REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : SIOD/M2/2020

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours :

SIOD

Réalisation d'une méthode méta heuristique pour la résolution d'un problème NP-complet

---

Par :

TOUBAKH RIHANA

Soutenu le ../10/2020, devant le jury composé de :

…..	…..	Président
KALFALI Toufik	M.C.B	Rapporteur
…..	…..	Examinateur

Remerciements

Avont tout « alhamdou li lah».

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances à :

*** *kalfali Toufik*, notre encadrant de mémoire de fin d'étude, pour ses précieux conseils et son orientation ficelée tout au long de notre recherche.**

*** A nos familles et nos amis qui par leurs prières et leurs encouragements, on a pu surmonter**

tous les obstacles.

Nous tenons à remercier toute personne qui a participé de près ou de loin à l'exécution de ce modeste travail.

Dédicaces

À MES CHERS PARENTS

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être. Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.

Résumé

le problème de voyageur de commerce(PVC) consiste à trouver le min distance, disons ,s'agit d'un voyageur en commerce qui souhaite visiter n villes données en passant par chaque ville exactement une fois (établir une tournée). Le problème consiste à trouver la tournée de longueur minimale passant par toutes les villes et revenant au point de départ.

Dans ce mémoire nous présentons une nouvelle approche d'optimisation par recuit simule pour la résolution de ce problème.

Les résultats obtenus dans l'expérimentation montrent l'efficacité de l'approche propose.

Abstract

Travelling Salesman Problem (TSP) is to find the min distance, say, is a business traveler who wants to visit n given cities through each city exactly once (set up a tour).

Is to find the minimum length tour passing through all cities and returning to the starting point.

In this thesis, we present a new approach of optimization by simulated annealing for the resolution of this problem.

The results obtained in the experiment show the effectiveness of the proposed approach.

الملخص

مشكلة التجاري المسافر هي العثور على الحد الأدنى للمسافة، على سبيل المثال، مسافر بغرض العمل يريد زيارة مدن معينة يعبر كل مدينة مرة واحدة.

وهو العثور على الحد الأدنى لطول الجولة التي تسمح بالمرور عبر جميع المدن والعودة إلى نقطة البداية.

في هذه الرسالة نقدم طريقة جديدة للتحسين عن طريق التلدين المحاكي لحل هذه المشكلة.

النتائج التي تم الحصول عليها في التجربة تظهر فعالية النهج المقترح.

Table des matières

Chapitre 2 : Réalisation

Table de figure

Introduction Générale.

L'industrie du transport s'est fortement développée depuis la révolution industrielle. Dans une économie ou les besoins en consommation sont de plus en plus grandissants, son importance est devenue indiscutable aussi bien dans le secteur public que dans le prive. Les revenus qu'elle peut dégager ainsi que les frais qu'elle occasionne imposent la mise en place d'une politique et d'un système tout sauf arbitraire.

Pour en tirer les meilleurs bénéfices, une stratégie visant a maximiser les revenus et minimiser les frais doit être développée et optimisée. Des modelés mathématiques inspires de problèmes réels sont conçus pour permettre à la communauté scientifique dans le domaine de l'optimisation de mettre en œuvre des méthodes de différentes catégories.

Le problème du voyageur de commerce (TSP) et ses variantes sont des Problèmes d'optimisation combinatoire (POC) NP_ complets qui contribuent considérablement à la résolution de divers problèmes réels, y compris ceux des domaines du transport. D'autre part, le TSP fait partie des premiers problèmes auxquels s'attaquent la plupart des nouvelles Meta heuristiques afin de valider la méthode proposée, les instances du TSP étant de différentes topologies et difficultés.

Différentes méthodes d'optimisation permettent de résoudre les POC. Les approches exactes apportent la solution optimale dans un temps exponentiel, ce qui les rend pratiquement stériles face aux instances larges.

Les Meta heuristiques sont par contre beaucoup plus rapides, mais ne garantissent pas d'atteindre l'optimalité. Plusieurs travaux ont cependant émerge dans la littérature fournissant des solutions de plus en plus satisfaisantes.

Cette classe d'algorithmes permet d'avoir des solutions dans un temps assez court, ce qui lui permet d'être praticable pour les instances larges contrairement aux méthodes exactes. Mais d'un autre cote, elle ne donne aucune garantie sur la qualité de la solution (par rapport à l'optimale). Cette

contrainte a suscité l'intérêt des chercheurs afin de proposer des heuristiques pouvant fournir de bonnes solutions (voire même les optimales) avec une probabilité très élevée indépendamment de la topologie de l'instance.

Parmi les différentes classes de méta heuristiques proposées dans la littérature, la recherche locale fait partie des plus sollicitées. Sa capacité à exploiter un ensemble de solutions pour en tirer la meilleure a fait en sorte de constituer la base ou d'être une composante majeure de plusieurs travaux de

1-Introduction.

L'optimisation combinatoire occupe une place très importante en recherche Opérationnelle, elle se trouve au carrefour de la théorie des graphes, de la programmation Linéaire et de la programmation en nombres entiers. Son importance se justifie d'une part, Par la grande difficulte des problèmes d'optimisation et d'autre part, par de nombreuses applications pratiques pouvant être formulées sous forme d'un problème d'optimisation Combinatoire.

2-Défition de L'Optimisation Combinatoire.

L'Optimisation Combinatoire consiste à trouver un "meilleur" choix parmi un ensemble fini (souvent très grand) de possibilités. Nous explorons et exploitons les propriétés structurelles des problèmes ("bonnes" caractérisations, décompositions, etc) qui permettent de concevoir des algorithmes efficaces (exacts ou approchés) ou alors montrent que de tels algorithmes n'existent pas.

Combinatoire, mathématiques discrètes, théorie des graphes : trois termes synonymes, ou presque. Ils représentent une branche des mathématiques née au vingtième siècle grâce aux besoins de divers domaines :

- l'informatique
- l'organisation de la production à grande échelle
- la gestion des opérations militaires (voir les origines du terme **recherche opérationnelle**)
- l'économie
- ...

Un grand nombre de ramifications se sont par la suite développées, celles que nous traitons peuvent être regroupées sous le terme d'Optimisation **Combinatoire.**
L'**Optimisation Combinatoire** consiste à trouver la meilleure solution parmi un nombre fini (mais souvent très grand) de choix. C'est une branche de la « **Programmation Mathématique** » qui recouvre les méthodes qui servent à déterminer l'optimum d'une fonction sous des contraintes données. Il s'agit donc de **minimiser** une fonction sur un **ensemble fini**, mais **éventuellement très grand**, et **dont les propriétés mathématiques ne sont pas**

facilement caractérisables. La recherche de ces caractérisations, et des algorithmes d'optimisation qui les utilisent, constitue l'essentiel du travail de l'équipe. Nous nous basons sur notre expertise pour cerner de nouveaux problèmes, théoriques ou appliqués, et les analyser pour en extraire les propriétés fondamentales qui permettront de les résoudre ou de montrer que leur résolution est difficile. De plus, nous utilisons et développons des outils théoriques qui permettent de traiter ces problèmes avec des méthodes appropriées (exactes, heuristiques, algorithmes d'approximation, etc), qui peuvent être « ad hoc » ou « génériques ».
Grâce aux résultats obtenus et aux activités qui en découlent - séminaires, projets internationaux, exposés et cours, accueil d'un grand nombre de jeunes chercheurs - Grenoble est un centre attractif du domaine et plus généralement des mathématiques discrètes [1]

3-C'est quoi l'optimisation combinatoire ?

Beaucoup de problèmes d'ordre pratique ou théorique nécessite de prendre le meilleur choix, selon un critère donné, parmi un ensemble de choix possibles, souvent très large.
Ces problèmes ont été étudiés depuis longtemps dans des axes de recherches indépendants et ce n'est qu'au milieu du vingtième siècle qu'ils ont ´été mis dans une seule structure en ´établissant des relations entre eux. Cette structure est nommée optimisation combinatoire.

4-Quelques problèmes classiques d'optimisation Combinatoire.

4-1/Le problème du plus cours chemin.

Soit un graphe G = (X, V), où X est l'ensemble des sommets et V est l'ensemble des arcs. On appelle le problème du plus cours chemin le problème suivant : Etant donné un graphe G, nous associons à chaque arc u un nombre l(u) ≥ 0 que nous appellerons la longueur de l'arc u. Trouver un chemin élémentaire µ, allant d'un sommet a à un sommet b, soit aussi petite que possible.

4-2/Le problème du voyageur de commerce.

Le problème du voyageur de commerce est l'un des problèmes classiques d'optimisation Combinatoire. Il s'agit d'un voyageur en commerce qui souhaite visiter n villes données en passant par chaque ville exactement une fois (établir une tournée). Il commence par une ville quelconque et termine en retournant à la ville de d´épart. Le problème consiste à trouver la tournée de longueur

minimale passant par toutes les villes et revenant au point de départ. La notion de distance peut être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense, dans tous les cas, on parle de coût. On distinguera ce problème par la notation PVC.

4-3/Le problème de sac à dos.

Le probl`eme de sac à dos modélise des situations analogues au remplissage d’un sac à dos , étant donne un sac de capacité b et n objets, où chaque objet j possède un poids et une valeur wj, le problème qui se pose est : comment remplir le sac de sorte que le poids

Total des objets choisi n’excède pas la capacité du sac b tout en maximisant la valeur totale.

4-5/Le problème d'appariement minimal.

Le problème d'appariement minimal (Minimum Matching Problem (MMP)) est le suivant : il s'agit d'apparier les points deux à deux (il en faut donc un nombre pair) de façon que la longueur totale de l'appariement soit minimale.

