الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية و اللاسلكية

قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : Réseaux informatiques et systèmes réparties (RISR)

Thème

Traffic light optimization

- **Présenté par :**

OTMANI Mohamed Redha

MEDKOUR Mekki

- **Dirigé par :**

Dr. KOUIDRI Siham (Encadreur)

Année universitaire 2024-2025

ملخص

في الوقت الحاضر، أصبحت الازدحامات المرورية تحدياً كبيراً بسبب جداول إشارات المرور الثابتة التي لا تتكيف مع ظروف الحركة الحقيقية. لذا، تُعدُّ الأنظمة الذكية لإدارة الإشارات المرورية ضرورية للتحكم في حركة المرور الحضرية، وضمان أداء أفضل وسريان أكثر سلاسة. يركز بحثنا على تعزيز التنقل الحضري من خلال تحسين توقيت الإشارات في الوقت الفعلي. بعد استعراض التقدم المحرز في مجالي أنظمة إشارات المرور و الخوارزميات المتقدمة، نقترح نهجاً يقوم بضبط توقيت الإشارات ديناميكياً لتقليل الازدحام واستهلاك الوقود، مع هدف تقليل أوقات انتظار المركبات وطول صفوف الانتظار و زمن الرحلة .

الكلمات المفتاحية: إشارات المرور، النقل، شبكة الطرق، الخوارزميات المتقدمة، تقليل طول صفوف الانتظار، تقليل أوقات الانتظار.

Abstract

Nowadays, traffic congestion has become a major challenge due to fixed traffic-light schedules that do not adapt to real traffic conditions. Intelligent signaling systems are therefore crucial for managing urban traffic, ensuring better performance and smoother flow. Our research focuses on enhancing urban mobility through real-time optimization of traffic signals. After reviewing the state of the art in traffic signal control and metaheuristics, we propose an approach that dynamically adjusts signal timings to reduce congestion and fuel consumption, with the objective of minimizing vehicle waiting times, queue lengths and travel time.

Keywords: traffic lights, transportation, road network, metaheuristic, queue minimization, waiting time minimization.

Résumé

De nos jours, la congestion routière est devenue un défi majeur en raison des horaires fixes des feux de circulation, inadaptés aux conditions réelles de circulation. Les systèmes de signalisation intelligents sont donc essentiels à la gestion du trafic urbain, garantissant une meilleure performance et une fluidité accrue. Nos recherches portent sur l'amélioration de la mobilité urbaine grâce à l'optimisation en temps réel des feux de circulation. Après avoir examiné l'état de l'art en matière de contrôle des feux de circulation et de méta-heuristiques, nous proposons une approche qui ajuste dynamiquement la synchronisation des feux afin de réduire la congestion et la consommation de carburant, dans le but de minimiser les temps d'attente des véhicules,la longueur des files d'attente et temps de trajet.

Mots-clés : feux de circulation, transport, réseau routier, méta-heuristique, minimisation des files d'attente, minimisation du temps d'attente.

Remerciements

Tout d'abord, nous remercions ALLAH Tout-Puissant,qui nous a accordé la patience, la santé et la détermination nécessaires pour accomplir ce travail.

Nous tenons à exprimer notre profonde gratitude à notre encadrante, Dr. KOUIDRI Siham, du département d'informatique de l'Université de Saïda.

Ses conseils avisés, son soutien indéfectible et sa disponibilité constante ont été essentiels à notre réussite. La confiance qu'elle nous a témoignée, ainsi que ses encouragements et son aide précieuse tout au long de notre parcours universitaire, ont grandement facilité notre progression.

Nous souhaitons également exprimer notre sincère gratitude aux membres du jury du département d'informatique.

Leur volonté d'évaluer notre travail avec rigueur et bienveillance nous honore profondément.

Enfin, nous adressons nos remerciements aux enseignants et aux personnels administratifs de l'Université de Saïda, et plus particulièrement à ceux du département d'informatique, qui ont participé activement à notre formation académique.

Leur enseignement, leur soutien logistique et leurs encouragements ont été des atouts majeurs tout au long de notre parcours universitaire.

Dédicaces

Je dédie ce modeste travail

À ma mère, la femme la plus chère du monde,
la source de tendresse qui a tout donné sans rien recevoir,
je la remercie du fond de mon cœur.

À mon père, qui a toujours cru en moi,
et a mis à ma disposition tous les moyens nécessaires
pour que je réussisse dans mes études.

À mes chers frères.

À toute ma famille.

À toutes mes chères amies et collègues du Master 2 Réseaux.

À tous ceux qui ont contribué de près ou de loin
à la réalisation de ce modeste travail.

Dédicaces

À l'âme de mon cher père, Medkour Abdelkrim,
Tu es parti de ce monde le 14 novembre 2024, laissant dans mon cœur une douleur qui ne s'effacera jamais et une nostalgie qui ne s'éteint pas.
Le jour de ma remise de diplôme, tu me manques énormément. Je demande à Dieu de faire de ton dernier lieu de repos le paradis et de te couvrir de Sa miséricorde infinie.

À ma chère mère et mes chers frères et sœurs,
Vous êtes mon soutien et la source qui ne se tarit jamais.
Je prie Dieu de prolonger ta vie, de te protéger et de te garder comme lumière dans ma vie.

À mon camarade et cher frère,Otmani Redha,
Merci pour ta belle âme, ta loyauté et tes efforts nobles.
Je n'oublie pas de remercier tes parents et tes frères, qui ont été ton soutien à chaque étape du chemin.

À tous ceux qui nous ont tendu la main et nous ont adressé leurs prières,
Je dédie ce travail en témoignage de fidélité et d'affection.

Table des matières

Liste des Figures **6**

Liste des Tables **7**

Abréviations & Notations **8**

Introduction Générale **9**

1 État de l'art sur la gestion des feux de circulation **11**
1.1 Introduction . . . 11
1.2 Définition des feux de circulation . . . 11
1.3 Terminologie . . . 12
1.4 Histoire et l'évolution des feux de circulation . . . 12
1.5 Domaines d'utilisation des Feux de Circulation . . . 13
1.6 Système existant de gestion des feux de circulation . . . 14
1.6.1 TRANSYT . . . 14
1.6.2 SCOOT . . . 14
1.6.3 SCATS . . . 14
1.7 Outils théoriques de la gestion de trafic routier . . . 15
1.8 Systèmes de transport intelligents . . . 16
1.9 Inconvénients des feux tricolores . . . 17
1.10 Conclusion . . . 17

2 État de l'art sur les méta-heuristiques **18**
2.1 Introduction . . . 18
2.2 Définition des méta-heuristiques . . . 18
2.3 Principe général des méta-heuristiques . . . 18
2.4 Classifications des Algorithmes méta-heuristiques . . . 19
2.4.1 Algorithmes basés sur l'évolution . . . 19
2.4.2 Algorithme basé sur les essaims . . . 20
2.4.3 Algorithmes basés sur la physique . . . 20
2.4.4 Algorithmes basés sur l'humain . . . 20
2.5 Avantages des méta-heuristiques . . . 21
2.6 Désavantages des méta-heuristiques [1] . . . 21
2.7 Présentation des principales méta-heuristiques utilisées en optimisation 21
2.7.1 Méta-heuristiques à solution unique . . . 21
2.7.2 Méta-heuristiques à population de solutions . . . 27
2.8 Approches hybrides dans les méta-heuristiques . . . 35

2.8.1 Hybrides Collaboratifs . . . 35
2.8.2 Hybrides Intégratifs . . . 36
2.9 Applications des méta-heuristiques à l'optimisation du timing des feux de signalisation dans la littérature . . . 37
2.10 Conclusion . . . 41

3 Approche Proposée **42**
3.1 Introduction . . . 42
3.2 Description de la problématique . . . 42
3.3 Architecture générale . . . 43
3.4 PSO . . . 44
3.4.1 Initialiser les paramètres . . . 44
3.4.2 Initialiser les particules . . . 44
3.4.3 Mise à jour de $PBest$ . . . 45
3.4.4 Mise à jour de $GBest$ . . . 45
3.4.5 Mettre à jour la vitesse et la position . . . 46
3.5 Recuit simulé (SA) . . . 47
3.5.1 Initialisation . . . 48
3.5.2 Perturbation . . . 48
3.5.3 Critères d'acceptation . . . 48
3.5.4 Refroidissement . . . 49
3.6 Hybridation de PSO et SA . . . 50
3.7 Fonction Objectif . . . 50
3.7.1 Variables de Décision . . . 51
3.8 Organigramme de l'approche proposée . . . 52
3.9 Conclusion . . . 53

4 Expérimentations et Résultats **54**
4.1 Introduction . . . 54
4.2 Environnement de développement . . . 54
4.2.1 Python 3.12.4 . . . 54
4.2.2 Javascript . . . 55
4.2.3 HTML . . . 55
4.2.4 Simulateur SUMO . . . 55
4.3 Étude de cas : zone d'Alger . . . 56
4.3.1 Préparation du scénario . . . 57
4.4 Configuration des Simulations . . . 58
4.5 Interaction avec la simulation . . . 59
4.5.1 Communication avec SUMO via TraCI . . . 59
4.6 Présentation de l'application . . . 62
4.7 Résultats expérimentaux . . . 63
4.8 Discussion des résultats . . . 66
4.9 Conclusion . . . 67

Table des figures

2.1 Hybrides Collaboratifs [8] 36

3.1 Architecture générale 43
3.2 représentation de particule 45
3.3 organigramme d'optimisation par essaim de particules 47
3.4 Organigramme de recuit simulé [9] 49
3.5 organigramme de l'approche proposée 52

4.1 simulateur sumo [10] 56
4.2 Zone d'Alger 57
4.3 Zone d'Alger en OpenStreetMap 57
4.4 Zone d'Alger dans SUMO 58
4.5 Alger.sumocfg 59
4.6 Alger.net.xml 60
4.7 Configuration de l'itinéraire du véhicule *alger.rou.xml* 60
4.8 Simulation en cours d'exécution 61
4.9 Fin de la simulation 61
4.10 Le fichier *sim_data.csv* 62
4.11 Application web 63
4.12 Application web suite 63
4.13 Résultat du temps d'attente moyen 64
4.14 Comparaison des émissions de CO_2 65
4.15 Comparaison des temps de trajet 66

Liste des tableaux

2.1 Comparaison des méthodes d'optimisation des feux de circulation . . 41

4.1 Comparaison des temps d'attente moyens et de la réduction entre l'approche statique et l'approche dynamique 65
4.2 Réduction des émissions de CO_2 entre l'approche statique et PSO-SA 66

Abréviations & Notations

PSO	Particle Swarm Optimization
SA	Simulated Annealing
TRANSYT	Traffic Network Study Tool
SCOOT	Split Cycle Offset Optimization Technique
SCATS	Sydney Coordinated Adaptive Traffic System
FLC	Fuzzy Logic Controller
GA	Genetic Algorithm
VNS	Variable Neighborhood Search
GWO	Grey Wolf Optimizer
ACO	Ant Colony Optimization
SUMO	Simulation of Urban MObility
TraCI	Traffic Control Interface
OSM	OpenStreetMap
XML	Extensible Markup Language
CSV	Comma-Separated Values
GUI	Graphical User Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
DLR	German Aerospace Center
ODbL	Open Database License
ES6	ECMAScript 6
ES8	ECMAScript 8
V2V/V2I	Vehicle-to-Vehicle/Vehicle-to-Infrastructure

Introduction Générale

1. Contexte

La gestion du trafic urbain est un défi majeur dans les grandes villes, et Alger n'échappe pas à cette problématique. L'augmentation constante du nombre de véhicules engendre une congestion accrue, impactant la mobilité, la productivité et la qualité de vie des citoyens. Les systèmes traditionnels de contrôle des feux de circulation sont souvent basés sur des règles fixes qui ne s'adaptent pas aux variations dynamiques du trafic, ce qui limite leur efficacité et leur capacité à fluidifier la circulation. Pour relever ce défi, les techniques d'optimisation basées sur les méta-heuristiques représentent une alternative innovante. Ces approches permettent de modéliser et de résoudre des problèmes complexes en s'appuyant sur des algorithmes intelligents capables d'ajuster les temps de signalisation en fonction des conditions réelles du trafic.

2. Problématique

L'optimisation des feux de circulation repose sur la capacité à trouver une solution qui améliore la fluidité du trafic tout en minimisant les temps d'attente et les émissions polluantes. Cependant, les méthodes conventionnelles présentent des limites en raison de leur incapacité à s'adapter aux fluctuations du trafic en temps réel. La question centrale de cette étude est donc : **Comment une approche hybride combinant l'optimisation par essaim de particules (PSO) et l'algorithme de recuit simulé (SA) peut-elle améliorer la gestion des feux de circulation à Alger en exploitant efficacement l'espace de recherche**

3. Objectif du travail

L'objectif de cette étude est de proposer et d'évaluer une approche hybride PSO–SA pour optimiser la régulation du trafic à Alger. Cette approche vise à exploiter les avantages de ces deux méta-heuristiques :

- **L'optimisation par essaim de particules (PSO)** permet une exploration efficace de l'espace de recherche en s'appuyant sur l'intelligence collective des particules.
- **Le recuit simulé (SA)** contribue à un raffinement progressif des solutions en évitant les minima locaux et en améliorant la convergence vers une solution optimale.

Plus spécifiquement, cette étude s'attache à :

— **Développer** un modèle hybride PSO–SA capable d'explorer et d'exploiter l'espace de recherche pour optimiser la gestion du trafic.
— **Analyser** les limites des méthodes classiques de régulation des feux de signalisation.
— **Évaluer** l'impact de cette approche sur la fluidité du trafic, les temps d'attente des véhicules et les émissions polluantes.

Grâce à cette combinaison innovante, l'approche hybride PSO–SA permet d'équilibrer l'exploration et l'exploitation des solutions tout en garantissant une optimisation efficace et adaptative des feux de circulation. Cette étude contribue ainsi au développement de systèmes intelligents capables de répondre aux défis de la mobilité urbaine à Alger et dans d'autres métropoles.

4. Organisation du manuscrit

Le travail que nous avons réalisé dans le cadre du problème est résumé dans ce document, qui est structuré en quatre chapitres encadré par une introduction et conclusion générale :

Le premier chapitre est dédié à un aperçu général sur le trafic. Il présente un rappel de quelques concepts de base sur les variables de trafic et la classification des modèles.

Le second chapitre présentera les différentes méta-heuristiques existantes, leur principe de fonctionnement, leurs avantages/inconvénients, et leur utilisation pour résoudre des problèmes d'optimisation complexes comme celui des feux de circulation.

Dans le chapitre trois, nous exposons l'approche d'optimisation appliquée dans la cadre de notre travail, à savoir son principe, ses variables, la fonction d'évaluation utilisée, ...etc.

Les expérimentations réalisées et les résultats obtenus par l'approche proposée sont exposées et analysées en dernier chapitre. Enfin, Une synthèse et un ensemble de perspectives feront l'objet de notre conclusion.

Chapitre 1

État de l'art sur la gestion des feux de circulation

1.1 Introduction

Un réseau routier est conçu pour permettre aux usagers de se déplacer. Dans ce contexte, il se compose d'un réseau de routes à nombre de voies différent, en fonction des zones d'activité et des zones résidentielles. Les intersections étant inévitables en zone urbaine en raison de leur densité, elles jouent un rôle crucial dans la gestion des flux de véhicules conflictuels, la sécurité, la prévention des embouteillages et la possibilité de changer d'itinéraire. Ces intersections, ou parfois les ronds-points, régulent la circulation et appliquent généralement les règles de priorité, ou sont équipées de feux de circulation pour gérer les situations particulièrement dangereuses.

Dans ce chapitre, nous présentons d'abord les concepts de base des feux de circulation et leur signification. Nous examinons ensuite leur histoire, leur évolution et leurs domaines d'application. Nous présenterons également quelques systèmes de gestion des feux de circulation existants, des outils théoriques pour la gestion du trafic routier et des systèmes de transport intelligents. Nous conclurons ce chapitre par quelques inconvénients des feux tricolores.

1.2 Définition des feux de circulation

Les feux de circulation sont des dispositifs généralement placés aux intersections pour contrôler l'ordre de passage des véhicules. Leur objectif principal est d'éviter les conflits à une intersection, malgré les embouteillages ou la règle de priorité [13].

Les feux de circulation destinés aux véhicules motorisés sont généralement tricolores, avec possibilité d'ajouter des flèches directionnelles. Ceux destinés aux piétons sont généralement bicolores et se caractérisent souvent par la représentation d'une silhouette de piéton. Les feux de circulation destinés aux cyclistes se distinguent par l'image d'un vélo.

Un feu de circulation est un élément essentiel de la signalisation routière. Il s'agit

d'un système électronique contrôlé conçu pour gérer efficacement la circulation des véhicules et des piétons aux intersections. Il est principalement composé de trois couleurs,Le feu rouge indique l'obligation d'arrêter les véhicules.Le feu orange, qui ne dure que quelques secondes, indique le passage du vert au rouge. Le feu vert indique ensuite que les véhicules peuvent circuler [13].

