N° Ordre...../Faculté/UMBB/2016

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **H**ydrocarbures et de la **C**himie

Thèse de Doctorat

Présentée par

MERAIHI Yassine

Filière: Génie Electrique et Electrotechnique

Option : Infotronique

Qualité de service dans les réseaux sans fil maillés/Vanet

Devant le jury:

AIT AOUDIA	Sami	Professeur	ESI Oued Smar	Président
ACHELI	Dalila	Professeur	UMBB	Rapporteur
HIDOUCI	Walid-Khaled	Professeur	ESI Oued Smar	Examinateur
KESRAOUI	Mohamed	MCA	UMBB	Examinateur
NADJI	Bouchra	Professeur	UMBB	Invitée

Année Universitaire : 2016/2017

REMERCIEMENTS

Je tiens tout d'abord à adresser mes plus vifs remerciements à ma directrice de thèse Dalila ACHELI pour l'aide qu'elle m'a apportée, ses précieux conseils, ses qualités humaines et scientifiques et son soutien constant tout au long de cette thèse.
Je tiens à remercier également Monsieur Amar RAMDANE-CHERIF Professeur des Universités à l'Université de Versailles Saint Quentin en Yvelines pour avoir codirigé ce travail, pour son aide, ces critiques, ces précieux conseils et sa disponibilité imminente malgré ses différents engagements.

Je voudrais également remercier sincèrement les membres du jury. D'abord, le professeur AIT AOUDIA Sami qui m'a honoré en acceptant d'être le président de jury formé pour la soutenance de ma thèse. Ensuite, les professeurs HIDOUCI Walid-Khaled et KESRAOUI Mohamed pour avoir évalué ma thèse et apporté des commentaires judicieux et le Professeur NADJI Becharia d'avoir accepté mon invitation.

Je remercie bien évidement mes parents, à qui je dédie ce travail. Ils m'ont non seulement encouragé et supporté tout au long de mes études mais dans toutes les sphères de ma vie. Je remercie amoureusement ma chère femme et mes chers adorables enfants Maya et Mohamed Wassim qui m'ont supporté et m'ont encouragée tout au long de la rédaction de cette thèse.

Je voudrais aussi remercier mes frères, mes sœurs, mes beaux frères, mes belles sœurs, mes neveux, mes nièces et mes amis qui me permettent de garder un équilibre de vie en alliant études, travail et loisir pour leur soutien durant la période de rédaction de cette thèse.

Je remercie également tous ceux qui ont contribué de près ou de loin pour l'aboutissement de ce travail, notamment MEGHRICHE Kamel, ARTIGAUD Alain, LAHLALI Mouna, LAHLALI Feriel, MAHSEUR Mohamed, RAHMOUNE Fayçal, BAICHE Karim, GHOMARI Lila, BENMESSAOUD Asma, OMARI Taher, DICHOU Karima, LALAMI Souad, MESSAD Zineb, SACI Nacera.

RÉSUMÉ

Une des contraintes des réseaux sans fil maillés et des réseaux véhiculaires est le problème transmission de données depuis une source vers un groupe de destinations avec un réel besoin de garantie sur la qualité de service (QoS). Ce problème appelé problème de routage multicast avec QoS est un problème d'optimisation NP-Difficile, il est impossible de le résoudre par les méthodes exactes traditionnelles. Plusieurs métaheuristiques ont été utilisées pour résoudre ce problème et trouver un arbre multicast à moindre coût qui satisfait les contraintes de délai, de gigue, de bande passante et de taux de perte des paquets.

Dans ce travail, nous proposons deux contributions. Dans la première contribution, nous proposons une amélioration de l'algorithme des chauves-souris BA pour résoudre le problème de routage multicast avec QoS. Nous introduisons deux méthodes de modification dans l'algorithme des chauves-souris binaire BBA. Dans la première méthode, nous utilisons les deux systèmes chaotiques les plus fréquemment utilisés, appelés fonction logistique et fonction tent, pour déterminer la valeur du paramètre β de l'impulsion de fréquence f_i. Dans la deuxième méthode, nous utilisons la formulation dynamique pour mettre à jour paramètre α de l'intensité A_i. Les résultats expérimentaux montrent la supériorité des algorithmes proposés par rapport à d'autres méthodes existantes dans la littérature.

Dans la deuxième contribution, nous proposons deux approches hybrides basées sur l'hybridation de l'algorithme des chauves-souris binaire BBA avec l'algorithme évolutionnaire quantique QEA. La première approche, nommée BBAQEA1, est basée sur l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. La deuxième approche, nommée BBAQEA2, est basée sur le remplacement de l'opérateur évolutionnaire quantique de QEA par l'équation d'évolution de BBA. Les résultats des expérimentations ont montré que BBAQEA2 est la meilleure solution dans la plupart des cas pour les deux types de réseaux sans fil.

Mots clés: Qualité de service (QoS), Routage multicast, Réseaux sans fil maillés, VANETs, Algorithme de chauves-souris BA, Algorithme évolutionnaire quantique QEA, Systèmes chaotiques, Méthodes d'optimisation.

ABSTRACT

One of the constraints of wireless mesh networks and vehicular ad hoc networks is the problem of data transmission from a source node to a group of destination nodes with a real need of quality of service (QoS) guarantee. This problem, called multicast routing with QoS problem, is an NP-hard optimization problem, it is impossible to solve it by traditional exact methods. Several metaheuristics have been used to solve this problem and find a minimized cost multicast tree that satisfies the constraints of delay, jitter, bandwidth, and packet loss rate. In this work, we propose two contributions. In the first contribution, we propose an improved bat algorithm BA to solve the problem of multicast routing with QoS. We introduce two modification methods in the binary bats algorithm BBA. In the first method, we use the two most representative chaotic maps, namely the logistic map and the tent map, to determine the value of the β parameter of the pulse frequency f_i. In the second method, we use the dynamic formulation to update the parameter α of the loudness A_i. The experimental results show the superiority of the proposed algorithms compared with other existing methods in the literature.
In the second contribution, we propose two hybrid approaches based on hybridization of the binary bats algorithm BBA with quantum evolutionary algorithm QEA. The first approach, called BBAQEA1, is based on the integration of the evolutionary equation of BBA in the quantum operator of QEA. The second approach, called BBAQEA2, is based on the replacement of quantum evolutionary operator of QEA by the evolutionary equation of BBA. The experimental results show that BBAQEA2 is the best solution in most cases for the two types of wireless networks.

Keywords: Quality of service (QoS), Multicast routing, Wireless mesh networks, VANETs, Bat Algorithm BA, Quantum Evolutionary Algorithm QEA, Chaotic maps, Optimization methods.

ملخص

واحدة من بين قيود الشبكات المعلوماتية العقدية اللاسلكية و شبكات السيارات هي مشكلة نقل و إرسال المعلومات من مصدر إلى مجموعة من الوجهات مع حاجة حقيقية لضمان جودة الخدمة (QoS). هذه المشكلة المسماة مشكلة الإرسال المتعدد مع ضمان جودة الخدمة هي مشكلة استمثال و تحسين صعبة من المستحيل حلها بالطرق الصحيحة التقليدية. عدة أدلة عالية استعملت لحل هذه المشكلة لإيجاد شجرة الإرسال المتعدد بأقل التكاليف التي تستجيب لقيود الوقت، عرض النطاق الترددي و معدل فقدان حزمة المعلومات.

في هذا العمل نقترح مساهمتين، في أول مساهمة نقترح تحسين خوارزمية الخفافيش (BA) لحل مشكلة الإرسال المتعدد مع ضمان جودة الخدمة. نقترح طريقتين لتعديل خوارزمية الخفافيش الثنائية، في الطريقة الأولى نستخدم النظامان العشوائيان الأكثر استعمالا: النظام اللوجستيكي و نظام التانت من أجل تحديد قيمة β للنبض الترددي f_i . في الطريقة الثانية نستعمل الصيغة الديناميكية لتعديل α للمكثفة A_i . النتائج التجريبية أظهرت تفوق الخوارزميات المقترحة مقارنة بالطرق الموجودة بالأدبيات.

في المساهمة الثانية نقترح منهجيتان مختلطتان مبنيتان على تهجين خوارزمية الخفافيش الثنائية (BBA) مع خوارزمية الكم التطوري (QEA) . المنهجية الأولى المسماة (BBAQEA1) ترتكز على التكامل بين معادلة التطوير لخوارزمية (BBA) و معالج الكم لخوارزمية (QEA). المنهجية الثانية المسماة (BBAQEA2) ترتكز على تحويل المعالج التطوري لخوارزمية (QEA) بمعادلة التطوير لخوارزمية (BBA) . النتائج التجريبية أظهرت أن (BBAQEA2) هي الحل الأفضل في معظم الحالات لكل من النوعين من الشبكات اللاسلكية.

الكلمات المفتاحية: ضمان جودة الخدمة (QoS)، الإرسال المتعدد، الشبكات المعلوماتية العقدية اللاسلكية، شبكات السيارات، خوارزمية الخفافيش (BA)، خوارزمية الكم التطوري(QEA) ، الأنظمة العشوائية ، الطرق الاستمثالية.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION GÉNÉRALE

Avec l'adoption et le développement rapide des technologies de communication sans fil ces dernières années, les réseaux sans fil connaissent un essor important et s'imposent aujourd'hui d'une façon indéniable dans notre vie quotidienne: dans nos sociétés, nos organisations, nos maisons, nos voitures… en somme partout. Parmi ces réseaux figurent les réseaux de capteurs (WSNs), les réseaux sans fil maillés (WMNs) et les réseaux véhiculaires (VANETs). Ces réseaux représentent un réel intérêt pour les opérateurs télécoms, les organisations et même les particuliers et constituent actuellement une technologie visant à améliorer notre vie quotidienne, en nous offrant un accès internet haut débit à n'importe quel moment et depuis n'importe quel endroit.

Avec la forte demande de l'internet et l'émergence des services et des applications multimédias tels que la vidéoconférence, le travail de collaboration, l'enseignement à distance et les jeux distribués en ligne, une des contraintes des réseaux sans fil maillés et des réseaux véhiculaires est le problème d'acheminement de données (routage) depuis une source vers un groupe de destinations appelé groupe multicast avec un réel besoin de garantie sur la qualité de service (QoS). Les métriques de QoS les plus importantes sont le coût, le délai, la gigue, la bande passante et le taux de perte des paquets. Ce problème appelé problème de routage multicast avec QoS est un problème d'optimisation NP-complet, il est impossible de le résoudre par les méthodes exactes traditionnelles. Les méthodes approchées généralement inspirées de la nature s'avèrent les plus appropriées pour le résoudre.
Dans ce travail de thèse, nous nous somme basés sur les métaheuristiques pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETs afin de trouver un arbre multicast à moindre coût et satisfaire les contraintes de délai, de gigue, de bande passante et de taux de perte des paquets.

Ce travail de thèse s'articule sur deux contributions majeures. Dans la première contribution, nous avons présenté une amélioration de l'algorithme des chauves-souris BA où nous avons proposé deux variantes ICBBA1 et ICBBA2 pour résoudre le problème de routage multicast avec QoS.

Dans la deuxième contribution, nous avons proposé deux approches hybrides basées sur l'hybridation de l'algorithme des chauves-souris binaire BBA avec l'algorithme évolutionnaire quantique QEA. La première approche, nommée BBAQEA1, est basée sur l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. La deuxième approche, nommée BBAQEA2, est basée sur le remplacement de l'opérateur évolutionnaire quantique de QEA par l'équation d'évolution de BBA.

Notre thèse est organisée en quatre chapitres:

Dans **le premier chapitre**, nous donnons un état de l'art sur les réseaux sans fil maillés et les réseaux véhiculaires, leurs caractéristiques, leurs limites ainsi que leurs applications. Nous décrivons également la qualité de service (QoS) et le routage dans les réseaux sans fil.

Dans **le deuxième chapitre**, nous traitons le problème d'optimisation, les notions et les concepts relatifs à l'optimisation, la classification des problèmes d'optimisation selon plusieurs critères tels que le type de la fonction objectif et le nombre de contraintes. Nous présentons également les méthodes d'optimisation où nous décrivons le principe de base de quelques algorithmes utilisés pour la résolution des problèmes complexes.

Dans **le troisième chapitre**, nous présentons en détail une amélioration de l'algorithme BA où nous proposons deux variantes ICBBA1 et ICBBA2 pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETS. Nous évaluons ensuite leurs performances et nous comparons leurs caractéristiques avec d'autres méthodes d'optimisation existantes dans la littérature.

Dans **le quatrième chapitre**, nous présentons deux approches hybrides basées sur l'hybridation de l'algorithme BA avec l'algorithme QEA (BBAQEA1 et BBAQEA2) proposées pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETS. Nous évaluons ensuite leurs performances et nous comparons leurs caractéristiques avec l'algorithme ICBBA2.

Ce travail ce termine par une conclusion générale et des perspectives que nous avons tracé pour d'éventuelles améliorations et une poursuite de ce travail.

CHAPITRE 1

RÉSEAUX SANS FIL

1.1 Introduction

De nos jours, grâce à l'adoption et l'évolution de la technologie radio et de la communication sans fil, les réseaux sans fil connaissent un essor considérable et s'imposent aujourd'hui d'une façon indéniable dans la vie quotidienne. Un tel essor est dû principalement à la vulgarisation des équipements mobiles offrant plus de flexibilité et moins de frais. Parmi ces réseaux figurent les réseaux de capteurs (WSNs), les réseaux sans fil maillés (WMNs) et les réseaux véhiculaires (VANETs). Ces réseaux sont de plus en plus utilisés et constituent actuellement une technologie visant à améliorer notre vie quotidienne, en nous offrant un accès internet haut débit n'importe où et n'importe quand.

Dans ce chapitre, nous allons présenter les réseaux sans fil, les réseaux sans fil maillés, leurs architectures, leurs caractéristiques, leurs avantages, leurs limites et leurs applications. Nous décrivons également les réseaux véhiculaires, leurs modes de communication, leurs caractéristiques, leurs limites et leurs applications pour enfin terminer avec la qualité de service (QoS) et le routage dans les réseaux sans fil.

1.2 Les réseaux sans fil

Les réseaux sans fil (en anglais Wireless network) sont des réseaux dans lesquels les machines participantes (par exemple ordinateur portable, téléphone mobile, PDA, etc.) peuvent communiquer sans liaison filaire. Ils sont basés sur des liaisons utilisant des ondes radioélectriques (radio ou infrarouge) à la place des câbles habituels (coaxial, paire-torsadée ou fibre optique). Dans ce type de réseaux, les utilisateurs ont la possibilité de se déplacer dans un certain périmètre de couverture géographique sans perdre le signal [1,2].

Les réseaux sans fil peuvent être divisés en deux grandes catégories savoir : les réseaux sans fil basés sur une infrastructure qui utilisent généralement le modèle de communication cellulaire dans lequel les clients sans fil sont connectés à une station de base (point d'accès) et les réseaux sans fil sans infrastructure ou bien les réseaux mobiles ad hoc (MANETs) où la notion de station de base ou point d'accès n'existe pas. Toutes les stations du réseau se connectent les unes aux autres afin de construire un réseau point à point (P2P pour *peer to peer*). Ainsi, chaque machine joue en même temps le rôle de client et le rôle de point d'accès [2,3].

Dans un environnement ad hoc, lorsqu'une donnée est envoyée d'un utilisateur source à un utilisateur destination, il se peut qu'elle transite par plusieurs utilisateurs intermédiaires avant d'arriver à la destination. C'est ce qui s'appelle le multi-saut (multi-hop).

1.3 Les réseaux sans fil maillés

Les réseaux sans fil maillés (WMNs: Wireless Mesh Networks) représentent une nouvelle génération des réseaux sans fil multi-sauts utilisés essentiellement pour interconnecter plusieurs équipements sans fil entre eux en constituant un maillage exempt de tout câblage [4,5]. Ils suscitent de plus en plus un réel intérêt auprès de la communauté R&D, des opérateurs télécom et de fournisseurs de service et ont pour objectif de fournir un accès internet à haut débit aux utilisateurs mobiles et fixes n'importe où et n'importe quand [5,6]. WMNs possèdent les caractéristiques d'auto-configuration, auto-organisation et la possibilité d'établissement et de maintien automatique de la connectivité entre les nœuds dans des zones et environnements dépourvus d'internet [5]. Ces caractéristiques procurent plusieurs avantages tels que les coûts de déploiement, facilité de maintenance du réseau, robustesse… etc.

1.3.1 Architecture des réseaux sans fil maillés

Un réseau sans fil maillé est composé de routeurs maillés (MRs: Mesh Routers) et de clients maillés (MCs: Mesh Clients) qui communiquent entre eux en mode multi-sauts [5]. Les routeurs maillés forment le réseau fédérateur (Backbone), ils ont généralement une mobilité réduite et la contrainte d'énergie ne se pose pas (branchés aux ressources énergétiques) [5,7]. Les MRs jouent le rôle de passerelle (Gateway) ou pont (Bridge), ils sont dotés de plusieurs interfaces sans fil. Les clients maillés sont des utilisateurs du réseau tels que les ordinateurs portables, les assistants personnels PDA, les Pockets PC et les téléphones mobiles. Ils peuvent être stationnaires ou mobiles et sont généralement dotés d'une seule interface sans fil.

Nous distinguons trois différentes architectures des réseaux sans fil maillés selon la fonctionnalité des nœuds [5-8]:

1.3.1.1 Architecte épine dorsale (Backbone)

Dans cette architecture, les MRs forment l'infrastructure du réseau fédérateur (Backbone), ils ont une mobilité réduite et n'ont pas de contraintes d'énergie. Les MRs peuvent être équipés des fonctions de passerelles ou ponts, ce qui leurs permet avec ces fonctions de fournir une infrastructure pour les clients afin de les connecter entre eux, à internet ou à d'autres réseaux sans fil. La figure 1.1 montre un exemple de ce type d'architecture.

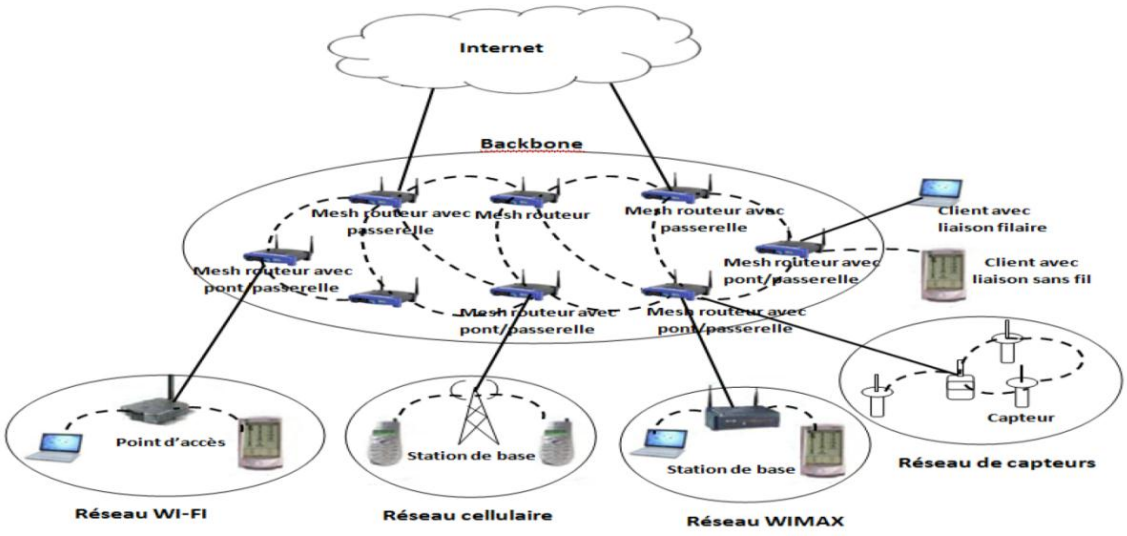


Figure 1.1: Architecture Backbone [5].