L'ensemble des configurations est constitué d'appariements c'est-à-dire de configurations où chaque point est relié à un et un seul autre point (une configuration acceptable est donc composée de N/2 liens). La fonction de coût est la somme des longueurs des liens de l'appariement.

4-6/L'arbre couvrant minimal.

Le problème de l'arbre couvrant minimal (Minimum Spanning Tree (MST)) consiste à trouver l'arbre passant par tous les points (d'où le nom) dont la somme des longueurs des liens est minimale.

L'ensemble des configurations est l'ensemble des arbres couvrants, c'est-à-dire l'ensemble des configurations constituées de N-1 liens et telles qu'il existe toujours un chemin entre deux points quelconques.

5-Outils de modélisation des problèmes d'optimisation combinatoire.

Un problème d'optimisation combinatoire peut être modélisé en utilisant déférents formalismes, par exemple :

5-1/La théorie des graphes.

Les graphes s'introduisent de manière très naturelle comme support de modélisation, car ils permettent d'expliciter des relations de structure des problèmes d'optimisation combinatoire.

Des points et des fleches, se sont les ingrédients d'une théorie qui est année en 1736 avec la Communication d'Euler (1707 − 1783) dans laquelle il proposait une solution au célèbre Problème des sept ponts de Königsberg. Cette théorie est nommée la théorie des graphes.

Euler avait la curiosité de traverser les sept ponts de la ville de Königsberg une fois et une seule et à revenir à son point de départ. Euler représenta cette situation à l'aide d'un dessin où les sommets représentent les terres et les arêtes représentent les ponts.

Le problème des ponts de Königsberg est identique à celui consistant à tracer une figure géométrique sans lever le crayon et sans passer plusieurs fois sur un même trait.

5-2/La programmation linéaire.

Nous pouvons exprimer un probl`eme d'optimisation combinatoire sous la forme d'un programme linéaire en variables continus (P L)

5-3/programmation linéaires en nombre entiers.

Nous parlons de la programmation linéaires en nombres entiers lorsque le domaine des

solutions d'un programme linéaire est restreint a des valeurs entières, car c'est le cas que nous rencontrons dans de nombreuses situations réelles

6-Méthodes de résolutions.

Une recherche exhaustive par énumération explicite de toutes les solutions est impensable pour résoudre un probl`eme d'optimisation difficile en raison du temps de calcul induit.

Néanmoins, la résolution d'un tel problème peut se faire de manière exacte en utilisant des méthodes permettent d'obtenir une ou plusieurs solutions dont l'optimalité est garantie.

Mais comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du probl`eme, ces méthodes rencontrent généralement des difficulte avec les applications de taille importante.

Selon Laporte et al, (1987) [5], les méthodes exactes de par leur nature, ne peuvent s'appliquer qu'à des problèmes spécifiques.

Cependant dans certaines situations, on peut se contenter de solutions de bonne qualité, sans garantie d'optimalité, mais au profit d'un temps de calcul réduit.

On utilise pour cela une méthode approchée, adaptée au problème considéré, mais avec l'inconvénient de ne disposer en retour d'aucune information sur la qualité des solutions obtenues.

6-1/Méthodes exactes.

6-1-1/La méthode du Simplexe

La méthode du simplexe a été développée par George Dantzig en 1947.

Cette méthode est un procède itératif permettant d'atteindre progressivement sans l'aide d'un graphique la solution optimale d'un programme linéaire.

L'idée de l'algorithme consiste à partir d'un sommet quelconque du polyèdre et, à chaque itération, d'aller à un sommet adjacent s'il est possible d'en trouver un meilleur pour la fonction objectif. S'il n'y en a pas, l'algorithme s'arrête en concluant que le sommet courant est optimal.

En général, il y a plusieurs sommets adjacents au sommet courant qui sont meilleurs pour l'objectif. Il faut en sélectionner un seul, la règle de s´élection étant appelée Règle de pivot âge.

6-1-2/La procédure de séparation et d'évaluation

La procédure de séparation et d'évaluation progressive, en anglais Branch and Bound est une mise en application du principe latin Divide ut Regnes qui conseille de diviser ses ennemis pour régner plus aisément.

Cette méthode a été proposée par Little et al en (1963) pour résoudre le probl`eme du Voyageur de commerce puis, repris par d'autres sous déférentes variantes.

L'algorithme du Branch-and-Bound permet de séparer, de façon récursive, le problème initial en sous problèmes de cardinalité moindre, pour en faciliter sa résolution. Le cardinal des sous problèmes étant réduit en imposant des contraintes supplémentaires à l'ensemble à explorer. Une série de tests est ensuite appliquée à tous les sous problèmes génères, afin de supprimer de l'espace de recherche les sous problèmes qui ne peuvent pas mener à la solution optimale.

Cette recherche par décomposition de l'ensemble des solutions peut être représentée graphiquement par un arbre. C'est de cette représentation que vient le nom de "Méthode de recherche arborescente". [6]

6-1-3/méthode des coupes de Gomory

Les méthodes de coupes ont permis de résoudre les problèmes particuliers de type partitionnement ou recouvrement (Delorme 1947).

Leurs principe consiste à ajouter des contraintes linéaires, appelées coupes, n'excluant aucun point entier admissible une par une jusqu'à ce que la solution optimale de la relaxation soit entière.

Dans l'approche Branch and Bound que nous avons d´enveloppé précédemment, la recherche des solutions est faite en éclatant le polyèdre convexe d´limité par les contraintes en trois partis distinctes et à éliminer celle qui ne contient pas de solutions entières et à explorer les deux autres.

L'approche par les coupes de Gomory est complétement déférente dans le sens où elle coupe le polyèdre et se rapproche de la solution optimale entière d'itération en itération.

Gomory a d'enveloppé une technique qui permet d'identifier automatiquement des inégalités valides, appelées coupes de Gomory.

6-1-4/La méthode de coupes et de séparation

Cette technique est connue également sous le terme anglais Branch and Cut.

L'idée générale des méthodes de coupes est de résoudre un programme en nombres entiers comme une séquence de programmes linéaires.

Un algorithme de coupes et branchements (Branch-and-Cut algorithm) est une technique de séparation et évaluation dans laquelle on applique un algorithme de coupes pour calculer la borne de chaque sous-probl`eme.

Cette méthode introduite par Padberg et al, (1991) [7] pour le probl`eme du voyageur de commerce s'est avérée très efficace, elle est maintenant largement utilisée pour résoudre d'une manière exacte des problèmes d'optimisation combinatoire.

6-2/Méthodes approchées.

6-2-1/Meta heuristiques

Le terme Meta heuristique a été inventé par Fred Glover lors de la conception de la recherche tabou.

Ces m´méthodes sont nées après des mises au point poussées sur les heuristiques.

Leurs but, autant que pour les heuristiques, est de réussir à trouver un optimum global.

Pour cela, l'idée est à la fois de parcourir l'espace de recherche et d'explorer les zones qui paraissent prometteuses mais sans être " piégé " par un optimum local.

Leur fonctionnement, au contraire des heuristiques, est donc indépendant du problème traité. Théoriquement, l'utilisation des Meta heuristiques n'est pas vraiment justifie et les résultats théoriques sont plutôt mauvais.

Un grand nombre de met heuristiques existent, nous présentons quelques-unes dans la suite de ce chapitre, nous les avons classes par type : celles à solution unique et celles a population de solutions.

6-2-1/Meta heuristiques à solution unique

Nous présentons des algorithmes à solution unique tels que le recuit simulé et la méthode tabou.

6-2-1-1/Le Recuit Simulé

L'idée principale du recuit simulé tel qu'il a été proposé par Metropolis en 1953 est de simuler le comportement de la matière dans le processus du recuit très largement utilisé dans la métallurgie.

Le but est d'atteindre un état d'équilibre thermodynamique, cet état d'équilibre (où l'énergie est minimale) représente - dans la méthode du recuit simulé – la solution optimale d'un problème ; L'énergie du système sera calculé par une fonction coût (ou fonction objectif) spécifique à chaque problème (Kendall).

La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objectif, pour cela, un paramètre fictif de température a été ajouté par Kirkpatrick, Gelatt et Vecchi.En gros le principe consiste à générer successivement des configurations à partir d'une solution initiale S0 et d'une température initiale T0 qui diminuera tout au long du processus jusqu'à atteindre une température finale ou un état d'équilibre (optimum global).

L'algorithme du Recuit Simulé permet de résoudre le problème de minimum local (plus précisément dans le cas non linéaire).

En effet, une nouvelle solution de cout supérieur à la solution courante ne sera pas forcément rejetée, son acceptation sera déterminée aléatoirement en tenant compte de la déférence entre les coûts ainsi que d'un autre facteur appelé " température ".

Ce paramètre, la température, sert à prendre en compte le fait que plus le processus d'optimisation est avancé, moins on est prêt à accepter une solution plus coûteuse, ou alors, elle ne doit pas être trop coûteuse.

Par contre, au début, l'acceptation de solutions fortement coûteuses permettra de mieux explorer tout l'espace des solutions possibles et par là-même, d'accroître nos chances d'approcher le minimum global.

Cet algorithme ne garantit pas d'atteindre le minimum global mais pour peu que la décroissance de la température soit très lente, on va beaucoup s'en approcher.

6-2-1-2/L'algorithme tabou

La recherche tabou a été initiée au début des années 1980.

Elle a été introduite par Glover en (1986) [8].Cette méthode est utilisée pour résoudre des problèmes complexes et/ou de très grande taille (souvent NP-difficiles).

Celle-ci à plusieurs applications en programmation non linéaire.

L'idée de la recherche tabou consiste, à explorer le voisinage d'une solution initiale donnée, et à choisir dans ce dernier une autre solution qui minimise la fonction objective. Il est essentiel de noter que cette opération peut ne pas améliorer la valeur de la fonction objective.

