_D5_7	25,53	11,03	12,22	33,81	17,56	27,04
P20_D5_8	15,14	0	7,03	10,41	29,19	12,79
P20_D5_9	19,76	26,27	26,84	19,42	33,38	21,15
P20_D5_10	18,02	30,19	25,37	9,75	8,71	20,52

Tab. 5.8 Le pourcentage moyen du temps ajoutés au temps nécessaire pour servir les demandes pour P20D5

Le tableau 5.9 montre le temps moyen de la tournée par jour.

Jours Problèmes	samedi	dimanche	lundi	mardi	mercredi	jeudi
P20_D5_1	150,68	181,96	171,11	196,59	202,45	177,88
P20_D5_2	206,25	140,28	204,19	132,63	130,25	147,34
P20_D5_3	158,36	204,67	202,28	189,66	217,18	153,76
P20_D5_4	130,17	163,09	138,49	124,77	162,53	108,21
P20_D5_5	134,33	143,25	141,42	200,73	202,45	196,38
P20_D5_6	131,03	129,37	190,8875	158,16	150,70	160,22
P20_D5_7	158,28	165,58	122,98	144,86	176,35	124,95
P20_D5_8	193,36	96,16	199,10	98,86	148,18	191,95
P20_D5_9	214,79	160,82	183,96	163	224,66	205,42
P20_D5_10	196,18	196,10	194,87	168,14	160,74	204,14
Temps moyen	167,34	158,12	174,92	157,74	177,54	167,02

Tab. 5.9 Le temps moyen d'une tournée par jour pour la série des problèmes P20D5

A travers les résultats présentés dans le tableau 5.8, nous pouvons remarquer que les pourcentages moyens du temps ajouté au temps de transport et un peu plus grand que dans la série de P10D5 ; il est en moyen de 24%, cela revient au nombre de demandes qui est plus grand, mais comme qualité de service, il reste bon pour les clients.

Le tableau 5.9 montre que le temps moyen de la tournée est 167 minutes, donc il représente 60% du temps maximal, il est mieux que dans les problèmes de la série P10D5, ceci justifie l'influence de la taille du problème sur le temps moyen de la tournée.

Parmi les solutions obtenues on a capturé les suivantes :