Le système de feux de circulation est un système efficace pour la gestion du trafic. Il aide à éviter les embouteillages et les collisions lors du franchissement d'une intersection. Cependant, ce système génère le plus de retard car il donne la priorité à un ou deux flux de trafic non conflictuels à la fois tout en exigeant que tous les autres flux entrants s'arrêtent systématiquement [14].

1.3 Terminologie

- **Flux de véhicules** : Un flux de véhicules regroupe tous les véhicules entrant par une voie spécifique et sortant par une autre. À une intersection, le trafic est composé de plusieurs flux de véhicules, chacun ayant une source distincte. Deux flux sont considérés comme compatibles s'ils peuvent traverser l'intersection simultanément sans conflit. Une intersection régulée par des feux de signalisation comprend plusieurs feux tricolores installés aux différentes voies d'entrée. [14].
- **Cycle d'un feu** : Le cycle d'un feu correspond à la durée qui sépare deux occurrences successives d'une même phase au sein d'une intersection. Il est défini par une séquence ordonnée de phases [14].
- **Phase d'un feu** : Une phase d'un feu représente une période au cours de laquelle un ou plusieurs flux cohérents sont autorisés à traverser l'intersection en toute sécurité [14].
- **Intersection** : Une intersection désigne un point de rencontre entre deux ou plusieurs routes sur une chaussée. Elle peut prendre différentes formes, comme un carrefour à quatre branches ou une jonction en T [15].

1.4 Histoire et l'évolution des feux de circulation

Au début des années 1900 [16], le monde connaissait une croissance rapide, et avec l'industrialisation croissante, les villes devenaient de plus en plus congestionnées. Parallèlement, l'invention de l'automobile entraînait une augmentation significative du trafic routier, ce qui rendait nécessaire un meilleur système de gestion de la circulation.

En 1912, un policier américain, Lester Wire, préoccupé par l'augmentation du trafic, eut l'idée d'un premier feu de signalisation électrique. Selon sa conception, les premiers phares furent installés à Cleveland, dans l'Ohio, le 5 août 1914, à l'angle de la 105th et de Euclid Avenue.

1914 : le premier signal électrique à commande manuelle arrive à Cleveland : Tout comme le sémaphore de 1910 [16], le premier signal électrique utilisait des mots. Cependant, ces mots n'étaient plus affichés sur les bras qui se soulevaient et tombaient. Les mots "Stop" ou "Move" étaient maintenant illuminés. Les poteaux s'éclairaient sur chacun des quatre coins d'une intersection. Toutefois, ce système n'était pas automatisé. Une cabine avec un opérateur était nécessaire pour actionner les commutateurs. Le système permettait aux policiers de passer du centre de

la rue à un coin, leur offrant la possibilité de surveiller la circulation. Si un véhicule d'urgence arrivait, l'agent pouvait actionner un commutateur et dégager l'intersection en allumant tous les feux rouges, permettant ainsi au véhicule d'urgence de passer sans obstacle.

En 1918, des signaux tricolores, actionnés manuellement depuis une tour située au milieu de la rue, furent installés à New York.

En 1920 [16], un policier nommé William Potts, à Detroit, Michigan, inventa les premiers feux de circulation à quatre couleurs et à trois couleurs. En plus des couleurs rouge et verte, une troisième couleur, l'orange (ou jaune), fut introduite. Detroit devint la première ville à mettre en place des feux tricolores. Dans les années 1920, plusieurs feux de signalisation automatisés furent installés dans les grandes villes du monde. Le modèle de feu de signalisation moderne utilise encore ce célèbre système en forme de T avec trois couleurs distinctes.

Les premières lumières de ce type apparurent en Grande-Bretagne, à Londres, en 1925, à l'intersection entre St James Street et Piccadilly. Elles étaient actionnées manuellement par des policiers à l'aide de commutateurs. Des signaux automatiques, fonctionnant sur un intervalle de temps, furent installés à Wolverhampton en 1926. Les Les premiers feux de signalisation activés par un véhicule en Grande-Bretagne ont été installés à l'intersection de Gracechurch Street et Cornhill dans la City en 1932. Depuis, les feux de signalisation normalisés rouge-orange-vert sont désormais adoptés partout dans le monde [16].

1.5 Domaines d'utilisation des Feux de Circulation

Les feux de circulation sont utilisés dans de nombreuses situations pour coordonner la circulation des voitures et des piétons, réduire les embouteillages et diminuer les risques d'accidents. Ils sont essentiels à la gestion du trafic et à la sécurité routière [17].

Voici les exemples suivants :

1. **Gestion du trafic urbain** :
   Les feux de signalisation sont largement utilisés en milieu urbain pour réduire les embouteillages et coordonner la circulation aux passages à niveau. Ils permettent d'augmenter la fluidité du trafic et de réduire les conflits entre les nombreux usagers de la voie en synchronisant les cycles des feux avec la circulation.
2. **Facilitation de la traversée des piétons** :
   Des feux dédiés aux piétons, souvent équipés de pictogrammes spécifiques, assurent la sécurité des piétons lors de la traversée des chaussées.
3. **Protection des obstacles intermittents** :
   Les feux sont utilisés pour signaler la présence d'obstacles temporaires ou intermittents, tels que les passages à niveau, les ponts mobiles ou les zones sujettes aux avalanches, protégeant ainsi les usagers de la route.
4. **Contrôle d'accès aux voies rapides** :
   Des feux de signalisation sont utilisés pour réguler l'accès aux voies rapides, assurant une entrée sécurisée et ordonnée des véhicules sur ces axes à fort trafic.

1.6 Système existant de gestion des feux de circulation

Plusieurs modèles de gestion des feux de circulation ont été développés et commercialisés. Ces systèmes reposent soit sur l'utilisation de capteurs capables de détecter en temps réel la position des véhicules, soit sur des méthodes de simulation des déplacements, voire sur une combinaison des deux approches.

1.6.1 TRANSYT

TRANSYT (Traffic Network Study Tool) est un logiciel d'optimisation des feux de circulation développé par le TRL (Transport Research Laboratory) au Royaume-Uni. Sa première version a été publiée en 1967 [18]. Il est largement utilisé pour simuler et optimiser la circulation aux intersections signalisées. TRANSYT commence par la collecte de données, notamment le nombre de véhicules, leur vitesse et les modèle d'arrivée. Il prend également en compte les informations de synchronisation des feux, telles que la séquence des phases, la durée des cycles et la géométrie du réseau routier. TRANSYT utilise ensuite ces données pour simuler les mouvements des véhicules à l'aide d'un modèle macroscopique de flux de trafic afin d'estimer les niveaux de congestion, la longueur des files d'attente et les temps de trajet, afin d'évaluer l'efficacité des réglages actuels des feux. TRANSYT utilise l'indice de performance (PI) pour l'optimisation. Le PI mesure l'efficacité des réglages des feux en évaluant le retard, le nombre d'arrêts et la longueur des files d'attente. TRANSYT ajuste ensuite les paramètres des feux, tels que la durée des cycles et la répartition des phases. Ces ajustements visent à synchroniser les phases vertes sur plusieurs intersections.
La version actuelle de TRANSYT est la V17 sortie en 2024 [19].

1.6.2 SCOOT

The Split Cycle Offset Optimization Technique (SCOOT) est un système adaptatif de gestion du trafic en temps réel conçu pour coordonner et contrôler les feux de circulation sur un réseau routier urbain [20]. Développé par le Transport Research Laboratory (TRL) au Royaume-Uni en 1979, SCOOT utilise des détecteurs de véhicules (généralement des boucles inductives ou des caméras) placés en amont des intersections, permettant de mesurer des paramètres tels que le flux de trafic, la longueur des files d'attente et la vitesse. Après la collecte des données, SCOOT utilise un modèle mathématique pour représenter les conditions de circulation, permettant de prédire les évolutions futures du trafic. Le système se concentre sur l'optimisation du temps de cycle, l'allocation des temps verts et le décalage. SCOOT vise à réduire les retards, à minimiser les arrêts et à améliorer l'efficacité globale du trafic.

1.6.3 SCATS

Le **SCATS** (Système de gestion adaptative du trafic coordonné de Sydney) a été conçu à l'origine pour Sydney et d'autres villes australiennes [21]. Ce système est totalement adaptatif et fonctionne selon un principe hiérarchique, ce qui permet de

répartir le contrôle sur le réseau. Des contrôleurs intermédiaires sont insérés entre les capteurs de terrain et le centre de contrôle, ce qui allège la charge du système et divise le contrôle en plusieurs zones. Tous les éléments du système communiquent de manière synchronisée.

De manière similaire à **SCOOT**, **SCATS** ajuste les cycles de feux et d'autres paramètres en fonction des données recueillies, dans le but de réduire les délais et les arrêts. Toutefois, la stratégie utilisée diffère. Les informations obtenues servent à choisir un plan de feux parmi une large bibliothèque, et le système utilise ce plan pour adapter les cycles de feux. À la différence de SCOOT, les détecteurs sont placés uniquement aux intersections.

Le fonctionnement du système repose sur des bibliothèques spécifiques pour la durée des cycles, les décalages et les temps de feu vert, ainsi que sur un algorithme de régulation en temps réel qui permet de générer dynamiquement le plan de feux. Ce plan n'est pas stocké dans une bibliothèque, mais est reconstruit à chaque ajustement.

1.7 Outils théoriques de la gestion de trafic routier

1. Logique floue :
La logique floue est une méthode mathématique permettant de représenter le flou et l'incertitude dans la prise de décision. Elle permet d'obtenir des vérités partielles. Par exemple, pour la température, une valeur de 25 °C peut être considérée comme moyenne avec une valeur de 60 %, basse avec une valeur de 40 % et élevée avec une valeur de 0 %, et c'est en fonction de ces valeurs que les décisions sont prises. Ce principe a été utilisé par divers auteurs pour la gestion des feux de circulation, offrant une solution plus simple que les méthodes d'optimisation mathématiques traditionnelles, souvent complexes.

L'un des travaux connexes utilisant la logique floue est [22]. Ce travail a développé un feu de circulation basé sur la logique floue avec une durée de cycle dynamique, visant à réduire les temps d'attente des véhicules aux intersections en tenant compte de la densité et du flux du trafic. Ce principe est pourtant simple et pourrait être appliqué à des problèmes plus complexes. Il offre un meilleur contrôle du comportement du système, réduisant ainsi le caractère aléatoire et la logique construite robuste. Cependant, l'établissement de règles précises et de fonctions d'appartenance dans les systèmes de logique floue peut s'avérer difficile et nécessite des connaissances approfondies [23].

2. Algorithmes du plus court chemin :
Les algorithmes du plus court chemin sont couramment utilisés dans le secteur des transports, par exemple :
Analyse des performances des réseaux de transport : Ces algorithmes sont très utiles à la conception des réseaux routiers. Avant toute conception, il est essentiel d'effectuer une phase de modélisation

et d'analyse des performances. Cette étape permet de déterminer l'emplacement optimal des lignes de transport, des arrêts et des stations, ce qui permet d'éviter les blocages dus à une mauvaise conception.
Optimisation des feux de circulation : Dans les systèmes de gestion du trafic, ces algorithmes peuvent être utilisés pour modifier la synchronisation des feux de circulation en fonction des itinéraires les plus fréquentés, réduisant ainsi la congestion et améliorant la fluidité du trafic.
des algorithmes tels que l'algorithme de Bellman-Ford-Moore, l'algorithme de Prim, l'algorithme de Kruskal et l'algorithme de Dijkstra...

1.8 Systèmes de transport intelligents

La croissance rapide de l'urbanisation et la demande croissante de transports efficaces ont favorisé le développement et l'évolution des systèmes de transport. Au fil des ans, les méthodes traditionnelles de contrôle du trafic ont eu du mal à faire face aux défis croissants posés par l'explosion démographique et l'augmentation des volumes de trafic. Cela a nécessité l'émergence de solutions intelligentes pour répondre aux complexités des réseaux de transport modernes.

Le Systèmes de transport intelligents désigne l'intégration de technologies avancées, de systèmes de communication et d'analyse de données dans la gestion des transports, ce qui renforce la sécurité des conducteurs et des piétons. Elle utilise des capteurs et des dispositifs intelligents pour collecter des données et les analyser en temps réel afin d'optimiser et de prendre de meilleures décisions [40]. Ce système comprend des systèmes de détection pour la collecte de données sur le trafic, tels que des boucles inductives et des capteurs vidéo, et des systèmes de communication pour connecter les éléments d'infrastructure via les réseaux de communication comme réseau cellulaire ou le système mondial de navigation par satellite. Les principaux systèmes de contrôle du trafic utilisent les données collectées pour optimiser la circulation et améliorer la sécurité. Des systèmes tels que la gestion du trafic et la gestion de la vitesse jouent un rôle essentiel à cet égard. Les véhicules intelligents constituent un autre élément utile. Ils sont équipés de capteurs, de systèmes de communication et de dispositifs de contrôle, leur permettant d'échanger des informations avec d'autres véhicules et infrastructures, facilitant ainsi la communication entre les véhicules et entre les véhicules et infrastructure [24, 25].

1.9 Inconvénients des feux tricolores

Les arrêts et redémarrages fréquents augmentent la consommation de carburant et les émissions de gaz à effet de serre, notamment le CO_2 et NO_x (oxydes d'azote).

Les cycles longs peuvent obliger les piétons ou cyclistes à attendre longtemps, ce qui les pousse parfois à traverser de manière imprudente.

Même lorsqu'il n'y a pas de circulation dans une direction, les véhicules doivent parfois s'arrêter à un feu rouge.

1.10 Conclusion

Dans ce chapitre, nous avons survolé de manière générale les feux tricolores.

Nous avons défini ces derniers, leur évolution et quelques systèmes de gestion. En- suite, nous avons présenté les outils théoriques de la gestion du trafic routier et les systèmes de transport intelligents.

Enfin, nous avons terminé par quelques inconvénients des feux tricolores.

Dans le prochain chapitre, nous allons decrire les differentes métaheuristiques proposer pour la gestion des feux de signalisations.

Chapitre 2

État de l'art sur les méta-heuristiques

2.1 Introduction

Les méthodes d'optimisation visent à trouver des solutions optimales ou quasi optimales aux problèmes et peuvent être divisées en méthodes exactes et méthodes (méta)heuristiques. les méta-heuristiques caractérisées par l'adaptabilité à divers types de problèmes de différents domaines. Dans ce chapitre, nous présenterons les concepts clés des méta-heuristiques, ces méthodes conçues pour traiter des problèmes d'optimisation complexes issus du monde réel.

2.2 Définition des méta-heuristiques

Les algorithmes méta-heuristiques sont des techniques d'optimisation conçues pour résoudre des problèmes complexes que les approches classiques ne peuvent pas traiter efficacement. Ils visent à trouver des solutions proches de l'optimum (ou idéalement optimales) en explorant un vaste espace de recherche. L'objectif est d'identifier le maximum ou le minimum global d'un problème. La plupart des méta-heuristiques s'inspirent de processus naturels tels que la génétique, la physique ou encore le comportement collectif des essaims [26].

2.3 Principe général des méta-heuristiques

Les méta-heuristiques sont des méthodes conçues pour trouver une solution optimale ou presque optimal dans l'espace de recherche. Les

méta-heuristiques sont des méthodes itératives et stochastiques qui progressent vers un optimum global ou presque global (meilleure solution possible) avec l'aide d'une seule solution ou d'une population de solutions dans l'espace de recherche. Un grand nombre de méta-heuristiques sont utilisées pour trouver des solutions optimales en utilisant différentes techniques telles que la recherche locale et la recherche globale.

Les méta-heuristiques peuvent s'adapter à divers types de problèmes tels que les problèmes combinatoires et continus, multi-objectifs, qui tentent d'optimiser des objectifs conflictuels.

Ces méthodes fonctionnent en explorant l'espace de recherche efficacement, en équilibrant l'exploration et l'exploitation. L'exploration consiste à rechercher dans de larges zones, ce qui enlève la chance de trouver de bonnes solutions dans des régions prometteuses de l'espace de recherche. L'exploration maintient une diversité de solutions pour ne pas être coincé à une région spécifique appelée optimum local.

L'exploitation est focalisée sur les régions locales pour raffiner les solutions, en explorant des solutions de voisinage et en raffinant les solutions graduellement.

Les méthodes qui se concentrent uniquement sur l'exploitation peuvent être bloquées à des optima locaux, et les méthodes qui se concentrent uniquement sur l'exploration peuvent ne pas donner de solutions précises. Ainsi, de nombreuses méthodes apparaissent pour équilibrer l'exploration et l'exploitation, telles que le recuit simulé. Et une approche hybride, qui est l'intégration de plusieurs algorithmes dans une seule approche pour équilibrer l'exploration et l'exploitation, comme l'optimisation par essaim de particules pour la recherche globale et l'escalade de col pour la recherche locale.

Les solutions générées sont évaluées à l'aide d'une fonction de fitness (ou fonction objective), qui est à quel point la solution s'adapte au problème. Il faut donc essayer d'améliorer les solutions au fil des itérations avec de nombreuses techniques comme les solutions évolutives et basées sur les essaims pour générer de meilleures solutions.

2.4 Classifications des Algorithmes méta-heuristiques

2.4.1 Algorithmes basés sur l'évolution

Les algorithmes évolutionnaires sont des techniques de recherche stochastiques inspirées par la biologie qui fonctionnent sur une population initiale de solutions potentielles. Les solutions sont classées par leur aptitude. À chaque génération, les individus ayant la meilleure aptitude sont choisis pour les combiner et partager leurs meilleures ca-

ractéristiques, ce qui conduit à de meilleures solutions qui s'adaptent mieux au problème [27].

