dre...../F.S.S.A/UAMOB/2018

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AKLI MOHAND OULHADJ-BOUIRA

Faculté des Sciences et des Sciences Appliquées
Département d'informatique

Mémoire de fin d'étude

Présenté par :

MESLEM yasmine.
MOSTEGHANEMI soraya.

En vue de l'obtention du diplôme de **Master 02**en **:**

Filière : INFORMATIQUE.
Option : Ingénierie des Systèmes d'Information et des Logiciels.

Thème :

Outil d'aide à la décision pour l'optimisation du problème de transport : Hybridation d'algorithmes métaheuristiques.

Devant le jury composé de :

ABBAS akli	Grade ph.D	UAMOB	Président
CHOUIREF zahira	Grade ph.D	UAMOB	Encadreur
BADIS lyes	Grade professeur	UAMOB	Examinateur
HAYI yacine	Grade M.A	UAMOB	Examinateur

Année Universitaire : 2018/2019

Remerciements

Nous voulons à remercier en premier lieu le bon Dieu de nous avoir donné la force et le courage pour accomplir ce travail.
nous voulons exprimer par ces quelques lignes de remerciements nos gratitudes envers tous ceux en qui, par leur présence, leur soutien, leur disponibilité et leurs conseils nous avons trouvé le courage afin d'accomplir ce mémoire.

Nous tenons encore à exprimer notre profonde gratitude et nos sincères remerciements à Mme.CHOUIREF zahira pour nous avoir confié ce travail, leur suivi, leur conseils.

Nous tenons à remercier Mr.HAYI yasine pour sa disponibilité et des conseils.

Aussi que les membres de jury trouvent ici nos remerciements les plus vifs pour avoir accepté d'honorer par leur jugement notre travail.

Dédicaces

Du profond de mon cœur je dédie ce travail à tous ceux qui me sont chers.

A mes chers parents,

Que ce travail soit l'expression de ma reconnaissance pour vos sacrifices consentis, votre soutient moral et matériel que vous n'avez cesse de prodiguer. Vous avez tout fait pour mon bonheur et ma réussite. Que dieu vous préserve en bonne santé et vous accorde une longue vie. Vous étiez toujours présents pour m'aider et m'encourager.

A mes sœurs et tout ma famille.

MESLEM yasmine.

Dédicaces

Je ne peux commencer sans évoquer le nom d'ALLAH notre Dieu le tout puissant qui m'a donné la patience, la santé et bien veillent qui ma trace le chemin de ma vie. Je dédie ce modeste travail à mes chers parents « Mosteghanemi Rabah et Zanoune Zahia » jamais je ne saurais m'exprimer quant aux sacrifices et aux dévouements qu'ils ont consacré à mon éducation et à mes études,

A mes adorables sœurs,

A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur,

A toute personne m'ayant aidé de près ou de loin.

MOSTEGHANEMI soraya.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AG	**A**lgorithme **G**énétique.
AT	**A**lgorithme **T**abou.
TGV	**T**rain à **G**rande **V**itesse.

Introduction générale

Aujourd'hui, nous constatons une évolution dans la structure des déplacements. Cette évolution est due à l'étalement urbain et à l'apparition des pôles attractifs aux périphéries des villes. En outre, l'augmentation considérable du nombre de citoyens, en particulier dans les grandes villes, la croissance de l'usage des voitures pose des problèmes aigus, tant sociaux qu'économiques et environnementaux. Dans ce contexte, les individus cherchent des chemins optimaux et proches de leurs attentes. La détermination de la route optimale est un problème multicritères non-linéaire.

Les métaheuristiques sont des algorithmes génériques, souvent inspirés de la nature, conçues pour résoudre des problèmes d'optimisation complexes.

Notre travail consiste à hybrider quelques méthodes métaheuristiques existantes pour résoudre les problèmes connus dans le domaine du transport qui correspondent généralement aux problèmes d'optimisation multi-objectifs, ensuite proposer un outil d'aide à la décision pour un choix efficace de la route optimale.

Problématique :

Dans le domaine de transport, la distance du trajet et très importante. Elle permet de déterminer le temps du trajet ainsi que les frais de circulation qui incluent le taux de l'énergie consommée. La longueur de la distance a des impacts dangereux sur la nature car elle est parmi les raisons principales de la pollution de l'air.

Dans le domaine de transport, elle se présente une forte demande d'optimisation de la distance parcourue pour le gains de temps, d'argent et réduire ainsi la pollution. Le problème étudié dans le cadre de ce mémoire est multiobjectifs. Pour résoudre ce type de problèmes, nous avons opté pour l'hybridation des métaheuristiques, vue qu'elles sont efficaces et fiables.

Objectifs :

Le travail réalisé vise à répondre aux objectifs suivant :

- Satisfaire au mieux les demandes de transport en minimisant les coûts engendrés par ces tournées (distance parcourue, temps de circulation) en fournissant une bonne qualité de service aux voyageurs.
- Fournir un outil d'aide à la décision permettant une utilisation souple et conviviale de notre application.

Pour ce qui est de la présentation du plan de ce mémoire, le travail que nous avons mené s'organise comme suit :

Le premier chapitre dresse un état de l'art du transport routier, sa modélisation et la présentation de l'optimisation combinatoire.

Le deuxième chapitre présente la notion des heuristiques et métaheuristiques avec quelques exemples, ainsi qu'une classification des métaheuristiques et la notion d'hybridation.

Dans le troisième chapitre nous mettons l'accent sur les trois algorithmes que nous avons étudié dans le cadre de ce mémoire : l'algorithme de Dijkstra, les algorithmes génétiques, la recherche tabou. Ainsi que l'hybridation.

Le quatrième chapitre présente la discussion des résultats obtenus et l'implémentation de notre application d'aide à la décision pour l'optimisation du problème de transport.

Enfin, nous terminerons ce mémoire par une conclusion générale qui résume le travail effectué et quelques perspectives.

Chapitre 1

Problème de transport et Optimisation combinatoire

1.1 Introduction

Le transport routier, par sa souplesse d'utilisation et grâce aux infrastructures, est devenu le mode de transport prépondérant. Le cheminement par route est fortement sollicité par les différentes activités. Au sein de l'ensemble de transport, le secteur routier occupe une place absolument prépondérante dans le transport extérieur.
Dans ce chapitre nous présentons l'état du l'art de transport routier et sa modélisation ainsi que les problèmes d'optimisation liés à ce domaine.

1.2 Le transport

Le transport [1] est le déplacement d'objets, de marchandises ou d'individus d'un endroit à un autre à travers l'espace caractérisant la circulation (ensemble des déplacements) dont l'intensité dans le temps et dans l'espace détermine le trafic et les flux (déplacements massifs de personnes, de biens, de l'argent et d'informations).
Ces déplacements font appel à des moyens techniques appelés moyens de transport (ensemble de techniques utilisées pour effectuer les déplacements) qui s'inscrivent dans les territoires grâce aux voies de communication (installations permettant la circulation des personnes et des biens) à savoir la route, le rail, la voie d'eau, les conduites, la voie aérienne, etc [1].

Le type de transport peut se caractériser par son appartenance au deux secteurs [2] :

- **Le transport public :** organisé par une personne pour le compte d'un client comme le transport des voyageurs.
- **Le transport privé :** organisé par une personne pour son propre compte comme le transport de marchandises.

Chacun de ces types de transport, incluant le transport de personnes et de marchandises, peut être subdivisé en sous-types [2] :

- Le transport terrestre (routier, ferroviaire, transport par canalisation).
- Le transport aérospatial (aérien et spatial).
- Le transport aquatique (maritime et fluvial).
- Le transport par câble (suspendu ou double contact).

La Figure 1.1 représente le taux d'utilisation des moyens de transport [1].

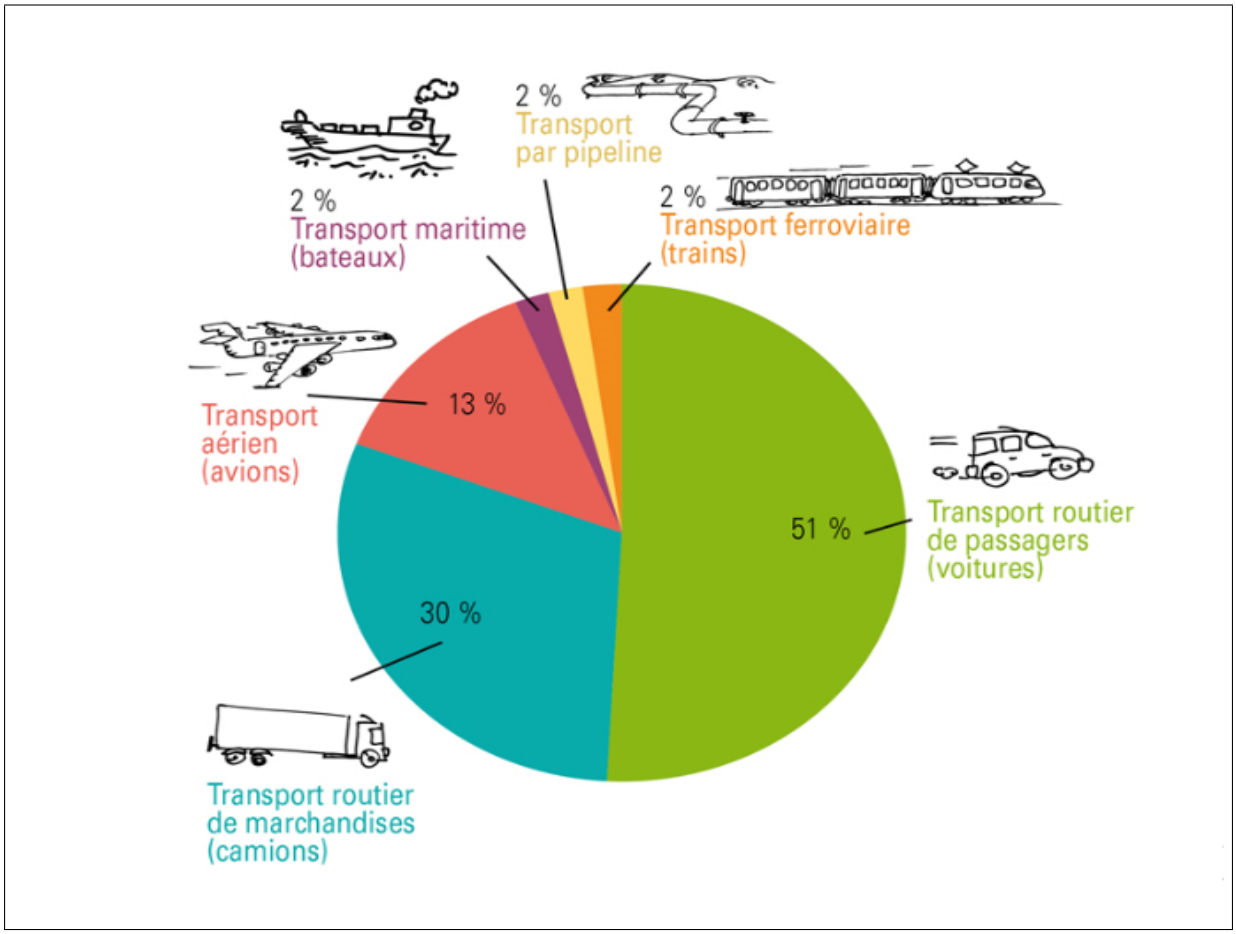

FIGURE 1.1 – Taux d'utilisation des moyens de transport.

Le transport représente un des piliers fondamentaux du développement durable et de la prospérité de tout pays. Des systèmes de transport efficaces et des réseaux modernes sont donc une nécessité pour le bien-être social, la production à grande échelle et le développement économique. Il est devenu un secteur économique lié à l'industrie du transport qui s'est développée simultanément dans les domaines public et privé depuis la révolution industrielle [3].

1.2.1 L'évolution du transport à travers l'histoire

Vu sa contribution majeure dans le développement , le transport a connu une évolution cruciale à travers l'histoire dont voici ses grandes étapes [4] :

I) Les transports durant l'ère pré-industrielle (avant 1800) :

- La force animale, pour le transport terrestre.
- La force du vent, pour le transport maritime.

II) Les transports et la révolution industrielle (1800-1870) :

- Invention du moteur à vapeur (profonde mutation du système de transport par la mécanisation du principe de fonctionnement des transports terrestres et maritimes).
- 1830 : La première ligne commerciale ferroviaire reliant Manchester à Liverpool.
- 1838 : Mise en place des premières lignes maritimes à service régulier de passagers par navire à vapeur (transatlantique).
- 1869 : Ligne ferroviaire transcontinentale entre New-York et San Francisco (réduisait la durée de la traversée de six mois à une semaine).

Conséquences :

- Dans cette période, il s'agit d'un accomplissement pivot, rendant accessible à l'est des États-Unis, de vaste stocks de ressources ainsi que de nouvelles régions agricoles.
- La principale conséquence de la révolution industrielle fut une spécialisation des services de transport jumelée à l'établissement de vastes réseaux de distribution d'énergie et de matières premières.

III) Émergence des transports moderne (1870-1920) :

- 1870 : Le développement du chemin de fer et le tramway pour le transport intra-urbain par l'énergie électrique.
- Invention du vélocipède comme un moyen de transport pour la classe ouvrière et de loisir pour la classe bourgeoise.
- Fin du 19ème siècle, forte croissance du transport international due à l'invention du moteur au mazout (augmentation de vitesse de propulsion).

Conséquences :

- Dans cette période, il y avait l'apparition des grandes villes portuaires.
- Réduction des frais de transport due à cette nouvelle invention (moteur au mazout).
- Le port devient un complexe industriel autour duquel s'agglomeraient les activités faisant appel à l'usage massif de matières premières.

IV) Les transports durant l'ère (1920/1970) :

- La création du moteur à combustion interne (qualité et rapidité), cela a développé l'industrie d'automobile, le car et le camion.
- 1903 : Le premier vol revenant aux frères Wright (il est exploité premièrement par les services postaux puis par le transport de passagers interurbains).
- 1919 : la première ligne commerciale de transport aérien entre la France et l'Angleterre, puis entre l'Europe et les États-Unis (1920 et 1930).

Conséquences :

- Cette période a été marquée par la production de masse des véhicules qui provoque un profond changement au système de production industriel à partir de 1913.
- La diffusion massive de l'automobile a changé le niveau de vie de la population ainsi que la structure des villes (fordisme [1]).
- Apparition des régions aussi denses et productives comme le nord-est des États-Unis.
- La structuration et l'interconnexion du système urbain par des réseaux de transport sous forme d'une masse urbaine intégrée (on parle ici de mégalopole [2]).

1. modèle d'organisation créait par Henry Ford, basé sur une production standardisée de masse.
2. Ville très grande et peuplée d'au moins un million d'habitants.

V) Nouveau contexte pour le transport (après 1970/ l'ère poste fordiste) :

- 1969 : Le premier vol public du Boeing 747 entre New York et Londres.
- 1970 : Développement des télécommunications.
- Développement du transport des conteneurs terrestres et maritimes.
- L'émergence du TGV [3] notamment en France (300km/h) et au japon (275km/h).

Conséquences :

- Cette période a été marquée par les crises pétrolières qui ont induit un ensemble d'innovations reliées aux modes de transport (recherche des sources d'énergie alternatives, création des voitures électriques).
- L'accroissement du nombre de mouvements de biens aux échelles tant locale, régionale que internationale qui correspondent au développement de l'économie mondiale.

1.2.2 Enjeux du transport

L'évolution de l'industrie de transport a permis à travers l'histoire de profondes mutations dans le monde. En effet, grâce à son effet multiplicateur et d'entrainement on a pu assister aux effets suivants [4] :

- Un accroissement de l'activité industrielle et l'apparition des villes.
- Création d'emplois directs et indirects.
- Une amélioration des revenus de ménages.
- Une nette augmentation de la consommation.
- Une amélioration de la qualité de vie et de bien-être.
- Une contribution dans le développement de la mondialisation.

1.2.3 Le transport routier

Le transport routier [5] occupe une place importante dans le système de transport du pays. La commission des transports, des télécommunications et du tourisme présente le transport comme nécessaire (indispensable) à l'enrichissement général et à la modernisation du pays, diversifié et adapté aux besoins des usagers . Le transport routier rassemble les modes de transport suivants :
les véhicules particuliers, les véhicules utilitaires (légers et lourds) et les deux-roues.

3. Train à grande vitesse.

Pour déterminer le Coût du transport routier, il faut d'abord savoir la catégorie du système routier utilisé.

I- Les catégories du transport routier :

Les système de transport routier sont classés en deux catégories [6] :

- Les système de transport non guidés (automobile, bus..).
- Les système de transport guidés (tramway, métro..).

Dans le tableau ci dessous nous avons présenté les avantages et les inconvénients des modes de transport routier.

Mode de transport	**Avantages**	**Inconvénients**
Non guidés (Bus)	-Souplesse dans le choix d'itinéraire. -Absence d'infrastructure dédiée. -Coût d'exploitation peu important.	-Instabilité face à un événement instantané. -Totale dépendance vis-à-vis de la circulation. -Gestion du personnel lourde et difficile.
Guidés (tramway , métro, train)	- Une exploitation en site propre. -Une limitation stricte des mouvements autorisés.	- Dimensions réduites des réseaux. -Des structures de lignes simples. -Fréquence d'exploitation élevée. -Comportement des usagers,ceux-ci arrivent aléatoirement en station.

TABLE 1.1 – Avantages et inconvénients des modes de transport routier.

II- Coût de transport routier :

Le coût du transport est une notion en partie abstraite (pour ce qui concerne par exemple l'internalisation des coûts socio-environnementaux ou les coûts pris en charge par la collectivité). il varie selon les pays, les époques et le mode de transport. Pour faire L'analyse on nécessite de le décomposer en plusieurs variables [6] :

a - Coût direct : le coût est la mesure, de la dépense associé à un événement ou une action de nature économique. Il est exprimé généralement sous forme d'un prix ou d'une valeur monétaire. Le coût direct dans le domaine de transport signifie comme par exemple le prix du ticket, de l'abonnement aux transports urbain. Il varie selon le mode de transport utilisé.

b - Coût d'investissement, de planification d'entretien : C'est une partie importante du coût caché des infrastructures de transport et de leurs fonctionnement. Par exemple : le réseau de transport urbain en site propre nécessite un plan de financement lourd pour l'infrastructure et l'exploitation (des projets sont souvent mis en retard à cause des difficultés d'investissement).

En effet, dans le cas d'infrastructures lourdes (tramway, métro, train, ect), l'investissement est également lourd, par contre dans le cas des bus, la voirie a aussi un coût, mais son investissement et son entretien sont proportionnellement moins coûteux, mais ces coûts augmentent rapidement avec l'étalement urbain et en zone d'habitat dispersé.

c - Coût en temps : Depuis l'apparition de l'automobile, le temps moyen passé dans le transport (véhicule en particulier) tend à augmenter, c'est aussi un aspect important. La fluidité de la circulation est d'ailleurs un des facteurs de ce coût.

La gestion de la voirie et des flux de circulation en cas d'embouteillages est une tâche importante.

d - Coût humain, environnemental : La marche à pied et le vélo sont des facteurs reconnus de bonne santé de l'individu, par l'exercice physique qu'il procure. Inversement, le véhicule est associé à un manque d'exercice physique. Le transport routier génère aussi des blessures, des handicaps, et morts accidentelles sans oublier les nuisances sonores et la pollution chimique de l'environnement.

1.2.4 Impacts environnementaux du transport

Les spécialistes [7] estiment que les impacts de la circulation automobile se mesurent plus particulièrement dans trois domaines :

- la santé publique.
- le milieu naturel.
- le climat.

En matière de santé publique [8], le rejet dans l'atmosphère de matières polluantes en fortes concentrations par les automobiles peut s'avérer nocif, en particulier pour les personnes sensibles. Elles favoriseraient l'apparition d'infections pulmonaires (surtout chez l'enfant), de maladies asthmatiformes, cardio-vasculaires et dans certains cas, faciliteraient

l'apparition de cancers du poumon.
Les impacts [9] sur le milieu naturel sont beaucoup plus visibles car les gaz d'échappement ternissent le mobilier urbain (bancs, lampadaires, etc.), corrodent les façades des immeubles et entraînent à la longue la défoliation des essences arborées et arbustives les plus fragiles. Sous l'effet du ruissellement de l'eau de pluie sur les chaussées souillées, des polluants sont évacués vers les caniveaux, ceux-ci alimentant à leur tour les rivières qui s'épanchent elles-mêmes dans le milieu marin. Dans les zones côtières les plus confinées se forment alors des concentrations de polluants (métaux lourds essentiellement) nocives pour la vie aquatique.
Bien que passées sous silence, les incidences micro climatiques [9] ne doivent pas être ignorées ou minorées. Lors des embouteillages, les gaz d'échappement des véhicules automobiles entraînent une augmentation de la température moyenne de (+2) à (+5) degrés celsius des parties basses de la troposphère, créant des îlots de chaleur accompagnés parfois d'une diminution de la visibilité. Quelle qu'en soit l'incidence sur l'homme, ces facteurs micro climatiques contribuent à accentuer durablement l'effet de serre.