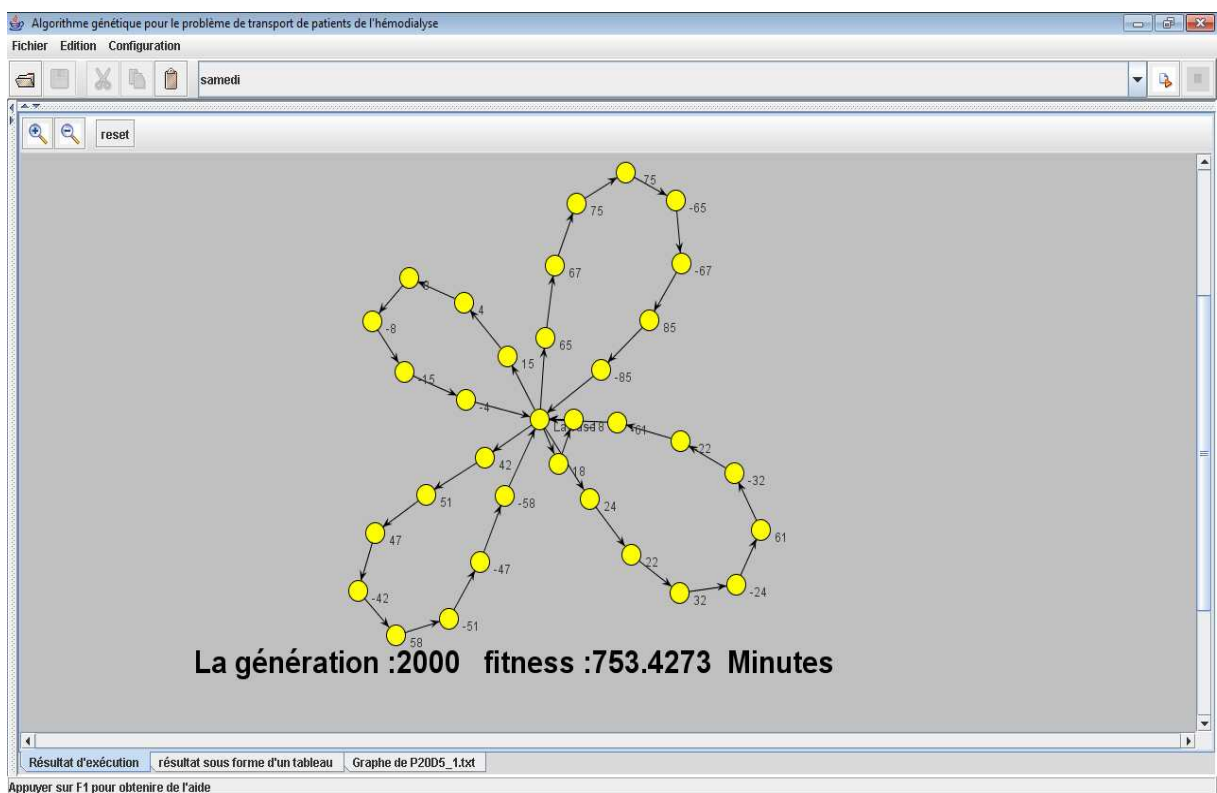


Fig. 5.10 la solution de l'instance P20D5_1 pour les demandes de samedi

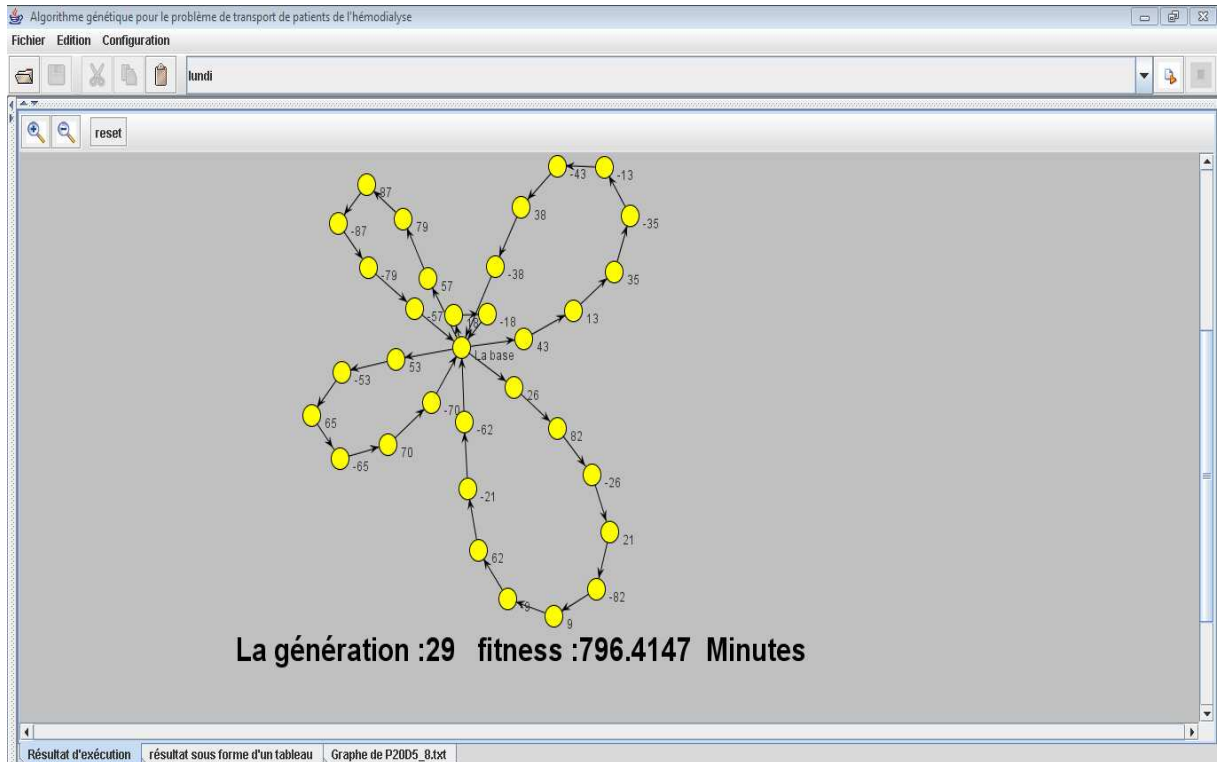


Fig. 5.11 la solution de l'instance P20D5_8 pour les demandes de lundi

5.10 Les données réelles

Ibn Sina assistance est un opérateur de transport sanitaire qui travail au niveau de la wilaya de Jijel, se situe dans la commune de Jijel à coté de l'hôpital Mohammed Sadik Ben Yahya. La flotte est composée de 16 véhicules de type VSL (Véhicule Sanitaire Léger) et une ambulance, son principal service est de transporter les insuffisants rénale depuis ses domiciles vers différent centres de l'hémodialyse, et assure leurs retour à la fin des séances. Chaque centre de dialyse fait trois séances par jour.

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 en six jours par semaine (du samedi au jeudi).

La figure 5.12 montre une capture de la solution appliquée par l'opérateur Ibn Sina pour jeudi sachant que :

- Les positions représentent les séances du jour.
- Chaque position (séance) a un ensemble de demandes d'aller (vers le centre de l'hémodialyse) et un autre ensemble de retour.

- Les étiquettes bleues représentent les conducteurs des véhicules.
- Les autres étiquettes représentent les patients transportés par chaque conducteur.



Fig. 5.12 la solution appliquée par l'opérateur Ibn Sina pour jeudi

5.10.1 Résultats pour les données réelles

Résultats Jours	La réalité appliquée (Manuel)	L'heuristique basée essaim	L'algorithme génétique	Gain par heure
Samedi	3813,53	3532,8708	2724,76	18,14
Dimanche	3619,67	3388,8474	2560,79	17,64
Lundi	3262,76	2706,114	2219,88	17,38
Mardi	4091,13	3790,6477	3035,39	17,59
Mercredi	3324,78	2901,050 3	2264,47	17,67
Jeudi	3566,11	3209,5776	2591,91	16,23

Tab. 5.10 Résultats du cout total des tournées obtenus pour les données réelles