Le processus est le suivant :

- la population initiale est générée aléatoirement. Celles-ci sont évaluées par leur aptitude ;
- tant que les critères d'arrêt ne sont pas atteints :
  - les meilleures solutions adaptées sont sélectionnées pour la reproduction ;
  - la meilleure solution sélectionnée passe à travers le processus de croisement et de mutation ;
  - de nouveaux descendants sont générés, remplaçant les parents dans la population, et évalués par leur aptitude et classés jusqu'à ce que les critères d'arrêt soient atteints.

2.4.2 Algorithme basé sur les essaims

Le comportement collectif des systèmes décentralisés et auto-organisés est connu sous le nom d'intelligence collective. Un système typique d'intelligence collective est composé d'une population d'agents simples qui peuvent interagir les uns avec les autres localement en agissant sur leur environnement local. Un agent individuel seul peut ne pas exhiber un comportement complexe, mais grâce à la collaboration entre les agents, lorsque plusieurs agents coopèrent en exploitant l'intelligence et les capacités de chacun, ils peuvent accomplir des comportements et des tâches complexes [28], tels que PSO (Optimisation par essaim de particules) et ACO (Optimisation par colonie de fourmis).

2.4.3 Algorithmes basés sur la physique

Les algorithmes basés sur la physique sont développés pour simuler certaines lois de la physique et de la chimie, ainsi que des phénomènes naturels (à l'exclusion des phénomènes biologiques) [29].

Parmi ces algorithmes, on trouve l'Algorithme de Recherche Gravitationnelle (GSA), proposé par Rashedi et al.[30], qui est inspiré par la loi de la gravité de Newton et le concept de masses s'attirant les unes les autres. On peut également citer le Recuit Simulé (SA), qui s'inspire du processus de recuit utilisé en métallurgie.

2.4.4 Algorithmes basés sur l'humain

Les algorithmes basés sur l'humain sont créés par l'intelligence des humains et également inspirés par la communication mutuelle et les interactions des humains dans la vie sociale et individuelle [31]. Tels que la recherche Tabou et l'optimisation basée sur l'apprentissage (TLBO),

qui est inspirée par l'enseignement et l'apprentissage entre le professeur et l'élève en classe.

2.5 Avantages des méta-heuristiques

— Le principal avantage des algorithmes d'optimisation méta-heuristique est leur capacité à fonctionner sur divers problèmes sans être adaptés à un domaine spécifique [1].
— Elles fournissent rapidement des solutions de bonne qualité,grâce à des mécanismes d'exploration et d'exploitation itératifs [32].

2.6 Désavantages des méta-heuristiques [1]

— L'ajustement des paramètres des algorithmes méta-heuristiques peut être difficile pour équilibrer l'exploration et l'exploitation.
— Les algorithmes méta-heuristiques peuvent générer différentes solutions pour le même problème lorsqu'ils sont exécutés plusieurs fois en raison du comportement aléatoire inhérent à certaines étapes de l'algorithme.

2.7 Présentation des principales méta-heuristiques utilisées en optimisation

Les méta-heuristiques sont des techniques d'optimisation puissantes et largement utilisées pour résoudre une vaste gamme de problèmes. Voici une présentation des principales méta-heuristiques couramment utilisées en optimisation :

2.7.1 Méta-heuristiques à solution unique

2.7.1.1 Recherche tabou

La recherche tabou est un algorithme d'optimisation méta-heuristiques fondé par Fred W. Glover en 1986 [33]. C'est un algorithme de recherche locale qui vise à échapper aux optima locaux en maintenant une liste tabou de mouvements interdits. Il utilise une mémoire à court terme des solutions précédemment visitées. La recherche tabou explore l'espace de solutions en passant de la solution actuelle à sa meilleure solution voisine et accepte des solutions moins bonnes afin d'éviter de revenir à des solutions précédemment visitées. La liste tabou est utilisée pour interdire certains mouvements précédemment effectués en les

stockant dans une liste et en les interdisant pendant un certain nombre d'itérations. Cette liste est appelée durée tabou.

Algorithme :

Algorithm 1 Pseudo-code de recherche taboue [2]

```
S ← initial_solution
best_solution ← S
tabu_list ← ∅
max_iter ← 100
max_x ← 50
for iter ← 1 to max_iter do
    best_neighbor ← None
    best_value ← −∞
    for x ← 1 to max_x do
        S_x ← generate_neighbor(S)
        if S_x ∉ tabu_list and f(S_x) > best_value then
            best_value ← f(S_x)
            best_neighbor ← S_x
        end if
    end for
    if best_neighbor ≠ None then
        tabu_list ← tabu_list ∪ { best_neighbor }
        S ← best_neighbor
        if f(S) > f(best_solution) then
            best_solution ← S
        end if
    end if
end for
return best_solution
```

Explication :

1. Initialiser :
   — Générer une solution initiale s
   — Définir la meilleure solution $s_best = s$
   — Initialiser la liste tabou pour qu'elle soit vide
   — Définir la durée du tabu t
   — Définir le nombre maximum d'itérations max_iter
   — Définir le compteur d'itération actuel $iter = 0$
2. Tant que $iter < max_iter$:
   — Générer un ensemble de solutions candidates $N(s)$ dans le voisinage de s

- Sélectionner la meilleure solution candidate s' parmi $N(s)$ qui n'est pas dans la liste tabou ou qui satisfait le critère d'aspiration
- Mettre à jour la meilleure solution : si $f(s') < f(s_best)$, alors $s_best = s'$
- Mettre à jour la solution actuelle : $s = s'$
- Mettre à jour la liste tabou :
  - Ajouter le mouvement qui a généré s' à la liste tabou
  - Si la taille de la liste tabou dépasse la durée tabou t, retirer le mouvement le plus ancien de la liste
- Incrémenter le compteur d'itérations : $iter = iter + 1$

Avantages [34] :

- La recherche tabou peut se libérer des optima locaux en permettant des transitions vers des solutions sous-optimales et en évitant de revenir à des solutions qui ont déjà été explorées.
- La liste tabou agit comme un outil de mémoire temporaire qui aide l'algorithme à explorer de nouvelles zones de l'espace de solution.

Inconvénients [34] :

- L'efficacité de la recherche tabou dépend de la configuration appropriée de la durée tabou ainsi que d'autres paramètres.
- La recherche tabou peut rencontrer des difficultés lorsqu'elle traite des problèmes ayant des contraintes strictes ou ceux avec de nombreux optima locaux.

2.7.1.2 Recuit simulé

Le recuit simulé, proposé par Kirkpatrick *et al.* en 1983 [35], s'inspire du processus de recuit physique pour explorer de manière probabiliste un espace de recherche. À partir d'une solution initiale S et d'une température de départ T, l'algorithme génère à chaque itération une solution voisine S' par une légère perturbation de S, puis calcule la différence de coût $\Delta E = f(S') - f(S)$. Si $\Delta E \leq 0$, la nouvelle solution est toujours acceptée ; sinon, elle est retenue avec la probabilité

$$P = \exp\left(-\frac{\Delta E}{T}\right),$$

ce qui permet d'accepter parfois des solutions moins bonnes afin d'éviter les minima locaux. Après chaque évaluation, la température est réduite selon un schéma de refroidissement (par exemple géométrique : $T_{k+1} = \alpha\, T_k$ avec $0 < \alpha < 1$), garantissant un passage progressif de l'exploration vers l'exploitation. Le processus se répète jusqu'à atteindre

un nombre maximal d'itérations, une température minimale ou la stagnation de la meilleure solution.

Algorithme :

Algorithm 2 Pseudo-code de recuit simulé [3]

Initialisation : Choisir une solution initiale x_0, température T, constante de Boltzmann k, facteur c, nombre d'itérations internes N_{iter}
while condition d'arrêt non satisfaite **do**
  for $i \leftarrow 1$ **to** N_{iter} **do**
    Générer $x_{\text{new}} \leftarrow x_0 + \Delta x$
    if $f(x_{\text{new}}) < f(x_0)$ **then**
      $x_0 \leftarrow x_{\text{new}}, \quad f \leftarrow f(x_{\text{new}})$
    else
      $\Delta f \leftarrow f(x_{\text{new}}) - f(x_0)$
      Générer $r \in (0, 1)$
      if $r > \exp(-\Delta f/(kT))$ **then**
        $x_0 \leftarrow x_{\text{new}}, \quad f \leftarrow f(x_{\text{new}})$
      end if
    end if
  end for
  $T \leftarrow c \times T$
end while
return x_0

Explication :

1. **Initialisation :**
   (a) Définir la température initiale et le programme de refroidissement.
   (b) Générer une solution initiale.
2. **Tant que les critères d'arrêt ne sont pas atteints :**
   (a) Générer une nouvelle solution en perturbant la solution actuelle.
   (b) Calculer la différence de la fonction objective entre la solution actuelle et la nouvelle solution.
   (c) Si la nouvelle solution est meilleure (valeur de la fonction objective plus basse) :
      i. Accepter la nouvelle solution comme la solution actuelle.
   (d) Sinon :
      i. Calculer la probabilité d'acceptation en fonction de la température actuelle et la différence de la fonction objective.
      ii. Générer un nombre aléatoire entre 0 et 1.

iii. Si le nombre aléatoire est inférieur à la probabilité d'acceptation :

A. Accepter la nouvelle solution comme la solution actuelle.

(e) Mettre à jour la température selon le calendrier de refroidissement.

3. Retourner la meilleure solution trouvée.

Avantages [36] :

— La capacité de naviguer avec succès dans l'espace de solution et de sortir des optima locaux.
— Convient à une variété de problèmes d'optimisation, tels que l'optimisation continue et combinatoire.

Inconvénients [36] :

— La fonction de perturbation et le choix du calendrier de refroidissement ont un impact significatif sur la performance.
— La meilleure solution pourrait être perdue en raison de la probabilité d'acceptation des pires solutions.

2.7.1.3 Recherche à Voisinage Variable (VNS)

VNS, proposé par Mladenović & Hansen en 1997 [37], a pour idée principale de changer systématiquement la structure de voisinage durant le processus de recherche. Cela permet à l'algorithme d'échapper aux optima locaux et d'explorer différentes régions de l'espace de solutions. Le processus de recherche alterne entre deux phases principales : la phase de secouage et la phase de recherche locale. Dans la phase de secouage, l'algorithme génère une nouvelle solution en perturbant aléatoirement la solution actuelle. La perturbation est effectuée à l'aide d'un ensemble de structures de voisinage prédéfinies, généralement ordonnées par taille ou complexité croissante. La phase de secouage aide l'algorithme à échapper aux optima locaux et à explorer de nouvelles régions de l'espace de solutions. Après la phase de secouage, une recherche de voisinage est effectuée à partir de la solution perturbée. La recherche locale vise à trouver une meilleure solution dans le voisinage actuel. Si une solution améliorée est trouvée, la recherche continue à partir de cette nouvelle solution ; sinon, l'algorithme passe à la prochaine structure de voisinage dans l'ensemble prédéfini. Le processus de recherche itère entre les phases de secouage et de recherche locale jusqu'à ce qu'un critère d'arrêt soit atteint, tel qu'un nombre maximum d'itérations ou une limite de temps.

Algorithme :

Algorithm 3 Pseudo-code de Recherche à Voisinage Variable (VNS) [4]

Initialisation :
  $S \leftarrow$ generate_initial_solution()
  best_solution $\leftarrow S$
  $l_{\max} \leftarrow 5$ ▷ Nombre maximal de structures de voisinage
  max_iterations $\leftarrow 100$ ▷ Condition d'arrêt
  iteration $\leftarrow 0$
repeat
  $l \leftarrow 1$
  while $l \leq l_{\max}$ **do**
    $S' \leftarrow$ Shaking(S, N_l) ▷ Perturbation de la solution avec le voisinage N_l
    $S^{**} \leftarrow$ Local_Search(S') ▷ Affinage de la solution
    if $f(S^{**}) < f(S)$ **then** ▷ Si la solution améliorée est meilleure
      $S \leftarrow S^{**}$
      $l \leftarrow 1$ ▷ Réinitialiser le voisinage
    else
      $l \leftarrow l + 1$ ▷ Passer au voisinage suivant
    end if
    if $f(S) < f$(best_solution) **then** ▷ Mettre à jour la meilleure solution
      best_solution $\leftarrow S$
    end if
  end while
  iteration $\leftarrow$ iteration $+ 1$
until iteration $\geq$ max_iterations
return best_solution

Explication :

1. **Initialiser l'algorithme :**
   (a) Définir l'ensemble des structures de voisinage $N_k, k = 1, \ldots, k_{\max}$, ordonné par taille ou complexité croissante.
   (b) Sélectionner une solution initiale x.
   (c) Définir la meilleure solution actuelle $x_{\text{best}} = x$.
2. **Répéter jusqu'à ce qu'un critère d'arrêt soit atteint :**
   (a) Définir $k = 1$.
   (b) Répéter jusqu'à ce que $k = k_{\max}$:

i. **Phase de secouage :** Générer une nouvelle solution x' en perturbant aléatoirement x en utilisant la structure de voisinage N_k.
ii. **Phase de recherche locale :** Appliquer une méthode de recherche locale à partir de x' pour obtenir une nouvelle solution x''.
iii. Si x'' est meilleur que x_{best}, mettre à jour $x_{\text{best}} = x''$ et définir $x = x''$.
iv. Si x'' est meilleur que x, définir $x = x''$ et $k = 1$; sinon, définir $k = k + 1$.

3. Retourner la meilleure solution trouvée x_{best}.

Avantages [38] :
— En modifiant méthodiquement la structure de voisinage tout au long de la recherche, il est possible d'éviter les optima locaux.
— Flexibilité dans la spécification de l'ensemble de la structure de voisinage, ce qui permet de modifier l'approche pour l'utiliser dans d'autres domaines problématiques.

Inconvénients [38] :
— La performance est affectée par le choix des structures de voisinage et l'ordre dans lequel elles sont appliquées.
— Déterminer des structures de voisinage efficaces pourrait nécessiter des connaissances pertinentes au problème.

2.7.2 Méta-heuristiques à population de solutions

Optimisation par essaims de particules

La méthode d'optimisation approximative *Particle Swarm Optimization* (PSO) est basée sur des « interactions sociales » entre des « agents » appelés *particules*, dans le but d'atteindre un objectif donné dans un espace de recherche commun. Chaque particule possède une certaine capacité à mémoriser et traiter des informations. Cette optimisation stochastique méta-heuristique a été proposée en 1995 par le psychologue social James Kennedy et l'ingénieur électricien Russell Eberhart [39].

Algorithme :

Algorithm 4 Pseudo-code de l'Optimisation par essaims de particules [5]

Paramètres :
  $swarm_size \leftarrow 30$ ▷ Nombre de particules
  $max_iter \leftarrow 100$ ▷ Nombre maximal d'itérations
  $w \leftarrow 0.7$ ▷ Poids d'inertie
  $c_1 \leftarrow 1.5,\ c_2 \leftarrow 1.5$ ▷ Facteurs d'apprentissage

Initialisation :
for $i \leftarrow 1$ **to** $swarm_size$ **do**
  $x[i] \leftarrow random_position()$ ▷ Position initiale aléatoire
  $v[i] \leftarrow random_velocity()$ ▷ Vélocité initiale aléatoire
  $pbest[i] \leftarrow x[i]$ ▷ Meilleure position individuelle initiale
  $pbest_fit[i] \leftarrow fitness(x[i])$ ▷ Fitness de la meilleure position individuelle
end for

Identifier la meilleure globale :
$gbest \leftarrow \arg\min_i \{\, pbest_fit[i] \,\}$ ▷ Indice de la particule la plus performante
$gbest_fit \leftarrow fitness\big(x[gbest]\big)$

Boucle d'optimisation :
for $iter \leftarrow 1$ **to** max_iter **do**
  for $i \leftarrow 1$ **to** $swarm_size$ **do**
    $r_1 \leftarrow random(0,1)$ ▷ Nombre aléatoire uniforme dans $(0,1)$
    $r_2 \leftarrow random(0,1)$
    $v[i] \leftarrow w \cdot v[i] \ + \ c_1 \cdot r_1 \left(pbest[i] - x[i]\right) \ + \ c_2 \cdot r_2 \left(gbest - x[i]\right)$ ▷ Mise à jour de la vitesse
    $x[i] \leftarrow x[i] + v[i]$ ▷ Mise à jour de la position
    $current_fit \leftarrow fitness\big(x[i]\big)$ ▷ Évaluer la nouvelle position
    if $current_fit < pbest_fit[i]$ **then** ▷ Mise à jour du meilleur individuel
      $pbest[i] \leftarrow x[i]$
      $pbest_fit[i] \leftarrow current_fit$
    end if
    if $current_fit < gbest_fit$ **then** ▷ Mise à jour du meilleur global
      $gbest \leftarrow i$
      $gbest_fit \leftarrow current_fit$
    end if
  end for
end for

return $x[gbest]$ ▷ Meilleure solution trouvée

Explication : L'algorithme PSO suit les étapes suivantes pour rechercher la solution optimale dans un espace de recherche :

1. **Initialisation** : Initialiser les paramètres de l'algorithme, ainsi que la position X_i et la vitesse V_i de chaque particule i de manière aléatoire dans l'ensemble des solutions possibles. Définir également la meilleure position personnelle P_b de chaque particule.
2. **Évaluation** : calculer, pour chaque particule i, la fitness $f\big(\mathbf{x}_i(t)\big)$.
3. **Mise à jour des meilleures solutions** *Pbest* **et** *Gbest* : pour chaque particule i :
   — Si $f(X_i^t) < f(P_b)$, alors mettre à jour $P_b \leftarrow X_i^t$;
   — Si $f(P_b) < f(P_g)$, alors mettre à jour $P_g \leftarrow P_b$.
4. **Mise à jour de la vitesse et de la position** : pour chaque particule i à l'itération t, effectuer :

$$V_i^{t+1} = w \cdot V_i^t + C_1 \cdot r_1 \cdot (P_i - X_i^t) + C_2 \cdot r_2 \cdot (g - X_i^t) \quad (2.1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2.2)$$

   où :
   — w est le facteur d'inertie,
   — C_1 et C_2 sont les coefficients d'accélération (cognitif et social),
   — r_1 et r_2 sont des nombres aléatoires entre $[0, 1]$,
   — $\mathbf{x}_i^t$ est la position, $\mathbf{v}_i^t$ la vitesse de particule i dans l'itération t.
   — $\mathbf{p}_i$ la meilleure position personnelle, $\mathbf{g}$ la meilleure position globale.
5. **Itération** : Répéter les étapes 2 à 4 jusqu'à atteindre un critère d'arrêt.

