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

Nº Réf :……

Centre Universitaire
Abd elhafid Boussouf Mila

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

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

En : Informatique

Spécialité : Sciences et Technologies de l'information et de la communication (STIC)

Résolution de problème de tournée de véhicule à base d'une méta-heuristique

Préparé par : BOULAYOUNE SELMA
TRIA ZAHIA

Soutenue devant le jury

Encadré par A.KIMOUCHE …… M.A.B
Président A.HADJI…… …... M.A.A
Examinateur A.BAZANIAR…….. M.A.B

Année universitaire : 2015/2016

Dédicace

A chaque fois qu'on achève une étape importante dans notre vie, on fait une pose pour regarder en arrière et se rappeler toutes ces personnes qui ont partagé avec nous tous les bons moments de notre existence, mais surtout les mauvais.

Ces personnes qui nous ont aidés sans le leur dire, soutenus sans réserve, aimé sans compter, ces personnes à qui notre bonheur devient directement le leur, se transforme en pleur.

Nous dédie ce modeste travail en signe de reconnaissance et de respect.

A nos chers pères et A nos très chères mères

A nos frères et sœurs

A tous nos amis

A tous nos familles

A chaque personne sincère dans la vie

Remerciement

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Un remerciement particulier à notre encadreur Ms Kimouche Abdelkader pour sa présence, son aide et surtout pour ses précieux conseils qui nous ont assistés pour l'accomplissement de notre projet.

Nos vifs remerciements vont également aux membres du jury Ms A.Bazaniar et Ms Hadji Atman, pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Nos remerciements à nos très chers parents, frères, sœurs, amis respectives spécialement « KHAOULA » qui nous ont encouragés, soutenu durant tout notre parcours.

A l'opérateur de transport « Dianephros » sur tout son directeur pour son réception est son Soutien

AUX employés de la direction de transport pour leurs aides et leur gentillesse

Nous remercions les étudiants de la promotion 2015/2016

Résumé : Le problème de tournée de véhicule (VRP) est l'un des problèmes les plus connus dans 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,. Dans ce travail nous nous intéressons 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 réelles d'un opérateur de transport sanitaire.

Mots-clés: VRP, la recherche opérationnelle, l'optimisation combinatoire, l'hémodialyse, algorithme génétique.

ملخص :

ان مشكل تحديد مسار السيارات المعروف بالإنجليزية VRP هو واحد من المشاكل الاكثر شيوعا في مجال البحث العملي بالإضافة الي التعامل بين تخصصات البرمجة الرياضية والتحسين التوافقي من جهة ومديري النقل من جهة اخرى .في هذا العمل اهتممنا بمشكل نقل مرضى العجز الكلوي الى مختلف مراكز تصفية الدم، مع العلم ان عدد هؤلاء المرضى في بلادنا يزداد بوتيرة سريعة في الاعوام الاخيرة (14000 مريض في عام 2011 -15700 مريض في عام 2012), ما أدى لطرح مشكل صعب بالفعل, سواء لعملاء النقل الصحي او مراكز تصفية الدم. لحل هذا المشكل، طرحنا خوارزميا يعتمد على الوراثة Algorithme génétique والذي تم تجربه بمعطيات حقيقية حصلنا عليها من أحد عملاء النقل الصحي

المفاتيح:

VRP, la recherche opérationnelle, l'optimisation combinatoire, l'hémodialyse, algorithme génétique.

Abstract The vehicle routing problem is one of the problems very successes of the operational research. Further to a very narrow collaboration between the specialists of the mathematical programming and the combinatorial optimization, on one hand, and the administrators of transport, on the other hand. in this work We are interested 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 real data of an operator of medical transport.

***Key-words*:** Méta-heuristic, Transportation problem, Transportation Network, NP-complete, ginitic algorithm

Table des matières

Chapitre03 : Conception et architecture de l'application

Chapitre 04 : Résultats et discussion

Liste des figures

Chapitre 1 : Les problèmes de tournées de véhicules

Chapitre02 : les algorithmes génétiques

Chapitre03 : Conception et architecture de l'application

Chapitre 04 : Résultats et discussion

Introduction générale :

Le problème de tournée des véhicules (VRP) a été le sujet d'une recherche intensive durant plus de cinquante ans, liée à son importance dans le domaine de la logistique, et à sa grande difficulté. Le VRP implique la planification de routes de livraison à moindre cout, afin de servir un ensemble de clients dispersés géographiquement, tout en respectant les contraintes de capacité des véhicules. Ce problème d'optimisation combinatoire présente à la fois des caractéristiques d'allocation de ressources (répartition des charges dans les camions), et de construction de séquence (problème du voyageur de commerce).

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.

Le problème de tournées de véhicules est classé par la littérature parmi les problèmes de classe NP-difficile, c.-à-d. on ne peut pas obtenir la solution optimale dans un temps raisonnable surtout dans le cas où la taille du problème augmente, ce qui dirige vers la recherche d'une solution proche de l'optimale en utilisant des méthodes dites approchées et en particulier les métaheuristiques.

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 qui est le transport à la demande « Dial A Ride Problem », spécialement destiné au transport des personnes, dans notre cas on s'intéresse au transport des patients de l'hémodialyse. Notre but donc est de concevoir un outil de résolution de ce problème, à base des algorithmes génétiques, permettra d'offrir à l'opérateur de transport une solution de meilleure qualité.

Ce mémoire, est organisée autour de quatre 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, nous y détaillons le principe des algorithmes génétique, leurs caractéristiques, les opérateurs génétiques utilisés ainsi que les codages possibles à utiliser. Par la suite, le chapitre 3, est consacré à l'adaptation des algorithmes génétiques au problème considéré. Et le dernier chapitre présent les résultats obtenus.

Chapitre 01 :

Problèmes de tournée des véhicules

1.1 Introduction :

Le Problème de Tournées de Véhicules (VRP, Vehicle Routing Problem) est l'un des problèmes d'optimisation combinatoire les plus étudié. 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, dont le but est de trouver un ensemble des routes (tournées) Pour une flotte de véhicules, afin de satisfaire les demandes d'un ensemble de clients en minimisant le coût de transport avec des tournées débutant et terminant à un dépôt.

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 distribution de produits aux hypermarchés et magasins, la distribution de courrier…

Dans ce qui suit, nous présentons les variantes de VRP, ensuite nous donnons quelques notions de base de l'optimisation, avec la modélisation mathématique de problème de tournées de véhicules. Enfin, nous présentons les méthodes de résolution du VRP les plus connues.

1.2 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) :**

  Le problème de tournées de véhicules consiste à trouver un nombre de tournées de véhicules qui doivent visiter, par exemple des clients pour leur desservir leurs demandes. Ces demandes ont un poids et une forme bien déterminés. Les véhicules possèdent une capacité de poids à ne pas dépasser. Ainsi pour les déterminer il faut bien prendre en compte la contrainte sur la capacité maximale des véhicules exécutant les tournées [1].

- **DVRP (DynamicVehicleRoutingProblem) :**

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 [2].

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

Ou VRP avec fenêtres de temps quant à lui, spécifie que chaque client a une fenêtre de temps. Celle-ci est un intervalle de temps au cours duquel son service doit être accompli .un véhicule peut arriver plus tôt, avant le début de la fenêtre de temps, mais il doit attendre jusqu'à ce que le service soit possible. Dans ce cas le temps d'attente totale peut être pris en compte dans le modèle et peut représenter un objectif à minimiser [3]. S'il arrive plus tard le service ne peut pas être rendu et le client correspondant ne sera jamais satisfait. Dans ce cas, le problème est dit 'à 'contraintes dures' le nombre de véhicules peut être fixé d'avance, ou être considéré comme l'une des variable du problème [4], l'objectif du problème est alors de minimiser la distance totale et le nombre du véhicules pour servir les clients sans voiler les fenêtres de temps.

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

Le problème de tournées de véhicules stochastiques (SVRP) se caractérise par un aspect aléatoire associé à une partie des données, comme les demandes des clients, les temps de parcours, ou les temps de service qui ne sont pas connus avec certitude mais plutôt, caractérisés par une distribution de probabilités. Parfois, c'est l'ensemble des clients à visiter qui est sujet à une distribution de probabilités [5].

- **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 [6].

- **VRPMT (Vehicle Routing Problem with multiple Trips):**

Se rencontre principalement lorsque la flotte de véhicule est petite .dans ce cas, certains véhicules peuvent exécuter plusieurs tournées au cours d'une même période. En respectant une durée maximum .qui correspond à la durée de travail de chaque véhicule par période [7].

- **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 livrer. Le deuxième sous-ensemble contient m clients, où une quantité de produit attendue doit être déchargé .Il existe deux types de VRPB.

Le premier type est appelé livré-premier et collecte-deuxième VRPB, qui exige quelles 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 suivant illustre les deux types.

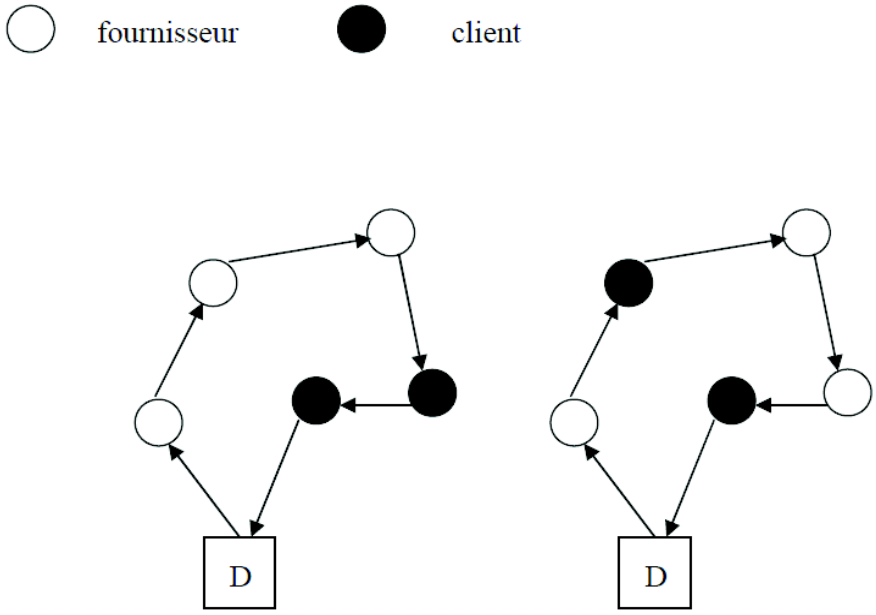

Figure 1 les deux types de VRPB

- **PVRP (PeriodicVehicleRoutingProblem) :**

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 [8].

- **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 [9].

- **TSP (Travelling Saleman Problem) :**

Le problème du voyageur de commerce est le plus simple, le plus connu et le plus étudié des problèmes de tournées. Il consiste à définir la tournée d'un vendeur visitant un ensemble de villes séparées par des distances données et retournant à la ville de départ (généralement nommée dépôt), une solution fixe l'ordre dans lequel les villes seront visitées. Le problème généralement modélisé sous forme d'un graphe dans lequel les nœuds représentent les villes à visiter, et les arcs représentent les routes liant les villes deux à deux. Ainsi, le vendeur doit visiter une et une seule fois chaque nœud, l'objectif de TSP est de minimiser la distance totale parcourue**.** C'est donc un cas particulier de VRP sans contrainte de capacité et avec un seul véhicule [10].

- **DARP (Dial A Ride Problem) ou VRPPD :**

Le problème de transport à la demande (**TAD**) ou Demand Responsive Transport (**DRT**) peut se présenter de la manière suivante : dans un espace donné (un réseau), on cherche à acheminer des demandes de clients à l'aide de moyens

de transport pouvant combiner différents types de véhicules. Chaque demande du client est définie par un lieu d'origine, un lieu de destination, un nombre de passagers et des contraintes temporelles [11].

1.3 Problème d'optimisation :

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 :

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 $s \in S$**.**

1.3.1 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 :

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

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*).

1.3.2 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.

1.3.3 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 ci par un ensemble d'inégalités, ou d'égalités de la forme :

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

1.4 Modélisation mathématique de VRP :

Soit G= (V, A) un graphe où V = {1,….n} est un ensemble de sommets avec le sommet 1 pris comme dépôt, et A= {(i, j) / i, $j \epsilon V$ et $i \neq j$} est l'ensemble des arcs. V'=V \ {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 [12], avec les notations suivantes :

a) 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*.

b) 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 :

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

Sous contraintes :

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

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

$$\sum_{i=1}^{n} x_{ip}^{k} - \sum_{j=1}^{n} x_{pj}^{k} = 0 \qquad k=1\ldots..m;\ p=1\ldots n \qquad (1.4)$$

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

$$\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^{k}\, x_{ij}^{k} \leq T_k \qquad k = 1 \ldots\ldots.. m \qquad (1.6)$$

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1 \qquad \text{k=1… m} \qquad (1.7)$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1 \qquad \text{k=1….m} \qquad (1.8)$$

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

Où *S* est proposé dans [12] de la façon suivante : Sachant que *Q* est un ensemble de sommets visités par un seul véhicule.

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

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

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), 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 :

Le VRP appartient à la grande famille des problèmes combinatoires NP-difficiles. Ceci signifie qu'il n'y a pas une méthode pouvant donner la solution optimale du problème dans un temps raisonnable. Dans ce cas, il est nécessaire d'utiliser des méthodes qui essayent de donner une solution pas forcément optimale mais proche de l'optimale dans un temps acceptable. Ainsi les méthodes de résolution consacrées aux VRP et à ses variantes sont principalement classées en deux catégories : les méthodes exactes et les méthodes approchées, ces derniers sont aussi diviser en deux groupes : les heuristiques et les métaheuristiques. Les figures suivant présentées les défirent méthodes de résolution.