Dans le tableau 5.10, le gain représente la différence entre la solution appliquée par l'opérateur et la solution obtenue par l'algorithme génétique développé. On remarque que l'application de l'algorithme génétique permet de gagner en moyen de **17** heures de travail par jour, ce qui montre l'efficacité de l'algorithme génétique développé.

5.11 Conclusion

Dans ce chapitre nous avons présenté les jeux de données utilisés et la façon d'organiser les données d'un problème dans notre application, en suite nous avons fait une présentation de l'interface de cette dernière, en fin nous avons discuté les résultats des instances d'un jeu de données aléatoire, ainsi que les données réelles d'un opérateur de transport sanitaire.

L'heuristique basée essaim développé ne sert que pour validation les résultats obtenus par algorithme génétique développé

Notre objectif est de donner une solution automatique pour le jeu de données réel. Le gain obtenu en nombre d'heures permet de :

- ✓ Réduire la facture payée par les services de sécurités sociales.
- ✓ La minimisation du temps de transport d'un patient.

Conclusion générale

Les problèmes de tournées de véhicules font partie intégrante de la vie quotidienne des décideurs et des planificateurs. Il s'agit de déterminer les tournées d'une flotte de véhicules afin de livrer une liste de clients, ou de réaliser des tournées d'interventions (maintenance, réparation, contrôles) ou de visites (visites médicales, commerciales, etc.). Le but est de minimiser le coût de livraison des biens.

Dans le cadre de ce mémoire, nous nous sommes intéressés à une importante variante de ces problèmes qui est le problème de transport à la demande à plusieurs véhicules de capacité limités et avec fenêtre de temps dans le cas statique c'est-à-dire toutes les demandes des clients sont connues à l'avance (Static Dial-A-Ride Problem). Ce problème, consiste à déterminer les tournées et les horaires pour les véhicules qui effectuent le transport d'usagers à leur demande, d'une origine à une destination en minimisant le coût total des tournées.

