الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ Dr. YAHYA FARES DE MEDEA
FACULTE DE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'informatique



Projet de Fin d'Etudes

Présenté pour l'obtention du diplôme

Master en Génie Électrique

Spécialité : Systèmes Intelligents et Communicants

THEME

Détermination du Chemin le Plus Rapide Point Vue Temps dans un Réseau Routier en Utilisant l'Algorithme de Colonie de Fourmis

Proposé et dirigé par :

Dr : BOUHEDDA Mounir

Préparé et présenté par :

BENGHERBIA Abderrahmane

Mémoire préparé au sein du Laboratoire des Systèmes Électroniques Avancés (LSEA) de l'Université de Médéa

دعاء

اللهم إنا نسألك إيمانا دائما وقلبا خاشعا وعلما نافعا ويقينا صادقا ودينا قيما، ونسألك دوام النجاة من كل بلية ونسألك دوام العافية ونسألك الغنى

يا رب لا تدعنا نصاب بالغرور إذا نجحنا ولا نصاب باليأس إذا فشلنا، بل ذكرنا دائما إن الفشل هو التجارب التي تسبق النجاح، وإن التسامح هو أكبر مراتب القوة وإن حب الانتقام هو أول مظاهر الضعف.

يا رب إذا أسأنا إلى الناس فأعطنا شجاعة الاعتذار وإذا أساء ألينا الناس فأعطنا شجاعة العفوي، يا رب إذا جردتنا من المال فاترك لنا الأمل وإذا جردتنا من النجاح فاترك لنا قوة الصبر يا رب العالمين

وصلى الله وسلم على سيدنا محمد بني الرحمة وكاشف الغمة

Dédicaces

Je dédie ce travail
A
Ma très chère mère et mon très cher père
Pour leur soutien, leur présences,
Leur amour qui m'a encouragé dans mes études
A
Mes chers frères
Haithem, Ibrahim
A
Mes deux grands-mères et grand-père que dieu les protège
A
Notre promoteur
Dr : BOUHEDDA Mounir

A Mes chers amis
Ishak, Oussama,
Raouf, Tahar, Riadh, Saidali, Djallal, Ahmed,
Mohamed, Nassim, Dhia, Lyes, Rafik, Oussama,
Meriem, Rabab, Yasmine
Et ma future femme
Insha Allah.

Pour leur encouragement, leur soutien et l'ambiance
Que nous avons vécu Pendant toute notre carrière d'étudiant

A cet effet, je tiens à remercier tous mes collègues
Tous mes amis et aussi A tous ceux qui ont une relation
De proche ou de loin avec la réalisation du présent rapport.

ABDOU

MERCI !

C'est un petit mot tout simple
Mais qui pèse lourd
Si mes lèvres l'expriment avec douceur
C'est qu'il prend naissance au fond de mon cœur
Un grand Merci, un petit Merci
Peu importe sa taille
Il n'a pas de dimension ...
Que ce soit dans la joie ou dans la tristesse !
C'est un signe de reconnaissance
Qui ne connait pas l'indifférence !
Merci !
Un petit mot qui fait du bien quand on le prononce
Un petit mot gracieux qui calme et réjouit
Merci ! Merci !
Merci de m'avoir permis de te dire,
De vous dire :
Merci !

Nous remercions en premier lieu, Dieu le tout puissant qui nous a accordé la volonté et le courage pour réaliser ce travail.
En second lieu, nous tenons à remercier chaleureusement notre encadreur de mémoire, Dr : BOUHEDDA Mounir pour ses conseils, sa disponibilité, sa gentillesse et surtout pour nous avoir proposé ce projet et avoir assuré le suivi.

Nous remercions du fond du cœur nos chers parents de tout ce qu'ils ont faisaient pour nous.

Nous présentons également notre plus sincère gratitude aux membres du jury qui ont bien voulu examiner et évaluer notre travail. Merci pour l'honneur que vous nous faites en participant à la soutenance.

Un grand MERCI à tous les enseignants de l'Université DR.YAHIA FARESS de MEDEA qui nous ont formés durant nos cinq années d'études.
Nous aimerions aussi remercier nos amis et camarades pour le soutien moral qu'ils nous ont apporté et qui nous a permis la force pour continuer. Qu'ils trouvent ici l'expression de notre gratitude.

Merci à tous.

Résumé

Le but de ce travail est la proposition d'un système intelligent qui détermine le chemin le plus court (point de vue temps) entre deux points. Une partie du réseau routier de la ville de Médéa a été choisi pour notre application. Comme première phase, Un algorithme de colonie de fourmis a été appliqué mais avec la mise en place de certains changements. En suite a été appliquée sur le réseau routier utilisant le Matlab.

Mots clés : Algorithme de colonie de fourmis, Chemin, Matlab, Plus court chemin, Point, Trafic routier.

ملخص

الهدف من هذا العمل هو اقتراح نظام يحدد أقصر طريق (من حيث الوقت) بين نقطتين، جزء من شبكة الطرق لمدينة المدية اختير لتطبيق نظامنا. كمرحلة أولى، استعملنا خوارزمية مستعمرة النمل ولكن بإحداث بعض التغييرات عليها حسب الحاجة المتطلبة. بعد ذلك تم تطبيقها على شبكة الطرق باستخدام Matlab.

كلمات مفتاحية: أقصر طريق، حركة مرور، خوارزمية مستعمرة النمل، طريق، نقطة، Matlab.

Abstract

The purpose of this work is a proposal of an intelligent system that determines the shortest path (time perspective) between two points, a part of road network in the city of Medea was chosen for our application. As first phase, an ant colony algorithm has been applied but with the establishment of some changes. By following an adaptation of the road network using Matlab.

Keywords: Ant Colony Algorithm, Matlab, Point, Road, Shortest Path, Traffic.

Sommaire

Sommaire VI

Liste des figures IX

Liste des tableaux XI

Introduction Générale 1

Chapitre I :Généralités sur la gestion du trafic routier 4

I. Introduction 5

II. Gestion du trafic routier 5

II.1. Définitions 5

II.2. Présentation du milieu urbain 5

II.3. Gestion du réseau routier 8

II.4. Mesures de gestion du trafic routier 9

III. Théorie du graphe appliquée à l'optimisation de la gestion de trafic routier 10

III.1. *Définition* 10

III.2. Intérêt des graphes 10

III.3. Algorithmes basés sur le principe d'optimalité 11

III.4. Les problèmes de chemins optimaux 12

III.5. Les grands types de problèmes 12

III.6. Les différents algorithmes 13

III.6.1. Algorithme de Kruskal 13

III.6.2. Algorithme de Bellman-Ford-Moore 13

III.6.3. Algorithme de Dijkstra 15

III.6.4. Algorithme génétique 16

III.7. Quelques exemples d'application dans les graphes 18

IV. Conclusion 20

Chapitre II : Algorithme de Colonie de Fourmis 21

I. Introduction 22

II. Définition 22
III. Historique 23
IV. Principe 23
V. Organisation de la méta- heuristique 24
VI. Origine 24
VII. Stigmergie 26
VIII. Les fourmis artificielles 27
IX. Description générale « Ant System » 29
X. L'Algorithme 30
XI. Conclusion 32
Chapitre III :Tsests et Résultats 33
I. Introduction 34
II. Présentation du problème 34
III. Algorithme de colonie fourmis modifié 34
III.1. Problème du voyageur de commerce 34
III.2. Les chemins existants 35
III.3. Le plus court chemin en termes de distance 40
IV. Appliqué l'algorithme au réseau routier de la ville de Médéa 45
IV.1. Présentation Réseau routier de la ville de Médéa 45
IV.2. Programmation dynamique 47
IV.3. Résultat de simulation sur le réseau routier de la ville de Médéa 49
IV.4. Résultats obtenus 53
IV.4.1. Choix du taux d'évaporation ρ 53
IV.4.2. Choix du nombre de fourmi approprié 54
IV.5. Interface graphique 54
V. Conclusion 57
Conclusion Générale 58
Référence Bibliographie 61
Références 62
Annexe 63

Liste des figures

Figure I.1. Les segments et les rues dans un milieu urbain.....6
Figure I.2. La position GPS ou coordonnée géographique est une information commune entre les quatre segments.....6
Figure I.3. La position GPS ou coordonnée géographique est une information commune entre les trois segments.....6
Figure I.4. Les quatre positions GPS coordonnée géographique dans un Rond-Point.....7
Figure I.5. Les segments à une seule voie (mono direction).....7
Figure I.6. Les segments à deux voies (bidirectionnel).....7
Figure I.7.quatres quartiers de la ville de Königsberg.....18
Figure I.8. quelques ville français.....19
Figure I.9. Organisation d'une session d'examens.....20
Figure II.1. Principe de l'algorithme de colonies de fourmis.....24
Figure II.2. Exemple de l'algorithme de colonies de fourmis.....26
Figure III.1. représentation des points et distances entre eux.....35
Figure III.2 . chemin parcouru dans l'exemple III.2.....36
Figure III.3. nouveau chemin.....37
Figure III.4. simulation MATLAB de la solution proposé à l'exemple III.2.....38
Figure III.5. chemin finale.....38
Figure III.6. simulation MATLAB de l'exemple III.3.....39
Figure III.7. simulation MATLAB de l'exemple III.4.....40
Figure III.8. simulation MATLAB de l'exemple III.5.....42
Figure III.9. chemins parcourus possibles.....43
Figure III.10. les chemins après décomposition.....43
Figure III.11. simulation MATLAB de l'exemple III.6.....44
Figure III.12. Représentation schématique d'une partie du réseau routier de la ville de Médéa.....46
Figure III.13. organigramme de l'algorithme proposé.....48
Figure III.14. simulation MATLAB de l'exemple III.7.....49
Figure III.15. simulation MATLAB de l'exemple III.8.....50
Figure III.16. simulation MATLAB de l'exemple III.8.....51
Figure III.17. Résultat des exemples III.7 et III.8 sur le réseau routier.....52
Figure III.18. interface de l'application.....55
Figure III.19. afficher la ville de Médéa sur l'interface.....56
Figure III.20 : trouver le plus court chemin sur l'interface.....56

Liste des tableaux

Tableau III.1 : les valeurs de données pour la simulation de l'exemple III.3. 39

Tableau III.2 : les valeurs de données pour la simulation d'exemple III.4. 40

Tableau III.3 : les valeurs de données pour la simulation d'exemple III.5. 42

Tableau III.4 : les valeurs de données pour la simulation d'exemple III.6. 44

Tableau III.5 : les valeurs de données pour la simulation d'exemple III.7. 49

Tableau III.6 : les valeurs de données pour la simulation d'exemple III.8. 50

Tableau III.7 : les valeurs de données pour la simulation d'exemple III.8. 51

Tableau III.8 : des valeurs aléatoires des points pour faire des tests. 53

Tableau III.9 : résultat des tests pour plusieurs valeurs d'évaporation. 53

Tableau III.10 : résultat des tests pour différents nombres de fourmis. 54

Introduction Générale

La circulation routière est le déplacement de véhicules automobiles sur une route. Se déplacement est devenu un aspect essentiel de la vie quotidienne : qu'il s'agisse de transports en commun ou de véhicules personnels, le vaste réseau formé de ces moyens de locomotion est immensément complexe à gérer. Dans ce document, nous nous intéressons de trouvé le plus court chemin en terme de temps [1].

Le trafic routier urbain est au cœur de nombreuses problématiques : plus encore ces dernières années, cet aspect critique intervenant au quotidien est défavorable à de nombreux domaines, tels que l'économie ou encore l'écologie. Pour ces raisons, les systèmes de transport intelligents (STI) sont apparus depuis la fin des années 1990, permettant d'optimiser au mieux les dépenses et l'expérience de l'utilisateur sur des réseaux souvent complexes. Parmi les problèmes les plus anciens et les plus traités on trouve le problème de plus court chemin. Il consiste à rechercher le meilleur chemin entre des points, sources et destinations, afin de minimiser un critère précis, généralement le temps et/ou le coût de trajet. Le recueil des données de trafic se fait grâce à l'utilisation de capteurs et de détecteurs. L'objectif est d'acquérir des données que les gestionnaires utiliseront en temps réel ou différé. Le traitement de ces données a pour but de surveiller et de gérer le trafic, de diffuser de l'information routière à destination des usagers et également d'établir des statistiques et des études de trafic [2].

Le problème du plus court chemin est un problème classique de programmation réseau qui a souvent été étudié et consiste à déterminer le chemin de coût minimum en terme de temps reliant une ville à une autre ville. L'objectif de notre projet est de modéliser diverses situations par des graphes pondérés connexes pour résoudre des problèmes d'optimisation, Optimiser une grandeur (distance, durée, etc.) et utiliser un algorithme pour trouver le plus court chemin optimal en terme de temps.

Nous présentons dans notre travail d'étude des nouvelles méta-heuristiques permettant de résoudre des problèmes tels que le voyageur de commerce, en s'inspirant du comportement social des fourmis [3]. En utilisant une colonie de fourmis qu'elle résout des problèmes de la recherche du plus court chemin en terme de temps entre une source et destination. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes.

Dans le cadre de la présente étude, notre mémoire a été scindé en trois parties principales.

La première partie présente la synthèse des notions étudiées. Cette partie sera abordée en chapitre I où une vue sur la présentation du milieu urbain, gestion du réseau routier, et Mesures de gestion du trafic routière.

Dans le chapitre II on a défini l'algorithme de colonie de fourmis, leur principe et ces étapes de l'algorithme.

Dans le chapitre III on a choisi la ville de Médéa pour appliquer l'algorithme de colonie de fourmis et résoudre notre problème en utilisant le logiciel MATLAB.

A la fin notre mémoire sera achevé par une conclusion générale et des perspectives.

Chapitre I :

Généralités sur la gestion du trafic routier

I. Introduction

Le problème de circulation est du essentiellement à un déséquilibre entre l'offre (c.-à-d. routes et de leur capacité) et la demande (à savoir, les besoins de voyage). Chaque fois que ce décalage augmente, les pouvoirs publics essayent d'équilibrer par la création des infrastructures réalisent des nouvelles routes.

II. Gestion du trafic routier

L'accélération de l'urbanisation et la congestion des réseaux routiers qui en résulte obligent les responsables politiques et les urbanistes à revoir l'urbanisme et leurs politiques de transport afin de répondre aux besoins futurs en anticipant les évolutions sociales et démographiques.

II.1. Définitions

L'urbain : agglomération d'une certaine importance, à l'intérieur de laquelle la plupart des habitants ont leur travail dans le commerce, l'industrie ou l'administration.

Le réseau routier : est l'ensemble des voies de circulation terrestres permettant le transport par véhicules routiers, et en particulier, les véhicules motorisés (automobiles, motos, autocars, poids lourds...).

Une rue : est une « voie bordé, au moins en partie, de maisons dans une agglomération».

Un segment : réunit tous les immeubles appartenant à un même îlot et situés en bordure d'une même rue, d'une même avenue ou d'une place.

Le Système d'information Géographique : est une méthode d'enregistrement, de gestion et d'extraction d'information relative à l'environnement terrestre et ayant une forme structurée. Cette information est définie, entre autre, par ses paramètres de localisation spatiale permettant de la manipuler et de la cartographier de plusieurs manières [4].

II.2. Présentation du milieu urbain

La route est organisée en réseau. Un réseau est un "ensemble de lignes connectant des nœuds pour la desserte d'un territoire" [4].

Les lignes sont représentées par les routes et les chemins qui desservent les lieux du territoire (Figure I.1).

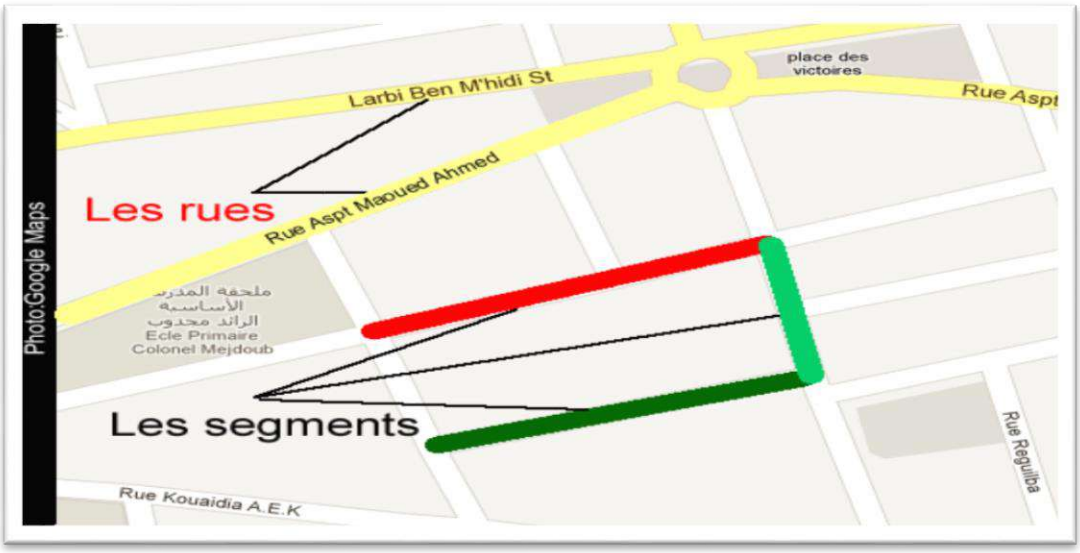


Figure I.1. *Les segments et les rues dans un milieu urbain*

Les nœuds étant les intersections se localisent par des coordonnées géographiques (C.G) ou GPS (Global Positioning System) (Figure I.2, I.3, I.4).