Avantages [40]

— PSO est moins sensible à la nature de la fonction objective que les approches mathématiques conventionnelles ou d'autres méthodes heuristiques.
— PSO comporte un nombre limité de paramètres (le facteur de poids d'inertie et deux coefficients d'accélération), ce qui le rend plus simple à paramétrer. De plus, l'impact de ces paramètres est considéré comme moins critique que dans d'autres algorithmes heuristiques.

Inconvénients [41]

— Le choix du poids d'inertie et des coefficients d'accélération idéaux peut être difficile, bien que PSO ait peu de paramètres.
— PSO peut souffrir de convergence prématurée et de piégeage dans des minima locaux, en particulier pour des problèmes de haute dimension.

Algorithme génétique

En 1975, John Holland a proposé l'algorithme génétique (GA), une technique d'optimisation basée sur une population et inspirée des principes de la sélection naturelle. Chaque membre de la population est représenté comme un chromosome, qui encode des solutions possibles à l'aide d'une série de gènes. La population initiale de l'algorithme est générée aléatoirement. Une fonction d'évaluation (fitness) est utilisée pour évaluer les individus de la population. Les meilleurs parents sont ensuite sélectionnés à travers un processus de sélection. Pour créer de nouveaux individus (enfants), le croisement est utilisé afin de recombiner le matériel génétique. De petits changements aléatoires sont introduits par mutation pour préserver la diversité. La population parentale est finalement remplacée par une nouvelle génération, et ce cycle se poursuit jusqu'à ce qu'une condition d'arrêt soit satisfaite [42].

Algorithme :

Algorithm 5 Pseudo-code d'algorithme génétique [6]

Initialisation : Générer aléatoirement la population initiale et évaluer la fitness de chaque individu
while condition d'arrêt non satisfaite **do**
    a. Sélectionner des parents selon leur fitness (roulette, tournoi, etc.)
    b. Appliquer le crossover aux parents pour produire des descendants
    c. Muter les descendants pour maintenir la diversité
    d. Combiner descendants et population actuelle pour former une population étendue
    e. Évaluer la fitness de tous les individus de la population étendue
    f. Sélectionner les meilleurs individus pour constituer la nouvelle génération
end while
return le meilleur individu (solution) trouvé

Explication :

L'algorithme génétique fonctionne généralement selon les étapes suivantes :

1. **Initialisation** : Générer une population initiale de solutions candidates (chromosomes) aléatoires.
2. **Évaluation** : Calculer la valeur de la fonction de fitness pour chaque individu de la population.
3. **Sélection** : Choisir les meilleurs individus selon leur aptitude pour être parents.
4. **Croisement (Crossover)** : Recombiner les parents pour générer de nouveaux enfants (solutions). Cette opération combine les caractéristiques génétiques de deux parents pour produire une ou plusieurs nouvelles solutions.Le but est d'exploiter les bonnes caractéristiques des parents pour créer une descendance potentiellement meilleure.
5. **Mutation** : Appliquer des changements aléatoires mineurs à certains gènes pour maintenir la diversité. Typiquement, un ou plusieurs bits ou valeurs d'un chromosome sont modifiés, afin d'introduire une variabilité supplémentaire dans la population.
6. **Remplacement** : Après les étapes de croisement et de mutation, une nouvelle population est formée à partir des enfants générés. Cette nouvelle génération peut soit remplacer entièrement la population précédente (remplacement total), soit être combinée avec les meilleurs individus de l'ancienne génération (remplacement partiel ou élitisme), afin d'assurer une certaine stabilité et d'éviter la perte de bonnes solutions. Ce processus permet de faire évoluer la population vers de meilleures solutions au fil des générations.
7. **Vérification de la condition d'arrêt** : Répéter les étapes précédentes jusqu'à ce qu'une condition (nombre d'itérations, convergence, etc.) soit satisfaite.

Avantages [43] :

— L'algorithme génétique est simple à implémenter par rapport à d'autres méta-heuristiques d'optimisation.
— Les algorithmes génétiques peuvent être appliqués à une large gamme de problèmes d'optimisation, car beaucoup de méta-heuristiques sont indépendantes du problème.

Inconvénients [43] :

— Les algorithmes génétiques sont sensibles au changement des paramètres tels que la taille de la population, le taux de croisement et le taux de mutation. Ces paramètres peuvent influencer la performance de l'algorithme et entraîner une convergence lente.
— Le coût computationnel des algorithmes génétiques peut être élevé, en particulier pour les problèmes de grande taille et avec un grand nombre de populations.

Grey Wolf Optimizer (GWO) :

Le Grey Wolf Optimizer est un algorithme proposé par Mirjalili et al. en 2014 [44]. Il s'inspire de la hiérarchie sociale des loups gris et imite leur comportement de chasse en groupe.

La stratégie de chasse repose sur une hiérarchie de loups : alpha, bêta, delta et oméga. Chacun joue un rôle spécifique. Le loup alpha est le chef de meute et prend les décisions.

Les loups bêta et delta assistent l'alpha dans la prise de décision et la chasse. Les derniers dans la hiérarchie sont les loups oméga, qui suivent les loups de rang supérieur (alpha, bêta, delta).

Le processus du GWO comprend trois phases principales : la recherche, l'encerclement et l'attaque de la proie. Lors de la phase de recherche, les loups explorent l'espace de recherche pour localiser une proie. Une fois celle-ci détectée, ils l'encerclent et l'attaquent.

Le comportement d'encerclement est modélisé par les équations suivantes :

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{2.3}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2.4}$$

où t représente le nombre d'itérations, $\vec{A}$ et $\vec{C}$ sont des vecteurs de coefficients, $\vec{X}_p(t)$ est la position de la proie, $\vec{D}$ est la distance entre le loup et la proie, et $\vec{X}(t+1)$ est la position du loup. Les vecteurs $\vec{A}$ et $\vec{C}$ sont définis comme suit :

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r}_1 - \vec{a} \tag{2.5}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{2.6}$$

où $\vec{r}_1$ et $\vec{r}_2$ sont des vecteurs aléatoires dans l'intervalle $[0, 1]$, et $\vec{a}$ est un vecteur décroissant linéairement de 2 à 0 au fil des itérations.

Pendant la chasse, les loups alpha (α), bêta (β) et delta (δ) représentent les meilleures solutions trouvées jusqu'à présent. Les autres loups (oméga) mettent à jour leur position en se basant sur ces trois loups, ce qui leur permet de converger vers une solution optimale. Ce comportement est modélisé par les équations suivantes :

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \tag{2.7}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \tag{2.8}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{2.9}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2.10}$$

Algorithme :

Algorithm 6 Pseudo-code de l'Optimiseur de loup gris [7]

Input : population size N (number of wolves), maximum number of iterations T
Initialize a population of wolves $\{ X_i \}_{i=1}^{N}$ **randomly**
Evaluate the fitness of each wolf X_i
Identify the three best wolves as :
X_α (best solution), X_β (second best), X_δ (third best)
Set iteration counter $t \leftarrow 0$
while $t < T$ **do**
  for $i = 1$ **to** N **do**
    Generate two random numbers $r_1, r_2 \in [0, 1]$
    Compute coefficient vectors :

$$A \leftarrow 2\,a\,r_1 - a, \qquad C \leftarrow 2\,r_2$$

    Calculate the distances between wolf X_i and each of the three best wolves :

$$D_\alpha = \left| C\,X_\alpha - X_i \right|, \quad D_\beta = \left| C\,X_\beta - X_i \right|, \quad D_\delta = \left| C\,X_\delta - X_i \right|$$

    Update the wolf's position with respect to each leader :

$$X_{i,\alpha} \leftarrow X_\alpha - A\,D_\alpha, \quad X_{i,\beta} \leftarrow X_\beta - A\,D_\beta, \quad X_{i,\delta} \leftarrow X_\delta - A\,D_\delta$$

    Compute the new position of X_i as the average of these three :

$$X_i \leftarrow \frac{X_{i,\alpha} + X_{i,\beta} + X_{i,\delta}}{3}$$

  end for
  Evaluate the fitness of all wolves at their new positions
  Update $X_\alpha, X_\beta, X_\delta$ if any wolf's new position is better
  Increment iteration counter : $t \leftarrow t + 1$
  Update parameter a (linearly decreased from 2 to 0) :

$$a \leftarrow 2\left(1 - \frac{t}{T}\right)$$

end while
return X_α ▷ Position of the Alpha wolf (best solution)

Explication :

Voici les étapes principales de l'algorithme GWO :

1. **Initialisation** : Générer aléatoirement une population de loups (solutions candidates) dans l'espace de recherche.

2. **Évaluation** : Calculer la valeur de la fonction objectif pour chaque loup. Identifier les trois meilleures solutions comme α, β, et δ.
3. **Mise à jour des coefficients** : Calculer les vecteurs $\vec{A}$ et $\vec{C}$ pour chaque loup à l'aide des équations (2.5) et (2.6).
4. **Encerclement de la proie** : Chaque loup met à jour sa position en fonction de la position de la proie en suivant les équations (2.3) et (2.4).
5. **Chasse** : Chaque loup actualise sa position en fonction des trois leaders (α, β, δ) selon les équations (2.7), (2.8), (2.9) et la moyenne (2.10).
6. **Boucle** : Répéter les étapes précédentes jusqu'à atteindre le nombre maximal d'itérations ou un critère d'arrêt défini.

Avantages : [45]

— Facile à mettre en œuvre.
— Nécessite peu de paramètres de réglage comparé à d'autres algorithmes.

Inconvénients : [45]

— Convergence relativement lente.
— Risque de se retrouver bloqué dans des optima locaux.

2.8 Approches hybrides dans les méta-heuristiques

Avec diverses méta-heuristiques existantes, chacun de ces algorithmes possède des points forts, tels qu'une convergence rapide, un mécanisme de recherche locale robuste ou encore un mécanisme garantissant une grande diversité. Cependant, ils présentent également des points faibles qui peuvent rendre l'optimisation moins précise. C'est pourquoi l'utilisation d'approches hybrides est apparue comme une solution possible pour exploiter les meilleures fonctionnalités de ces algorithmes et les intégrer ensemble dans le processus d'optimisation.
De manière générale, les algorithmes hybrides se divisent en deux catégories, détaillées dans les sous-sections suivantes [8].

2.8.1 Hybrides Collaboratifs

Cette approche implique la coopération entre deux ou plusieurs algorithmes, exécutés soit de façon parallèle, soit de manière séquentielle ou multi-étape [8].

(i) **Multi-étape** : Cette approche comporte deux phases : la première utilise un algorithme jouant le rôle d'optimiseur global, tandis que la seconde applique un algorithme dédié à la recherche locale.
(ii) **Séquentiel** : Les deux algorithmes sont exécutés de manière alternée, chacun pendant un nombre d'itérations comparable, avant de céder la place à l'autre.
(iii) **Parallèle** : Deux algorithmes sont exécutés simultanément sur la même population. L'un des algorithmes peut être appliqué à un pourcentage prédéfini de la population.

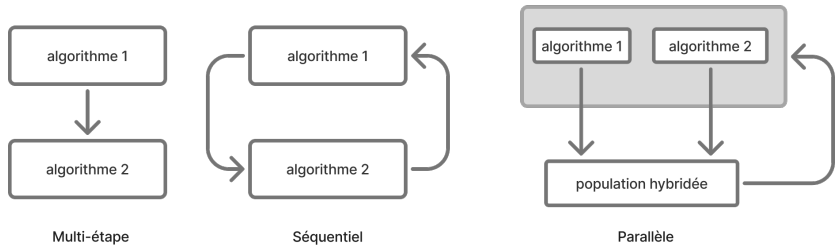

FIGURE 2.1 – Hybrides Collaboratifs [8]

2.8.2 Hybrides Intégratifs

Dans une approche hybride de ce type, un algorithme est considéré comme subordonné et intégré au sein de l'algorithme méta-heuristique principal. Cela se réalise en utilisant des opérateurs de l'algorithme subordonné dans l'algorithme principal. Par exemple, on peut intégrer des opérateurs de croisement ou de mutation issus d'un algorithme génétique dans l'algorithme PSO [8].

(i) **Manipulation complète** : Toute la population est modifiée à chaque itération. Cette opération, généralement sous forme de sous-routine ou de sous-fonction, peut être directement intégrée dans le code source existant.
(ii) **Manipulation partielle** : Dans ce type de manipulation partielle, seule une portion de la population est accélérée à l'aide de techniques de recherche locale, telles que la recherche tabou. Le

choix des bons candidats et de la portion appropriée à améliorer constitue l'un des défis majeurs pour assurer l'efficacité de cette structure hybride.

2.9 Applications des méta-heuristiques à l'optimisation du timing des feux de signalisation dans la littérature

Les méta-heuristiques sont largement utilisées dans la littérature pour aborder les problèmes d'optimisation du timing des feux de signalisation. Cette application joue un rôle crucial dans l'amélioration de la fluidité du trafic, la réduction des temps d'attente et des émissions de gaz à effet de serre, ainsi que dans l'optimisation des réseaux de transport urbain. Voici quelques exemples d'applications des méta-heuristiques pour optimiser le timing des feux de signalisation :

les auteurs dans [46] ont proposé une stratégie de contrôle en temps réel des feux de circulation basée sur un algorithme génétique, afin de parvenir à un contrôle adaptatif des feux aux intersections. Ce système adaptatif évalue en continu s'il est nécessaire de prolonger les phases vertes en fonction des conditions de circulation. Les auteurs ont utilisé **MATLAB** pour implémenter et exécuter l'algorithme génétique, et ont également développé un émulateur en langage Java pour simuler un flux de trafic dynamique.

Après chaque intervalle de surveillance prédéfini, l'algorithme génétique est invoqué avec, en entrée, le nombre total de véhicules sur chaque voie. en commencent avec la fonction d'évaluation suivante (fitness function) :

$$f = P.I.1 + P.I.2 + P.I.3 + P.I.4 \tag{2.11}$$

où l'**indice de performance (P.I.)** pour chaque route dépend du poids attribué à chaque voie, ainsi que du nombre total de véhicules sur la route, noté $S = S_1 + S_2 + S_3 + S_4$.

$$P.I._{.i} = \frac{W_i \cdot S_i}{GT_i} \tag{2.12}$$

où W_i est le poids attribué à la route i, S_i est le nombre de véhicules sur la route i, et GT_i est la somme du **temps vert minimal (Gmin)** et du **temps de prolongation vert (g)**.

Pour la simulation, les auteurs utilisent un algorithme génétique afin d'**optimiser les temps de prolongation des feux verts**.

— L'émulateur commence par simuler les conditions de circulation dynamiques à une intersection à quatre voies.
— Après la première phase, le **nombre de voitures** (sur chaque route) est déterminé et envoyé à l'algorithme génétique, implémenté dans MATLAB, pour calculer les ajustements de temps vert qui conduisent à la valeur de fonction de fitness la plus basse possible.

Les **temps de prolongation verts (g)** suivent les contraintes suivantes :

— $g_i \in [0, 5]$ secondes pour $i = 1, 2, 3, 4$
— $g_1 + g_2 + g_3 + g_4 = 10$ secondes (temps total de prolongation)

Ensuite,les **étapes de l'algorithme génétique (GA)** exécutées : Étapes de l'algorithme :

1. Générer aléatoirement une population initiale de temps de prolongation verts.
2. Évaluer la population à l'aide de la fonction de fitness f mentionnée ci-dessus.
3. Vérifier si le critère d'arrêt est atteint (par exemple 6 ou 100 itérations) ; sinon, continuer à une nouvelle itération.
4. Effectuer la **sélection** de la solution la plus adaptée, puis le **croisement**, et enfin la **mutation**. Le nouveau temps vert calculé est ensuite envoyé à l'émulateur pour une nouvelle itération.