1.3.1.2 Architecture client

Dans ce type d'architecture, les nœuds clients forment un réseau maillé auto-organisé et auto-configurable permettant d'assurer des connexions point à point (peer to peer) entre différents utilisateurs. Un MR n'est pas requis dans ce type de réseaux, les MCs sont les seuls constructeurs de la maille, ils sont dotés d'une seule interface radio et peuvent effectuer des fonctions supplémentaires telles que le routage. L'architecture Client est illustrée dans la figure 1.2.

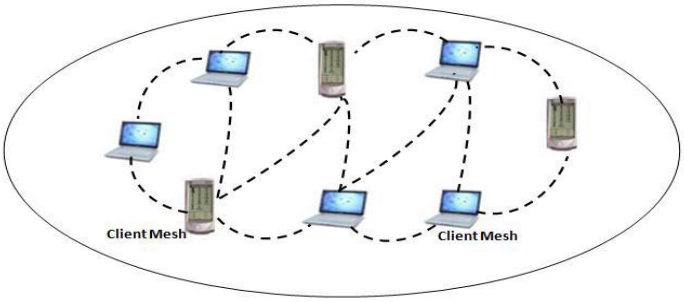


Figure 1.2: Architecture client [5].

1.3.1.3 Architecture hybride

Cette architecture est une combinaison des deux architectures précédentes (Backbone et Client) comme le montre la figure 1.3. Elle permet au MCs d'accéder au réseau sans fil maillé à travers les MRs ou bien directement en communication avec d'autres MCs. La fonctionnalité du routage attribuée au MCs permet de renforcer la connectivité et assurer une large couverture réseau à l'intérieur du WMN.

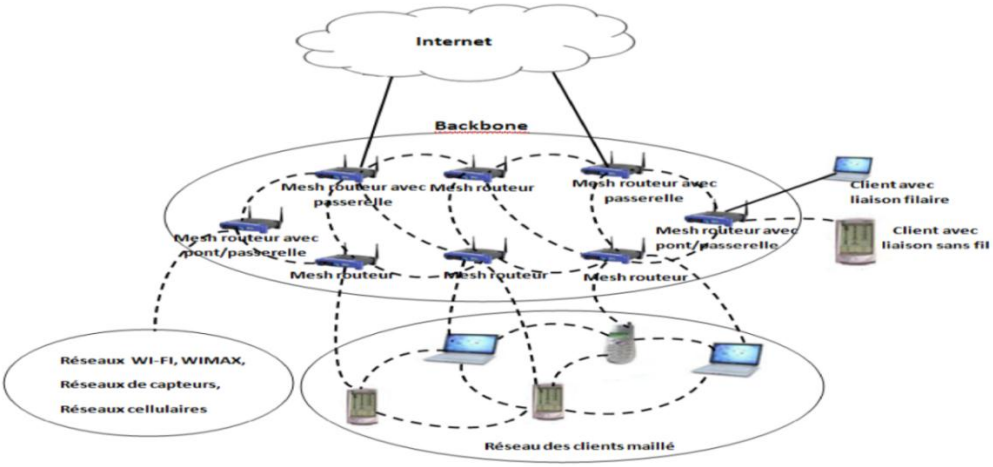


Figure1.3: Architecture Hybride [5].

1.3.2 Caractéristiques des réseaux sans fil maillés

Les réseaux sans fil maillés présentent plusieurs caractéristiques, à savoir [4,5,8,9,10]:

Réseaux multi-sauts

Les WMNs utilisent souvent des sauts multiples pour éviter les obstacles, minimiser la consommation d'énergie ou pour joindre un nœud qui n'est pas dans la portée de communication de l'émetteur.

Interfaces de transmission radio multiples

Les MRs sont dotés de plusieurs interfaces de transmission radios pour renforcer la fonctionnalité du routage et améliorer la capacité du réseau.

Mobilité

La mobilité dans les réseaux sans fil maillés varie selon le type du nœud, les MRs ont une mobilité réduite par contre les MCs peuvent être stationnaires ou de forte mobilité.

Contrainte d'énergie

La contrainte d'énergie dépend du type de nœud. Contrairement au MCs, les MRs n'ont pas de limite en terme d'énergie parce qu'ils sont alimentés directement par des ressources énergétiques.

Compatibilité et interopérabilité

Les WMNs basés sur les standards IEEE 802.11 peuvent supporter les clients conventionnels Wifi et les clients maillés. Ils sont également interopérables avec d'autres technologies sans fil tels que le Zigbee et le WiMax.

Plusieurs fonctionnalités des MRs

En plus de la fonctionnalité principale assurée par les MRs qui est le routage, certains MRs peuvent jouer le rôle d'une passerelle ou d'un pont.

1.3.3 Avantages des réseaux sans fil maillés

Parmi les avantages des réseaux sans fil maillés nous citons [5,6,7,9,10]:

Elimination totale du câblage

Le déploiement d'un WMN ne nécessite aucun câblage ce qui facilite son installation, son utilisation et sa maintenance.

Coût réduit de déploiement

Contrairement aux réseaux filaires où le câblage représente un coût supplémentaire, les WMNs s'affranchissent de ce coût. De plus leur capacité de fonctionner d'une manière autonome minimise toute forme d'intervention.

Forte tolérance aux pannes

Lors de la communication entre deux nœuds distants, si la route devient défaillante, il est possible d'établir une connexion entre ces deux nœuds grâce à l'aspect multi-chemins.

Auto-configuration et auto-organisation

Les WMNs sont capables de s'auto-configurer et de s'auto-réorganiser, donc capables de s'adapter aux changements fréquents de la topologie et de maintenir la connectivité dynamiquement en cas de panne ou de défaillance.

Equilibrage de charges

L'équilibrage de charges est effectué grâce à l'aspect multi-chemins, d'où l'existence de plusieurs routes reliant les nœuds sources avec les nœuds destinataires.

1.3.4 Limites des réseaux sans fil maillés

Parmi les limites et les enjeux des réseaux sans fil maillés, nous citons [5-7]:

La sécurité

Le problème de sécurité dans les WMNs se pose du moment où la confidentialité des données circulantes dans les réseaux n'est pas assurée. Car les transmissions radioélectriques sont sensibles aux interférences et sujettes à l'écoute par un utilisateur mal intentionné. Ainsi, le mécanisme d'authentification et de sécurité fiable s'avère nécessaire.

Le débit

La transmission des données saut par saut et le nombre important d'utilisateurs dans les WMNs entrainent forcément à une dégradation importante du débit, cette dégradation peut même conduire à une perte de connectivité qui est très contraignant.

Dépoilement des routeurs

La planification des routeurs a une grande influence sur les performances des WMNs, un compromis entre la taille de la zone desservie par un routeur et celle de la zone extérieure qu'il dessert doit être optimisé afin d'assurer et garantir la meilleure qualité de service.

Scalabilité (passage à l'échelle)

La taille du réseau a une grande influence sur les performances de celui-ci. La capacité des WMNs se dégrade d'un facteur de $O(\frac{1}{\sqrt{n}})$ à mesure que le nombre d'utilisateurs *n* augmente.

Qualité de service

L'une des contraintes des WMNs est le problème d'acheminement de données avec prise en charge de la qualité de service (QoS). Plusieurs études et travaux de recherche ont été réalisés pour développer des mécanismes qui permettent de garantir la qualité de services des WMNs.

1.3.5 Applications des réseaux sans fil maillés

Les WMNs sont des réseaux robustes, peu coûteux et s'adaptent aussi bien aux milieux urbains qu'aux milieux ruraux. Vu leur importante, ces réseaux s'ouvrent aujourd'hui à de nombreuses applications concrètes. Parmi ces applications nous citons [5,6,8]:

1.3.5.1 Réseaux domestiques (Broad Home Networking)

Les réseaux domestiques sont basés principalement sur la technologie IEEE 802.11 qui présente certaines limites. La présence des zones non couvertes (mortes) dans la maison et les communications qui doivent passer impérativement par les points d'accès (APs) sont quelques exemples de ces limites. L'introduction des WMNs avec l'utilisation des MRs au lieu des APs comme infrastructure de communication devient une solution prometteuse capable de faire face à ces limites.

1.3.5.2 Réseaux communautaires ou de voisinage

Dans les réseaux de communauté, les communications entre maisons sont basées sur internet. Ce qui fait que n'importe quelle communication entre deux utilisateurs doit passer obligatoirement par internet. Ceci réduit d'une manière significative l'utilisation des ressources du réseau. L'introduction des WMNs peut faire face à ce problème par l'installation des MRs dans chaque maison. Dans ce type de réseaux, les communications sont assurées par le Backbone.

1.3.5.3 Réseaux de zone métropolitaine

Les communications entre les utilisateurs dans les WMNs ne se basent pas sur des liaisons filaires. L'utilisation des liaisons sans fil dans les réseaux métropolitaines représente une alternative économique, particulièrement dans les régions reculées. Grâce à l'utilisation du principe multi-sauts entre les utilisateurs, une zone de service plus large qu'une maison, bâtiment ou entreprise est offerte. La mise en échelle devient un facteur important à prendre en considération dans ce type d'applications.

1.3.5.4 réseaux véhiculaires (Transportation System)

Dans les réseaux véhiculaires, les liaisons filaires entre les stations de base posent un problème lors de la communication entre les véhicules ou avec leur environnement. L'introduction des WMNs avec l'utilisation des MRs au lieu des stations de base filaires

permet de résoudre ce problème et offre une robustesse et une flexibilité dans les communications inter-véhiculaires.

1.4 Les réseaux véhiculaires

Les réseaux véhiculaires (VANETs : Vehicular Ad hoc NETworks) sont une particularité des réseaux MANETs où les nœuds mobiles sont des véhicules intelligents (figure 1.4) équipés de calculateurs, de carte réseaux et de capteurs. Ils permettent d'établir des communications entre les véhicules (pour échanger les informations sur le trafic par exemple) ou avec les stations de base placées tout au long des routes (pour demander des informations ou accéder à internet…) [11,12].

Les VANETs sont devenues l'une des technologies sans fil les plus pertinentes. Ils regroupent deux classes d'applications, à savoir les applications qui permettent l'implémentation des systèmes de transport intelligents (ITS : Intelligent Transport Systems) et celles liées au confort ou à la sécurité routière du conducteur et des passagers [12].

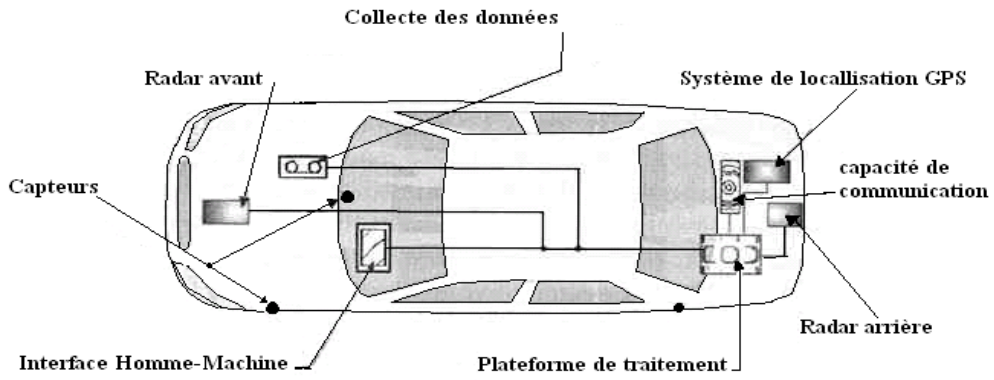


Figure 1.4: Véhicule intelligent [12].

1.4.1 Modes de communication dans les réseaux véhiculaires

Nous pouvons distinguer trois modes de communication dans les réseaux véhiculaires, à savoir [12-14]:

1.4.1.1 Communication de véhicule à véhicule (V2V)

Dans ce mode de communication, un réseau de véhicules est vu comme un cas particulier du réseau MANET (Mobile Ad Hoc Network) où les contraintes d'énergie, de mémoire et de capacité sont relaxées et où le modèle de mobilité n'est pas aléatoire mais prévisible avec une très forte mobilité. Aucune infrastructure n'est utilisée, aucune installation n'est nécessaire sur

les routes et tous les véhicules sont équipés de calculateurs, de carte réseaux et de capteurs pour communiquer directement entre eux n'importe où, que se soit sur les autoroutes, des routes de montagnes ou des routes urbaines. Les communications V2V sont utilisées pour le transfert des informations concernant les services liés à la sécurité routière, dans le scénario de diffusion d'alertes (freinage d'urgence, collision, ralentissement…) ou pour la conduite coopérative.

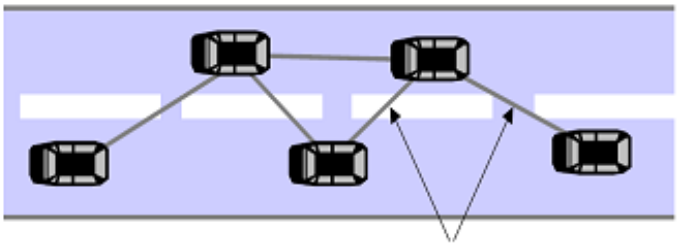


Figure 1.5: Communication véhicule à véhicule

1.4.1.2 Communication de véhicule avec utilisation d'infrastructures

Dans ce mode de communication, on ne se concentre pas seulement sur des simples systèmes de communications inter véhicules mais aussi ceux qui utilisent des stations de bases ou points d'infrastructure RSU (Road Side Unit). Ce mode de communication repose sur le modèle client/serveur où les véhicules sont les clients et les stations installées le long de la route sont les serveurs. Ces serveurs sont connectés entre eux via une interface filaire ou sans fil. Toute communication doit passer par eux. Ils permettent d'offrir aux utilisateurs une meilleure utilisation des ressources partagées et plusieurs services concernant le trafic, accès à internet, échange de données de voiture-à-domicile et même la communication de voiture-à-garage pour le diagnostic distant.

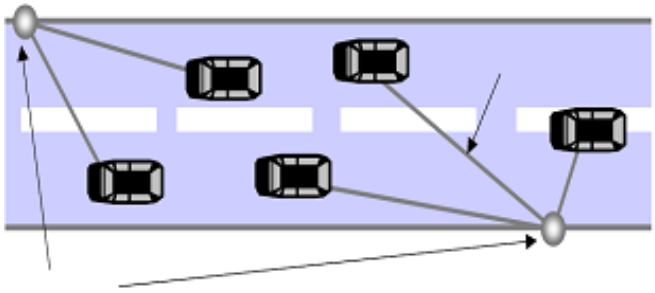


Figure 1.6: Communication véhicule à station de base.

1.4.1.3 Communication hybride

Ce mode de communication très intéressant résulte de la combinaison des communications véhicules à véhicules avec les communications de véhicules avec utilisation d'infrastructures. En effet, les portées des infrastructures étant limitées, l'utilisation des véhicules comme relais permet d'étendre cette distance. Dans un but économique et afin d'éviter la multiplication des stations de bases à chaque coin de rue, l'utilisation des sauts par véhicules intermédiaires prend tout son importance.

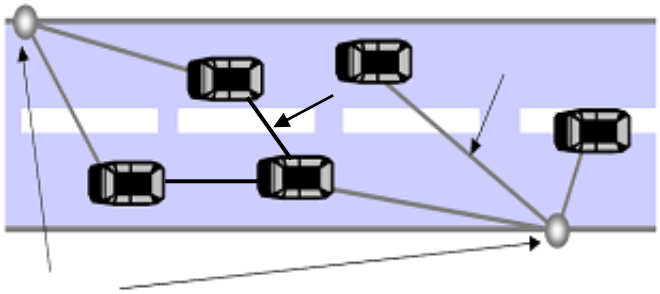


Figure 1.7: Communication hybride.

1.4.2 Caractéristiques des réseaux véhiculaires

Les réseaux véhiculaires ont des caractéristiques spécifiques qui les distinguent des réseaux Ad Hoc, à savoir [12,14,15]:

1.4.2.1 Collecte d'informations et perception de l'environnement proche

Différents capteurs sont utilisés pour la collecte d'informations (caméras, capteurs de pollution, capteurs de pluies, capteurs de l'état de la route et de voiture, etc…). Ces informations collectées permettent au conducteur à bord de son véhicule de disposer d'une meilleure visibilité pour pouvoir réagir d'une manière adéquate aux changements de son environnement proche.

1.4.2.2 Capacité de traitement, d'énergie et de communication

Contrairement au contexte des réseaux sans fil traditionnels tels que les réseaux de capteurs et les MANETs où la contrainte d'énergie représente un défi pour les chercheurs, les éléments du réseau VANET n'ont pas de limite en terme d'énergie et disposent d'une grande capacité de traitement et peuvent avoir plusieurs interfaces de communication (WIFI, Bluetooth et autres). Grâce aux Nouvelles Technologies de l'Information et de la Communication (NTIC), le conducteur peut prendre une décision à l'aide des traitements et des interprétations des informations collectées.

1.4.2.3 Environnement de déplacement et modèle de mobilité

Contrairement à d'autres réseaux sans fil qui ont un environnement bien spécifique selon leur fonctionnement souvent limité à des espaces ouverts ou indoor (comme le cas d'une conférence ou à l'intérieur d'un bâtiment), les réseaux véhiculaires sont liés aux structures des routes (intersections, panneaux de signalisation, etc…) et aux stations de base routières (infrastructures) que se soit dans les autoroutes ou au sein d'une zone métropolitaine. Les contraintes imposées par ce type d'environnement affectent considérablement le modèle de mobilité et la qualité des transmissions radio. En outre la mobilité est un facteur lié directement au conducteur du véhicule.

1.4.2.4 Forte mobilité, topologie du réseau et connectivité

Les réseaux véhiculaires se distinguent également des réseaux sans fil ordinaires par la forte mobilité des nœuds (véhicules), due à la vitesse des voitures qui est très importante dans les autoroutes. Par conséquent, les déplacements des véhicules sont liés à la volonté des conducteurs et entrainent des scénarios très dynamiques. Un véhicule peut rejoindre ou quitter le réseau en un temps très court, ce qui rend les changements de topologie très fréquent. De plus, des problèmes peuvent apparaître quand le système IVC (Inter Vehicle Communication) n'est pas équipé dans la majorité des véhicules.