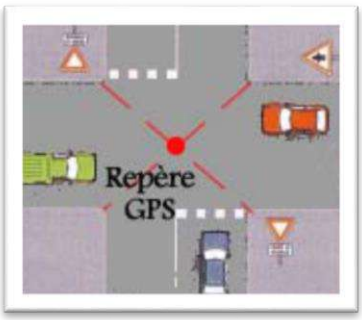


Figure I.2.*La position GPS ou coordonnée géographique est une information commune entre les quatre segments.*



Figure I.3. *La position GPS ou coordonnée géographique est une information commune entre les trois segments.*

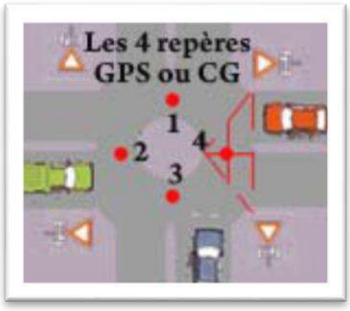


Figure I.4.*Les quatre positions GPS coordonnée géographique dans un Rond-Point.*

Le nœud d'arrivée (localisé par son GPS ou Coordonnée géographique) d'un segment orienté (mono directionnel) est un nœud de départ pour ses voisins (Figure I.5).

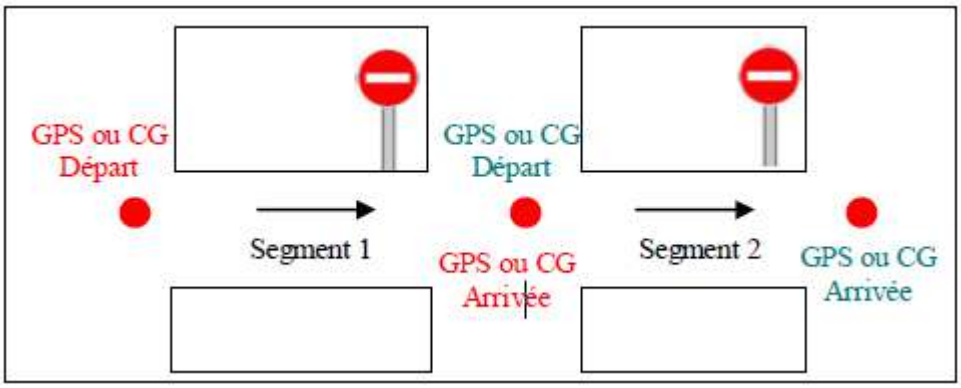


Figure I.5.*Les segments à une seule voie (mono direction)*

Segment non orienté (bidirectionnel) ses deux nœuds se sont les nœuds de départ et d'arrivée en même temps (Figure I.6) [4].

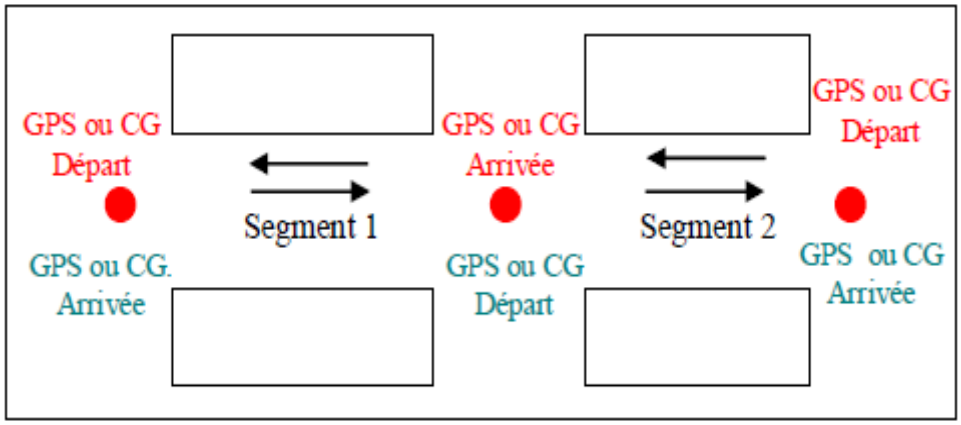


Figure I.6.*Les segments à deux voies (bidirectionnel).*

II.3.Gestion du réseau routier

Les systèmes de gestion des réseaux routiers assurent deux types de gestion qui sont :

- La gestion de circulation : vise uniquement à intervenir directement sur le comportement de chaque usager en présence d'une difficulté.
- La gestion du trafic : cherche à déplacer ou à modifier des flux de circulation, parfois à longue distance des problèmes rencontrés, pour optimiser globalement la gestion du réseau routier.

La gestion de la circulation routière est une activité aujourd'hui largement répondue sa nécessité est en effet reconnue par la majorité des collectivités dans le monde. Son visage prend les traits d'une panoplie d'objets et de procédés familiers : panneaux de signalisation, marques au sol, règles de conduite inscrites dans le code de la route, aménagements routiers, réglementations relatives aux véhicules automobiles, …etc [4].

La signalisation routière est composée de panneaux, du marquage au sol et des feux. Elle permet d'informer l'usager des règles en vigueur et de l'orienter dans ses déplacements. Bien conçue et réalisée, elle réduit les causes d'accident et facilite la circulation.

La modélisation du trafic routier est un sujet relativement ancien, puisque les premiers travaux sur le sujet remontent à 1935. Ces travaux ont été à l'origine menés essentiellement par des ingénieurs soucieux de mieux maîtriser et optimiser les flux de trafic s'écoulant sur les grands réseaux routiers, soucieux également de mieux dimensionner les infrastructures nouvelles par une meilleure prévision des flux et des capacités.

L'élaboration de la théorie du trafic a permis de mieux comprendre et prévoir les interactions inter véhiculaires ainsi que le mouvement global du flux au niveau des réseaux autoroutiers. Chaque niveau de détail est caractérisé par des variables et des paramètres spécifiques. Lorsque les variables sont utilisées pour décrire le véhicule individuel et pour exprimer le comportement de chaque conducteur, on parle des variables microscopiques. La description du comportement global du flux de véhicules fait appel à des variables dites macroscopiques.

- Variables microscopiques du trafic : ce sont des variables liées à un véhicule individuel et sont interaction avec son environnement. Ces variables sont exprimées pour chaque véhicule et sont étroitement liées à sa dynamique. (Exp. Espace inter – véhicule, Temps inter véhiculaire, Vitesse du véhicule, Accélération instantanée).

- Variables macroscopiques du trafic : ces variables permettent de décrire l'état du flux de trafic d'une manière agrégée. (ex. Débit du trafic, La densité moyenne (appelée concentration), Taux d'occupation, Vitesse moyenne du flux de trafic).

Les informations mesurées concernent le nombre de véhicules (poids lourd, véhicules légers), et parfois la vitesse moyenne pratiquée. Les données sont exprimées en Trafic Moyen Journalier – Annuel (TMJ/A) pour l'ensemble des véhicules et pour les poids lourds. Le trafic tous véhicules correspond à la moyenne journalière de la fréquentation de tous les véhicules confondus circulant en un point de comptage donné pour tous les jours de l'année [4].

II.4. Mesures de gestion du trafic routier

La mise en œuvre effective de toute action d'exploitation s'appuie sur la connaissance des trafics et des perturbations qui peuvent être constatées, quel que soit le type de réseau : urbain, de rase campagne ou autoroutier.

Les mesures de trafic se sont progressivement développées pour répondre à différents objectifs :

- ✓ adéquation entre offre et demande, support des études de trafic.
- ✓ politiques d'entretien des réseaux.
- ✓ stratégies de régulation et gestion des flux en temps réel.
- ✓ détection des perturbations.
- ✓ évaluation des nuisances et suivi d'indicateurs de sécurité routière.
- ✓ évolution de la circulation des poids lourds et des vitesses pratiquées.
- ✓ information des usagers.

Pour répondre à ces demandes, on a vu émerger ces dernières années de nouveaux capteurs et de nouvelles techniques de recueils de données. Concernant la technique qu'on a choisi c'est la technique de plus court chemin [5].

III. Théorie du graphe appliquée à l'optimisation de la gestion de trafic routier

Le problème du plus court chemin est un problème classique de programmation réseau qui a souvent été étudié. Quand on veut aller d'un point A à un point B, expérimen affirme que le chemin le plus court est la ligne droite. Malheureusement, il n'est pas toujours possible de transposer ce résultat géométrique dans les cas concrets. Ainsi, on se contente souvent de suivre les axes routiers. Dans ce cas comment trouver le plus court chemin entre deux villes ?

Ce problème se modélise très simplement à l'aide des schémas. Les sommets du schéma sont les intersections entre les différentes routes et les arcs représentent les tronçons routiers. Notons que l'usage des schémas orientés permet de distinguer les voies à double sens des voies à sens unique. Il faut maintenant modéliser la distance entre les villes sur le schéma [6].

III.1. *Définition*

La résolution du problème des k plus courts chemins fait appel à la théorie des graphes. Un réseau N est alors vu comme une liste comportant n éléments appelés nœuds ou sommet (vertex, notés vi) et m éléments nommés liens ou arcs notés ai.

N est une séquence alternant nœuds et arcs de la forme {s = v1, a1, v2, a2 …, vr = t}.

On dit qu'un chemin de s à t ne forme pas de boucle si tous les nœuds de ce chemin sont différents.

Enfin, deux chemins sont dits disjoints si tous les nœuds empruntés par le premier chemin sont différents de ceux empruntés par le second [7].

III.2. Intérêt des graphes

Les graphes représentent un instrument puissant pour modéliser de nombreux problèmes combinatoires, qui seraient sans cela difficilement abordables par des techniques classiques comme l'analyse mathématique.

En plus de son existence en tant qu'objet mathématique, le graphe est aussi une structure de données puissante pour l'informatique.

Les graphes sont irremplaçables dès qu'il s'agit de décrire la structure d'un ensemble complexe, en exprimant les relations, les dépendances entre ces éléments. Des exemples sont les diagrammes hiérarchiques en sociologie, les arbres génétiques, et les diagrammes de succession de tâches en gestion de projet.

Un autre grand groupe d'utilisation est la représentation de connexion et de possibilités de cheminement : détermination de la connexité d'un réseau quelconque (électrique, d'adduction d'eau), calcul de plus court chemin dans un réseau routier, Les graphes sont enfin précieux pour décrire des systèmes dynamiques, c'est-à-dire qui évoluent dans le temps : diagrammes de transition, automates d'état finis, chaînes de Markov.

Il existe deux familles des graphes : orientés et non orientés c'est-à-dire que les relations entre les éléments d'un ensemble sont orientés ou non.

Un graphe est défini par un couple G = (X ; U) de deux ensembles :

² X est un ensemble (x1 ; x2; : : : xN) de sommets, également appelés nœuds ;

² U = (u1 ; u2; : : : uM) est une famille de couples ordonnés de sommets appelés arêtes (arcs).

Un graphe peut être value, ou pondéré, il est alors muni de poids ou coût sur ses arêtes [8].

III.3. Algorithmes basés sur le principe d'optimalité

Ce principe affirme qu'il existe un plus court chemin formé par de plus courts chemins élémentaires.

Pour résoudre le problème du plus court chemin en utilisant cette classe d'algorithmes, nous pouvons nous baser sur la généralisation des algorithmes à étiquetage (labeling algorithms) utilisés pour le problème du plus court chemin.

Les algorithmes à étiquetage ont en commun l'utilisation d'une ou plusieurs étiquettes (labels) assignées à chaque nœud. Alors que pour le problème du plus court chemin il n'y a qu'une seule étiquette assignée à un nœud, dans le cas des k plus courts chemins il peut y avoir une ou plusieurs étiquettes assignées à un seul et même nœud.

Une étiquette permet de reconstituer les chemins auxquels appartient le nœud. Pour un nœud quelconque i, elle contient les informations suivantes :

- la distance du chemin partant de s jusqu'à i.
- le nœud précédent i dans ce chemin.

Ces informations suffisent pour parcourir le réseau puis reconstituer les k plus courts chemins en partant de la destination et en remontant jusqu'à la source.

Comme il peut y avoir plusieurs étiquettes associées à un même nœud, chacune contenant des informations différentes, il faut définir une fonction, notée h, permettant de retrouver le nœud auquel appartient l'étiquette [7].

III.4. Les problèmes de chemins optimaux

Les problèmes de chemins optimaux sont très fréquents dans les applications pratiques. On les rencontre dès qu'il s'agit d'acheminer un objet entre deux points d'un réseau, de façon à minimiser un coût, une distance ou une durée. Ils apparaissent aussi en sous-problèmes combinatoires, notamment les flots dans les graphes et les ordonnancements. Tout ceci a motivé très tôt la recherche d'algorithmes efficaces [8].

III.5. Les grands types de problèmes

Considérons un graphe value G = (X;A;W). X désigne un ensemble de N sommets (ou nœuds) et A un ensemble de M arêtes. W (i, j) aussi noté Wi ; j, est 'évaluation (aussi appelée poids ou coût) de l'arête (i, j), par exemple une distance, un coût de transport, ou temps de parcours. Pour la fonction économique la plus répandue, le coût d'un chemin entre deux sommets est la somme des coûts de ces arêtes. Les problèmes associés consistent à calculer des chemins de coût minimal (en abrégé chemins minimaux, ou plus courts chemins). Ils ont un sens si G n'a pas un circuit négatif, sinon on pourrait diminuer infiniment le coût d'un chemin en tournant dans un tel circuit, appelé pour cette raison circuit absorbant.

Bien entendu on peut aussi chercher des chemins de valeurs maximales. En l'absence de circuit absorbant, on peut restreindre la recherche des plus courts chemins aux seuls chemins élémentaires, c'est-à-dire ne passant pas deux fois par un même sommet. En effet, si un chemin emprunte un circuit, on peut enlever la portion passant par le circuit sans augmenter le coût. Le problème de recherche d'un chemin optimal en présence de circuits absorbants existe, mais il est NP-difficile. Il existe d'autres fonctions économiques que la somme des coûts des arêtes, mais elles sont moins répandues dans les applications.

La littérature distingue trois types de problème, que nous notons A, B et C :

A. **Plus court chemin à origine unique** étant donné un sommet de départ s. Trouver un plus court chemin de s vers tout autre sommet.
B. **Plus court chemin pour un couple de sommets** étant donné deux sommets s et t. Trouver le plus court chemin de s à t.
C. **Plus court chemin pour tout couple de sommets** trouvé un plus court chemin entre tout couple de sommets, c'est-à-dire calculer une matrice N £ N appelée Distancier **[8]**.

III.6. Les différents algorithmes

III.6.1. Algorithme de Kruskal

L'algorithme de Kruskal est le utilisé dans la théorie des graphes pour construire une arborescence de chemin minimal. Il peut être résumé par le pseudocode suivant [9]:

1) Calculer toutes les distances entre les sommets.
2) Trier ces distances par ordre croissant les mettre dans un tableau
3) Inclure la plus courte liaison entre deux sommets (i, j) dans l'arbre
4) Enlever cette liaison entre (i, j) du tableau
5) SI l'inclusion de la liaison (i, j) forme un cycle ALORS Enlever cette liaison (i,j) de l'arbre
6) Aller au 3
7) Fin.

Inconvénients :

Il se trouve que cet algorithme implique une connaissance totale des distances qui séparent des sommets deux à deux. Ce n'est pas tout car d'une part, à chaque itération il faut vérifier constamment si la nouvelle liaison que l'on vient d'insérer dans l'arbre ne forme pas un cycle. Et d'autre part, à l'étape 6, nous devons aussi vérifier si l'arbre n'est pas connexe. De plus, il faut ensuite parcourir tout le tableau de distances pour trouver le plus court chemin entre deux sommets. Dans tous les cas, cet algorithme nous demande une taille de mémoire importante pour stocker toutes ces informations.

Au moindre changement de la carte, il faut réitérer l'algorithme afin de maintenir des informations à jour. Or, les niveaux de trafics varient constamment dans le jeu. Ceci posera donc un sérieux problème de performance [9].