Avant d'aborder la résolution de ce problème, nous avons présenté les différents problèmes de d'élaboration de tournées de véhicules. Nous avons détaillé le VRP (Vehicule Routing Problem) qui constitue le problème originel des différentes variantes aussi présentées et dont fait partie le DARP. Nous avons ainsi présenté un état de l'art de ces deux problèmes ainsi que leurs formulations mathématiques respectives.

En suite, après avoir rappelé le principe et les caractéristiques de l'algorithme génétique, nous avons proposé notre approche de résolution du problème de transport à la demande, cette approche est basée principalement sur l'algorithme génétique avec une nouvelle technique qui consiste à faire le croisement et la mutation non pas sur l'individu en entier, mais seulement sur la succession d'ordre d'exécution des requêtes.

Cette technique permet d'éviter les problèmes de violation des contraintes lors de l'application des deux étapes de croisement et de mutation, et donc d'accélérer le processus d'évolution.

En fin nous avons exposé les résultats d'expérimentation effectués sur un jeu de données conçu aléatoirement, et des données réelles d'un opérateur de transport sanitaire.

Afin de mesurer l'efficacité de l'algorithme génétique, nous avons implémenté une heuristique basée essaim qui utilise deux voisinages (un voisinage basé sur l'expérience et un autre voisinage local).

Les tests effectués sur ces problèmes ont montré que notre approche fait une bonne combinaison entre le coût total de transport et la durée de transport pour chaque client.

Pour les données réelles, l'algorithme génétique développé a prouvé son efficacité par un gain de 17 heures de travail par jour, ce gain permet de :

- ✓ Réduire la facture payée par les services de sécurités sociales.
- ✓ La minimisation du temps de transport d'un patient.

Malgré les bons résultats obtenus, on remarque que le temps des tournées reste moyen à cause de la difficulté d'exploration de l'espace des solutions pour les problèmes de grande taille, pour cela, et dans une perspective d'amélioration de notre approche, nous nous proposons :

- D'améliorer l'heuristique d'insertion des nœuds de déchargement pour augmenter le degré d'exploration de l'espace de solution par des méthodes basées sur des formules mathématiques, ou par des heuristiques de recherche locale.
- De prendre en compte la qualité des individus générés pour la population initiale afin d'accélérer la convergence vers la solution optimale.
- D'intégrer d'autres contraintes comme l'état du patient pour traiter le cas de la flotte hétérogènes de véhicules.

Références bibliographiques

- [Archetti et al. 2002] C. Archetti, A. Hertz and U. Derigs “A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem”. *Mathematics of Information Technology and Complex Systems*, 2002.
- [Attanasio et al. 2004] Attanasio A, Cordeau JF, Ghiani G, Laporte G. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*. 2004;30:377–387.
- [Beaudry et al. 2008] Beaudry A, Laporte G, Melo T, Nickel S. Dynamic transportation of patients in hospitals. *O.R. Spectrum*. 2008;32:1-31.
- [Bodin et al.1979] L. Bodin and L. Berman. “Routing and scheduling of school buses by computer”. *Transportation Science* 13(2), pages 113-129, 1979.
- [Bono et Lutt 1988] E. Bonomi et J.-L. Lutton ; le recuit simulé pour la science ; numéro 129, pages 68-77 ; Juillet 1988.
- [Borndörfer et al. 1997] JR. Borndörfer, F. Klostermeier, M. Grötschel, and C. Küttner. *Telebus berlin: vehicle scheduling in a dial-a-ride system*. Technical report, SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1997.
- [Borne et al 1990] P. Borne, G.D. Tanguy, J.P. Richard, F. Rotella, I. Zambettakis, *Commande et optimisation de processus*, Edition Technip, France, 1990.[CDG 1999] D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*. McGraw-Hill, 1999.
- [CLR 1990] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*, chapter 15, pages 350—355. The MIT Press, Cambridge, Massachusetts, 1990.
- [Colorni et al. 1992] A. Colorni, M. Dorigo, and V. Maniezzo. “Distributed optimization by ant optimization colonies”. *Proceeding of the first European Conference on Artificial Life (ECAL 91)*, pages p. 134-142, 1992.
- [Cordeau et al. 2000] J.-F Cordeau, G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis. “The VRP with Time Windows”. Technical Report, Département de Mathématiques et de Génie Industriel, Ecole polytechnique de Montréal, Canada, June 2000.