.

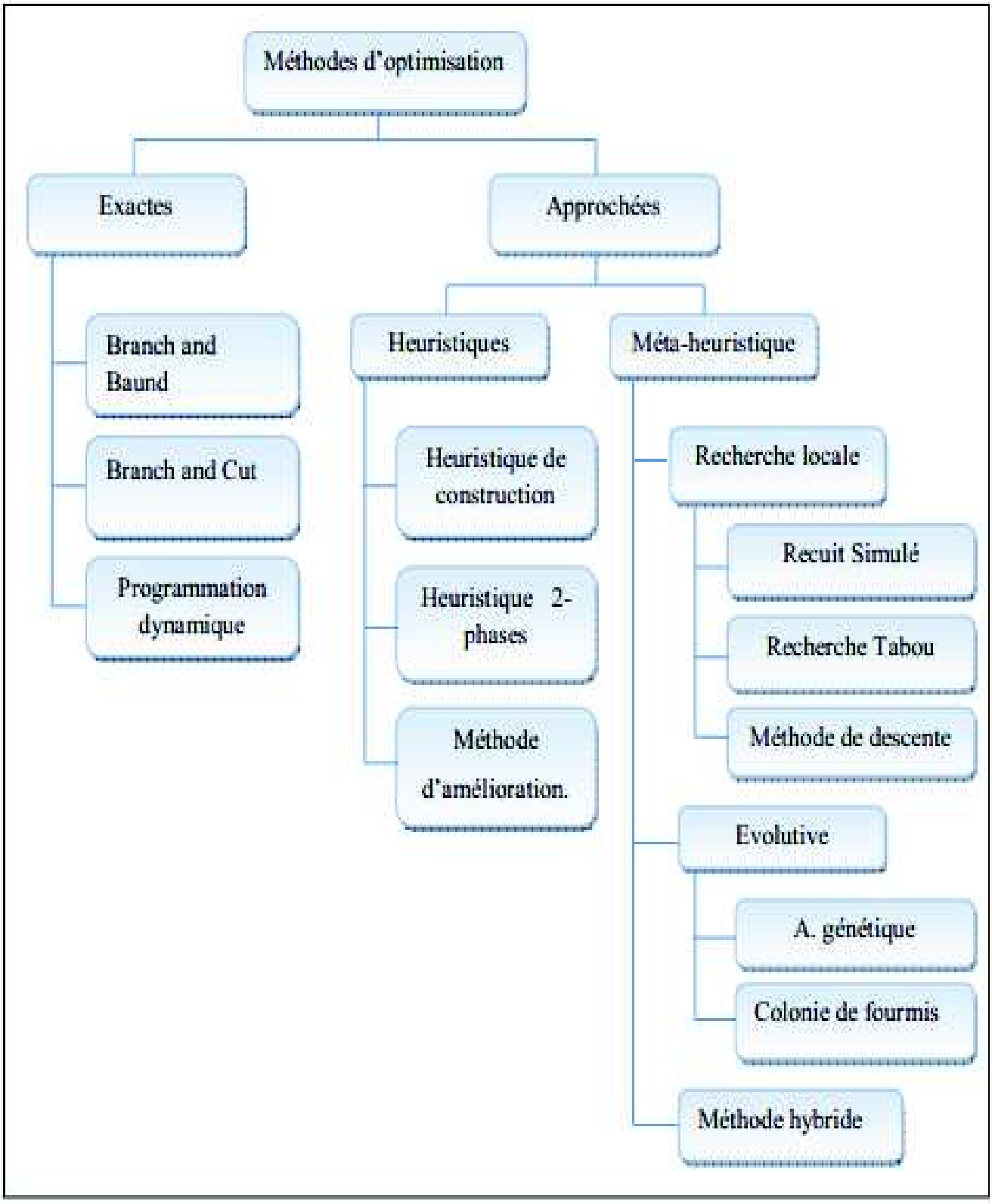

Figure 2 Les principales méthodes de résolution de VRP.

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 de taille allant jusqu'à 50 clients.

Parmi ces méthodes on peut citer :

- **Procédures par évaluation et séparation progressive (Branch and Bound) :**

Cette méthode appartient à la classe des méthodes de recherche arborescente. Elle a été proposée pour la première fois par land et doig.1960.

Principalement elle énumère de manière intelligente l'ensemble des solutions. Pour cela, elle décompose l'espace des solutions en sous-ensembles de plus en plus petits dont une bonne partie est éliminée à l'aide de bornes. Ce type d'énumération peut donc fournir une solution optimale en un temps réduit par rapport à une énumération complète. Cependant, pour les instances de grande taille des problèmes NP- difficiles, leur durée d'exécution est encore trop importante pour pouvoir être utilisable dans des applications réelles et il faut alors se tourner vers les approches heuristiques [14].

- **Programmation en nombre entier : 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 en train 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 [15].

- **La programmation dynamique :**

La programmation dynamique est un paradigme de conception qu'il est possible de voire comme une amélioration ou une adaptation de la méthode «diviser et régner », ce concept a été introduit par Bellman, dans les années 50, pour résoudre les problèmes d'optimisation.

La programmation dynamique repose sur le principe d'optimalité «toute politique optimale est composée de sous-politiques optimale », elle résout chaque sous-problèmes une seule fois et stocke les solutions de tous les sous-problèmes rencontrés dans une seule matrice, pour éviter d'avoir à les recalculer par la suite .enfin, elle cherche une relation de récurrence entre les sous-problèmes de sorte à la résoudre le problème principale.

Cette méthode a été utilisée pour des problèmes pour des problèmes de VRP de très petites tailles pour la résolution de problèmes allant de 10 à 25 clients.

1.5.2 Les méthodes approchées :

Les méthodes exactes rencontrent généralement des difficultés face à des applications de grande taille, et peuvent nécessiter un temps de calcul important même sur des instances de petite taille .C'est pour cette raison que l'utilisation des méthodes approchées s'est avérée d'une grande utilité .ces méthodes permettent d'obtenir en temps raisonnable des solutions de bonne qualité pour des problèmes de grande taille mais garantie d'optimalité .on peut les subdiviser en deux classe :les heuristiques et les métaheuristiques.

1.5.2.1 Les heuristiques :

Comme pour tous les problèmes NP-difficile, la recherche d'heuristiques performantes est l'une des principes préoccupation des spécialistes du domaine. Le principe d'une méthode heuristique est de trouver, en temps raisonnable une solution de bonne qualité. Dans ce qui suit, nous décrirons les trois grandes familles d'heuristiques développées pour les problèmes de tournées : méthodes constructives, méthodes de recherche locales et méthodes en deux phases.

- **Méthode constructives :**

L'idée de base d'une heuristique constructive est de réduire la taille du problème à chaque étape pour limiter progressivement l'ensemble des solutions réalisables, Les heuristiques constructives partent d'une solution vide et construisent étape par étape une solution finale *s* de *S*. les choix effectués à chaque étape *k* sont faits selon certaines règles [16]. Ces règles dépendent donc du problème considère. Bien évidente, à chaque étape k, l'ensemble des solutions réalisables S est donc réduit en un ensemble $s^k \subseteq S$. Ces heuristiques sont caractérisées par la facilité de mise en œuvre et la rapidité d'exécution. Par contre, la faible qualité des solutions trouvées est en général leur grand défaut [17]. Parmi ces méthodes, nous pourrons citer la méthode des gains et la méthode d'insertion.

- **Les méthodes d'amélioration :**

Ces méthodes peuvent être utilisées dans la génération d'une solution initiale pour les méta-heuristiques, comme elles font appel à des méthodes de recherche locale, qui améliorent progressivement une solution initiale déjà obtenue. À chaque itération d'une recherche locale, il y aura une exploration d'un voisinage de la solution actuelle pour trouver une meilleure solution. Dans ce voisinage des solutions, nous cherchons à trouver en général la meilleure solution et pour chaque cas nous avons un critère d'arrêt spécial. Ces méthodes ont été développées pour aider la recherche à échapper aux minima locaux [17].

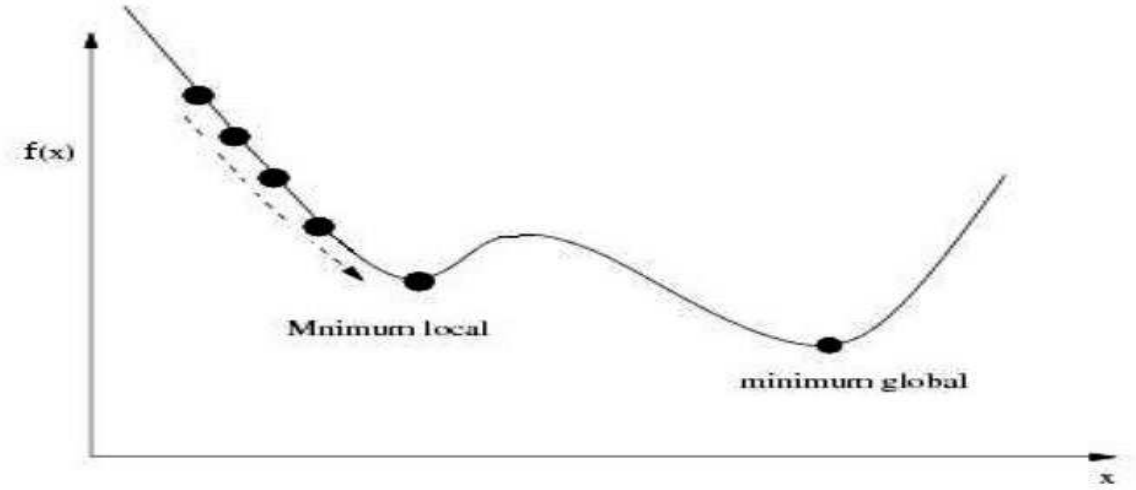

Figure 3 le principe des méthodes de voisinage

- **Méthodes en deux phases :**

Les méthodes en deux phases sont basées sur une décomposition du problème en une partition en sous-groupe de l'ensemble des clients et la détermination de la tournée associée à chaque sous-groupe. Deux classes d'algorithmes peuvent être considérées selon l'ordre dans lequel les deux phases sont effectuées : méthode de <<route-first et cluster-second>> et méthode de <<cluster-first et route-second>>.

a) L'heuristique « groupe en premier, route en second » :

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

b) L'heuristique « route en premier, groupe en second » :

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

1.5.2.2 Les métaheuristiques :

Les métaheuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile [15]. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. 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.

- **Recuit simulé :**

  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,….

- **Recherche tabou :**

Comme le recuit simulé, il s'agit au moins dans sa version de base, d'une variante de l'algorithme de recherche par voisinage. Tant que l'on ne se trouve pas dans un optimum local, la recherche tabou se comporte comme toute méthode de voisinage et améliore à chaque étape la valeur objective. Lorsque l'on atteint par contre un minimum local, on se permit de passer au moins mauvais des voisins.

L'inconvénient de cette dernière démarche est que parfois, une itération ne suffit pas pour s'en sortir d'un minimum local et un déplacement peut provoquer un déplacement inverse à une étape ultérieure, ce qui provoquera un cycle autour du minimum local. C'est pour cette raison

que RT garde en mémoire les dernières solutions visitées pour éviter d'y revenir, c'est ce qu'on appelle « liste tabou ».

- **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, déposées sur les trajets parcourus. Cela peut paraître surprenant au premier abord mais un chemin plus court reçoit plus de phéromones qu'un chemin plus long.

Cette métaheuristique a été introduite pour la première fois par Colorni ,et a été appliquée sur le problème du voyageur de commerce. [18] ont appliqué cette métaheuristique sur les VRP.

- **Les algorithmes génétiques**

L'une des métaheuristiques les plus utilisées pour la résolution du problème de routage de véhicules est les algorithmes génétiques. Il s'agit d'une méthode bio-inspirée introduite par Holland 1975 dans le cadre d'une analogie avec la sélection naturelle des espèces. Elle a été formalisée ensuite par Goldberg 1989 pour être appliquée à la résolution de problèmes d'optimisation.

Un algorithme génétique entretient une population de solutions au problème à optimiser. La qualité de ces solutions (appelées individus) est donnée par une fonction d'évaluation (dite Fitness). L'algorithme applique les opérateurs génétiques de croisement et de mutation à la population en vue d'en générer une autre. À partir de deux individus parents, l'opérateur de croisement génère un ou deux individus enfants héritant chacun une partie de la structure des parents. L'opérateur de mutation agit quant à lui sur un seul individu en modifiant une partie de sa structure.

1.6 Conclusion :

Dans ce chapitre nous avons présenté le problème du VRP. Suite par ses variantes et sa formulation mathématique, nous avons énuméré les différentes méthodes de résolution qui ont été utilisées pour le résoudre. Le chapitre suivant va détailler la méthode de résolution qui nous allons utiliser.

Chapitre 02 :

Algorithmes génétiques

2.1 Introduction :

Les algorithmes génétiques appartiennent à une catégorie d'algorithmes appelés Meta heuristiques, dont l'objectif est de repérer une solution approche à un problème d'optimisation en un temps raisonnable, lorsqu'il n'existe pas de méthode exacte pour le résoudre ou que la durée de la phase de calcul est trop longue à l´échelle de la vie humaine. Les algorithmes génétiques s'inspirent de la théorie de la sélection naturelle d´enveloppée au XIXe siècle par le biologiste Charles Darwin.