Avantage :

Le plus court chemin entre deux sommets quelconques est connu à tout instant [9].

III.6.2. Algorithme de Bellman-Ford-Moore

Cet algorithme peut être vu comme une version améliorée de l'algorithme proposé par Ford. Cette version est aussi appelée l'algorithme à correction d'étiquette. Ci-dessous se trouve son pseudo-code [9] :

1) Initialisation pour tout nœud
   a. distance = infinie
   b. père = NULL
2) Initialisation du nœud de départ

a. Distance = 0
b. père = NULL

3) Enfiler (L, nœud de départ)
4) Tant que A non NULL Faire
   a. Nœud à traiter A = défiler(L)
   b. Pour chaque nœud B relié à A Faire
      - ✓ distance D = distance de A + longueur de l'arc (A, B)
      - ✓ Si distance actuelle de B > D Alors
        - distance de B = D
        - père de B = A
        - Enfiler (, B)
      - ✓ Fin Si
   c. Fin Pour
5) Fin Tant que.

Inconvénients :

À partir du nœud de départ, cet algorithme se réitère jusqu'à ce qu'il ne trouve plus de nœud dont la distance (ou étiquette) peut être améliorée. Dans de nombreux cas, il examine plusieurs fois un nœud déjà visité. Ceci pose aussi un problème de performance non négligeable.

De plus, le danger d'un tel algorithme est le fait d'être pris dans le piège d'un cycle dont la somme des distances (ou étiquettes) est négative. Puisque l'algorithme compare la distance de chaque nœud, cette dernière se trouve être améliorée continuellement dans un tel cycle [9].

Avantage :

En insérant astucieusement une instruction de comparaison entre l'étape 1 et 2, on peut terminer l'algorithme dès que l'on tombe sur le nœud d'arrivé. Ceci permet d'éviter de parcourir tous les nœuds pour trouver un chemin. Cependant, en court-circuitant l'algorithme ainsi, on risque de passer à côté d'un autre chemin plus optimal.

Un autre avantage non négligeable est sa capacité à travailler avec les arcs de poids négatifs. Néanmoins, l'algorithme doit être capable de détecter les cycles d'arcs de poids tous négatifs comme cité précédemment [9].

III.6.3. Algorithme de Dijkstra

Dijkstra a eu l'idée de marquer chaque nœud visité et empêché ainsi son algorithme de revenir en arrière. Ci-dessous ce trouve le pseudo-code de l'algorithme de Dijkstra adapté au jeu de simulation Open City [9].

1) Initialisation de toutes les cases « routes »
   a. La case incidente est indéterminée
   b. La longueur du trajet vaut plus infini

2) Initialisation de la case de départ
   a. La case incidente est indéterminée
   b. La longueur du trajet vaut 0
   c. Insertion de cette case dans la liste A des cases à traiter
3) Tant que A est non vide ET non trouvé Faire
   a. Trier la liste A
   b. Sortir la case m ayant la longueur minimale
   c. Si la case à traiter est le point d'arrivé Alors trouvé est vrai
   d. Pour chaque case voisine v non marquée de la case à traiter Faire
      - ✓ Nouvelle longueur l = longueur actuelle de m + le trafic en v
      - ✓ Si la nouvelle longueur l < la longueur actuelle de v Alors
        - La longueur de v est maintenant l
        - La case incidente de v est m
        - Insérer v dans la liste A des cases à traiter
      - ✓ Fin Si
   e. Fin Pour
   f. Marquer la case m comme traitée
   g. Insérer m dans la liste B des cases déjà traitées
4) Fin Tant que
5) Construire le chemin minimal en parcourant B.

Inconvénients :

Avec cette version, nous devons réitérer l'algorithme pour chaque véhicule présent sur la carte, ce qui le rend gourmand en temps de calcul. Nous pensons que si le nombre de véhicules est important et la carte à traiter est relativement petite, nous pourrons faire fonctionner l'algorithme sur toute la carte et obtenir ainsi l'ensemble des chemins optimaux entre deux cases

quelconques. Ensuite, il nous reste plus qu'à reconstituer le chemin optimal pour chaque véhicule.

Cet algorithme est de type glouton[1].

C'est pourquoi dans un certain nombre de cas où les cases à traiter ont un même niveau de trafic l'algorithme n'est pas très performant car il les parcourt toutes. C'est pareil si nous prenons un autre critère de tri comme la distance entre la case à traiter et la case d'arrivée.

Puisque chaque nœud traité est marqué, nous ne pouvons pas revenir en arrière. De ce fait, l'algorithme de Dijkstra est inadapté quand nous avons des arcs de poids négatif. Dans ce genre de situation, l'algorithme doit pouvoir revenir en arrière pour modifier les étiquettes. C'est le cas de l'algorithme Bellman-Ford-Moore [9].

Avantage :

Avec cet algorithme, nous pouvons nous arrêter dès que le point d'arrivé devient la case à traiter puisque le chemin que nous avons construit jusque-là est un chemin minimal.

Le système de marque empêche l'algorithme de revenir en arrière. Il améliore ainsi sa performance [9].

III.6.4. Algorithme génétique

Les algorithmes génétiques (AGs) sont des algorithmes d'optimisation stochastiques itérés fondés sur les mécanismes de la sélection naturelle et de la génétique.

Leur fonctionnement est relativement simple :

1. Nous partons avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies
2. Nous évaluons leur performance (fitness) relative
3. Sur la base de ces performances, nous créons une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation.
4. Nous recommençons ce cycle jusqu'à ce que nous trouvions une solution satisfaisante.

[1] Un **algorithme glouton** (*greedy algorithm* en anglais, parfois appelé aussi algorithme gourmand) est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local. Dans certains cas cette approche permet d'arriver à un optimum global, mais dans le cas général c'est une heuristique.

Inconvénients :

- Les algorithmes génétiques ne sont encore actuellement pas très efficaces en coût (ou vitesse de convergence), vis-à-vis de méthode d'optimisation plus classiques.
- Parfois les algorithmes génétiques convergents très vite vers un individu particulier de la population dans la valeur d'adaptation est très élevée.
- Le respect de la contrainte de domaine par la solution codée sous forme de chaine de bits pose parfois problème. Il faut bien choisir le codage, voire modifier les opérateurs.
- L'utilisation d'un algorithme génétique ne garantit pas le succès de l'optimisation.
- En pratique l'efficacité d'un AG dépond souvent de la nature du problème d'optimisation. Solon les cas de choix des opérateurs et des paramètres seront souvent critiques, mais aucune théorie générale ne permet de connaitre avec certitude la bonne paramétrisation, il faudra faire plusieurs expériences pour s'en approcher **[10]**.

Avantage :

- Les algorithmes génétique opèrent au niveau du codage des paramètres sans se soucier de leur nature, donc ils s'appliquent à de nombreuses classes de problèmes, qui dépendent éventuellement de plusieurs paramètres de natures différentes (booléens, entiers, réels, fonctions...).

- Pour les mêmes raisons un algorithme génétique est dans l'idéal totalement indépendant de la nature du problème et de la fonctionnelle à optimiser, car il ne se sert que des valeurs d'adaptation, qui peuvent être très différentes des valeurs de la fonction à optimiser, même si elles sont calculées à partir de cette dernière.

- Potentiellement les algorithmes génétiques explorent tous l'espace des points en même temps, ce qui limite les risques de tomber dans des optimums locaux.

- Les algorithmes génétiques ne se servent que des valeurs de la fonctionnelle pour optimiser cette dernière, il n'y a pas besoin d'effectuer de coûteux et parfois très complexes calculs.

- Les algorithmes génétiques présentent une grande robustesse c'est-à-dire une grande capacité à trouver l'optimum globeux des problèmes d'optimisation **[10]**.

III.7. Quelques exemples d'application dans les graphes

Exemple 1 : *Le problème des ponts de Königsberg*

La ville de Königsberg comprenait 4 quartiers, séparés par des ponts. Les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir à leur point de départ.

Le plan de la ville peut se modéliser à l'aide d'un graphe ci-dessous, les quartiers sont représentés par les 4 sommets, les 7 ponts par des arêtes [11]:

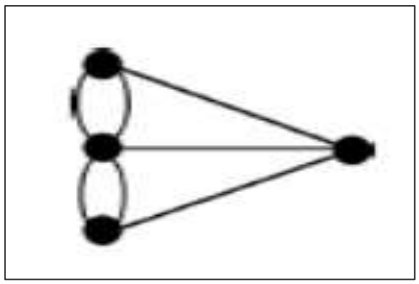


Figure I.7.*quatres quartiers de la ville de Königsberg.*

Exemple 2 : *Choix d'un itinéraire*

Sachant qu'une manifestation d'étudiants bloque la gare de Poitiers, et connaissant la durée des trajets suivants [11] :

Bordeaux ⟶ Nantes 4 h

Bordeaux ⟶ Marseille 9 h

Bordeaux ⟶ Lyon 12 h

Nantes ⟶ Paris-Montparnasse 2 h

Nantes ⟶ Lyon 7 h

Paris Montparnasse ⟶ Paris Lyon 1 h (en autobus)

Paris-Lyon ⟶ Grenoble 4 h 30

Marseille ⟶ Lyon 2 h 30

Marseille ⟶ Grenoble 4 h 30

Lyon ⟶ Grenoble 1 h 15

Question :

Comment faire pour aller le plus rapidement possible de Bordeaux à Grenoble ?

Les données du problème sont faciles à représenter par un graphe dont les arêtes sont étiquetées par les durées des trajets [11]:

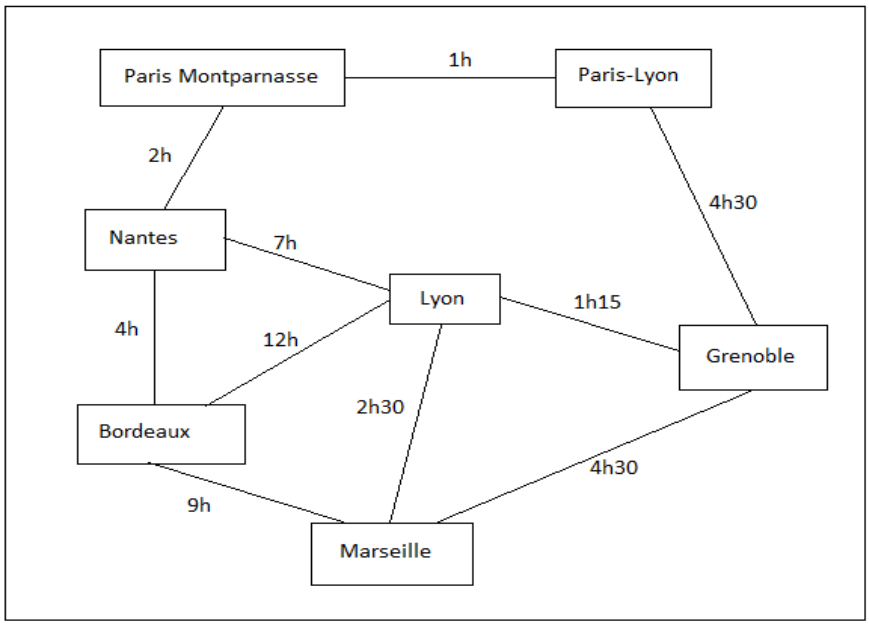


Figure I.8. *quelques ville français.*

Il s'agit de déterminer, dans ce graphe, le plus court chemin entre Bordeaux et Grenoble.

Exemple 3 : *Organisation d'une session d'examens*

Des étudiants A, B, C, D, E et F doivent passer des examens dans différentes disciplines, chaque examen occupant une demi-journée :

Chimie : étudiants A et B

Électronique : étudiants C et D

Informatique : étudiants C, E, F et G

Mathématiques : étudiants A, E, F et H

Physique : étudiants B, F, G et H

On cherche à organiser la session d'examens la plus courte possible.

On peut représenter chacune des disciplines par un sommet, et relier par des arêtes les sommets correspondant aux examens incompatibles (ayant des étudiants en commun) :

Il s'agit alors de colorier chacun des sommets du graphe en utilisant le moins de couleurs possible, des sommets voisins (reliés par une arête) étant nécessairement de couleurs différentes [11].

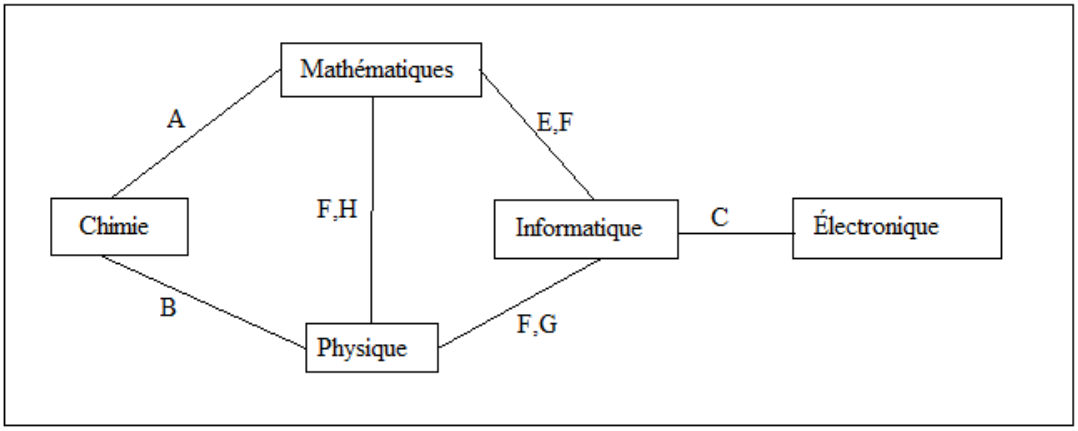


Figure I.9. *Organisation d'une session d'examens.*

IV. Conclusion

A partir d'une problématique qui consiste à déterminer un chemin optimal entre deux points, nous avons travaillé sur des divers algorithmes afin de trouver la solution la plus adaptée.

En effet, il existe plusieurs algorithmes permettant d'obtenir le fameux chemin optimal. Cependant, on en étudiant de façon plus approfondie, nous avons pu déterminer leurs inconvénients et leurs avantages par rapport aux besoins du simulateur. Il se trouve que l'algorithme de colonie de fourmis est le plus adapté pour notre problème.

Chapitre II : Algorithme de Colonie de Fourmis

I. Introduction

Le comportement des fourmis est un comportement collectif. Chaque fourmi a pour priorité le bien être de la communauté. Chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre. Ce concept est appelé Hétérarchie (s'opposant à la Hiérarchie) chaque individu est aidé par la communauté dans son évolution et en retour il aide au bon fonctionnement de celle-ci. La colonie est donc autocontrôlée par le biais de mécanismes relativement simples à étudiera [12].

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résoudre des problèmes tels que celui de la recherche du plus court chemin. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes.

On imagine alors un voyageur de commerce qui doit réaliser sa tournée en visitant une et une seule fois l'ensemble des villes. Il souhaite déterminer à la suite le plus court chemin possible pour aller à la ville demandé.

II. Définition

Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de méta heuristiques d'optimisation. Initialement proposé par Marco Dorigo[2]. Dans les années 1990, pour la recherche de chemins optimaux dans un graphe, le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes et plusieurs algorithmes ont vu le jour, s'inspirant de divers aspects du comportement des Fourmies [12].

[2] **Marco Dorigo**, né le 26 août 1961 à Milan, est directeur de recherche au Fonds national de la recherche scientifique (F.R.S.-FNRS), et codirecteur de l'Institut de recherches interdisciplinaires et de développements en intelligence artificielle (IRIDIA) à l'Université libre de Bruxelles.

III. Historique

Dans les sections qui viendront plus tard, nous présenterons la méta-heuristique ACO, pour le "Ant Colony Optimization". Toutes ces idées abstraites sont inspirées des travaux de Deneubourg sur les fourmis.

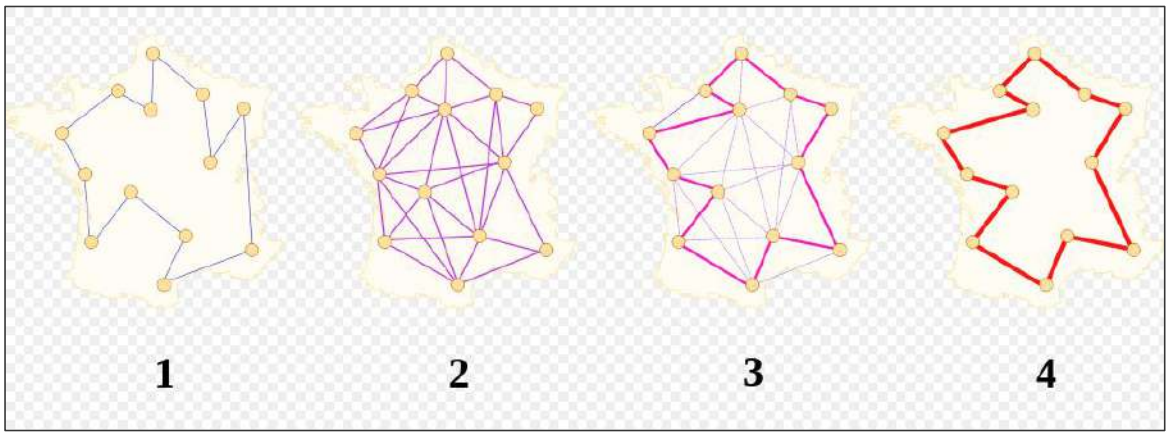
Cette méta-heuristique est relativement récente. Elle a été introduite en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994 [13].