Ce processus est répété jusqu'à ce que les **critères d'arrêt soient satisfaits**.

Le système en temps réel a permis d'accroître de 21,9% le nombre de véhicules sortants, passant de 573 à 662 véhicules. Ce gain illustre l'efficacité de l'algorithme génétique adapté aux conditions réelles de trafic.

Silva et al. [47] proposent une approche de simulation-optimisation basée sur l'algorithme PSO et le simulateur SUMO pour régler les feux de circulation sur des routes à voie unique. Chaque particule encode une configuration des durées vertes, et la performance est évaluée via une simulation SUMO mesurant le temps d'attente moyen des véhicules. En pilotant les durées de vert par PSO, ils montrent qu'il est possible de réduire de manière significative (environ 26 %) le temps d'attente moyen par rapport à une configuration statique. Leur démarche se concentre sur la simplicité du modèle (deux phases de vert, une voie dans chaque sens) tout en démontrant l'intérêt d'une optimisation itérative en boucle fermée avec SUMO.

Par ailleurs, les autres approfondissent l'étude du réglage des paramètres de PSO (poids d'inertie, coefficients cognitifs et sociaux, facteur

de constriction) dans ce même contexte. Ils comparent plusieurs stratégies d'inertie (fixe faible, fixe élevée, décroissante linéaire) et intègrent un facteur de constriction pour stabiliser la vitesse des particules. Leurs expériences, menées sur différentes densités de trafic, démontrent que :

- une inertie décroissante permet un équilibre satisfaisant entre exploration et exploitation.
- l'introduction d'un facteur de constriction améliore la convergence vers des solutions de meilleure qualité.
- les coefficients cognitifs et sociaux doivent être ajustés en fonction du profil de trafic pour éviter la stagnation prématurée.

Ces travaux soulignent l'importance d'un calibrage fin de PSO pour garantir des gains de performance robustes dans l'optimisation des feux de circulation.

Abdulhameed et al. [48]. Cette étude explore l'application de l'algorithme de recuit simulé (Simulated Annealing, SA) afin d'optimiser la gestion du trafic dans une intersection complexe, comprenant à la fois un carrefour traditionnel et un carrefour en T. Les auteurs ont conçu un modèle de simulation sous MATLAB, doté d'une interface graphique utilisateur (GUI), permettant de tester différentes configurations de circulation.
Simulated Annealing a été utilisé pour optimiser les temps de signalisation, en minimisant une fonction objective définie comme :

$$f = 20(C_1)^4 + 16(C_2)^2$$

où C est le cycle.

Deux cas ont été étudiés, différenciés par la distance entre les intersections (≥ 500 m pour le cas 1 et ≤ 100 m pour le cas 2).

Cas 1 (distance ≥ 500 m) : Les temps de décongestion variaient entre 9 s et 41 s, avec une meilleure performance en termes de valeurs de fitness.

Cas 2 (distance ≤ 100 m) : Les temps de décongestion étaient plus longs, allant de 2 s à 58 s.

Odeh et al. [49] proposent un système combinant un *Fuzzy Logic Controller* (FLC) pour estimer dynamiquement les durées de phases vertes aux intersections, et un *Genetic Algorithm* (GA) qui optimise les règles floues et les fonctions d'appartenance. Le modèle prend en

compte quatre intersections interconnectées et s'appuie sur la détection de véhicules par traitement d'image.

Entrées

- C_{IM} : Nombre de véhicules détectés sur la voie M de l'intersection I (vidéo–détection).
- (Pour FLC2) : Durée verte estimée par FLC1 et nombre de véhicules sur la voie affectée.

Sortie

- E : Durée du vert estimée (en secondes), codée en trois niveaux linguistiques {G1, G2, G3} puis affinée par le GA.

Fonction de fitness Pour chaque chromosome (configuration de passages verts), on calcule le nombre total de véhicules *restant* après la phase verte :

$$C'_{IM} \;=\; \sum_{I=1}^{4} \Big| C_{IM} \;-\; E \times \tfrac{6}{10} \Big| \tag{2.13}$$

où l'on suppose que 6 véhicules traversent l'intersection tous les 10 s, et C'_{IM} est la somme des véhicules bloqués sur les quatre intersections, E est la Durée de la lumière verte (estimée par le contrôleur logique flou).L'objectif du GA est de **minimiser** C'_{IM}.

Les résultats montrent une amélioration notable de la performance du système hybride FLC-GA. En particulier, les auteurs rapportent jusqu'à **34 %** de réduction du temps moyen de passage par rapport à un contrôleur fixe (CTC), et jusqu'à **31 %** d'amélioration par rapport à un contrôleur flou non optimisé (FLC seul). Selon les données du tableau 2, la moyenne du nombre de véhicules restants après chaque cycle est de 52 pour FLCGA, contre 59 pour FLC et 60 pour CTC. De plus, le temps total de feu vert moyen est réduit à 32,4 s avec FLCGA, contre 44,8 s pour FLC et 60 s pour CTC, ce qui confirme l'efficacité de l'approche hybride dans la gestion intelligente du trafic.

Titre	Auteur(s)	Année	Approche	Avantages et inconvénients	Fonction de Fitness
Time Optimization for Traffic Signal Control Using Genetic Algorithm [46]	Singh et al.	2009	GA	- Adaptabilité en temps réel. - mais les temps d'extension verts sont limités à 5 secondes par phase (un total de 10 secondes pour toutes les phases), ce qui peut être insuffisant pour des scénarios de forte congestion.	Réduire le nombre de voitures à chaque intersection.
On Tuning the Particle Swarm Optimization for Solving the Traffic Light Problem [47]	Silva et al.	2022	PSO	- Gains significatifs via les tests paramétriques du PSO. - optimisation isolée négligeant la coordination des pelotons et du réseau de débordement, entraînant des inefficacités.	Temps d'attente moyen total des véhicules.
Traffic signal control based on particle swarm optimization [50]	Lo, K.-R & Tung, Shen-Lung	2014	PSO	Convergence plus rapide que GA. Hypothèse de schémas de circulation stables.	Temps total passé dans le flux de trafic pour voitures et scooters.
A Hybrid Algorithm based on GWO and GOA for Cycle Traffic Light Timing Optimization [51]	T.-C. Teng and M.-C. Chiang and C.-S. Yang	2019	GWO/GOA	- L'algorithme hybride exploite les forces combinées de GWO et GOA. - l'utilisation parallèle de GWO et GOA est coûteuse en termes de calcul.	Minimiser le temps d'attente total de tous les véhicules et Maximiser le nombre de véhicules atteignant leur destination dans un délai donné.
Hybridization of Racing Methods with Evolutionary Operators for Simulation Optimization of Traffic Lights Programs [52]	Cintrano et al.	2021	IRACE + DE/GA	Optimisation efficace des horaires. Coûteux en calculs en raison des simulations étendues.	Temps total de trajet, temps d'attente total, émissions de CO_2, consommation de carburant, nombre de véhicules non terminés.
Analysis of Simulated Annealing to Decongest Traffic in a Multi-Road Coordinated Intersection [48]	abdulhameed et al.	2023	Recuit Simulé	- SA évite les optimaux locaux. - convergence de SA est très lente.	Minimise le temps d'attente moyen, la longueur des files d'attente, maximise le débit.
Hybrid Fuzzy Genetic Algorithm for an Adaptive Traffic Signal System [49]	Odeh, M.; Azzam, M.; Al-Khateeb, R.	2015	Contrôleur logique floue + algorithme génétique	Minimisation du nombre de véhicules restant après le passage au vert.	Délai moyen (minimisé), véhicules restants (minimisés)
Intelligent Meta-Heuristic-Based Optimization of Traffic Light Timing Using Artificial Intelligence Techniques [53]	Khasawneh, M.A.; Awasthi, A.	2023	Hybride EBAT + IA	Réduction des retards, émissions et consommation. Complexité pour l'exécution en temps réel.	Délai moyen (minimisé), longueur de file (minimisée), émissions de CO_2 (réduites).

TABLE 2.1 – Comparaison des méthodes d'optimisation des feux de circulation

2.10 Conclusion

Nous avons présenté dans ce chapitre les méta-heuristiques, ces méthodes d'optimisation inspirées de la nature ou de phénomènes physiques, en définissant leurs principes et en analysant leurs avantages et leurs limites. Puis nous avons présenté des classifications distinguant les approches mono-solution des approches populationnelles. Nous avons décrit les principes et le fonctionnement de quelques exemples de méta-heuristiques, comme le recuit simulé, la recherche taboue, l'algorithme génétique, l'essaim de particules, la recherche à voisinage variable et l'Optimiseur du loup gris. Nous avons ensuite exploré les stratégies hybrides, combinant plusieurs méta-heuristiques pour bénéficier de leurs forces respectives, et conclu par des applications à l'optimisation des feux tricolores dans la littérature, présentant les méthodes retenues et leurs résultats.

Chapitre 3

Approche Proposée

3.1 Introduction

La congestion routière est un problème majeur dans les zones urbaines, entraînant une augmentation du temps de trajet, de la consommation de carburant et des émissions polluantes. L'optimisation des feux de circulation est une solution essentielle pour améliorer le flux du trafic et réduire ces impacts négatifs.

Dans ce chapitre, nous présenterons une approche méta-heuristique destinée à l'optimisation du flux de trafic. L'approche proposée repose sur une hybridation de l'optimisation par Essaim de Particules (PSO) et du Recuit Simulé (SA), combinant ainsi les avantages des deux algorithmes.

L'objectif de cette approche est d'explorer un large espace de solutions afin d'identifier une configuration optimale des feux de circulation, adaptée aux conditions dynamiques du trafic. Le PSO est utilisé pour une exploration efficace de l'espace de recherche, tandis que le SA permet un raffinement des solutions afin d'éviter les optima locaux et d'améliorer la convergence vers une solution optimale.

3.2 Description de la problématique

Le problème consiste à coordonner plusieurs feux de circulation au sein d'un réseau routier en déterminant les configurations optimales des durées de feu vert pour chaque intersection, en fonction des conditions de trafic en temps réel. Plus précisément, il s'agit d'optimiser la répartition des temps de passage pour les quatre directions principales (Nord-Sud et Est-Ouest) afin d'améliorer la fluidité du trafic et de minimiser les congestions.

3.3 Architecture générale

L'architecture de l'approche proposée repose sur une méthodologie hybride combinant l'algorithme d'Optimisation par Essaim de Particules (PSO) et le Recuit Simulé (SA) avec le simulateur SUMO (Simulation of Urban MObility). Cette combinaison vise à optimiser la gestion des feux de circulation en exploitant les capacités d'exploration du PSO et le raffinement des solutions apporté par le SA, tout en intégrant une simulation réaliste du trafic urbain à l'aide de SUMO comme il est présenté dans la figure 3.1.

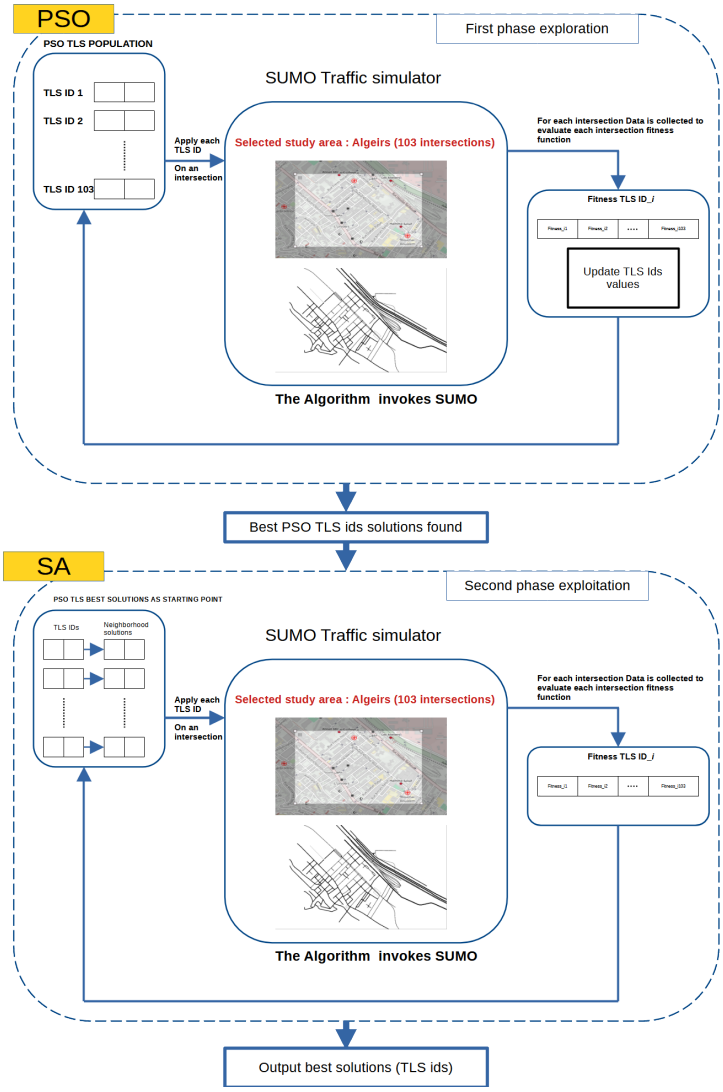

FIGURE 3.1 – Architecture générale

3.4 PSO

L'algorithme d'optimisation méta-heuristique basé sur la population Particle Swarm Optimization (PSO) a été présenté pour la première fois par James Kennedy et Russell Eberhart en 1995 et s'inspire du comportement social des poissons et des oiseaux. En mettant à jour les solutions potentielles (particules) en réponse à l'expérience utilisateur et à l'influence sociale, il optimise un problème de manière itérative [39].

3.4.1 Initialiser les paramètres

L'initialisation des paramètres est une étape cruciale dans l'optimisation par essaim de particules (PSO), car elle peut affecter les performances de l'algorithme. Les coefficients d'accélération c_1 et c_2 contrôlent l'influence de l'expérience personnelle et de l'expérience collective de l'essaim. avec $r1$ et $r2$ poids aléatoire qui sont des nombres aléatoires générés à chaque itération $\in [0, 1]$

c_1 est un composant cognitif qui détermine dans quelle mesure une particule tend à suivre sa meilleure solution personnelle trouvée (son expérience personnelle), plus c_1 est élevé, plus la particule se fie à sa meilleure position.

c_2 est le composant social qui détermine à quel point la particule est attirée (ou repoussée) par la meilleure solution globale de l'essaim , plus c_2 est élevé, plus les particules sont influencées par la meilleure particule globale.

Le poids d'inertie ω est un paramètre qui contrôle dans quelle mesure la particule conserve sa vitesse précédente. Une inertie élevée signifie que les particules ont tendance à maintenir leur vitesse actuelle, ce qui permet d'échapper aux optima locaux mais ralentit la convergence.

À l'inverse, une faible inertie signifie que la particule a tendance à modifier rapidement sa vitesse, ce qui entraîne une convergence rapide et exploite les régions prometteuses. Cependant, cela peut aussi conduire à une convergence prématurée et à une perte de diversité.

3.4.2 Initialiser les particules

Dans PSO, chaque particule représente une solution candidate caractérisée par deux attributs : position et vitesse. Pour modéliser la configuration des feux de circulation dans l'algorithme PSO, chaque particule correspond à une configuration de feux pour une intersection.

La position de la particule représente la variable de décision réelle, définie dans la figure 3.2.

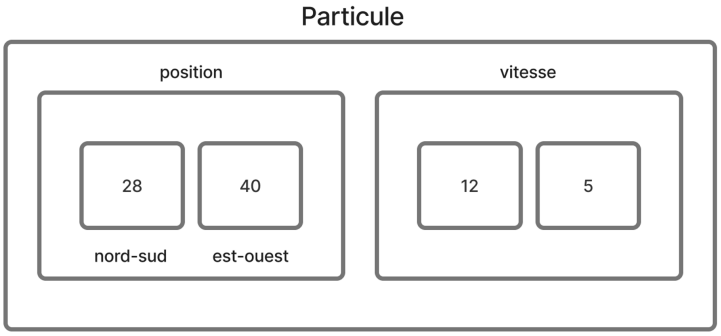

FIGURE 3.2 – représentation de particule

3.4.3 Mise à jour de *PBest*

Après avoir calculé la valeur de fitness de chaque particule en fonction de la fonction de fitness Eq.3.3, chaque particule conserve son propre meilleur personnel (*PBest*), qui représente la meilleure solution qu'elle a trouvée jusqu'à présent (c'est-à-dire la meilleur configuration des feux de circulation pour une intersection i).

À chaque itération, la nouvelle position de la particule est évaluée à l'aide de la fonction de fitness. Si cette nouvelle position a une meilleure valeur de fitness que le *PBest* actuel, alors *PBest* est mis à jour avec cette nouvelle position. Sinon, il reste inchangé.

3.4.4 Mise à jour de *GBest*

Après avoir calculé *PBest* pour chaque particule, il est nécessaire de déterminer *GBest* (Global Best), qui représente la meilleure solution trouvée parmi toutes les particules de l'essaim.