1.4.2.5 Modèle de communication

Les réseaux véhiculaires ont été conçus principalement pour les applications de la prévention et la sécurité routière. Dans ce type d'applications, les communications se font exclusivement par la diffusion des messages d'une source vers une ou plusieurs destinataires. Néanmoins, les véhicules sont concernés par la diffusion d'informations en fonction de leurs positions géographiques et leurs degrés d'implication dans l'évènement déclenché. Le modèle de transmission en broadcast ou en multicast est largement présent dans les réseaux véhiculaires, ce qui entraine une charge du réseau très importante.

1.4.3 Applications des réseaux véhiculaires

Les principales applications des réseaux VANET peuvent être classées en trois grandes catégories [12,14]:

1.4.3.1 Applications de sécurité du trafic routier

Les applications de sécurité routière se basent généralement sur une diffusion, périodique ou non, de messages informatifs permettant aux conducteurs d'avoir une connaissance de l'état de la route et des véhicules voisins. Elles permettent de prévenir les collisions et les travaux sur les routes, de détecter les obstacles (fixes ou mobiles) et de distribuer les informations météorologiques par envoi de messages d'alerte. A titre d'exemple, alerter un conducteur en cas d'accidents permet d'avertir les véhicules qui se dirigent vers le lieu de l'accident que les conditions de circulations se trouvent modifiées et qu'il est nécessaire de redoubler de vigilance.

1.4.3.2 Application pour l'optimisation du trafic et aide dans la conduite

Les applications de gestion du trafic routier permettent d'optimiser le trafic routier et améliorer les conditions de circulation dans le but de réduire les embouteillages et les risques d'accidents. Le trafic routier peut être grandement optimisé et amélioré grâce à la collecte et au partage de données collectées par les véhicules, ce qui devient un support technique pour les conducteurs. Un véhicule peut, par exemple, être averti en cas d'un ralentissement anormal (bouchon, embouteillage, éboulement de rochers ou travaux).

1.4.3.3 Applications au confort du conducteur et des passagers

Les applications de confort ou de divertissement permettent d'améliorer le confort des conducteurs et des passagers. Ce confort est illustré par l'accès à internet, la messagerie, le chat inter-véhicule, etc. Les passagers dans la voiture peuvent jouer en réseaux, télécharger des fichiers MP3, envoyer des cartes à des amis, etc.

1.4.4 Limites et défis des réseaux véhiculaires

Parmi les défis des réseaux véhiculaires nous citons [12,14,15,16,17]:

1.4.4.1 Sécurité

La sécurité est un défi majeur ayant un grand impact sur le déploiement des réseaux véhiculaires. En raison de la sensibilité des domaines d'utilisation des réseaux véhiculaires, une intrusion d'un véhicule malicieux aurait des conséquences graves sur l'ensemble des véhicules interconnectés. Ainsi, le mécanisme d'authentification, confidentialité et de sécurité fiable s'avère nécessaire. Beaucoup de travaux de recherche ont été réalisés pour développer

un mécanisme de sécurité instituant les relations de confiance entre les nœuds communicants et garantissant le contrôle d'accès aux services.

1.4.4.2 L'accès au canal

Le rôle des mécanismes de gestion du canal de transmission radio est d'offrir des communications fiables et un partage équitable du médium de communication. Pour atteindre cet objectif dans les réseaux véhiculaires où les communications se font en environnement externe défavorable en raison de la multitude d'obstacles, il est nécessaire de concevoir des méthodes qui permettent de faire face aux problèmes d'interférences radio, problèmes de propagation à multi-trajets des ondes ainsi que les irrégularités électromagnétiques.

1.4.4.3 Localisation des véhicules

Les véhicules du réseau doivent être informés de la position d'un véhicule si ce dernier doit être localisé (dans le cas d'un accident par exemple). Le problème est que tous les véhicules ne sont pas équipés d'un système de repérage par satellite (GPS). Pour cette raison, un mécanisme de localisation sans utilisation de GPS est nécessaire.

1.4.4.4 Routage

Le routage dans les réseaux véhiculaires est un problème très difficile à gérer et un axe de recherche pour beaucoup de chercheurs. Pour que les véhicules puissent communiquer entre eux, un protocole de routage doit être défini pour répondre aux problèmes de connectivités intermittentes et du partitionnement du réseau qui empêche la propagation de l'information. En effet quand les terminaux ne sont pas à une portée de transmission radio directe, le routage est exigé pour établir la communication entre les véhicules.

1.4.4.5 Qualité de service

La qualité de service (QoS) dans les réseaux véhiculaires se dégrade avec l'augmentation du nombre de véhicules ainsi que leur forte mobilité. La principale contrainte des applications de sécurité est la latence. La validité des informations étant limité dans le temps, l'information doit parvenir aux véhicules destinataires dans des délais courts pour être considérée pertinente. Plusieurs études et travaux de recherche ont été réalisés pour développer des mécanismes qui permettent de garantir la qualité de services des réseaux véhiculaires.

1.5 Qualité de service et routage dans les réseaux sans fil

1.5.1 Qualité de service

Dans les réseaux sans fil, l'objectif principal de la qualité de service est d'atteindre un meilleur comportement de la communication, afin que le contenu de cette dernière soit acheminé correctement d'une source à sa destination, et les ressources du réseau sont utilisées d'une façon fiable et optimale [18,20].

La qualité de service (QoS : Quality of Service) est un terme adaptatif à plusieurs domaines. Généralement, il peut être défini comme le degré de satisfaction d'un utilisateur des services fournis par un système de communication et les aspects concernant ces services: leur accessibilité, leur continuité, leur maintenabilité, leur disponibilité, leur fiabilité, etc [18-20].

La QoS est considérée comme la capacité d'un élément du réseau (routeur, nœud ou application) de fournir un niveau de garantie pour un problème d'acheminement de données [19-21].

La QoS est définie dans RFC 2386 [19,22] comme un ensemble d'exigences et de besoins à assurer par le réseau pour la transmission de données d'une source à sa destination.

1.5.1.1 Métriques de la qualité de service

Les métriques de qualité de service peuvent être traduites en un ensemble d'attributs et de services pré-spécifiés et mesurables en termes de [19,20,22]:

Délai de bout en bout (Delay)

Le délai de bout en bout est la durée de temps prise pour transférer un paquet à partir de sa source d'envoie vers sa destination de réception. Ce délai pour chaque paquet est calculé suivant cette formule :

Délai de bout en bout = instant de réception du paquet – instant d'émission du paquet.

Variance de délai (Jitter)

La variance de délai ou gigue (en anglais jitter) est la différence entre le délai de bout en bout max et le délai de bout en bout min.

Gigue = Délai de bout en bout max - Délai de bout en bout min.

En général, si la gigue est faible cela peut vouloir dire que les nœuds sont proches les uns des autres donc la connectivité du réseau est forte. A l'inverse, si la valeur de la gigue est grande cela veut dire que les vitesses des nœuds sont très variables et que la connectivité du réseau est faible.

Bande passante (Bandwidth)

La bande passante est le volume de Méga bit de données reçus par seconde. Périodiquement on recalcule ce nombre en utilisant la formule suivante :

Bande passante = taille du message reçu en bits/délai de bout e bout pour ce paquet (secondes).

Plus la bande passante est grande, plus le réseau est chargé et plus il y a des pertes de paquets.

Pertes de paquets (Packet loss)

Le taux de perte des paquets est le nombre de paquets perdus par rapport au nombre total de paquets envoyés.

Taux de perte = nombre de paquets perdus/nombre de paquets émis.

Dans le nombre des paquets perdus, sont inclus les paquets de données et les paquets de contrôle qui servent à trouver les routes et à les maintenir. Ce taux est recalculé à chaque émission ou perte d'un paquet dans le réseau.

1.5.2 Routage

Le routage dans les réseaux sans fil est la brique technologique fondamentale permettant d'assurer la connectivité du réseau. C'est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Dans un tel réseau, les nœuds radio ne sont pas nécessairement à portée directe. Un paquet peut donc devoir être relayé par des nœuds intermédiaires pour atteindre sa destination finale [23,24].

1.5.2.1 Classification du routage

Le routage peut être classé de différentes manières et selon plusieurs critères. Il peut être classé selon le mode de communication (unicast, multicast ou broadcast), selon la prise en charge ou non de la qualité de service ainsi que selon l'architecture (uniforme ou non uniforme)

a) Classification du routage selon le mode de communication

Nous distinguons trois types de routage pour la transmission de données dans les réseaux sans fil [23-26]:

a.1) Routage Unicast

Le terme unicast définit une connexion réseau point à point. On entend par unicast, le fait de communiquer entre deux machines identifiées chacune par une adresse réseau unique. Les

paquets de données sont routés sur le réseau suivant l'adresse du destinataire. Ces paquets transitent d'un nœud source vers un nœud destinataire, seul le destinataire intercepte et décode le paquet qui lui est adressé.

a.2) Routage multicast

On entend par multicast, le fait de communiquer simultanément avec un groupe d'utilisateurs identifié par une adresse spécifique (adresse de groupe). Son avantage par rapport au mode classique unicast devient évident quand on veut diffuser de la vidéo. Les paquets de données sont routés sur le réseau selon l'adresse des destinataires encapsulée dans la trame transmise. Seuls les destinataires interceptent et décodent les paquets qui leurs sont adressés.

a.3) Routage broadcast (diffusion)

Le broadcast est un terme anglais définissant une diffusion de données depuis une source unique à tous les utilisateurs présents autour de cette dernière. Contrairement à une communication Point à Point ou multicast, il est possible d'adresser des paquets de données à un ensemble de machines d'un même réseau uniquement par des adresses spécifiques qui seront interceptées par toutes les machines du réseau ou sous réseau**.** La figure 1.8 illustre les différents types de routage

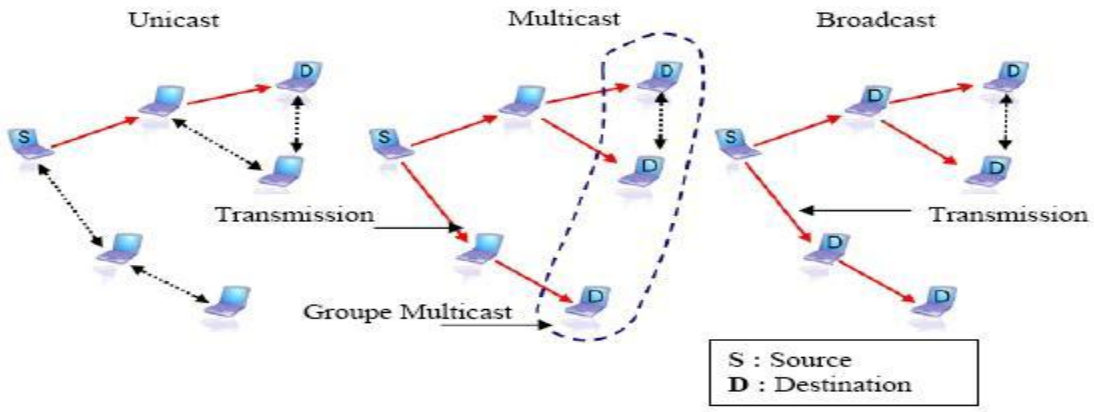


Figure 1.8: Types de routage [26].

b) Classification du routage selon la prise en charge de la qualité de service

Un autre critère de classification du routage est la prise en charge ou non de la qualité de service, nous trouvons deux types de routage dans cette classification, à savoir le routage avec prise en charge de la qualité de service et le routage sans prise en charge de la qualité de service.

b.1) Routage avec prise en charge de la qualité de service

Dans les réseaux sans fil, pour le transport d'un trafic multimédia (voix et vidéo), un ensemble de besoins de QoS doit être assuré pour le bon acheminement de données. Par exemple, le délai de bout en bout d'un paquet doit être limité autrement le paquet est inutile.

b.2) Routage sans prise en charge de la qualité de service

Dans ce type de routage, l'acheminement de données se fait sans tenir compte de métriques de la qualité de service. Comme dans les applications non temps réel (messagerie et transfert de fichier).

c) Classification du routage selon l'architecture

Ce critère divise le routage en deux catégories [20,23]

c.1) Routage uniforme

Le routage uniforme « à plat » considère que tous les nœuds sont égaux, dans le même niveau hiérarchique et possèdent, ainsi, les mêmes rôles et fonctions. Par conséquent, aucune hiérarchie n'est définie entre les nœuds du réseau. La décision d'un nœud de router des paquets dépendra de sa position.

c.2) Routage non uniforme

Le routage non uniforme « hiérarchiques » tente de limiter la complexité du routage en réduisant le nombre de nœuds qui contribuent à la détermination des routes, ils fonctionnent en attribuant aux nœuds des rôles qui varient de l'un à l'autre. Certains nœuds sont élus pour accomplir des tâches bien particulières qui conduisent à une vision en plusieurs niveaux de la topologie du réseau afin de faciliter l'équilibrage de la charge et de mieux la gérer, ce qui conduit à une meilleure qualité de service.

1.6 Conclusion

Dans ce chapitre, nous avons présenté les réseaux sans fil maillés et les réseaux véhiculaires, leurs caractéristiques, leurs limites ainsi que leurs applications. Nous avons également décrit la qualité de service (QoS) et le routage dans les réseaux sans fil.

Avec l'émergence des services et des applications multimédias temps réel, une des contraintes des réseaux sans fil maillés et des réseaux véhiculaires est le problème d'acheminement de

données depuis une source vers un groupe de destinations appelé groupe multicast. Ce problème appelé problème de routage multicast avec QoS est un problème NP-Complet, il est impossible de le résoudre par les méthodes exactes traditionnelles. Les méthodes approchées s'avèrent les plus appropriées pour le résoudre.

Dans le chapitre suivant, nous présenterons les problèmes d'optimisation et leur classification ainsi que les méthodes d'optimisations exactes et approchées où nous allons décrire le principe de base de quelques algorithmes utilisés pour la résolution des problèmes complexes.

CHAPITRE 2

MÉTHODES D'OPTIMISATION

2.1 Introduction

L'optimisation occupe une place très importante dans beaucoup de domaines, comme dans la recherche opérationnelle, l'intelligence artificielle, la biologie, les mathématiques et l'informatique. Un grand nombre de problèmes peuvent être définis et décrits sous forme de problèmes d'optimisation. Généralement ces problèmes appartiennent à la classe des problèmes NP-difficiles qui ne possèdent pas de solution optimale pour l'ensemble de données. Le problème de routage multicast avec QoS est un problème d'optimisation classé NP-difficile, il est impossible de le résoudre par les méthodes exactes traditionnelles. Pour résoudre ce problème, les études convergent vers l'utilisation de méthodes approchées généralement inspirées de la nature. Les méthodes d'optimisation approchées peuvent être classées en deux catégories à savoir les heuristiques et les métaheuristiques,

Dans ce chapitre, nous allons présenter le problème d'optimisation, les notions et les concepts relatifs à l'optimisation, la classification des problèmes d'optimisation selon plusieurs critères : complexité des problèmes, nature des problèmes, le nombre d'optimums, le type de la fonction objectif et le nombre de contraintes. Nous présentons ensuite les méthodes d'optimisation où nous décrivons le principe de base de quelques algorithmes utilisés pour la résolution des problèmes complexes.

2.2 Problème d'optimisation

Un problème d'optimisation noté *P(X, f),* se définit comme la recherche, parmi un ensemble de solutions X réalisables ou admissibles (appelé aussi espace de décision ou espace de recherche), de la solution qui minimise ou maximise la fonction objectif f [27-29].

Dans le cas d'un problème de minimisation, résoudre le problème revient à trouver une solution x^* telle que $f(x) \geq f(x^*)$, pour tout élément *x* dans X.

Dans le cas d'un problème de maximisation, résoudre le problème revient à trouver une solution x^* telle que $f(x) \leq f(x^*)$, pour tout élément *x* dans X.

Mathématiquement, un problème d'optimisation est généralement représenté sous la forme suivante [30,31]:

$$\text{PO} \begin{cases} \text{Minimiser/Maximiser } f(x) \text{ (fonction à optimiser)} \\ \text{Sous les contraintes :} \\ g(x) \leq 0 \text{ (n contraintes d'inégalité)} \\ h(x) = 0 \text{ (m contraintes d'égalité)} \\ x_I \leq x \leq x_S \end{cases} \tag{2.1}$$

Où $x = (x_1, \ldots, x_d)$ représente le vecteur des variables de décision ou variables indépendantes, $g(x)$ et $h(x)$ représentent respectivement les contraintes d'inégalité et d'égalité, x_I et x_S sont respectivement les bornes inférieures et supérieures du domaine de recherche des variables et $f: D \subset R^d \rightarrow R$ est la fonction objectif et D est le domaine réalisable.

Il est possible de passer d'un problème de minimisation à un problème de maximisation et vice versa grâce aux propriétés suivantes [32]:

$$min_{x \in D} f(x) = max_{x \in D}(-f(x)) \tag{2.2}$$

$$max_{x \in D} f(x) = min_{x \in D}(-f(x)) \tag{2.3}$$

Tout point $x \in R^d$ appartenant à *D* est appelé point réalisable.

2.3 Notions et concepts relatifs à l'optimisation

Voisinage

Le voisinage de *x*, noté *V(x)*, est un sous ensemble de solutions réalisables de X atteignables à partir d'une transformation donnée de *x*.
$x^* \in V(x)$ est dite voisine de *x*.

Optimum

L'optimum est le point où la fonction objectif atteint son minimum ou son maximum [29,30].

Optimum local

On dit qu'une solution x^* est un optimum local de la fonction objectif f sur $D \subset R^d$ si on a :

$$\forall x \in V(x^*), \begin{cases} f(x^*) \leq f(x) \text{ dans le cas de minimisation} \\ f(x^*) \geq f(x) \text{ dans le cas de maximisation} \end{cases} \quad (2.4)$$

Avec $x^* \neq x$ et $V(x^*)$ représente l'ensemble des solutions voisines de x^*.

Optimum global

On dit qu'une solution x^* est un optimum global de la fonction objectif f sur $D \subset R^d$ si on a :

$$\forall x \in D, \begin{cases} f(x^*) \leq f(x) \text{ dans le cas de minimisation} \\ f(x^*) \geq f(x) \text{ dans le cas de maximisation} \end{cases} \quad (2.5)$$

Avec $x^* \neq x$.