1.3 Modélisation graphique du transport routier

La modélisation [10] reste un moyen très efficace pour représenter la réalité, elle permet de commander, d'analyser et éventuellement d'améliorer les performances des systèmes. En effet, la modélisation s'est vite imposée et devenue indispensable dans toutes les disciplines : automatique, mécanique, économique, etc. Dans le domaine du transport routier, la modélisation est une tâche complexe qui nécessite l'élaboration de modèles appropriés pour assurer la satisfaction de la clientèle, à savoir, proposer un service de transport en tenant compte des contraintes de fonctionnement telles que le respect des horaires théoriques, la garantie des correspondances, la réduction des temps d'attente, etc. Ceci a conduit naturellement les chercheurs à s'intéresser à ce problème et à proposer des modèles adéquats [10].
Un modèle d'un objet ou d'un système [1], est une représentation simplifiée où certains aspects sont privilégiés. Le modèle est destiné à l'analyse scientifique ou à la simulation qualitative ou opérationnelle.

Les graphes sont actuellement l'outil privilégie pour modéliser des ensembles structurés complexes. Leurs applications sont très nombreuses [10] :

- modélisation de l'évolution d'un système dans le temps (en économie, en automatique).
- réseaux divers (électriques, routiers, ou d'adduction d'eau).
- décomposition en tâches d'un projet (en informatique, dans le bâtiment et les travaux publics).

La modélisation à l'aide de la théorie des graphes [10] est souvent utilisée. Cette modélisation consiste à représenter les différents itinéraires possibles qu'un bus peut utiliser. Un réseau de transport urbain est généralement représenté par un graphe, dans lequel l'ensemble des sommets représente les stations, et les arcs représentent les déplacements .

Définition 1 :

Un graphe **G=(X, U)** est déterminé par les données suivantes [10] :

- Un ensemble **X** dont les éléments sont appelés sommets ou nœuds.
- Si **N=card(X)** est le nombre de sommets (de nœuds), on dit que le graphe **G** est d'ordre **N**.
- Un ensemble **U** dont les éléments sont des couples ordonnés de sommets appelés arcs.
- Si **u= (i, j)** est un arc de **G** , **i** est l'extrémité initiale de **u** et **j** est l'extrémité terminale de **u** . on notera **card (U) = M** .

Les stations sont appelées nœuds et les trajets parcourus par un moyen de transport successives sont appelés arcs.

Définition 2 :

Considérons un graphe orienté évalué [10] **G=(X,U,W)** présenté dans la Figure 1.3 .

X désigne un ensemble de **N** sommets (ou nœuds) comme par exemple : **A, B, C**.

U un ensemble de **M** arcs comme l'arc **(AB)** et l'arc **(CG)**.

W(i,j) est la valuation (aussi appelée poids ou coût) de l'arc **(i,j)**, par exemple une distance, un coût de transport, ou un temps de parcours. Dans la figure suivante l'ensemble **X** est égal **{A,B,C,D,E,F,G}**.

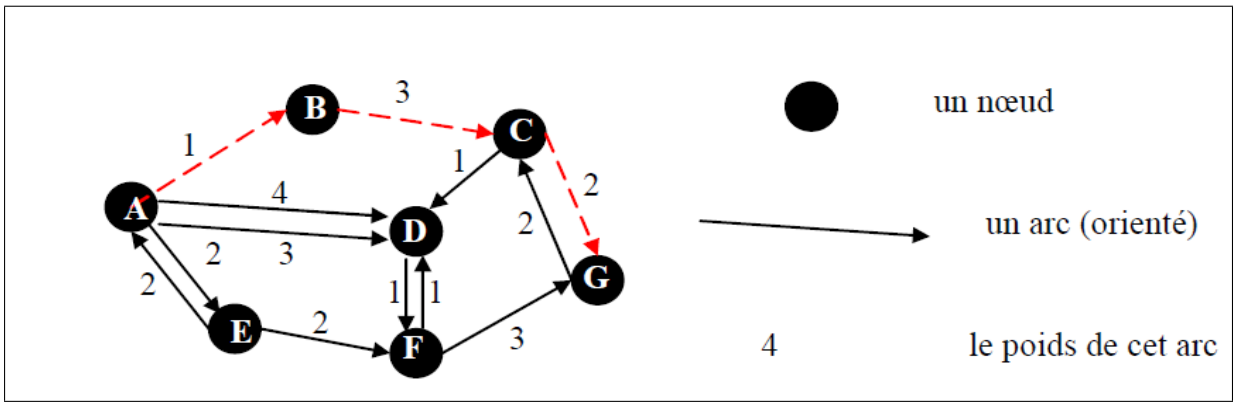

FIGURE 1.2 – Modélisation de réseau de transport.

1.4 L'optimisation combinatoire

L'optimisation [11] est un problème qui peut se poser simplement. Il s'agit de trouver le minimum ou le maximum d'une fonction à plusieurs variables sur un certain domaine de définition, de l'étude de leur existence à leur détermination, en général par la mise en œuvre d'un algorithme et par suite d'un programme.

Notre quotidien fourmille de questions d'optimisation : quel est le trajet le plus rapide ? quel est le moyen de transport le plus économique ?.

Optimiser, c'est chercher à améliorer les systèmes en visant la meilleure solution.

Un problème d'optimisation combinatoire [12] consiste à trouver dans un ensemble discret un parmi les meilleures solutions réalisables, la notion meilleure solution étant définie par une fonction objectif.

En pratique, l'optimisation mathématique [11] consiste à trouver la meilleure solution à un problème qu'on a préalablement représenté par un modèle qui fait intervenir un ou plusieurs objectifs. Ces objectifs sont exprimés sous la forme d'une fonction mathématique, appelée souvent fonction objectif, qui peut être soumise à des contraintes. La solution optimale correspond à une valeur extrême, appelée aussi extremum maximum ou minimum de cette fonction objectif. Le but d'un problème d'optimisation est d'obtenir une solution maximisant ou minimisant une fonction objectif donnée.

Dans sa forme la plus générale [12], un problème d'optimisation combinatoire consiste à trouver dans un ensemble discret un parmi les meilleurs sous-ensembles (ou solutions) réalisables, la notion de meilleure solution étant définie par une fonction objectif. Formellement, étant donnés :

- Un ensemble discret N.

- Une fonction d'ensemble $f : 2^N \rightarrow R$, dite fonction objectif.
- Un ensemble $\Re$ de sous-ensembles de N, dont les éléments sont appelés les **solutions réalisables**.

un problème d'optimisation combinatoire consiste à déterminer :

$$\textbf{MAX ou MIN}_{S \subseteq N} \{f(S) : S \in \Re\}$$

Dans le transport par exemple, on s'en sert pour **minimiser** les coûts de voyage (temps, frais, pollution).

1.4.1 Classification des problèmes d'optimisation

Indépendamment du degré de difficulté du problème d'optimisation, sa résolution nécessite tout d'abord sa classification par rapport aux problèmes existants. Cette classification n'est pas seulement un moyen pour montrer l'importance du problème mais également, une façon de faciliter sa résolution en le traitant par analogie par exemple par rapport à un autre problème existant. Le domaine de transport est regorge des différents types de problèmes qui ont été groupés en deux classes principales [12] :

i) La première classe représente les **problèmes de décision** qui sont caractérisés par une solution réduite à une simple réponse (oui) ou (non) .

ii) Par contre, la deuxième classe concerne les **problèmes d'optimisation** possédant une solution admissible qui doit être construite et qui permet de minimiser ou maximiser la fonction objectif. Un type particulier de problèmes n'ayant aucune méthode de résolution et qui sont les plus difficiles sont les problèmes indécidables.

Malgré que cette classification [12] montre une indépendance entre ces deux types de problèmes, il existe un lien très fort entre les problèmes de décision et les problèmes d'optimisation. La résolution d'un problème d'optimisation dont l'objectif est de minimiser ou maximiser la valeur de la fonction objectif, nécessite indirectement la résolution d'un problème de décision qui consiste à étudier l'existence ou non d'une solution admissible optimale.

D'une manière réciproque, l'étude de la complexité d'un problème de décision permet de donner les indications relatives aux problèmes d'optimisation associés D'une manière générale, un problème quelconque qu'il soit de décision ou d'optimisation doit appartenir à l'une de ces trois classes suivantes [121] :

I- Problème de classe P : Les problèmes appartenant à la classe P sont ceux dont le problème de décision correspondant donne la réponse correcte (oui) ou (non) et qui peut être résolue avec un algorithme nommé facile ou de classe P.

II- Problème de classe NP : Le problème de cette classe ne peut pas être résolu dans un temps polynomial, il est dit NP difficile.

Les méthodes de résolution utilisées sont les heuristiques qui permettent de trouver une solution optimale mais non démontrables. Les problèmes de classe P peuvent être résolus par les algorithmes ordinaires ou exacts qui sont inclus dans la famille des heuristiques, par la suite la classe P est inclus dans la classe NP.

III- Problème de classe NP-Complet : Problème de classe NP-Complet [11] est un problème complexe et difficile à résoudre, il représente les problème d'industrie de nos jour.

Il existe différentes méthodes de résolution résoudre les problèmes de la classe NP-Complet : les méthodes approchées (heuristiques ou métaheuristiques) et les méthodes exactes [12]. Les méthodes exactes réussissent à trouver la solution optimale pour des problèmes de petite taille. Cela dit, au fur et à mesure que la taille des problèmes augmente, l'obtention de la solution optimale nécessite un temps exponentiel et ces méthodes deviennent impraticables. On applique alors des méthodes approchées qui donnent, en un temps raisonnable, une solution proche de l'optimum.

Notre problème appartient aux problèmes de classe NP-Complet. Nous allons par la suite résoudre notre problème d'abord en utilisant l'une des méthodes exactes, ensuite nous allons le résoudre à l'aide des méthodes approchées zt enfin en hybridant la méthode exacte avec les méthodes métaheuristiques.

1.4.2 La complexité d'un algorithme

La complexité d'un algorithme [12] est le nombre d'opérations (boucles, itérations) effectué par l'algorithme.

Pour effectuer un choix pertinent de l'algorithme de résolution [11], il est donc important de bien identifier à quelle catégorie le problème d'optimisation appartient.

On quantifie la complexité d'une question en mesurant sa réponse. Autrement dit, pour

voir si un problème est compliqué, on s'intéresse aux algorithmes qui peuvent le résoudre. La question de la complexité algorithmique nécessite une certaine étude. Il faut d'abord s'intéresser à l'efficacité des algorithmes ainsi qu'à la difficulté des problèmes traités par ces algorithmes [12].

- Complexité en temps :

La complexité en temps [11] représente le temps pris par un algorithme pour retourner le résultat (temps d'exécution).

Un algorithme **A** [10] étant une suite d'actions élémentaires à effectuer, on compte donc, indépendamment du processus, pour un problème à $\boldsymbol{n}$ entrées, les opérations élémentaires nécessaires pour terminer le calcul. On obtient d'abord $\mathbf{t\ (A,}\boldsymbol{i}\mathbf{\ )}$, le temps d'exécution de l'algorithme **A** pour l'instance $\boldsymbol{i}$. Ensuite, on regroupe les instances selon leur taille afin d'obtenir une mesure qui ne dépend pas d'une instance particulière. C'est ce qu'on nomme **la complexité en temps**, $\mathbf{C(A,}\boldsymbol{n}\mathbf{)}$, dans le pire des cas pour les entrées de taille inférieure à $\boldsymbol{n}$.

C'est évidemment une fonction croissante avec $\boldsymbol{n}$. Cette complexité en temps reste une notion très abstraite. Cependant, on sait que l'ordinateur dont on dispose est capable d'effectuer une opération en un temps maximum **K**, il suffit de multiplier la fonction $\mathbf{C(A,}\boldsymbol{n}\mathbf{)}$ par **K** pour obtenir le temps maximal requis .

Dans le chapitre 3, nous allons présenter des exemples détaillés de calcul de complexité appliqué sur des exemples de notre application.

1.4.3 Problème du plus court chemin multi-objectifs

Le problème du plus court chemin [12] représente la base de tous les problèmes de planification d'itinéraire. Le plus court chemin d'un point de départ à un point d'arrivée minimisant soit la distance, soit un autre critère similaire comme par exemple le temps . Plusieurs algorithmes ont été mis au point : l'algorithme de Bellman-Ford, l'algorithme de Johnson, et le plus connu, l'algorithme de Dijkstra Il est courant, lorsque l'on cherche à se rendre d'un point à un autre dans un réseau, de chercher le plus court chemin, c'est-à-dire, celui dont la longueur est la plus petite. Si le nombre de trajets possibles entre le point de départ et le point d'arrivée est faible, il suffira de calculer les longueurs de chacun des trajets en additionnant la longueur des liens qui le composent et de comparer

directement les longueurs obtenues Mais une telle solution exhaustive devient rapidement impraticable si le nombre de trajets possibles est grand .

Aujourd'hui, dans l'évaluation des routes optimales [11], la principale préoccupation, outre la sécurité, est généralement accordée aux temps de navigation (ou vitesse de navigation) plutôt que de la distance. En considération des critères de la sécurité et de la consommation d'énergie, la plus courte distance entre deux ports n'est pas nécessairement une route optimale. Si il ya de nombreuses routes à choisir entre elles, la route qui prend le moins de temps est la route optimale.

La notion de distance peut être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense (problème multi-objectif) dans tous les cas, on parle de coût [12] .

1.5 Conclusion

L'homme cherche à améliorer sa vie quotidienne, il essaye à minimiser ses charges, son loyer ou la consommation de sa voiture. Il tente toujours à vrai dire à optimiser que se soit minimiser ses dépenses et maximiser ses biens. Les chercheurs viennent pour concrétiser le vœux de l'homme en modélisant les problèmes de la vie sous des fonctions coûts en utilisant les différents types d'optimisation.

Nous avons présenté dans ce chapitre un des domaines indispensables dans la vie humaine, qui est le transport routier ainsi que quelques notions fondamentales de l'optimisation combinatoire.

Dans le prochain chapitre, nous allons présenter quelques méthodes de résolution permettant d'optimiser les coûts du problème de transport.

Chapitre 2

Métaheuristiques et hybridation

2.1 Introduction

Dans la vie courante, nous sommes souvent confrontés à des problèmes qui peuvent être décrits sous forme d'un problème d'optimisation [1], comme le fait de réduire le coût et le temps de la circulation par minimiser la distance parcourue.
Avec le développement de notre société, les problèmes d'optimisation sont devenus plus complexes et difficiles à résoudre par les méthodes classiques. Pour cela, on fait appel aux méthodes approchées [2] qui parviennent à trouver des solutions de bonne qualité, parfois même optimales, dans des délais raisonnables. Ceci est obtenu en utilisant les méthodes regroupant les heuristiques et les métaheuristiques [13].
Dans ce chapitre nous allons présenter les méthodes de résolution d'un problème d'optimisation combinatoire qui incluent les heuristiques, les métaheuristiques et les méthodes hybrides qui nous allons détailler leurs classifications ainsi que les différentes méthodes d'hybridations.

2.2 Méthodes de résolution d'un problème d'optimisation combinatoire

Dans le monde réel, en particulier dans le domaine industriel tel que la mécanique, la chimie, la télécommunication, l'environnement, le transport, etc, les problèmes sont

1. Action d'obtenir le meilleur, d'améliorer un fonctionnement, un rendement, une utilisation.
2. Basée sur le voisinage, elle consiste à trouver une solution proche de l'optimalité.

complexes et de grandes tailles, ce qui les rendent difficiles à résoudre.

L'optimisation [13], qui représente une partie très importante de la recherche opérationnelle [3] s'occupe de ce type de problèmes. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-Complet et ne possèdent donc pas à ce jour de solutions algorithmiques efficaces et valables pour toutes les données.

Les chercheurs [13] ont divisé ce type de problème en deux groupes : problème mono-objectif et problème multi-objectif.

Les problèmes mono-objectifs [12] existent rarement dans les applications réelles par contre les problèmes multi-objectifs caractérisés par des critères généralement contradictoires et qui doivent être satisfaits simultanément, représentent une majorité des situations réelles. Les problèmes multiobjectifs ont la particularité d'être beaucoup plus difficiles à traiter que leur équivalent mono-objectif. La difficulté réside dans l'absence d'une relation d'ordre total entre les solutions.

Une solution peut être meilleure qu'une autre sur certains objectifs et moins bonne sur les autres. Donc il n'existe généralement pas une solution unique qui procure simultanément la solution optimale pour l'ensemble des objectifs. Voilà pourquoi le concept de solution optimale devient moins pertinent en optimisation multiobjectif. Dans ce cas la solution optimale ou de bonne qualité n'est plus une solution unique mais, un ensemble de solutions compromis entre les différents objectifs à optimiser [13].

Exemple :

dans le cas de deux objectifs à minimiser, toute amélioration de l'un des objectifs se fait au détriment de l'autre et que la solution optimale ou proche de l'optimum est un compromis entre les deux .

Dans l'achat d'une voiture d'occasion, la voiture idéale est celle qui est peu chère (critère économique) avec peu de kilomètres (critère qualitatif), il n'est pas évident de pouvoir regrouper en un seul objectif ces deux critères non commensurables. Ainsi il n'existe plus une solution optimale unique mais un ensemble de solutions. Nous allons donc devoir identifier les meilleurs compromis possibles suivant notre budget.

Les différentes méthodes de résolution du problème d'optimisation multiobjectifs sont illustrés dans la figure ci dessous :

3. Une discipline qui permet fournir des bases rationnelles à la prise de décisions

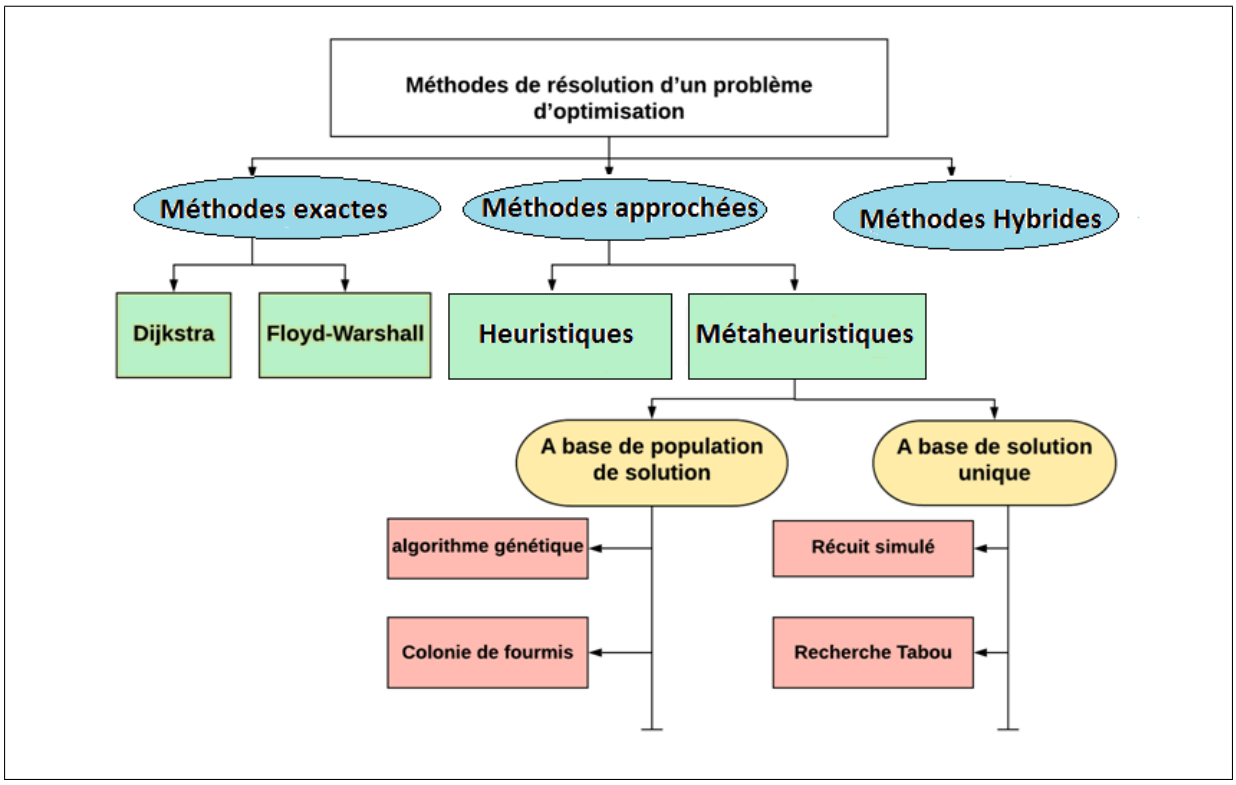

FIGURE 2.1 – Les méthodes de résolution d'un problème d'optimisation.

le choix de la méthode de résolution à mettre en œuvre dépendra souvent de la complexité du problème. En effet, suivant sa complexité, le problème pourra ou non être résolu de façon optimale.