C'est à partir de ce mécanisme que l'on échappe aux minima locaux. Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper.

C'est pourquoi il faut que l'heuristique ait de la mémoire, le mécanisme consiste à interdire (d'où le nom de tabou) certains mouvements ou certaines composantes de ce mouvement (l'exemple le plus simple est d'interdire les derniers mouvements).

Les positions déjà explorées sont conservées dans ce qu'on appelle la Liste Tabou d'une taille donne, qui est un paramètre ajustable de l'heuristique.

6-2-2/Méta heuristiques à population de solutions

6-2-2-1/La méthode par essaims particulaires

L'optimisation par essaims particulaires a été développée par Kennedy et al, (1995)[9],en s'inspirant du comportement social des individus qui ont tendance à imiter les comportements réussis qu'ils observent dans leur entourage tout en y apportant leur variations personnelles, ce qui offre un caractère adaptatif a la méthode.

De ce fait, cette technique est fondée sur la notion de coopération entre agents qui peuvent être vus comme des animaux peu intelligents ayant peu de m´mémoire et de facultés de raisonnement.

6-2-2-2/La méthode par colonie de fourmis

L'idée initiale de cette méthode provient de l'observation du comportement des fourmis lorsqu'elles cherchent de la nourriture.

En effet, celles-ci parviennent à trouver le chemin le plus court entre leur nid et une source de nourriture, sans pour autant avoir des capacités cognitives individuelles très développées.

Après cette étude, un principe étonnant fut révélé : les fourmis communiquent indirectement via leur environnement (on parle alors de "stigmergie"), en déposant des phéromones sur le sol pour éclairer leurs comparses sur le chemin qu'elles ont déjà parcouru.

Ainsi, lorsqu'elles reviennent au nid avec de la nourriture, les autres fourmis pourront suivre la piste de phéromones pour retrouver la source de nourriture.

Mais, plus important, les phéromones s'évaporant avec le temps, les chemins les plus courts seront forcément ceux qui auront une intensité de phéromones plus forte.

Ce qui, au final, mènera automatiquement les fourmis vers le chemin le plus court.[9]

L'algorithme de colonie de fourmis artificielles (Ant Colony Optimization ou ACO) a été proposé par Marco Dorigo lors de sa thèse de doctorat [10].

Il était initialement destiné à la recherche de chemins optimaux dans un graphe.

Il reprend la notion de système multi-agent, chaque agent étant une fourmi.

Une fourmi parcourt alors le graphe de manière aléatoire, mais avec une probabilité plus importante de suivre une arête du graphe, en fonction de la quantité de phéromones déposée dessus.

Lorsque le graphe est entièrement parcouru, elle laisse sur le chemin qu'elle a pris une quantité de phéromones proportionnelle à la longueur de ce chemin.

La piste la plus courte sera donc de plus en plus renforcée, et donc de plus en plus attractive.

L'exploration de l'espace de recherche se fait via un phénomène d'évaporation se produisant à chaque itération, et retirant une partie des phéromones présentes dans l'environnement. Sans ce dernier phénomène, l'algorithme serait facilement piégé dans un optimum local.

Dans les premières itérations, les phéromones sont déposées uniformément sur les arêtes graphe et les fourmis se déplacent donc aléatoirement.

Mais, au fur et à mesure des itérations, le phénomène d'évaporation et l'intensification" des fourmis sur les arêtes des chemins les plus courts, fait disparaître les phéromones des arêtes les moins intéressantes.

Ce processus continue jusqu'à ce que le plus court chemin (trouvé par les fourmis, ce n'est pas forcément l'optimum global) apparaisse comme une piste de phéromones.

L'algorithme s'est depuis optimisé, et permet maintenant de résoudre d'autres types de problèmes d'optimisation.

Par exemple, une fourmi est assimilée à une solution possible du problème, et peut se déplacer dans l'espace de recherche.

6-2-2-3/Les algorithmes évolutionnaires

Les algorithmes évolutionnaires sont des algorithmes d'optimisation qui s'appuient sur des techniques inspirées de la génétique et de l'évolution naturelle, Charles Darwin en 1859 propose ce mécanisme que l'on désigne sous le terme de darwinisme ou sélection darwinienne : sélection, croisement, mutation.

C'est au début des années 60 que John Holland, (1975) a commencé à s'intéresser à ce qui allait devenir les algorithmes évolutionnaires.

Ses travaux trouvent un premier aboutissement en 1975.

L'ouvrage de Goldberg (1989) a également largement contribué à les vulgariser.

Les quatre éléments fondamentaux des algorithmes évolutionnaires sont :

- L'évaluation du niveau d'adaptation d'un individu (évaluation de la fonction objective à optimiser),

- La sélection : représentant le choix des individus en fonction de leur niveau d'adaptation,

- le croisement : correspond au mélange entre individus,

– la mutation : traduisant une modification d'un individu.

Les algorithmes évolutionnaires servent à simuler le processus d'évolution d'une population. A partir, d'une population de n individus (n solutions d'un probléme donné), des opérateurs de sélection, croisement et mutation sont appliqués à l'ensemble de ces individus pour en définir des nouveaux.

La sélection a pour but de favoriser les meilleurs éléments de la population, le croisement et la mutation assurent une large exploration de l'espace de recherche de part en diversifiant la population d'individus.

De nouveaux individus vont être évalués et vont venir remplacer certains plus anciens ou moins bons, Ainsi la population des n individus évolue au cours du temps et contient des individus de mieux en mieux adaptés au problème.

Elle se dirige donc vers l'optimum.

Les critères d'arrêt de la méthode sont un nombre fixé de générations, une limite de convergence de la population, ou une population qui n'évolue plus suffisamment.

Les algorithmes évolutionnaires different des algorithmes classiques d'optimisation et de recherche essentiellement en quatre points fondamentaux :

- Ils utilisent un codage des éléments de l'espace de recherche,

- Ils recherchent une solution à partir d'une population de solutions et non pas à partir d'une seule solution,

- Ils n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité,...),

- Ils ne sont pas déterministes, et utilisent des règles de transition probabilistes.

7-Conclusion.

Dans ce chapitre, nous avons essayé de donner une définition des problèmes d'optimisation combinatoire, de les classifier et d'énumérer quelques méthodes de résolution en allant des méthodes exactes aux méthodes approchées.

Nous avons pu constater, qu'elles étaient adaptables à un très grand nombre de problèmes.

D'abord, nous avons présenté avec peu de détails le fonctionnement de l'algorithme du Simplexe puis la méthode de séparation et d'évaluation et ses dérivées.

Dans une deuxième étape, nous avons présente des méthodes approchées, d'abord les méta heuristiques un exemple sur les méthodes Meta heuristiques à solution unique et les méthodes Méta heuristiques à population de solutions.

1-Introdction

Dans le chapitre précédent, nous avons présenté la notion de l'Optimisation Combinatoire et définir Quelques problèmes classiques d'optimisation Combinatoire

Nous avons donné les déférentes méthodes de résolutions : les méthodes exactes et les méthodes approchées.

Dans ce chapitre, nous allons essayer d'appliquer notre approche en mettant en œuvre Plusieurs étapes, nous allons présenter l'environnement de travail, le langage de programmation et les outils que nous avons utilisés pour construire ce système. Par la suite, nous allons expliquer son mécanisme d'action et les résultats obtenus.

2-Environnement de développement.

Avant de commencer l'implémentation de notre application, nous préciserons d'abord les langages de programmation et les outils utilisés que nous avons trouvés choix compte tenu des avantages qu'ils offrent.

3-Environnement de travail.

L'implémentation de l'algorithme a été effectuée en langage java en utilisant l'environnement de programmation : **Eclipse**, sous système d'exploitation Windows **7**.
Les expérimentation ont été menées sur un pc de type **i5- 4210U** CPU @ 1.7GHZ 2.40GHZetde **RAM 4GO .**