- [Cordeau et al. 2003] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B*, 37 :579_594, 2003.
- [Cordeau et al. 2007] Cordeau JF, Laporte G, Potvin JY, Savelsbergh MWP. Transportation on demand. In: *Transportation*, editor. *Handbooks in Operations Research and Management Science*. Vol 14. North-Holland, Amsterdam 2007. p. 429-466.
- [Cormen et al., 1990] Cormen, T., CE, L., and RL, R. (1990). *Introduction to algorithms*. Cambridge MA:MIT Press.
- [Coslovich et al. 2006] Coslovich L, Pesenti R, Ukovich W. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*. 2006;175:1605-1615.
- [Cubillos et al. 2007] C. Cubillos, N. Rodriguez, and B. Crawford. A study on genetic algorithms for the darp problem. *IWINAC*, I:498_507, 2007.
- [Dav 1991] L Davis. *Handbook of Genetic Algorithms*. Van nostrand reinhold, New-York, 1991.
- [Desrosiers et al. 1991] J. Desrosiers, Y. Dumas, F. Soumis, S. Taillefer, and D. Villeneuve. An algorithm for mini-clustering in handicapped transport. Technical report, *Les Cahiers du GERAD*, G-91-02, HEC Montréal, 1991.
- [Dumas et al. 1989] Y. Dumas, J. Desrosiers, and F. Soumis. Large scale multi-vehicle dial-a-ride problems. Technical report, *Les Cahiers du GERAD*, G-89-30, HEC Montréal, 1989.
- [Esqui 2001] P. Esquirol, P. Lopez, Concepts et méthode de base en ordonnancement de la production. Chapitre dans l'ouvrage: *Ordonnancement de la production*, Edition Hermès, France, 2001.
- [Fischetti et al. 1999] M. Fischetti, A. Lodi and P. Toth "A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem". Dipartimento di Elettronica e Informatica, Università di Padova, Italy, 1999.
- [FSJP 1993] S Forrest, R.E Smith, B Jakornik, et A.S Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191—211, 1993.
- [Fu et al. 2003] Z. Fu, R. W. Eglese and L. Li "A tabu search heuristic for the open vehicle routing problem". LUMS working paper, 2003.
- [FW 1993] S.J Flockton et M.S White. Pole-zero system identification using genetic algorithms. Dans *Proceedings of the Fifth International Conference on Genetic Algorithm*. ICGA, 1993.

- [Jaw et al. 1986]** Jaw J, Odoni A, Psaraftis H, Wilson N. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research*. 1986;B 20:243–257.
- [Gambardella et al. 2003]** L.M. Gambardella, A.E. Rizzoli, F. Oliverio, N. Casagrande, A.V. Donati, R. Monternanni and E. Lucibello. “Ant Colony Optimization for vehicle routing in advanced logistics systems” IDSIA, Galleria 2, 6928 Manno, Switzerland and AntOptima, via Fusion 4, 6900 Lugano, Switzerland, 2003.
- [Gendreau et al 1998a]** M. Gendreau and J.Y. Potvin. “Dynamic vehicle routing and dispatching”. Working paper, Université de Montréal, 1998.
- [Gendreau et al. 1994]** M. Gendreau, A. Hertz, G. Laporte “A tabu search heuristic for the vehicle routing problem”, *Manag. Sci.*, vol. 40, pp. 1276-1290, 1994.
- [Glover 1989]** F. Glover, “Tabu Search, Part I”, *ORSA Journal on Computing* 1(3), pp. 190-206.
- [Glover 1990]** F. Glover, “Tabu Search, Part II”, *ORSA Journal on Computing* 2(1), pp. 4-32.
- [Gold 1989b]** D.E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley, 1989.
- [Golden et al. 1982]** B. Golden, A. Assad, A. Levy and F. Ghemais “The fleet size and mix vehicle routing problem”, *Management Science and Statistics*. Working paper, n° 82-020, 1982.
- [Groth 2002]** M. J. Groth “Stochastic considerations in vehicle routing Problems”. *Transport Optimization*, June 17 2002.
- [Hol 1975]** J.H Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1975.
- [Horn 2002a]** Horn MET (2002a) Fleet scheduling and dispatching for demand-responsive passenger services. *Transport Res C-Emer* 10:35–63.
- [HSL 1993]** A Homaifar, G Shanguchuan, et G.A Liepins. A new approach on the traveling salesman problem by genetic algorithms. Dans *Proceedings of the Fifth International Conference on Genetic Algorithms*. ICGA, 1993.
- [Jacobs-Blecha et al. 1998]** C. Jacobs-Blecha and M. Goetschalckx. “The vehicle Routing Problem with Backhauls: Properties and Solution Algorithms”.
- [Joh54]** S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1 :61—68, 1954.