L'utilisation des algorithmes génétiques AG, dans la résolution de problèmes, est à l'origine le fruit des recherches de John Holland et de ses collègues et élèves de l'Université du Michigan qui ont, dès 1960, travaillé sur ce sujet. La nouveauté introduite par ce groupe de chercheurs a été la prise en compte de l'opérateur d'enjambement en complément des mutations. Et c'est cet opérateur qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes contenus dans les différents individus de la population. Le premier aboutissement de ces recherches a été la publication en 1975 d'Adaptation *in Natural and Artificial System*.

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.

2.2 La sélection naturelle de Darwin :

Avant de détailler le fonctionnement d'un algorithme génétique, il serait utile de rappeler quelques principes de base de la sélection naturelle, nécessaires à la compréhension de ce chapitre.

La théorie de la sélection naturelle permet de comprendre l'action de l'environnement sur l'évolution des populations. Dans une population d'individus, un caractère présente des variations. Certains individus portent des variations qui leur permettent d'être mieux adapté à leur environnement. Ils ont donc plus de chances de survivre. Ils ont donc une meilleure probabilité de se reproduire. Leur descendance est donc plus nombreuse, et portera cette variation si elle est héréditaire. La conséquence logique est que cette variation héréditaire verra sa fréquence augmenter dans la génération suivante.

Et ainsi de suite... On parle d'avantage sélectif. Les variations qui présentent un désavantage sélectif, voient leur fréquence diminuer au fil des générations, et en général finissent par disparaître [18].

2.3 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. Les algorithmes génétiques sont les plus populaires des algorithmes évolutionnaires .ces algorithmes s'inspirent des principes de la génétique (la survie des individus des mieux adaptés à un environnement, la recombinaison génétique, quelquefois l'apparition d'une mutation)

Les algorithmes génétiques peuvent être particulièrement utiles dans les domaines suivants :

- Optimisation : optimisation de fonctions, planification, etc...
- Apprentissage : classification, prédiction, robotique, etc...
- Programmation automatique : automates cellulaires, etc...
- Etude du vivant, du monde réel : marchés économiques, comportements sociaux, systèmes immunitaires, etc...[19].

2.4 Fonctionnement des algorithmes génétiques :

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection. Le principe d'un algorithme génétique est : A partir d'un ensemble des solutions initiales, ou population de N individus, elle consiste à faire évoluer cette population en utilisant des opérateurs de sélection, de croisement et de mutation. A chaque itération de l'algorithme, une nouvelle population de solutions ou d'individus est générée. Tout d'abord, un ensemble d'individus est sélectionné pour générer la population suivante. Ces individus sont ensuite croisés pour créer de nouveaux individus et compléter la nouvelle population. Certains de ces nouveaux individus peuvent subir une mutation.

La structure générale de l'algorithme génétique est comme la suite :

Algorithme : Structure générale de l'algorithme génétique

1) Initialiser la population initiale P.
2) Evaluer P.
3) Tant Que (Pas Convergence) faire :
a) P ' = Sélection des Parents dans P
b) P ' = Appliquer Opérateur de Croisement sur P '
c) P ' = Appliquer Opérateur de Mutation sur P '
d) P = Remplacer les Anciens de P par leurs Descendants de P '
e) Evaluer P
Fin Tant Que

•Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

a. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. 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.

b. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

c. Une fonction à optimiser. Celle-ci retourne une valeur de Â $^{+}$ appelée *fitness* ou fonction d'évaluation de l'individu.

d. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. 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 d'états.

e. Des paramètres de dimensionnement : la taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation. [18]

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure 2.1 :

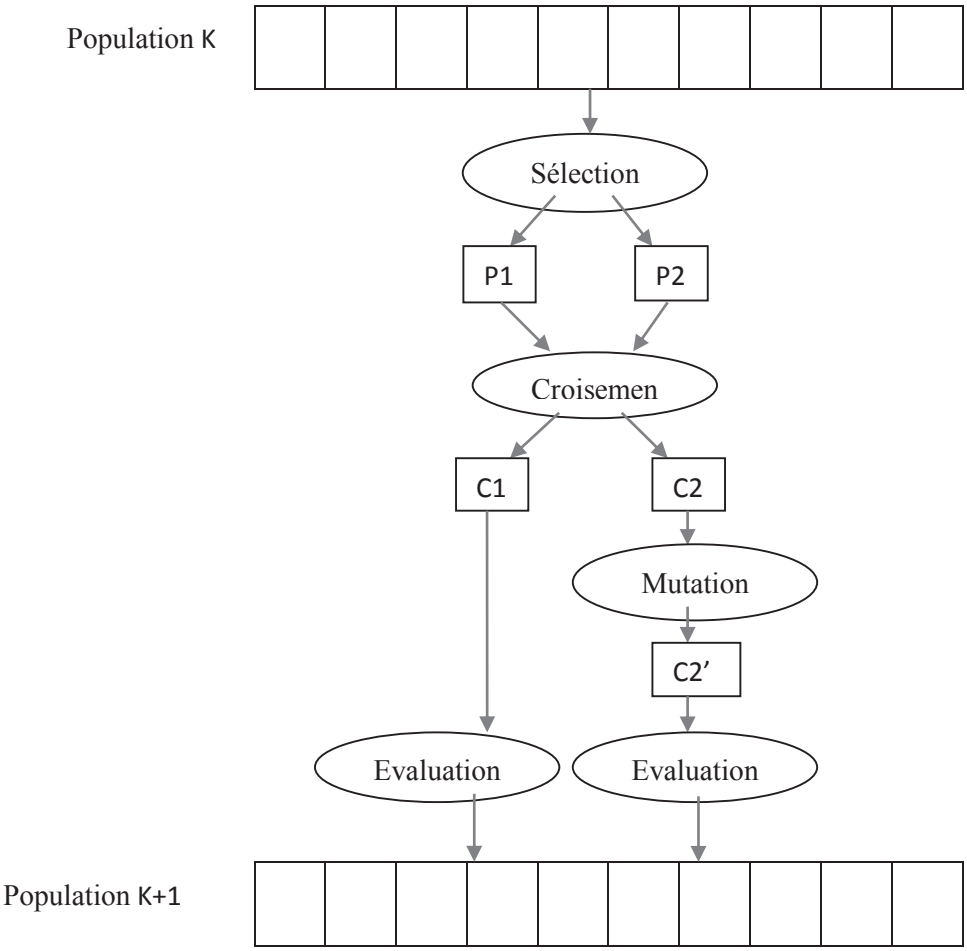

Figure 4 : principes généraux des algorithmes génétiques

2.4.1 Le codage :

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).

Un chromosome est une suite des gènes, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position : son locus sur le chromosome en question.

Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus [19].

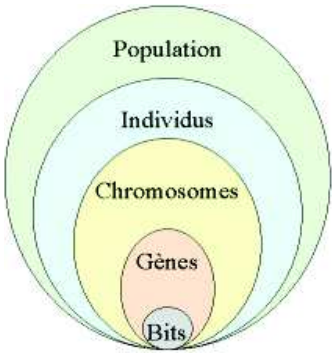

Figure 5 :les cinq niveaux d'organisation d'un algorithme génétique

Il y a trois principaux types de codage utilisables, le choix entre eux dépond du problème ciblé:

a. le codage binaire :

C'est le plus utilisé. Chaque gène dispose du même alphabet binaire {0, 1}, un gène est alors représenté par un entier long (32 bits), les chromosomes qui sont des suites de gènes sont représentés par des tableaux de gènes et les individus de notre espace de recherche sont représentés par des tableaux de chromosomes.

Ce cas peut être généralisé à tout alphabet allélique n-aire permettant un codage plus intuitif, par exemple pour le problème du voyageur de commerce on peut préférer utiliser l'alphabet allélique {c1, c2, c3, ..., cn} où ci représente la ville de numéro i.

b. le codage réel :

Cela peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle.

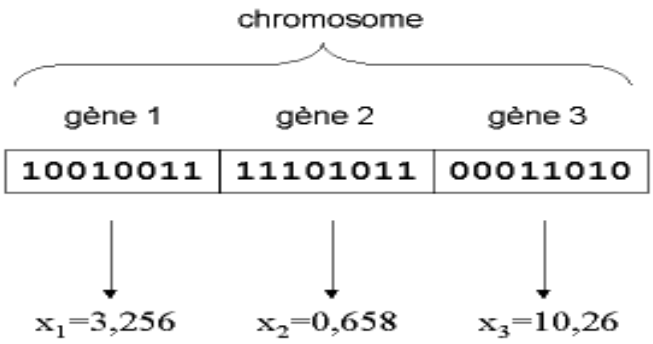

Figure 6 : illustration schématique du codage des variables réelles

2.4.2 Génération aléatoire de la population initiale :

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état en veillant à ce que les individus produits respectent les contraintes. Si par contre, des informations à priori sur le problème sont disponibles, il parait bien évidemment naturel de générer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités. Il est clair qu'il vaut mieux, lorsque c'est possible ne générer que des éléments de population respectant les contraintes [18].

2.4.3 L'opérateur de sélection :

Cet opérateur est chargé de définir quels seront les individus de P qui vont être dupliqués dans la nouvelle population P' et vont servir de parents (application de l'opérateur de croisement).

Soit *n* le nombre d'individus de P, on doit en sélectionner *n/2* (l'opérateur de croisement nous permet de repasser à *n* individus).

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.
On trouve plusieurs types de méthodes de sélection différentes :

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 en regarde quel est le segment sélectionné.

b. Sélection par rang :

La sélection par roulette présente des inconvénients lorsque la valeur d'évaluation des individus varie énormément. En effet, on risquerait d'atteindre une situation de stagnation de l'évolution. Imaginons le cas où 90% de la roulette est allouée à l'individu qui a la meilleure évaluation, alors les autres individus auront une probabilité très faible d'être sélectionnés.

La sélection par rang trie d'abord la population par évaluation. Ensuite, chaque individu se voit associé un rang en fonction de sa position. Ainsi le plus mauvais individu aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur individu qui aura le rang N, pour une population de N individus.

La sélection par rang d'un individu est identique à la sélection 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 individus 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 individus ne diffèrent pas énormément des plus mauvais [20].

c. La sélection par tournois :

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants, le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de P, et on le fait "combattre". Celui qui a la fitness la plus élevée l'emporte avec une probabilité p comprise entre 0.5 et 1. On répète ce processus n fois de manière a obtenir les n individus de P' qui serviront de parents.

La variance de cette méthode est élevée et le fait d'augmenter ou de diminuer la valeur de *p* permet respectivement de diminuer ou d'augmenter la pression de la sélection.

2.4.4 L'opérateur de croisement ou « Crossover » :

Le croisement utilisé par les algorithmes génétiques est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

Son rôle fondamental est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population.

Cet opérateur est appliqué après l'étape de sélection sur la population P; on se retrouve donc avec une population P' de *n/2* individus et on doit doubler ce nombre pour que notre nouvelle génération soit complète.

On va donc créer de manière aléatoire *n/4* couples et on les fait se "reproduire". Les chromosomes (ensembles de paramètres) des parents sont alors copiés et recombinés de façon à former deux descendants possédant des caractéristiques issues des deux parents [19].

Plusieurs variantes de croisement existent selon le type de codage utilisé :

a. Croisement binaire

Le croisement binaire se fait par Un, deux, voire jusqu'à *lg - 1* (où lg est la longueur du chromosome) points de croisements sont tirés au hasard, chaque chromosome se retrouve donc séparé en "segments". Puis chaque segment du parent 1 est échangé avec son "homologue" du parent 2 selon une probabilité de croisement p_c .De ce processus résulte 2 fils pour chaque couple et notre population P' contient donc bien maintenant *n* individus.

On peut noter que le nombre de points de croisements ainsi que la probabilité de croisement p_c permettent d'introduire plus ou moins de diversité.

En effet, plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus il y aura d'échange de segments, donc d'échange de paramètres, d'information, et plus le nombre de points de croisements sera petit et plus la probabilité de croisement sera faible, moins le croisement apportera de diversité.

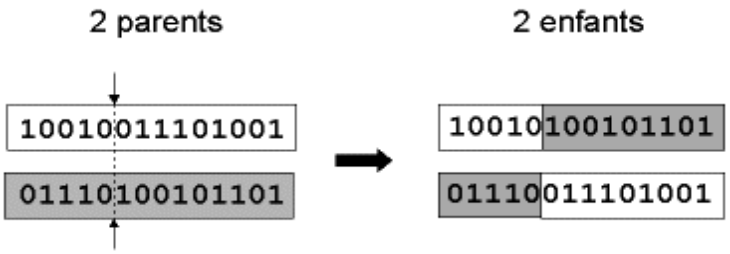

Fig. 2.4 croisement avec un point de coupure

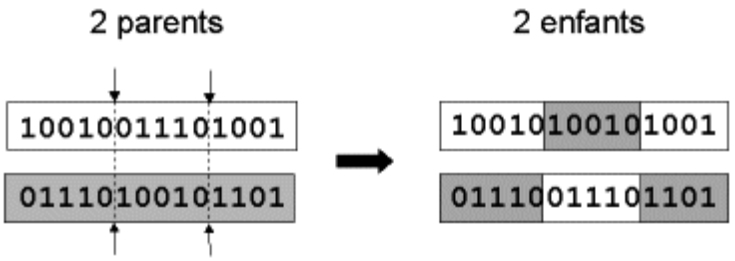

Figure 7 :croisement avec 2 points de coupure

On peut citer aussi une autre méthode très utilisée dans le cas des problèmes modélisés par un codage binaire, il s'agit du ***croisement uniforme***. La mise en œuvre de ce procédé est fort simple, elle consiste à définir de manière aléatoire 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 0, le fils héritera le gène s'y trouvant du parent n° 1, s'il présente un 1 il en héritera du parent n° 2. La création du fils n° 2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils n° 1 devra recevoir celui-ci du parent n° 1 alors le fils n° 2 le recevra du parent n°2, et si le fils n° 1 le reçoit du parent n° 2 alors le fils 2 le recevra du parent n° 1.