La notion d'optimum local et optimum global est illustrée dans la figure 2.1

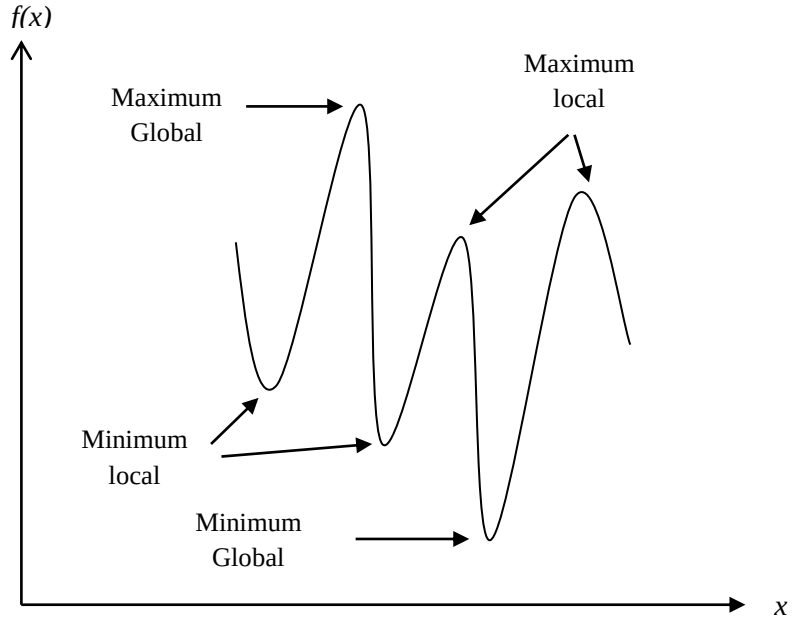


Figure 2.1: Optimum local et global.

Intensification et diversification

La recherche de l'optimum d'une fonction objectif est réalisée généralement à l'aide de deux opérateurs de recherche fondamentaux à savoir : l'intensification et la diversification [29,33].

Intensification

L'intensification ou l'exploitation permet d'affiner et d'améliorer la valeur d'une solution trouvée dans un certain voisinage en augmentant la précision de l'optimum.

Diversification

La diversification ou l'exploration permet une localisation imprécise de l'optimum global dans un espace de recherche de plus grande taille de telle sorte que la recherche ne soit pas concentrée sur une zone de l'espace de recherche particulière.

Fonction objectif

La fonction objectif représente le but à réaliser ou à atteindre (minimisation ou maximisation de la fonction). Elle définit un espace de solutions potentielles au problème [30].
$min_{x \in D}\ f(x) \Rightarrow$ la fonction objectif $f(x)$ est appelée fonction coût.
$max_{x \in D}\ f(x) \Rightarrow$ la fonction objectif $f(x)$ est appelée fitness.

Espace d'état

Appelé aussi domaine de recherche est l'ensemble des domaines de définition des différentes variables du problème d'optimisation [30].

Contraintes

Représentent des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité utilisées généralement pour limiter l'espace de recherche.

2.4 Classification des problèmes d'optimisation

La classification des problèmes d'optimisation est un élément crucial pour leur résolution, ces problèmes peuvent être classés selon différents critères (comme le montre la Figure 2.1) : complexité des problèmes, nature des problèmes, le nombre d'optimums, le type de la fonction objectif et le nombre de contraintes [29,30,33,34,35].

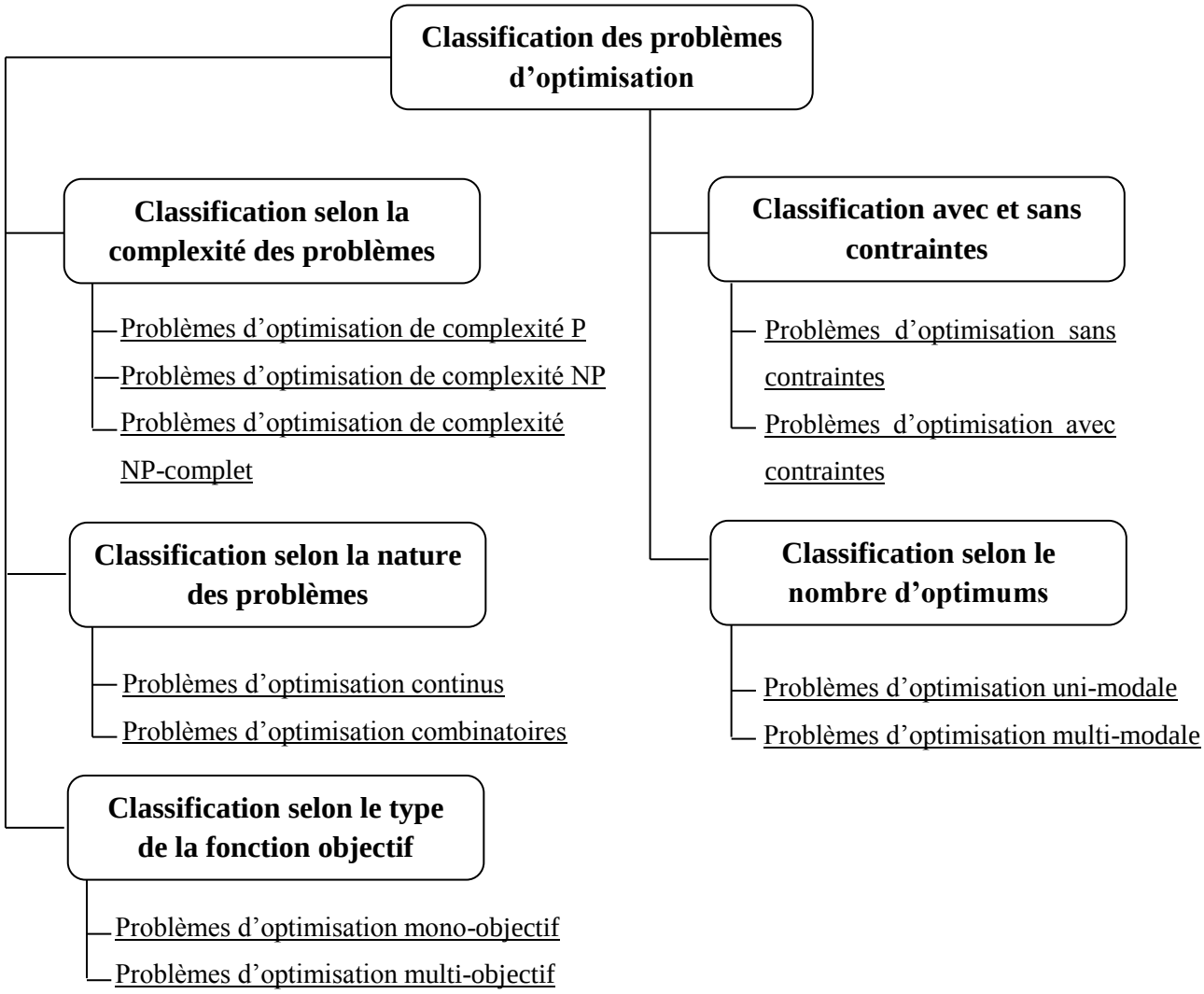


Figure 2.2: Classification des problèmes d'optimisation.

2.4.1 Classification selon la complexité des problèmes d'optimisation

Nous trouvons généralement trois classes importantes des problèmes d'optimisation selon la complexité des algorithmes utilisés pour la résolution des problèmes

2.4.1.1 Problèmes d'optimisation de complexité P

C'est un ensemble de problèmes polynomiaux qui peuvent être résolus par un algorithme déterministe de complexité polynomiale. Ces problèmes sont de complexité $O(n^k)$ et sont relativement faciles à résoudre.

2.4.1.2 Problèmes d'optimisation de complexité NP

C'est un ensemble de problèmes non déterministes polynomiaux qui peuvent être résolus par une machine de Turing non-déterministe en temps polynomial. Ils peuvent être également résolus par un algorithme polynomial énumérant l'ensemble des solutions possibles.

2.4.1.3 Problèmes d'optimisation de complexité NP-complet

C'est un sous-ensemble des problèmes NP qui contient les problèmes les plus difficiles de NP. Un problème est NP-complet si et seulement s'il vérifie ces deux conditions :

$$\begin{cases} \mathrm{P} \in \mathrm{NP}. \\ \forall\, \mathrm{P}' \in \mathrm{NP} - \text{complet}, \mathrm{P}' \text{ se réduit à P par un algorithme polynomial.} \end{cases}$$

2.4.2 Classification selon la nature des problèmes

Les problèmes d'optimisation peuvent être classés en fonction de la nature des espaces dans lesquels les variables de décision prennent leurs valeurs, nous distinguons deux classes, à savoir : les problèmes d'optimisation continus et problèmes d'optimisation combinatoires (discrets).

2.4.2.1 Problèmes d'optimisation continus (numériques)

Un problème d'optimisation continu est un problème d'optimisation dont les éléments d'entrées du problème changent en fonction du temps. Dans ce type de problèmes, le cas linéaire (qui relève notamment de la programmation linéaire) est séparé sommairement du cas non-linéaire, où le problème d'optimisation est difficile. La fonction objectif varie avec le temps :

$f(x) = f_t(x)$, avec t le temps où la fonction objectif est évaluée.

2.4.2.2 Problèmes d'optimisation combinatoires (discrets)

Un problème d'optimisation combinatoire est un problème dans lequel les variables de décisions sont limitées à un ensemble de valeurs discrètes ou dénombrables. Ce type de problème est souvent facile à définir mais généralement difficile à résoudre. En effet la plupart des problèmes combinatoires appartiennent à la classe des problèmes NP-complets.

2.4.3 Classification selon le type de la fonction objectif

2.4.3.1 Problèmes d'optimisation mono-objectif

Un problème d'optimisation mono-objectif est défini par un ensemble de variables, un ensemble de contraintes et une seule fonction objectif (un seul objectif). La solution à ce problème est un seul point.

2.4.3.2 Problèmes d'optimisation multi-objectif

Un problème d'optimisation multi-objectif est défini par un ensemble de variables, un ensemble de contraintes et un ensemble de fonctions objectif qui nécessite la résolution de plusieurs problèmes simultanément, souvent contradictoires. La solution à ce problème est un ensemble de points connus comme l'ensemble Pareto-Optimal.

2.4.4 Classification selon les contraintes

Un autre critère de classification des problèmes d'optimisation est l'utilisation ou non des contraintes sur l'espace d'état que les variables doivent satisfaire, nous trouvons deux types de problèmes dans cette classification, à savoir les problèmes d'optimisation sans contraintes et les problèmes d'optimisation avec contraintes.

2.4.4.1 Problèmes d'optimisation sans contraintes

Un problème d'optimisation sans contrainte est un problème où la fonction objectif non linéaire est définie sur un ensemble de valeurs réelles sans contraintes.

2.4.4.2 Problèmes d'optimisation avec contraintes

Un problème d'optimisation avec contraintes est un problème où la fonction objectif non linéaire est définie sur un ensemble de valeurs réelles limitées. En général, les problèmes d'optimisation sont des problèmes d'optimisation avec contraintes, ces contraintes peuvent avoir des formes mathématiques ou symboliques.

2.4.5 Classification selon le nombre d'optimums

2.4.5.1 Problèmes d'optimisation uni-modale

Un problème d'optimisation uni-modale est un problème dans lequel l'espace de recherche ne contient qu'un optimum global (un minimum global dans le cas d'une minimisation ou un maximum global dans le cas d'une maximisation).

2.4.5.2 Problèmes d'optimisation multimodale

Un problème d'optimisation multimodale est un problème dans lequel l'espace de recherche contient plusieurs optimums (locaux et globaux). Ce problème évite les optimums locaux et permet la localisation de plusieurs optimums globaux en même temps.

2.5 Les méthodes d'optimisation

Vu l'importance des problèmes d'optimisation, plusieurs méthodes de résolution ont été développées dans la littérature en intelligence artificielle et en recherche opérationnelle. En général, les méthodes de résolution suivent quatre approches différentes pour la recherche d'une solution : l'approche de construction, l'approche de relaxation, l'approche de voisinage et l'approche d'évolution. Dans le domaine de l'optimisation, nous trouvons deux grandes catégories de méthodes d'optimisation (Figure 2.3): les méthodes exactes (complètes) qui garantissent la complétude de la solution et les méthodes approchées (incomplètes) qui perdent la complétude afin de gagner en temps d'exécution.

2.5.1 Les méthodes exactes

Les méthodes d'optimisation exactes cherchent à trouver de manière certaine la solution optimale en énumérant et examinant, de manière implicite, l'ensemble des solutions de l'espace de recherche.

Elles sont utilisées généralement pour résoudre des problèmes de taille raisonnable et permettent de trouver des solutions optimales néanmoins, le temps de calcul nécessaire pour trouver une solution peut devenir très excessif et augmente exponentiellement avec la taille du problème et le nombre de fonctions objectif à optimiser. À titre d'exemple, nous citons la méthode de séparation et évaluation et la méthode de programmation dynamique.

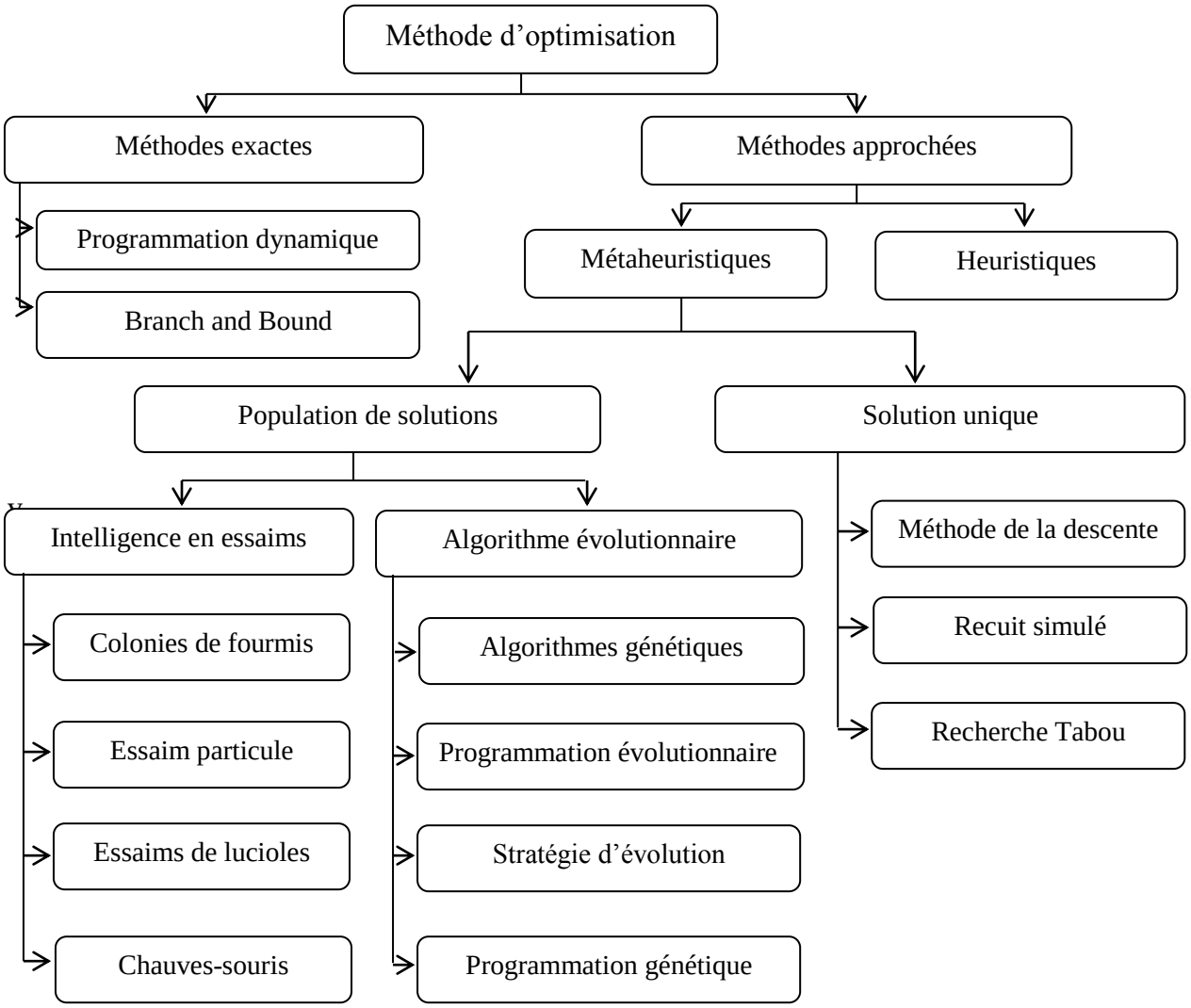


Figure 2.3: Classification des méthodes d'optimisation.

2.5.1.1 La méthode de séparation et évaluation (Branch and Bound)

La méthode de séparation et évaluation, appelée en anglais Branch and Bound Method [36], est une méthode de résolution exacte utilisée pour la résolution des problèmes d'optimisation combinatoire [27]. Cette méthode est basée sur la recherche arborescente d'une solution optimale par séparation et évaluation [27]. Tout d'abord, l'ensemble des solutions est séparé en sous-ensembles plus petits. Ensuite, une évaluation optimiste est appliquée pour majorer les sous-ensembles et choisir une solution potentiellement bonne et meilleure que la solution courante. La méthode Branch and Bound est généralement coûteuse en temps de calcul et ne peut donc s'appliquer qu'à des problèmes spécifiques.

2.5.1.2 La programmation dynamique

La programmation dynamique est une méthode récursive de résolution exacte, utilisée pour résoudre un grand nombre de problèmes d'optimisation [27,37]. Cette méthode repose sur la division récursive d'un problème en sous-ensembles de problèmes simples. Elle est basée sur le principe d'optimalité de Bellman qui dit qu'un sous problème appartenant à un problème optimal est lui-même optimal [27,37,38]. La programmation dynamique évite l'énumération totale de l'espace de recherche en éliminant les décisions partielles qui ne conduisent pas à la solution optimale.

2.5.2 Les méthodes approchées

Les méthodes d'optimisation approchées constituent une alternative très intéressante pour résoudre des problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale. Ces méthodes sont souvent inspirées des mécanismes d'optimisation rencontrés dans la nature qui permettent de trouver des solutions approximatives au problème. Elles sont utilisées pour résoudre des problèmes où on ne connait pas d'algorithmes de résolution en temps polynomial et pour lequel on cherche à trouver une solution approchée de l'optimum global. Les méthodes d'optimisation approchées peuvent être divisées en deux grandes catégories, à savoir les heuristiques et les métaheuristiques.

2.5.2.1 Les heuristiques

Les heuristiques, ou méthodes approximatives, sont des méthodes dépendantes qui fournissent rapidement des solutions réalisables, pas nécessairement optimales, pour des problèmes d'optimisation NP-difficiles. Elles sont conçues généralement pour résoudre un problème particulier, en s'appuyant sur sa structure propre. Les heuristiques s'opposent donc aux méthodes exactes, qui trouvent toujours des solutions optimales si on leur laisse le temps. Les méthodes d'optimisation exactes étant de complexité exponentielle, il est généralement plus approprié de faire appel aux heuristiques pour des problèmes difficiles.