- [KBH 1994] S Khuri, T Back, et J Heitkoter. The zero/one multiple knapsack problem and genetic algorithm. Dans *Proceedings of the Symposium of Applied Computation*. ACM, 1994.
- [Kilby et al. 1998] P. Kilby, P. Prosser, and P. Shaw. "Guided Local Search for the Vehicle Routing Problem with Time Windows". *METAHEURISTICS Advanced and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publisher, Boston, 1998.
- [Laporte et al 1992] G. Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms". *European Journal of Operational*.
- [LeBouthillier 2000] A. LeBouthillier. "Modélisation UIVIL pour une architecture coopérative appliquée au problème de tournées de véhicules avec fenêtres de temps". Technical Report, Département d'informatique et de recherche opérationnelle. Faculté des arts et des sciences. Université de Montréal, Canada, Avril 2000.
- [Madsen et al. 1995] Madsen OBG, Ravn HF, Rygaard JM. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*. 1995;60:193–208.
- [Mechti 1995] R. Mechti, "Tournée de véhicules à la demande: problèmes et méthodes". Technical report, Laboratoire PRiSM, Université de Versailles-St. Quentin, France, 1995.
- [Mos 1989] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report 826, California Institute of Technology, Pasadena, California, USA, 1989.
- [Papa 1976] C.H. Papadimitriou. The complexity of combinatorial optimization problems. PhD thesis, Princeton University, New Jersey, USA, 1976.
- [Papa et Stei 1982] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity*. Prentice-Hall, 1982.
- [Prins 2002] C. Prins, "Efficient heuristics for the heterogeneous fleet multitrip vehicle routing problem". *Journal of Mathematical Modelling and Algorithms*, 1 (2), pages pp. 135-150, 2002.
- [Psaraftis 1980] H. Psaraftis, "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem", *Transportation Science* 14.130-154(1980).

- [Psaraftis 1983a]** H.N. Psaraftis, "An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem with time windows". *Transportation Science*, 17, 351-357, 1983.
- [Ralphs 2002]** T.K. Ralphs, "Parallel branch and cut for capacitated vehicle routing". *Parallel Computing archive*, Volume 29, Issue 5, 2 December 2002
- [Ralphs et al 2001]** T.K. Ralphs, L. Kopman, W.R. Pulleyblank and L.E. Trotter, "On the capacitated vehicle routing problem", *Mathematical Programming* 94, 343-359, 2001.
- [Rego et al 1994]** C. Rego et C. Roucairol. "Le problème de tournées de véhicules: Etude et Résolution Approchée". Technical Report, inria, Février 1994.
- [Rego et al.1996]** C. Rego, C. Roucairol, "A Parallel Tabu Search Algorithm using Ejection Chains for the Vehicle Routing Problem". In I.H. Osman and J.P. Kelly (eds), *MetaHeuristics: Theory and Applications*. Kluwer, Boston, 1996.
- [Rekiek et al. 2006]** B. Rekiek, A. Delchambre, and H. A. Saleh. Handicapped person transportation: an application of the grouping genetic algorithm. *Engineering Application of Artificial Intelligence*, 19 :511_520, 2006.
- [Ryan et al. 1993]** Ryan, D.M. Hjorring, C., Glover, F. "Extensions of the petal method for vehicle routing", *Journal of the Operational Research Society* 44, 289-296, 1993.
- [Sait et Youssef 1999]** S. M. Sait et H. Youssef; iterative computer algorithms with applications in engineering: solving combinatorial optimization problems, *IEEE computers society*, 1999.
- [Savelsbergh 1992]** M. W. P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal on Computing*, 4 :146_154, 1992.
- [Savelsbergh 1995b]** M.W.P. Savelsbergh, "Local Search for Routing Problems with Time Windows". *Annals of Operations Research* 4, 285-305, 1995.
- [sexton et al. 1985 a]** T.R. Sexton, L.D. Bodin, "Optimising single vehicle many-to-many operations with desired delivery time: I. Scheduling", *Transportation Science* 19(4), 378-410, 1985.