2.3 Méthodes exactes

Les méthodes exactes [14] sont dédiées à résoudre optimalement les problèmes de petites instances (problèmes classés dans la classe P), elles cherchent à trouver de manière certaine la solution optimale en examinant de manière explicite ou implicite la totalité de l'espace de recherche. Elles ont l'avantage de garantir la solution optimale néanmoins le temps de calcul nécessaire pour atteindre cette solution peut devenir très excessif en fonction de la taille du problème (explosion combinatoire) et le nombre d'objectifs à optimiser Ce qui limite l'utilisation de ce type de méthode aux problèmes bi-objectifs de petites tailles.

Les algorithmes exacts [15] garantissent de trouver la solution optimale d'un problème d'optimisation combinatoire, mais celle-ci demeure parfois introuvable, faute de temps.

De façon pratique, seuls les problèmes de petites et moyennes tailles peuvent être résolus de

façon optimale par des algorithmes exacts. De plus, pour certains problèmes la consommation de mémoire de ces algorithmes peut être très grande et peut parfois entraîner l'arrêt prématuré de l'application informatique.
parmi les algorithmes appartenant à la classe des méthodes exactes, on peut citer :

2.3.1 Algorithme de Dijkstra

L'Algorithme de Dijkstra [10] permet de calculer le plus court chemin entre deux sommets d'un graphe connexe non orientés.
On construit tous les chemins optimaux progressivement : on part du point de départ et on regarde les chemins de taille 1 (une arête). On conserve le chemin de longueur minimal car celui-ci ne pourra être amélioré par la suite. On construit ensuite des débuts de chemin et on conserve à chaque fois le plus court d'entre eux (car ceux là ne pourront être améliorés). À chaque étape, on traite définitivement un nouveau sommet, qui est marqué. Une fois tous les sommets marqués, on a accès à la longueur des plus courts chemins issus du point de départ, ainsi qu'aux chemins eux-mêmes.

L'algorithme de Dijkstra garantit de trouver le plus court chemin du point de départ vers le point d'arrivée, tant qu'aucun des bords n'a un coût négatif [16].
L'utilisation de l'algorithme des graphes [10] pour le calcul d'un chemin nécessite l'utilisation d'une discrétisation de l'espace car un noeud du graphe est assimilé à un point de l'environnement. Les arcs représentent alors une notion d'accessibilité entre deux points et sont values par une estimation de l'effort à fournir pour relier ces points (il s'agit le plus souvent de la distance). La recherche d'un chemin va donc se résumer à la recherche d'une suite de noueds, reliés par des arcs dont la somme des valeurs associées minimise le coût global du chemin.
Nous avons choisi de travailler avec cet algorithme pour trouver le plus court chemin après avoir fait une étude comparative présentée dans le chapitre 3 entre l'algorithme de Dijkstra et celui de Floyd-Warshall.

2.3.2 Algorithme de Floyd-Warshall

Floyd-Warshall est un algorithme utilisé pour déterminer les distances des plus courts chemins entre toutes les paires de sommets dans un graphe orienté et pondéré, en temps cubique en le nombre de sommets.

Il prend en entrée un graphe orienté et évaluée, décrit par une matrice d'adjacence donnant le poids d'un arc lorsqu'il existe . Le poids d'un chemin entre deux sommets est la somme des poids sur les arcs constituant ce chemin. Les arcs du graphe peuvent avoir des poids négatifs, mais le graphe ne doit pas posséder de circuit de poids strictement négatif. L'algorithme calcule, pour chaque paire de sommets, le poids minimal parmi tous les chemins entre ces deux sommets [10].

2.4 Méthodes approchées

Les problèmes d'optimisation combinatoire sont souvent des problèmes très difficiles dont la résolution par des méthodes exactes peut s'avérer très longue ou peu réaliste. L'utilisation de méthodes heuristiques permet d'obtenir des solutions de bonne qualité en un temps de résolution raisonnable [14].

2.4.1 Les heuristiques

Une heuristique [17] est un algorithme approché qui permet d'identifier en temps polynomial [4] au moins une solution réalisable rapide, pas obligatoirement optimale.
L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Elle permet de guider les choix que doit faire un algorithme pour réduire sa complexité, généralement une heuristique est conçue pour un problème particulier en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée.
Elles peuvent être classées en deux catégories [18] :

- **Méthodes constructives :** qui génèrent des solutions à partir d'une solution initiale, en essayant d'ajouter des éléments petit à petit jusqu'à ce qu'une solution complète soit obtenue.
- **Méthodes de recherche locale :** qui démarrent d'une solution initialement complète (Probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage.

4. Un algorithme est efficace si sa complexité en temps est polynomiale, c'est-à-dire en O (n^k) pour un entier k.

2.4.2 Les métaheuristiques

Apparues dans les années 1980 [14], les métaheuristiques sont inspirées de mécanismes d'optimisation rencontrés dans la nature, elles forment un ensemble d'algorithmes permettant de trouver la solution la plus rapide et la plus efficace pour une large gamme de problèmes d'optimisation difficile et pour lesquels on ne connaît pas de méthode classique plus efficace. Elles parviennent à trouver des solutions de bonne qualité, parfois même optimales, dans des délais raisonnables permettant d'approximer les meilleures solutions sur les plus grandes instances (problèmes NP-Complet).

I- Les propriétés fondamentales [17] :

- Elles sont des approches coopératives qui permettent à différentes méthodes d'optimisation de combiner leurs avantages dans le but d'améliorer les performances globales afin d'obtenir de bon résultats.
- Ce sont des stratégies permettant de guider la recherche et d'explorer son espace efficacement afin d'obtenir une solution proche de la solution optimale.
- Les algorithmes peuvent être de simple procédure de recherche locale ou des processus d'apprentissage complexes.

II- Objectifs :

L'objectif est de concevoir des algorithmes de plus en plus performants [17] :

- En temps de calculs (raisonnable).
- En qualité de solutions produites (bonne qualité).
- Permettant de traiter des problèmes de plus grande taille.

III- Avantages [18] :

- Elle est simple à utiliser puisqu'elle est basée sur un principe simple qui nous permet de changer systématiquement de voisinage lorsqu'on se retrouve bloqué dans un minimum local.
- Étant très généraliste, elle s'applique à un grand nombre de problèmes d'optimisation combinatoire.
- Elle est facile à utiliser : les différentes étapes de l'algorithme sont faciles à comprendre et à mettre en œuvre.
- Elle est efficace : les meilleures solutions sont obtenues en un temps de calcul

modéré.

– Le fait d'utiliser plusieurs voisinages permet de diversifier l'exploration de l'espace de solution afin d'accéder à un plus grand nombre de régions intéressantes, ce qui conduit à une méthode plus robuste que le recuit simulé ou la recherche tabou.

Il existe deux catégories des métaheuristiques illustrées dans la Figure 2.2, elles seront détaillées dans la partie ci dessus.

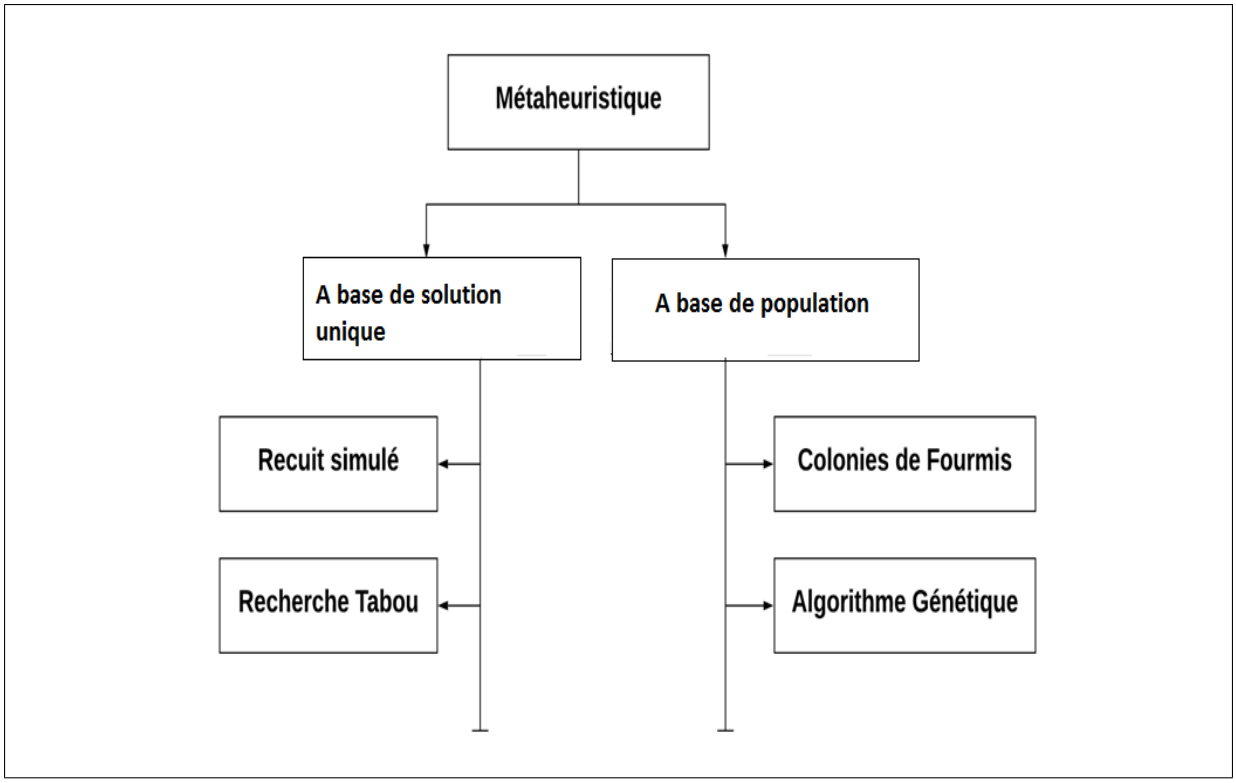

FIGURE 2.2 – Catégories des métaheuristiques.

2.4.3 Le recuit simulé

Il appartient aux métaheuristiques a base de solution unique. Le recuit simulé [19] a été Introduit par Kirkpatrick en 1982. Son principe de fonctionnement repose sur une imitation du phénomène de recuit en science des matériaux basé sur les principes d'équilibre énergétique lors de la cristallisation des métaux. L'analogie avec une méthode d'optimisation est trouvée en associant une solution à un état du métal, on équilibre thermodynamique est la valeur de la fonction objectif de cette solution.

Passer d'un état du métal à un autre correspond à passer d'une solution à une solution voisine. Le recuit simulé est une technique stochastique pour la résolution approchée

de problèmes NP-Complet de l'optimisation combinatoire basé sur un paramètre appelé température, qui sera ajusté au cours de la recherche. A partir d'une solution initiale il recherche dans son voisinage une autre solution de façon aléatoire qui peut être de moins bonne qualité permettant d'échapper aux optima locaux en acceptant temporairement une dégradation de la fonction objectif. Initialement, la température est élevée autorisant une forte dégradation de qualité puis décroît pour diminuer la probabilité des dégradations importante.

Pour comprendre l'algorithme [20], prenons l'exemple d'un problème informatique caractéristique, celui du voyageur de commerce. Un voyageur de commerce veut se rendre dans un grand nombre de villes le plus rapidement possible, en passant une fois et une seule par chaque ville. Il s'agit donc de déterminer le meilleur ordre dans lequel il doit visiter ces villes, de sorte que le chemin parcouru soit le plus court possible. Étonnamment, il s'agit d'un problème très difficile. Peut-être trouverez-vous la meilleure solution en utilisant le recuit simulé. Il est facile de trouver une solution quelconque : prenons un ordre au hasard ! Dans la seconde applet, nous voyons un certain nombre de villes reliées par un parcours aléatoire.

Il est bien sûr peu probable que ce chemin soit très court. Comment pouvons-nous améliorer ce parcours ? On peut le faire par petites touches en apportant des modifications locales, et si le parcours obtenu est meilleur, nous conservons la modification, sinon nous l'annulons. Une modification locale possible est d'inverser l'ordre des villes sur une partie du parcours choisie au hasard Itérer des modifications locales pour ne conserver que celles qui améliorent le parcours, c'est exactement ce que réalise l'applet suivante pour les températures très basses. En laissant le nombre de villes à 50, abaissez la température à 0, puis appuyez sur le bouton de démarrage. Vous verrez que la solution s'améliore continuellement puis se stabilise. Étant donné que la température est basse, seules des modifications améliorant le parcours seront retenues, et il arrive un moment où une telle modification devient impossible à trouver. Mais plus la température est élevée, plus des modifications qui dégradent la solution pourront être retenues. C'est essentiel pour échapper à une solution qui ne serait pas la meilleure, mais ne pourrait toutefois pas être améliorée localement par de simples substitutions. Notez la longueur du parcours calculé et élevez à nouveau la température à 100. Le parcours se modifie à nouveau et vous pouvez alors baisser la température très lentement. Si vous avez suffisamment de

patience, vous devriez obtenir une meilleure solution.
Ce procédé d'optimisation simule un processus naturel et ne s'applique pas seulement au problème du voyageur de commerce. Il est également utilisé pour résoudre de nombreux problèmes d'optimisation délicats, pour lesquels il est possible de trouver facilement une solution quelconque et de l'améliorer localement. Ces méthodes, qui fonctionnent souvent très bien, mais n'offrent aucune garantie d'obtenir la solution optimale ou ne serait-ce qu'une bonne solution, sont appelées heuristiques. Le recuit simulé est un procédé inspiré par la nature [20].

2.4.4 La recherche tabou

La recherche tabou [20] est une méthode d'optimisation mathématique, appartenant à la classe des techniques de recherche locale (métaheuristiques a base de solution unique). La recherche tabou améliore les performances d'une méthode de recherche locale en utilisant des structures adaptées de mémoire : une fois une solution potentielle a été déterminée, elle est marquée comme taboue afin que l'algorithme ne visite pas cette possibilité à plusieurs reprises. Pour explorer les régions de l'espace de recherche qui seraient laissées inexplorées par la procédure de recherche locale, la recherche tabou modifie la structure du voisinage de chaque solution au fur et à mesure que la recherche progresse. Les solutions admises sont déterminées par l'utilisation de structure de mémoire.
Dans sa forme la plus simple, une liste tabou [20] est une mémoire à court terme qui contient les solutions qui ont été visitées dans le passé récent. C'est le type le plus important de la structure de la mémoire utilisée pour déterminer les solutions admises pour la liste tabou.
La recherche tabou présente l'avantage de posséder une mémoire sous la forme de la liste tabou. Cette mémoire peut être à court terme si on mémorise la solution telle qu'elle est et à long terme si on mémorise la fréquence d'apparition au lieu d'interdire complètement le mouvement introduisant cette solution. Ainsi, les paramètres à fixer sont moins nombreux. Il s'agit de la taille de la liste tabou qui peut être statique déterminée à l'avance et constante durant tout le processus de résolution ou dynamique variante durant la résolution et du nombre total d'itérations. Cependant, dans plusieurs cas d'optimisation, la recherche tabou demande de lui ajouter les mécanismes d'intensification et de diversification qui introduisent de nouveaux paramètres à régler [20].

2.4.5 Les colonies de fourmis

L'algorithme des colonies de fourmis [21] sont des algorithmes non déterministes qui utilisent la nature comme modèle. Elles sont des métaheuristiques a base de population, cette méthode est basée sur le comportement des colonies de fourmis dans la nature. Des biologistes ont observé que les fourmis sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid. Cette faculté trouve son origine dans les phéromones, substance chimique sécrétée par les insectes, qui permet aux fourmis de s'échanger des informations.
La règle de déplacement, appelée « règle aléatoire de transition proportionnelle » est écrite mathématiquement sous la forme suivante : Une des villes est désignée comme four-

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in J_i^k} \tau_{il}(t)^{\alpha} \cdot \eta_{il}^{\beta}} & \text{si } j \in J_i^k \\ 0 & \text{si } j \notin J_i^k \end{cases}$$

milière et une première fourmi en sort pour prendre un chemin aléatoire et revient à la fourmilière. Sur le chemin, elle a laissé une trace de phéromone sur chaque arête. Une deuxième fourmi sort à son tour de la fourmilière et effectue un autre tour, en déposant elle aussi, des phéromones en chemin. Les fourmis privilégient le chemin ayant une forte concentration en phéromones. Plus l'arête est empruntée, plus les autres fourmis ont tendance à la parcourir. Or, les phéromones se dissipent avec le temps. Plus une arête est longue, plus l'évaporation de phéromones est élevée et moins les fourmis circulent sur l'arête. Cette tendance permet d'éliminer du circuit les arêtes les plus longues. Le programme fixe donc un indice d'évaporation des phéromones afin que le circuit puisse évoluer en mieux. Il fait en sorte que seuls les meilleurs chemins soient conservés [10].

Le premier algorithme de colonies de fourmis proposé est appelé le système fourmi.il vise notamment à résoudre le problème du voyageur de commerce, où le but est de trouver le plus court chemin permettant de relier un ensemble de villes.
L'algorithme général est relativement simple, et repose sur un ensemble de fourmis, chacune parcourant un trajet parmi ceux possibles.

À chaque étape, la fourmi choisit de passer d'une ville à une autre en fonction de quelques règles [21] :

- Elle ne peut visiter qu'une fois chaque ville .
- Plus une ville est loin, moins elle a de chance d'être choisie.
- Plus l'intensité de la piste de phéromones disposée sur l'arête entre deux villes est grande, plus le trajet aura de chance d'être choisi .
- Une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones si le trajet est court .
- Les pistes de phéromones s'évaporent à chaque itération. Le partage des données sur les phéromones est le point fort de cette technique [21] .

2.4.6 Les algorithmes génétiques

Les algorithmes génétiques [22] sont des algorithmes itératifs (basés sur la répétition) et stochastiques (en référence au hasard), Ils reposent sur les principes du néodarwinisme [5] (fondé par Charles Darwin [6]) : la sélection naturelle et la recombinaison génétique. D'une part, la sélection naturelle, selon Charles Darwin, énonce que les individus les mieux adaptés sont plus aptes à survivre et à devenir parents de la prochaine génération. D'autre part, Gregor Mendel, père de la génétique, explique que des mutations au niveau des gènes permettent l'évolution des espèces. Ainsi, les mutations les plus adaptées à l'environnement sont les plus transmises et grâce à cet héritage génétique, les espèces actuelles deviennent des "versions optimisées" de leurs ancêtres.

Les phénomènes biologiques [22] ont été une grande source d'inspiration pour les informaticiens. Le parfait exemple est donné par les algorithmes génétiques. Ceux-ci ont été initiés dans les années 1970 par le psychologue et scientifique John Holland. Ils imitent au sein d'un programme les mécanismes d'évolution dans la nature : croisement, mutation, sélection. En effet, dans un algorithme génétique, on part d'une population de solutions potentielles au problème, initialement choisies aléatoirement. Les solutions sont ensuite évaluées afin de déterminer leur capacité de survie et de désigner celles qui transmettront leurs gènes à la génération suivante. Les solutions changent donc constamment, les plus

5. une théorie explique l'évolution des organismes au cours du temps.

6. Charles Darwin est un naturaliste anglais ayant révolutionné la biologie par ses travaux sur l'évolution des espèces.

adaptées survivent et se reproduisent, les autres disparaissent. On recommence ce cycle jusqu'à obtenir une solution satisfaisante au problème.

- Principe [21] :

- Les algorithmes génétiques tentent de simuler le processus d'évolution des humains. Ainsi, on retrouve les termes : sélection, croisement et mutation qui sont les opérations de base des algorithmes génétiques. Elles s'appuient dans leur fonctionnement sur les processus naturels.

- Les algorithmes génétiques partent d'une population de base, qui est améliorée au fil des générations par l'introduction de nouveaux éléments dans la population.

L'opération de sélection choisit dans la population un ensemble de couples d'individus selon un certain critère, pour être utilisés comme parents afin de produire de nouveaux individus.

- Le croisement combine les parents choisis par l'opération de sélection, pour générer un ou plusieurs enfants.

- Enfin, la mutation apporte un facteur de diversification à la population, en modifiant les enfants générés par le croisement, avec une certaine probabilité généralement très petite, vu qu'un tel phénomène est rare en milieu naturel (selon la théorie de Darwin).

La qualité d'une solution est donnée par la valeur de la fonction de fitness. Celle-ci est généralement une fonction de gain associée à la solution. La solution est codée sous forme de chromosome, cette codification joue un rôle prépondérant dans l'efficacité de l'algorithme, car l'application des opérateurs génétiques peut changer selon le codage retenu.

La figure ci dessous présente le principe de fonctionnement de l'algorithme génétique.

2.5 Méthodes hybrides

L'hybridation [23] est une tendance observée dans de nombreux travaux réalisés sur les métaheuristiques ces dix dernières années. L'hybridation consiste à combiner les caractéristiques de plusieurs méthodes différentes pour tirer leurs avantages .