2.5.2.2 Les métaheuristiques

Les métaheuristiques représentent une nouvelle génération de méthodes d'optimisation approchées puissantes et générales, adaptables et applicables à une large classe de problèmes. Elles sont souvent inspirées des systèmes naturels, soient pris en physique, en biologie de l'évolution ou encore en éthologie. Ce sont des méthodes stochastiques itératives, qui échappent aux minima locaux et progressent vers un optimum global d'une fonction. Elles

permettent de fournir des solutions réalisables de bonne qualité en temps raisonnable. Nous distinguons deux classes de métaheuristiques : celles basées sur une solution unique et celles basées sur une population de solution.

a) Les métaheuristiques à solution unique (à trajectoire)

Les métaheuristiques à solution unique, appelées aussi méthodes de recherche par voisinage ou méthodes de trajectoire, sont des méthodes itératives qui partent d'une solution initiale et s'en éloignent progressivement, en construisant des suites de solutions formant une trajectoire dans l'espace de recherche [33]. Le processus itératif s'arrête lorsqu'une solution localement optimale est trouvée et on ne peut plus l'améliorer dans son voisinage. Les métaheuristiques à solution unique englobent essentiellement la méthode de descente, la méthode de recuit simulé, la recherche tabou, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes.

a.1) La méthode de descente

La méthode de descente (DM : Descent Method) consiste, à partir d'une solution initiale *S*, à choisir à chaque itération une solution *S'* dans le voisinage de la solution courante S qui améliore strictement la fonction objectif (généralement telle que $f(S') < f(S)$) [33].

L'algorithme de la méthode de descente peut être formalisé comme suit :

Initialisation

1 : Choisir une solution réalisable initiale *S*

Processus itératif

2 : **Tant que** le critère d'arrêt n'est pas satisfait **faire**

3 : Choisir *S* dans N(*S*)

4 : **Si** $f(S') < f(S)$ **alors**

5 : $S \longleftarrow S'$

6 : **Sinon** le critère d'arrêt est satisfait

7 : **Fin si**

8 : Fin tant que

Algorithme 2.1: Algorithme de la descente.

a.2) La méthode de recuit simulé

La méthode de recuit simulé (SA: Simulated Annealing) a été introduite et mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983 [39]. Cette méthode tire son origine du processus de recuit physique utilisé en métallurgie. Ce processus vise à améliorer la qualité des métaux en alternant des cycles de refroidissement lent et réchauffage (recuit) qui ont pour objectif de minimiser l'énergie des matériaux jusqu'à ce que l'état d'énergie soit stabilisé.

Dans la méthode SA, utilisée en optimisation pour résoudre les problèmes d'optimisation, les mécanismes d'intensification et de diversification sont contrôlés par la température, l'idée revient à diminuer petit à petit la chance d'accepter des solutions qui dégradent la fonction objectif.

L'algorithme de la méthode de recuit simulé est illustré dans l'algorithme 2.2.

1 : Initialisation du temps *t*
2 : **Tant que** le critère d'arrêt n'est pas satisfait **faire**
3 : Initialisation de la température à T_{max}
4 : **Tant que** la température est supérieure à 0 **faire**
5 : $t = t+1$
6 : Mise à jour de la solution courante
7 : **Si** la nouvelle solution est meilleure que la précédente **alors**
8 : Remplacer la solution précédente par la solution courante
9 : **Sinon**
10 : Diminuer la température
11 : **Fin si**
12 : **Fin tant que**
13 : **Fin tant que**

Algorithme 2.2: Algorithme de la méthode de recuit simulé.

a.3) La méthode de recherche avec Tabous

La méthode de recherche avec Tabous, ou simplement dite recherche tabou (TS : Tabu Search), proposée par Fred Glover en 1986 [40], est une métaheuristique itérative qualifiée de recherche locale qui utilise explicitement l'historique de la recherche pour échapper au minima locaux de la fonction objectif et explorer le voisinage. Cette méthode est basée sur

l'utilisation de mécanismes inspirés de la mémoire humaine. Une mémoire appelée liste tabou est utilisée pour garder la trace du parcours effectué et éviter les retours en arrière.

Le pseudo-code de la méthode de recherche avec tabous est illustré dans l'algorithme 2.3:

1 : Générer une configuration aléatoire initiale S_0
2 : $S^+ := S_0$
3 : $t := 0$
4 : Initialiser une liste tabou vide
5 : **Tant que** (*le critère d'arrêt n'est pas satisfait*) **faire**
6 : Choisir S_{t+1} dans le voisinage de la configuration St en tenant en compte la liste tabou
7 : **Si** (S_{t+1} est meilleure que S_t) **alors**
8 : $S^+ := S_{t+1}$
9 : **Fin Si**
10 : $t := t+1$
11 : Mettre à jour la liste tabou
12 : **Fin Tant que**

Algorithme 2.3: Algorithme de la méthode de recherche avec tabous.

b) Les métaheuristiques à population de solutions (P-Métaheuristiques)

Les métaheuristiques à population de solutions sont des méthodes itératives qui améliorent, au fur et à mesure des itérations, une population de solutions. Elles partent d'une population de solutions initiales, puis une nouvelle population de solution est générée. Cette nouvelle population est intégrée dans la population courante en utilisant quelques procédures de sélection. Le processus itératif s'arrête quand un état donné est satisfaisant. Les métaheuristiques à population de solutions englobent les algorithmes évolutionnaires, qui sont une famille d'algorithmes inspirés de la théorie de l'évolution et les algorithmes d'intelligence en essaims, qui proviennent d'analogies avec des phénomènes biologiques naturels.

b.1) Les algorithmes évolutionnaires

Les algorithmes évolutionnaires (EC : Evolutionary Computation) sont une classe d'algorithmes d'optimisation issus de la théorie de l'évolution pour résoudre des problèmes difficiles divers [33]. Ils sont composés principalement de trois éléments fondamentaux:

- Une génération, ou une population constituée de plusieurs individus représentant des configurations (solutions potentielles) du problème donné capable de se reproduire.

- Un mécanisme de sélection naturelle et d'évaluation des individus permettant de sélectionner les individus les mieux adaptés à leur environnement à l'aide de la fonction d'adaptation.
- Un mécanisme d'évolution de la population permettant, grâce à des opérateurs de variation génétiques (croisement et / ou mutation) de générer de nouveaux individus descendants appelés enfants.

La figure 2.4 résume le principe des algorithmes évolutionnaires

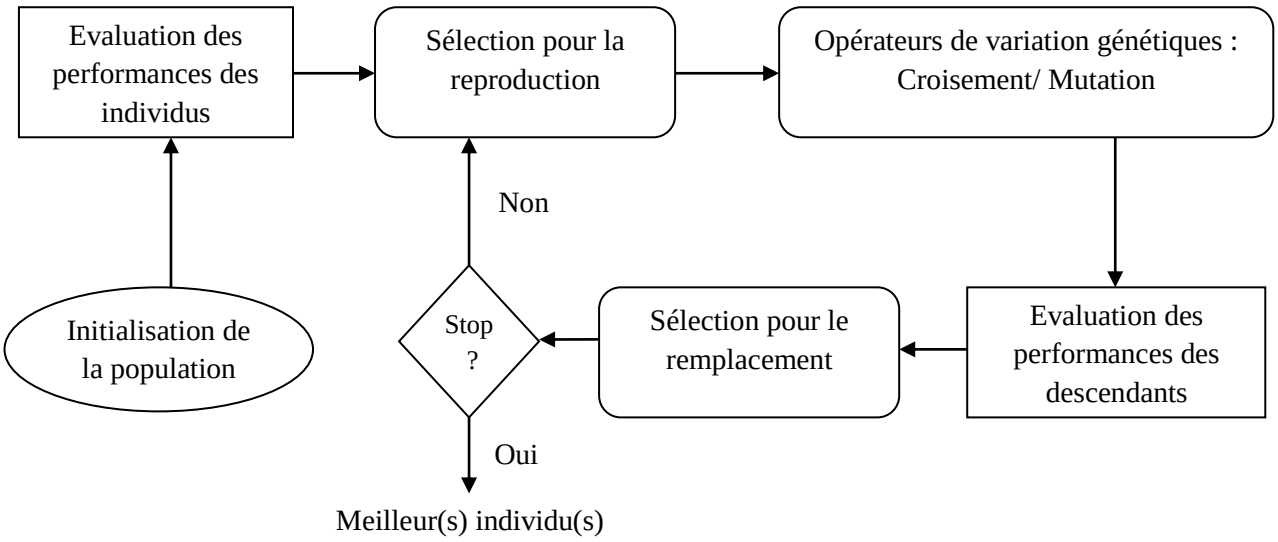


Figure 2.4: Principe d'un algorithme évolutionnaire [41].

b.1.1) Les algorithmes génétiques

Les algorithmes génétiques (GA: Genetic Algorithms) sont les algorithmes les plus populaires et les plus utilisés des algorithmes évolutionnaires. Ils ont été développés dans les années soixante-dix par John Holland, ses collègues et ses élèves à l'université de Michigan [42]. La particularité de ces algorithmes est le fait qu'ils font évoluer initialement des populations d'individus par des vecteurs binaires et plus généralement par des chaines de caractères. Un algorithme génétique dispose généralement de quatre éléments [33]:

- Une population constituée de plusieurs individus, où chaque individu de la population représente une solution potentielle au problème d'optimisation.
- Une fonction objectif à optimiser, appelée aussi fitness ou fonction d'évaluation d'individus, utilisée pour choisir et reproduire les meilleurs individus de la population.
- Une fonction de sélection qui permet de choisir et de déterminer les meilleurs individus pour la génération de la population suivante.

- Des opérateurs de variation génétiques tels que le croisement et la mutation, qui permettent de diversifier la population au cours de génération et d'explorer de nouvelles régions de l'espace de recherche.

Le pseudo-code de l'algorithme génétique de base est formalisé comme suit:

1 : Générer aléatoirement une population initiale *P(0)*
2 : *t=0*
3 : **Tant que** (*t< nombre maximum d'itérations*) ou (*le critère d'arrêt n'est pas satisfait*) **faire**
4 : Calculer la fitness de chaque individu de la population courante *P(t)*
5 : Sélectionner les parents pour la reproduction
6 : Produire les enfants des parents sélectionnés par croisement
7 : Muter les individus
8 : Etendre la population en y ajoutant les enfants
9 : Réduire la population
10 : *t=t+1*
11 : **Fin tant que**
12 : Retourner le meilleur individu trouvé
13 : **Fin**

Algorithme 2.4: Pseudo-code de l'algorithme génétique de base.

b.1.2) La programmation évolutionnaire

La programmation évolutionnaire (EP : Evolutionary Programming) est une méthode évolutive de l'intelligence artificielle, présentée et développée par L.J. Fogel [43] et son équipe au début des années soixante pour résoudre les problèmes d'apprentissage à partir d'automates à états finis. La particularité de cette méthode est qu'elle n'utilise que les opérateurs de mutation et de remplacement sans tenir compte de l'opérateur de croisement. La méthode EP ne simule pas le mécanisme de transmission génétique, mais plutôt l'adaptabilité du comportement. Son objectif est de maximiser l'adéquation d'une collection de solutions candidates dans le cadre d'une fonction objectif à partir d'un domaine donné.
Généralement, la probabilité de la mutation dans l'algorithme EP est supérieure à celle de l'algorithme génétique.

b.1.3) La stratégie d'évolution

La stratégie d'évolution (ES : Evolution Strategy) est une méthode évolutionnaire introduite par I. Rechenberg [44,45] et développée par H.P. Schwefel [46] pour résoudre les problèmes

d'optimisation pratiques. Son objectif est de maximiser l'aptitude d'un ensemble de solutions candidates dans le cadre d'une fonction objectif à partir d'un domaine donné.
Au départ, la méthode de stratégie d'évolution manipule un seul individu, un individu enfant est généré par mutation à partir de l'individu parent. Avec l'introduction de l'opérateur de variation recombinaison (semblable au croisement dans les AGs), la création d'une nouvelle population consiste à générer *m* individus enfants à partir de *n* individus parents.

b.1.4) La programmation génétique

La programmation génétique (GP : Genetic Programming) est une méthode d'optimisation évolutionnaire automatisée développée par Koza en 1992 [47] et utilisée pour la création de programmes informatiques à partir d'un énoncé de haut niveau du problème. Techniquement la méthode de la programmation génétique est une extension de l'algorithme génétique, son objectif consiste à utiliser l'induction pour élaborer un programme d'ordinateur. Ce résultat est obtenu en utilisant des opérateurs de variation génétique sur les programmes candidats avec une représentation arborescente dans le but d'améliorer l'ajustement adaptatif entre la population des programme candidats et la fonction objectif [33,47].

b.2) Les algorithmes d'intelligence en essaims

L'intelligence en essaims (SI: Swarm Intelligence) est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [33,48]. Les algorithmes de l'intelligence en essaims sont des algorithmes à base de population d'agents simples fondés sur le comportement des essaims qui existent dans la nature tels que les essaims de poissons, d'insectes ou des oiseaux. Les algorithmes de colonies de fourmis et les algorithmes d'optimisation par essaim particulaire sont les premiers algorithmes de l'intelligence en essaims. D'autres algorithmes d'optimisation qui proviennent d'analogies avec des phénomènes biologiques naturels ont été proposés. Parmi les plus significatifs nous citons l'algorithme des essaims de lucioles et l'algorithme des chauves-souris.

b.2.1) Les colonies de fourmis

L'optimisation par colonies de fourmis (ACO : Ant Colony Optimization), introduite par Marco Dorigo dans les années quatre-vingt-dix [49-52], est une métaheuristique inspirée de la nature pour résoudre des problèmes relativement complexes. Cette méthode a été conçue pour résoudre le problème de voyageur du commerce (TSP: Travelling Salesman Problem). Elle est inspirée du comportement des fourmis réelles dans la recherche de la nourriture [53]. En effet, les fourmis parviennent à trouver le chemin le plus court entre le nid et une source de

nourriture en utilisant des substances chimiques volatiles appelées phéromones qu'elles déposent sur le sol pour marquer les trajets et les chemins favorables. Après un certain temps, le chemin le plus court entre le nid et la source de nourriture présente une plus grande concentration en phéromone. Par conséquent, ce trajet aura une probabilité plus grande d'être choisi et emprunté par la grande majorité des fourmis. L'algorithme général de l'ACO est décomposé principalement, pour chaque itération, en trois étapes principales [33]:

Construction des solutions S

Dans cette étape, une colonie de fourmis artificielles génère itérativement des solutions à partir de l'ensemble des composants de solutions possibles
$S = \{s_i^j\}$ $(i = 1,2, \dots, n\ ;\ j = 1,2, \dots |D_i|)$. Tout d'abord, on commence par une solution partielle vide $s_p = \emptyset$, puis, à chaque étape de construction, la solution partielle est étendue en y ajoutant un composant de solution s_i^j parmi l'ensemble des voisins réalisables $N(s_p) \subseteq S$. le choix d'un composant dans $N(s_p)$ se fait d'une manière probabiliste. Chaque composant $s_i^j \in N(s_p)$ a une probabilité $P(s_i^j | s_p)$ d'être choisi.

Les actions Daemon

Les actions Daemon représentent des actions spécifiques à un problème donné qui ne peuvent pas être effectuées séparément par chaque fourmi. Généralement, ces actions consistent en une recherche locale parmi les solutions construites, où seulement les solutions localement optimisées sont utilisées dans la mise à jour des traces de phéromones.

Mise à jour des phéromones

La mise à jour des traces des phéromones s'effectue en deux étapes : l'étape de réduction des valeurs de phéromones par un procédé appelé évaporation et l'étape d'augmentation des valeurs de phéromones, associées à un ensemble de meilleures solutions choisies *BSol*, par un procédé appelé dépôt de phéromones. La mise à jour des traces des phéromones diffère selon l'algorithme ACO utilisé. Elle est généralement donnée par l'équation 2.6

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \sum_{s \in BSol | s_i^j \in S} g(s) \qquad (2.6)$$

Où ρ est le taux d'évaporation des phéromones généré aléatoirement dans l'intervalle [0, 1] et $g: s \rightarrow R^+$ est la fonction qualité (Quality function) telle que $f(s) < f(s') \Rightarrow g(s) \geq g(s')$.

b.2.2) L'optimisation par essaim particule

L'optimisation par essaim particule (PSO: Particle Swarm Optimization), introduite et développée par Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue) en 1995 [54], est un algorithme d'optimisation simple, efficace et global utilisé pour résoudre des problèmes relativement complexes, discontinus et non-convexes. Il s'inspire du comportement social des animaux évoluant en essaim tels que les bancs de poissons, les volées d'oiseaux migrateurs ou l'essaim s'insectes. L'essaim particule correspond à une population d'agents simples appelés particules. Chaque particule est considérée comme une solution du problème, où elle possède une vitesse et une position (vecteur de solutions) [55].

Pendant le processus de recherche, à chaque itération, la particule ajuste sa position et se déplace en prenant en compte sa meilleure position *Pbest* (déplacement égoïste) et aussi la meilleure position de son voisinage *Gbest*. Le déplacement des particules est influencé par les trois composantes suivantes [55,56]:

- Une composante physique (d'inertie) : la particule tend à suivre sa direction courante de déplacement.
- Une composante cognitive: la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée.
- Une composante sociale: la particule tend à se diriger vers le meilleur site par lequel ses voisins sont déjà passés.

La stratégie de déplacement d'une particule est illustrée dans la figure 2.5

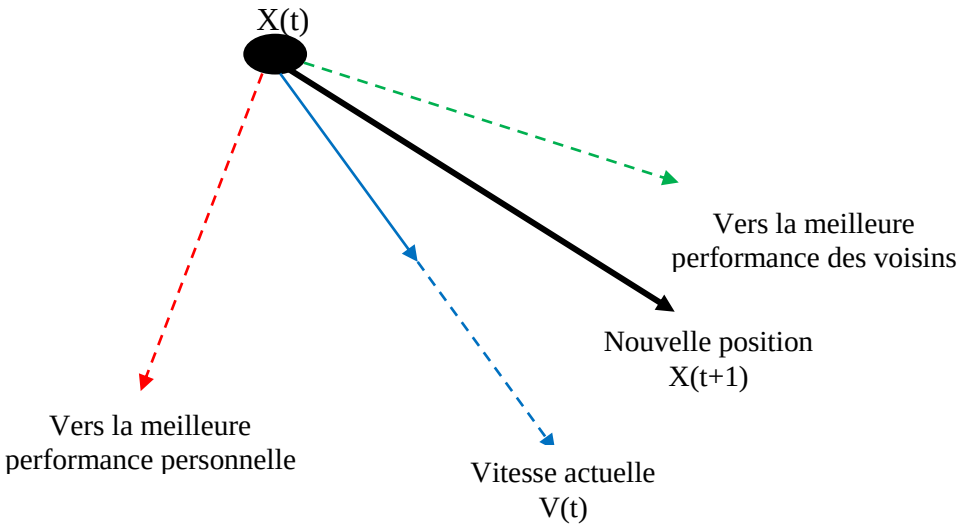


Figure 2.5: Stratégie de déplacement d'une particule [33].