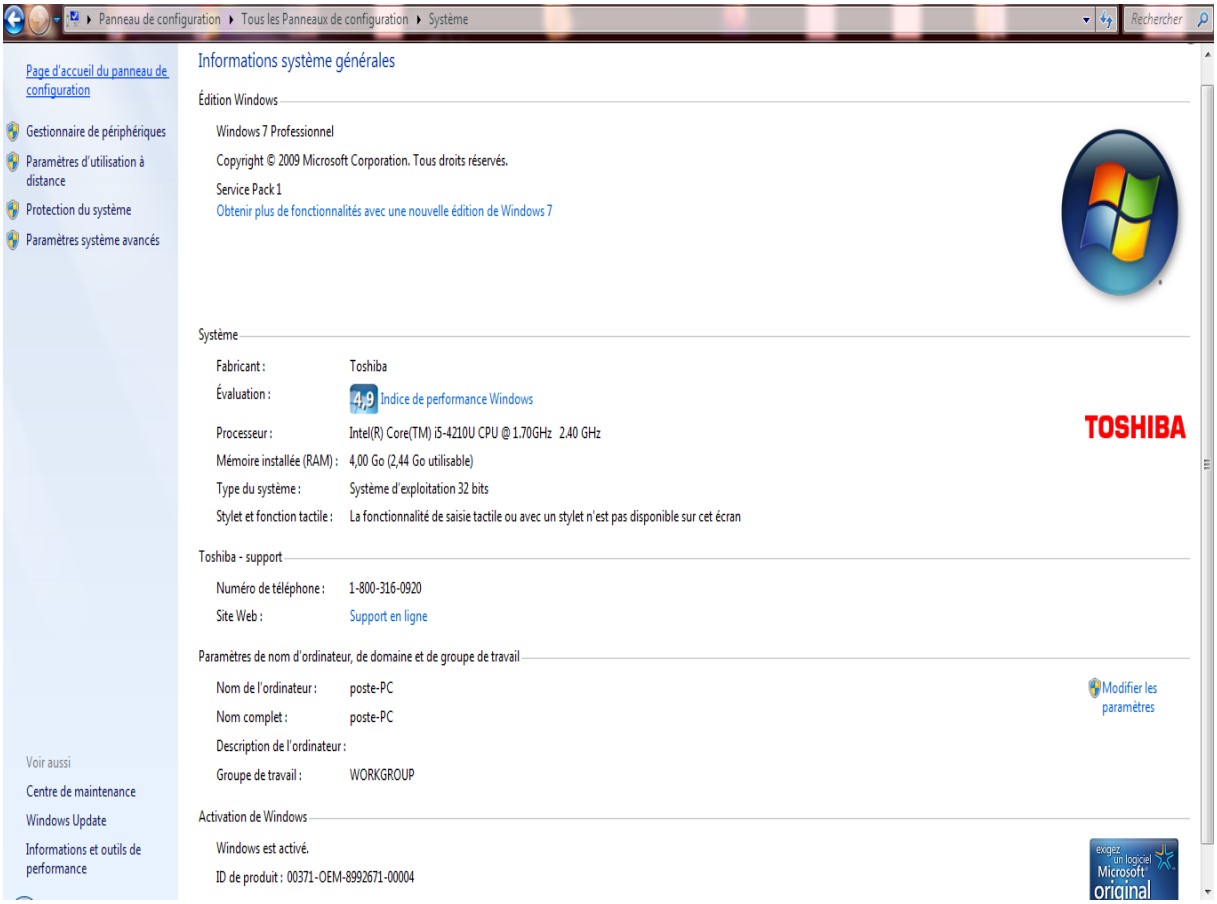


Figure 1: Système utilisé

4-Propriété de l'environnement Eclipse

Eclipse

Eclipse est un environnement de développement intégré (IDE) utilisé dans l'ordinateur Programmation. Il contient un espace de travail de base et un système de plug-in extensible pour personnaliser l'environnement.
Eclipse est écrit principalement en Java et son utilisation principale est le développement d'applications Java, mais il peut également être utilisé pour développer des applications dans d'autres langages de programmation via des plug-ins, notamment Ada, ABAP, C, C ++, C #, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (y compris le Framework Ruby on Rails), Rust, Scala, et Schème.
Il peut également être utilisé pour développer des documents avec Latex (via un Plug-in Tex Lipse) et des packages pour le logiciel Mathématique.

Développement Les environnements incluent les outils de développement Java Eclipse (JDT) pour Java et Scala, Eclipse CDT pour C / C ++ et Eclipse PDT pour PHP, entre autres.

La base de code initiale provient d'IBM Visual Age. Le kit de développement logiciel (SDK) Eclipse, qui comprend les outils de développement Java,

Figure 2:Logo de l'environnment Eclips

Est destiné aux développeurs Java. Les utilisateurs peuvent étendre ses capacités en installant plug-ins écrits pour la plate-forme Eclipse, tels que les kits d'outils de développement pour d'autres langages de programmation, et peuvent écrire et contribuer leurs propres modules d'extension.
Depuis l'introduction de l'implémentation OS Gi (Equinoxe) dans la version 3 d'Eclipse, les plug-ins peuvent être branchés-arrêtés dynamiquement et sont appelés bundles (OSGI).
Le kit de développement logiciel (SDK) Eclipse est un logiciel gratuit et open source,
Publié sous les termes de la licence publique Eclipse, bien qu'il soit incompatible avec la licence publique générale GNU.
C'était l'un des premiers IDE pour fonctionner sous GNU Classpath et il fonctionne sans problèmes sous IcedTea [12].

5-Langage De Programmation

De nos jours, il existe de nombreux langages de programmation, plus au moins dédiés à un type d'application particulier.
Parmi eux, notre choix s'est porté sur la Langue JAVA.

Java

Java est un langage de programmation généraliste basé sur les classes, orienté objet et conçu pour avoir le moins de dépendances d'implémentation possible.
Il est destiné à permettre aux développeurs d'applications d'écrire une fois, s'exécuter n'importe où (WORA), ce qui signifie que le code Java compilé peut s'exécuter sur tous plates-formes qui prennent en charge Java sans avoir besoin de recompilation.

Chapitre 2 : Réalisation

Java les applications sont généralement compilées en byte code qui peut s'exécuter sur n'importe quel Java machine virtuelle (JVM) quelle que soit l'architecture de l'ordinateur sous-jacente.
La syntaxe de Java est similaire à C et C ++, mais elle a moins installations de bas niveau que l'un d'eux.
À partir de 2019, Java était l'un des les langages de programmation les plus utilisés selon Git Hub, en particulier pour les applications Web client-serveur, avec 9 millions développeurs.
Java a été initialement développée par James Gosling chez Sun Microsystems et publié en 1995 en tant que composant principal de Sun Microsystems Java Plate-forme.

Les compilateurs Java d'implémentation d'origine et de référence, les machines virtuelles et les bibliothèques de classes ont été initialement publiées par Sun Sous licences propriétaires.

Depuis mai 2007, conformément à les spécifications du Java Commuait Processus, Sun avait renouvelé la de ses technologies Java sous la licence publique générale GNU.

Pendant ce temps, d'autres ont développé des implémentations alternatives de ces Sun technologies, telles que le compilateur GNU pour Java (compilateur byte code), GNU Classpath (bibliothèques standard) et IcedTea-Web (plugin de navigateur pour les applets).

Les dernières versions sont Java 14, sortie en mars 2020, et Java 11, une version de support à long terme (LTS) actuellement prise en charge, publiée le 25 septembre 2018 ;
Oracle a publié le dernier Java 8 LTS hérité mise à jour publique gratuite en janvier 2019 pour un usage commercial, alors qu'il prendra autrement toujours en charge Java 8 avec des mises à jour publiques pour un usage personnel jusqu'à au moins en décembre 2020.

Oracle (et d'autres) recommandent vivement de désinstaller les anciennes versions de Java en raison des risques graves dus à des problèmes non résolus problèmes de sécurité.

Étant donné que Java 9, 10, 12 et 13 ne sont plus pris en charge, Oracle conseille à ses utilisateurs de passer immédiatement à la dernière version (actuellement Java 14) ou une version LTS [11].

Figure 3:Logo de langage java

6-Le structure de donnée.

On utilisé un seul structure de donnée seulement le matrice Int mat[][] ;

```
nbP=listPoints.size();
int i,j;
int matrice[][] = new int[nbP][nbP];
for(i=0;i<nbP;i++){
    for(j=0;j<nbP;j++){
        matrice[i][j]=(int)getDis(listPoints.get(i),listPoints.get(j));
```

Figure 4:le code de structure

7-L'interface de l'application.

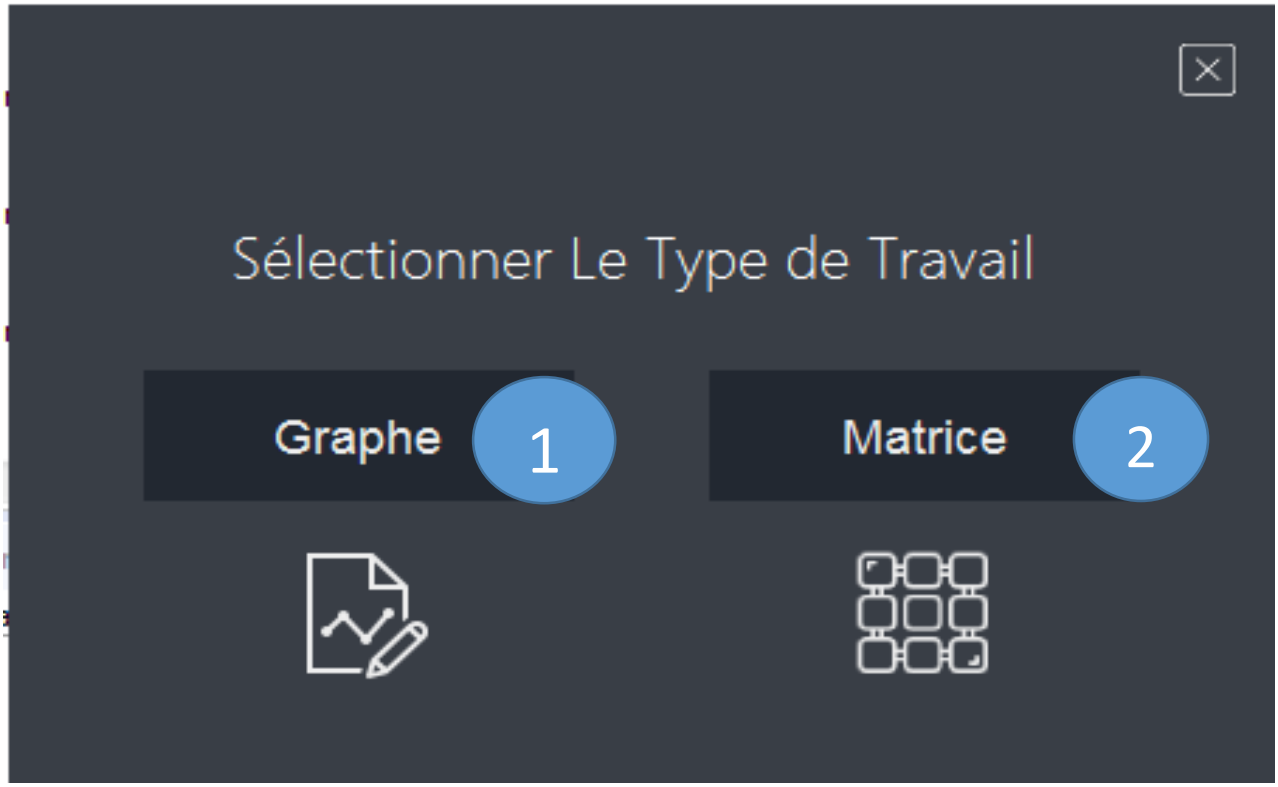


Figure 5:L'interface de type de travail

1- Botton accède au l'interface de travail graphe
2- Botton accède au l'interface de travail matrice

8-L'interface de graphe.