GBest correspond à la particule ayant la plus faible valeur de fitness dans le cas d'un problème de minimisation (ou la plus élevée pour un problème de maximisation). Cette particule sert de référence globale et influence l'évolution de toutes les autres particules en guidant leur mouvement vers une solution optimale.

3.4.5 Mettre à jour la vitesse et la position

Après avoir calculé $PBest$ et $GBest$, nous mettons à jour la vitesse et la position des particules en utilisant les formules suivantes :

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (p_i - x_i^{(t)}) + c_2 r_2 (g - x_i^{(t)}) \tag{3.1}$$

où :

- v_i^{t+1} : est la nouvelle vitesse de la particule i.
- w : est le facteur d'inertie.
- v_i^t : est la vitesse actuelle.
- c_1 et c_2 sont les coefficients d'accélération
- r_1, r_2 : sont des valeurs aléatoires uniformément distribuées dans l'intervalle $[0, 1]$.
- g : est la meilleure position trouvée par l'essaim.
- pi : est la meilleure position atteinte par la particule i.
- x_i^t : est la position actuelle de la particule.

Une fois la vitesse calculée, la position de la particule est mise à jour selon :

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3.2}$$

où :

- x_i^{t+1} : La nouvelle position de la particule i à l'itération $t + 1$.
- x_i^t : La position actuelle de la particule i à l'itération t.
- v_i^{t+1} : Nouvelle vitesse de la particule calculée précédemment avec (3.1).

La figure 3.3 illustre les différentes étapes d'optimisation par essaim de particules.

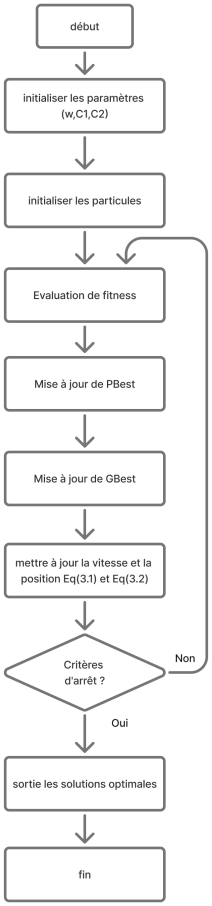

FIGURE 3.3 – organigramme d'optimisation par essaim de particules

3.5 Recuit simulé (SA)

Le recuit simulé est un algorithme d'optimisation basé sur une solution unique, introduit par Kirkpatrick, Gelatt Jr. et Vecchi en 1983. Il s'inspire du processus physique de recuit en métallurgie. Le recuit simulé reproduit ce processus : un matériau est chauffé puis refroidi progressivement pour réduire les défauts [35].

Le recuit simulé démarre et s'exécute par défaut sur une seule solution. Cependant, dans notre scénario multi-intersections, nous partons de plusieurs solutions candidates — les meilleures identifiées par l'algo-

rithme PSO — et appliquons le recuit simulé à chacune d'elles. Chaque solution est affinée par de petites perturbations pour améliorer davantage sa fonction de fitness.

3.5.1 Initialisation

La procédure commence avec les solutions identifiées par l'algorithme PSO. la définition des paramètres, en commençant par l'attribution d'une température initiale favorisant l'exploitation. Un programme de refroidissement est également établi afin de réguler le taux de diminution de la température au fil des itérations.

3.5.2 Perturbation

À chaque itération et pour chaque solution candidate, une nouvelle solution voisine S' est générée (nouveau plan de durée des feux de circulation). Ceci est réalisé en appliquant une petite perturbation à la solution courante. Cette perturbation consiste à apporter de légères modifications aléatoires aux valeurs de la variable de décision. Cela permet d'explorer davantage de solutions possibles dans le voisinage. La fitness de la nouvelle solution $f(S')$ est ensuite calculée à l'aide de la fonction de fitness Eq.3.3.

3.5.3 Critères d'acceptation

Dans le contexte du recuit simulé.Les critères d'acceptation déterminent si la nouvelle solution S' remplace la solution actuelle S. La nouvelle solution S' est acceptée si :

1. Si la nouvelle solution est meilleure, c'est-à-dire si $f(S') < f(S)$, alors mettez à jour la solution :

$$S \leftarrow S'$$

2. Si la nouvelle solution est pire, c'est-à-dire $f(S') >= f(S)$, acceptez-la avec une probabilité :

$$P = \exp\left(\frac{-\Delta f}{T}\right)$$

où $\Delta f = f(S') - f(S)$ est positif (puisque $f(S')$ est pire).
Δf est le Changement de fitness et $f(S')$ est le valeur de fitness de la nouvelle solution, $f(S)$ est le valeur de fitness de la solution actuelle et T est le Température actuelle.

Ensuite, un nombre aléatoire $r \in [0, 1]$ est généré. Si $r < P$, la pire solution est acceptée pour faciliter l'échappement des optima locaux.

3.5.4 Refroidissement

Le refroidissement est la réduction progressive de la température T pendant le processus de recuit simulé. À mesure que T diminue, la probabilité d'accepter les pires solutions diminue, ce qui conduit à un réglage plus précis autour de la région prometteuse dans les étapes ultérieures.

Une façon courante de mettre à jour la température consiste à utiliser un programme de refroidissement géométrique :

$$T_{k+1} = \alpha T_k, \quad \text{où } 0 < \alpha < 1$$

où :

— T_k : la température à l'itération k.
— α : la vitesse de refroidissement (un facteur constant).
— T_{k+1} : la température mise à jour pour l'itération suivante.

La figure 3.4 illustre les différentes étapes du processus de recuit simulé dans un organigramme.

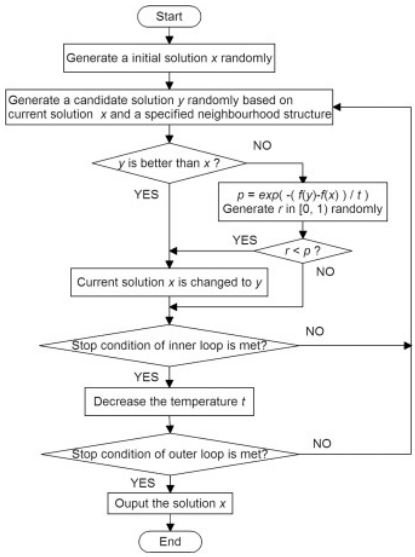

FIGURE 3.4 – Organigramme de recuit simulé [9]

3.6 Hybridation de PSO et SA

L'approche proposée combine l'Optimisation par Essaim de Particules (PSO) et le Recuit Simulé (SA) afin d'exploiter les avantages de chaque algorithme tout en compensant leurs limites respectives.

L'algorithme PSO est reconnu pour sa rapidité de convergence et son efficacité dans l'exploration de l'espace de recherche. Cependant, il présente un inconvénient majeur : il tend à se bloquer dans des optima locaux.
C'est dans ce contexte que le Recuit Simulé (SA) intervient. Initialement conçu comme un algorithme de recherche locale et globale, SA possède des propriétés de recherche locale dans le voisinage et de recherche globale grâce à son mécanisme d'acceptation probabiliste des solutions moins bonnes. En permettant d'échapper aux optima locaux, le Recuit Simulé améliore les solutions explorées et affine les résultats obtenus par PSO.

Ainsi, l'hybridation PSO-SA permet d'exploiter la capacité exploratoire de PSO pour parcourir efficacement l'espace de recherche et la capacité d'exploitation de SA pour raffiner les solutions et converger vers une meilleure optimisation des feux de circulation.

Dans notre approche, le processus d'optimisation se déroule en deux phases. Tout d'abord, nous utilisons l'optimisation par essaim de particules (PSO) pour effectuer une recherche globale. Un groupe de particules, chacune représentant une configuration de feux tricolores pour un intersection, des paramètres prédéfinis guident l'algorithme pour rechercher dans un large espace de solutions possible, l'algorithme évalue et ajuste ces configurations, découvrant ainsi différentes solutions potentielles.

Une fois la recherche globale avec PSO terminée, nous passons à la phase de recherche locale avec recuit simulé (SA). Bien que SA affine généralement une solution à la fois, nous l'adaptons pour travailler sur plusieurs solutions simultanément. Chaque solution est affinée jusqu'à atteindre une condition d'arrêt. Au final, notre approche fournit les meilleures configurations de feux tricolores pour les directions nord-sud et est-ouest, optimisant ainsi la fluidité du réseau routier.

3.7 Fonction Objectif

La fonction objective vise à minimiser le temps d'attente moyen des véhicules tout en maximisant le débit de circulation. Elle s'inspire de

[47]. La fonction objective s'écrit ainsi :

$$F(X_j) = \frac{1}{N} \sum_{i=1}^{N} W_i \tag{3.3}$$

contrainte :

$$N = 0 \Rightarrow F(X_j) = 0$$

où :

- X_j est la variable de décision.
- $F(X_j)$ représente la fonction objectif à minimiser (fonction de fitness).
- W_i est le temps d'attente du véhicule i à l'intersection j
- N est le nombre total de véhicules présents à l'intersection j pendant un cycle.

3.7.1 Variables de Décision

Pour modéliser les feux de circulation de plusieurs intersections au sein d'un réseau routier, nous avons utilisé, pour chaque intersection, un vecteur à deux composantes représentant les durées des feux verts pour les directions Nord-Sud (NS) et Est-Ouest (EW). Cette approche permet une représentation structurée de la configuration de signalisation, facilitant ainsi l'optimisation des temps de passage en fonction des conditions de traffic.

$$\mathbf{V} = [NS_i, EW_i]$$

où :

- NS_i représente la durée du feu vert pour la direction Nord-Sud à l'intersection i.
- EW_i représente la durée du feu vert pour la direction la direction Est-Ouest à l'intersection i.

Définition du cycle :

$$C_i = NS_i + EW_i, \quad \forall i = 1, \ldots, n \tag{3.4}$$

Contraintes :

$$15 \leq NS_i \leq 120, \quad \forall i \quad \text{(Durée du feu vert pour NS)} \tag{3.5}$$
$$15 \leq EW_i \leq 120, \quad \forall i \quad \text{(Durée du feu vert pour EW)} \tag{3.6}$$

3.8 Organigramme de l'approche proposée

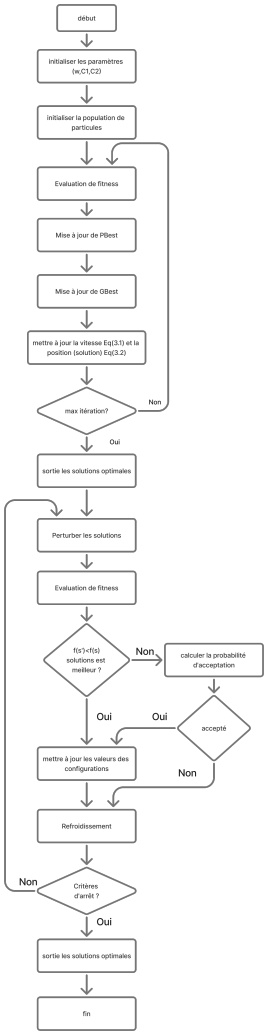

FIGURE 3.5 – organigramme de l'approche proposée

3.9 Conclusion

En conclusion, l'approche proposée combine l'Optimisation par Essaim de Particules (PSO) et le Recuit Simulé (SA) afin d'améliorer la gestion des feux de circulation dans un réseau routier urbain. Cette hybridation exploite la capacité exploratoire du PSO et le raffinement du SA pour éviter les optima locaux et assurer une convergence plus efficace. En intégrant le simulateur SUMO, cette méthodologie permet d'évaluer dynamiquement les performances des configurations de feux optimisées, offrant ainsi une solution adaptable aux variations du trafic en temps réel.

Chapitre 4

Expérimentations et Résultats

4.1 Introduction

Dans ce chapitre, nous présentons les expérimentations réalisées afin d'évaluer l'efficacité de notre approche. Nous décrivons d'abord l'environnement de développement utilisé, incluant les outils adoptés. Ensuite, nous détaillons les scénarios de simulation mis en place, en expliquant les paramètres, les hypothèses, et les métriques d'évaluation sélectionnées. Enfin, nous analysons les résultats obtenus en mettant en évidence les performances et les améliorations apportées par notre solution.

4.2 Environnement de développement

4.2.1 Python 3.12.4

Python est un langage de programmation de haut niveau, interprété, reconnu pour sa lisibilité, sa simplicité et sa polyvalence. Il prend en charge plusieurs paradigmes de programmation, notamment procédural, orienté objet et fonctionnel. Ce qui le rend adapté à un large éventail d'applications. Sa syntaxe, conçue pour améliorer la clarté et la maintenabilité du code.
De plus, sa vaste bibliothèque standard et ses capacités de typage dynamique contribuent à son adoption généralisée dans divers domaines, notamment le calcul scientifique, le développement web, l'intelligence artificielle et l'automatisation [54].

4.2.2 Javascript

JavaScript est un langage de programmation interprété léger principalement connu comme le langage de script pour les navigateurs Web, développé à l'origine par Brendan Eich en 1995 pour navigateur Netscape. JavaScript fonctionne également dans d'autres environnements tels que Node.js et est également utilisé côté serveur.
C'est un langage multi-paradigme et dynamique, il prend en charge des paradigmes tels que les styles orientés objet, impératif et déclaratif (par exemple, la programmation fonctionnelle). Il est mono-thread, une instruction exécuté à la fois [55]. Ce qui était une limitation jusqu'à l'introduction de Promises dans la spécification ECMAScript 6 (ES6), publiée en 2015 [56], la syntaxe async/await a été ajoutée plus tard dans ECMAScript 2017 (ES8) [57], simplifiant encore le développement de code asynchrone.

4.2.3 HTML

HTML (HyperText Markup Language) est l'élément de base du page web. Il définit le sens et la structure du contenu de page web. D'autres technologies que HTML sont généralement utilisées pour décrire l'apparence (CSS) ou les fonctionnalités et le comportement d'une page web (JavaScript). La première version de HTML a été écrite par Tim Berners-Lee en 1993. HTML 4.01 a été la version dominante dans les années 2000, ayant été officiellement normalisée en décembre 1999. La dernière version de HTML est HTML5 en 2008 [58].

HTML se compose de **balises**, des éléments spécifiques utilisés pour définir la structure et le contenu d'une page web. En HTML,Les balises se présentent généralement par paires : une balise d'ouverture et une balise de fermeture, comme par exemple `<p></p>` ou `<html></html>`. Chaque balise indique au navigateur l'élément à afficher (paragraphe, titre, bouton, image, etc..), leur nom est explicite et leur interprétation est aisée pour le navigateur [59].

4.2.4 Simulateur SUMO

SUMO (Simulation of Urban MObility) est un logiciel de simulation de trafic open source. Développé par le centre aérospatial allemand (DLR) en 2001 et publié en tant que projet open source en 2002. Il permet la modélisation du réseau routier, des mouvements des véhicules et des systèmes de contrôle du trafic, ce qui en fait un logiciel de simulation utile et robuste. SUMO prend en charge divers formats d'entrée, notamment OpenStreetMap (OSM) et les définitions de réseau routier basées sur XML. Il fournit une modélisation détaillée du comportement

des véhicules, y compris le suivi des véhicules et le changement de voie. Le simulateur s'intègre à des outils externes via TraCI (Traffic Control Interface), ce qui permet le contrôle de la simulation et de l'analyse des feux de circulation et des véhicules, ce qui en fait un logiciel robuste pour la simulation du trafic [10].

sumo inclut des outils comme :

— *sumo* : la simulation microscopique sans visualisation ; application en ligne de commande
— *sumo − gui* : une interface utilisateur graphique pour SUMO qui fournit une visualisation en temps réel des simulations de trafic.
— *netconvert* : un outil permettant de convertir et de générer des réseaux routiers à partir de différents formats en fichiers de réseau compatibles avec SUMO.
— *netedit* : un éditeur de réseau graphique
— *Netgenerate* : Un outil de création de réseaux routiers synthétiques pour les simulations SUMO, permettant aux utilisateurs de générer des tracés routiers basés sur une grille, aléatoires ou personnalisés. `https://sumo.dlr.de/docs/SUMO_at_a_Glance.html`

FIGURE 4.1 – simulateur sumo [10]

4.3 Étude de cas : zone d'Alger

Pour tester notre approche dans un environnement proche du monde réel, nous avons sélectionné une zone d'Alger, plus précisément Belouizdad, qui comprend 103 intersections et 1834 voie afin d'évaluer notre approche dans des conditions de circulation variables.

La première étape consiste à importer des données géographiques pour la zone ciblée. Pour cela, nous avons utilisé OpenStreentMap (OSM), une plateforme de cartographie gratuite et open source fournissant des données géographiques détaillées et disponible sous licence ODbL. La figure 4.2 illustre la zone sélectionnée à Alger en OpenStreetMap.