IV. Principe

Le premier algorithme de colonies de fourmis proposé est appelé le Ant system (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 :

1. elle ne peut visiter qu'une fois chaque ville.
2. plus une ville est loin, moins elle a de chance d'être choisie (c'est la « visibilité »).
3. plus l'intensité de la piste de phéromone disposée sur l'arête entre deux villes est grande, plus le trajet aura de chance d'être choisi.
4. une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones si le trajet est court.
5. les pistes de phéromones s'évaporent à chaque itération [12].



L'algorithme «système formique» optimisant le problème du voyageur de commerce :

1) une fourmi choisit un trajet, et trace une piste de phéromone.
2) L'ensemble des fourmis parcourt un certain nombre de trajets, chaque fourmi déposant une quantité de phéromone proportionnelle à la qualité du parcours.
3) Chaque arête du meilleur chemin est plus renforcée que les autres.
4) L'évaporation fait disparaître les mauvaises solutions.

Figure II.1. *Principe de l'algorithme de colonies de fourmis*.

V. Organisation de la méta- heuristique

En plus des règles régissant le comportement des fourmis, un autre processus majeur a cours : l'évaporation des pistes de phéromone. En effet, à chaque itération, la valeur des pistes de phéromone est diminuée. Le but de cette diminution est d'éviter une convergence trop rapide et le piégeage de l'algorithme dans des minimums locaux, par une forme d'oubli favorisant l'exploration de nouvelles régions [14].

Selon les auteurs du formalisme ACO, il est possible d'implémenter d'autres processus nécessitant un contrôle centralisé (et donc ne pouvant être directement pris en charge par des fourmis), sous la forme de processus annexes. Ce n'est, à notre sens, que peu souhaitable ; en effet, on perd alors la caractéristique décentralisée du système. De plus, l'implémentation de processus annexes entre difficilement dans une formalisation rigoureuse [14].

VI. Origine

L'idée originale provient de l'observation de l'exploitation des ressources alimentaires chez les fourmis. En effet, celles-ci, bien qu'ayant individuellement des capacités cognitives

limitées, sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid [12].

Des biologistes ont ainsi observé, dans une série d'expériences menées à partir de 1989, qu'une colonie de fourmis ayant le choix entre deux chemins d'inégale longueur menant à une source de nourriture avait tendance à utiliser le chemin le plus court.

Un modèle expliquant ce comportement est le suivant :

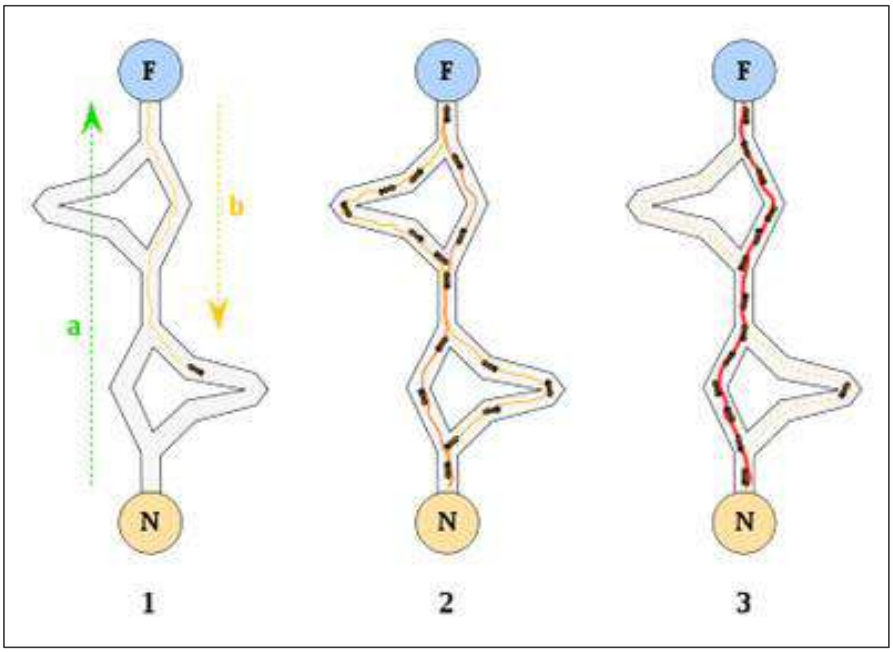
1) une fourmi (appelée « éclaireuse ») parcourt plus ou moins au hasard l'environnement autour de la colonie.
2) si celle-ci découvre une source de nourriture, elle rentre plus ou moins directement au nid, en laissant sur son chemin une piste de phéromones.
3) ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre, de façon plus ou moins directe, cette piste.
4) en revenant au nid, ces mêmes fourmis vont renforcer la piste.
5) si deux pistes sont possibles pour atteindre la même source de nourriture, celle étant la plus courte sera, dans le même temps, parcourue par plus de fourmis que la longue piste.
6) la piste courte sera donc de plus en plus renforcée, et donc de plus en plus attractive.
7) la longue piste, elle, finira par disparaître, les phéromones étant volatiles.
8) à terme, l'ensemble des fourmis a donc déterminé et « choisi » la piste la plus courte.

Les fourmis utilisent l'environnement comme support de communication : elles échangent indirectement de l'information en déposant des phéromones, le tout décrivant l'état de leur « travail ». L'information échangée a une portée locale, seule une fourmi située à l'endroit où les phéromones ont été déposées y a accès. Ce système porte le nom de « stigmergie », et se retrouve chez plusieurs animaux sociaux (il a notamment été étudié dans le cas de la construction de piliers dans les nids de termites) [12].

Le mécanisme permettant de résoudre un problème trop complexe pour être abordé par des fourmis seules est un bon exemple de système auto-organisé. Ce système repose sur des rétroactions positives (le dépôt de phéromone attire d'autres fourmis qui vont la renforcer à leur tour) et négatives (la dissipation de la piste par évaporation empêche le système de s'emballer). Théoriquement, si la quantité de phéromone restait identique au cours du temps sur toutes les branches, aucune piste ne serait choisie. Or, du fait des rétroactions, une faible variation sur une branche va être amplifiée et permettre alors le choix d'une branche. L'algorithme va permettre

de passer d'un état instable où aucune branche n'est plus marquée qu'une autre, vers un état stable où l'itinéraire est formé des « meilleures » branches [12].

Exemple II.1 :



1) la première fourmi trouve la source de nourriture (F), via un chemin quelconque (a), puis revient au nid (N) en laissant derrière elle une piste de phéromone (b).
2) Les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.
3) Les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones.

Figure II.2. *Exemple de l'algorithme de colonies de fourmis*.

VII. Stigmergie

La stigmergie est un des concepts à la base de la création des méta- heuristiques de colonies de fourmis. Elle est précisément définie comme une «forme de communication passant par le biais de modifications de l'environnement», mais on peut rencontrer le terme « interactions sociales indirectes » pour décrire le même phénomène. La spécificité de la stigmergie est que les individus échangent des informations par le biais du travail en cours, de l'état d'avancement de la tache globale à accomplir [14].

VIII. Les fourmis artificielles

La fourmi artificielle se présente sous la forme d'un ensemble de procédures qui définissent son comportement. Celui-ci est très semblable à celui de la fourmi naturelle quand elle recherche de la nourriture. Dans ce cas, une fourmi n'a qu'un rôle assez simple qui consiste à se déplacer du nid jusqu' à la source de nourriture et à y revenir. Le code qui définit leur comportement permet aux fourmis artificielles de se déplacer dans l'espace combinatoire formé par les différents éléments qui peuvent être utilisés pour le problème à résoudre. Pour utiliser un vocabulaire informatique, nous dirons qu'elle construit une solution. La mémorisation de ces déplacements donne la forme d'une solution où chaque étape est désignée par I' indice de l'élément et où I' ordre de parcours désigne la position des éléments dans la solution.

Dans le cas des fourmis artificielles, il n'y a pas obligatoirement de phase de recrutement [14].

1. Qui est capable d'agir dans un environnement.
2. Qui peut communiquer directement avec d'autres agents.
3. Qui, est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser).
4. Qui possède des ressources propres.
5. Qui, est capable de percevoir (mais de manière limitée) son environnement.
6. Ne dispose que d'une représentation partielle et éventuellement nulle de cet environnement.
7. Qui, possède des compétences et offre des services.
8. Qui, peut éventuellement se reproduire.
9. Dont le comportement tend à, satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

En se basant sur ces définitions, il est possible de décrire le comportement des fourmis virtuelles en tant qu'agents [14]:

1. Les entités informatiques que nous venons de définir sont dites virtuelles car elles n'ont pas d'existence matérielle au contraire des entités physiques qui interagissent dans le monde concret (Des exemples de ces agents physiques sont un robot, un avion, une voiture).
2. Les agents sont capables d'agir : dans le cas des fourmis virtuelles, elles modifient les valeurs de phéromone associées aux différents éléments. Par cette action elles changent leur environnement, ce qui influera sur le choix des autres fourmis à l'itération suivante.
3. Les agents sont capables de communiquer : les fourmis utilisent comme on I' a vu précédemment la phéromone comme médium de communication indirecte.
4. Les agents sont doués d'autonomie : chaque fourmi â pour but de construire une solution pour un problème donné, se contentant pour cela d'appliquer les règles de sélection qui définissent son comportement, la fourmi utilise la phéromone et parfois des valeurs heuristiques.
5. Les agents n'ont qu'une représentation partielle de leur environnement : lors de la construction d'une solution, la fourmi ne connaît à chaque étape que les éléments qu'elle a déjà choisis et les valeurs de phéromone correspondant aux éléments qui pourront l'être.

Comme nous avons pu le voir, les fourmis peuvent résoudre collectivement des problèmes complexes. Un des exemples marquant est celui de la découverte du chemin le plus court dans l'expérience du choix entre les deux chemins. Chacune des fourmis, isolément, ne peut trouver la meilleure solution. C'est grâce au mécanisme d'auto- organisation qui émerge de la communication indirecte (stigmergie) que le plus court chemin peut être découvert. Le modèle de la stigmergie peut être facilement étendu aux agents artificiels en associant aux éléments d'un problème des variables d'état spécifique, qui serviront de support à la communication indirecte entre fourmis.

Pour simuler ce mécanisme, il faut intégrer un processus de renforcement positif. Dans le cas des fourmis, plus le chemin est court, plus les fourmis le parcourront vite et plus la quantité de phéromone associée sera élevée. Afin de pouvoir comparer les solutions, il est nécessaire d'introduire une mesure qui donne l'adéquation de la solution au problème : c'est la fonction qualité (en anglais fitness). La valeur de cette qualité est utilisée dans le calcul de la mise à jour des valeurs associées aux éléments du problème lors de la phase de renforcement. Pour la recherche chemin le plus court, on peut prendre tout simplement la distance comme mesure de qualité : plus le chemin est court, plus il sera renforcé [14].

Une des propriétés de la phéromone est son caractère volatil. Dans le modèle virtuel, un mécanisme similaire sera utilisé afin d'éviter une convergence prématurée (stagnation) due à la découverte d'un optimum local.

Entre ces deux modèles, nous passons d'un univers continu à un univers discret. Ainsi, les fourmis virtuelles sautent d'un élément à un autre, tandis que les fourmis naturelles progressent de façon continue sur le chemin. Cette discrétisation impose que l'évaporation ou la modification des valeurs de phéromone ne puisse se faire en continu. Cet ajustement de valeurs n'est souvent réalisable qu'après la construction de la solution complète [14].

IX. Description générale « Ant System »

Dans l'algorithme Ant System (AS), chaque fourmi est initialement placée sur une ville choisie aléatoirement, chacune possède une mémoire qui stocke la solution partielle qu'elle a construite auparavant. Initialement, la mémoire contient la ville de départ. Commençant à partir de cette ville, une fourmi se déplace itérativement d'une ville à une autre. Quand elle est à une ville i, une fourmi k choisit d'aller à une ville non encore visitée j avec une probabilité donnée par [15]:

$$\begin{cases} P_{ij}^{K}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{y \epsilon N_i^k} \left([\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}\right)} & Si \quad j \epsilon N_i^k \\ 0 \quad sinon \end{cases} \qquad \text{(II. 1)}$$

$\boldsymbol{\tau_{ij}(t)}$ Est l'intensité de la trace de phéromone dans l'arête (i, j) à l'instant t.

$\boldsymbol{\eta_{ij} = \frac{1}{d_{ij}}}$ Est une information heuristique à priori valable, où $\boldsymbol{d_{ij}}$ est la distance entre la ville i et la ville j ; l'idée étant d'attirer les fourmis vers les villes les plus proches.

$\boldsymbol{\alpha, \beta}$ Sont deux paramètres qui déterminent l'influence relative de la trace de phéromone et de l'information heuristique.

$\boldsymbol{N_i^k}$ Est le voisinage faisable de la fourmi **k** c'est à dire l'ensemble des villes non encore.

La construction de solution se termine après que chaque fourmi ait complété un tour. Ensuite, les traces de phéromone sont mises à jour. Dans AS, la mise à jour se fait, d'abord, en réduisant les traces de phéromone avec un facteur constant ρ (c'est l'évaporation de phéromone)

et, ensuite, en permettant à chaque fourmi de déposer de la phéromone sur les arêtes qui appartiennent à son tour. Ainsi la formule de mise à jour de phéromone est comme suit [15]:

$$\boldsymbol{\tau_{ij}(t+1)} = (1-p) \times \tau_{ij}(t) + \sum_{k=1}^{nbAnts} \Delta\tau_{ij}^{k} \qquad \text{(II. 2)}$$

Avec $0 \leq p \leq 1$ et $nbAnts$ le nombre de fourmis.

Le paramètre ρ est ainsi utilisé pour éviter l'accumulation illimitée de phéromone et permet à l'algorithme d'oublier les mauvaises décisions précédemment prises. Sur les arêtes qui n'ont pas été choisis par les fourmis la force associée va décroître rapidement avec le nombre d'itérations [15].

$\Delta\,\tau_{ij}^{k}$ Est la quantité de phéromone que la fourmi k dépose sur l'arête (i, j).

$$\Delta\,\tau_{ij}^{k} = \begin{cases} \dfrac{\mathcal{Q}}{\mathcal{L}^{k}} & Si\ \ (i,j)\ \epsilon\ Tabou^{k} \\ 0 & Sinon \end{cases} \qquad \text{(II. 3)}$$

Où $\mathcal{L}^{k}$ est la longueur du tour généré par la fourmi $\mathcal{K}$, $\mathcal{Q}$ une constante de l'algorithme et $Tabou^{k}$ est la liste des villes déjà visitées.

Avec cette formule, les arêtes du tour le plus court recevront la plus grande quantité de phéromone. En général, les arêtes qui sont utilisées par plusieurs fourmis et qui appartiennent aux tours les plus courts recevront plus de phéromone et en conséquence seront plus favorisés dans les itérations futures de l'algorithme [15].

X. L'Algorithme

Étant donné les définitions de la section précédente, l'algorithme dit ant-cycle est simplement déclaré comme suit. Au temps zéro une phase d'initialisation les fourmis sont positionnés sur différentes villes et valeurs initiales pour l'intensité de la piste sont fixés sur les bords [15].

Le premier élément de la liste tabou de chaque fourmi est réglé pour être égale à sa ville de départ. Tous les fourmis Se déplacent ensuite de la ville i à j choisissant la ville pour aller avec une probabilité qui est une fonction (avec des paramètres d'un α et β, voir l'équation (II. 2)) de deux mesures de désirabilité. Le premier, le $\boldsymbol{\tau_{ij}}$ $\tau_{ij}(t)$ de sentiers (t), donne des informations sur la façon dont beaucoup de fourmis dans le passé ont choisi ce même bord (i, j) ; la seconde, la visibilité, dit que la plus proche ville est la plus souhaitable. Il est évident

que la mise α = 0, $\boldsymbol{\eta}_{ij}$ les traces de niveau n'est plus considéré, et un algorithme glouton stochastique avec plusieurs points de départ est obtenu.