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**

Figure 8 :croisement uniforme

b. Croisement réel :

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

Dans PMX, qui a été proposé par (Golberg et lingle, 1985), deux parents sont considérés et deux positions de découpage sont sélectionnées uniformément au hasard. Les deux points de croisement (P1 et P2) définissent une section compatible qui est utilisée pour effectuer des opérations d'échange de gènes, position par position

			P1			P2				
Parent 1	9	8	4	5	6	7	1	3	2	10
			P1			P2				
Parent 2	8	7	1	2	3	10	9	5	4	6

Figure 9 :Illustration du croisement PMX, étape 1

PMX procède par des échanges de position. Au départ, les enfants 1et 2 sont des copies conformes des parents 1et 2, respectivement, puis les gènes se trouvant entre les points de coupure sont échangés deux à deux, position par position.

			P1			P2				
Enfant 1	9	8	4	2	3	10	1	3	2	10
			P1			P2				
Enfant 2	8	7	1	5	6	7	9	5	4	6

Figure 10 : Illustration du croisement PMX, étape 2

Ensuite, pour les gènes placés avant et après les points de coupure, il reste à traiter le cas des gènes répétés (en double). Dans l'enfant 1, c'est le cas pour les gènes qui étaient placés dans la zone délimitée par les points de coupure dans le parent 2, mais en dehors de cette zone dans le parent 1. Dans ce cas, le gène placé en dehors de la zone prend la valeur du gène parent 1 placé à la même position à l'intérieure de la zone, et par conséquence à la même position dans l'enfant 2. Par exemple, dans la figure x, le gène 3 apparait à deux endroits dans l'enfant 1 : une première fois en cinquième position (entre les points de coupure) et une seconde fois en huitième position, après le second point de coupure. La valeur 6, placée en cinquième position dans le parent 1 et le second enfant (entre les points de coupure P1 et P2) est donc utilisée pour remplacer en quatrième et sixième position dans l'enfant 2 sont utilisés pour remplacer les gènes 2 et 10 aux neuvième et dixième positions de l'enfant 1. La même procédure est répétée jusqu'à éliminer tous les gènes apparaissant deux fois dans l'enfant 2. Les enfants finalement obtenus sont donc :

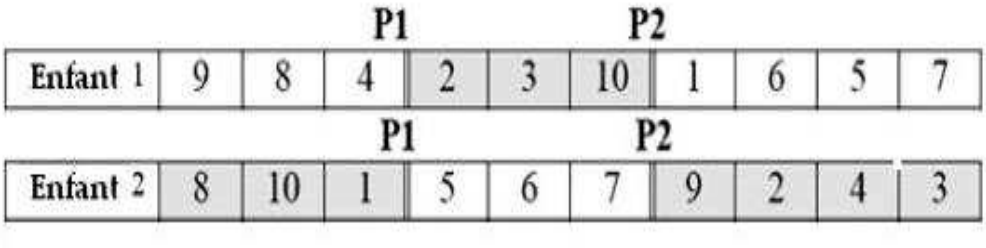

Figure 11: Illustration du croisement PMX, étape 3

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

L'opérateur CX est un opérateur de croisement . Il préserve l'information contenue dans les parents. Pour expliquer ce type de croisement, considérons l'exemple représenté à la figure ().

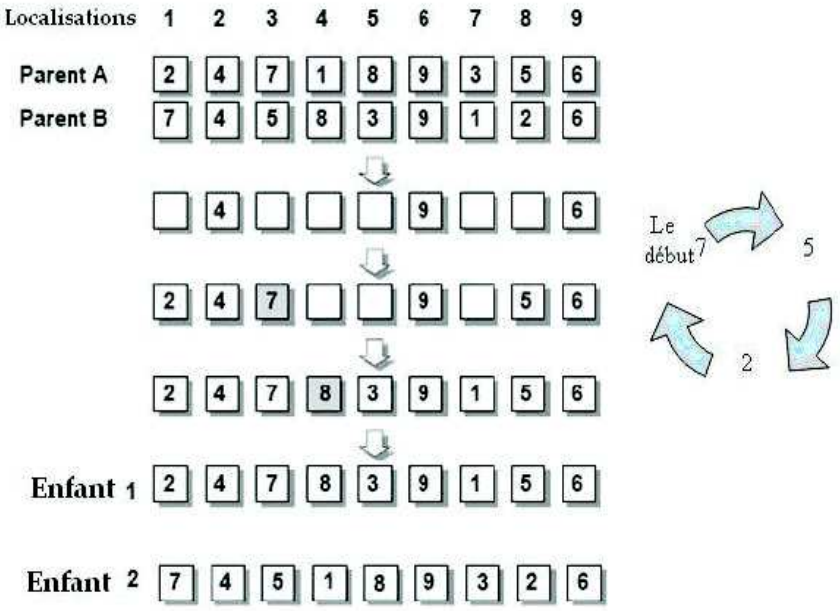

Figure 12: Illustration du croisement CX

D'abord, tous les gènes qui sont assignés à la même place dans les deux parents sont hérités par les enfants. Ce sont les gènes 4,9 et 6. Puis, commençant par une position choisie aléatoirement (dans cet exemple, localisation 3), un gène est choisi aléatoirement parmi un des parents et est assigné à la même place dans l'enfant. Dans l'exemple, pour construire l'enfant 1, c'est le gène 7ndu parent A qui est copié dans l'enfant 1. Puis l'algorithme cherche quel gène est en face de 7 (ou dans la même position) dans le parent B : c'est le gens 5.

Ensuite, le gène 5 est placé dans l'enfant mais dans la même position que celle qu'il occupe dans le parent A (position8). De la même manière, le gène 2 du parent A est placé dans l'enfant 1 parce que le gène 2 du parent B est en face du gène 5 du parent A. ceci ferme le cycle car le gène suivant est le gène 7, qui est déjà inclus dans l'enfant 1. Finalement, les gènes encore à compléter dans l'enfant 1 sont trouvés en examinant les positions correspondantes du parent B. c'est-à-dire, les gènes 8 ,3 et 1 sont placés dans l'enfant 1 aux positions 4,5 et 7 respectivement. Enfin, cette procédure est répétée en inversant le rôle des parents pour obtenir l'enfant 2 [21].

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 El 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 El 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 El et E2 tel qu'illustré à la Figure 2.11 partie (c).

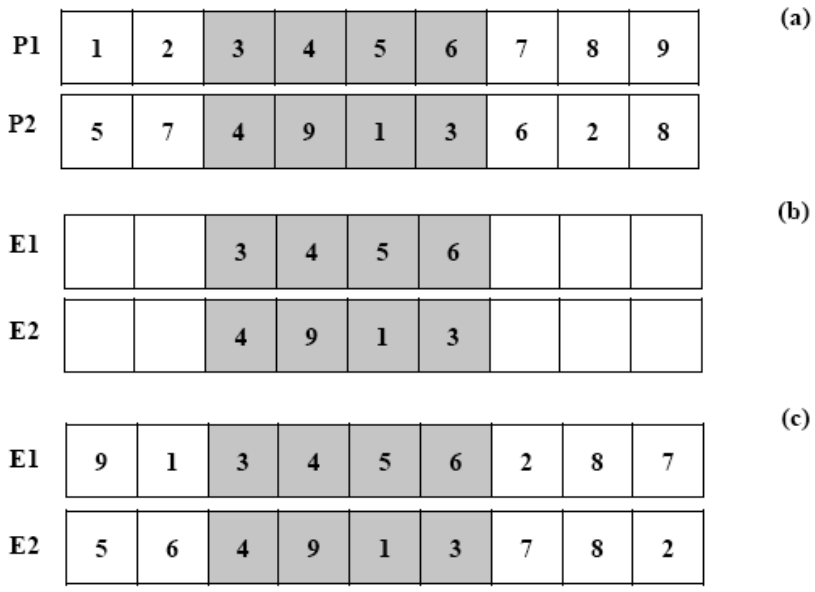

Figure 13: Opérateur de Croisement OX

2.4.5 Mutation :

Apres le croisement, un opérateur de mutation est appliqué au sein de la population des enfants avec une probabilité très faible, nommée taux de mutation ou Pm. Il permet de maintenir la diversité de la population des enfants. Les petits taux de mutation sont recommandés car un grand taux peut occasionner une destruction de l'information utile contenue dans les solutions et être considéré probablement comme une recherche aléatoire [21].

Les types de mutation les plus communément sont :

a. insertion :

L'operateur d'insertion choisit un gène aléatoirement et l'insère dans une autre position du chromosome.

b. Echange :

L'operateur d'échange consiste à prendre au hasard deux génes du cromosome et à les permuter.

c. Inversion :

L'insertion inverse l'ordre des genes entre deux points de coupure choisis aléatoirement

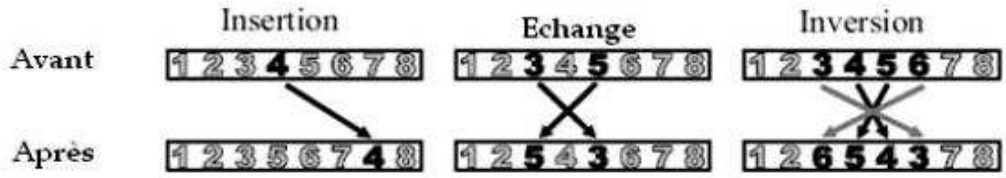

Figure 14: opérateurs d'insertion, échange et inversion.

2.4.6 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].

1.5 Conclusion :

Les algorithmes génétiques s'inspirent de la théorie de la sélection naturelle de Darwin. Par analogie avec le monde biologique, l'algorithme génétique fait évoluer un échantillon d'individus à l'aide de trois opérateurs principaux : sélection, croisement, mutation.

Ils admettent une grande liberté dans la configuration des paramètres et dans l'implémentation des différents traitements. Pour atteindre des performances optimales, adapter les paramètres de l'algorithme et introduire des méthodes spécifiques au problème traité sont

indispensables. On peut également réaliser des algorithmes hybrides mêlant différentes méthodes.

Ils synthétisent des solutions nouvelles et originales à des problèmes connus sans idées préconçues, si ce n'est les paramètres et l'espace de recherche qu'on leur impose. Ils permettent donc d'insuffler un peu de créativité dans l'ordinateur, cette machine qui en est si cruellement dépourvue !

Chapitre03 :

Conception et architecture de l’application

3.1 Introduction :

Dans ce chapitre 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. L'approche de résolution développée se base sur les algorithmes génétiques développés détaillé précédemment.

Dans ce qui suit, on va décrire les techniques utilisées dans chaque étape de l'algorithme génétique.

Nous proposons des heuristiques pour réaliser les différant étapes d'algorithme génétique (population initial, sélection, croisement et mutation).

3.2 Description du problème :

Le problème qu'on a traité dans ce travail est le transport des patients hémodialysés, 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.

Le problème peut être représenté sous forme d'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_{ij}, 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.

Opérateur de transport : un opérateur possède un nombre connu de véhicules et un nombre connu des malades, qu'il peut augmenter à chaque inscription des nouveaux malades.

- **Les véhicules :** chaque véhicule possède une vitesse et une capacité (nombre de places).
- **Les malades :** chaque malade possède des informations personnelles (nom, prénom, âge, adresse) et les séances de dialyse.

- **les séances :** les séances pour chaque malade peut être un, deux ou trois au maximum par semaine, ces séances doivent être faites dans des jours non consécutifs.
  Une solution faisable doit satisfaire les contraintes suivantes :
- **Pairing :** les sites de départ P_K et d'arrivé D_Kd'une demande k doivent être visités dans une même tournée.
- **Précédence :** le site de dépareP_K doit être visitée le site d'arrivé D_K.
- **Capacité de véhicule :** le nombre des malades par un véhicule ne doit pas dépasser la capacité de ce dernier (3 malades).
- **Durée maximal des tournées :** la durée d'une tournée ne doit pas dépasser T car le véhicule doit retourner au dépôt pour ravitailler en carburant.
- **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.
- **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'arrivé.

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.

3.3 Méthode de résolution :

Pour résoudre ce problème, nous allons utiliser l'algorithme génétique. La section suivante décrit ses caractéristiques principales qui sont : le codage de chromosome, l'évaluation de chromosome, la population initiale, la procédure de sélection, l'opérateur de croisement, l'opérateur de mutation, la reconstruction de la population, et la condition d'arrêt.

3.3.1 Le codage de Chromosome :

Pour pouvoir faire le codage on a numéroté puis regroupé les requêtes des malades dans une table qu'on l'appel table des requêtes, elle contient 2*M requêtes, où M est le nombre des séances de dialyse programmées pour une journée, donc pour chaque séance on a associé deux requêtes une pour l'aller vers l'hôpital et l'autre pour le retour vers la maison du malade.

Pour le codage des Chromosomes, on utilise un codage réel. Donc Un chromosome est une séquence S des nœuds K, où K représenter la requête numéro K, 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. La figure suivant représente un exemple de chromosome satisfaisant quatre demandes.