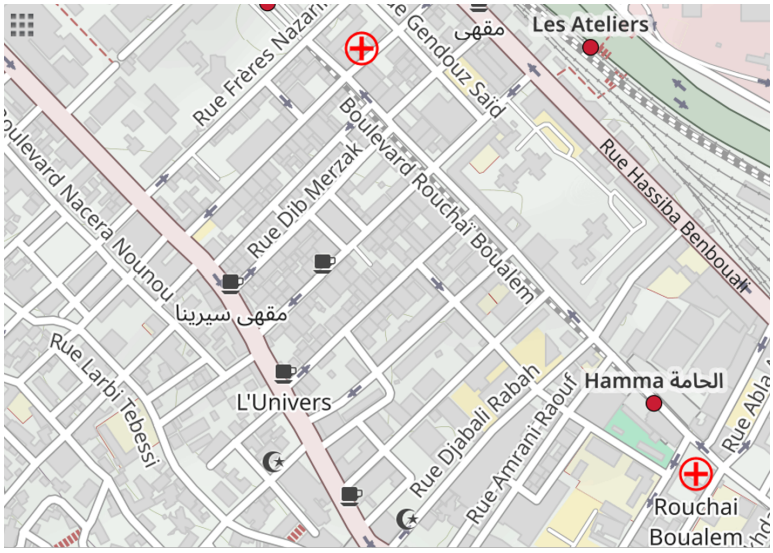

FIGURE 4.2 – Zone d'Alger

4.3.1 Préparation du scénario

Afin de simuler un environnement de circulation réaliste, nous avons sélectionné une zone représentative d'Alger comportant plusieurs intersections et axes routiers. Cette zone a été extraite à partir de la plateforme OpenStreetMap, puis convertie au format compatible avec le simulateur SUMO. Cette étape permet de reproduire fidèlement la topologie routière réelle pour l'évaluation des performances de notre approche d'optimisation. Les figures 4.3 et 4.4 illustrent la zone sélectionnée à Alger, telle qu'elle apparaît dans OpenStreetMap et dans le simulateur SUMO.

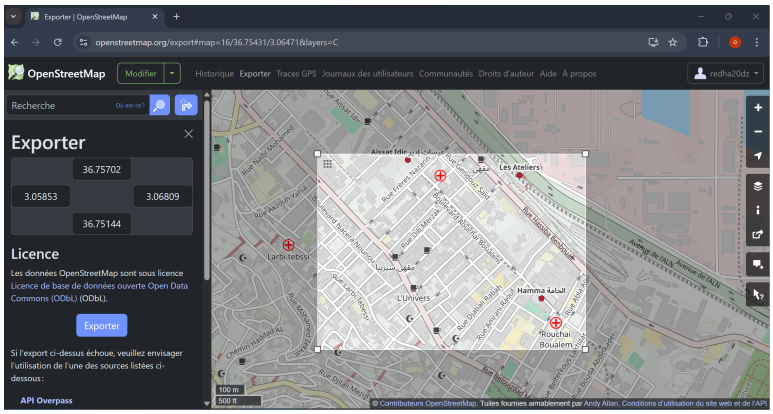

FIGURE 4.3 – Zone d'Alger en OpenStreetMap

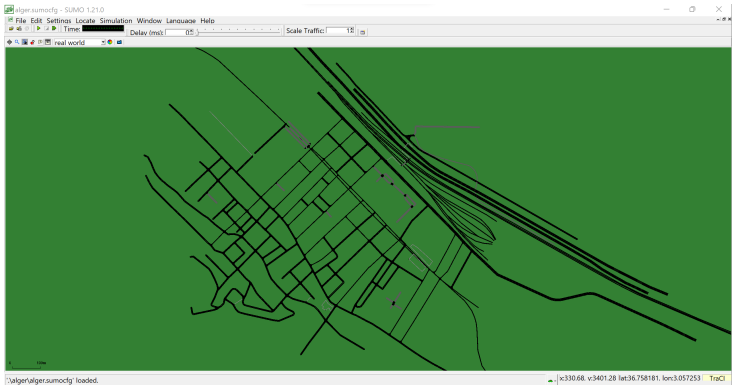

FIGURE 4.4 – Zone d'Alger dans SUMO

4.4 Configuration des Simulations

La première étape pour configurer une simulation dans **SUMO** consiste à créer un fichier de configuration (`.sumocfg`). Ce fichier spécifie les composants essentiels nécessaires à la simulation, notamment : le fichier de réseau (`.net.xml`) et le fichier de routes (`.rou.xml`). Le fichier `.net.xml` contient la description du réseau routier utilisé dans la simulation, incluant les routes, les intersections, les voies, les limitations de vitesse et les feux de signalisation. Il peut être généré à partir de données OpenStreetMap (OSM), créé manuellement avec `netedit` ou construit à l'aide de `NETCONVERT` à partir de données brutes. Le fichier `.rou.xml`, quant à lui, définit les trajets et les itinéraires des véhicules, en précisant les types de véhicules, les horaires de départ ainsi que les parcours prédéfinis ou générés dynamiquement. Le fichier `.sumocfg` peut également inclure des configurations supplémentaires telles que les programmes de feux de signalisation (`.tll.xml`), des définitions de demande de trafic supplémentaires (`.add.xml`) ou des paramètres `TraCI` pour le contrôle dynamique de la simulation via Python. Ce fichier de configuration sert ainsi de point d'entrée pour exécuter la simulation SUMO.

Pour générer des trajets de véhicules aléatoires pour la simulation, un

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on 2025-03-13 14:47:41 by Eclipse SUMO sumo Version 1.21.0
-->

<sumoConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

    <input>
        <net-file value="alger.net.xml"/>
        <route-files value="alger1.rou.xml"/>
        <additional-files value=""/>
    </input>