Après n itérations toutes les fourmis ont terminé une tournée, et leurs listes de tabou seront pleines ; à ce point pour chaque fourmi k la valeur de $\mathcal{L}_k$ est calculée et les valeurs $\Delta\,\tau_{ij}^{k}$ sont mise à jour selon l'équation (II. 3). Aussi, le plus court chemin trouvé par les fourmis (c.-à-d. min $\mathcal{L}_k, k = 1, \dots, m$) est enregistrée et toutes les listes taboues sont vidées.

Ce processus est réitéré jusqu'à ce que le compteur de visite atteint le maximum (défini par l'utilisateur) nombre de cycles NC_{MAX}, ou toutes les fourmis font la même tournée. Nous appelons ce dernier cas le comportement de stagnation car il désigne une situation dans laquelle l'algorithme s'arrête la recherche des solutions alternatives. Nous étudions cette situation à la section IV [15].

Formellement l'algorithme ant-cycle est :

1. Initialisation
   Set t : = 0 {t est le compteur de temps}
   Set NC : = 0 {NC est le compteur de cycles}
   Pour chaque bord (i, j), définir une valeur initiale $\tau_{ij}(t) = c$ pour l'intensité de la piste et $\Delta\tau_{ij} = 0$.
   Placez les m fourmis sur les n nœuds.
2. Set s : = 1 {s est l'indice de liste tabou}
   For k := 1 to m do
   Placez la ville de départ de la fourmi k-ième à Tabou k (s)
3. Répétez l'opération jusqu'à liste tabou est plein {cette étape sera répétée (n-1) fois} Set S = S + 1
   For k : = 1 to m do
   Choisissez la ville j pour déplacer, avec une probabilité $P_{\ddot{g}}^{k}\ (t)$ donnée par l'équation (II.1) {au temps t la fourmi k-ième est en ville i = tabou k (s-1)}
   Déplacez la fourmi k-ième à la ville j
   Insérez ville j à Tabou k (s)
4. For k : = 1 to m do
   Déplacez la fourmi k-ième de Tabou k (n) à destination de Tabou k (1)
   Calculer la longueur$\mathcal{L}_k$ de la tournée décrit par la fourmi k-ième
   Mettre à jour la plus courte tournée trouvé
   Pour chaque bord (i, j)

For k=1 to m do

$$\Delta \tau_{ij}^{k} = \begin{cases} \frac{Q}{\mathcal{L}^{k}} & Si \ \ (i,j) \ \epsilon \ Tabou^{k} \\ 0 & Sinon \end{cases}$$

5. Pour chaque bord (i, j) calculer $\boldsymbol{\tau_{ij}(t+n)}$ selon l'équation

$$\boldsymbol{\tau_{ij}(t+n) = P\tau_{ij}(t) + \Delta\tau_{ij}}$$

Set t : = t + n

Set NC : NC = + 1

Pour chaque bord (i, j) définir $\Delta\tau_{ij} := 0$

6. If (NC <NCMAX) and (not stagnation behavior)

Then

Videz toutes les listes de tabou

Aller à l'étape 2

Else

Imprimer le chemin le plus court

Stop [15].

XI. Conclusion

Nous avons présenté dans ce chapitre l'algorithme de colonies de fourmis qui a été appliquées avec succès à de nombreux problème combinatoires et commencent à être adaptées à des problèmes continus. Dans le chapitre suivant on va appliquer cet algorithme avec quelque modification pour résoudre le problème de plus court chemin point vue de temps, utilisant logiciel Matlab.

Chapitre III : Tsests et Résultats

I. Introduction

L'algorithme de colonie fourmis a fait l'objet d'une étude théorique. Dans le chapitre II, nous allons utiliser cet algorithme avec quelques modifications en utilisant Matlab, et l'appliquer sur une partie du réseau routier de la ville de Médéa, pour trouver le plus court chemine entre deux points.

II. Présentation du problème

Le temps déplacement est une importante dans notre projet où le système développé consiste à acheminer les véhicules dans un réseau, routier autrement, le système conçu permet de déterminer le plus court chemin entre un point A et un point B en point de vue temps.

III. Algorithme de colonie fourmis modifié

Avant que nous commençons l'application de l'algorithme de colonie de fourmis, nous extrayons toutes les chemins qui relient les points (point de départ et point d'arrivé) et en prenant quelques idées de l'application de l'algorithme de colonie de fourmis au Problème du voyageur de commerce.

III.1. Problème du voyageur de commerce

Le problème du voyageur de commerce, étudié depuis le 19e siècle, est l'un des plus connus dans le domaine de la recherche opérationnelle. Il s'agit de trouver le meilleur parcours possible et découvrir les différentes méthodes informatiques proposées pour résoudre ce problème.

C'est déjà sous forme de jeu que *William Rowan Hamilton*[3] a posé pour la première fois ce problème, dès 1859. Sous sa forme la plus classique, son énoncé est le suivant : « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur ».

[3] Sir **William Rowan Hamilton** (4 août 1805 - 2 septembre 1865) est un mathématicien, physicien et astronome irlandais (né et mort à Dublin). Il est connu pour sa découverte des quaternions, mais il contribua aussi au développement de l'optique, de la dynamique et de l'algèbre. Ses recherches se révélèrent importantes pour le développement de la mécanique quantique.

Le problème du voyageur de commerce peut être modélisé à l'aide d'un graphe constitué d'un ensemble de sommets et d'un ensemble d'arêtes. Chaque sommet représente une ville, une arête symbolise le passage d'une ville à une autre.

III.2. Les chemins existants

A partir de problème précédent, nous allons prendre neuf villes et les remplacer par des points, et extraire les chemins, voir l'exemple III.1.

Exemple III.1 :

Dans cet exemple, nous avons 9 points et les distances entre deux points sont différentes (Figure III.1).

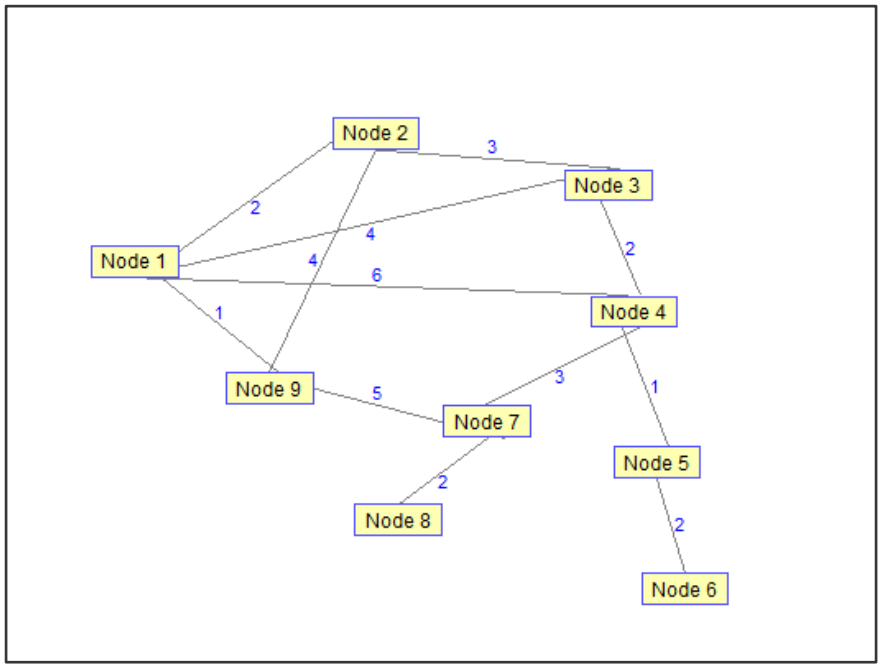


Figure III.1. *représentation des points et distances entre eux.*

Les étapes de travail

- Mettre un certain nombre des fourmis dans les différents points et on les laisse se déplacer aléatoirement. Chaque fourmi sauvegarde les points visités dans une liste de tabou.
- Les fourmis choisissent à chaque fois un point adjacent qui n'est pas encore visité, et cela avec une vérification dans le tabou de cette fourmi.
- Si la fourmi arrive au point qu'on a fixé précédemment elle s'arrête.

- Après que toutes les fourmis terminent leur cycle on cherche dans le tabou de chaque fourmi le trajet voulu.
- Ainsi on remet les fourmis au point de départ où les laisse refaire le même travail plusieurs fois pour obtenir tous les trajets possible relie entre notre points de départ et l'arrivée.
- Le nombre de trajets augmente avec le nombre d'itérations.

Blocage

On a suivi les étapes de l'algorithme colonie fourmis mais après l'exécution de programme on s'est heurté à un problème.

1. Problème rencontré

Comme on l'a déjà expliqué, si la fourmi trouve un point adjacent dans la liste tabou elle choisit un autre point, mais dans quelques cas elle trouve tous les points adjacents dans le tabou sans arriver au point d'arrêt. L'explication est donnée à travers l'exemple III.2.

Exemple III.2 :

En appliquent l'algorithme de colonie de fourmis.

- Le point de départ est 9 et le point d'arrivée 4.
- Une fourmi démarre de point 9 et suit son trajet jusqu'au point 2 mais elle est bloquée car les points adjacents à 2 existent déjà dans la liste tabou

Tabou= [9 1 3 2]

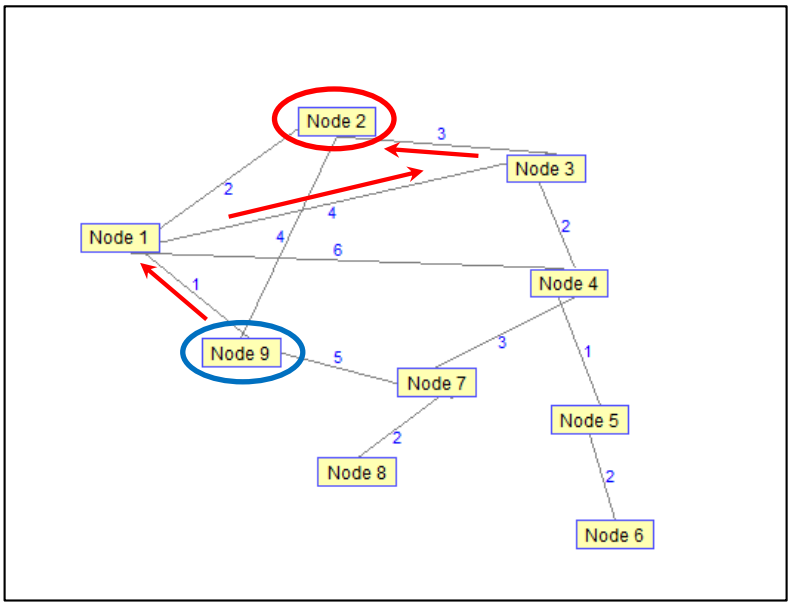


Figure III.2 . *chemin parcouru dans l'exemple III.2.*

C'est un problème et on a proposé une modification à l'algorithme de colonie fourmis.

2. Solution

Après plusieurs essais on a trouvé une solution plus appropriée est comme suit :

- A chaque fois que la fourmi se trouve bloqué, car tous les points adjacents sont dans la liste tabou elle ajoute ce point au tabou et revient au point précédent. Au niveau de ce dernier elle vérifie avec la même méthode et si elle ne trouve pas les points adjacent dans le tabou, elle termine son trajet sinon elle revient au précédent de ce point et ainsi de suite, jusqu'on trouve un point qui n'existe pas dans la liste tabou.

3. Déblocage

Dans notre exemple précèdent la fourmi se trouve bloquée au niveau du point 2, la solution est de revenir au point 3 et rajoute ce point à liste tabou, à partir du point 3, elle peut aller au point 4 car il n'existe pas dans la liste tabou. (Figure III.3).

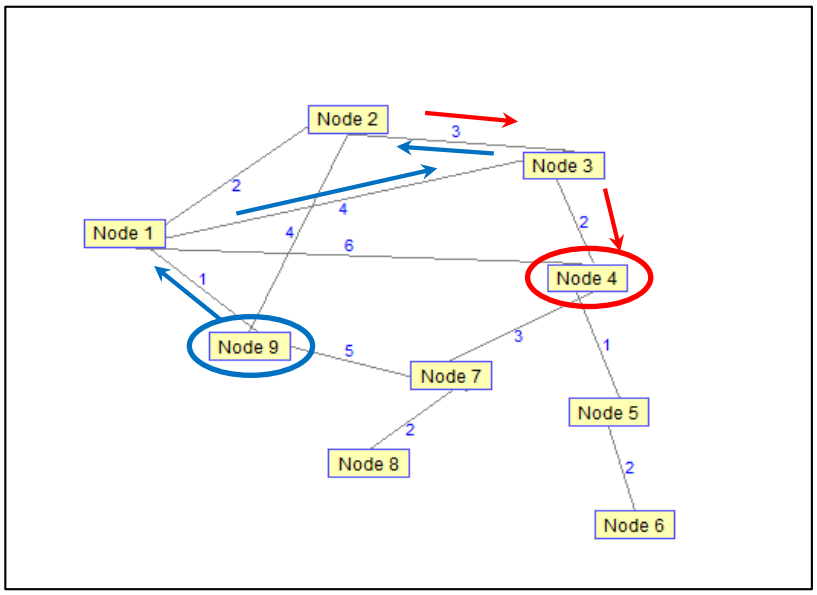


Figure III.3. *nouveau chemin.*

Le résultat de simulation est indiqué par la figure III.4.



Figure III.4. *simulation MATLAB de la solution proposé à l'exemple III.2.*

On a résolu le problème précédent mais on s'est lente encore à un autre petit problème, c'est que les fourmis passent par quelques points plus d'une seule fois, et c'est des répétitions qu'on doit supprimer.

On explique comment faire la suppression de ces répétitions avec l'exemple de la liste tabou :

Tabou= [9 1 3 2 3 4]

Dans la liste on remarque que la fourmi passe du point 3 au point 2 et revient au point 3, alors il faut supprimer le point 3 répété et le point 2 pour que le trajet devient comme l'indique la figure III.5 et il devient.

Tabou= [9 1 3 4]

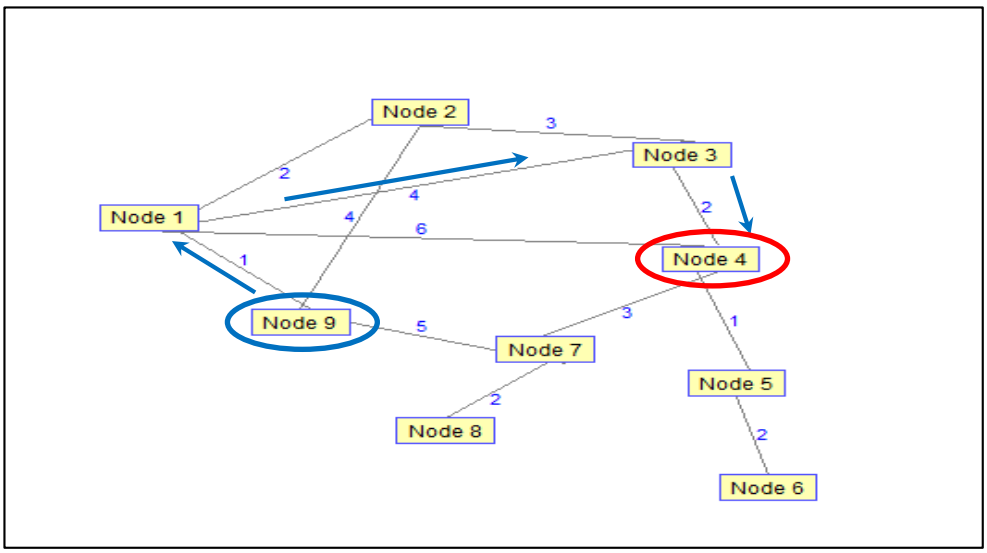


Figure III.5. *chemin finale.*

4. La liste tabou finale :

Après que toutes les fourmis terminent ces cycles, Nous avons extraire les trajets les plus important pour nous, et on les mettre dans une matrice appelé **tabou finale**.

Et pour vérifier le fonctionnement de notre programme on va donner 2 exemples :

Exemple III.3 :

On définit les entrées par le tableau III.1 :

Tableau III.1. *les valeurs de données pour la simulation de l'exemple III.3.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles
Valeur	**1**	**8**	**6**	**2**

Le résultat de simulation est indiqué dans la figure III.6 :

On remarque l'existence de 4 chemins possible.