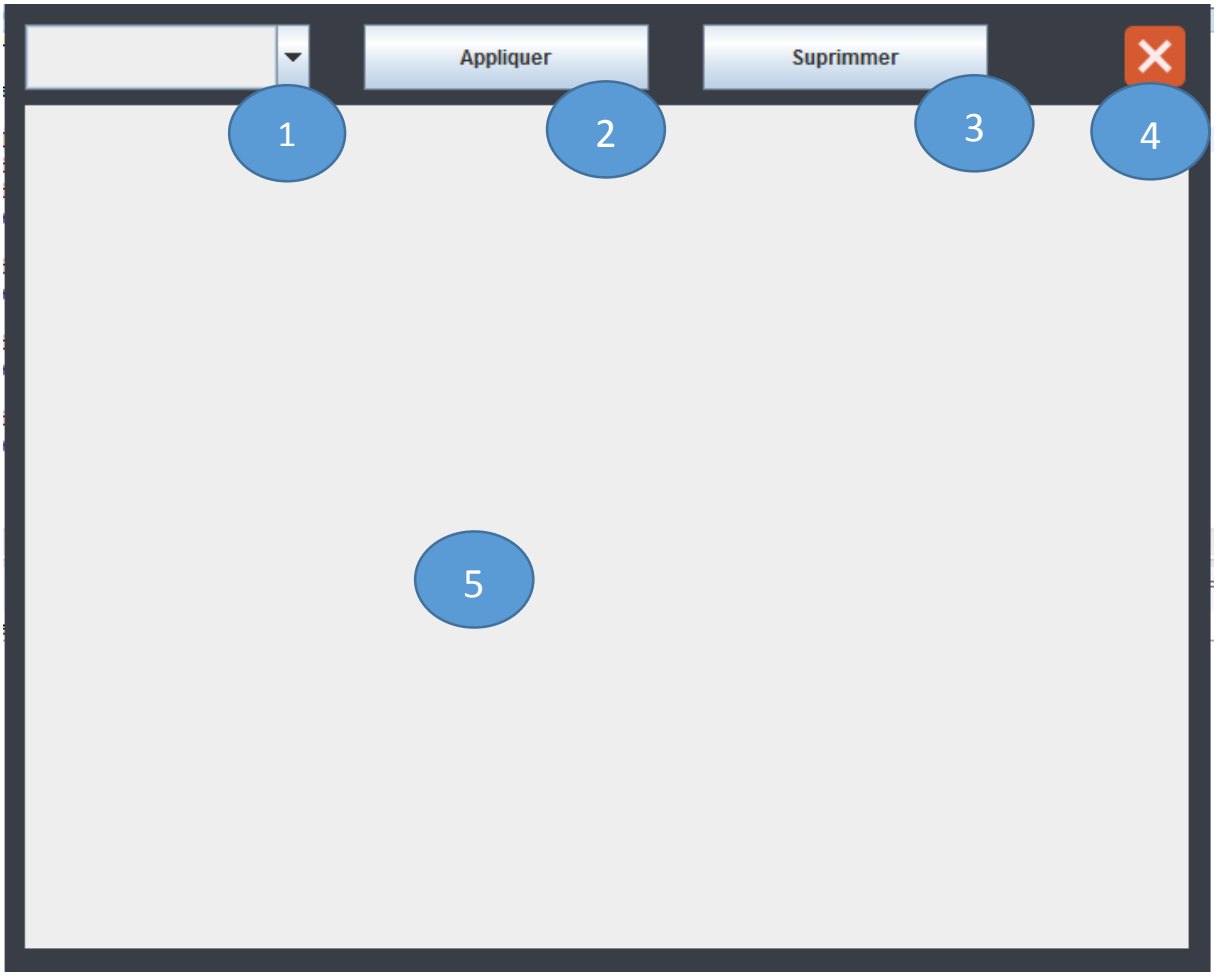


Figure 6:L'interface de type de travail graphique.

1- C'est un combo box vide pour écrire les villes la tempe de clique sur l'espace 5.
2- Botton appliquer pour applique l'algorithme de recuit simule.
3- Bouton supprimer pour effacer tous les points.
4- Botton pour fermer la fenêtre.
5- Espace pour click les points de ville.

9-Exemple de travail graphique.