Actuellement, les métaheuristiques hybrides sont devenues plus populaires car les meilleurs résultats trouvés pour plusieurs problèmes d'optimisation combinatoires ont étés obtenus

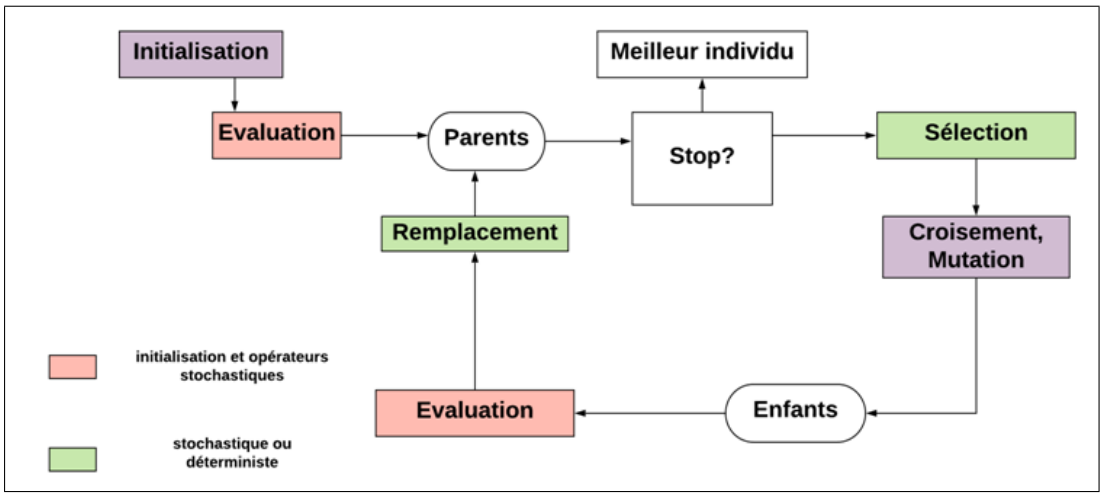

FIGURE 2.3 – Principe de l'algorithme génétique .

avec des algorithmes hybrides.

L'hybridation des métaheuristiques peut être divisée en deux grandes parties [22] :

- hybridation des métaheuristiques avec des métaheuristiques.
- hybridation des métaheuristiques avec des méthodes exactes.

2.6 Hybridation métaheuristiques/métaheuristiques

Selon la taxonomie [24] proposée dans l'hybridation des métaheuristiques entre elles, l'hybridation se fait en deux classifications principales :

- Une classification hiérarchique .
- Une classification à plat.

Cette classification [24] est caractérisée par le niveau et le mode de l'hybridation.

Le niveau d'hybridation peut être :

- bas (Low-Level) .
- haut (High-Level).

Dans le niveau bas, une métaheuristique remplace un opérateur d'une autre méthode qui l'englobe. Par contre, dans le niveau haut de l'hybridation, chaque métaheuristique garde sa propriété au cours de l'hybridation.

A. Bas niveau :

Dans le niveau bas [24], une métaheuristique remplace un opérateur d'une autre méthode qui l'englobe. La combinaison des modes et des niveaux donne deux classes d'hybridation qui sont l'hybridation relais de bas niveau, l'hybridation co-évolutionnaire de bas niveau.

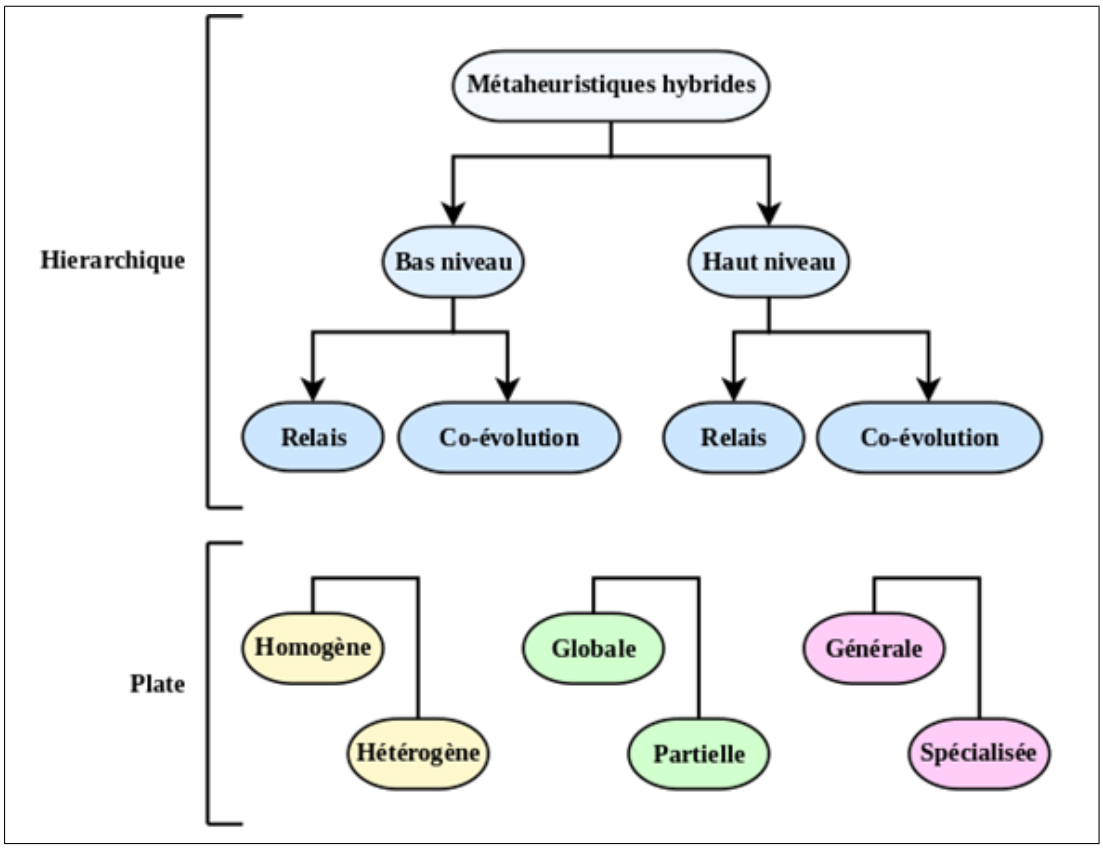

FIGURE 2.4 – La Taxonomie de l'hybridation.

A.1 L'hybridation relais de bas niveau :

Elle englobe les métaheuristiques à base de solution unique dans lesquelles une autre méthode est incorporée pour former un nouvel algorithme. Pour résoudre le problème du voyageur de commerce et le problème de la partition de graphe [24].

A.2. L'hybridation co-évolutionnaire de bas niveau :

Consiste à incorporer une ou plusieurs métaheuristiques à base de solution unique dans une métaheuristique à population de solutions. L'avantage de ce type d'hybridation est de compenser la puissance d'exploitation d'une recherche locale et celle d'exploration d'une recherche globale [24].

B. Haut niveau :

La combinaison des modes et des niveaux donne deux classes d'hybridation qui sont l'hybridation relais de haut niveau, l'hybridation co-évolutionnaire de haut niveau [24].

B.1 L'hybridation relais de haut niveau :

Elle a lieu lorsque les métaheuristiques sont utilisées de manière séquentielle c'est-à-dire la (ou les) solution(s) finale(s) de la première métaheuristique est la (ou les)

solution (s) initiale(s) de la métaheuristique suivante. Dans cette procédure, toutes les méthodes gardent leur intégrité [25].

B.2 hybridation co-évolutionnaire de haut niveau :

Les métaheuristiques utilisées travaillent en parallèle en échangeant des informations entre elles afin de trouver la solution optimale du problème posé. Pour cette hybridation, la population est divisée en sous-populations réparties sur les sommets d'un hypercube dont lesquels un algorithme génétique est lancé. Chaque sommet correspond à une zone de recherche de solutions et périodiquement, des individus migrent entre les sommets en contribuant à trouver les solutions optimales [25].

C. La classification à plat des métaheuristiques :

Elle est caractérisée par le type des méthodes hybridées, leur domaine d'application et la nature de leurs fonctions. Selon le type d'hybridation, on trouve des méthodes hybridées homogènes où les algorithmes utilisés se basent sur la même métaheuristique comme le modèle insulaire et des méthodes hybridées hétérogènes où les métaheuristiques utilisées sont différentes. L'hybridation globale a lieu lorsque toutes les méthodes hybridées sont appliquées à la totalité de l'espace de recherche. Toutes les méthodes que nous avons vus précédemment sont des hybridations globales. A l'opposé, l'hybridation partielle décompose un problème en sous-problèmes où chacun a son propre espace de recherche [26].

2.6.1 Classification hiérarchique des métaheuristiques

La classification à plat des métaheuristiques est caractérisée par le type des méthodes hybridées, leur domaine d'application et la nature de leurs fonctions. Selon le type d'hybridation, on trouve des méthodes hybridées homogènes où les algorithmes utilisés se basent sur la même métaheuristique comme le modèle insulaire et des méthodes hybridées hétérogènes où les métaheuristiques utilisées sont différentes.

Le modèle proposé dans est une hybridation de haut niveau co-évolutionnaire hétérogène.

Le domaine d'application des métaheuristiques hybridées permet de distinguer deux grandes classes d'hybridation, les hybridations globales et les hybridations partielles.

L'hybridation globale a lieu lorsque toutes les méthodes hybridées sont appliquées à la totalité de l'espace de recherche. Toutes les méthodes que nous avons étudiées précédemment sont des hybridations globales.

A l'opposé, l'hybridation partielle décompose un problème en sous-problèmes où chacun a son propre espace de recherche [23].

2.6.2 L'hybridation relais de bas niveau

L'hybridation relais de bas niveau englobe les métaheuristiques à base de solution unique dans lesquelles une autre méthode est incorporée pour former un nouvel algorithme.

2.6.3 L'hybridation relais de haut niveau

L'hybridation relais de haut niveau [24] a lieu lorsque les métaheuristiques sont utilisées de manière séquentielle c'est-à-dire la (ou les) solution(s) finale(s) de la première métaheuristique est la (ou les) solution(s) initiale(s) de la métaheuristique suivante.Dans cette procédure, toutes les méthodes gardent leur intégrité.
C'est la méthode que nous avons utilisé dans notre recherche.

2.7 Hybridation métaheuristiques/méthodes exactes

Il est possible d'hybrider de plusieurs façons une méthode exacte avec une métaheuristique. Cette classification proposée permet de différencier la plupart des formes d'hybridations. On divise les méthodes hybrides en deux catégories [25] :

- Les hybridations collaboratives.
- les hybridations intégratives.

Les algorithmes qui échangent des informations de façon séquentielle, parallèle ou entrelacée entrent dans la catégorie **des hybridations collaboratives**. Les algorithmes à **hybridation intégrative** font en sorte qu'une technique est une composante incorporée à une autre technique. Autrement dit, il y a un algorithme maître et un algorithme esclave.

1-L'hybridation collaborative

L'hybridation collaborative [26] séquentielle est exécutée de façon à ce que la méthode exacte soit un prétraitement de la métaheuristique, ou vice-versa. Dans un premier temps, une RL est exécutée pour créer un ensemble de solutions. Un sous-problème est créé à

partir des arêtes qui se retrouvent dans les solutions de l'ensemble. Finalement, on trouve la solution optimale du graphe restreint à l'aide d'un algorithme exact. Un prétraitement est tout d'abord réalisé pour réduire le graphe, suivi d'un algorithme mémétique. Finalement, un algorithme est appliqué au sous-problème donné par Jà fusion des solutions trouvées par l'algorithme mémétique.

2-L'hybridation intégrative

Les combinaisons d'une méthode exacte et d'une métaheuristique de façon intégrative se font de manière à ce qu'un des deux algorithmes soit une composante intégrée à l'autre algorithme. On peut donc intégrer une méthode exacte à une métaheuristique, et inversement. Premièrement, voici comment incorporer un algorithme exact à une métaheuristique. On peut résoudre de façon exacte une relaxation du problème. Ceci peut être pratique pour guider heuristiquement la recherche dans un voisinage, la recombinaison, la mutation, la réparation et l'amélioration locale [26].

2.8 Conclusion

Nous avons présenté dans ce chapitre les méthodes de résolution d'un problème d'optimisation combinatoire : les méthodes exactes, les méthodes approchées et les méthodes hybrides.
Les métaheuristiques qui sont des méthodes stochastiques qui visent à résoudre un large panel de problèmes. Ces méthodes sont inspirées d'analogies avec des domaines aussi variés que la physique, la génétique ou encore l'éthologie. Les métaheuristiques ont vite rencontré un vif succès grâce à leur simplicité d'emploi mais aussi à leur forte modularité. On a vu tout au long de ce chapitre que les métaheuristiques sont facilement adaptables et hybridables en vue d'obtenir les meilleures performances possibles.
Nous avons tiré que la richesse des différentes méthodes d'optimisation nous mène à choisir le bon compromis de méthodes et algorithmes afin de résoudre notre problème.
Dans le prochain chapitre nous allons étudier les algorithmes que nous appliqué dans notre approche : l'algorithme de Dijkstra de la classe des méthodes exactes, l'algorithme génétique et la recherche tabou de la classe des méthodes approchées et enfin nous allons proposer notre approche d'hybridation des algorithmes exactes et métaheuristiques.

Chapitre 3

Approche proposée

3.1 Introduction

Le développement et l'application des métaheuristiques hybrides est de plus en plus susciter l'intérêt académique. Les méthodes hybrides combinent les différents concepts ou les composants de divers métaheuristiques, et à cette fin, ils tentent de fusionner les points forts et éliminent les faiblesses des différents concepts de métaheuristiques.
Par conséquent, l'efficacité de l'espace de solutions recherché peut être encore amélioré et de nouvelles opportunités peuvent émerger, ce qui peut conduire à des méthodes de recherche encore plus puissantes et plus flexibles.
Dans ce chapitre, nous allons présenter notre approche qui sert à l'hybridation de deux métaheuristiques (l'algorithme génétique et la recherche tabou) avec l'algorithme de Dijkstra afin d'optimiser le problème de transport.

3.2 Modélisation mathématique du problème

Soit **G= (N, A)** un graphe, l'ensemble des nœuds liés par des arcs.
- $\boldsymbol{NI = \{NI_1, NI_2...., NI_p\}}$: ensemble de nœuds intermédiaires de cardinalité p.
- $\boldsymbol{D = \{D_1, D_2...., D_m\}}$: ensemble de villes de cardinalité m.
- $\boldsymbol{N = NI \cup D}$: ensemble de nœuds de cardinalité m+p.
- $\boldsymbol{d_{ij}}$: distance entre un nœud **i** et un nœud **j** de l'arc.
- $\boldsymbol{X_{ij}}$: est une variable de décision .

$$\boldsymbol{X_{ij}} \begin{cases} 1 \text{ fait partie du plus court chemin trouvé} \\ 0 \text{ sinon} \end{cases}$$

Soit G= (N, A) un graphe représentant l'ensemble des nœuds liés par des arcs.

$$\boldsymbol{\{A = (i,j), \forall i \in N et \forall j \in N : i \neq j\}}$$

$\boldsymbol{\forall j \in N : i \neq j}$ est l'ensemble des arcs a, où à chaque arc **a = (i,j)** on associe la valeur $\boldsymbol{d_{ij}}$ représentant la distance de l'arc a avec deux sommets distingués s (la source ou l'origine) et t (le but ou le puits) .

On considère le programme linéaire en les variables $\boldsymbol{X_{ij}}$ suivant :

$$\mathbf{F} = \sum_{ij \in N} \boldsymbol{d_{ij}} * \boldsymbol{X_{ij}}.$$

La fonction F est l'agrégation des distances, et qui optimisent respectivement la distance totale parcourue par les nœuds. L'explication intuitive est que $\boldsymbol{X_{ij}}$ est une variable qui sert à indiquer si l'arc (i,j) fait partie ou non du plus court chemin trouvé :
elle vaut 1 si c'est le cas, et 0 sinon.

1. On cherche à choisir l'ensemble d'arcs de poids total minimal, pourvu que les arcs composent un chemin de s à t ; cette condition est représentée par la contrainte qui exprime que, pour tout sommet autre que s et t, le nombre d'arcs entrants choisis doit être égal au nombre d'arcs sortant, et ainsi l'ensemble forme un chemin de s à t.

2. On cherche à minimiser la formule de coût avec les algorithmes qu'on a choisi et l'hybridation de ces dernières .

Donc notre fonction objective est :

$$\mathbf{Min\ F} = \text{Min} \sum_{ij \in N} \boldsymbol{d_{ij}} * \boldsymbol{X_{ij}}$$

3.3 Démarche du travail

Pour la représentation des villes de notre problème, nous avons suivi les étapes suivant :

a- Nous avons créé une matrice Random M1 qui contient 1000 nœuds (Départ/Arrivée) (voir l'annexe 1).

b- Nous avons créé une autre matrice M2 de la taille (254 * 254) qui représente les distances entre les villes, ses valeurs sont compromis entre 1 et 9999(voir l'annexe2).

c- La matrice M2 et une matrice non orientée :

La distance (A, B) = la distance (B, A) et la distance (A, A) = 0.

d- Cette matrice sera ensuite utilisée comme une entrée pour l'algorithme de Dijkstra pour calculer les plus courts chemins .

La figure suivante est un extrait de la matrice M2 qui représente la distance entre 10 couples (position de départ, position d'arrivée).

	pos 1	pos 2	pos 3	pos 4	pos 5	pos 6	pos 7	pos 8	pos 9	pos 10
pos 1	0	8189	6137	5410	2463	4462	2134	9759	3184	5655
pos 2	8189	0	9263	3262	2847	8733	915	7524	6568	64
pos 3	6137	9263	0	2880	3751	150	3931	1148	6728	8729
pos 4	5410	3262	2880	0	7625	7822	447	4558	6794	5319
pos 5	2463	2847	3751	7625	0	8419	1374	7626	9527	3226
pos 6	4462	8733	150	7822	8419	0	8558	9605	8853	3787
pos 7	2134	915	3931	447	1374	8558	0	9065	8379	1766
pos 8	9759	7524	1148	4558	7626	9605	9065	0	7395	4750
pos 9	3184	6568	6728	6794	9527	8853	8379	7395	0	131
pos 10	5655	64	8729	5319	3226	3787	1766	4750	131	0

FIGURE 3.1 – Extrait de la matrice M2 qui représent la distance entre 10 positions.

Exemple 1 :

Nous avons un graphe non orienté qui contient 5 nœuds (5 positions) dont chaque nœud a des arcs pondérés relié avec les autres nœuds (voir la Figure 3.2) .

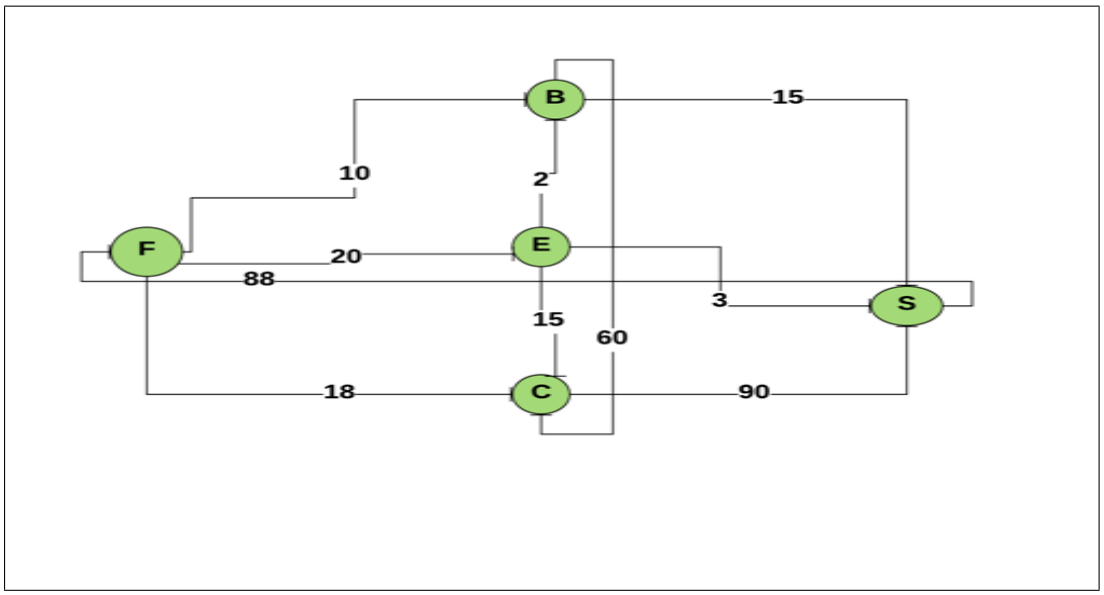

FIGURE 3.2 – Exemple de la modélisation d'un graphe reliant 5 positions (villes).