2	1	3	4

Codage de chromosome

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

La fonction de *découpage* :

La fonction de *découpage* permet de découper un chromosome (voire la figure 16) en tournées réalisables, avec l'ajout des nœuds de déchargement pour chaque requête (chaque requête doit être représentée par deux nœuds un pour le chargement représenté par le numéro de la requête K et l'autre pour le déchargement par –K). L'objectif de cette fonction de découpage est de faire distribuer les requêtes sur les véhicules de façon à minimiser la durée totale en respectant les contraintes.

Soit *S* une séquence de *n* nœuds. L'objectif de *découpage* est de trouver un plus court chemin dans un graphe auxiliaire $H = (X, A, Z)$. Une telle tournée est réalisable si toutes les contraintes citées précédemment sont respectées.

La fonction de *découpage* décompose le chromosome en tournées, par l'ajout du 0 qui représente la base des véhicules après chaque tournée.

Algorithme 3.1 fonction *découpage*

S=un chromosome non découpé ;
Comp=100 ;// initialiser le variable comp à 100
Capacité=3 ; // initialiser la capacité de véhicule à 3
Tant que (le chromosome S n'est pas terminé)
  Tant que (comp satisfaite pour un autre nœud & la capacité de véhicule est satisfaite)
    Ajouter le nœud ;
    Comp=comp-distance entre anciens nœud et nouveau nœud ;
    Capacité= capacité- 1 ;
  Fin tant que
  Ajouter les nœuds de déchargement le plus proche d'abord ;
  Ajouter le nœud 0 ;
  Comp=100 ;
  Capacité=3 ;

Fin tant que

Voici un exemple qui illustre le fonctionnement de la procédure de découpage, la figure 15 représente le chromosome avant le découpage et la figure 16 représente le résultat après le découpage.

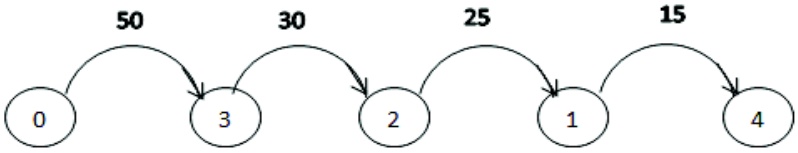

Figure 15: Le chromosome S sans découpage

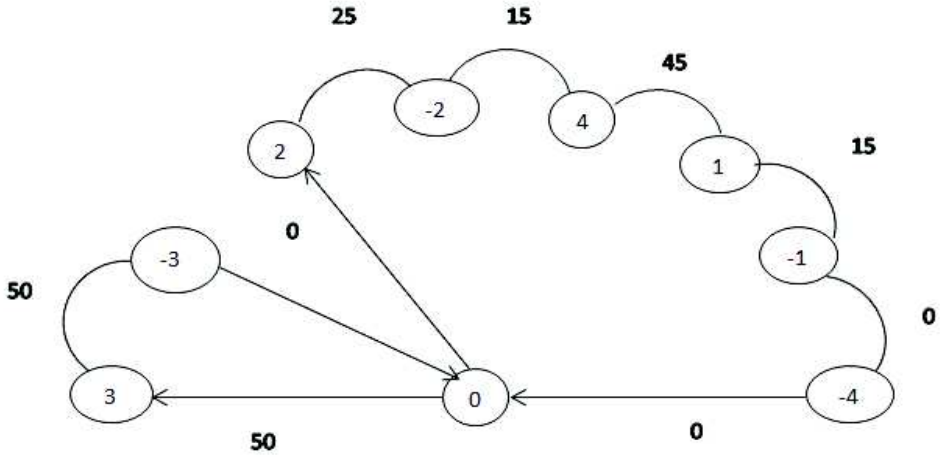

Figure 16: Le chromosome S après découpage

3.3.2 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 précédence et la satisfaction de toutes les demandes, leurs fenêtres de temps, et le temps de transport. Le deuxième problème 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.

Ensuite, on va appliquer l'algorithme de *découpage*décrit ci-dessus sur cet individu pour créer les tournées de façon optimale avec les nœuds de déchargement. 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.

Algorithme 3.2 : L'heuristique d'insertion aléatoire

```
Début
    Pour (i=1 à la taille de la population) Faire // chargement de la population initiale
        Pour (j=1 à la taille de l'individu) Faire // génération aléatoire d'un individu
            Tant que (les contraintes ne sont pas vérifiées) Faire
                Générer un site aléatoirement ;
                Tester la contrainte de la précédence ;
                Tester la répétition ;
                Si (les contraintes sont vérifiées) alors
                    Ajouter le site à l'individu ;
                Fin Si
            Fin tant que
        Fin pour
        Appliquer la procédure Découpage sur l'individu généré
        Si (l'individu généré n'existe pas dans la population) alors
            Ajouter l'individu à la population;
        Sinon
            Régénérer l'individu ;
        Fin Si
    Fin pour
Fin
```

3.3.3 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 sélectionner1/4 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 3.3.

Algorithme 3.3 : Le principe de la sélection

```
Début
    Sélectionner les 1/4 meilleurs individus de la population ;
    // Sélectionner un quart des individus parmi le reste des individus par la technique du tournoi
    Tant que (le nombre des individus sélectionnés < taille de population /4) Faire
        Choisir l'individu le plus moins longueur des deux individus pour le croisement
    Fin tant que
Fin
```

3.3.4 L'opérateur de croisement :

- **Le principe :**

Pour le croisement, nous avons utilisé un croisement à deux points de coupure. Après avoir sélectionné deux parents P1 et P2, cet opérateur de croisement consiste à sélectionner aléatoirement deux points de coupure *i* et *j* entre les séances, le premier point i peut prendre ses valeurs entre 1 et 6, et le deuxième point entre 2 et 7 (on a six séance de travail par jour), à condition que le rang de *i* est inférieur au rang de *j*. La séquence des gènes des séances qui se trouve entre i et j du parent P1 est tout d'abord copiée dans le fils C1. Puis P2 est parcouru de façon circulaire à partir du premier gène de la séance *j* +1 pour compléter C1.Un deuxième fils C2 est construit de façon similaire en échangeant les rôles de P1 et P2 en utilisant les mêmes points de coupure.

Mais pour connaitre les limites des séances, if faut faire une fonction pour annuler le découpage des parents.

Algorithme 3.4 : Le principe de fonction annul_découpage

```
Début
    Pour (i=1 à la taille d'individu) faire
        Si (le numéro du nœud P(i) est négatif) alors
            Supprimer  le nœud P(i) de l'individu ;
        Fin si
        Si (le nœud P(i) est un 0) alors
            Si( P(i-1) et P(i+1) de la même séance) alors
                Supprimer  le nœud P(i) de l'individu ;
            Fin si
        Fin si
    Fin pour
Fin
```

Après l'application de la fonction annul_découpage sur les parents, on peut appliquer le croisement.

Algorithme 3.5 : Le principe de croisement

```
Début
    Appliquer la fonction annul_découpage a 2 individus de sélection ;
    Générer aléatoirement deux points de coupure pour l'individu 1
    Calculer la distance entre ces deux points
    Faire le croisement
    Appliquer la fonction découpage a  les 2 nouveaux individus
Fin
```

Exemple :

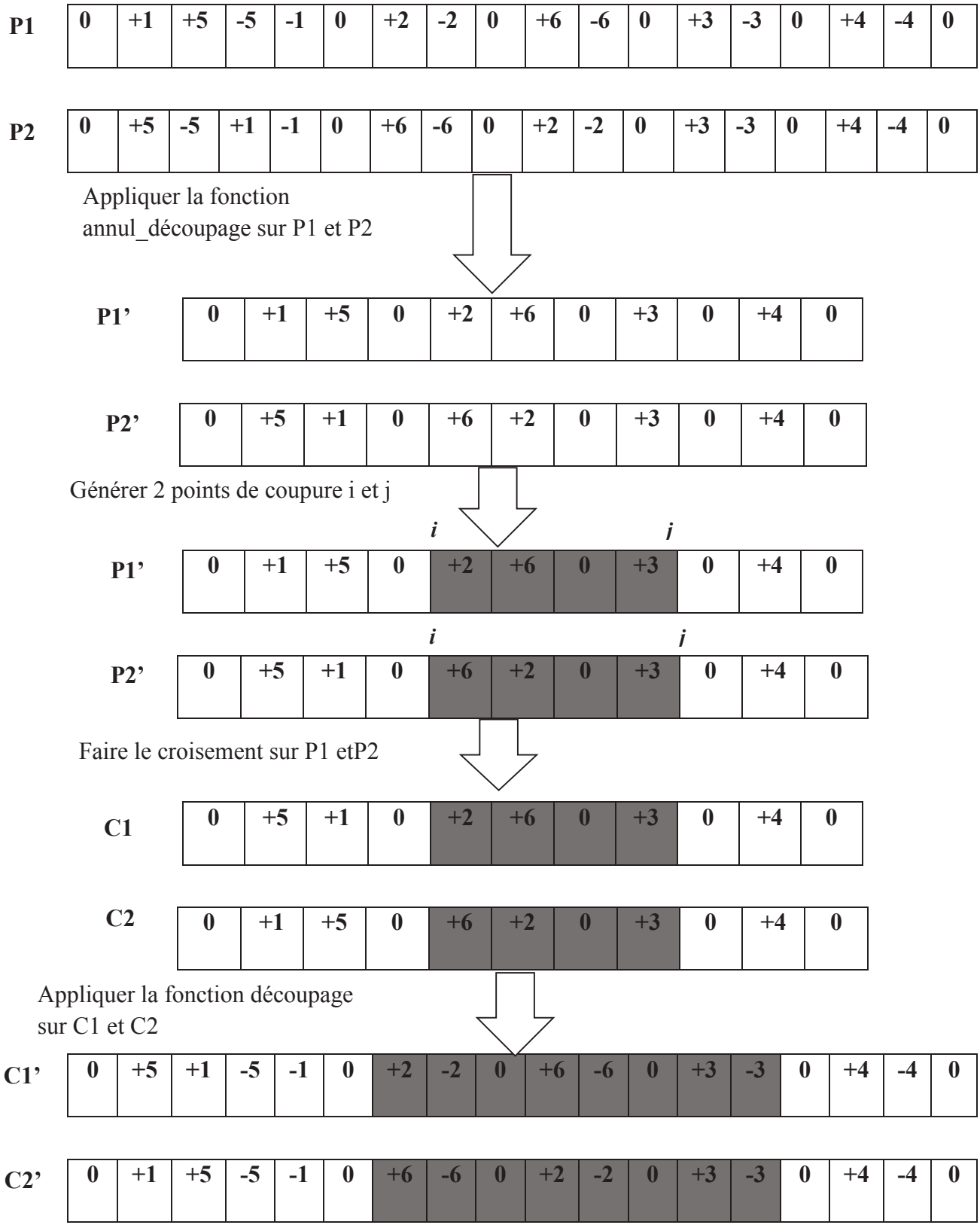

Figure 17: exemple complet de croisement

Après le croisement on applique la fonction *découpage* sur les fils, c'est l'heuristique qui permet de réinsérer les nœuds de déchargement dans les bonnes positions pour construire une solution réalisable.

3.3.5 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 inversion » décrit dans le chapitre précèdent qui sera appliqué sur un individu qui contient seulement les nœuds de chargement (on élimine les nœuds de déchargement).

L'individu résultant est rajouté à la population après une étape de réinsertion des nœuds de déchargement pour obtenir une solution réalisable par appel à la fonction découpage.