- [Sexton et al. 1985 b] T.R. Sexton, L.D. Bodin, "Optimising single vehicle many-to-many operations with desired delivery time: I I. routing", *Transportation Science* 19(4), 411-435, 1985.
- [SRD 1993] M Schoenauer, E Ronald, et S Damonr. Evolving nets for control. Dans *Proceeding of the Sixth International Conference on Neural Networks and their Industrial and Cognitive Applications*. AFIA, 1993.
- [Taillard 1993] E. D. Taillard, "Parallel iterative search methods for vehicle routing problems". *Networks*, vol. 23, pp. 66 1-673, 1993.
- [Taillard 1999] E.D. Taillard, "A heuristic column generation method for the heterogeneous fleet VRP". *RAIRO-Operations Research*, 33, pages PP. 1-14, 1999.
- [UM 1993] R Unger et J Moul. A genetic algorithm for 3d protein folding simulations. Dans *Proceedings of the Fifth International Conference on Genetic Algorithm*. ICGA, 1993.
- [Witucki et al. 1997] M. Witucki, P. Dejax and M. Haouari. "Un modèle et un algorithme de résolution exacte pour le problème de tournées de véhicules multipériodique". Ecole Centrale Paris, Laboratoire Productique-Logistique, Grande Voie des Vignes, 92295 Châtenay-Malabry. Ecole Polytechnique de Tunisie, 2070 La Marsa, Tunisia 1997.
- [WO 1993] H Watabe et N Okino. A study on genetic shape design. Dans *Proceedings of the Fifth International Conference on Genetic Algorithm*. ICGA, 1993.
- [Wong et al. 2006] K. I. Wong and M. G. H. Bell. Solution of the dial-a-ride problem with multidimensional capacity constraints. *International Transactions in Operational Research*, 13 :195_208, 2006.
- [Xiang 2006] Z. Xiang, C. Chu, and H. Chen. A fast heuristic for solving a large-scale static dial-aride problem under complex constraints. *European Journal of Operational Research*, 174 :1117_1139, 2006.
- [Xiang 2008] Xiang Z, Chu C, Chen H (2008) The study of a dynamic dial-a-ride problem under time dependent stochastic environments. *Eur J Oper Res* 185:534–551.

- [Xu et al. 1996] J. Xu, J. P. Kelly, “A network flow-based tabu search heuristic for the vehicle routing problem”, *Transportation Science* 30,379-393, 1996.
- [YG 1991] X Yin et N Gernay. Investigations on solving the load flow problem by genetic algorithms. *Electric Power Systems Research*, 22:151 163, 1991.

Les livres utilisés

- Paolo Toth, Danielle VIGO. 2002: “The Vehicule Routing Problem”. SIAM; Society for Industrial and Applied Mathematics. Philadelphia
- Colin R.reeves, Jonathan E.rowe.2003: “Genetic Algorithms: Principles and Perspectives. A Guide to GA Theory”. Kluwer Academic Publishers.
- S.N.Sivanandam, S.N.Deepa.2008:” Introduction to Genetic Algorithms”. Springer
- Mitchell Melanie.1999:”An Introduction to Genetic Algorithms”. A Bradford Book the Mit Press.
- Sophie N. Parragh · Karl F. Doerner · Richard F. Hartl .2008: “A survey on pickup and delivery problems .Part II: Transportation between pickup and delivery locations”. Springer.
- Francisco Baptista Pereira and Jorge Tavares.2009: “Bio-inspired Algorithms for the Vehicle Routing Problem”. Springer.