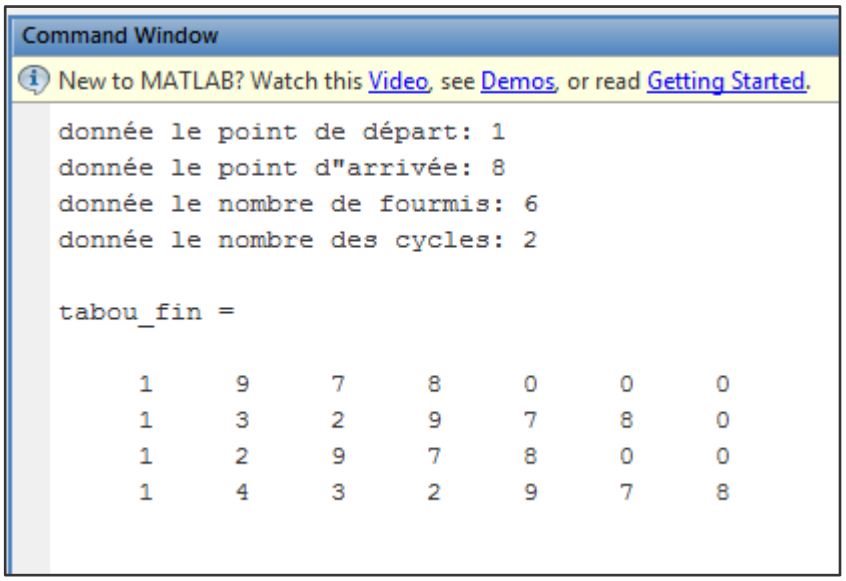


Figure III.6. *simulation MATLAB de l'exemple III.3.*

Exemple III.4 :

Dans cet exemple on va augmenter le nombre des fourmis aussi que le nombre des cycles.

On définit les entrées par le tableau III.2 :

Tableau III.2. *les valeurs de données pour la simulation d'exemple III.4.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles
Valeur	**1**	**8**	**20**	**8**

Le résultat de simulation est indiqué par la figure III.7.

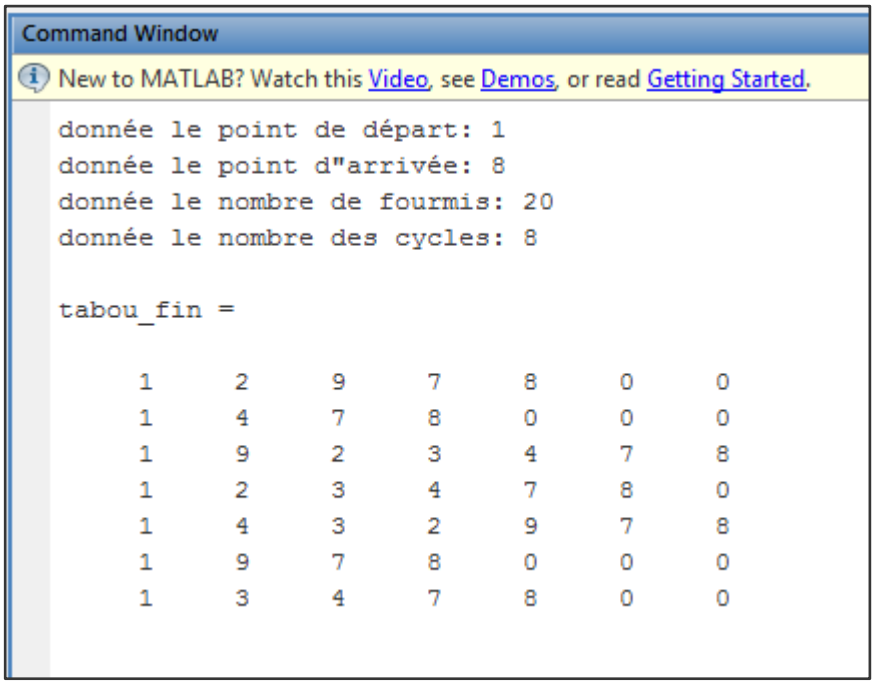


Figure III.7. *simulation MATLAB de l'exemple III.4.*

On remarque l'existence de 7 chemins possibles.

On remarque qu'à chaque fois le nombre des fourmis et le nombre de cycles augmentent, le nombre des chemins possibles augmente aussi, et par conséquent, nous avons extrait tous les chemins possibles.

III.3. Le plus court chemin en termes de distance

Dans cette partie nous appliquons l'algorithme de colonie de fourmis.

Les étapes à suivre

- Entre chaque deux point on ajoute une quantité aléatoire de phéromone.
- Mettre toutes les fourmis au point de départ.

- Laisser les fourmis se déplace en appliquant l'algorithme de colonie de fourmis.

Blocage

A chaque fois qu'on applique l'algorithme de colonie de fourmis ne trouve pas le meilleur trajet selon nos analyses, puis que dans notre problème il ne s'agit pas exactement du problème de voyageur de commerce, il s'agit de trouver le meilleur trajet entre deux points.

Solution proposée

Pour que le problème soit au niveau de phéromone, on a proposé de faire la mise à jour du phéromone selon les étapes suivantes :

- Après l'obtention de matrice **tabou finale**, on remet toutes les fourmis au point de départ.
- Les fourmis se divisent dans les trajets selon les quantités de phéromones existantes dans tous les trajets, ces quantités sont initialement posées aléatoirement et le nombre de fourmis le plus grand passe aux trajets qui contiennent des quantités les plus grandes.
- Chaque fourmi ajoute une quantité de phéromone quand elle passe par certain trajet, cette quantité est différente et dépend de la distance parcouru par cette fourmi, si la distance est grande la quantité ajoutée est petite et vice versa.
- A chaque cycle, le phéromone s'évapore dans tous les chemins avec un taux d'évaporation ρ .
- Dans le cycle suivant les fourmis se divisent et se déplacent en fonction de la quantité de phéromone dans chaque trajet.
- Après plusieurs cycles la quantité de phéromone augmente plus dans le plus court trajet, remarquant que la plus grande partie des fourmis passe par ce trajet, mais il y a des cas où une partie du trajet est commune à plusieurs trajets possible, alors la quantité de phéromone augmente à ce niveau, ce qui fait que les fourmis favorisent ce trajet au lieu de passer par le trajet le plus court. On explique ce problème par un exemple.

Exemple III.5 :

On définit les entrées par le tableau III.3 :

Tableau III.3. *les valeurs de données pour la simulation d'exemple III.5.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles
Valeur	**1**	**4**	**50**	**10**

Le résultat de simulation est indiqué par la figure III.8 suivante :

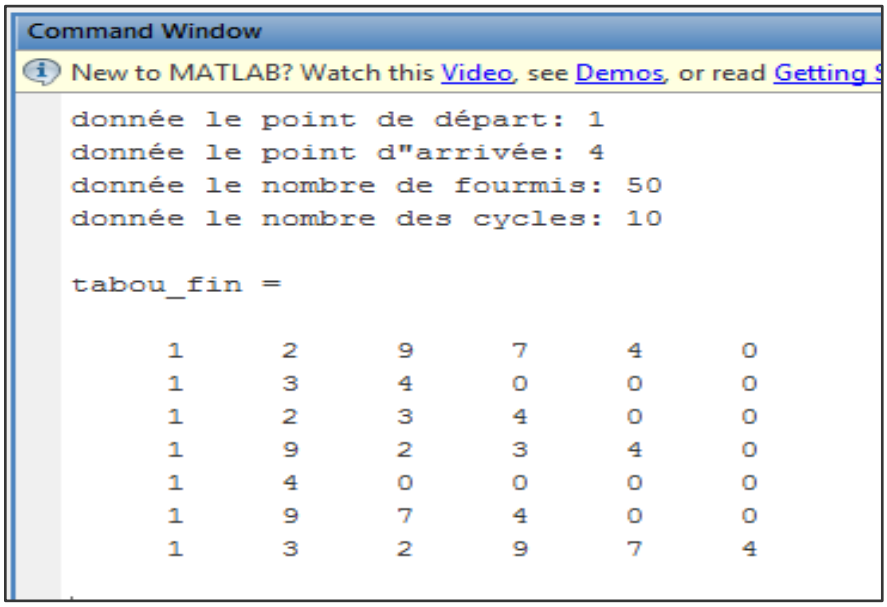


Figure III.8. *simulation MATLAB de l'exemple III.5.*

1. **Problème**

- A partir de l'exécution on trouve que les fourmis passent par 7 chemins pour un cycle.
- Prendre par exemple les trajets (1-2-9-7-4),(1-2-3-4),(1-4), nous remarquons que les fourmis passe par l'arc (1-2) deux fois, et passe par l'arc (1-4) une seule fois, alors la quantité de phéromone après un cycle est plus grande dans l'arc (1-2) par apport aux arc (1-4) , donc le nombres de fourmis dans le cycle suivant est plus grand dans l'arc (1-2) par apport à l'arc (1-4) malgré que le plus court trajet est le (1-4) et ce n'est pas le (1- 2-9-7-4), (figure III.9).

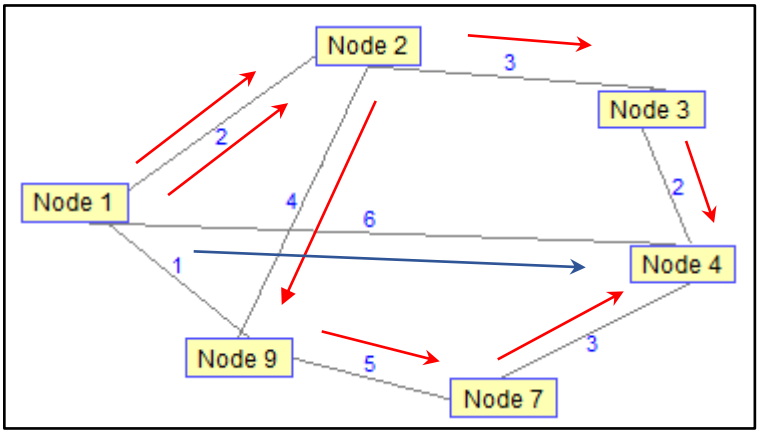


Figure III.9. *chemins parcourus possibles.*

2. Solution :

Pour résoudre ce problème, on propose la décomposition des chemins, comme l'indique la figure III.10.

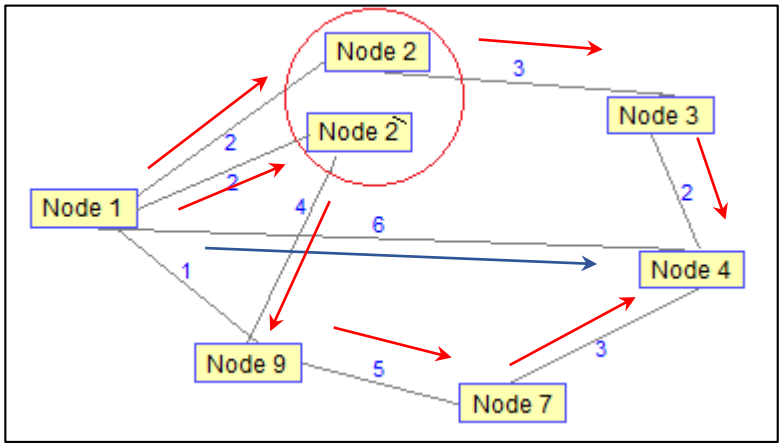


Figure III.10. *les chemins après décomposition.*

3. Déblocage

L'idée est de partager les chemins parcourue plusieurs fois en plusieurs chemins en gardant la même quantité de phéromones selon l'exemple on obtient deux trajets différents (1- 2'- 9- 7- 4) et (1- 2- 3- 4).

Cet exemple montre l'obtention de plus court chemin en termes de distance entre deux points.

Exemple III.6 :

Dans cet exemple on va utiliser le même graphe précédant, et déterminer le plus court chemin.

On définit les entrées par le tableau III.4 :

Tableau III.4. *les valeurs de données pour la simulation d'exemple III.6.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles
Valeur	**1**	**8**	**50**	**50**

Le résultat de simulation est indiqué par la figure III.11 suivante :

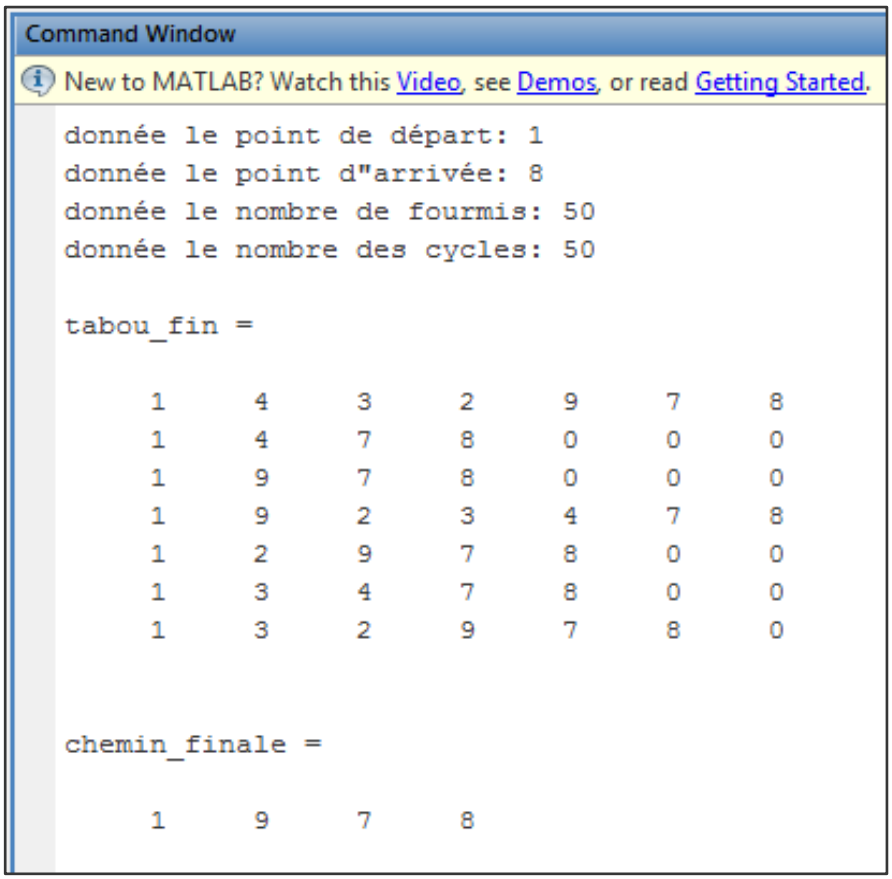


Figure III.11. *simulation MATLAB de l'exemple III.6.*

Après l'exécution de l'algorithme on trouve le chemin le plus court entre tous les chemins Possibles.

IV. Appliqué l'algorithme au réseau routier de la ville de Médéa

Dans cette partie on va procéder à l'application de l'algorithme de colonie de fourmis modifie à une partie du réseau routier de la ville de Médéa. Pour trouver les plus courts chemins entre deux points en termes de temps.

IV.1. Présentation Réseau routier de la ville de Médéa

Ce réseau routier est tiré de **Google map** qui contient 76 points (nœuds) où chaque nœud est identifié par un nombre (1,2,...76) et les sens de circulation sont indiqués dans la figureIII.12. On a calculé les distances selon l'échelle indiquée par **Google map** et le temps réelle dans la matrice est appelé **sans_circulation**.

Figure III.12. *Représentation schématique d'une partie du réseau routier de la ville de Médéa.*

IV.2. Programmation dynamique

Notre programme est composé essentiellement de deux parties principales la première est « les chemins possibles » cette partie va permettre de donner tous les chemins possibles entre les deux points, et la deuxième partie « le chemin le plus court »c'est la base de notre travail, cette partie permet de donner le chemin le plus court avec point de vue temps.

Les entrées de notre programme sont :

Point de départ : Le point de départ du conducteur.

Point d'arrivée : Le point d'arrivée du conducteur.

Horaire : est le moment de départ du conducteur (l'heure de départ).

Les Matrices de temps : Quatre matrices représentent tous les temps entre les nœuds, chaque matrice associée à la circulation de route pour une période de la journée.

- **Sans_circulation :** cette matrice indique le temps dans les horaires où il n'y pratiquement pas de trafic (exemple à partir de 23h jusqu'à 5h).
- **Temps_min :** cette matrice indique le temps dans les horaires où il y a un peu de trafic (exemple de 5h à 6h, de 13h à 15h, et de 22h à 23h).
- **Temps_moy :** cette matrice indique le temps dans les horaires où il y a un trafic moyen (exemple de 6h à 7h, de 15h à 16h, et de 18h à 22h).
- **Temps_max :** cette matrice indique le temps dans les horaires où il y a une grande circulation (exemple de 7h à 9h, de 11h à 13h, et de 16h à 18h).

Et les sorties sont :

Le chemin : est un vecteur qui contient les nœuds du chemin le plus court ou rapide point de vu temps.

Le temps de déplacement : est le temps total de déplacement entre le point de départ et le point d'arrivée, estimé en seconde.