Exemple :

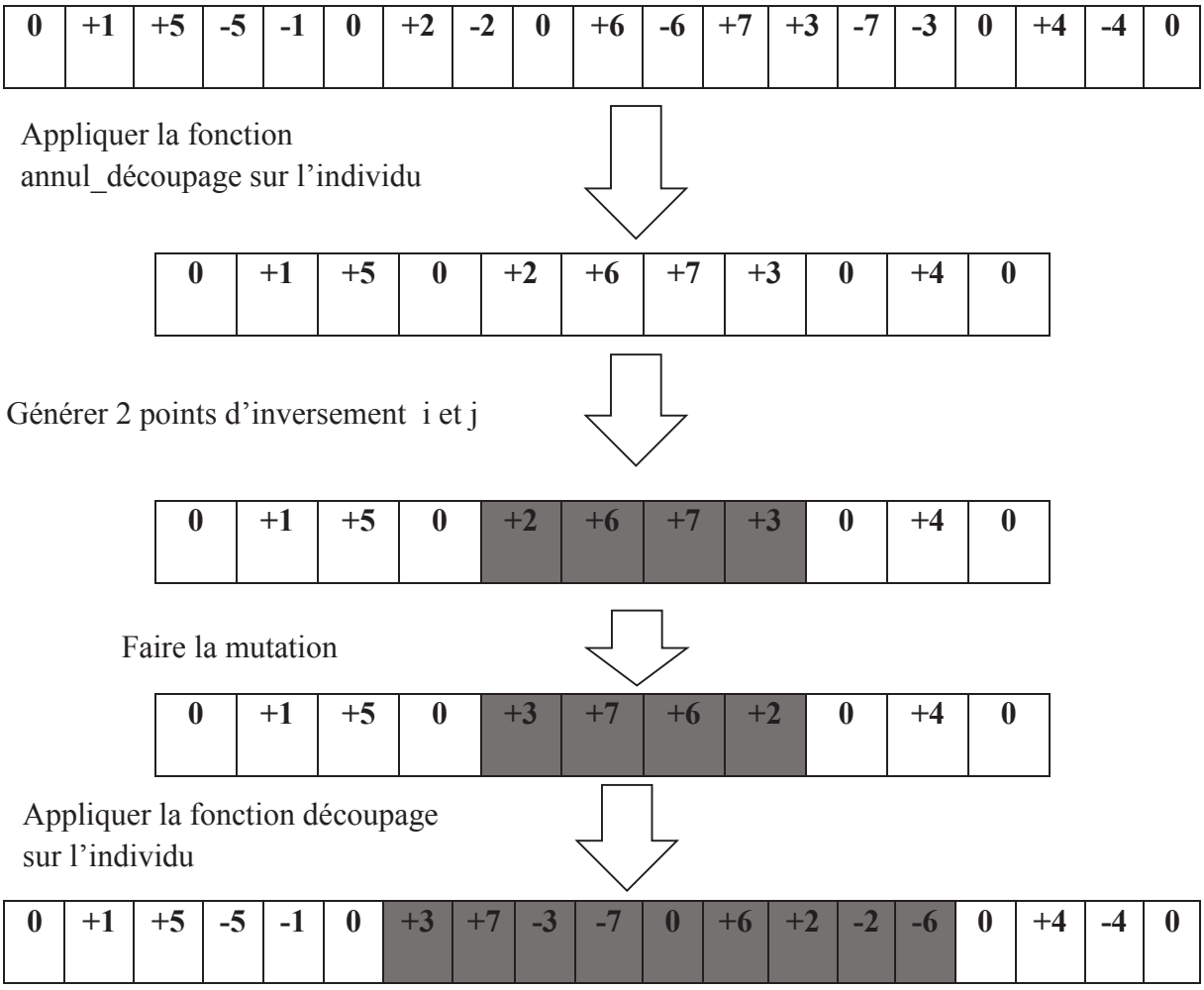

Figure 18: exemple complet de mutation

3.3.6 La reconstruction de la population :

Dans cette étape, on remplace la totalité de la population P (ancienne population) par la nouvelle population P' qui a été obtenue par application successive des opérateurs de sélection, de croisement et de mutation.

3.3.7 La condition d'arrêt :

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

3.3.8 L'algorithme génétique pour le problème de transport de patient :

L'algorithme 3.6 illustre notre algorithme génétique pour résoudre le problème de transport de patient qui contient toutes les techniques décrites ci-dessus :

Algorithme 3.6 : L'algorithme génétique pour le problème de transport de patient

```
Début
    Génération de la population initiale ;
    Tant que (le nombre des générations inférieur au nombre des itérations choisies) Faire
        Sélectionner un demi des éléments de l'ancienne population par la méthode de sélection
        Ajouter ces éléments à la nouvelle population ;
        Tant que (il reste des individus sélectionnés qui n'ont pas participé dans le croisement)
            Faire
            Choisir deux individus aléatoirement ;
            Croiser ces deux individus ;
            Ajouter les deux individus fils à la population ;
        Fin tant que
        Si (la population n'est pas complète) alors
            Compléter la population à partir du reste des éléments de l'ancienne population qui ne
            sont pas sélectionnés ;
        Fin si
        Pour (chaque élément de la population) Faire
            Appliquer la mutation selon une probabilité générée ;
        Fin pour
            Remplacer l'ancienne population par la nouvelle ;
    Fin tant que
```

Fin.

3.4 Architecture de l'application :

Notre application se compose de 10 classes, qui sont groupées dans un seul package projet :

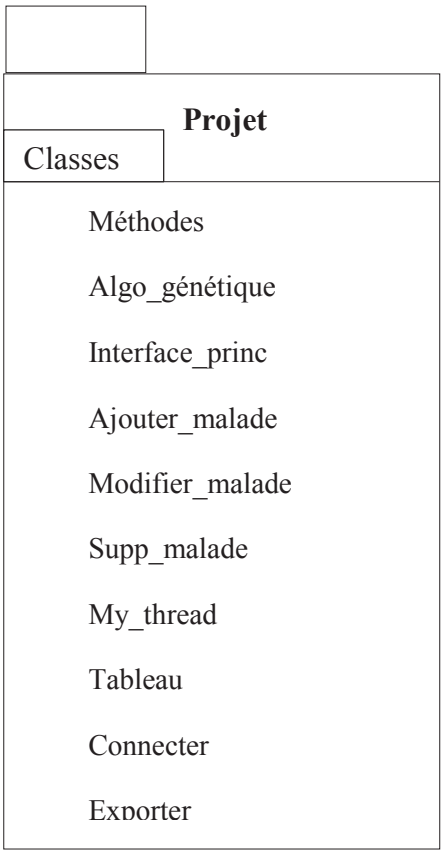

Figure 19: le package « projet »

3.4.1 Diagramme de classes de l'application :

Afin de voir les relations entre les classes de l'application, nous allons les représenter dans un Organigramme présenté dans la figure 20 suivante :

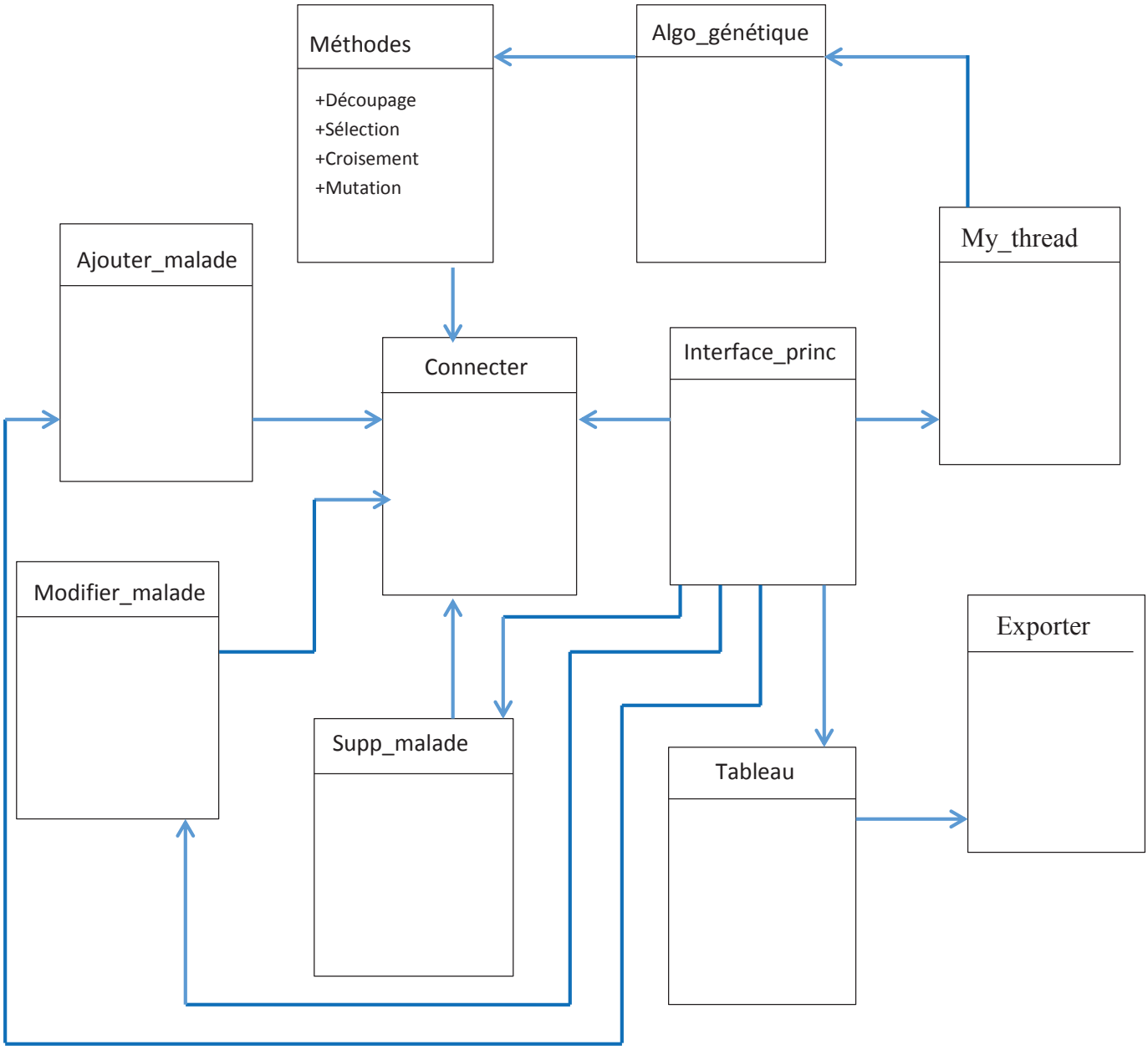

Figure 20: Organigramme de l'application

- **La classe Méthodes :**

La classe Méthodes contient tous les fonctions et les procédures qui réalisent les défirent étape d'algorithme génétique. Parmi les méthodes de cette classe on peut citer :

+Cursor : cette méthode permet d'extraire les informations des malades qui dialysent dans un seul jour, à partir de la base de données, et les mettre dans une table.

+Requêtes : la méthode Requêtes permet de construire une table de 2n requêtes (n est le nombre des malades dans la table de la méthode Cursor), donc pour chaque malade on crée 2 requête, une pour l'aller vers l'hôpital, et l'autre pour le retourné à son maison.

+individu : cette méthode permet de construire un individu par l'insertion aléatoire des nœuds en respectant les contraintes de précédence et de répétition.

+Mat_distance : elle permet de charger la matrice de distance entre les nœuds.

+découpage : permet de découper un individu en tournées, et calculer la longueur de l'individu.

+population_initiale : permet de construire une population initiale de taille définis par l'utilisateur.

+sélection : permet de sélectionner des individus par la méthode de sélection précédant.

+annule_découpage : permet d'annuler le découpage de l'individu en tournées.

+Croisement : permet d'appliquer l'étape de croisement avec une probabilité définis par l'utilisateur.

+Mutation : permet d'appliquer l'étape de mutation avec une probabilité définis par l'utilisateur.

+Evaluation : cette méthode permet de trouver la meilleure solution d'un ensemble des solutions.

+Représentation : permet d'organiser une solution dans un tableau, en remplaçant les requêtes par les informations des malades (nom, prénom…) à partir de la base de données.

- **La classe Algo_génétique :**

Cette classe contienne une seul méthode, qui appeler les méthodes de la classe Méthode, pour appliquer l'algorithme génétique.

- **La classe Interface_princ :**

La classe Interface_princ est la classe principale de projet. Elle permet de construire l'interface graphique.

- **La classe Tableau:** Cette classe contient un tableau qui se charge automatiquement à la fin d'exécution par la meilleure solution trouvée.
- **La classe Exporter:**

La classe Exporter contient une seul méthode Export, qui permet de sauvegarder la table de classe Tableau dans un fichier PDF.

- **La classeAjouter_malade, Modifier_malade et Supp_malade:**

Ces classes permettent la gestion des malades (ajouter, modifier et supprimer).

- **La classe Connecter:**

Cette classe contient une seul méthode Connecter, qui permet la connexion à la base de données.

La classe My_thread :

La classe My_thread contient une seul méthode Run permet de crée un thread.

3.5 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.

Chapitre04 :

Résultats et discussion

4.1 Introduction :

Dans ce dernier chapitre nous allons présenter les différents outils de programmation utilisés dans le développement de notre application et nous donnons aussi une description des différents composants de cette dernière.

4.2 Environnement de travail :

4.2.1 Présentation des outils de développement de l'application :

➢ ***Langage de programmation : JAVA***

Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- Java est interprétée : la source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java (Java Virtual Machine (JVM)).

- Java est portable : il n'y a pas de compilation spécifique pour chaque plate-forme. Le code reste indépendant de la machine sur laquelle il s'exécute.
- Java est simple :le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs,
- Java est fortement typée : toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données [22].

➢ ***Netbeans :***

NetBeans est à l'origine un EDI (environnement de développement intégré) Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelque part en 2002, Sun a décidé de rendre NetBeans open-source, sa conception est complètement modulaire. Ce qui fait de NetBeans une boite à outils facilement amélioriable ou modifiable.

NetBeans comprend les fonctions générales suivantes [22]:

- configuration et gestion de l'interface graphique des utilisateurs,
- support de différents langages de programmation,

- traitement du code source (édition, navigation, formatage, inspection..),
- fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- accès et gestion de bases de données, serveurs Web.

➢ ***SGBD : ORACLE***

Pour créer la base de données de notre application, nous avons utilisé Oracle qu'est un système de gestion de base de données relationnel. On peut définir le SGBD comme un système complexe permettant de gérer de manière efficace, un volume important de données structurées. Oracle permettant d'assurer [23] :

- La définition et la manipulation des données.
- La cohérence des données.
- La confidentialité des données.
- L'intégrité des données.
- La sauvegarde et la restauration des données.
- La gestion des accès concurrents.

4.3 Présentation de l'application développée :

L'interface principale de notre application est composée de deux anglés le premier pour configurer les paramètres de l'algorithme génétique et le deuxième pour la gestion des patients.