La matrice résultante de la modélisation graphique présentée dans la Figure 3.2 est comme suit :

$$M = \begin{array}{c} \\ \boldsymbol{Pos1} \\ \boldsymbol{Pos2} \\ \boldsymbol{Pos3} \\ \boldsymbol{Pos4} \\ \boldsymbol{Pos5} \end{array} \begin{array}{c} \begin{array}{ccccc} \boldsymbol{Pos1} & \boldsymbol{Pos2} & \boldsymbol{Pos3} & \boldsymbol{Pos4} & \boldsymbol{Pos5} \end{array} \\ \begin{pmatrix} \mathbf{0} & \mathbf{8189} & \mathbf{6137} & \mathbf{5410} & \mathbf{2463} \\ \mathbf{8189} & \mathbf{0} & \mathbf{9263} & \mathbf{3262} & \mathbf{2847} \\ \mathbf{6137} & \mathbf{9263} & \mathbf{0} & \mathbf{2880} & \mathbf{3751} \\ \mathbf{5410} & \mathbf{18} & \mathbf{20} & \mathbf{0} & \mathbf{7625} \\ \mathbf{2463} & \mathbf{2874} & \mathbf{3751} & \mathbf{7625} & \mathbf{0} \end{pmatrix} \end{array}$$

Dans ce qui suit, nous allons appliquer l'algorithme de Dijkstra, l'algorithme génétique et la recherche tabou sur l'exemple ci-dessous (Exemple 1).

3.4 Algorithme de Dijkstra

Pour la première étape, nous allons calculer le plus courts chemins entre les villes en utilisant l'algorithme de Dijkstra ou l'algorithme de Floyd-Warshall.
Nous avons testé les deux algorithmes afin de choisir l'algorithme le plus performant.
Le tableau suivant représente les résultats obtenus :

Départ → Arrivée	**Dijkstra**	**Floyd-Warshall**
141 → 16	109	446
165 → 151	151	201
20 → 59	174	238
99 → 81	120	424
60 → 233	106	285

TABLE 3.1 – Les distances trouvées par l'algorithme de Dijkstra et Floyd-Warshall.

Les résultats présentés dans le Tableau 3.1 montrent que l'algorithme de Dijkstra trouve des résultats mieux que l'algorithme de Floyd-Warshall avec une différence considérable.

Pour le temps d'exécution pris par les deux algorithmes (Dijkstra et Floyd-Warshall) : nous avons eu (0.8750498249 ms) pour l'algorithme de Dijkstra et (0.0670037269 ms) pour l'algorithme de Floyd-Warshall.

Nous remarquons que l'algorithme de Floyd-Warshall est plus rapide que l'algorithme de Djikstra mais La différence n'est pas importantes (elle est mesuré par ms) par contre, les distances retournées par Dijkstra (Table 3.1) sont meilleurs que celles de Floyd-Warshall . Donc nous avons choisis l'algorithme de Djikstra.

L'algorithme de Dijkstra est utilisé pour trouver le plus court chemin entre les villes en utilisant la fonction Dijkstra path length (G, départ, arrivée) (voir l'annexe 3) qui permet de retourner la longueur du chemin le plus court de départ à l'arrivé.

Exemple 2 :

On appliquant l'algorithme de Dijkstra sur l'exemple 1, nous cherchons à trouver le plus court chemin entre la première ville (Pos1) et la cinquième ville (Pos5).

Le plus court chemin entre Pos1 et Pos5 est De longueur 2463 (voir la Figure 3.3).

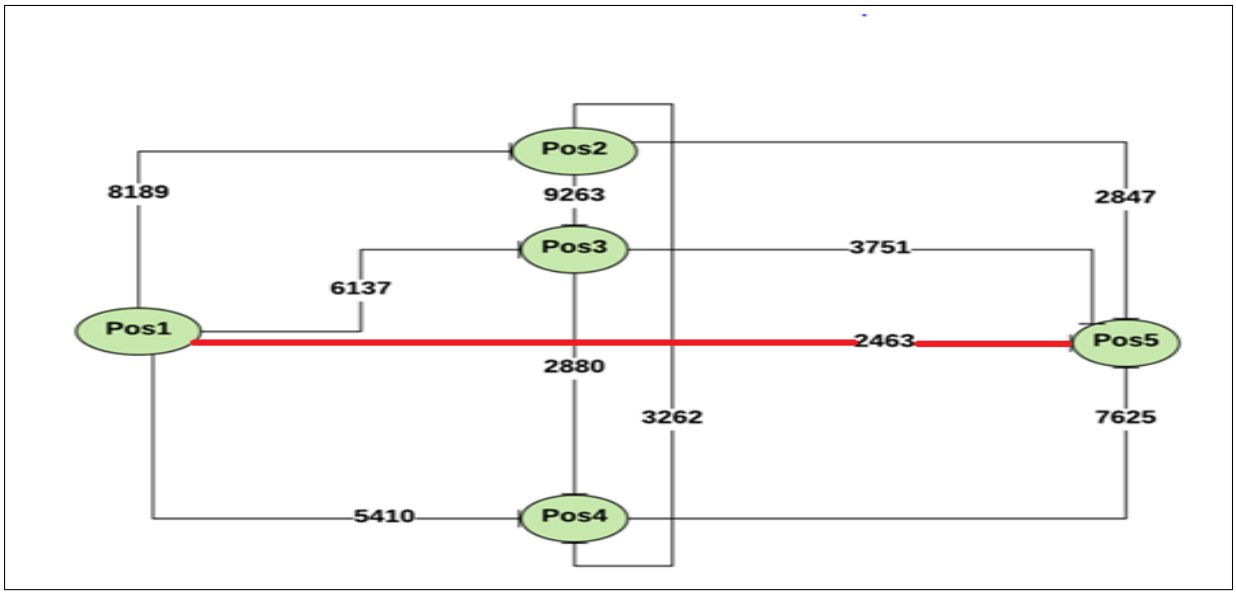

FIGURE 3.3 – Le plus court chemin trouvé par Dijkstra entre Pos1 et Pos5.

les résultats obtenus par l'algorithme de Dijkstra représentent la population initiale de l'algorithme génétique dans la deuxième étape de l'hybridation.

3.4.1 Calcul de la complexité

Soient **n** le nombre de nœuds et **m** le nombre d'arcs.

- L'étape de sélection du nœud qui devient permanent requiert au plus **n** opérations.

- On l'a fait au plus n fois ; donc cela demande un total d'au plus $\boldsymbol{n^2}$ opérations.

Le réajustement des distances demande qu'on examine chaque arc au plus une fois donc ce travail requiert **m** opérations.

Au total, l'algorithme effectue au plus $\boldsymbol{n^2}$ + **m** opérations $\leq 2\ \boldsymbol{n^2}$ opérations

Donc :

$$\textbf{Complexité (Dijkstra)} = \textbf{O (m + n log(n)).}$$

3.5 Algorithme génétique

Le fonctionnement de l'algorithme génétique est simple, on part d'une population de solutions initiales (les résultats obtenus de l'algorithme de Djikstra), arbitrairement choisies.

On évalue leur performance (Fitness) relative. Sur la base de ces performances, on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaire s simples, à savoir : la sélection, le croisement et la mutation. Quelques chemins se reproduisent et d'autres disparaissent et seuls les chemins les plus courts sont supposés survivre. On recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante (nombre de la populations finales =100) (voir l'annexe 4).

3.5.1 La génération de la population initiale

Cette phase consiste à la génération de la population initiale, c'est-à-dire le choix des chemins de départ que nous allons faire évoluer. Le choix de la population initiale conditionne fortement la rapidité de l'algorithme et elle doit être suffisamment diversifiée et de taille assez importante pour que la recherche puisse parcourir l'espace d'état dans un temps limité.

Pour choisir la population initiale nous allons tester trois choix avec le changement du critère de la population et le nombre de changement. Les résultats sont présentés dans le Tableau 3.2 :

	141→ 16	165→ 151	20→ 59	60→ 233	99→ 81	248→ 234	10→ 15	245→ 145	100→ 222	18→ 247
Population=50 NbChang = 5	1024	2760	2298	3236	1620	3233	1529	4417	5309	2103
population=100 NbChang = 10	151	183	181	129	129	159	185	87	107	135
Population=500 NbChang = 5	629	1228	348	425	684	853	731	1096	1152	506

TABLE 3.2 – les distances trouvées par l'algorithme génétique .

Selon le Tableau 3.2, les plus courts chemins sont trouvés dans la deuxième colonne. Donc nous allons fixer la population initiale par 100 et le nombre de changement par 10.

3.5.2 La sélection

Cette phase vise à sélectionner un nombre de chemins pour former la population initiale et pour les opérations de croisement et de mutation. Dans cette étape, les chemins de la population initiale et les chemins de croisement sont sélectionnés aléatoirement. Ensuite, le descendant ayant une valeur de la fonction coût qui est meilleure que celles de ses parents, va remplacer l'un de ses parents.

Également pour le chemin après mutation, il sera inséré dans la population si la valeur de sa fonction coût est meilleure que celle avant la mutation. Pour choisir les chemins meilleurs, nous appliquons la méthode de sélection pour la variété qu'elle peut assurer à la nouvelle population.

Nous appliquons cette étape sur l'exemple 1, pour sélectionner deux individus (Figure 3.4).

On fixe le Départ (Pos1) et l'arrivée (Pos5).

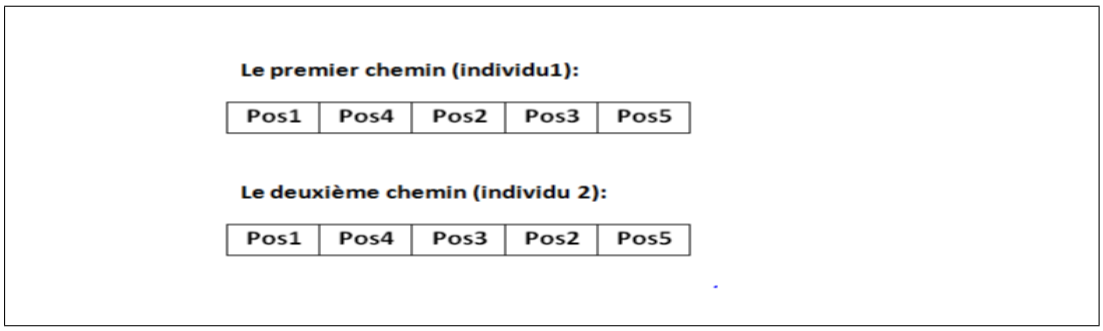

FIGURE 3.4 – Exemple de la sélection.

3.5.3 Le croisement

Le croisement doit se faire entre deux chemins. L'opération consiste à modifier la répartition des nœuds entre les deux chemins afin de trouver le plus court chemin que celui de leurs parents. Par exemple on a deux individus, le parent P1 et le parent P2 présentent deux chemins. Nous choisissons le croisement à un point.
Nous appliquons cette étape sur l'exemple 1, pour donner deux nouveaux individus fils à partir des individus sélectionnés (parents).

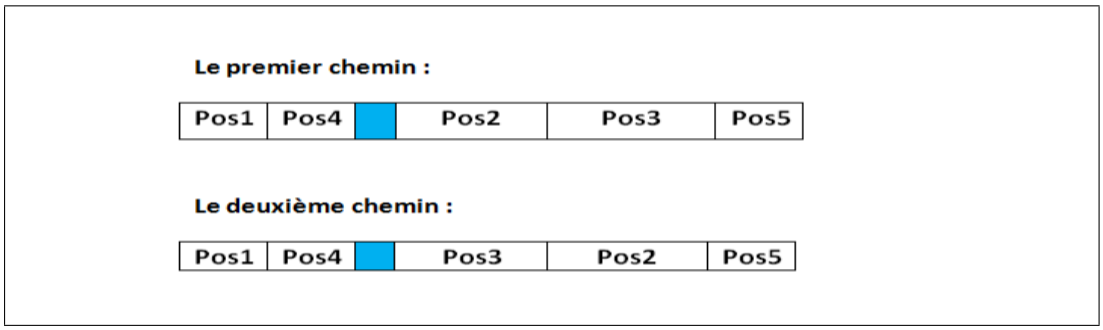

FIGURE 3.5 – Exemple de deux chemins avant le croisement.

Les résultats après le croisement sont illustrés dans la figure 3.6.

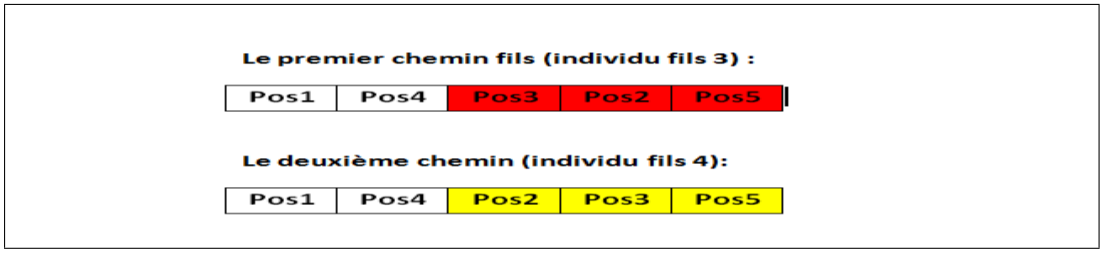

FIGURE 3.6 – Exemple de deux chemins après le croisement.

3.5.4 La mutation

En règle générale, l'algorithme génétique effectue une mutation en modifiant ou en inversant le chromosome candidat. Ici, l'opération de mutation tente de maintenir la diversité dans la population en modifiant le chemin actuel représenté par un chromosome. Pour un chromosome, un gène est choisi par hasard comme un point de mutation.

Nous appliquons cette étape sur l'exemple 1, pour donner deux nouveaux individus fils à partie des individus sélectionnés voire la figure 3.7.

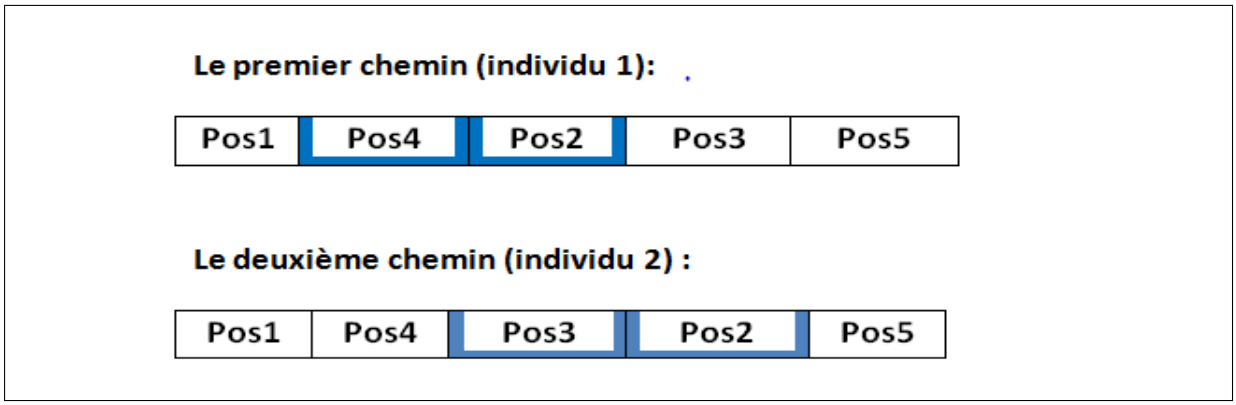

FIGURE 3.7 – Exemple de deux chemins avant la mutation.

Les résultats après mutation sont illustrés dans la figure 3.8.

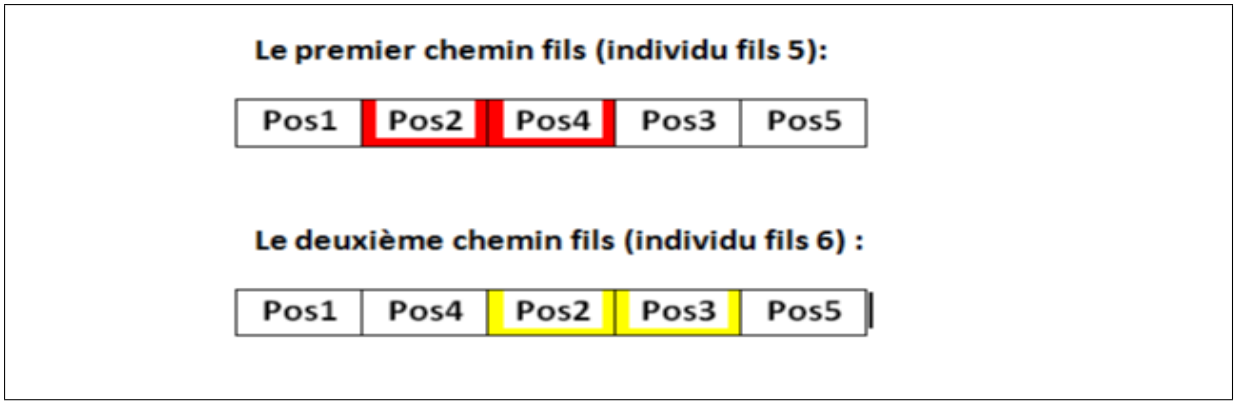

FIGURE 3.8 – Exemple de deux chemins après la mutation.

3.5.5 Calcul de la complexité

pour calculer la complexité on a :

z : la taille du jeu d'instructions pour le nombre de (départ, arrivée).

n : la taille du jeu d'instructions pour la population.

Donc :

$$\textbf{Complexité (AG)} = \textbf{O}\ (\textbf{z}\ (n+1)^{\mathbf{3}}\ (\tfrac{n}{5}+1)).$$

3 : c'est le nombre de fois nous utilisons le (n+1) dans l'algorithme d'hybridation.

$\frac{n}{5}+1$: Le nombre de jeu d'instruction utiliser donc cette boucle.

3.6 Recherche tabou

La recherche tabou garde temporairement en mémoire les transformations effectuées sur la solution. Cette mémoire s'appelle la liste de tabous et son rôle est d'empêcher l'algorithme de revenir sur ses pas. La nouvelle solution est trouvée en fouillant l'ensemble du voisinage de la solution courante. Un exemple de critère d'acceptation serait de retirer de la liste de tabous une transformation qui permettrait d'obtenir une solution de qualité supérieure à la meilleure solution rencontrée.

3.6.1 Voisinage (Random)

Un mouvement permet de passer d'une solution valide à une autre. On utilise la fonction Random () qui donne des chemins choisit au hasard entre le départ et l'arrivée et on les stocke dans une liste pour éviter d'avoir deux même chemins. On dit que la nouvelle solution est un voisin de la précédente.

L'ensemble des solutions que l'on peut atteindre à partir d'une solution s'appelle le voisinage dans la première itération. On améliore les chemins dans la deuxième itération jusqu'à 20 itérations.

3.6.2 Calcul de la distance de chemin

On calcule la distance entre les deux sommets de départ et l'arrivée de chaque chemin choisit par Random et on stock le résultat.

Par exemple, on prend un couple de Pos1 ($\boldsymbol{x_1}$, $\boldsymbol{y_1}$) et de Pos5 ($\boldsymbol{x_2}$, $\boldsymbol{y_2}$) puis on applique

la formule euclidienne pour calculer la distance :
Soient Pos1 et Pos5 deux points dans le plan cartésien, ($\boldsymbol{x_1}$, $\boldsymbol{y_1}$) les coordonnées du point Pos1 et ($\boldsymbol{x_2}$,$\boldsymbol{y_2}$) les coordonnées du point Pos5.
Alors la distance (Pos1,Pos5) sur le plan vaut :

$$\mathbf{Pos1Pos5} = \sqrt{(\boldsymbol{x_2} - \boldsymbol{x_1})^2 + (\boldsymbol{y_2} - \boldsymbol{y_1})^2}.$$

3.6.3 Gestion de la liste tabou

La liste tabou stocke la distance entre le départ et l'arrivé de chaque itération, les anciennes solutions trouvées seront remplacées par des nouvelles solutions retenues. Pour donner le plus court chemin on retourne le minimum de la liste tabou.
Pour le critère d'arrêt de la recherche tabou, nous avons effectué 20 itérations donc nous avons 20 cases de mémoire à parcourir dans la liste tabou (voir l'annexe 5).

3.6.4 La recherche tabou pour trouver le plus court chemin

Étape 1 : choisir les deux points (source/destination) on fixe nb_dep_ari=10 (l'ensemble des solutions modifiable).

Étape 2 : Donner un chemin au hasard qui contient taille des nœuds entre 4 et 15 (fonction Random()).

Étape 3 : cpt=cpt+sh1.cell (a1, arrivé).value, alors nous avons calculé la distance de chemin.

Étape 4 : mettre à jour la liste T et les critères d'aspiration.

Étape 5 : si une condition d'arrêt est atteinte, stop.Sinon, retour à étape 2.

Étape 6 : min (tabu_list) donner la plus courte distance.