</sumoConfiguration>
```

FIGURE 4.5 – Alger.sumocfg

script Python appelé *randomTrips.py* est inclus dans SUMO. Ce script est responsable de la création de trajets générés aléatoirement pour les véhicules, y compris leurs itinéraires, leur heures de départ,arrivée et la distribution du trafic. Le script fournit un ensemble de paramètres configurables pour contrôler la génération des trajets. il permet d'ajuster les heures de départ et les distances des trajets, parmi d'autres paramètres, en affinant ces paramètres, nous pouvons simuler différents niveaux de congestion similaires au monde réel.

4.5 Interaction avec la simulation

Pour interagir avec la simulation, le simulateur SUMO fournit une API appelée TraCI pour l'interaction avec la simulation. TraCI est l'abréviation de « Traffic Control Interface », c'est une API qui donne accès à une simulation SUMO, permettant de contrôler la simulation à partir d'applications externes qui permettent la surveillance, la capture et modification de la simulation pendant l'exécution [60].

4.5.1 Communication avec SUMO via TraCI

Pour capturer la complexité de la dynamique du trafic dans notre région d'étude, nous avons développé un script Python qui interagit avec notre simulation de trafic. Ce script permet un contrôle en temps réel ainsi qu'un ajustement dynamique des durées des feux de signalisation. Il commence par l'importation de la bibliothèque `traci`, qui fournit les fonctions essentielles pour gérer la simulation, ainsi que du module `csv`, utilisé pour enregistrer les données de simulation dans un fichier nommé `sim_data.csv`. Une fois ces bibliothèques importées, nous chargeons le fichier de configuration (`.sumocfg`), qui initialise et

FIGURE 4.6 – Alger.net.xml

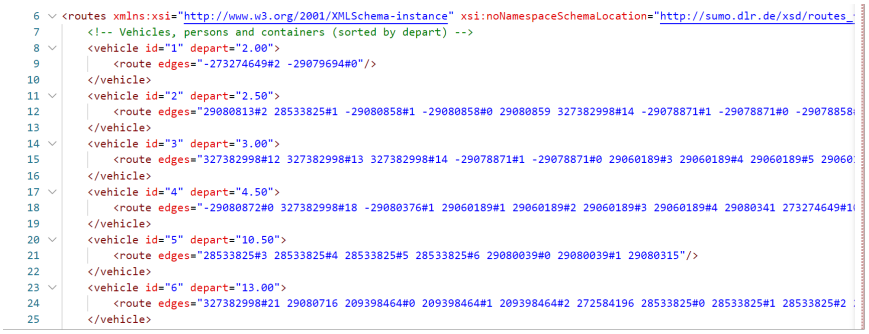

FIGURE 4.7 – Configuration de l'itinéraire du véhicule *alger.rou.xml*

lance l'environnement de simulation.

Après le lancement de la simulation, une boucle est utilisée pour simuler l'évolution au fil des pas de temps. La simulation progresse d'une seconde à chaque itération, correspondant au pas de simulation par défaut. À chaque étape, des données clés sur le trafic sont recueillies, telles que les temps d'attente, le nombre de véhicules et les émissions de CO_2. Nous suivons également le nombre de véhicules actuellement en circulation ainsi que ceux ayant atteint leur destination. De plus, nous enregistrons le pire temps d'attente observé à une intersection lors du cycle précédent, ainsi que la longueur maximale de file d'attente constatée afin d'évaluer plus finement les performances du système.

Ce flux continu de données sur le trafic permet d'ajuster en temps réel la durée des feux de signalisation, afin de les adapter en perma-

nence aux conditions de circulation actuelles. La simulation se poursuit jusqu'à ce qu'elle atteigne un nombre prédéfini d'étapes par exemple 7200 étapes, soit deux heures ou jusqu'à ce qu'il ne reste plus aucun véhicule dans la zone de simulation.

L'exécution du programme Python démarre le simulateur SUMO et lance la simulation présentée par les figures 4.8 et 4.9

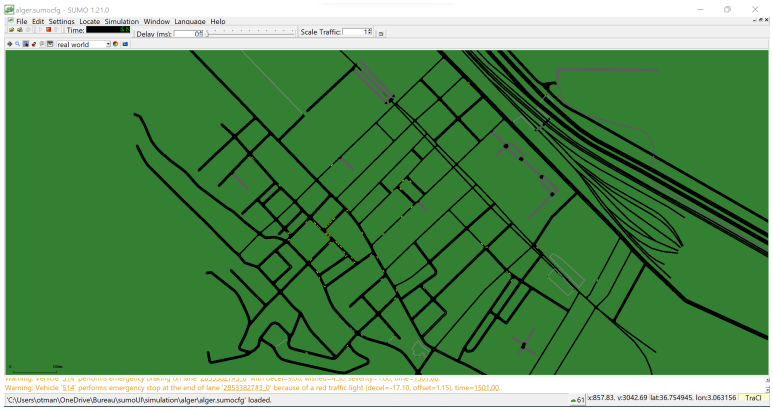

FIGURE 4.8 – Simulation en cours d'exécution

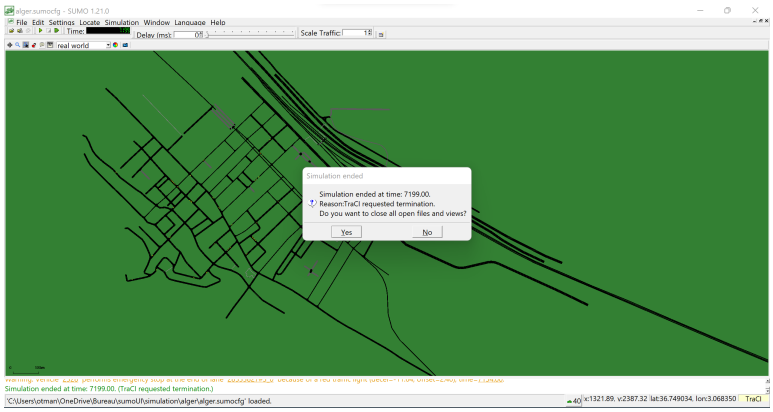

FIGURE 4.9 – Fin de la simulation

Une fois la simulation terminée, un fichier *sim_data.csv* sera créé dans le même répertoire

	cycle	avg waiting time	active cars	new car	arrival cars	worst intersection time	longest queue
2	0	22	33	34	1	138.0	3
3	1	33	58	28	6	229.0	6
4	2	32	77	22	16	278.0	6
5	3	33	93	35	25	348.0	5
6	4	26	104	36	46	234.0	5
7	5	28	95	34	37	148.0	4
8	6	26	85	30	22	141.0	5
9	7	29	91	28	30	240.0	6
10	8	29	91	32	27	322.0	5
11	9	28	102	37	27	190.0	7
12	10	34	112	36	45	595.0	8
13	11	27	113	45	38	335.0	7
14	12	29	96	24	38	303.0	6
15	13	24	89	31	47	233.0	5
16	14	28	81	38	31	130.0	4
17	15	32	83	34	29	298.0	6
18	16	33	80	25	20	422.0	9
19	17	32	79	18	28	504.0	9
20	18	33	96	43	29	569.0	9
21	19	42	95	28	31	571.0	9
22	20	38	84	22	31	389.0	6
23	21	34	86	31	23	429.0	8
24	22	31	89	26	35	498.0	9
25	23	32	86	31	24	283.0	10

FIGURE 4.10 – Le fichier *sim_data.csv*

4.6 Présentation de l'application

L'objectif de notre application est d'ajuster la synchronisation des feux de circulation en fonction des conditions de circulations. Cela permet de réduire les embouteillages, d'améliorer les temps de trajet, ainsi que de diminuer les temps d'attente et les émissions. Dans cette section, nous décrivons les différents éléments de l'application.

Notre application web s'ouvre sur un tableau de bord centralisé, conçu pour piloter facilement la simulation et l'optimisation du trafic. Dès le chargement, l'utilisateur dispose de deux boutons principaux : l'un pour lancer la simulation et l'optimisation en temps réel, l'autre pour l'interrompre instantanément.

Une fois la simulation démarrée, la partie gauche de l'écran présente deux graphiques dynamiques : l'un retrace l'évolution du temps d'attente moyen à chaque cycle, l'autre affiche la quantité de CO_2 émise au fil du temps. À droite, un panneau d'informations de statut se met à jour en continu pour indiquer le numéro du cycle en cours, le nombre total de véhicules ayant circulé dans le réseau, le nombre de véhicules encore actifs et le volume d'émissions généré lors du cycle actuel.

Pour explorer différents scénarios de trafic, l'application propose un champ de saisie permettant d'ajuster la densité de circulation : on peut ainsi simuler à volonté des conditions allant du flux fluide à la saturation urbaine. Cette fonctionnalité facilite la comparaison de performances du système de feux sous divers niveaux de charge.

Enfin, un bouton « Exporter » permet de télécharger l'intégralité

des données de simulation au format CSV. Le fichier généré contient, pour chaque cycle, des données telles que le temps d'attente moyen, le nombre de véhicules actifs, le nombre de voitures nouvellement parties, ainsi que le nombre d'arrivées à destination.

FIGURE 4.11 – Application web

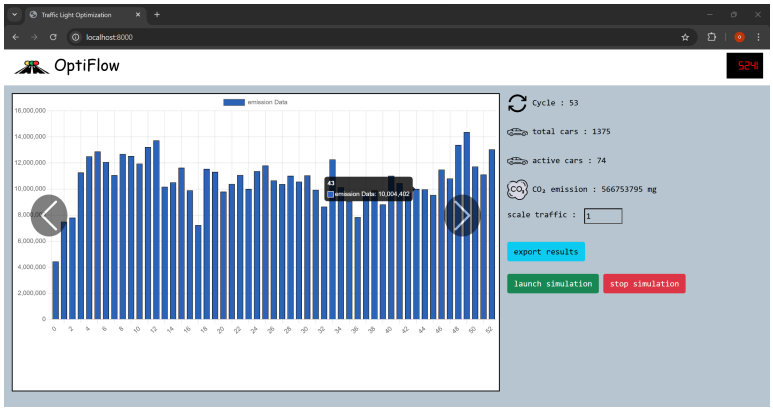

FIGURE 4.12 – Application web suite

4.7 Résultats expérimentaux

Dans cette section, nous présentons les résultats obtenus à partir des simulations de notre approche proposée et de l'approche statique de contrôle des feux tricolores. Au total, 63 simulations ont été réalisées sur

une période de 7 jours pour chaque approche, à raison de 9 simulations par jour, afin de couvrir un large éventail de conditions de circulation. Les données collectées comprennent des indicateurs clés de performance tels que les **émissions de CO_2**, le **temps d'attente moyen** et le **temps de trajet moyen**.

Les scénarios de simulation ont été conçus pour représenter différentes intensités de trafic, allant d'une circulation fluide pendant les heures creuses à une forte congestion pendant les heures de pointe (par exemple de 12h00 à 14h00). Cette diversité permet d'évaluer de manière efficace la performance et l'adaptabilité de chaque approche dans un contexte urbain dynamique et réaliste.
les résultats sont présentés ci-dessous.

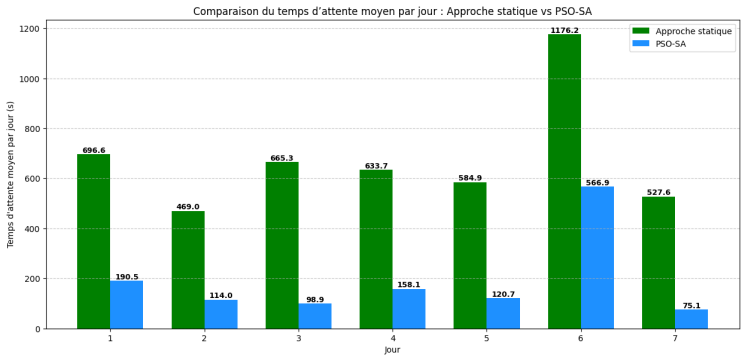

FIGURE 4.13 – Résultat du temps d'attente moyen

Le graphique présente les temps d'attente moyens, calculés à partir des données de trafic en temps réel. On observe que l'approche dynamique (en vert) génère systématiquement des temps d'attente inférieurs à ceux de l'approche statique (en bleu) pour chaque jour analysé. Ces résultats suggèrent qu'une adaptation dynamique des durées de feux en fonction des conditions de circulation permet de réduire significativement les temps d'attente globaux.
Les pourcentages de réduction des temps d'attente moyens obtenus avec notre approche par rapport à ceux de l'approche statique pour chaque jour, présentés dans le tableau 4.1.

Le graphique compare les émissions de CO_2 générées par l'approche statique (en bleu) et par l'approche dynamique (en rouge) au cours des différents jours. Il met en évidence une réduction systématique des émissions avec la stratégie dynamique par rapport à la méthode sta-

Jour	statique (s)	dynamique (s)	Réduction (%)
1	696.59	190.47	72.66
2	469.01	113.98	75.70
3	665.28	98.89	85.14
4	633.75	158.14	75.05
5	584.87	120.74	79.36
6	1176.15	566.89	51.80
7	527.61	75.06	85.77

TABLE 4.1 – Comparaison des temps d'attente moyens et de la réduction entre l'approche statique et l'approche dynamique

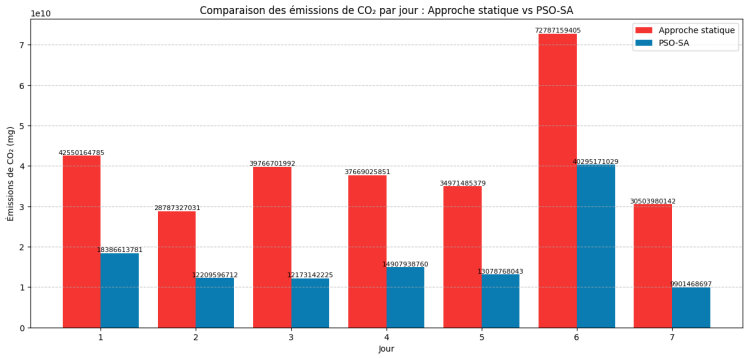

FIGURE 4.14 – Comparaison des émissions de CO_2

tique.
Les pourcentages de réduction des émissions obtenus avec notre approche par rapport à ceux de l'approche statique pour chaque direction, présentés dans le tableau ci-dessus.

Le graphique compare les temps de trajet moyens entre l'approche statique (en gris) et l'approche dynamique (en jaune) sur plusieurs jours. Il met en évidence une réduction notable des temps de trajet avec l'approche dynamique, en comparaison avec la méthode statique.

Jour	Statique (mg)	PSO-SA (mg)	Réduction (%)
1	4.26×10^{10}	1.84×10^{10}	56.79
2	2.88×10^{10}	1.22×10^{10}	57.59
3	3.98×10^{10}	1.22×10^{10}	69.39
4	3.77×10^{10}	1.49×10^{10}	60.42
5	3.50×10^{10}	1.31×10^{10}	62.60
6	7.28×10^{10}	4.03×10^{10}	44.64
7	3.05×10^{10}	9.90×10^{9}	67.54

TABLE 4.2 – Réduction des émissions de CO_2 entre l'approche statique et PSO-SA

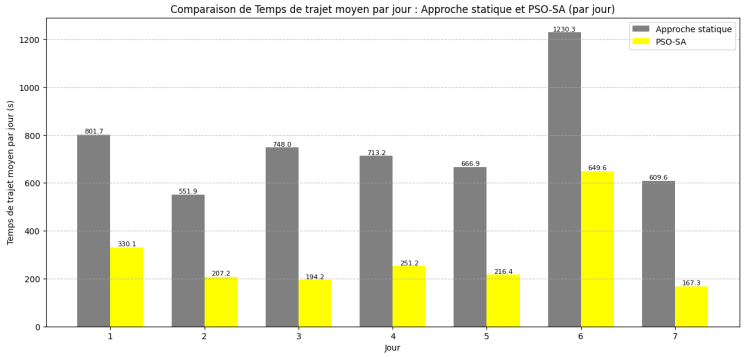

FIGURE 4.15 – Comparaison des temps de trajet

4.8 Discussion des résultats

L'objectif de cette simulation est de comparer l'efficacité de notre approche dynamique à celle d'une stratégie statique, en termes de réduction des émissions de CO_2, des temps d'attente et des temps de trajet.

Notre méthode démontre des performances supérieures grâce à l'utilisation de plans de feux circulation dynamiques et à l'intégration de mécanismes d'adaptation en temps réel, permettant d'ajuster les cycles des feux de circulation en fonction des conditions de trafic observées. Par exemple, le jour 3 enregistre une réduction maximale des émissions de CO_2 de **69,39 %**, tandis que la réduction la plus faible est observée le jour 6, avec **44,64 %**.

Ces résultats mettent en évidence la capacité des méta-heuristiques en particulier l'optimisation par essaim de particules (PSO) et le recuit simulé (SA) à s'adapter dynamiquement aux variations de tra-

fic. Cela permet une optimisation efficace des temps de feux en temps réel, contrairement à l'approche statique qui reste figée face aux pics de circulation, entraînant ainsi des files d'attente plus longues et une consommation de carburant accrue.

Les figures 4.14 et 4.15 illustrent les écarts significatifs de performance entre notre approche dynamique et l'approche statique.

les résultats obtenus confirment l'intérêt d'une stratégie de régulation dynamique des feux tricolores basée sur des algorithmes métaheuristiques, capable de répondre efficacement aux besoins réels du trafic urbain.

4.9 Conclusion

Dans ce chapitre, nous avons présenté les outils techniques utilisés, l'étude de cas choisie ainsi que l'application d'optimisation mise en œuvre pour la régulation du trafic dans la zone étudiée. L'analyse des résultats a permis de démontrer l'efficacité de l'approche proposée, qui s'est révélée performante de manière constante lors de chaque simulation.

Conclusion Général

La régulation moderne du trafic aux intersections, notamment par l'utilisation des feux de signalisation, joue un rôle essentiel dans l'amélioration de la fluidité de la circulation. Dans cette optique, notre travail a visé à concevoir et à mettre en œuvre une approche dynamique de gestion des feux, fondée sur des algorithmes méta-heuristiques.
L'utilisation d'algorithmes méta-heuristiques, tels que l'optimisation par essaim de particules (PSO) et le recuit simulé (SA), permet d'améliorer la gestion des feux de circulation. Le PSO s'inspire du comportement des oiseaux volant en groupe ou des poissons nageant en bancs, et le recuit simulé s'inspire du procédé de recuit utilisé en métallurgie. Notre approche proposée ajuste la synchronisation des feux de circulation en temps réel. Cela contribue à fluidifier de la circulation sur le réseau et à réduire le temps d'attente aux intersections.
Les résultats obtenus montrent un fort potentiel de l'approche proposée pour réduire efficacement les temps d'attente aux intersections, la congestion, ainsi que les durées de trajet des usagers. En contribuant à une circulation plus fluide, cette méthode pourrait améliorer la qualité de vie en zone urbaine, en limitant le bruit et les risques d'accidents liés aux embouteillages. Ainsi, cette étude ouvre de nouvelles perspectives pour une gestion intelligente et optimisée du trafic routier, en s'appuyant sur les avancées récentes en matière d'algorithmes d'optimisation.
Dans le cadre de travaux futurs, il serait intéressant d'étendre notre approche à un cadre d'optimisation multi-objectifs afin de prendre en compte simultanément plusieurs critères contradictoires. De plus, l'intégration de la logique floue pourrait offrir plus de flexibilité dans la gestion des incertitudes et du flou des schémas de trafic, permettant ainsi un processus décisionnel plus précis.
Bien que les résultats soient prometteurs, il serait intéressant de tester cette approche sur un réseau routier réel et l'intégration des données de trafic historiques issues du monde réel pour valider ses performances dans des conditions opérationnelles.

Bibliographie

[1] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system : A comprehensive review," *Robotics and Autonomous Systems*, vol. 174, p. 104630, 2024.

[2] A. Burduk, A. Balashov, D. Łapczyńska, and K. Musiał, "The pallet-loading problem solution with the use of heuristic algorithms," in *Journal of Physics : Conference Series*, vol. 2198, p. 012063, 2022.

[3] L. M. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.

[4] H. Asefi, S. Lim, M. Maghrebi, and S. Shahparvari, "Mathematical modelling and heuristic approaches to the location-routing problem of a cost-effective integrated solid waste management," *Annals of Operations Research*, vol. 273, pp. 1–22, 2019.

[5] K. M. Amine, N. Akkouche, S. Awad, K. Loubar, and M. Tazerout, "Kinetic study of transesterification using particle swarm optimization method," *Heliyon*, vol. 5, no. 7, p. e02146, 2019.

[6] I. Önder, "A genetic algorithm for tsp with backhauls based on conventional heuristics," 2007.

[7] V. Chawla, A. Chanda, and S. Angra, "Automatic guided vehicles fleet size optimization for flexible manufacturing system by grey wolf optimization algorithm," *Management Science Letters*, vol. 8, no. 1, pp. 79–90, 2018.

[8] T. Ting, X.-S. Yang, S. Cheng, and K. Huang, "Hybrid metaheuristic algorithms : Past, present, and future," in *Metaheuristics : Progress in Complex Systems Optimization*, pp. 57–77, Springer, 2015.

[9] J. Lin, Y. Zhong, E. Li, X. Lin, and H. Zhang, "Multi-agent simulated annealing algorithm with parallel adaptive multiple sampling

for protein structure prediction in ab off-lattice model," *Applied Soft Computing*, vol. 62, pp. 491–503, 2018.

[10] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018.

[11] C. Caron-Telders, "La mobilité urbaine, le défi de la ville de demain," *Les Echos*, 2020. Accessible depuis Les Echos.

[12] OCDE, *Gérer la congestion urbaine*. Paris : Organisation de Coopération et de Développement Économiques, 2010.

[13] Ministère des Transports, "Quelles informations cachent les feux tricolores ?," 2025. Consulté le 02/04/2025.

[14] M. Tlig, *Coordination locale et optimisation distribuée du trafic de véhicules autonomes dans un réseau routier*. Thèse de doctorat, Université de Lorraine, France, 2015. Intelligence artificielle [cs.AI], NNT : 2015LORR0049.

[15] DriversEducationUSA.com, "What is an intersection ?," 2023. Consulté le 29/04/2025.

[16] H. Wei, "Modern traffic light – material 1 final," 2019. Consulté le 19/04/2025.

[17] Ministère de l'Écologie, du Développement Durable et de l'Énergie, "Instruction interministérielle sur la signalisation routière – 6e partie – vc20121231," 2012. Consulté le 19/04/2025.

[18] D. I. Robertson, "TRANSYT : A Traffic Network Study Tool," Tech. Rep. LR 253, Road Research Laboratory, 1969. This report introduces the TRANSYT model, detailing its methodology for optimizing traffic signal timings to improve network efficiency.

[19] TRL Software, "TRANSYT Version History." `https://trlsoftware.com/software/junctions-signal-design/transyt/transyt-version-history/`, n.d. Consulté le 19/04/2025.

[20] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The scoot online traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, pp. 190–192, 1982.

[21] P. R. Lowrie, "Scats : The sydney co-ordinated adaptive traffic system - principles, methodology, algorithms," in *IEE Conference Publication No. 355, International Conference on Road Traffic Signalling*, pp. 67–70, IET, 1992.

[22] N. Sharma and S. Sahu, "Fuzzy logic-based traffic signal control," *Indian Journal of Science and Technology*, vol. 11, no. 23, pp. 1–7, 2018.

[23] Great Learning, "Fuzzy logic tutorial : History, implementation and advantages," November 18 2024.

[24] M. Smadi, "Intelligent transportation systems : Architecture, deployment, and advances." `https://www.ugpti.org/events/seminar/downloads/2008-05-07MohammadSmadi.pdf`, 2008. [Presentation], Upper Great Plains Transportation Institute.

[25] S. A. Shaheen and R. Finson, "Intelligent transportation systems," in *Encyclopedia of Energy* (C. J. Cleveland, ed.), pp. 487–496, New York : Elsevier, 2004.

[26] S. Almufti, A. Shaban, R. Ali, and J. Fuente, "Overview of metaheuristic algorithms," *Polaris Global Journal of Scholarly Research and Trends*, vol. 2, pp. 10–32, 2023.

[27] K. Cherifi, "An overview on recent machine learning techniques for port hamiltonian systems," *Physica D : Nonlinear Phenomena*, vol. 411, p. 132620, 2020.

[28] Y. Tan, "Introduction," in *GPU-based Parallel Implementation of Swarm Intelligence Algorithms* (Y. Tan, ed.), pp. 1–7, Morgan Kaufmann, 2016.

[29] S. Sridharan, R. K. Subramanian, and A. K. Srirangan, "Physics based meta heuristics in manufacturing," *Materials Today : Proceedings*, vol. 39, no. Part 1, pp. 805–811, 2021.

[30] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa : A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[31] P. Trojovský, "A new human-based metaheuristic algorithm for solving optimization problems based on preschool education," *Scientific Reports*, vol. 13, p. 21472, 2023.

[32] Artificial Intelligence Master Class, "What is metaheuristic algorithms in ai ?," 2025.

[33] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[34] Algorithm Afternoon, "Tabu search," n.d. consulté le 27/01/2025.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[36] A. Afternoon, "Simulated annealing." `https://algorithmafternoon.com/physical/simulated_annealing/`, n.d. Consulté le 27/01/2025.

[37] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[38] A. Afternoon, "Variable neighborhood search." `https://algorithmafternoon.com/stochastic/variable_neighborhood_search`, n.d. Consulté le 27/01/2025.

[39] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[40] K. Y. Lee, "Particle swarm optimization in power systems," in *IEEE Power Systems Conference and Exposition (PSCE)*, 2006.

[41] A. G. Gad, "Particle swarm optimization algorithm and its applications : A systematic review," *Archives of Computational Methods in Engineering*, vol. 29, pp. 2531–2561, 2022.

[42] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI : University of Michigan Press, 1975.

[43] Algorithm Afternoon, "Genetic algorithm," n.d. Retrieved January 28, 2025.

[44] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[45] Y. Hou, H. Gao, Z. Wang, and C. Du, "Improved grey wolf optimization algorithm and application," *Sensors*, vol. 22, no. 10, p. 3810, 2022.

[46] L. Singh, S. Tripathi, and H. Arora, "Time optimization for traffic signal control using genetic algorithm," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, pp. 239–241, 2009.

[47] G. Silva, A. Rocha, G. Witeck, A. Silva, D. Durães, and J. Machado, "On tuning the particle swarm optimization for solving the traffic light problem," in *Intelligent Systems and Applications*, pp. 68–77, Springer, 2022.

[48] I. Abdulhameed, A. Akanni, and E. Omidiora, "Analysis of simulated annealing to decongest traffic in a multi-road coordinated intersection," *FUOYE Journal of Engineering and Technology*, vol. 8, no. 2, 2023.

[49] S. M. Odeh, A. M. Mora, M. N. Moreno, and J. J. Merelo, "A hybrid fuzzy genetic algorithm for an adaptive traffic signal system," *Advances in Fuzzy Systems*, vol. 2015, p. 11 pages, 2015.

[50] K.-R. Lo and S.-L. Tung, "Traffic signal control based on particle swarm optimization," in *21st World Congress on Intelligent Transport Systems (ITSWC 2014) : Reinventing Transportation in Our Connected World*, 2014.

[51] T.-C. Teng, M.-C. Chiang, and C.-S. Yang, "A hybrid algorithm based on gwo and goa for cycle traffic light timing optimization," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, (Bari, Italy), pp. 774–779, IEEE, 2019.

[52] C. Cintrano, J. Ferrer, M. López-Ibáñez, and E. Alba, "Hybridization of racing methods with evolutionary operators for simulation optimization of traffic lights programs," in *Evolutionary Computation in Combinatorial Optimization. EvoCOP 2021* (C. Zarges and S. Verel, eds.), vol. 12692 of *Lecture Notes in Computer Science*, pp. 24–40, Springer, 2021.

[53] M. A. Khasawneh and A. Awasthi, "Intelligent meta-heuristic-based optimization of traffic light timing using artificial intelligence techniques," *Electronics*, vol. 12, no. 24, p. 4968, 2023.

[54] Python Software Foundation, "Python programming language." `https://www.python.org/`, 2025. Consulté le 20/04/2025.

[55] MDN contributors, "Javascript - mdn web docs." `https://developer.mozilla.org/en-US/docs/Web/JavaScript`, 2025. consulté le 2025-04-12.

[56] Ecma International, "ECMAScript® 2015 Language Specification (6th ed.)." `https://262.ecma-international.org/6.0/`, 2015. consulté le 2025-04-12.

[57] Ecma International, "ECMAScript® 2017 Language Specification (8th ed.)." `https://262.ecma-international.org/8.0/`, 2017. consulté le 2025-04-12.

[58] University of Washington, "A brief history of html." `https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html`, n.d. consulté le 13/04/25.

[59] University of Washington, "Web design and development curriculum." `https://www.washington.edu/accesscomputing/webd2/student/lessons.html`, n.d. consulté le 20/04/25.

[60] Eclipse SUMO Project, "TraCI – Traffic Control Interface Documentation." `https://sumo.dlr.de/docs/TraCI.html`, 2025. consulté le 20/04/25.