Le vecteur vitesse et le vecteur position de la particule, à l'itération *t+1*, sont calculés selon les équations suivantes:

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_{1ij}^{t}\left[Pbest_{ij}^{t} - x_{ij}^{t}\right] + c_2 r_{2ij}^{t}\left[Gbest_{j}^{t} - x_{ij}^{t}\right],\ \ j \in \{1,2,\dots,D\} \tag{2.7}$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1},\quad j \in \{1,2,\dots,D\} \tag{2.8}$$

Où *w* est une constante qui représente le coefficient d'inertie ; c_1 et c_2 sont des constantes qui représentent les coefficients d'accélérations ; et enfin r_1 et r_2 sont deux nombres aléatoires dans l'intervalle [0, 1].

Une fois les particules mises à jour, $Pbest_i$ et *Gbest*, à l'itération *t+1*, sont calculés suivant les deux équations suivantes :

$$Pbest_i^{t+1} = \begin{cases} Pbest_i^{t}, si\ f(x_i^{t+1}) \geq Pbest_i^{t} \\ x_i^{t+1}, sinon \end{cases} \tag{2.9}$$

$$Gbest^{t+1} = \arg\ min_{Pbest_i} f(Pbest_i^{t+1}),\quad 1 \leq i\ \leq N \tag{2.10}$$

Le pseudo-code de l'algorithme de PSO est illustré dans l'algorithme 2.5.

1 : Initialiser aléatoirement *N* particules : positions et vitesses
2 : Evaluer les positions des particules
3 : Pour chaque particule *i*
4 : Initialiser la meilleure position de la particule comme étant la position initiale $pbest_i = x_i$
5 : Calculer *gbest* en utilisant l'équation (2.10)
6 : **Tant que** le critère d'arrêt n'est pas satisfait **faire**
7 : Calculer la vitesse des particules en utilisant l'équation (2.7)
8 : Mettre à jour et évaluer la position des particules en utilisant l'équation (2.8)
9 : Mettre à jour $pbest_i$ et *gbest* en utilisant les équations respectivement (2.9) et (2.10)
10: **Fin tant que**

Algorithme 2.5: Pseudo-code de l'algorithme PSO.

b.2.3) L'algorithme des essaims de lucioles (Firefly Algorithm)

L'algorithme des essaims de lucioles (FA: Firefly Algorithm), développé par Xin-She Yang en 2009 [57], est une nouvelle métaheuristique inspirée de la nature qui est utilisée pour résoudre divers problèmes d'optimisation. Cet algorithme d'optimisation est inspiré du comportement de lucioles qui est basé sur le système d'attraction en utilisant des lumières clignotantes. Généralement, les lucioles émettent une lumière clignotante produite par un processus de bioluminescence pour identifier l'emplacement de la nourriture et attirer d'autres

lucioles. Pour plus de simplicité, l'algorithme de lucioles utilise les trois règles idéalisées suivantes [57-59]:

1) Toutes les lucioles sont unisexes, ce qui signifie qu'une luciole sera séduite par d'autres lucioles indépendamment de leur sexe ;
2) L'attractivité d'une luciole est directement proportionnelle à sa luminosité et les deux diminuent au fur et à mesure que leur distance augmente. Ainsi, pour deux lucioles clignotantes, la luciole la moins lumineuse sera attirée par la plus lumineuse. Si aucune luciole n'est plus lumineuse qu'une luciole particulière, cette dernière se déplacera de façon aléatoire dans l'espace de recherche ;
3) La luminosité d'une luciole est déterminée par la valeur de la fonction objectif à optimiser. Dans un problème de maximisation, la luminosité de chaque luciole est proportionnelle à la valeur de la fonction objectif. Dans le cas du problème de minimisation (comme celui du problème de routage multicast), la luminosité de chaque luciole est inversement proportionnelle à la valeur de la fonction objectif.

Le pseudo-code de l'algorithme des essaims de lucioles original est illustré dans l'algorithme 2.6.

1 : Définir une fonction objectif $f(x)$, $x= (x_1, x_2, \ldots, x_d)^T$
2 : Générer une population initiale de lucioles $xi\ (i=1, 2, \ldots, n)$
3 : Définir l'intensité de la lumière I_i à un point x_i par la fonction objectif $f(x_i)$
4 : Définir le coefficient d'absorption γ
5 **: Tant que** ($t<$ *nombre maximum d'itérations*) **faire**
6 : **Pour** i=1 à n **faire** //(toutes les lucioles)
7 : **Pour** j=1 à n **faire** //(toutes les lucioles)
8 : **Si** ($Ii < Ij$) **alors**
9 : Déplacer la luciole i vers la luciole j en utilisant l'équation (2.14)
10 : **Fin si**
11 : L'attractivité varie en fonction de la distance r via $e^{-\gamma r^2}$
12 : Evaluer les nouvelles solutions et mettre à jour la nouvelle intensité de lumière I_i en utilisant l'équation (2.11)
13 : **Fin pour**
14 : **Fin pour**
15 : Classer les lucioles et trouver la meilleure solution courante x_{Gbest}
16 : **Fin tant que**

Algorithme 2.6: Pseudo-code de l'algorithme des essaims de lucioles.

L'algorithme des essaims de lucioles travaille en se basant sur deux principes: la variation de l'intensité lumineuse (I) et la formulation de l'attractivité (β). Pour plus de simplicité, l'attractivité d'une luciole est déterminée par l'intensité de la lumière ou de la luminosité qui

est associée à la fonction objectif. L'intensité de la lumière I varie en fonction de la distance r de façon exponentielle et peut être exprimée comme suit:

$$I(r) = I_0 e^{-\gamma r^2} \tag{2.11}$$

Où I_0 représente l'intensité lumineuse initiale d'une luciole à l'état d'origine *(r=0)*, γ est le coefficient d'absorption lumineuse et r_{ij} désigne la distance entre deux lucioles x_i et x_j respectivement. Cette distance est simplement exprimée sous la forme euclidienne de la façon suivante :

$$r_{ij} = \|x_{i-}x_j\| = \sqrt{\sum_{k=1}^{d}\left(x_{ik-}x_{jk}\right)^2} \tag{2.12}$$

Où x_{ik} et x_{jk} représentent les $k^{ème}$ éléments des $i^{ème}$ et $j^{ème}$ lucioles, respectivement, et *d* représente la dimension d'un problème. L'attractivité d'une luciole est proportionnelle à sa luminosité, de telle sorte qu'elle peut être définie comme suit :

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{2.13}$$

Où β_0 dénote l'attractivité initiale d'une luciole à $r = 0$.
Le mouvement d'une luciole *i* qui est attiré par une autre luciole *j* plus brillante est déterminé par :

$$x_i^{t+1} = x_i^t + \beta_0 e^{(-\gamma r_{ij}^2)}\left(x_j^t - x_i^t\right) + \alpha(rand() - 0.5) \tag{2.14}$$

Où x_i^t et x_j^t représentent les positions de la luciole *i* avec une faible intensité de la lumière et de la luciole j avec une intensité lumineuse plus élevée au temps t respectivement.
α est le paramètre de répartition aléatoire qui détermine au hasard le comportement du mouvement des lucioles et *rand* est un nombre aléatoire généré à partir d'une distribution uniforme dans l'intervalle [0, 1].

Le processus est répété jusqu'à ce que le nombre maximal de cycles est atteint ou bien la meilleure solution n'a pas changé pour une suite d'itérations successives.

b.2.4) L'algorithme des chauves-souris (Bat Algorithm)

L'algorithme des chauves-souris (BA : Bat Algorithm), développé par Xin-She Yang en 2010 [60], est une nouvelle et intéressante métaheuristique utilisée pour résoudre divers problèmes d'optimisation. Cet algorithme est basé sur le comportement des chauves-souris qui cherchent leurs proies/nourritures en utilisant leur capacité d'écholocation [60]. L'écholocation

fonctionne comme un type de sonar, les chauves-souris émettent des pulsations sonores très fortes par la bouche ou par le nez (ultrason), attendent qu'elles se heurtent à un objet (proie/aliment) et après un certain temps les échos retournent à leurs oreilles. Ainsi les chauves-souris sont capables de déterminer à quel point elles sont loin de l'objet.

Pour plus de simplicité, Yang a idéalisé quelques caractéristiques d'écholocation des chauves-souris. Les règles idéalisées suivantes sont utilisées [60]:

1) Toutes les chauves-souris utilisent l'écholocation pour détecter la distance, et connaissent aussi la différence entre la nourriture / les proies et les obstacles;
2) Les chauves-souris volent au hasard avec une vitesse v_i à la position x_i avec une fréquence fixe f_{min}, une longueur d'onde λ et une intensité sonore A_0 pour la recherche d'une proie. Elles peuvent ajuster automatiquement le taux d'émission d'impulsions $r \in [0, 1]$, en fonction de la proximité de leur cible;
3) Bien que l'intensité sonore puisse varier de plusieurs façons, il est supposé que l'intensité varie d'une grande valeur A_0 (positive) à une valeur constante minimale A_{min}.

Le pseudo-code de l'algorithme des chauves-souris original est illustré dans l'algorithme 2.7.

1 : Initialisation de la population des chauves-souris x_i *(i=1, 2, …, n)* et v_i
2 : Définir la fréquence d'impulsion f_i à la position x_i
3 : Initialiser les taux d'émissions de pulsation r_i, et l'intensité A_i
4 : **Tant que** (*t< nombre maximum d'itérations*) **faire**
5 : Générer de nouvelles solutions en ajustant la fréquence, mettant à jour les vitesses et les positions / solutions [équations (2.15) à (2.17)]
6 : **Si** ($Rand( ) > r_i$) **alors**
7 : Sélectionner une solution parmi les meilleures solutions
8 : Générer une solution locale autour de la meilleure solution sélectionnée
9 : **Fin si**
10 : Générer une nouvelle solution en volant aléatoirement
11 : **Si** ($Rand( ) < A_i$ et $f(x_i) < f(x_{Gbest})$) **alors**
12 : Accepter les nouvelles solutions
13 : Incrémenter r_i et réduire A_i
14 : **Fin si**
15 : Classer les chauves-souris et trouver la meilleure solution courante x_{Gbest}
16 : **Fin tant que**

Algorithme 2.7: Pseudo-code de l'algorithme des chauves-souris

Tout d'abord, pour chaque chauve-souris (*i*), la position x_i, la vitesse v_i et la fréquence f_i sont initialisés au hasard. Le mouvement des chauves-souris est donné en mettant à jour leurs vitesses et leurs positions à l'instant *t* en utilisant les équations suivantes:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (2.15)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{Gbest})f_i \quad (2.16)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (2.17)$$

Où β désigne un vecteur aléatoire généré à partir d'une distribution uniforme dans l'intervalle [0, 1]. La variable x_{Gbest} représente la meilleure solution globale courante qui est obtenue par comparaison de toutes les solutions obtenues par les *m* chauves-souris à chaque itération *t*. Les valeurs de f_{min}, f_{max} dépendent de la taille du domaine du problème d'intérêt à résoudre.

En second lieu, afin de maintenir et d'améliorer la diversité des solutions possibles, une approche de recherche locale en utilisant une marche aléatoire locale est appliquée pour générer une nouvelle solution autour des meilleures solutions courantes, telle que donnée par l'équation suivante :

$$x_{new} = x_{old} + \varepsilon A^t \quad (2.18)$$

Où ε représente un nombre généré aléatoirement dans l'intervalle [-1, 1] et A^t représente la moyenne d'intensité de toutes les chauves-souris à l'instant *t*. L'intensité varie d'une grande valeur A_0 (positive) à une valeur constante minimale A_{min}. L'intensité A_i et le taux d'émission d'impulsions r_i sont mis à jour à chaque itération. L'intensité diminue généralement quand une chauve-souris trouve sa proie tandis que le taux d'émission d'impulsion augmente. L'intensité A_i et le taux d'émission d'impulsions r_i sont mis à jour en utilisant les équations suivantes:

$$A_i^{t+1} = \alpha A_i^t \quad (2.19)$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \quad (2.20)$$

Où α ($\alpha \in [0, 1]$) et γ ($\gamma > 0$) sont des constantes. La constante α, comme dans l'algorithme du recuit simulé, contrôle la convergence de l'algorithme.

Le processus est répété jusqu'à ce que le nombre maximal de cycles est atteint ou bien la meilleure solution n'a pas changé pour une suite d'itérations successives.

2.6 Conclusion

Dans ce chapitre, nous avons présenté le problème d'optimisation, les notions et les concepts relatifs à l'optimisation, la classification des problèmes d'optimisation selon plusieurs critères. Nous avons également présenté les méthodes d'optimisation où nous avons décrit le principe de base de quelques algorithmes utilisés pour la résolution des problèmes complexes.

Dans le chapitre suivant, nous présenterons en détail une amélioration de l'algorithme BA où nous proposerons deux variantes ICBBA1 et ICBBA2 appliquées pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETS.

CHAPITRE 3

ICBBA POUR LE ROUTAGE MULTICAST: APPLICATION EN WMNs et VANETs

3.1 Introduction

Ce chapitre a pour objet de présenter une amélioration de l'algorithme des chauves-souris BA pour résoudre le problème de routage multicast avec QoS. BA a été utilisé à l'origine pour résoudre des problèmes d'optimisation continus alors que le problème de routage multicast est un problème combinatoire. Donc, une version binaire de BA, à savoir BBA, est nécessaire pour résoudre ce problème.

Deux variantes de l'algorithme BBA ont été proposées pour assurer la diversité des solutions. La première variante, nommée ICBBA1, utilise la fonction logistique pour déterminer le paramètre β de la fréquence f_i. La deuxième variante, nommée ICBBA2, utilise la fonction tent pour déterminer le paramètre β de la fréquence f_i.

Dans ce chapitre, nous présentons un rappel des méthodes et approches proposées dans la littérature pour résoudre le problème de routage multicast, nous décrivons ensuite la formulation du problème de routage multicast, nous présentons également en détail une amélioration de l'algorithme BA où nous proposons deux variantes ICBBA1 et ICBBA2 pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETs pour enfin évaluer leurs performances et comparer leurs caractéristiques avec d'autres méthodes d'optimisation existantes dans la littérature.

3.2 Approches d'optimisation pour le routage multicast avec QoS

Au cours de ces dernières années, plusieurs métaheuristiques tels que les algorithmes génétiques (GA) [61-64], les essaims particules (PSO) [65-68], les colonies de fourmis (ACO) [69,70] et la recherche tabou (TS) [71,72] ont été utilisés pour résoudre le problème de

routage multicast avec qualité de service (QoS) dans les réseaux sans fil particulièrement dans les WMNs et les VANETs.

Un certain nombre d'approches basées sur les algorithmes génétiques ont été proposées pour résoudre le problème de routage multicast. Hwang et al. [61] ont proposé une approche basée sur l'algorithme génétique pour résoudre le problème de routage multicast sans contraintes. L'optimisation de l'arbre multicast a été réalisée par une sélection de série de chemins ainsi que par des opérations de croisement et de mutation. Les résultats ont montré que l'algorithme proposé est capable de trouver une meilleure solution en le comparant à d'autres algorithmes existants dans la littérature. Haghighat et al. [62] ont proposé un nouvel algorithme de routage avec QoS basé sur l'algorithme génétique qui permet de trouver un arbre multicast à moindre coût et satisfaire les contraintes de délai et de bande passante. Les résultats de simulation ont révélé que l'algorithme proposé est plus performant que d'autres méthodes existantes dans la littérature. Sun et al. [63] ont proposé un algorithme d'optimisation basé sur l'algorithme génétique pour résoudre le problème de routage multicast avec contraintes multiples et trouver un arbre multicast qui satisfait les contraintes de la bande passante et de délai. Les résultats obtenus ont démontré l'exactitude et l'efficacité de l'approche proposée pour la stabilité de la route dans les MANET. Yen et al. [64] ont proposé une méthode d'optimisation basée sur l'algorithme génétique et le principe d'inondation limitée des réseaux ad hoc mobiles pour la résolution du problème de routage multicast avec contraintes multiples. Les résultats expérimentaux ont révélé les meilleures performances de l'algorithme proposé par rapport à d'autres algorithmes et protocoles existants dans la littérature.

Des variantes de l'algorithme PSO ont été proposées par les chercheurs pour résoudre le problème de routage multicast avec QoS. Liu et al. [65] ont proposé une méthode basée sur l'algorithme d'optimisation d'essaims particules (PSO) pour résoudre le problème de routage multicast en utilisant la sélection de série de chemins pour obtenir une solution réalisable (arbre multicast) en se basant sur l'échange des chemins dans le vecteur. Les résultats de simulation indiquent que l'algorithme proposé peut converger vers la solution optimale ou presque optimale avec un coût de calcul plus faible. Il est également apparu que PSO est plus performant que l'algorithme génétique. Sun et al. [66] ont proposé une variante de l'algorithme d'optimisation par essaims particules basée sur le principe du quantum appelé Quantum-Behaved Particle Swarm Otpimization (QPSO) pour résoudre le problème de routage multicast. L'algorithme proposé convertit le problème de routage avec qualité de service en un problème de programmation linéaire en nombres entiers, puis résout le

problème par l'algorithme QPSO et suppression de boucles. Les résultats de simulation ont montré l'efficacité et la supériorité de l'algorithme proposé sur les algorithmes basés sur PSO et GA. Qu et al. [67] ont proposé un nouvel algorithme PSO basé sur le saut. Un opérateur de remplacement de chemins a été utilisé dans le déplacement des particules afin d'améliorer la position de la particule par rapport à la structure de l'arbre. Les résultats expérimentaux ont montré que l'algorithme proposé donne de meilleures performances par rapport à d'autres approches existantes dans la littérature. Shen et al. [68] ont proposé une autre variante de PSO appelée Bi-Velocity Discret Particle Swarm pour résoudre le problème de routage multicast sans contrainte, Cette nouvelle méthode est représentée comme une stratégie bi-vitesse pour décrire la caractéristique binaire du problème de routage multicast. Les résultats ont montré la supériorité de l'algorithme sur les méthodes basées sur les algorithmes GA et PSO.

Tseng et al. [69] ont proposé une nouvelle approche basée sur l'algorithme des colonies de fourmis pour construire un arbre multicast qui vérifie les contraintes de délai et de degré. Les résultats ont montré que l'algorithme donne un coût réduit, mais un temps de calcul trop long. Wang et al. [70] ont proposé une nouvelle méthode basée sur l'algorithme des colonies de fourmis et la croissance de l'arborescente afin d'obtenir un arbre de routage multicast avec le moindre coût qui satisfait les contraintes de délai, de la gigue et de la bande passante. Les résultats de simulation ont démontré que l'algorithme proposé donne de meilleures performances dans la recherche, le temps de convergence, et à l'échelle de la capacité d'adaptation par rapport à d'autres approches telles que GA et TS.