On peut représenter l'organigramme de fonctionnement de l'algorithme Proposé, qui donne le plus court chemin point de vue distance ou temps par la figure III.13 suivante.

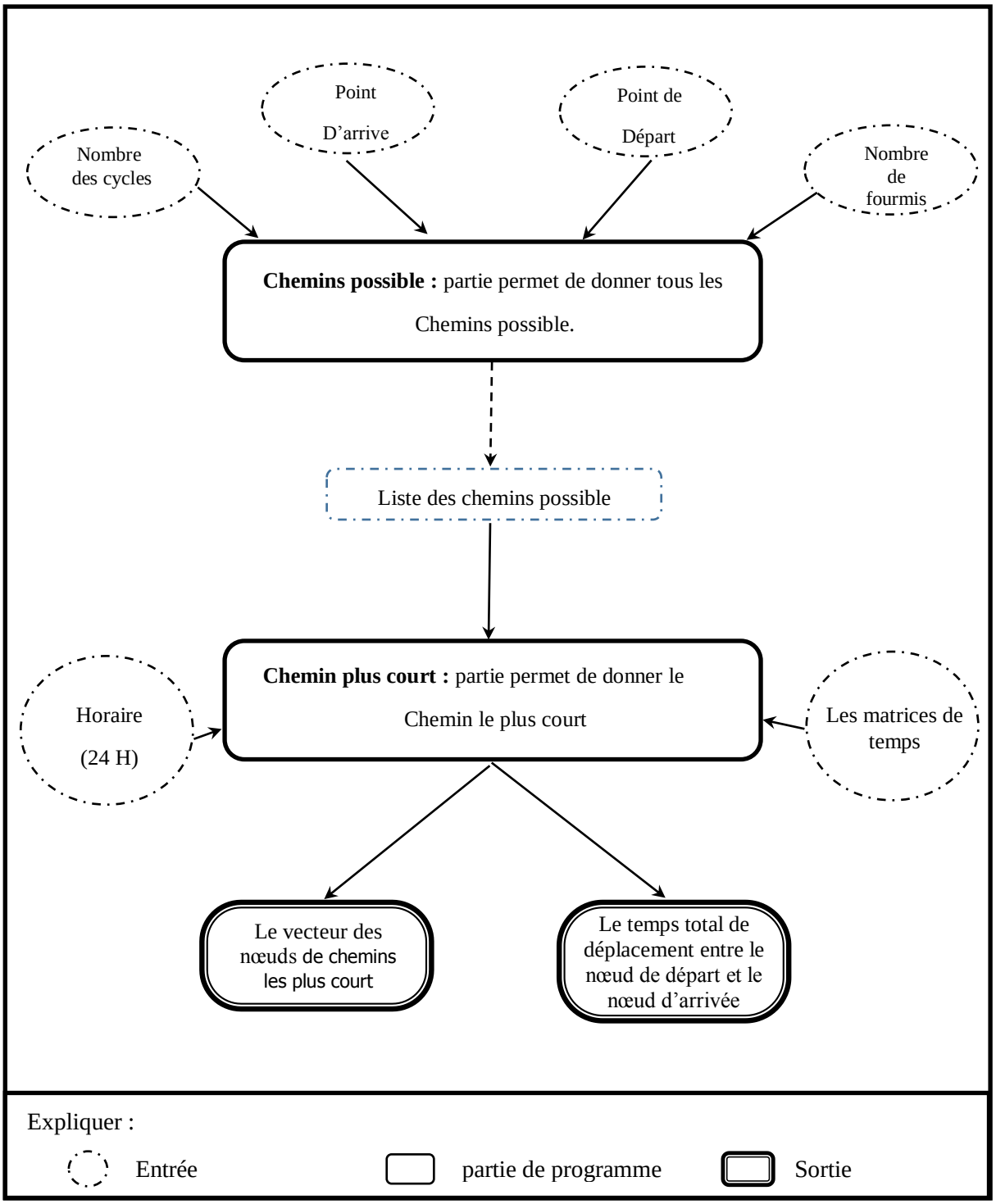


Figure III.13. *organigramme de l'algorithme proposé.*

IV.3. Résultat de simulation sur le réseau routier de la ville de Médéa

Pour vérifier le fonctionnement de notre programme on va donner deux exemples.

Exemple III.7 :

Dans cet exemple on va utiliser la matrice de temps sans_circulation.

On définit les entrées par le tableau II.5 :

Tableau III.5. *les valeurs de données pour la simulation d'exemple III.7.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles	Horaire (24 h)
Valeur	**1**	**30**	**5000**	**100**	**3**

Le résultat de simulation est indiqué par la figure III.14 suivante :

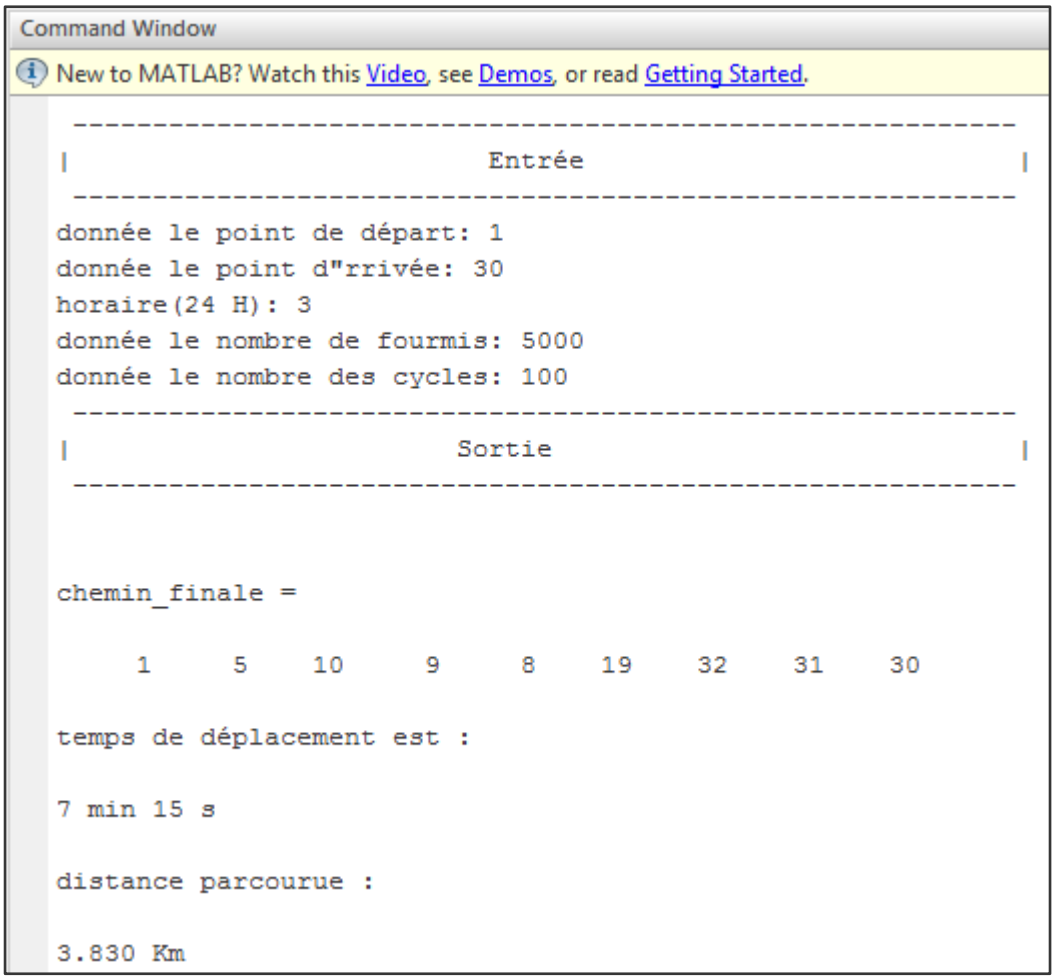


Figure III.14. *simulation MATLAB de l'exemple III.7.*

Exemple III.8 :

Dans cet exemple on explique comment le trajet le plus rapide et le temps de déplacement change, lorsqu'on change les matrices de temps.

1. On définit les entrées par le tableau III.6 :

Tableau III.6. *les valeurs de données pour la simulation d'exemple III.8.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles	Horaire (24 h)
Valeur	**9**	**22**	**5000**	**100**	**24**

Le résultat de simulation est indiqué par la figure III.15 suivante :

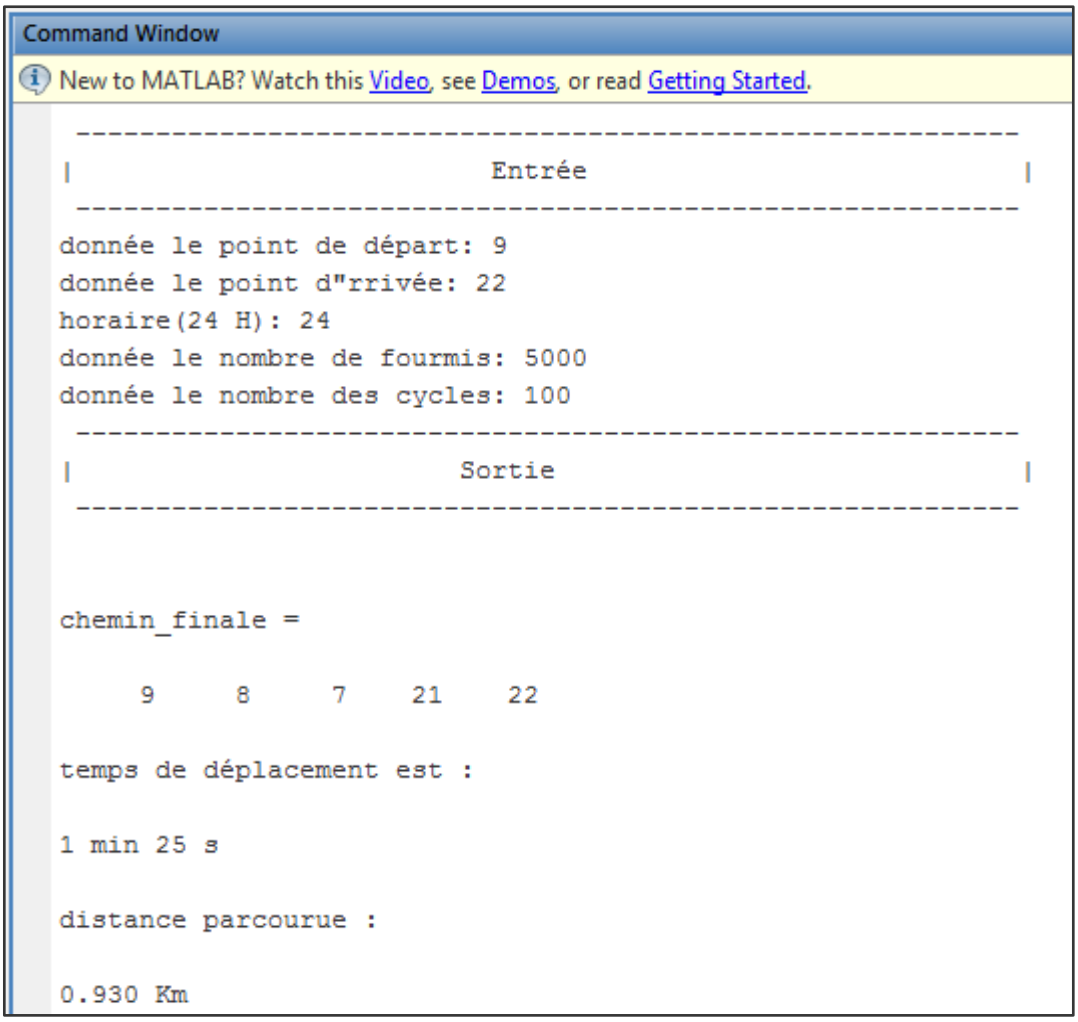


Figure III.15. *simulation MATLAB de l'exemple III.8.*

2. On définit les entrées par le tableau III.7 :

Tableau III.7. *les valeurs de données pour la simulation d'exemple III.8.*

Donnée	Point de départ	Point d'arrivée	Nombre des fourmis	Nombre des cycles	Horaire (24 h)
Valeur	**9**	**22**	**5000**	**100**	**8**

Le résultat de simulation est indiqué par la figure III.16 suivante :

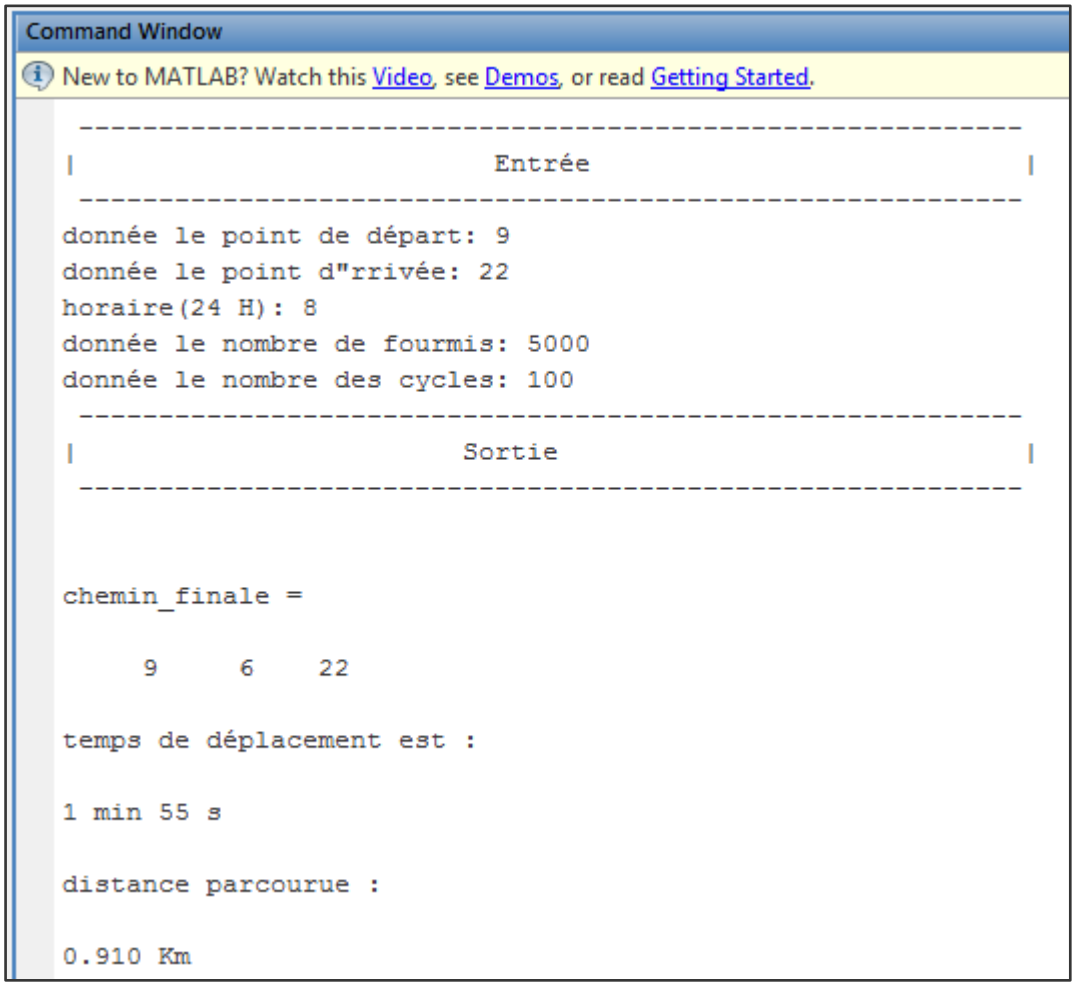


Figure III.16. *simulation MATLAB de l'exemple III.8.*

Dans cet exemple on a testé deux cas où les valeurs des entrées sont identiques sauf que dans l'horaire on a remarqué que les résultats des deux cas sont différents, dans la partie 1 on a utilisé la matrice « sans_circulation » et dans la partie 2 la matrice « temps_max », après l'exécution de programme nous donne deux chemins différents.

Figure III.17. *Résultat des exemples III.7 et III.8 sur le réseau routier*.

IV.4. Résultats obtenus

On choisit quelque trajets tel que les points de départs et d'arrivés sont aléatoires et on les sauvegarde dans un tableau III.8 pour faire des tests.

Tableau III.8. *des valeurs aléatoires des points pour faire des tests.*

	Point départ	**Point d'arrivée**
Chemin 1	15	5
Chemin 2	41	28
Chemin 3	1	30

IV.4.1. Choix du taux d'évaporation ρ

Prendre des valeurs différentes de $\rho(0 < \rho < 1)$ et choisir le meilleur après avoir effectué des tests.