3.6.5 Calcul de la complexité

Nous avons les paramètres suivants :
z : la taille de jeu d'instruction pour le nombre de départ et arrivée.
n : la taille de jeu d'instruction pour n le nombre des positions (nœuds) dans la matrice.
Donc la complexité de l'algorithme de la recherche tabou est :

$$\textbf{complexité (RT)} = \textbf{O (z(n)(nc-1))}$$

3.7 L'approche proposée : l'hybridation

Dans notre approche proposée, nous avons hybridé un algorithme exacte (Dijkstra) avec deux algorithmes métaheuristiques (l'algorithme génétique, l'algorithme de la recherche tabou).

- Comme première étape, nous appliquons l'algorithme de Djikstra afin de retrouver les N nœuds les plus proche du point de départ.
- Les différents chemins retenus de la première phase seront considérés comme les chemins de la population initiale de l'AG choisis (sélectionné) d'une manière aléatoires.
- Pour la dernière étape, on stocke les solutions calculées dans une liste Tabou afin de retourner le plus court chemin en utilisant l'algorithme de recherche tabou.

3.7.1 Les étapes de l'hybridation

Nous avons neuf étapes à suivre comme suit :

1. Choisir le point de départ et le point d'arrivée.
2. Trouver les N nœuds les plus proches du départ par l'algorithme de Dijkstra.
3. Générer la population initiale aléatoirement (l'algorithme génétique).
4. Ajouter des nœuds dans chaque ligne de la population, par la fonction de Random ().
5. Calcule la distance qu'on à eu pour chaque ligne de la population.
6. Choisir les meilleure 10 nœuds les plus proches du noeud de départ, on applique des mutations et des sélections sur la population.
7. Enregistrer les solutions calculées dans la liste Tabou.
8. La liste tabou est formée des deux premier meilleurs résultats prisent de la liste doivent être supprimée de cette dernière.
9. Répéter ça plusieurs fois (nombre d'itérations = 10).

Dans ce qui suit, nous allons illustrer quelques codes sources des algorithmes appliqués.

Étape I : Algorithme de Dijkstra

On trouve les N nœuds les plus proches du départ par l'algorithme de Dijkstra, et on fixe la deuxième colonne de la liste (voir la Figure 3.9).

Les entrées, sorties de l'algorithme de Dijkstra sont citées ci dessus :

- **Input :**Départ, Arrivé, Matrice M2.
- **Output :** Le résultat de la deuxième colonne de la matrice finale (best first).

```
for i in range(1,best_first +1):
    ligne=[]
    cpt=0
    depart=int(sh2.cell(z,0).value) #lire la source
    arriver=int(sh2.cell(z,1).value) #lire la destination
    nb_case=random.randint(4,15)  #taille de la solution entre 4 et 15
    ligne.append(depart)
    # donner une liste contient 10 mielleures solution on utilise l'algorithme Dijkstra
    print(i,depart)
    ligne.append(sh1.cell(i,depart).value)
    a1=depart
    # on prond les valeures de la matrice à partire de la troisiéme colonne juska l'arriver
    for j in range(3,nb_case-1):
          var=random.randint(1,taille_matrice)
          ligne.append(var)
          print(i,arriver)
          cpt=cpt+sh1.cell(i,arriver).value
          a1=var
    ligne.append(arriver)
```

FIGURE 3.9 – Code de calcul du plus court chemin à l'aide de l'algorithme de Dijkstra .

Étape II : Algorithme génétique

Les entrées, sorties de l'algorithme génétique sont citées ci dessus :

- **Input :**Départ, Arrivé, Matrice.
- **Output :**résultats des chemins à partie de la troisième colonne de la matrice finale.

On appliquer des mutations et des sélections de l'algorithme génétique sur la population.

Exemple :

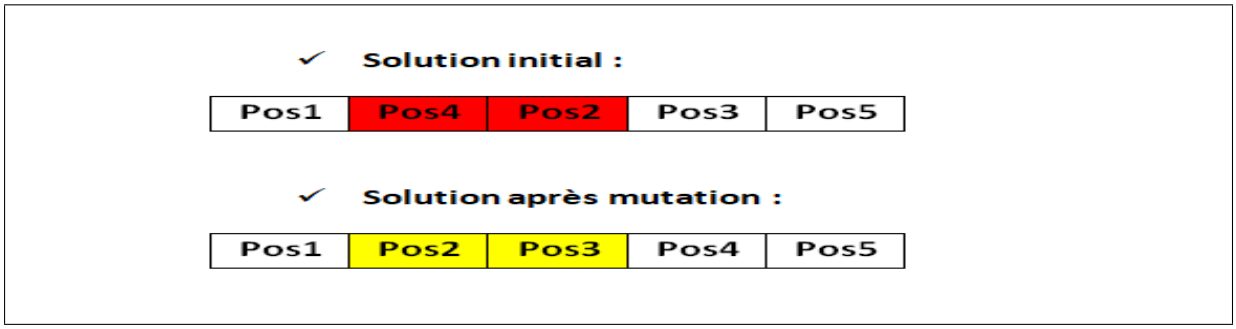

FIGURE 3.10 – Exemple de mutation entre deux chemins.

le code de la mutation est présenté dans la figure suivante :

```
for j in range(0,nb_changement): #faire la mutation
    var1=random.randint(1,taille_ligne-2)
    var2=random.randint(1,taille_ligne-2)
    aaa=liste [i][var1]
    liste[i][var1]=liste [i][var2]
    liste[i][var2]=aaa
for j in range(0,nb_changement): #faire la mutation 10 fois
    var1=random.randint(1,taille_ligne-2)
    var2=random.randint(1,taille_matrice)
    liste[i][var1]=var2
```

FIGURE 3.11 – Code de mutation .

Après la mutation on applique le croisement selon le code suivant :

```
for i in range(0,population):
   cpt=cpt+resultat[i]
for i in range(0,int(population/5)):
 for i in range(0,population):
     # c'est l'opération de croisement
   for j in range(0,population):
       if resultat[i]<resultat [j]:
           aaa=resultat [j]
           resultat [j]=resultat [i]
           resultat [i]=aaa
           aaa=liste [j]
           liste[j]=liste [i]
           liste[i]=aaa
```

FIGURE 3.12 – Code de croisement .

Étape III : Algorithme de recherche tabou

Les entrées, sorties de la recherche tabou sont citées ci dessus :

Input : Résultat Dijkstra, résultat Génétique de chaque itération.

Output : Résultat final de plus court chemin.

On stocke les solutions calculées dans une liste tabou selon le code dans la figure suivante . La liste TABOU est formée des deux premiers meilleurs résultats pris par la liste vont être supprimée de cette dernière. On répète ça dix fois.

```
        # stocker les 2 meilleurs solution dans la liste tabou
        for i in range(int(population/5),population +1):
            tabou.append(liste[i])
```

FIGURE 3.13 – Code de stockage liste Tabou .

pour la suppression on applique le code suivant :

```
# cette partie parmet de supprimer les données de la liste tabou dans la liste
for iii in range (0, len(tabou)):
    for ii2 in range (0, len(liste)):
        if liste(iii)==liste[ii2]:
            del(liste[ii2])
```

FIGURE 3.14 – Code de suppression liste Tabou .

3.7.2 Complexité de l'hybridation

Pour calculer la complexité de l'hybridation on a décomposé notre code en 3 parties :

- Complexité de la première partie :

On calcule la complexité de la première partie des l'hybridation de trois algorithmes .

m : la taille de jeu d'instruction pour le best first.

s : la taille de jeu d'instruction pour le nb case.

F : la taille de jeu d'instruction pour le tabou.

A : la taille de jeu d'instruction pour la liste.

Donc la complexité de cette partie est comme suit :

$$\mathbf{T(1)= O(m((F+1)(A+1)))}$$

- Complexité de la deuxième partie :

On calcule la complexité de la deuxième partie des l'hybridation de trois algorithmes n : la taille de jeu d'instruction pour la population.

donc la complexité est :

$$\mathbf{T(2)= O((n+1)(((n/5)+1)(n^2)))}$$

- Complexité de la troisième partie :

On calcule la complexité de la troisième partie des l'hybridation de trois algorithmes .

n : la taille de jeu d'instruction pour la population.

nb : la taille de jeu d'instruction pour le nb changement.

Donc la complexité de cette partie est comme suit :

$$\mathbf{T(3)= O\ (((4/5n)\ +1)\ (nb+1))}$$

Complexité de l'algorithme total

La complexité de l'algorithme totale est présentée dans le tableau 3.3 :

Z : la taille de jeu d'instruction pour le nombre (départ, arrivée).

La première partie	**T(1)= O (m ((F+1) (A+1)))**
La deuxième partie	**T(2)= O((n+1)(((n/5)+1)(n^2)))**
La troisième partie	**T(3)= O (((4/5n) +1) (nb+1))**
Algorithme d'hybridation	**T_total=O(z((n+1)((n/5+)1)(n^2))))**

TABLE 3.3 – La complexité de l'hybridation.

3.8 Conclusion

Avec la complexité du réseau de transport, les algorithmes de la théorie des graphes seuls comme l'algorithme de Dijkstra ou Floyd-Warshall sont incapables de résoudre les différents problèmes d'aide au déplacement.

Les algorithmes métaheuristiques comme l'algorithme génétique et la recherche tabou fournissent rapidement une solution réalisable, ils sont souvent spécifique au problème qu'il traite : il exploite certaines propriétés du problème pour orienter la recherche vers des zones susceptibles de contenir la solution.

Ce chapitre a été consacré à la présentation des algorithmes que nous avons utilisé dans notre recherche et les étapes de l'hybridation.

La discussion de nos résultats et la description détaillée de notre application auront lieu dans le prochain chapitre.

Chapitre 4

Implémentation et tests

4.1 Introduction

Ce chapitre présente l'environnement de développement de notre application. Il se compose de deux sections :
La première section présente une comparaison entre les résultats du test des trois algorithmes et l'algorithme hybride proposé.
La deuxième section quant à elle porte sur la description de notre application.

4.2 Environnement informatique utilisé

Cette partie décrit l'environnement logiciel et matériel que nous avons utilisé.

4.2.1 Environnement logiciel

Nous avons choisi Python pour implémenter nos algorithme, Microsoft Office Excel 2007 pour enregistrer les résultats d'exécution et Android studio pour l'implémentation de l'application.

a- Python

Python [27] est un langage portable, dynamique, extensible, gratuit, qui permet (sans l'imposer) une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles.

La version de python que nous avons utilisé est Python 3.7.4 [1].

Caractéristiques du langage

Python a plusieurs caractéristiques parmi lesquels on peut citer [27] :

- Gratuit, mais on peut l'utiliser sans restriction dans des projets commerciaux.
- La syntaxe de Python est très simple et, combinée à des types de données évolués (listes, dictionnaires,...), conduit à des programmes à la fois très compacts et très lisibles.
- Python gère ses ressources (mémoire, descripteurs de fichiers...) sans intervention du programmeur, par un mécanisme de comptage de références (proche, mais différent, d'un garbage collector).
- Python est orienté-objet.

Domaines d'application

Les domaines d'application de Python incluent entre autres [27] :

- L'apprentissage de la programmation objet.
- Les scripts d'administration système ou d'analyse de fichiers textuels.
- L'accès aux bases de données (relationnelles).
- La réalisation d'interfaces graphiques utilisateurs.
- Le calcul scientifique et l'imagerie.

  Python ne sert alors pas à écrire les algorithmes, mais à combiner et mettre en œuvre rapidement des librairies de calcul écrites en langage compilé.

Les bibliothèques utilisée :

- **random :** fonction permettant de travailler avec des valeurs aléatoires .
- **math :** toutes les fonctions utiles pour les opérations mathématiques .
- **time :** fonction permettant de travailler avec le temps (calculer le temps d'exécution).
- **xlrd :** pour la lecture des données et la mise en forme d'informations à partir de fichiers Excel, qu'il s'agisse de fichiers .xls ou .xlsx.
- **networkx :** pour l'optimisation de graphe .
- **xlwt :** pour générer des fichiers tableurs compatibles avec Microsoft Excel.

1. lien de téléchargement : https ://www.python.org/

b- Office Excel :

Excel [28] est un logiciel de la suite bureautique Office de Microsoft qui permet la création des tableaux, de calculs automatisés, de plannings, de graphiques et de bases de données. On appelle ce genre de logiciel un (tableur).

Il permet de créer facilement des tableaux de toutes sortes, et d'y intégrer des calculs.

Il permet de générer de graphique pour mieux visualiser les valeurs et les interpréter.

C'est un puissant outil de visualisation mathématique.

Nous avons utiliser Office Excel 2007 pour stocker les résultats d'exécution des différents algorithmes implémentés..

c- Android studio :

Android Studio [27] est un nouvel environnement pour le développement et programmation entièrement intégré qui a été récemment lance par Google pour les systèmes Android. Il a été conçu pour fournir un environnement de développement et une alternative à Eclipse qui est l'IDE le plus utilisé.

Android Studio permet de voir chacun des changements visuels que vous effectuez sur votre application et en temps réel, vous pourrez voir aussi son effet sur différents appareils Android.

Android Studio offre aussi d'autres fonctionnalités [27] :

- Un environnement de développement robuste.
- Une manière simple pour tester les performances sur d'autres types d'appareils.
- Des assistants et des modèles pour les éléments communs trouvés sur tous les développeurs d'android.

4.2.2 Environnement matériel

L'application de ces approches est faite avec un ordinateur ayant 4GO de mémoire RAM, 2.0 GHz vitesse du processeur, CPU i3, Disque dur : 500 GB et Carte graphique Intel HD 5500.

4.3 Scénario expérimental du travail

Nous avons testé les trois algorithmes avec un nombre différent des couples (départ et arrivée).

Les paramètres de *l'algorithme génétique* sont :

- La probabilité des chemins sélectionnés pour le croissement à partir de chaque population à chaque génération qui est de 0,2

- La probabilité des chemins sélectionnés pour la mutation est de 0,7.

Nous supprimons d'abord les cases de la population entre 20 % et 100 %, après nous faisons la mutation 10 fois pour générer une nouvelle génération. Pour cela, nous avons réalisé 100 tests.

Les paramètres de l'algorithme *recherche tabou* sont :

- Nous avons fixé le nombre de couples (source et destination) à 10 et on choisit le chemin aléatoirement avec la fonction Random (). La taille de chemin doit être comprise entre 4 et 15.

Nous avons fait 100 tests. Après cette étape, nous faisons une étude comparative entre les résultats obtenus par les trois algorithmes.

4.4 Scénario expérimental de l'hybridation

L'application de ces approches est faite avec un ordinateur ayant 4GO de mémoire RAM et 2.0 GHz vitesse du processeur. Nous avons fait l'hybridation des trois algorithmes avec des nombre différents des couples (départ et arrivée).

- Les paramètres de l'hybridation sont : on a fixe le nombre de couples (source et destination)à 10 et nous avons fixé le nombre de meilleurs solutions à 20.

Nous avons d'abord utilisé l'algorithme de Dijkstra pour trouver les nœuds les plus proches aux nœuds de départ. A partir de la troisième colonne de la matrice nous avons utilisé *l'algorithme génétique* (nous avons utilisé les même paramètres, population = 100 et le nombre de changement = 5).

Ensuit, nous avons stocké les meilleurs résultats dans la liste tabou.

4.5 Résultats numériques

Dans cette section nous allons comparer les résultats de l'algorithme génétique en modifiant à chaque fois ses paramètre.
L'histogramme ci dessous montre la distance entre le départ et l'arrivé de 27 couples des 3 algorithmes Dijkstra, algorithme génétique et recherche tabou.

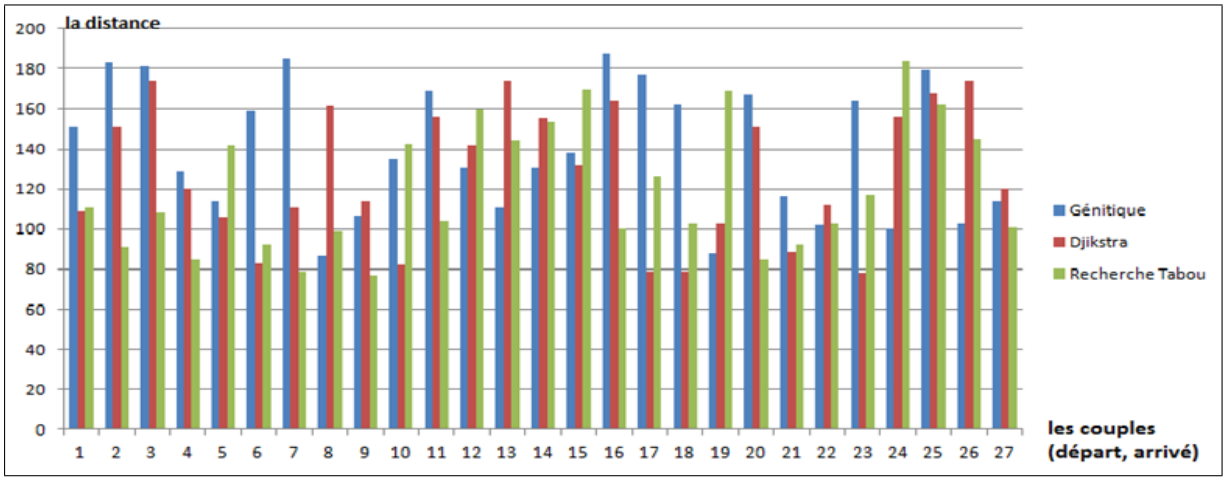

FIGURE 4.1 – Résultats du plus court chemin trouvés par : Dijkstra, AG, RT.

Le Tableau 4.1 représente les résultats (le plus court chemin entre la source et la destination pour 27 couples) trouvés par trois algorithmes que nous avons utilisé : *Dijkstra, algorithme génétique* et *recherche tabou*.

	Source	Destination	Dijkstra	Génétique	Recherche tabou
Couple 1	141	16	109	151	111
Couple 2	165	151	151	183	91
couple 3	20	59	174	181	108
couple 4	99	81	120	129	85
couple 5	60	233	106	114	142
couple 6	248	234	83	159	92
couple 7	10	15	111	185	79
couple 8	245	145	161	87	99
couple 9	100	222	114	107	77
couple 10	18	247	82	135	143
couple 11	26	174	156	169	104
couple 12	147	106	142	131	160
couple 13	78	15	174	111	144
couple 14	212	110	155	131	153
couple 15	130	163	132	138	170
couple 16	226	48	164	188	100
couple 17	50	42	79	177	126
couple 18	173	87	79	162	103
couple 19	23	118	103	88	169
couple 20	186	122	151	167	85
couple 21	44	146	89	116	92
couple 22	219	229	112	102	103
couple 23	27	16	78	164	117
couple 24	11	63	156	100	184
couple 25	226	76	168	179	162
couple 26	63	39	174	103	145
couple 27	76	108	120	114	101

TABLE 4.1 – Distances trouvées par : Dijkstra, algorithme génétique et la recherche tabou.

Le tableau 4.2 montre la somme et la moyenne des trois algorithmes (Dijkstra, Algorithme génétique, recherche tabou) ainsi que l'hybridation des trois algorithmes. Cette table nous a permet de comparer le résultat de 1000 couples (source, destination).

	Djikstra	Algorithme génétique	Recherche tabou	Hybridation
La somme	131930	129079	132112	103896
La moyenne	263,596404	257,9001	263,96004	207,53047

TABLE 4.2 – La somme et la moyenne des distances calculée par les trois algorithmes et l'hybridation.

Selon le Tableau 4.2 nous avons remarqué que l'algorithme de l'hybridation a trouvé des bons résultats avec des différences considérables.

4.6 Interprétation des résultats

- Les tests numériques (Tableau 4.1) montrent une différence importante entre les valeurs des trois algorithmes (Dijkstra, génétique et recherche tabou).
Par exemple : pour 1000 couples (départ, arrivée) (Tableau 4.2) , la valeur de la moyenne calculée par l'algorithme de Dijkstra est mieux que celles calculés par l'AG ou par RT.

La différence en terme de moyenne (Tableau 4.2) des trois algorithmes avec l'algorithme de l'hybridation est considérable. La moyenne calculée par l'algorithme hybride est la meilleur.
Le Tableau 4.3 présent le temps d'exécution pris par les différents algorithmes utilisés :

Les algorithmes	Le temps d'exécution
Dijkstra	1 (s) 15(ms)
Algorithme Génétique	3 (s) 47 (ms)
Recherche Tabou	1 (s) 67 (ms)
Hybridation	4 (s) 60 (ms)

TABLE 4.3 – Le temps d'exécution des trois algorithmes et l'algorithme d'hybridation.

Les résultats de temps d'exécution (Tableau 4.3) montrent que l'algorithme de Dijkstra n'a pas augmenté considérablement le temps d'exécution de l'approche. Mais Dijkstra améliore considérablement les valeurs de la fonction objective.
L'algorithme génétique a pris beaucoup de temps par apport Dijkstra et la recherche tabou à cause des opérations de sélection, croisement et mutation.
L'algorithme de l'hybridation a pris un temps considérable pour retourner les résultats car il est plus complexe.

4.7 Métriques d'évaluation

Comment mesurer les performances de chaque algorithme, nous avons calculer le rappel, la précision et F-mesure

a- La précision :

Représente le nombre de données classées correctement [28].

$$\text{Précision} = \text{VP}/(\text{VP}+\text{FP}).$$

VP : vraie positive.