Ghaboosi et al. [71] ont proposé divers algorithmes de recherche tabou pour construire un arbre multicast à moindre coût et qui satisfait les contraintes de délai et de bande passante. Les résultats de simulation ont montré que la combinaison de leur mouvement complet ainsi que la combinaison de la stratégie de diversification proposée avec la nouvelle stratégie de reconstruction donne une meilleure performance en terme du coût total des arbres. Wang et al. [72] ont proposé une nouvelle approche basée sur l'algorithme de recherche tabou qui permet de trouver un arbre multicast à moindre coût et satisfaire les contraintes de délai et de la gigue. Cet algorithme basé sur la sélection de la racine de l'arbre (point central du rendez vous: RP) permet d'améliorer la structure et les performances de l'arbre multicast. Les tests numériques ont donné de bons résultats en termes de coût, de délai et de la gigue. Tableau 3.1 présente une synthèse des travaux sur le routage multicast avec QoS basés sur des approches non hybrides.

Algorithmes	Références	Fonction objectif (coût)	Délai	gigue	Bande passante	Taux de perte
GA	Hwang et al. [61]	*				
GA	Haghighat et al. [62]	*	*		*	
GA	Sun et al. [63]	*	*		*	
GA	Yen et al. [64]	*	*	*	*	
PSO	Liu et al. [65]	*	*	*	*	*
PSO	Sun et al. [66]	*	*	*	*	*
PSO	Qu et al. [67]	*	*			
PSO	Shen et al. [68]	*				
ACO	Tseng et al. [69]	*	*			
ACO	Wang et al. [70]	*	*	*	*	
TS	Ghaboosi et al. [71]	*	*		*	
TS	Wang et al. [72]	*	*			

Tableau 3.1: Synthèse des travaux sur le routage multicast avec QoS basés sur des approches non hybrides.

3.3 Formulation du problème de routage multicast

Le réseau de communication peut être modélisé et présenté par un graphe orienté et pondéré *G = (V, E),* où *V* dénote l'ensemble des nœuds et *E* dénote l'ensemble des arcs du graphe représentant les liens sans fil entre les nœuds. $|V| = n$ est le nombre de nœuds et $|E| = l$ est l'ensemble d'arcs du réseau. Chaque arc $e = (i, j) \in E$ qui relie le nœud *i* avec le nœud *j* est associé à un coût de l'arc $Cost(e): E \rightarrow R^+$, délai de l'arc $Delay(e): E \rightarrow R^+$, gigue de l'arc $Jitter(e): E \rightarrow R^+$, bande passante de l'arc $Bandwidth(e): E \rightarrow R^+$, taux de perte de paquets de l'arc $LR(e) : E \rightarrow R^+$, où R^+ désigne l'ensemble de tous les nombre réels non négatifs.

Dans le cas général, un réseau sans fil est asymétrique (les liens sont bidirectionnels). Soit *e = (i, j)* et *é = (j, i),* alors il est souvent possible que $Cost(e) \neq Cost(é)$, $Delay(e) \neq Delay(é)$,

Jitter (e) ≠ Jitter(é), Bandwidth(e) ≠ Bandwidth(é), LR(e) ≠ LR(é). On suppose que $s \in V$ représente le nœud source et $M \subseteq \{V - \{s\}\}$ représente l'ensemble des nœuds destinations tel que *s* et *M* forment un arbre multicast (multidiffusion) *T(s, M)*. La figure 3.1 montre un exemple d'un graphe orienté pondéré.

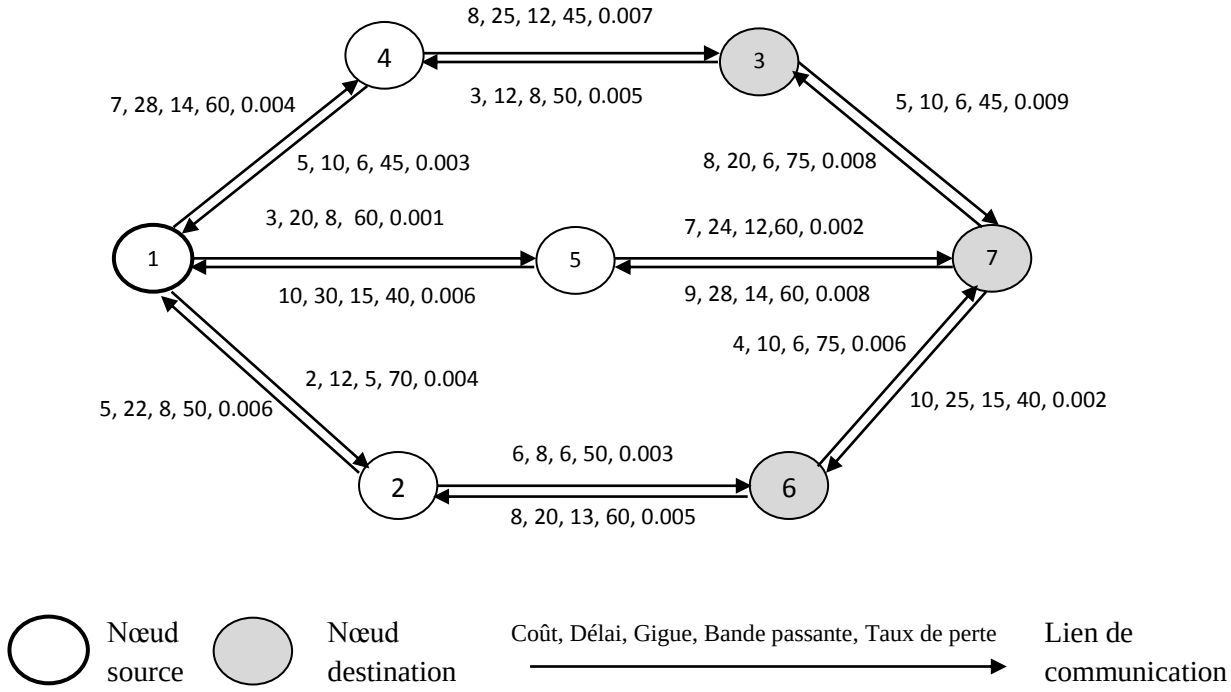


Figure 3.1: Exemple d'un graphe orienté pondéré.

L'arbre multicast a les paramètres suivants [66,73]:

Le coût total de l'arbre multicast *T(s, M)*, désigné par $Cost(T(s,M))$, est égal à la somme des coûts de tous les liens qui composent cet arbre. Il peut être donné par :

$$Cost(T(s,M)) = \sum_{e \in T(s,M)} Cost(e) \quad (3.1)$$

Le délai total du chemin $P_T(s, m)$, désigné par $Delay(P_T(s,m))$, est simplement la somme des délais des liens qui le composent.

$$Delay(P_T(s,m)) = \sum_{e \in P_T(s,m)} Delay(e) \quad (3.2)$$

Où $P_T(s, m)$ représente le chemin de routage de l'arbre multicast *T(s, M)* qui relie le nœud source *s* avec le nœud destination $m \in M$ (Il est clair que $P_T(s, m)$ est un sous-ensemble de *T(s, M)*).

La gigue de l'arbre multicast, notée par $Jitter(T(s,M)$, est définie comme la différence moyenne des délais des chemins qui relient le nœud source à chaque nœud destination et peut être donné par l'équation (3.3).

$$Jitter(T(s,M) = \sqrt{\sum_{m\in M}(delay(P_t(s,m) - dela\_avg)^2} \quad (3.3)$$

Où *delay-avg* représente la valeur moyenne des délais des chemins qui relient le nœud source avec tous les nœuds destinations.

La bande passante du chemin $P_T(s, m)$, notée par by $Bandwidth\big(P_T(s,m)\big)$, est définie comme la bande passante minimale requise dans chaque lien du chemin qui relie le nœud source *s* avec le nœud destination *m*:

$$Bandwidth\big(P_T(s,m)\big) = \min_{e\in P_{T(s,m)}} (Bandwidth(e)) \quad (3.4)$$

Le taux de perte des paquets du chemin qui relie le nœud source *s* avec le nœud destination *m*, noté par $LR\big(P_T(s,m)\big)$, est donné par :

$$LR(P_T(s,m)) = 1 - \prod_{e\epsilon \mathrm{P_T(s,m)}} (1 - LR(e)) \quad (3.5)$$

Le problème de routage multicast avec contraintes multiples (Multi-constraind Least-cost) peut être représenté comme un problème de minimisation dont l'objectif principal est de construire un arbre multicast qui minimise la fonction coût et satisfait les contraintes suivantes [66,73]:

1) La contrainte de délai: $Delay(P_T(s, m)) \leq D_{max}$, $\forall\, m \in M$, c.à.d. le délai maximum autorisé pour chaque chemin qui relie le nœud source avec l'ensemble des nœuds destinations ne doit pas dépasser la contrainte du délai D_{max} ;

2) La contrainte de la gigue: $Jitter(T(s, M)) \leq J_{max}$ c.à.d. la gigue autorisée des services temps réel doit être inférieure à la contrainte de la gigue J_{max} ;

3) La contrainte de la bande passante: $min(Bandwidth\ (P_T(s, m))) \geq B_{min}$, $\forall\, m \in M$, c.à.d. la bande passante minimale requise dans chaque lien doit être supérieure ou égale à la contrainte de la bande passante B_{min} ;

4) La contrainte de taux de perte des paquets: $LR(P_T(s, m)) \leq LR_{max}$, $\forall\, m \in M$, c.à.d. le taux de perte du chemin qui relie le nœud source au nœud destination ne doit pas dépasser la contrainte du taux de perte des paquets LR_{max}.

Ainsi, le problème de routage multicast avec qualité de service (QoS) peut être formulé comme suit :

$$Minimiser\ Cost\ (T(s, M)) \tag{3.6}$$

Sous les contraintes:

$$Delay(P_T(s, m)) \leq D_{max} \tag{3.7}$$

$$Jitter(P_T(s, m)) \leq J_{max} \tag{3.8}$$

$$Min(Bandwidth(P_T(s, m))) \geq B_{min} \tag{3.9}$$

$$LR(P_T(s, m)) \leq LR_{max} \tag{3.10}$$

3.4 ICBBA pour le routage multicast (Improved Chaotic Binary Bat Algorithm)

3.4.1 Algorithme des chauves-souris binaire (BBA: Binary Bat Algorithm)

La version originale de l'algorithme des chauves-souris a été conçue pour résoudre des problèmes d'optimisation continus alors que le problème de routage multicast est un problème d'optimisation combinatoire qui a un espace de recherche discret binaire. Donc, une version binaire de l'algorithme des chauves-souris, à savoir BBA (pour Binary Bat Algorithm), est nécessaire pour résoudre le problème de routage multicast. Dans l'espace de recherche discret binaire, la mise à jour de la position signifie à la base la commutation entre les deux valeurs 0 et 1 [74,75]. Cette commutation devrait être faite en se basant sur la vitesse des agents [74,75]. Le problème est de savoir comment utiliser le concept de vitesse dans l'espace réel, afin de changer la position dans l'espace binaire. L'idée consiste à mettre à jour la position d'un agent en se basant sur la probabilité de sa vitesse [74-78]. Pour ce faire, une fonction de transfert et une méthode de binarisation sont utilisées. La fonction de transfert définit la probabilité de changement des positions des éléments d'un vecteur de la position 0 à la position 1 et vice versa [74], ce qui oblige les chauves-souris à se déplacer dans un espace de recherche binaire.

La fonction de transfert arc tangente est utilisée pour la binarisation de l'algorithme de chauves-souris. Elle est donnée par l'équation (3.11):

$$V(v_{ik}^t) = \left|\frac{2}{\pi}\text{arctang}(\frac{2}{\pi}v_{ik}^t)\right| \tag{3.11}$$

Où v_{ik}^{t} désigne la vitesse de l'i^{ème} chauve-souris à l'itération *t* dans la k^{ème} dimension. Le résultat de cette opération est un nombre réel compris entre 0 et 1. La méthode de binarisation est alors nécessaire pour mettre à jour la position et obtenir une valeur 0 ou 1. Cette méthode est représentée par l'équation (3.12).

$$x_{ik}^{t+1} = \begin{cases} (x_{ik}^{t})^{-1}, & \text{si } rand(\,) < V(v_{ik}^{t+1}) \\ x_{ik}^{t}, & \text{si } rand(\,) \geq V(v_{ik}^{t+1}) \end{cases} \qquad (3.12)$$

Où x_{ik}^{t} désigne la position de l'i^{ème} chauve-souris à l'itération *t* dans la k^{ème} dimension.

Le pseudo-code de l'algorithme des chauves-souris binaire (BBA) est illustré dans l'algorithme 3.1. Les principaux changements par rapport à l'algorithme des chauves-souris sont soulignés.

1 : Initialisation de la population des chauves-souris: x_i *(i=1, 2, ..., n)= rand (0* ou *1)* et v_i*=0*
2 : Définir la fréquence d'impulsion f_i à la position x_i
3 : Initialiser les taux d'émissions de pulsation r_i, et l'intensité A_i
4 : **Tant que** (*t< nombre maximum d'itérations*) **faire**
5 : Ajuster la fréquence et mettre à jour les vitesses
6 : Calculer la valeur de la fonction de transfert v-shaped en utilisant l'équation (3.11)
7 : Mettre à jour les positions/ solutions en utilisant l'équation (3.12)
8 : **Si** (*Rand( ) > r_i*) **alors**
9 : Sélectionner une solution parmi les meilleures solutions
10 : Changer certaines dimensions du vecteur de positions avec certaines dimensions de xGbest
11 : **Fin si**
12 : Générer une nouvelle solution en volant aléatoirement
13 : **Si** (*Rand( ) < A_i* et *f(x_i) < f(x_{Gbest})*) **alors**
14 : Accepter les nouvelles solutions
15 : Incrémenter r_i et réduire A_i
16 : **Fin si**
17 : Classer les chauves-souris et trouver la meilleure solution courante x_{Gbest}
18 : **Fin tant que**

Algorithme 3.1: Pseudo-code de l'algorithme des chauves-souris binaire.

3.4.2 Algorithme des chauves-souris chaotique binaire amélioré ICBBA

Afin d'améliorer les performances du BBA, nous avons proposé deux méthodes de modification. La première méthode utilise des systèmes chaotiques pour le choix de la bonne valeur du paramètre β de l'impulsion de fréquence f_i. Un ajustement correct de la fréquence f_i joue un rôle très important dans la mise à jour des positions des chauves-souris. Il existe plusieurs systèmes chaotiques telles que la fonction logistique, la fonction cercle, la fonction tent, la fonction piecewise, la fonction itérative, la fonction de Chebyshev, la fonction sinus, la fonction sinusoïdale, etc [79-82]. Les fonctions logistique et tent sont les systèmes chaotiques les plus fréquemment utilisées avec de simples opérations [80,83,84]. Ces deux systèmes chaotiques sont adoptés pour générer la valeur du paramètre β à chaque itération.

Le paramètre β généré par la fonction logistique est donné par l'équation suivante :

$\beta_{i+1} = a\beta_i(1-\beta_i),\ i = 0, 1, 2\ldots$ (3.13)

Où β_i est la valeur de la fonction chaotique à l'i^{ème} itération, elle est dans l'intervalle [0, 1]. a est un paramètre de contrôle et $0 < a < 4$. Lorsque $3{,}571448 \leq a \leq 4$, la fonction logistique est dans un état chaotique. En particulier, lorsque a = 4 et β_i ne fait pas partie de {0, 0.25, 0.5, 0.75, 1}, la fonction logistique est en plein chaos. Dans le cas contraire, l'équation logistique n'a pas un comportement chaotique [85].

Le paramètre β généré par la fonction tent est donné par l'équation suivante:

$$\beta_{i+1} = \begin{cases} \frac{\beta_i}{0.7}, & \beta_i < 0.7 \\ \frac{10(1-\beta_i)}{3}, & \beta_i \geq 0.7 \end{cases} \quad (3.14)$$

Où β_i est la valeur de la fonction chaotique à la i^{ème} itération, elle est dans l'intervalle [0, 1].

La nouvelle équation de la fréquence est donnée par l'équation suivante:

$$f_i = f_{min} + (f_{max} - f_{min})\beta_i \quad (3.15)$$

La deuxième méthode est la formulation dynamique utilisée pour mettre à jour le paramètre α de l'intensité A_i durant l'optimisation adaptative. α est calculé en utilisant la formulation dynamique suivante:

$$\alpha^{new} = \alpha^{old}(0.5 \times iter)^{\frac{1}{iter}} \quad (3.16)$$

Où *iter* est le nombre d'itération courant. L'équation ci-dessus est obtenue expérimentalement, après plusieurs exécutions de l'algorithme.

Le pseudo-code de l'algorithme des chauves-souris chaotique binaire amélioré (ICBBA) est illustré dans l'algorithme 3.2. Les principaux changements par rapport à l'algorithme des chauves-souris binaire sont soulignés.

1 : Générer la population initiale
2 : Réparer les solutions
3 : Initialiser la vitesse v_i
4 : Définir la fréquence d'impulsion f_i à la position x_i
5 : Initialiser les taux d'émissions de pulsation r_i, et l'intensité A_i
6 : **Tant que** (t< *nombre maximum d'itérations*) **faire**
7 : Calculer la valeur du paramètre β en utilisant la cartes chaotiques (carte logistique ou carte tent) en utilisant l'équation 3.13 ou l'équation 3.14
8 : Ajuster la fréquence en utilisant l'équation 3.15
9 : Mettre à jour les vitesses en utilisant l'équation 2.16
10 : Calculer la valeur de la fonction de transfert v-shaped en utilisant l'équation (3.11)
11 : Mettre à jour les positions/ solutions en utilisant l'équation (3.12)
12 : Réparer les solutions
13 : **Si** (*Rand( )* > r_i) **alors**
14 : Sélectionner une solution parmi les meilleures solutions
15 : Changer certaines dimensions du vecteur de positions avec certaines dimensions de xGbest
16 : **Fin si**
17 : Générer une nouvelle solution en volant aléatoirement
18 : Réparer la solution
19 : **Si** (*Rand( )* < A_i et $f(x_i) < f(x_{Gbest})$) **alors**
20 : Accepter les nouvelles solutions
21 : Mettre à jour la valeur de α de l'intensité A_i en utilisant l'équation (3.16)
22 : Incrémenter r_i et réduire A_i
23 : **Fin si**
24 : Classer les chauves-souris et trouver la meilleure solution courante x_{Gbest}
25 : **Fin tant que**

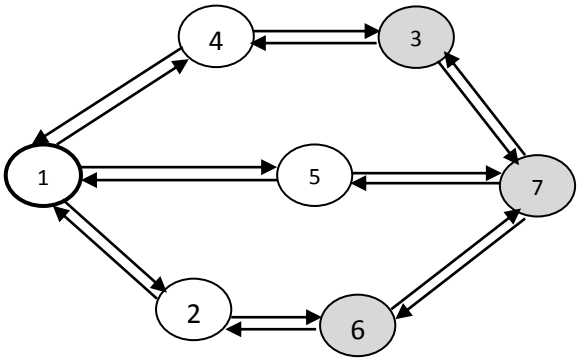
Algorithme 3.2: Pseudo-code de l'algorithme des chauves-souris chaotique binaire amélioré.