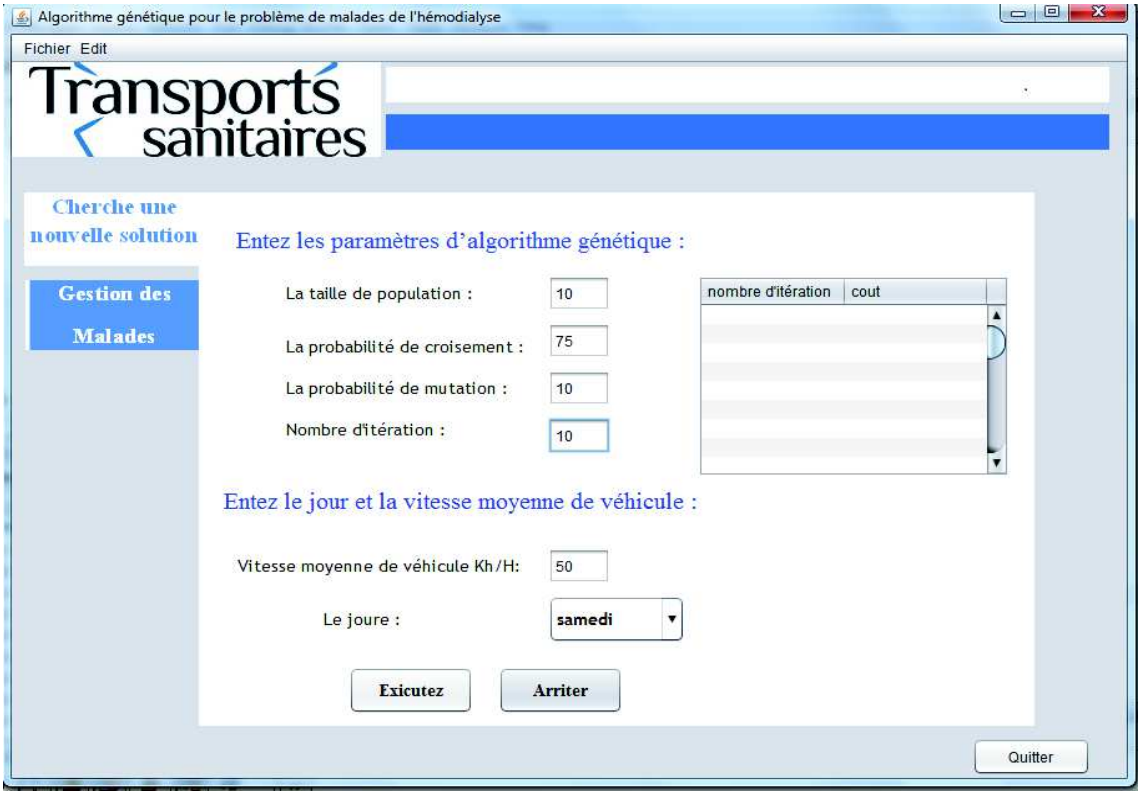

Figure 21: L'interface principale d'application

- **L'onglés de l'algorithme génétique :**

A l'aide de cet onglés l'utilisateur doit spécifier les paramètres de l'algorithme génétique qui sont le nombre des itérations, la taille de la population, la probabilité de croisement, et celle de la mutation.

L'utilisateur doit préciser la vitesse moyenne des véhicules utilisés, puis sélectionner un jour pour traiter ses requêtes, enfin lancer l'exécution par le bouton concerné.

Dans le cas où l'utilisateur a oublié de mettre ces paramètres, des valeurs par défaut ont été posé pour chaque paramètre.

Pour les valeurs de chaque champ il y a des contraintes à respecter, dans le cas où il y a une violation de la valeur d'une contrainte, un message qui précise cette violation est affiché.(voir les figure 22, 23, 24, 25).

Figure 22: Message d'erreur 1

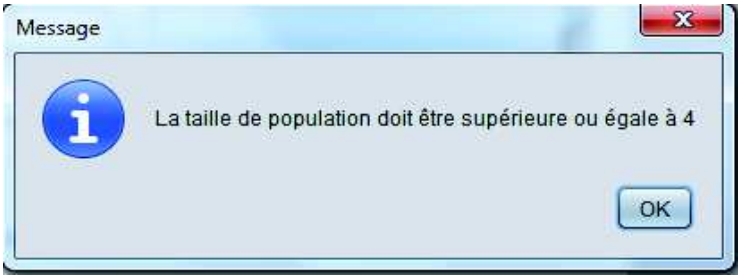

Figure 23: Message d'erreur 2

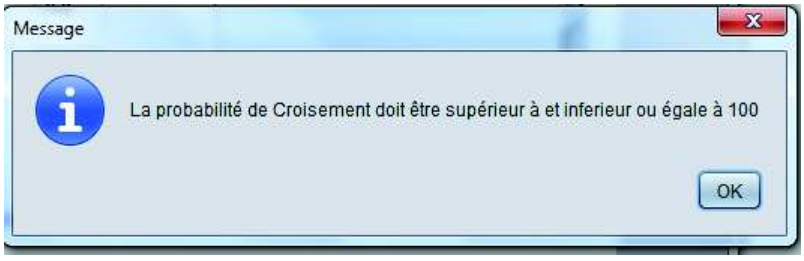

Figure 24:Message d'erreur 3

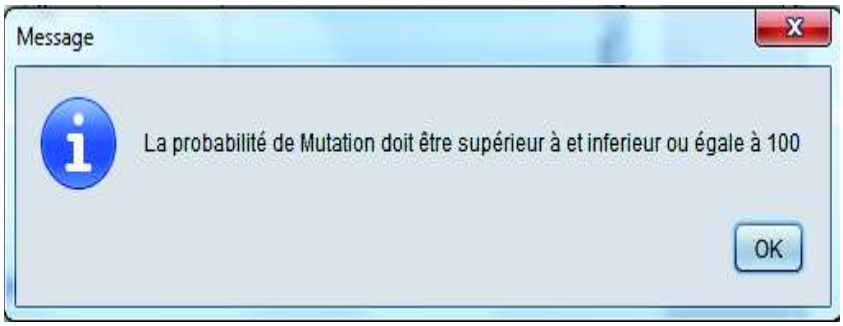

Figure 25: Message d'erreur 4

Si les valeurs des paramètres sont valides, l'application lance l'exécution de l'algorithme génétique et affiche la meilleure solution de chaque itération dans le tableau encerclé dans la figure ci-dessous.

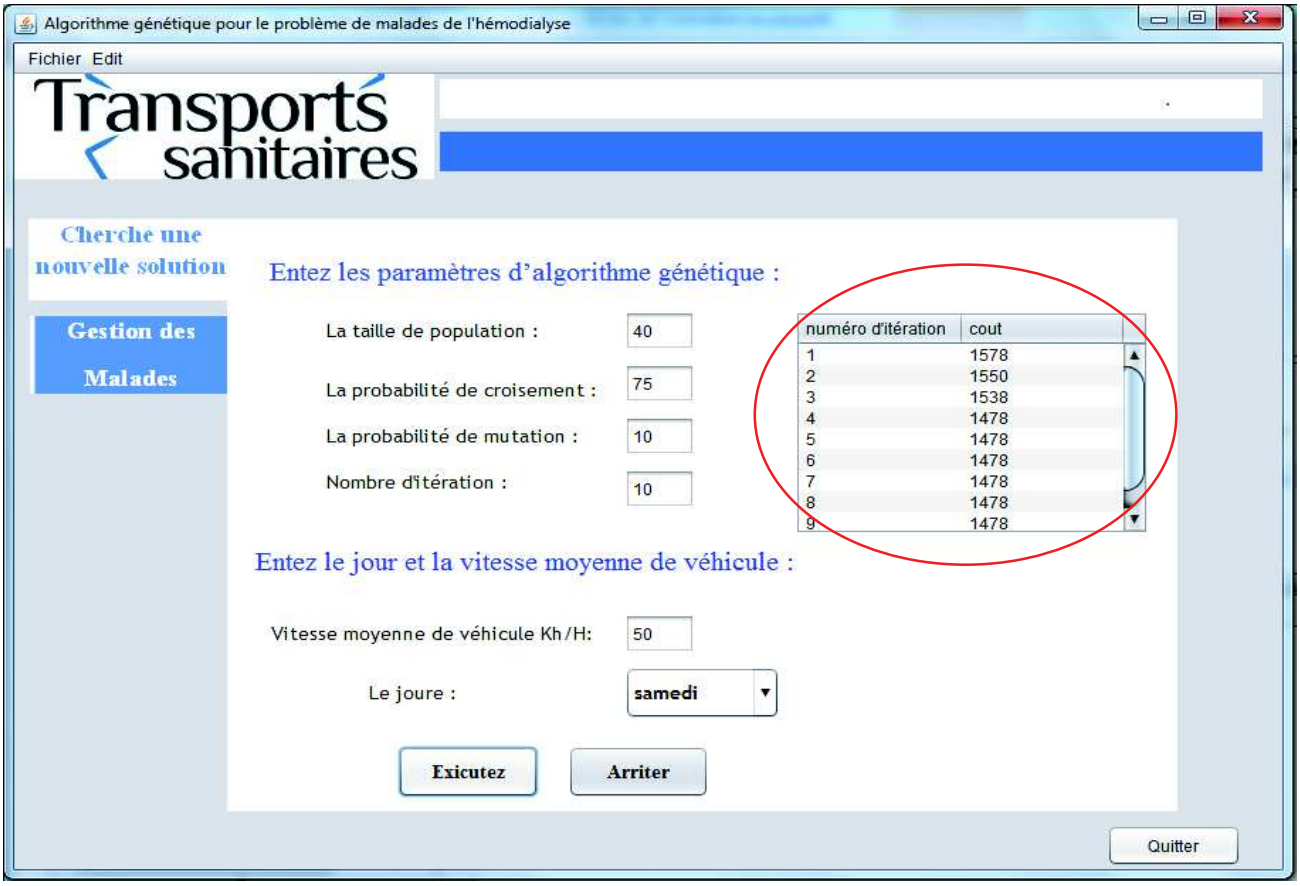

Figure 26:l'exécution de l'algorithme génétique

On remarque que le cout de la solution est amélioré à chaque itération, et à la fin de l'exécution on obtient la meilleure solution obtenu par l'algorithme génétique qui va être organisée dans un tableau par extraction des informations de chaque patient à partir de la base de données. La figure suivante donne un exemple d'une solution. Chaque requête de transport et composée de deux partie la première contient l'arrivé vers l'hôpital, et la deuxième c'est le

retour vers son domicile, donc la première moitié du tableau contient les arrivées et la deuxième moitié contient le retour.

La première colonne spécifie le véhicule qui doit réaliser l'action dans la deuxième colonne, cette action peut être un embarquement (prendre) ou un débarquement (déposer).

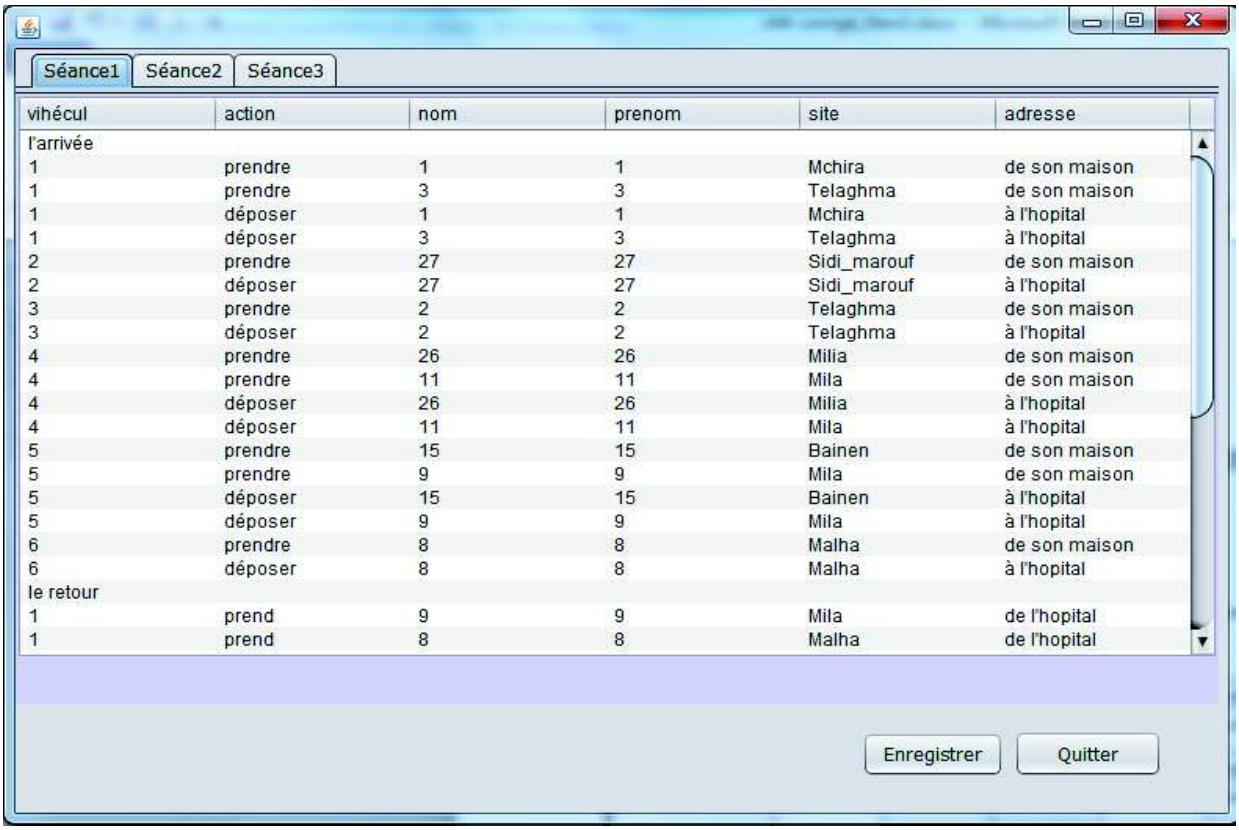

Figure 27:la représentation finale de la meilleure solution

On peut sauvegarder cette solution dans un fichier pour l'afficher aux conducteurs, elle contient le programme détaillé pour chaque conducteur et le cout de la solution et la distance totale parcourue.