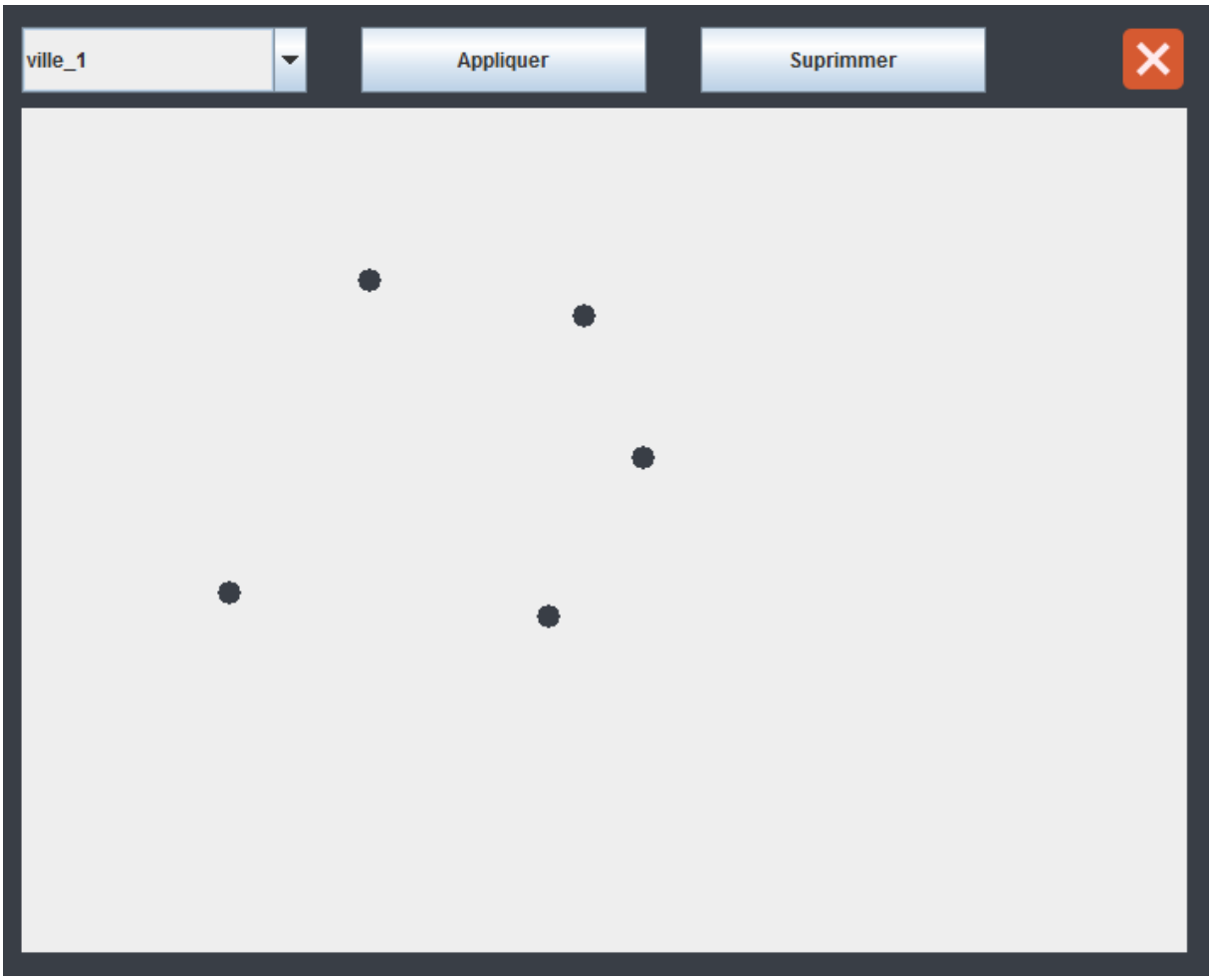


Figure 7:le point de ville

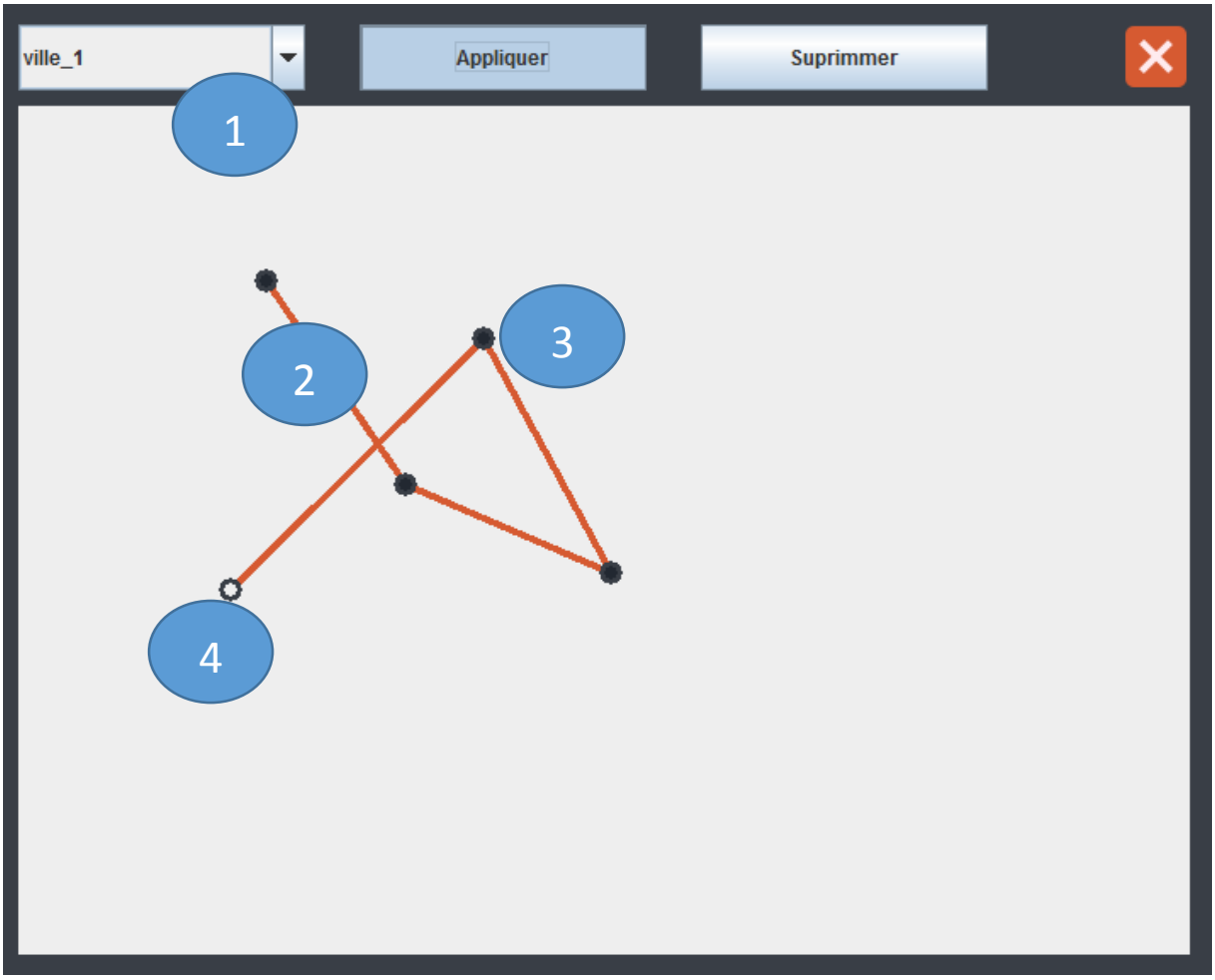


Figure 8:l'applique de l'algorithme dane cas intermédiaire.

1- Contient la ville de départ.
2- Présenté les distance entre les villes.
3- Point présenté les villes
4- Point vide pour présenter ville départ.

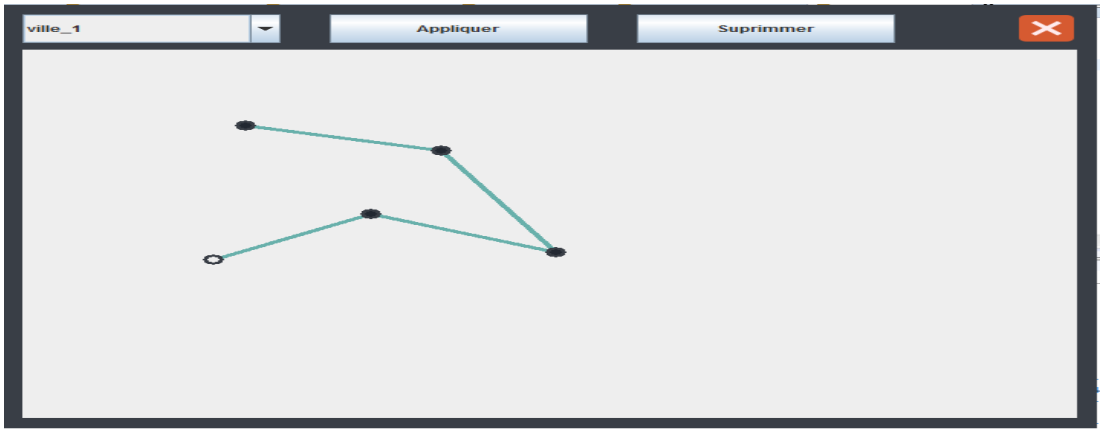


Figure 9:Le résultat finale.

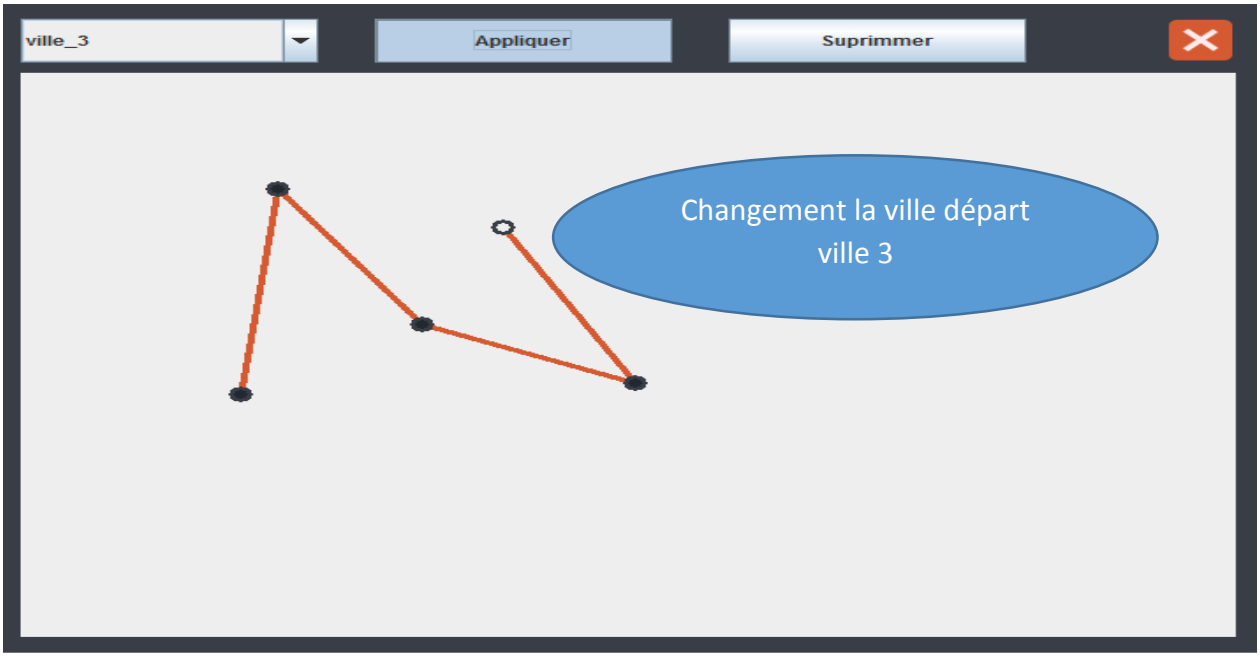


Figure 10:le départ ville 3,le cas intermédiaire.

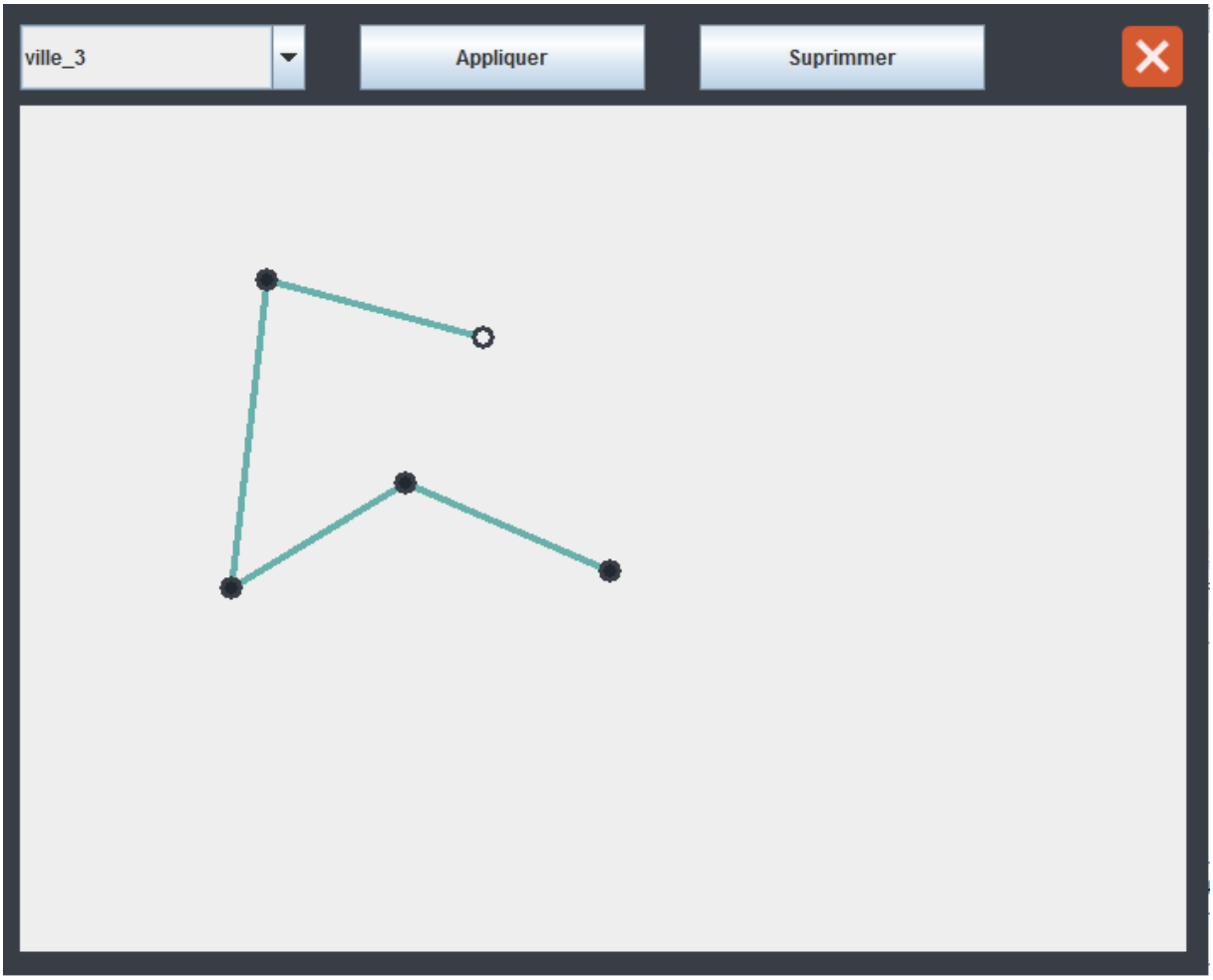


Figure 11: Le résultat finale 2.