Les thèses utilisés

(Saadi Leila): “ Optimisation Multiobjectifs par Programmation Génétique”. Université de batna, département informatique.08/07/2007

(Kaifeng Zeng): “Dynamic Vehicle Routing Problem with Backhaul and Time Window and Its Application in the Less-Than-Truckload (LTL) Trucking Industry”. A thesis submitted to the division of graduate studies and research of university of Cincinnati, 03/02/2006.

Web

[Labosun] <http://www.labo-sun.com>

RÉSUMÉ

Ce mémoire porte sur les services de transport à la demande offerts à des personnes à mobilité réduite, typiquement des personnes âgées, malades ou handicapées...

Le problème de transport à la demande (DARP) est un problème d'optimisation qui consiste à déterminer les tournées et les horaires pour les véhicules qui effectuent le transport d'usagers à leur demande, d'une origine à une destination. C'est un cas particulier du problème de tournées de véhicule (VRP) qui appartient à la famille des problèmes NP-complets d'où la nécessité de l'utilisation des méthodes approchées ((méta-) heuristiques) pour résoudre ce type de problèmes.

Nous nous intéressons dans ce travail au transport des patients aux différentes unités de l'hémodialyse, sachant que le nombre de ces patients dans notre pays augmente d'une façon très rapide ces dernières années (14000 malades en 2011, 15700 en mars 2012), cela pose un problème vraiment difficile que ce soit pour les opérateurs de transport sanitaire que pour les unités de l'hémodialyse.

Pour résoudre ce problème on a proposé un algorithme génétique, qui est testé sur des données générées aléatoirement et des données réelles d'un opérateur de transport sanitaire.

Mots-clés: Méta-heuristique, Problème de transport, Réseau de transport, NP-complets, VRPTW, DARP.

ملخص

هذه المذكرة تتناول خدمات النقل حسب الطلب المعروضة للأشخاص الذين لهم عجز في التنقل أمثال الكبار في السن، المرضى أو ذوي الاحتياجات الخاصة...

مشكل النقل حسب الطلب و الذي يرمز له بالانجليزية (DARP) هو أحد مشاكل التحسين و الذي يتمثل في تحديد مسارات وأوقات سيارات النقل التي تقوم بنقل الأشخاص حسب طلبهم، من منطقة ركوب إلى منطقة توصيل. هذا المشكل هو حالة خاصة من مشكل تحديد مسارات السيارات المعروف بالانجليزية باسم VRP و الذي ينتمي إلى مجموعة المشاكل المستعصية الحل و المعروفة بالانجليزية باسم NP-complete والتي تتطلب لحلها استعمال طرق تعطي نتائج جيدة و لكن ليس الحل الأمثل، هذه الطرق تسمى بالانجليزية ((meta-) heuristics approximate methods .

لقد اهتمنا في هذا العمل بمشكل نقل مرضى العجز الكلوي إلى مختلف وحدات تصفية الدم، بالعلم أن عدد هؤلاء المرضى في بلادنا يزداد بوتيرة سريعة جدا في الأعوام الأخيرة (14000 مريض في عام 2011 إلى 15700 في مارس 2012)، ما أدى إلى طرح مشكل صعب حقيقة ، سواء بالنسبة إلى عملاء النقل الصحي أو مراكز تصفية الدم.

لحل هذا المشكل، طرحنا خوارزمية يعتمد على مبدأ الوراثة (genetic algorithm) و الذي تم تجربته على معطيات تم تشكيلها عشوائيا و أخرى حقيقية حصلنا عليها من أحد عملاء النقل الصحي.

المفاتيح :

Méta-heuristic, Transportation problem, Transportation Network, NP-complete, DARP, VRPTW.