Le progromme de'samedi'

véhicul	action	nom	prénom	adresse	le site
l'arrivée					
1	prend	9	9	Mila	de son maison
	prend	11	11	Mila	de son maison
	prend	27	27	Sidi_maro uf	de son maison
	déposer	9	9	Mila	à l'hopital
	déposer	11	11	Mila	à l'hopital
	déposer	27	27	Sidi_maro uf	à l'hopital
2	prend	15	15	Bainen	de son maison
	déposer	15	15	Bainen	à l'hopital
3	prend	3	3	Telaghma	de son maison
	prend	1	1	Mchira	de son maison

Figure 28:l'enregistrement de solution dans fichier PDF

- **L'onglés de la gestion des patients :**

Figure 29: la gestion des malades

Cet onglé représente un outil de gestion de la base de données des patients, il nous permet d'ajouter, de modifier, ou de supprimer les informations d'un patient, ces informations contient les coordonnées du patient et le données concernant ses séances de dialyse qui sont nécessaire

pour l'exécution de l'algorithme génétique. Les figures suivantes représente quelques fonctionnalités de cet onglé.

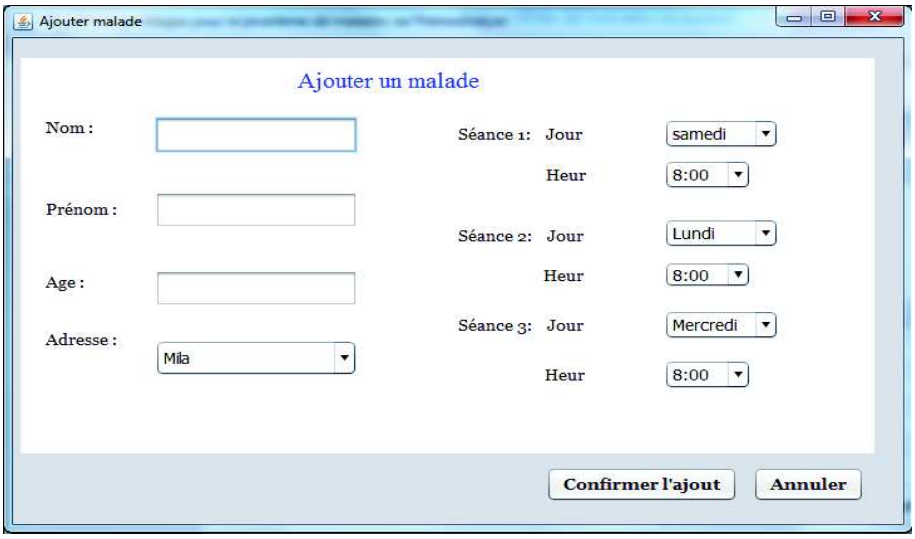

Figure 30: Ajouter un malade à la base de données

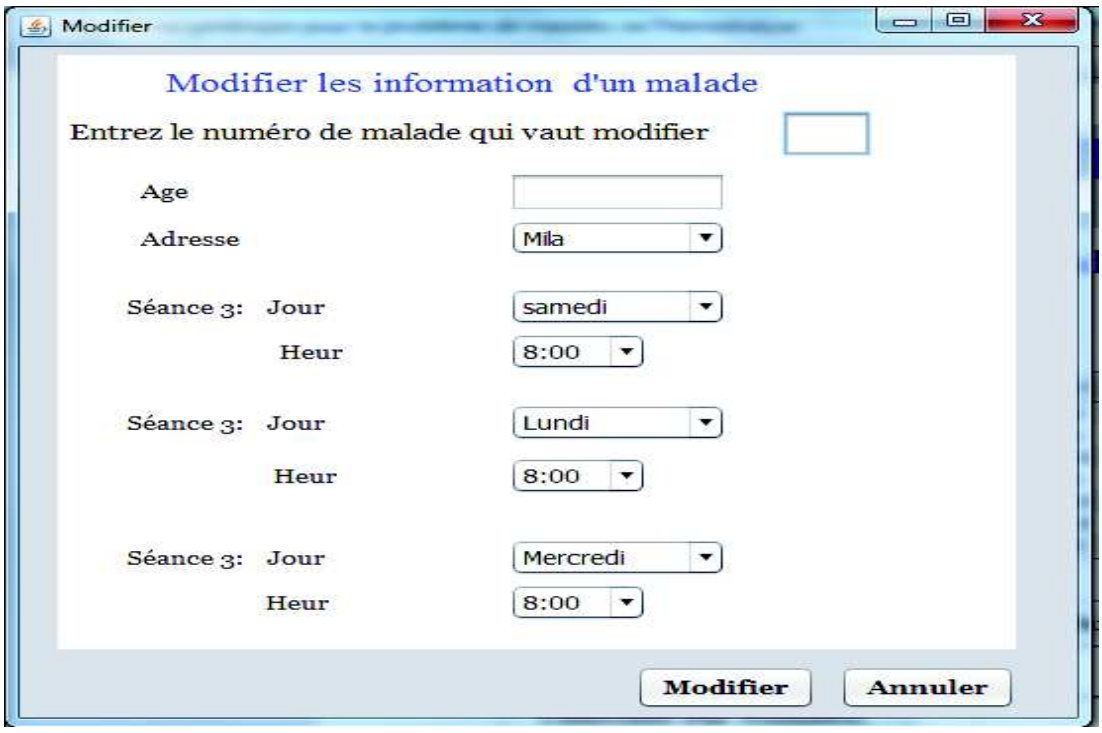

Figure 31 : Modifier un malade à la base de données

Figure 32: Supprimer un malade à la base de données

4.4 Les données réelles de l'opérateur de transport sanitaire :

Pour tester l'efficacité de l'algorithme génétique développé, on a contacté un opérateur de transport sanitaire pour obtenir les données réelles des malades. L'opérateur s'appelle « Dianephros » qui travaille au niveau de la wilaya de Mila et une partie de la wilaya de Jijel (la commune de Sidi-Maarouf et la daïra d'El-milia), il contient 31 malades et 7 voitures.

Après une forte négociation on a pu le convaincre pour nous donne les informations de ses patients et même le programme journalier appliqué présenté dans la figure suivante.

Figure 33: Programme des véhicules utilisable par l'opérateur

4.5 Comparaison des résultats :

Le tableau ci-dessous donne le calcul de la distance totale parcouru par les véhicules en kilomètre et le nombre des voitures utilisés par l'opérateur, et la solution obtenue par l'application de notre algorithme génétique sur les mêmes données.

Résultats / jours	La réalité appliquée par l'opérateur de transport sanitaire (Manuel)		La solution obtenue par application de l'algorithme génétique développé		Gain (la différence de distance totale et le nombre de véhicules utilisé)	
	Distance totale parcourue	**Nombre des véhicules utilisé**	**distance**	**Nbr des véhicules**	**Distance**	**Nbr des véhicules**
Samedi	1536km	**5**	1336km	4	200km	1
Dimanche	1292km	4	1074km	3	218km	1

Tab.1 *Table de comparaison des résultats obtenus par l'AG et La réalité appliquée*

A partir du tableau on remarque que la solution obtenue par notre algorithme génétique permet de nous donner un gain de 200 km à samedi et 218 km à dimanche (sa veut dire 1254 km sur semaine) et de minimiser le nombre de véhicule utilisé, tout cela justifie l'efficacité de la méthode appliquée.

4.6 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 présenté l'interface de cette dernière, en fin nous avons fait une comparaison entre le programme utiliser par un opérateur de transport sanitaire et notre résultat.

L'heuristique basée essaime 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

Conclusion générale :

Dans ce travail, nous avons développé une application pour le transport sanitaire, plus particulièrement le transport des patients aux différents centres de l'hémodialyse. Ce problème est une variante du problème de tournées de véhicules qui est le transport à la demande, ce dernier est appartient à la classe des problèmes dites NP-complet d'où il n'existe pas des algorithmes qui donnent la solution optimale en un temps raisonnable. Cela nous a conduit à utiliser des méthodes dites approchées dans le but de trouver une solution proche de l'optimale dans un temps raisonnable. Pour cela on a choisit l'algorithme génétique.

L'algorithme génétique est une métaheuristique basée sur le principe de la sélection naturelle et de la génétique, qui est plus adaptée aux problèmes d'optimisation difficiles.

Afin de mesurer l'efficacité de notre l'algorithme génétique nous avons utilisé des donnée réelles d'un opérateur de transport sanitaire qui travail au niveau de la wilaya de Mila.

Les résultats obtenus montrent un gain de 200 km dans la distance totale parcouru par les véhicules et une réduction du nombre de véhicules utilisés par un ce qui montre l'efficacité de la méthode utilisée.

[1] C.H. Christiansen.Elements of vehicle routing under uncertainty. PhD Thesis,Departement of Business Studies, Aarhus School of Business, University of Aarhus.April, 2007.

[2] M. Gendreau and J.Y. Potvin. "Dynarnic vehicle routing anddispatching". Working paper, Université de Montréal, 1998.

[3] M.Dell'Amico, M.Monaci, C .Pagani, and D.Vigo.Heuristic approaches for the Fleet Siza and Mix Vehicle Rpoting Problem withe Time Windows. Working paper, University of Modena, Italy,2006

[4] F.Pereira ,J.Tavares , P.Machado , and E.Costa.GVR : a New Genetic Representation for the Vehicle Routing Problem.Artificialintellegence and Cognitive Science : 13th Irish Conference Proceedings , page 95-102,2002.

[5] M. J. Groth "Stochastic considerations in vehicle routingProblems". Transport Optimization, June 17 2002.

[6] E.D. Taillard, "A heuristic column generation method for theheterogeneous fleet VRP". RAIRO-Operations Research, 33,pages PP. 1-14, 1999.

[7] F.Alonso , M.J. Alvarez, and J.E Beasley. A Tabu Search Algorithm for the Periodic Vehicle Routing Problem With Miltiple Vehicle Trips and Accessibility Restriction.To appear in journal of the OR Society ,2006

[8] M. Witucki, P. Dejax and M. Haouari. "Un modèle et unalgorithme de résolution exacte pour le problème de tournées devéhicules multipériodique". Ecole Centrale Paris, Laboratoire Productique-Logistique, Grande Voie des Vignes, 92295Châtenay-Malabry. Ecole Polytechnique de Tunisie, 2070 LaMarsa, Tunisia 1997.

[9] H.N. Psaraftis, "An exact algorithm for the single vehiclemany-to-many immediate request dial-a-ride problem withtime windows". Transportation Science, 17, 351-357, 1983.

[10] C. Rego et C. Roucairol. "Le problème de tournées de véhicules:Etude et Résolution Approchée". Technical Report, inria, Février 1994.

[11] Z. BOUSMINA , H.GUELLIL«Une approche d'optimisation coopérative à base d'agents optimiseurs appliquée au problème de transport à la demande(TAD)». Mémoire préparé En vue de l'obtention du diplôme de Masteren Informatique, 2013.

[12] G. Laporte. "The vehicle routing problem: An overview of exactand approxirnate algorithrns". European Journal of Operational research ,59,page 345-358 1992b

[13] S. M. Sait et H. Youssef; iterative computer algorithms withapplications in engineering: solving combinatorial optimizationproblems, IEEE computers society, 1999.

[14] L. M. Gambardella, A.E. Rizzoli, F. Oliveriob, N. Casagrande,A.V. Donati, R. Monternanni and E. Lucibello. "Ant ColonyOptirnization for vehicle routing in advanced logistics systems"IDSIA, Galleria 2, 6928 Manno, Switzerland and AntOptima,via Fusion 4, 6900 Lugano, Switzerland, 2003.

[15] M. E. Bouzgarrou. Parallélisation de méthode du "Branche and Cut" pour résoudre le problème de voyageur de commerce. thése,laboratoire de Modélisation et calcul , Institut d'informatique et de Mathématique Appliquées de Grenoble,1998

[16] LAGA SAIDA, NOUIOUA LEILA, « Deux approches méta-heuristiques hybridesbasées sur les colonies d'abeilles pour la résolution du problème de la Tcoloration des graphes», Mémoire de fin d'études Pour l'obtention du diplômed'ingénieur en Systèmes Informatiques, Ecole National Supérieure d'Informatique(ESI) Oued-Smar, Alger,2008/2009.

[17] H. Zgaya, « Conception et optimisation distribuée d'un système d'information d'aide à la mobilité urbaine : Une approche multi-agent pour la recherche et lacomposition des services liés au transport », 240p, thèse de doctorat en Automatiqueet Informatique Industrielle, Ecole centrale de tille, 6 juillet 2007.

[18] J. Alliot, Nicolas Durand., « Algorithmes génétiques », March 14, 2005.

[19] L Davis. Handbook of Genetic Algorithms. Van nostrand reinhold, New-York, 1991.

[20] 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.

[21] M.HAJ RACHID, « les problèmes de tournées de véhicules en planification industrielle : classification et comparaison d'opérateurs évolutionnaire », Thèse de doctorat 2010.

Sites Web :

[22] www . jmdoudoux.fr. /java/ dej/ chap-presentation.htm.

[23] www.it--expertise.com/ /wp-content/uploads/2102/07/IT-84. pdfOraclz .