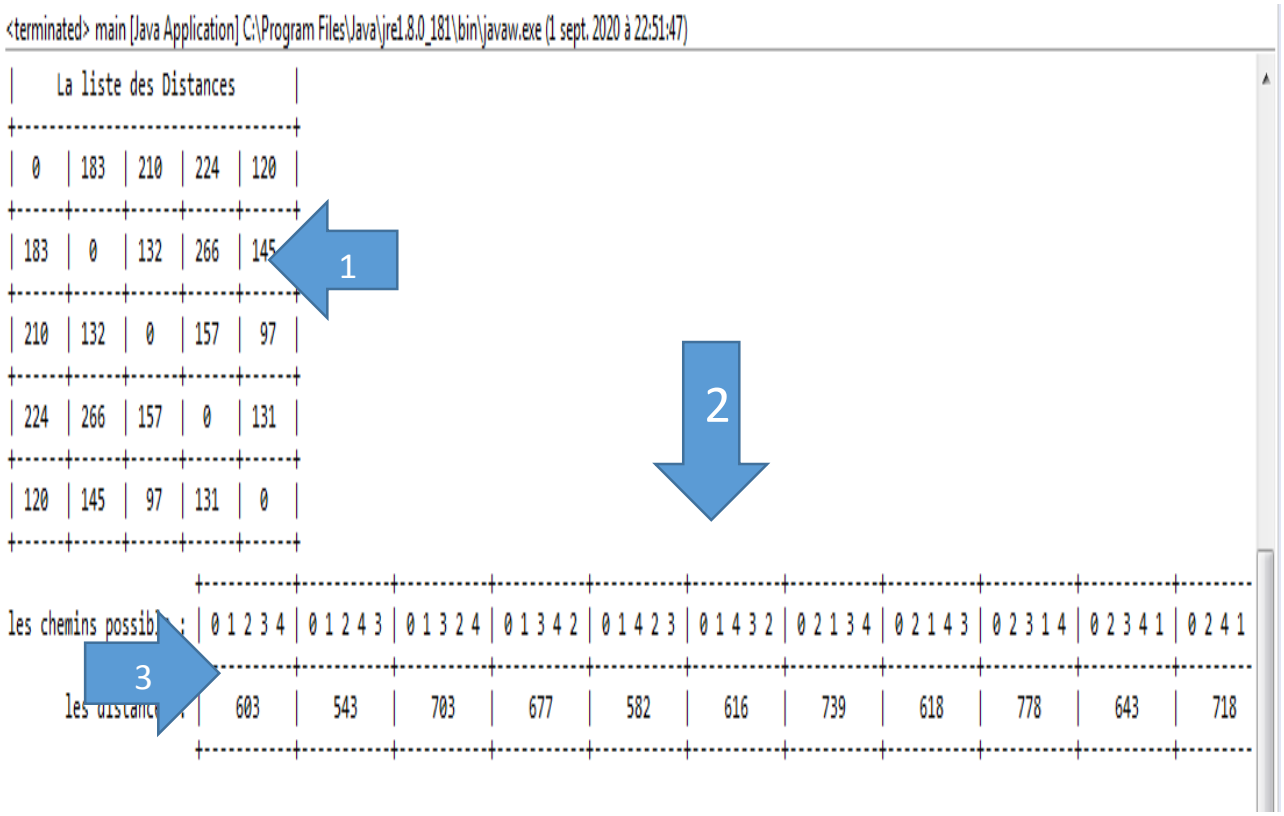


Figure 12: Le résultat de calcule pour le premier cas.

1- Tableau de de distances entre les villes.
2- Toutes les channes possibles.
3- Les distances finales des channes.

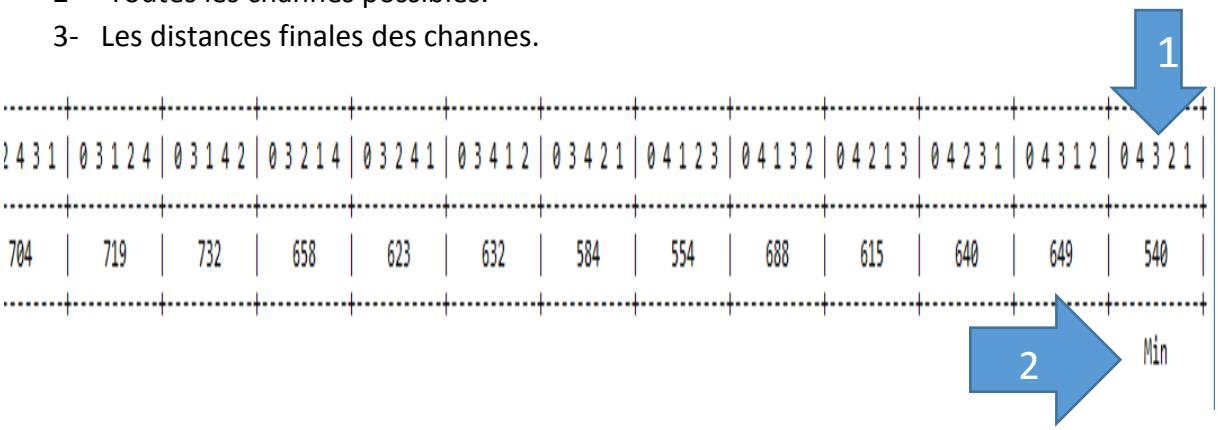
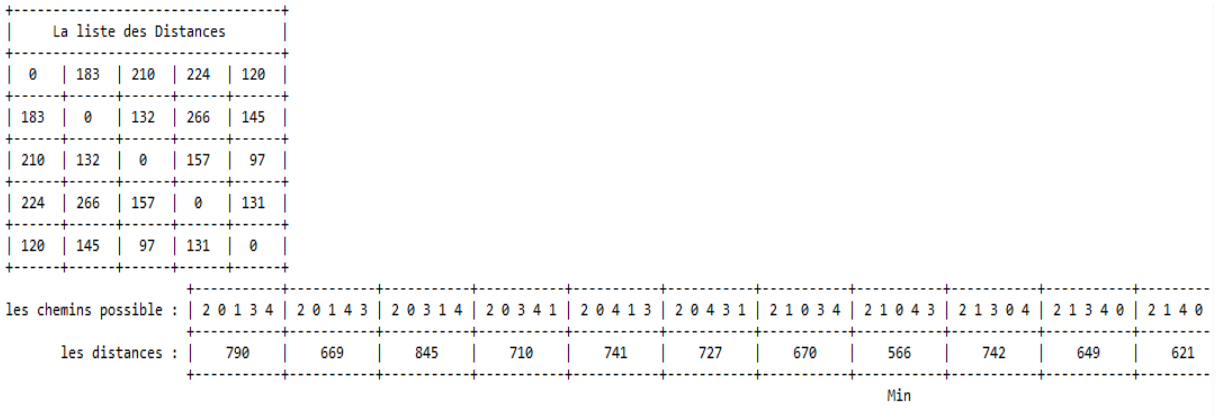


Figure 13: présenté le chaine minimal pour le premier cas.

1- 04321 La chaine min.
2- 548 la min distance.



La liste des Distances				
0	183	210	224	120
183	0	132	266	145
210	132	0	157	97
224	266	157	0	131
120	145	97	131	0

les chemins possible :	2 0 1 3 4	2 0 1 4 3	2 0 3 1 4	2 0 3 4 1	2 0 4 1 3	2 0 4 3 1	2 1 0 3 4	2 1 0 4 3	2 1 3 0 4	2 1 3 4 0	2 1 4 0
les distances :	790	669	845	710	741	727	670	566	742	649	621

Min

Figure 14: chemin minimal pour la premier cas.

10-L'interface de travail matrice.

Figure 15: l'interface pour les nombre des villes.

1- Espace pour entrer les nombre des villes.
2- Pour transfert à l'autre interface et continue la calcule.

11-Exemple travail matricielle.

Figure 16: entre 4 ville.

Figure 17: la distance entre v1 et v4.

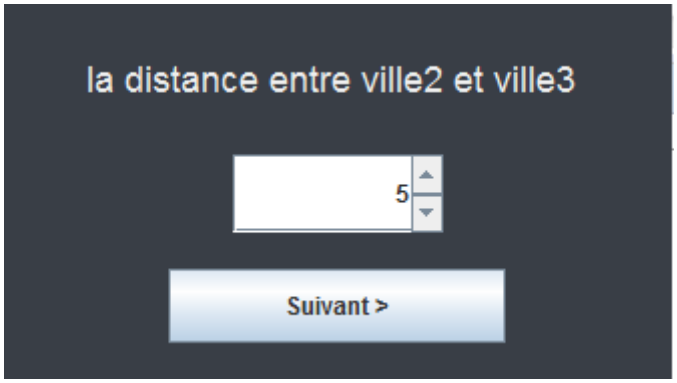


Figure 18: la distance entre v2 et v3.