Les entrées :

- Nombre de fourmis : 5000
- Horaire : 1h

Tableau III.9. *résultat des tests pour plusieurs valeurs d'évaporation.*

	Chemin 2								
p	**Nbr cycle**	**Temps d'exécution**	**Nbr chemin (Qun-pheor >10^-10)**	**Nbr cycle**	**Temps d'exécution**	**Nbr chemin (Qun-pheor >10^-10)**	**Nbr cycle**	**Temps d'exécution**	**Nbr chemin (Qun-pheor >10^-10)**
0.2	30	10.50s	2	50	10.09s	1	80	10.32s	1
0.4	30	10.32s	173	50	10.31s	4	80	10.43s	1
0.5	30	10.10s	1398	50	10.11s	34	80	10.56s	1
0.6	30	10.82s	1422	50	10.64	237	80	10.82	6
0.8	30	10.25s	1413	50	10.36s	1408	80	10.57s	1420

- **Qun-pheor -> quntité de phéromone**

A partir de ce tableau on trouve que la meilleure valeur de ρ est 0.2, parce qu'elle nous a donné un seul chemin et moins de cycles d'exécution.

IV.4.2. Choix du nombre de fourmi approprié

Les entrées :

- Horaire : 1h
- Nombre de cycles : 50

Tableau III.10. *résultat des tests pour différents nombres de fourmis.*

	Chemin 1		**Chemin 2**		**Chemin 3**	
Nbr fourmi	**temps de déplacement**	**distance parcourir**	**temps de déplacement**	**distance parcourir**	**temps de déplacement**	**distance parcourir**
2000	2m55s	1.65Km	3m15s	2.04Km	7m15s	3.83Km
4000	2m55s	1.65Km	3m15s	1.89Km	7m15s	3.83Km
5000	2m55s	1.65km	3m15s	2.04Km	7m15s	3.83Km
10000	2m55s	1.65Km	3m15s	1.89Km	7m15s	3.83Km
20000	2m55s	1.65Km	3m15s	1.89Km	7m15s	3.83Km
30000	2m55s	1.65Km	3m15s	1.89Km	7m15s	3.83Km

A parti de ce tableau on conclut que le meilleur choix est 10000 fourmis.

IV.5. Interface graphique

L'objectif de la conception d'une interface graphique est de permettre à l'utilisateur de mieux gérer son système, il est guidé dans chaque étape de la procédure, il peut tout commander simplement à travers le clavier et supervise les changements qui affectent l'instrument sur l'écran de son ordinateur.

L'interface réalisée permet de faire entrer :

- Point de départ
- Point d'arrivé
- Horaire
- Nombre de fourmis
- Nombre des cycles

Et permet de voir le résultat de l'application de l'algorithme de colonie de fourmis en donnant :

- Temps de déplacement
- Distance parcourue
- Chemin final

Exécuter l'application figure III.18 :

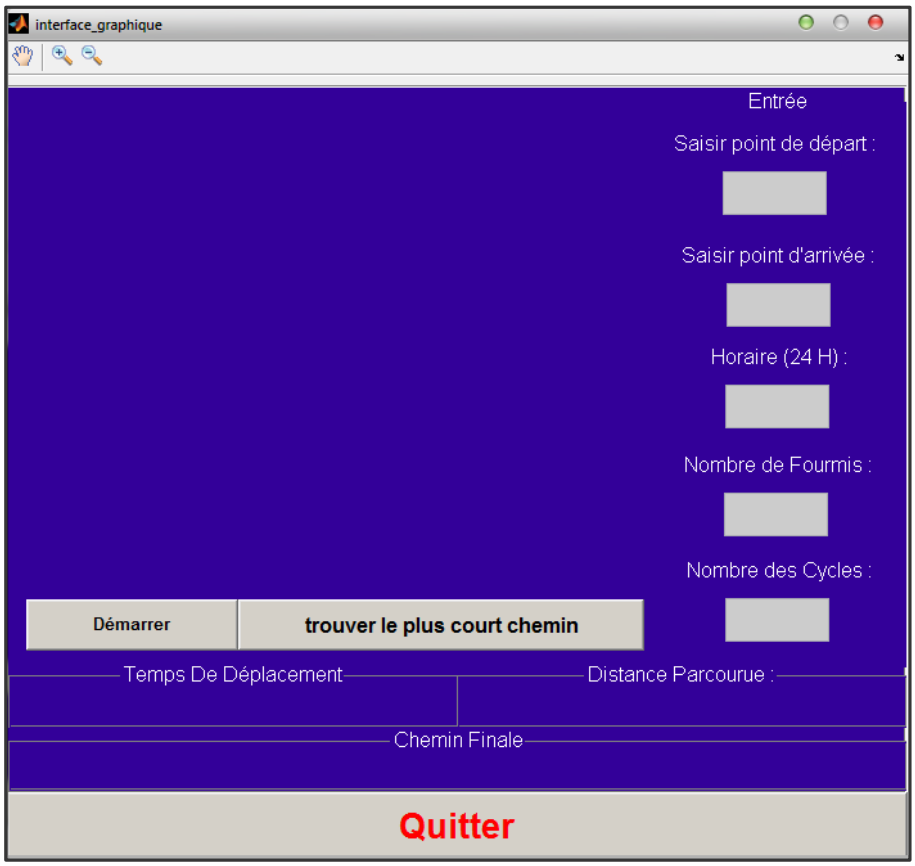


Figure III.18. *interface de l'application.*

Après l'exécution de l'application, nous cliquons sur le bouton démarrer pour afficher la carte de la ville de Médéa, figure III.19 :

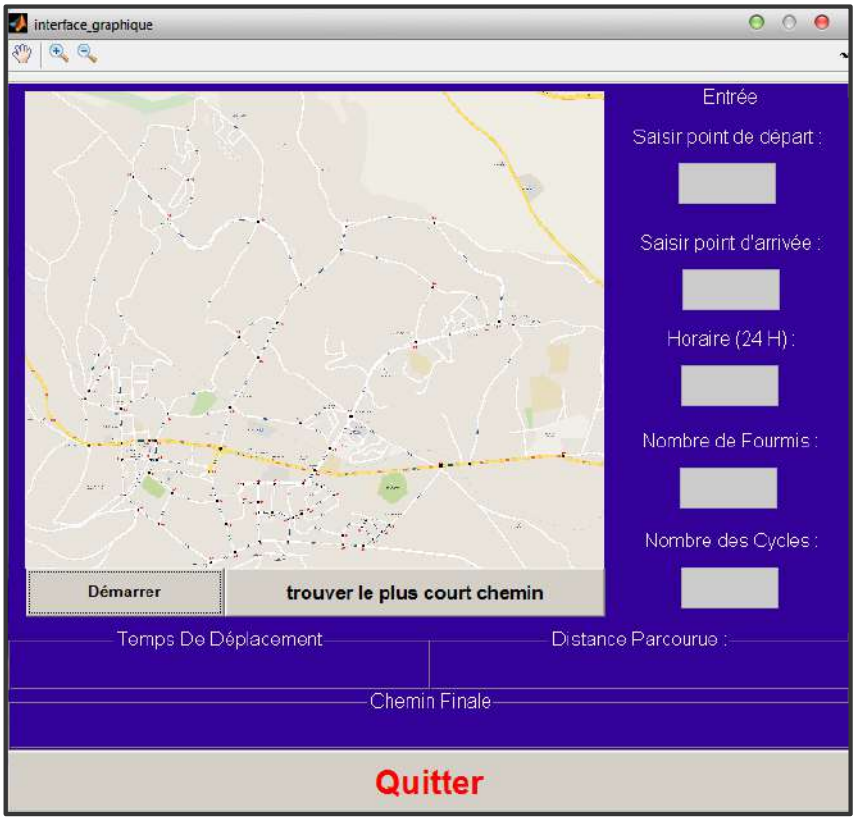


Figure III.19. *afficher la ville de Médéa sur l'interface.*

Maintenant on saisit les données, et on clique sur le bouton trouver le plus court chemin, figure III.20.

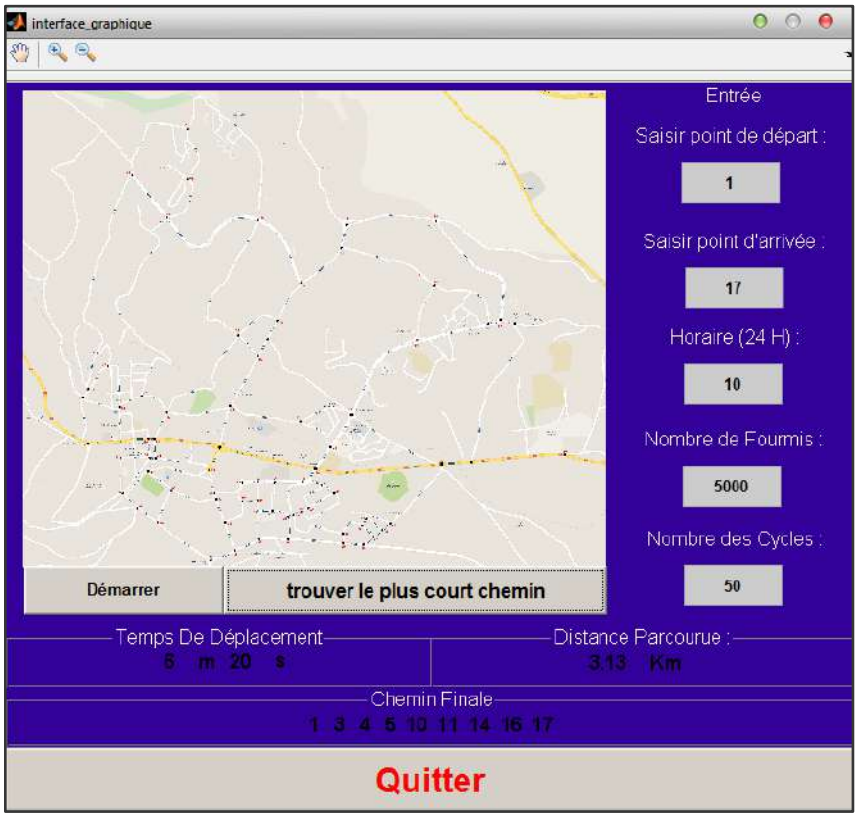


Figure III.20 : *trouver le plus court chemin sur l'interface.*

V. Conclusion

Dans ce chapitre, on a pu présenter un algorithme de colonie de fourmis modifié a fin de trouver le trajet optimal entre deux points en termes de temps. Les résultats des tests appliqués en utilisant différentes périodes de temps de la journée montrent les performances de l'algorithme proposé.

Conclusion Générale

Un algorithme intelligent a été développé et appliqué pour la résolution de notre problématique qui s'agit d'acheminement d'un conducteur au point désiré en moins de temps possible.

Pour arriver à cette fin, une étude théorique sur les réseaux routiers et leur gestion était nécessaire. Le problème traité a été transformé en un problème d'optimisation par L'algorithme de colonie de fourmis qui a été appliqué et adapté à une implémentation MATLAB avec succès après modifications car l'algorithme standard bloquait à certains niveaux.

Ce travail ne s'est pas fait sans difficultés. En effet, il a fallu prendre connaissance de l'étendue des différents domaines qui font l'objet de cette étude. Tout d'abord, le domaine de gestion de trafic routier ensuite la proposition d'un algorithme intelligent qui nous permet de déterminer le plus court chemin entre deux points en point vue temps.

Dans ce projet on a validé un modèle par le logiciel MATLAB de notre problématique.

L'application a été faite sur une partie de réseau routier de la ville de Médéa et donnée des résultats pour des différentes périodes de temps pendant la journée. Le trajet donnée par l'algorithme développe dépendait au débit des véhicules au niveau de réseau routier au le conducteur peut aller du point A au point B en un minimum de temps possible.

Donc l'objectif principal de notre projet est de trouver le plus court chemin entre deux points en terme temps dans un réseau routier et ce travail rendue dans le domaine de système de transport intelligent.

Notre travail présente beaucoup d'avantages :

- Alléger la circulation dans un réseau routier en favorisant l'emprunt des Donner plus de confort au conducteur.
- Organiser le trafic routier.
- Modularité du système conçu ou il peut être appliqué à n'importe quel autre réseau routier, il suffit de remplacer une mémoire.
- chemins au il y'a mois de congestion.

Comme perspectives, à court termes pourquoi pas prendre le réseau routier de la ville de Médéa ensuite à toutes les villes de l'Algérie et faire une application Android ça serait très intéressant.

Référence Bibliographie

Références

[1] S. Faye, «Contrôle du trafic routier urbain par un réseau fixe de capteurs sans fil,» Télécom ParisTech, Paris, 2011.

[2] Mémoire H. HADJADJ et K. HADJADJ , «Smart Auto Router,» UNIVERSITE DR. YAHIA FARES, médéa, 2011.

[3] A. COSTANZO et G. MARILL , «Optimisation par colonies de fourmis,» Université Nice Sophia Antipolis, Nice, 2006.

[4] Mémoire C. MOUILAH et K. BELKADI, «Application du « AntNet » dans un réseau routier urbain» chez *International Conference on Education and e-Learning Innovations*, ORAN, 2012.

[5] S. COHEN, «Ingéniérie du trafic 1 et 2,» École nationale des ponts et chaussées, Champs-sur-Marne, 2014.

[6] J.-P. Zanotti, «Université de Toulon,» Université de Toulon, [En ligne]. Available: http://zanotti.univ-tln.fr/algo/DIJKSTRA.html. [Accès le 4 1 2008].

[7] S. HOCEINI, «ROUTAGE INTELLIGENT DANS LES RÉSEAUX HAUT DÉBIT,» [En ligne]. Available: http://malm.tuxfamily.org/doc/qr_chap3_kshortest.htm. [Accès le 06 2003].

[8] Mémoire K. ABDERRAHMANE, «PLANIFICATION DE TRAJECTOIRES DE ROBOTS MOBILES VIA DES METHODES ENSEMBLISTES,» Laboratoire d'Ingénierie des Systèmes Automatisés, Angers, 2005.

[9] D.-K. NGUYEN et J. POLO, «Recherche du plus court chemin,» open city, 2000.

[10] Mémoire K. Safia, «Optimisation Des Paramétres D'une Cellule Photovoltaique Par Les Alqorithmes Génétique,» Université FERHAT ABBAS, Setif, 2014.

[11] D. Rebaïne, «Les graphes et leurs algorithmes,» Université du Québec à Chicoutimi, Québec (canada).

[12] Soulimane et Djellat, «Optimisation Par Colonie de Fourmis,» 2012.

[13] O. GERYVILLE, «Optimisation Par Colonie de Formis,» Université des Science et de la Technologie d'Oran, Oran, 2011/2012.

[14] Mémoire B. Sihem, «Optimisation de la production et de la structure d'électrique par les colonies de fourmis,» Université Jilali Laibés, sidi bel_abbés, 2007.

[15] D. Marco, «The Ant SystemMOptimization By a colony of cooperating agents,» Université Libre de Bruxelles, Bruxelles, 1996.

Annexe

On va présenter tous les données qui utilisé pour l'implémentation du système d'acheminement de véhicules en langage MATLAB.

Matrice de temps

Quatre matrices de taille (76,76) représentent tous les temps entre les nœuds, chaque matrice associée à la circulation de route.

- *Sans_circulation*

Cette matrice indique le temps dans les horaires où il n'y a pratiquement pas de trafic.

Columns 1 through 24

```
   0   60   30  Inf  120  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
  60    0  Inf   30  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
  30  Inf    0   30  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf   30   30    0   75  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  160  Inf
 120  Inf  Inf   75    0   95  Inf  Inf  Inf   45  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf   95    0   15  Inf   25  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   75  Inf  Inf
 Inf  Inf  Inf  Inf  Inf   15    0   10  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   30  Inf  Inf
 Inf  Inf  Inf  Inf  Inf   10    0   10  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   75  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf   25  Inf   10    0   25  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf   45  Inf  Inf  Inf   25    0   30  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   30    0   10  Inf   70  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   10    0   60  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   60    0  Inf  Inf  Inf   90  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf    0   35   45  Inf  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   30    0   40  Inf  Inf   70  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   45   40    0    5  Inf  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   90  Inf  Inf    5    0   10  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   10    0  Inf  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf   75  Inf  Inf  Inf  Inf  Inf  Inf   70  Inf  Inf  Inf    0   30  Inf  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   30    0   40  Inf  Inf
 Inf  Inf  Inf  Inf  Inf  Inf   30  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   40    0   35  Inf
 Inf  Inf  Inf  Inf  Inf   75  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   35    0   15
 Inf  Inf  Inf  160  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   15    0
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf   15
 Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
```

Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf

Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
25 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf

Columns 49 through 72

Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf

Columns 73 through 76

```
Inf   Inf   Inf   Inf
Inf   Inf   Inf   Inf
Inf   Inf   Inf   Inf
```