FP : faux positif.

b- Le rappel :

Représente le nombre de données classées correctement, comme positive par exemple [28].

$$\text{Rappel} = \text{VP}/(\text{VP}+\text{FN}).$$

FN : faux négatif.

Nous utilisons une troisième mesure d'évaluation qui est **F-Mesure** afin de comparer les différentes approches qui aboutissent à un meilleur compromis entre précision et rappel. Cette mesure est la moyenne harmonique entre la précision et le rappel.

$$\text{F- Mesure} = 2^* (\text{Précision} * \text{Rappel})/(\text{Précision} + \text{Rappel})$$

Le tableau 4.4 représente le rappel et la précision et F-Mesure de Dijkstra pour 10 couples.

	Rappel	Précision	F-Mesure
Couple 1	0	0	0
Couple 2	0	0	0
couple 3	0	0	0
couple 4	0	0	0
couple 5	0	0	0
couple 6	0	0	0
couple 7	0	0	0
couple 8	0	0	0
couple 9	0	0	0
couple 10	0,04	1	0,076

TABLE 4.4 – Résultats de rappel, précision et F-mesure de l'algorithme de Dijkstra.

La figure 4.2 présente rappel et précision de Dijkstra pour 20 couples.

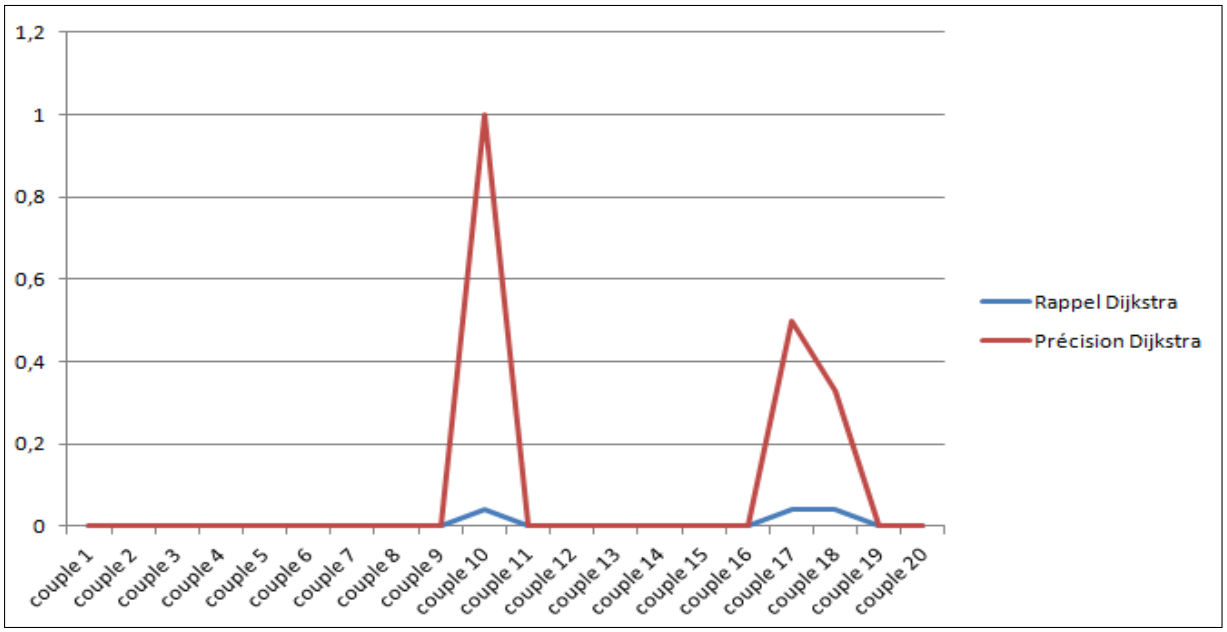

FIGURE 4.2 – Rappel et précision de Dijkstra.

Le tableau 4.5 présente rappel et précision de l'algorithme de génétique pour 10 couples.

	Rappel	Précision	F-Mesure
Couple 1	0	0	0
Couple 2	0	0	0
couple 3	0	0	0
couple 4	0	0	0
couple 5	0	0	0
couple 6	0	0	0
couple 7	0	0	0
couple 8	0,052	1	0,098
couple 9	0	0	0
couple 10	0,04	1	0,076

TABLE 4.5 – Résultats de rappel, précision et F-mesure de l'algorithme de génétique.

La figure 4.3 présente rappel et précision de l'algorithme génétique pour 20 couples.

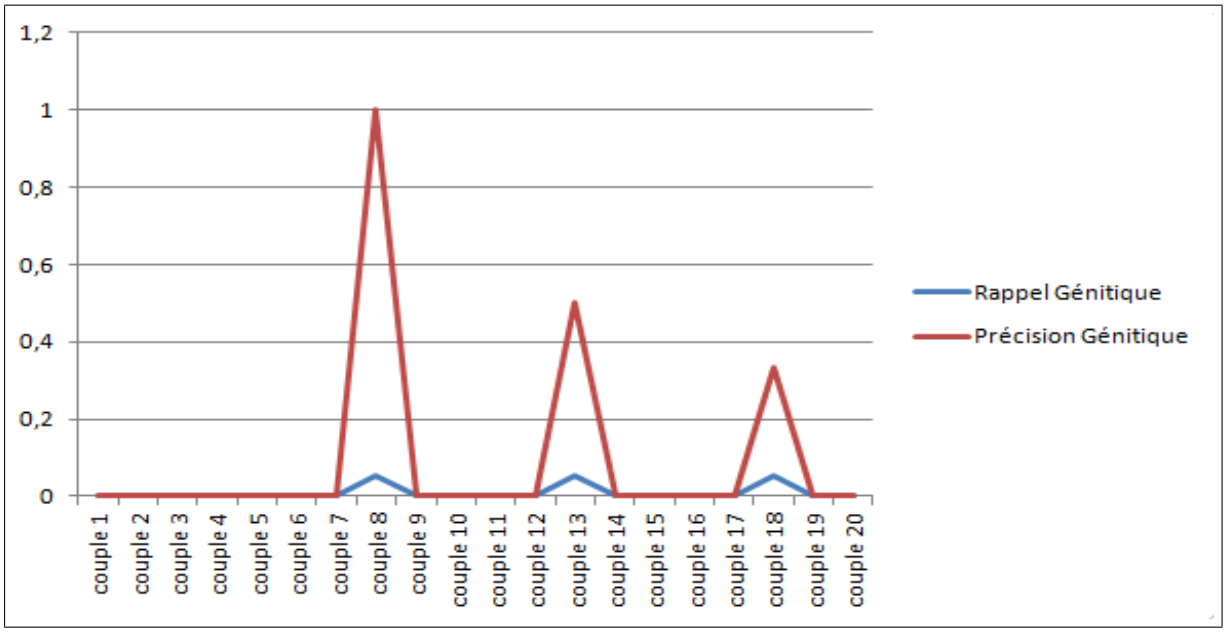

FIGURE 4.3 – Rappel et précision de l'algorithme génétique.

Le tableau 4.6 présente rappel et précision de l'algorithme recherche tabou pour 10 couples.

	Rappel	Précision	F-Mesure
Couple 1	0	0	0
Couple 2	0,058	1	0,111
couple 3	0,058	0,5	0,105
couple 4	0,058	0,333	0,098
couple 5	0	0	0
couple 6	0	0	0
couple 7	0,058	0,25	0,095
couple 8	0	0	0
couple 9	0,058	0,2	0,09
couple 10	0	0	0

TABLE 4.6 – Résultats de rappel, précision et F-mesure de l'algorithme recherche tabou.

La figure 4.4 présente rappel et précision de la recherche tabou pour 20 couples.

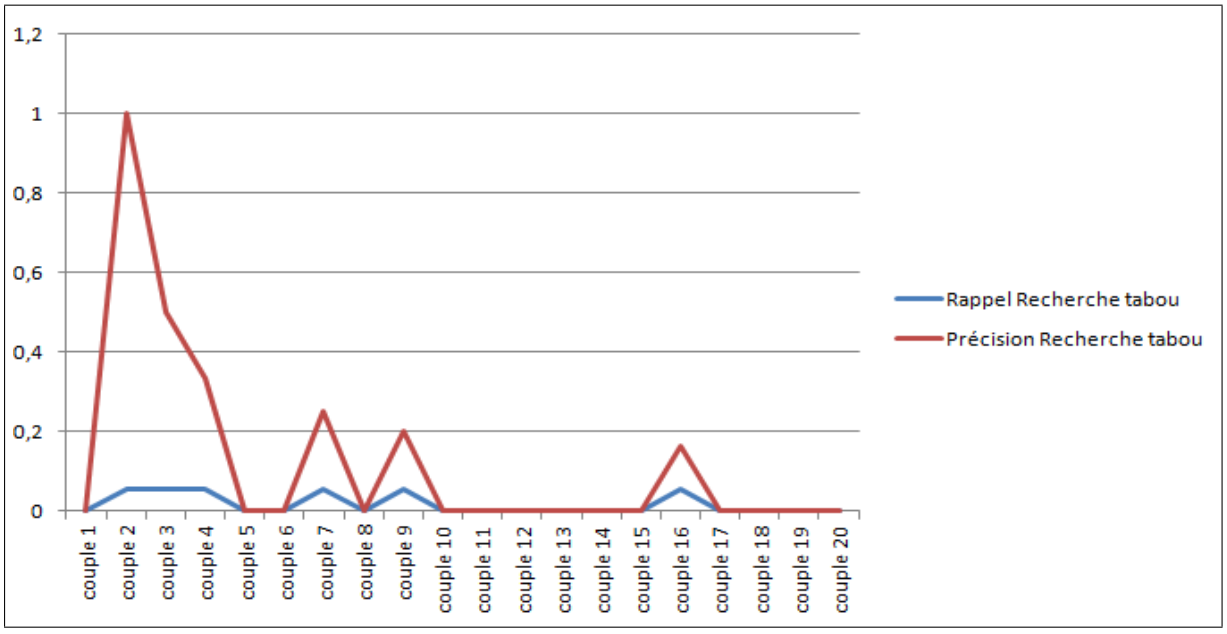

FIGURE 4.4 – Rappel et précision de l'algorithme de recherche tabou.

La figure 4.5 représente une courbe de la précision et le rappel de l'algorithme d'hybridation des 100 couples (source/destination), nous remarquons que la précision égal à un(1) pour la première chemin car ce chemin sont utilisé donne la partie de l'hybridation, ensuite nous remarquons que la précision est forte au début des chemins retourné après la précision diminue et augmente jusqu'à 0.4, depuis cette courbe nous jugeons que la méthode d'hybridation a donné des bon résultats pour les chemins. Nous remarquons que le rappel égal à un(0) pour la première chemin car ce chemin sont utilisé donne la partie de l'hybridation, ensuite nous remarquons que le rappel est enlevé au début des chemins retourné après le rappel augmente jusqu'à 1, depuis cette courbe nous jugeons que la méthode d'hybridation a donné des bon résultats pour les chemins.

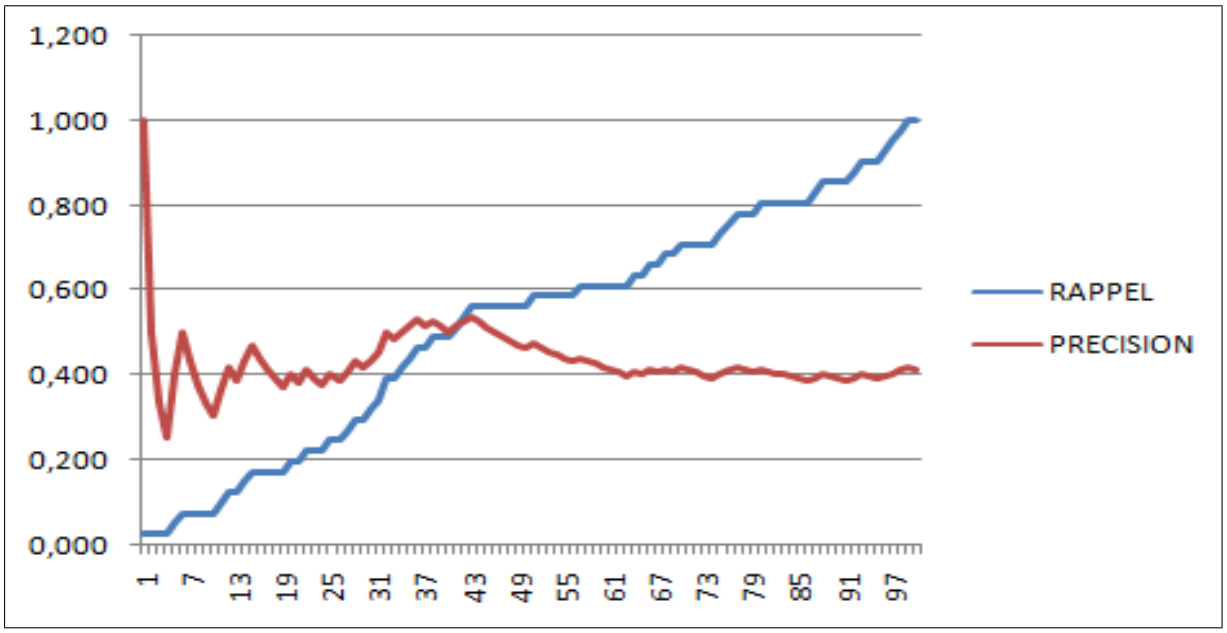

FIGURE 4.5 – Rappel et précision de l'hybridation .

4.8 Présentation de l'application

Notre application prend le nom de ≪ SMART ROAD≫ .

L'application permet de tracer l'itinéraire entre la ville de départ et de l'arrivée. pour se faire nous avons besoins d'une connexion internet et on doit activer notre localisation.

l'application contient deux activités : la première activité permet de remplir les champs qui vont prendre les coordonnés géographiques des villes (départ,arrivée)après on passe vers la deuxième activité qui va prendre les paramètres reçus et les affiches dans une map sous forme des **markers** ,finalement le google map api va tracer gratuitement une ligne entre les deux **markers**.

I- Google Maps :

Google Maps est un service de cartographie en ligne. Le service a été créé par Google. Lancé en 2004 aux États-Unis. Deux types de vue sont disponibles dans Google Maps : une vue en plan classique, avec nom des rues, quartier, villes et une vue en image satellite, qui couvre aujourd'hui le monde entier [29].

II- Fonctionnalité :

Google Maps offre les fonctionnalités suivantes :

- rechercher des lieux.
- obtenir un itinéraire en voiture, en transport en commun, à pied ou à vélo.
- afficher des informations sur le trafic.
- visualiser un lieu avec Google Street View.
- connaitre sa position en se géolocalisant sur la carte.
- accéder à des images satellite et 3D.

pour afficher toutes les informations sur l'itinéraire, on doit utiliser un service web de géocodage qui permet de retrouver une adresse postale sur base de coordonnées géographiques (x,y) et vice versa. Il peut être utilisé notamment pour positionner une adresse sur une carte ou pour lister le nom des rues appartenant à une localité.

Le service web de géocodage contient plusieurs services :

Le service web de calcul d'itinéraire : qui permet sur base des coordonnées de deux points, de calculer l'itinéraire le plus court ou le plus rapide entre ces points.

Le service web de recherche cadastrale : permet notamment d'identifier la

parcelle cadastrale sur base de coordonnées (X,Y). Il permet également à l'inverse, d'obtenir les coordonnées géographiques d'une parcelle sur base de ses références cadastrales.

Service web de segmentation dynamique : La segmentation dynamique consiste à localiser des objets sur une carte à partir d'un réseau de lignes.

Elle est notamment utilisée pour localiser précisément à l'aide de coordonnées (X,Y), des équipements situés le long de voiries (panneaux de signalisation, bornes kilométriques, etc.) [30].

La figure 4.6 présente l'interface d'accueil de notre application.

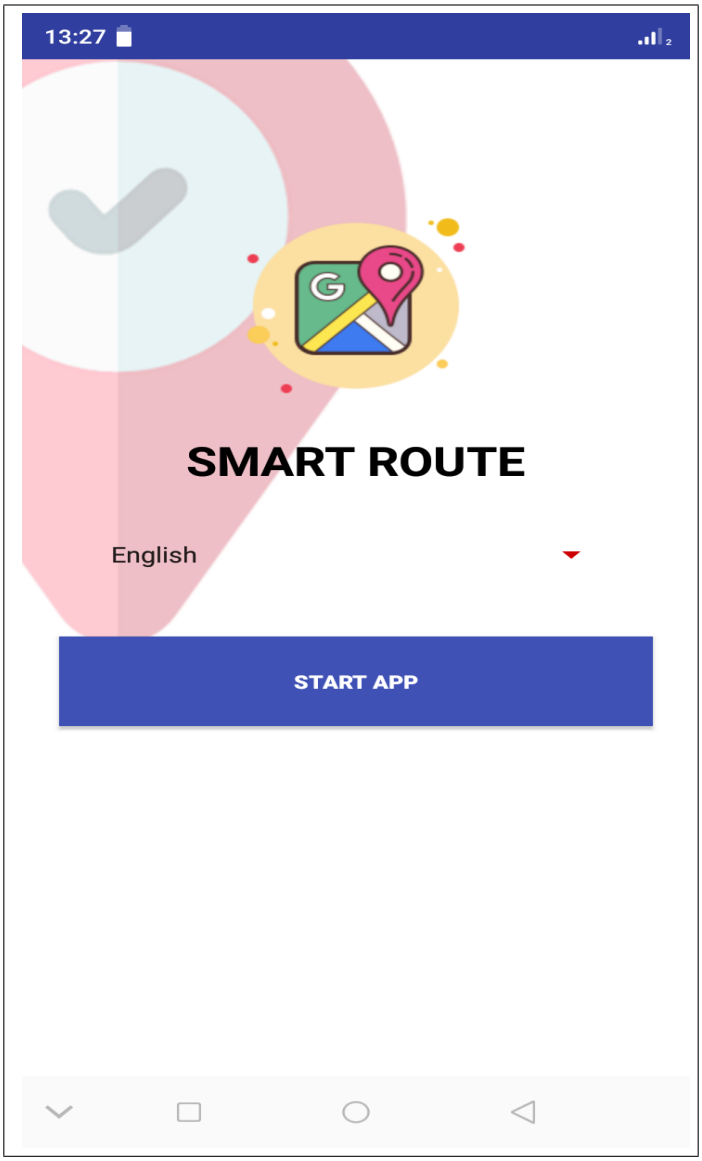

FIGURE 4.6 – L'interface d'accueil.

La deuxième interface permet de choisir la langue :
Français, Anglais, Arabe. (la figure 4.7) :

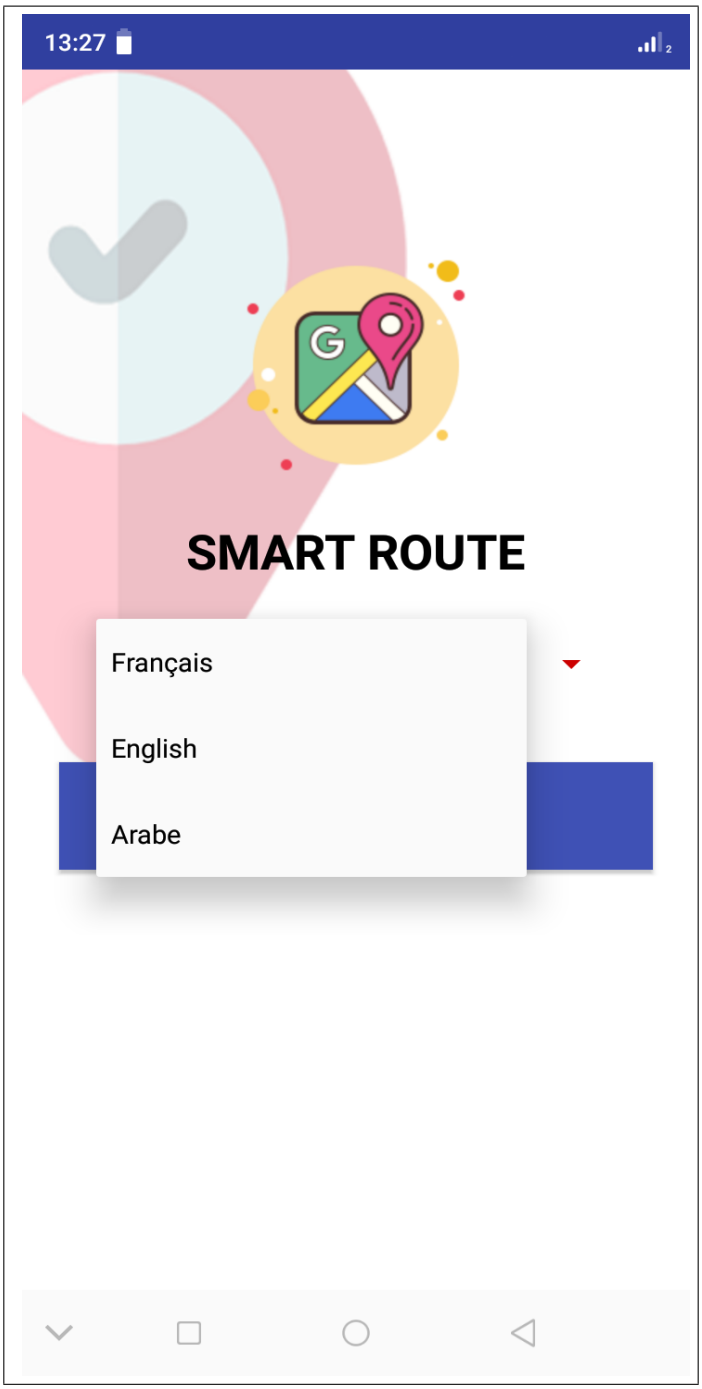

FIGURE 4.7 – L'interface utilisateur.