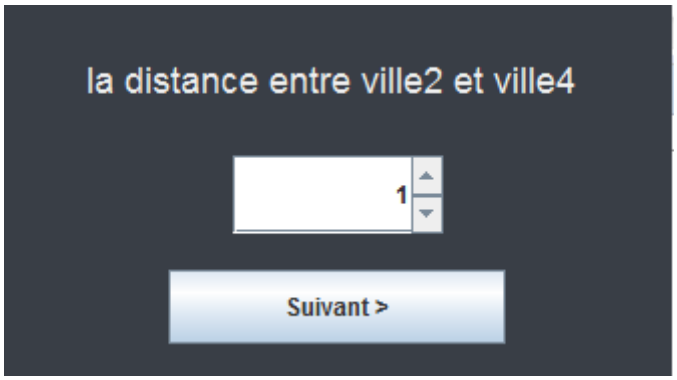


Figure 19: la distance entre v2 et v4.

Figure 20: la distance entre v3 et v4.

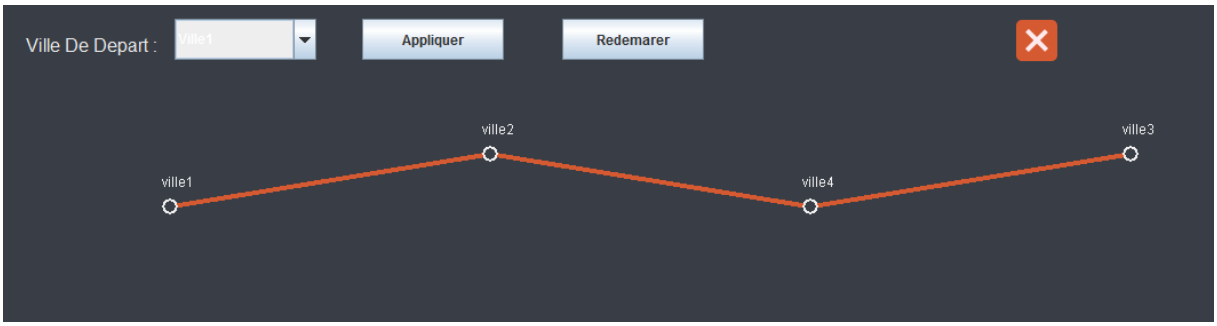


Figure 21: l'appliquer de l'algorithme, ville de départ v1.

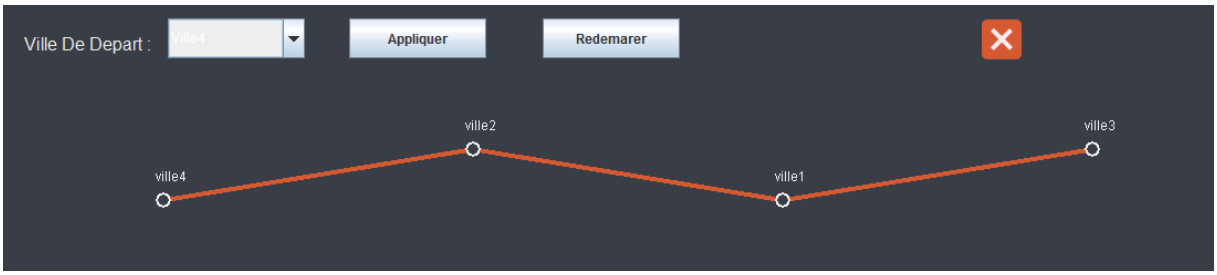


Figure 22 : Figure 22 : deuxième cas ville de départ v4.

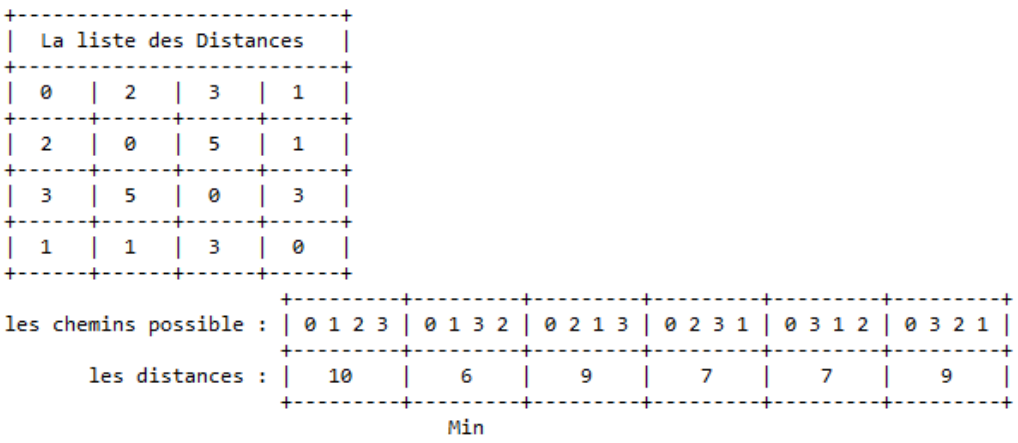


Figure 23: Le résultat de calcule pour le premier cas.

-Donc la chaine minimale est « 0231 »

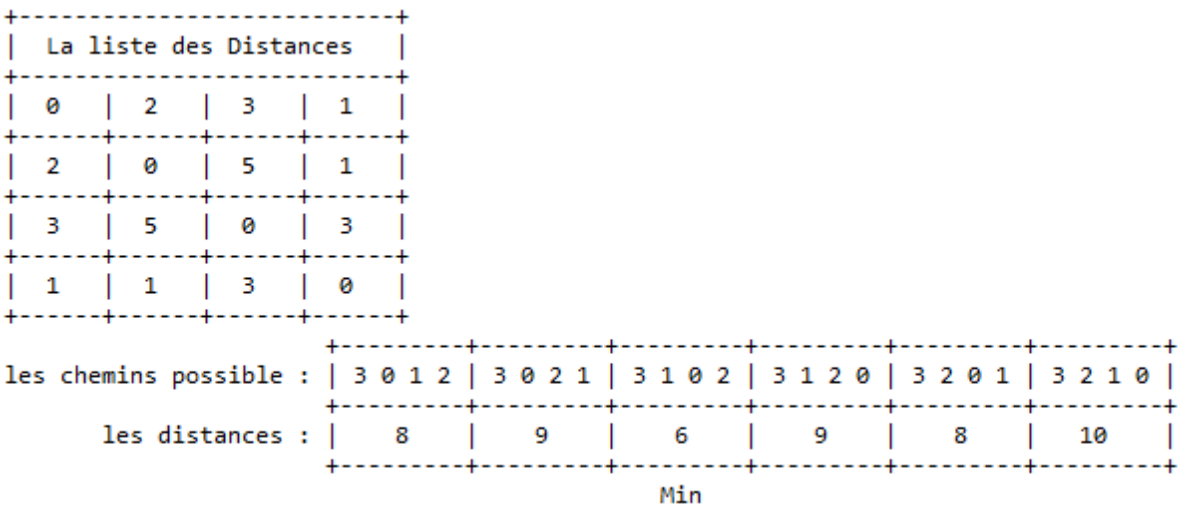


Figure 24 : Le résultat de calcule pour le deuxième cas.

-Donc la chaine minimale est « 3102 » .

12-Conclusion

Dans ce dernier chapitre nous avons présenté les résultats obtenus par l'algorithme proposé.

À savoir l'algorithme d'optimisation par recuit simulé. Pour résoudre le problème de voyageur de commerce (VC).

Les exemples donnés obtenus par méthode recuit simulé pour trouver une solution idéelle du problème de voyageur.

Nous concluons que le recuit simulé représente une bonne approche pour le problème de voyageur de commerce.

Conclusion Générale.

Le travail élaboré dans ce mémoire porte essentiellement sur le problème du voyageur de commerce.

L'objectif était d'appliquer l'algorithme par recuit simulé pour résoudre le problème NP-complet du voyageur de commerce.

Dans le premier chapitre nous avons présenté des notions liées à l'optimisation combinatoire, ensuite nous avons introduit la notion du méta heuristique en présentant des généralités sur ces derniers.

Ensuite, nous avons cité quelques problèmes de l'optimisation combinatoire, ainsi que ses différentes méthodes. Dans notre travail nous avons opté pour la méthode Recuit simulé pour atteindre notre objectif.

Dans le deuxième chapitre nous avons présenté les outils dont nous avions utilisé pour l'implémentation de notre système en démontrant les captures d'écran de notre exécution.

Bibliographie.

[1] Web site, url:"https://g-scop.grenoble-inp.fr/fr/recherche/optimisation-combinatoire",

[2] G. Laporte and Y. Nobert, Exact algorithms for the vehicle routing problem, Annals Of Discrete Mathematics, vol(31) :147-184, 1987.

[3] Michel Sakarovitch, Optimisation Combinatoire, Méthodes mathématiques et algorithmiques. Edition Herman. (1984).

[4] M. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large scale Symmetric traveling salesman problems, SIAM Review, vol(33) :60-100, 1991.

[5] F. Glover, Future paths for integer programming and links to articial intelligence. Computers and Operations Research, vol(5) :533-549, 1986.

[6] Batiste Autin Les métaheuriqtiques en Optimisation combinatoire. Conservatoire National des Arts et Métiers Paris. (2006).

[7] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. Elsevier Publishing, pages 134-142, 1992.

[8]Website, url:"https://en.wikipedia.org/wiki/java_programming_language",

[9] Website, url:"https://fr.wikipedia.org/wiki/eclipse_projet",