L'interface de la langue française est présentée dans la Figure 4.8.

FIGURE 4.8 – L'interface utilisateur : Langue française.

L'interface de la langue anglaise est présentée dans la Figure 4.9.

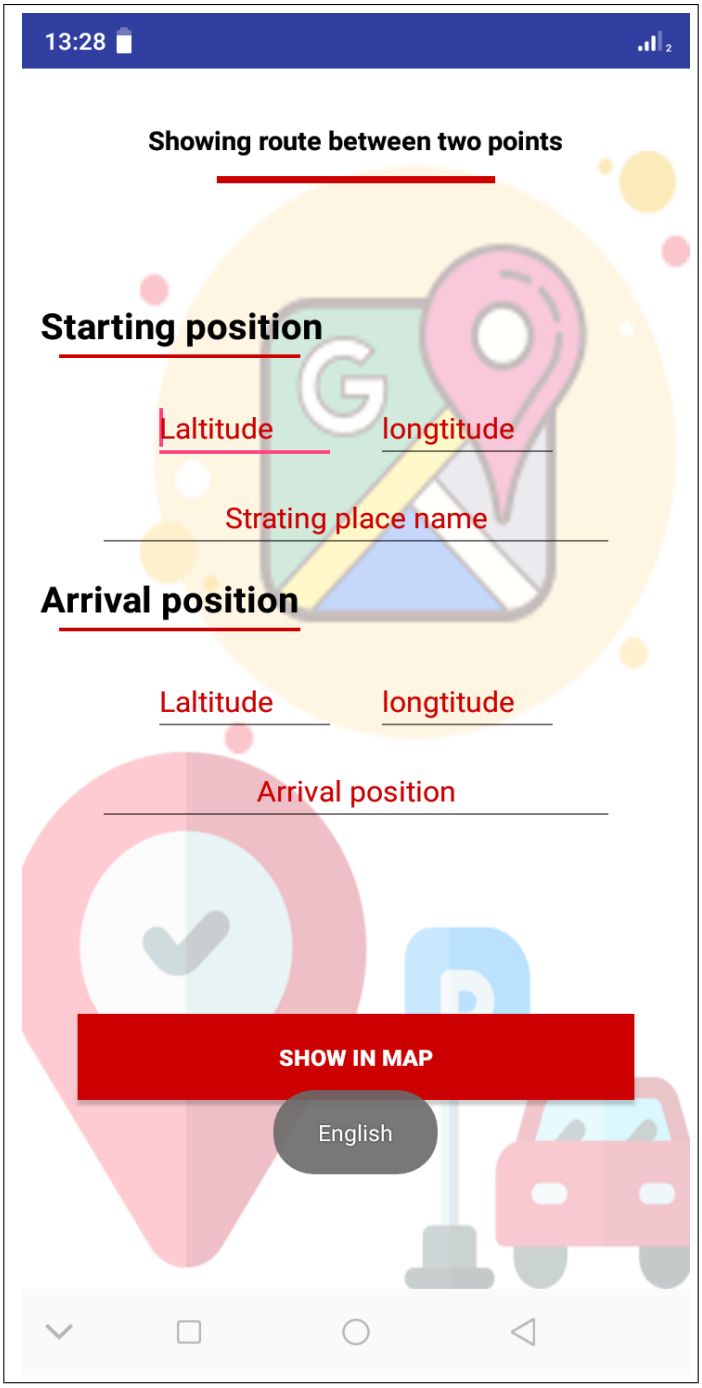

FIGURE 4.9 – L'interface utilisateur :Langue anglaise.

L'interface de la langue arabe est présentée dans la Figure 4.10.

FIGURE 4.10 – L'interface utilisateur : Langue arabe.

Après nous allons insérer les cordonnées le la ville de départ et la ville d'arrivée (Figure 4.11).

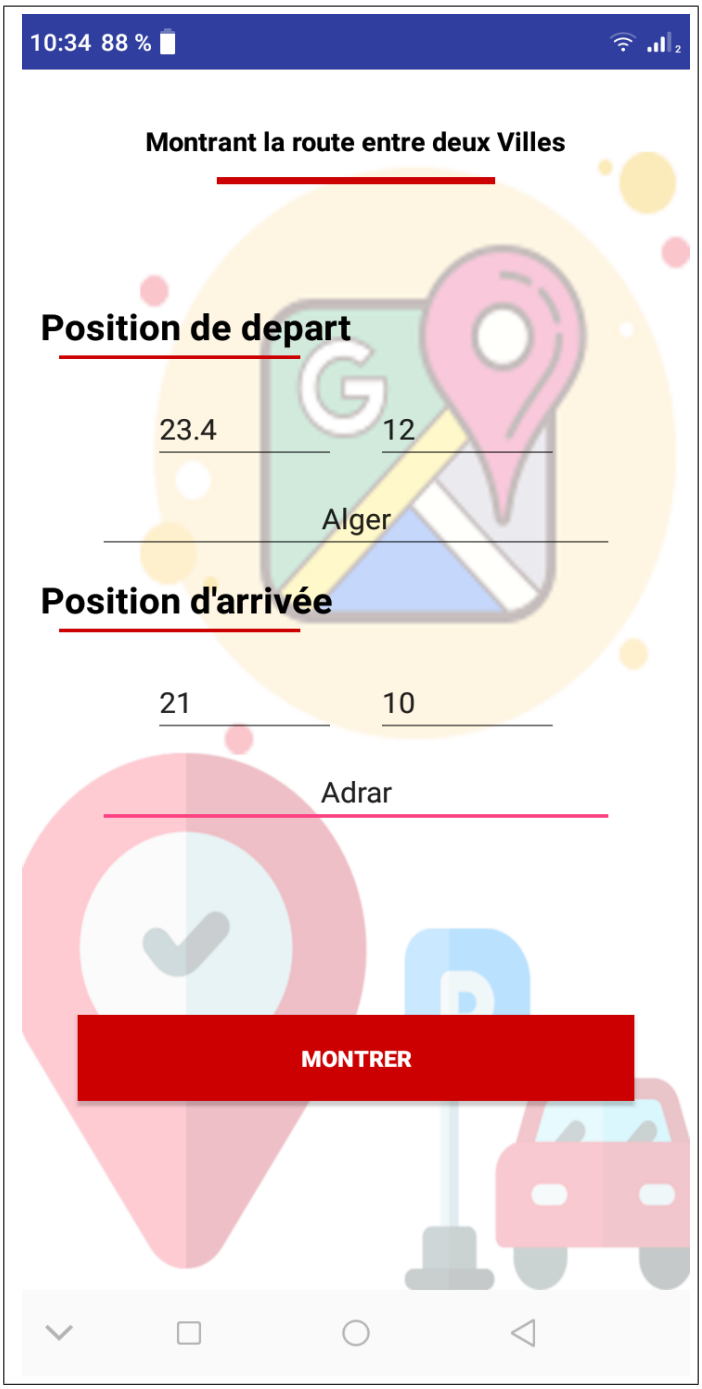

FIGURE 4.11 – L'insertion des cordonnées le la ville de départ et la ville d'arrivée.

Après l'insertion des cordonnées, nous obtenons une ligne droite entre la ville de départ et la ville d'arrivée (Figure 4.12).

FIGURE 4.12 – Affichage de la ligne entre le départ et l'arrivée .

Malheureusement nous pouvons pas afficher l'itinéraire qui contient toutes les informations entre les positions car nous avons pas l'API (geocoding web service) qui est réellement payant .cet API nous générés un fichier json qui contient tous les informations entre les deux position(la distance,le temps de circulation,le mode de déplacement,ect) et l'exploitation de ce dernier dans notre plateforme de développement java.

Pour l'affichage d'itinéraire nous devons utiliser l'API du service web de géocodage, on doit payer pour chaque service utilisé avec la création d'un compte d'accès pour la première fois (200 dollars) . selon le principe (payer au fur et à mesure) déclaré dans sa politique de facturation présentée la figure 4.13.

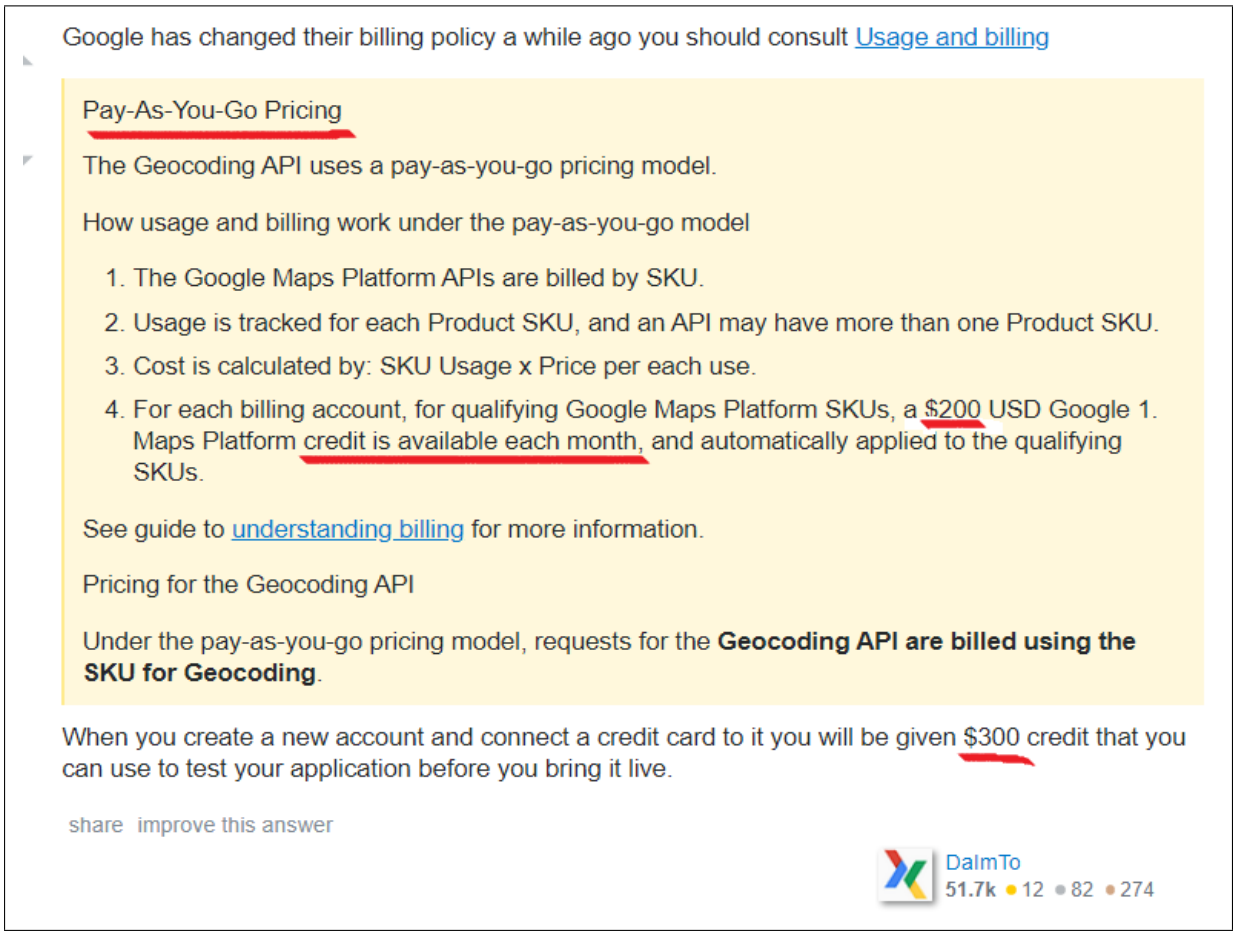

FIGURE 4.13 – Politique de facturation du service web de géocodage.

4.9 Conclusion

Dans ce chapitre nous avons présenter les résultats des différents algorithmes utilisés dans le troisième chapitre. Nous avons remarqué que l'hybridation a trouvé des bons résultats ce qui montre l'efficacité des méthode hybride.

Nous avons présenté aussi notre application qui permet de dessiner la ligne entre de départ et l'arrivée.Nous espérons de l'améliorer dans le future à l'aide des perspectives cités dans la conclusion générale.

Conclusion générale

Nous avons présenté le problème d'aide à la décision pour l'optimisation de déplacement qui représentent une partie importante dans le problème de transport routier. Le problème a été assimilé à un problème combinatoire de la classe NP-Complet. L'étude a été initiée par la formulation du modèle mathématique, une fonction d'optimisation multi-objectif globale a été proposée.
Avec le développement sociale, les problèmes sont devenus très complexes se qui à permet l'invention de plusieurs méthodes de résolution.
L'impossibilité technique de résoudre exactement les problèmes NP-Complet de grande taille et/ou des problèmes avec plus de deux objectifs, impose l'utilisation des heuristiques et en particulier les métaheuristiques. Néanmoins, les méthodes exactes peuvent être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet. Ces travaux sont généralement efficaces, car les deux méthodes coopérant ont alors des particularités bien différentes, et associent leurs avantages afin d'obtenir de bons résultats.

L'objectif de ce travail est de développer un outil d'aide à la décision pour le transport routier. on a cherché de résoudre le problème multi-objectif optimisant la durée totale des déplacement . L'approche proposée est l'hybridation des algorithmes métaheuristiques avec les algorithmes exactes.
Les algorithmes sont : l'algorithme génétique, la recherche tabou, l'algorithme de Dijkstra.
Pour choisir la meilleure approche, une étude comparative a été réalisée entre les différents

algorithmes. Les résultats numérique ont montré que l'approche de l'hybridation est la meilleure.

Dans nos perspectives globales, nous envisageons d'intégrer notre travail avec d'autre travaux. Dans une plate-forme complète de transport . Cette plate-forme vise à optimiser au mieux le domaine de transport par l'affichage des itinéraires les plus court, on fournissant tous les informations nécessaire pour l'utilisateur comme les distances parcourue, le taux d'énergie consommé, le temps nécessaire pour atteindre l'arrivée, ect. On utilisant les informations fournit par objets connectés que se soit l'utilisateur ou son environnement.

Bibliographie

[1] Alexis sofia. "Modélisation des réseaux de transports collectifs métropolitains pour une structuration des territoires par les réseaux". PhD thesis, Université des Sciences et Technologies de Lille, France, Mars 2010.

[2] Bouamrane salim, Beghdadi Hadj Ali." Modélisation numérique d'un réseau de transport urbain". Mascara, Algérie, Avril 2012.

[3] http ://www.andi.dz/index.php/fr/secteur-de-transport/dv :juillet 2019.

[4] http ://mostafabenkacem.skyrock.com/3102997247-l'économie du transport et son impact sur le développement.html/ dv :Août 2019.

[5] Bouamroni mohammed. "Un système interactif d'aide à la décision pour la régulation d'un réseau de transport urbain bimodal : approche multiagent et raisonnement à base de cas". PhD thesis, Département d'informatique, Université d'Oran, Algérie, 2006.

[6] Belayachi Naima. "Étude et modélisation du fonctionnement d'un réseau de transport modèle (RTM)",thèse de magister,2012.pp 21.

[7] http ://www.presse-france.com/comment-reduire-limpact-technologique-de-vos-transports . DV :Août 2019.

[8] https ://journals.openedition.org/etudescaribeennes/1042/ :Les types d'impacts de la pollution automobile sur l'environnement. DV :Août 2019.

[9] Jennie Cugny-Seguin."Transports et environnement :comparaisons européennes", paris, Chromatiques Éditions, Avril 2009.

[10] P. lambard, C. sevaux. "Algorithmes de graphes". Eyrolles deuxième édition, 2003.

[11] I. Charon, A. Germa, O. Hudry." Méthodes d'optimisation combinatoire". Edition Masson, France (2006).

[12] Siarry. "Optimisation Multi-Objectif". Eyroll.2002.

[13] ZIDI Kamel, "Système Interactif d'aide au déplacement multimodal",2006 .

[14] MAHDI Karim, "Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte",2012.

[15] http ://www.alcandre.net/uploads/TD-dijkstra.pdf ,DV :Septembre 2019.

[16] http ://thesis.univ-biskra.dz/3447/3/2eme%20chapitre.pdf, DV :Septembre 2019, pp 37.

[17] A.Silia,"Métaheuristiques pour l'optimisation multiobjectif, Approches coopératives, prise en compte de l'incertitude et application en logistique". Thèse de doctorat de l'université Lille 1 - Sciences et Technologies, 2009.

[18] AMAZOUZ Sofiane,DJEDDAI Mahdi, "Optimisation multi-objectif pour un ordonnancement intégré des tâches de production et de maintenance", Mémoire de fin d'études en vue de l'obtention du Diplôme de Master en Informatique,2016.pp 34,38.

[19] Souier, M., "Métaheuristiques pour la manipulation de routages alternatifs en temps réel dans un Job Shop", Mémoire de Magister,Université Abou Bakr Belkaid, Tlemcen, 2009.

[20] I.H. Osman et C.N. Potts, "Simulated annealing for permutation flowshop scheduling" . OMEGA, vol. 17, pp. 551-552.

[21] D. G. Dannenbring, An evaluation of flow-shop sequencing heuristics . Management Science, vol. 23, pp. 1174.

[22] H. Lin and K. Yamashita. "Hybrid simplex genetic algorithm for blind equalization using RBF networks". Mathematics and Computers in Simulation, 2002.

[23] Alan R. McKendall Jr. "Hybrid ant systems for the dynamic facility layout problem. Computers and Operations Research", 2006.

[24] Talbi, "A taxonomy of hybrid metaheuristics". Journal of Heuristics 8(5), pp.541-543, 2002.

[25] D. Duvidier, "Etude de l'hybridation des méta-heuristiques, application à un problème d'ordonnancement de type jobshop", Thèse de Doctorat, université du littoral France, décembre 2000.

[26] Zahra Naji Azimi. "Hybrid heuristics for examination timetabling problem. Applied Mathematics and Computation", 2005.

[27] http ://www.linux-center.org/articles/9812/python.html/ :Présentation du langage, DV : Novembre2019.

[28] https ://developers.google.com/machine-learning/crash-course/classification/precision-and-recall ?hl=fr, DV : Novembre2019.

[29] https ://searchmobilecomputing.techtarget.com/définition/Android-Studio/ : Android Studio , DV : Novembre2019.

[30] https ://cours-informatique-gratuit.fr/dictionnaire/office-excel/ :Office Excel,DV : Novembre2019.

[31] https ://geoportail.wallonie.be/servicesweb-traitement/ :Services web de traitement de données,DV : Novembre2019.

Résumé

.....

Dans cette dernière décennie, les chercheurs visent a résoudre de nombreux problèmes difficiles rencontrés dans différents domaines en utilisant des métaheuristiques. Ces méthodes qui sont devenues très populaires grâce a leur simplicité de mise en œuvre sont enconstante évolution d'ou l'apparition de leur hybridation que ce soit entre elles ou avec d'autres méthodes d'optimisation. Cette technique est de plus en plus publiée sans toute fois justifier les raisons de l'hybridation. A travers ce mémoire, nous avons présenter une technique d'hybridation des métaheuristiques pour l'optimisation du problème de transport.
Pour bien illustrer nos resultats, une application d'une méthode hybridée de deux métaheuristiques (Algorithme génétique, Recherche tabou) avec une méthode exacte (Algorithme de Dijkstra) pour réaliser un Outil d'aide a la décision pour choisir le chemin optimale.
Mots clés : Métaheuristiques, Optimisation combinatoire, Aide a la décision, Route Optimale . . .

Abstract

.....

In this last decade, researchers aim to solve many problems difficult in different areas using metaheuristics. These methods which have become very popular thanks to their simplicity of implementation are in constant evolution of or the appearance of their hybridization whether with each other or with other optimization methods. This technique is more and more published without any justify the reasons for the hybridization. Through this brief, we have presented a Hybridization technique of metaheuristics for the optimization of the transport problem.
To illustrate our results, an application of a hybridized method of two metaheuristics (Genetic Algorithm, Taboo Search) with an exact method (Dijkstra Algorithm) to realize a decision-making tool to choose the optimal path. **Key words :** Metaheuristics, combinatorial optimization, Decision support, Optimal Route. . .. . .