3.4.2.1 Représentation de la solution

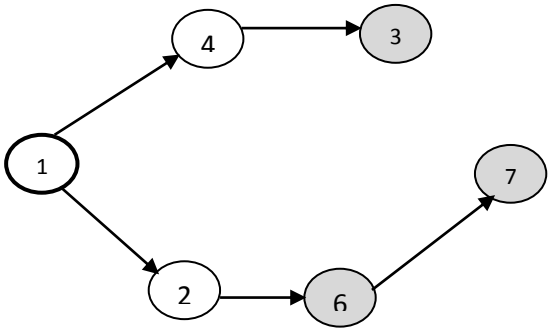
La solution de chaque chauve-souris, qui est un arbre de diffusion multicast, est représentée par une matrice binaire X de dimension NxN, où N représente le nombre total de nœuds dans le réseau. La matrice binaire NxN est définie comme suit :

$$x_{ij} = \begin{cases} 1 \text{ si l'arc } e_{ij} \text{ est sélectionné pour construire l'arbre multicast} \\ 0, \text{ sinon} \end{cases} \quad (3.17)$$

La figure 3.2(c) montre une représentation matricielle binaire de l'arbre multicast (figure 3.2(b)) obtenu à partir d'un graphe donné (figure3.2(a)), où 1 est le nœud source et 3, 6 et 7 représentent les nœuds destinations.



(a) Graphe original.



(b) Arbre multicast obtenu.

	Nœud 1	Nœud 2	Nœud 3	Nœud 4	Nœud 5	Nœud 6	Nœud 7
Nœud 1	0	1	0	1	0	0	0
Nœud 2	0	0	0	0	0	1	0
Nœud 3	0	0	0	0	0	0	0
Nœud 4	0	0	1	0	0	0	0
Nœud 5	0	0	0	0	0	0	0
Nœud 6	0	0	0	0	0	0	1
Nœud 7	0	0	0	0	0	0	0

(c) Représentation matricielle binaire de l'arbre multicast obtenu.

Figure 3.2: Exemple d'une représentation binaire d'un arbre multicast.

3.4.2.2 Initialisation de la population

Avant de générer la population initiale, nous devons supprimer tous les liens dont la bande passante est inférieure à la bande passante minimale requise B_{min}. L'algorithme de recherche aléatoire Depth-First est utilisé pour générer une solution initiale aléatoire (arbre multicast). Nous commençons à partir du nœud source, et nous le connectons aléatoirement à l'un de ses successeurs. Si ce nœud constitue une destination, le chemin est valide, sinon on refait le même processus jusqu'à atteindre toutes les destinations. Le même processus est utilisé pour générer *m* arbres multicast initiaux [T_1, T_2, …, T_m]. Afin d'éviter la génération des boucles ou cycles, une vérification est effectuée à chaque fois qu'un nœud est ajouté.

3.4.2.3 Réparation de la solution

La solution obtenue peut être illégale ou infaisable pour cinq raisons possibles :

1) L'arbre de diffusion multicast peut contenir des liens qui n'existent pas dans le graphe original $G = (E, V)$.
2) Des cycles ou des boucles peuvent exister dans l'arbre multicast.
3) L'arbre multicast peut ne pas relier le nœud source avec tous les nœuds destination. Ce qui signifie que les routes du nœud source vers un ou plusieurs nœuds destinations sont déconnectés.
4) L'arbre multicast peut contenir des nœuds feuilles qui ne constituent pas des destinations.
5) L'arbre multicast peut ne pas satisfaire simultanément les contraintes du délai, de la gigue et du taux de perte de paquets.

Afin de transformer une solution illégale ou infaisable en une solution faisable, une procédure de réparation de la solution est nécessaire. La procédure de réparation de l'arbre de diffusion multicast est donnée comme suit :

1) Supprimer tous les arcs qui n'existent pas dans le graphe original.
2) Connecter le nœud source avec tous les nœuds destinations
3) Eliminer toutes les boucles qui existent dans le réseau multicast
4) Supprimer les nœuds feuilles qui ne constituent pas de destinations ainsi que leurs arcs adjacents
5) Remplacer, si possible, les chemins indirects par les chemins directs. Un chemin d'accès direct est un chemin qui connecte deux nœuds directement, tandis qu'un

chemin indirect est un chemin qui relie deux nœuds indirectement par d'autres nœuds intermédiaires.

Un exemple de réparation d'une solution illégale est donné dans la figure 3.3.

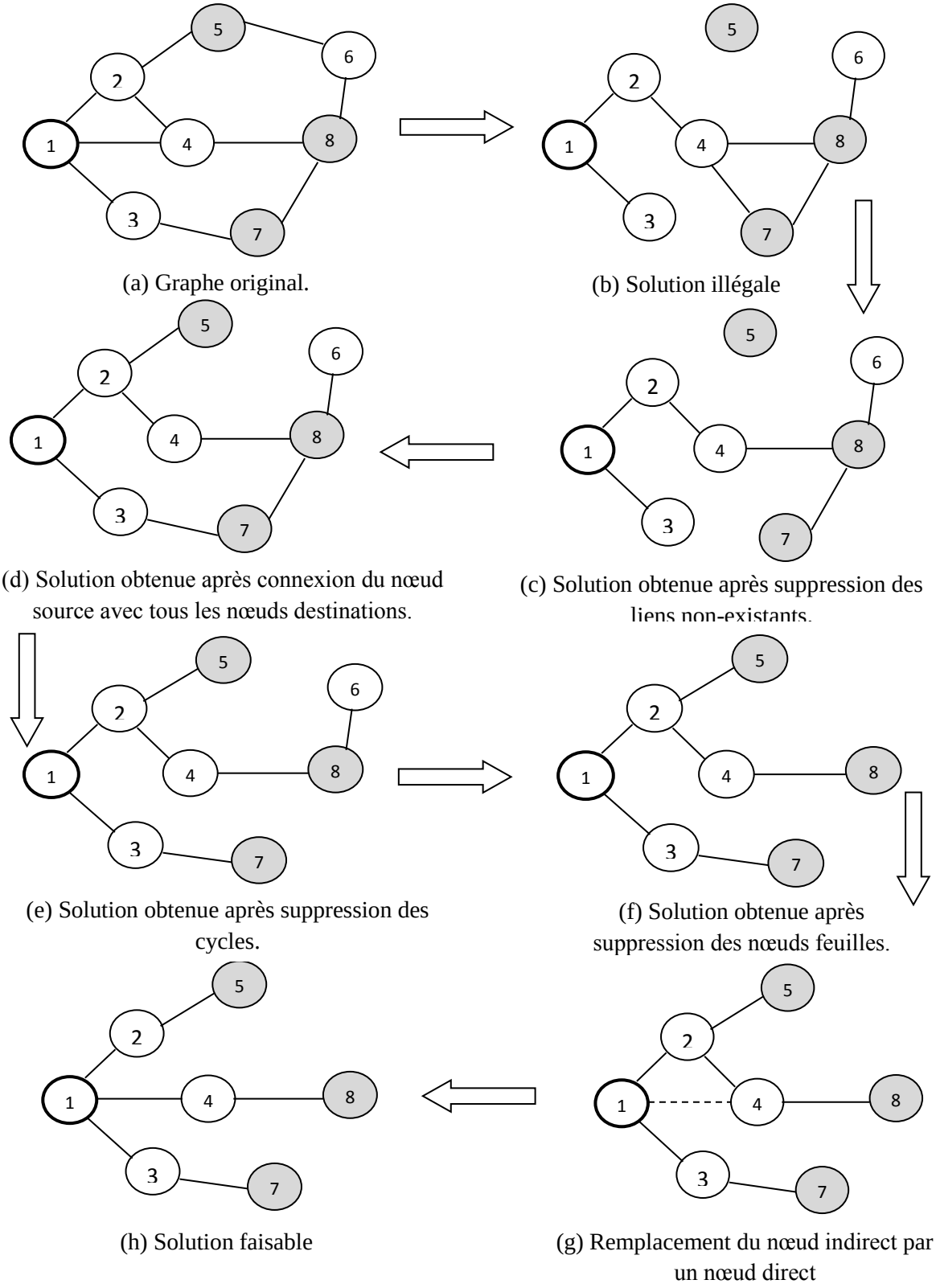


Figure 3.3: Exemple de réparation d'une solution illégale.

3.5 Résultats de simulation

Afin d'évaluer l'efficacité et la performance de nos algorithmes proposés ICBBA1 (utilisant la fonction logistique pour déterminer β) et ICBBA2 (utilisant la fonction tent pour déterminer β), nous avons mené une série de tests où nous avons comparé ICBBA1 et ICBBA2 avec les algorithmes GA, PSO, JPSO et BBA. Tous les algorithmes ont été implémentés en utilisant Visual C++. Les simulations sont exécutées sur un PC doté d'un processeur Core I3 2.27GHz, une RAM de 4Go et un système d'exploitation Windows 7. Les positions des nœuds sont fixées aléatoirement dans un rectangle de dimension 4000Km x 2400Km. Le modèle de topologie WAXMAN [86] est utilisé afin de générer le graphe aléatoirement. Dans ce modèle, l'arc est introduit entre deux nœud *i* et *j* avec une probabilité qui dépend de leur distance. La probabilité de l'arc est donnée par:
$P(i, j)= \beta exp(-l(i, j /\alpha L)$, où $l(i, j)$ est la distance euclidienne qui relie le nœud *i* au nœud *j* et *L* est la distance maximale entre deux nœuds quelconques dans le réseau. La valeur du paramètre α contrôle le nombre de liens courts dans le graphe généré aléatoirement. La valeur du paramètre β contrôle le nombre d'arcs dans le graphe. Nous fixons les valeurs de α et β à 0.8 et 0.7 respectivement dans notre configuration de simulation. Le coût, le délai, la gigue, la bande passante et le taux de perte de chaque lien sont répartis uniformément dans les intervalles [2, 10], [0, 30], [40, 80] et [0.0001, 0.01] respectivement. D_{max}, J_{max}, et LR_{max} sont fixés à 120 ms, 60 ms et 0.05 respectivement. La bande passante minimale requise est générée de façon aléatoire dans l'intervalle [40, 80]. Le nœud source et les nœuds destinations sont sélectionnés aléatoirement. Le coût moyen de l'arbre multicast est obtenu en effectuant 50 exécutions de chacun de ces algorithmes. Nous avons appliqué nos algorithmes pour deux types de réseaux sans fil, à savoir les WMNs et les VANETs.

3.5.1 Application pour les WMNs

Nous avons étudié les performances de nos algorithmes proposés dans deux scénarios différents

Dans le premier scénario, nous avons varié le nombre de nœuds de 10 à 100 par incrément de 10 nœuds et avec 10% de nœuds comme destinations afin d'évaluer la robustesse des algorithmes proposés en termes du coût, temps de convergence, délai, gigue et taux de perte de paquets en les comparant avec d'autres algorithmes qui existent dans la littérature. Les résultats sont donnés dans les figures 3.4 à 3.8.

La figure 3.4 montre le coût de l'arbre de diffusion multicast généré par chaque algorithme en variant le nombre de nœuds de 10 à 100 par incrément de 10 et avec 10% de nœuds comme destinations. Les résultats montrent que le coût de l'arbre multicast augmente pour chaque algorithme en augmentant le nombre de nœuds. Il en ressort que ICBBA2 a le coût le plus bas tandis que GA a le coût le plus élevé.

La figure 3.5 montre le temps de convergence de chaque algorithme en faisant varier le nombre de nœuds dans le réseau de 10 à 100 avec 10% de nœuds comme destinations. Les résultats révèlent clairement que le temps de convergence de chaque algorithme augmente en augmentant le nombre de nœuds. Les algorithmes proposés ont les temps de calcul les plus faibles par rapport à GA, PSO, JPSO et BBA.

Le délai de l'arbre multicast généré par chaque algorithme en faisant varier le nombre de nœuds du réseau avec 10 % des nœuds comme destinations est représenté dans la figure 3.6. Il est observé que le délai généré par ICBBA2 est le plus bas par rapport à ICBBA1 et les algorithmes existants.

La figure 3.7 représente la gigue de l'arbre multicast générée par chaque algorithme dans un réseau de 100 nœuds avec 10 % de nœuds comme destinations. Les résultats obtenus confirment notre conclusion sur ceux du délai. Comme pour le délai, ICBBA2 a la gigue la plus faible par rapport à la gigue d'ICBBA1, BBA, GA, PSO et JPSO.
La figure 3.8 montre le taux de perte de paquets des six algorithmes testés dans un réseau de 100 nœuds avec 10 % de nœuds comme destination. On observe que les algorithmes proposés ont les plus faibles taux de perte de paquets par rapport à GA, PSO, JPSO, et BBA.

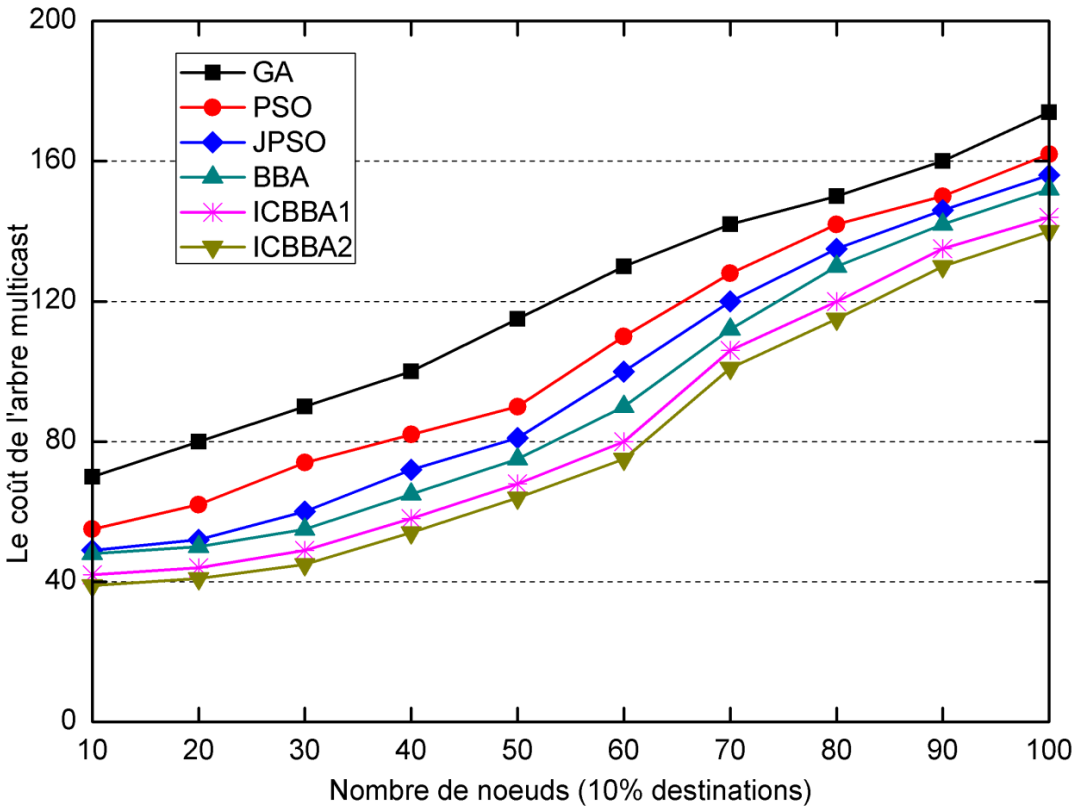


Figure 3.4: Coût de l'arbre multicast en variant le nombre de nœuds.

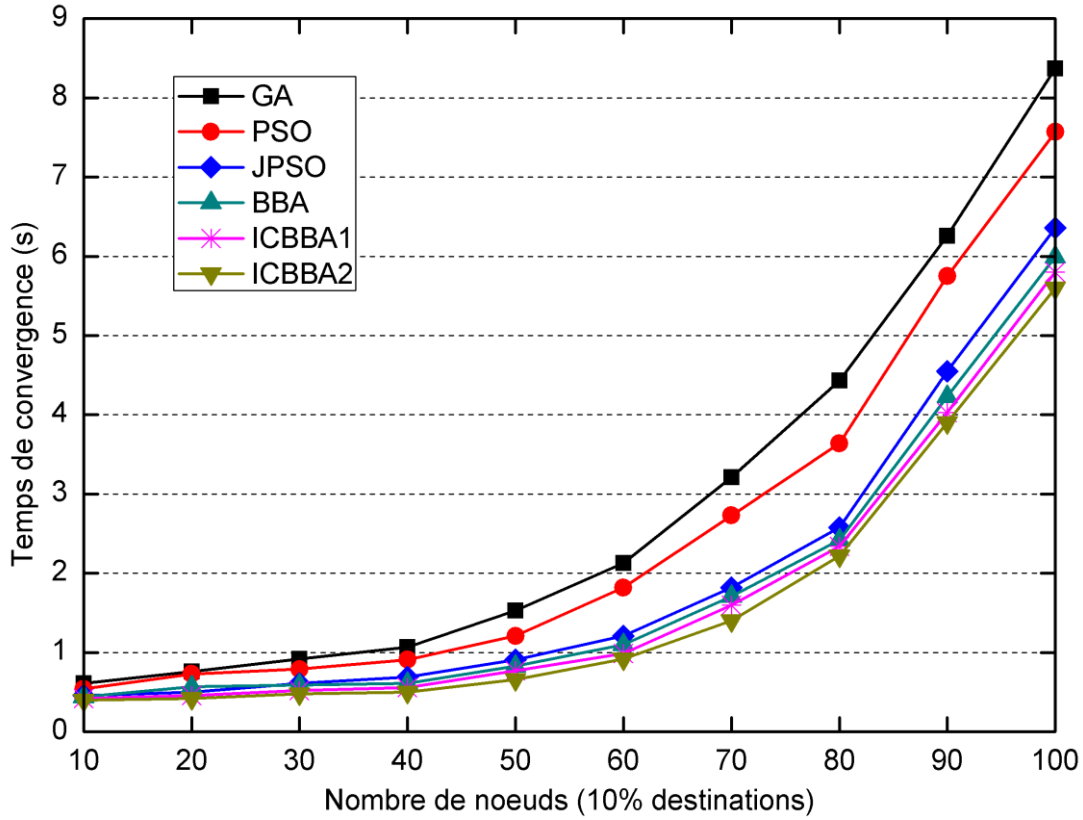


Figure 3.5: Temps de convergence de chaque algorithme en variant le nombre de nœuds.

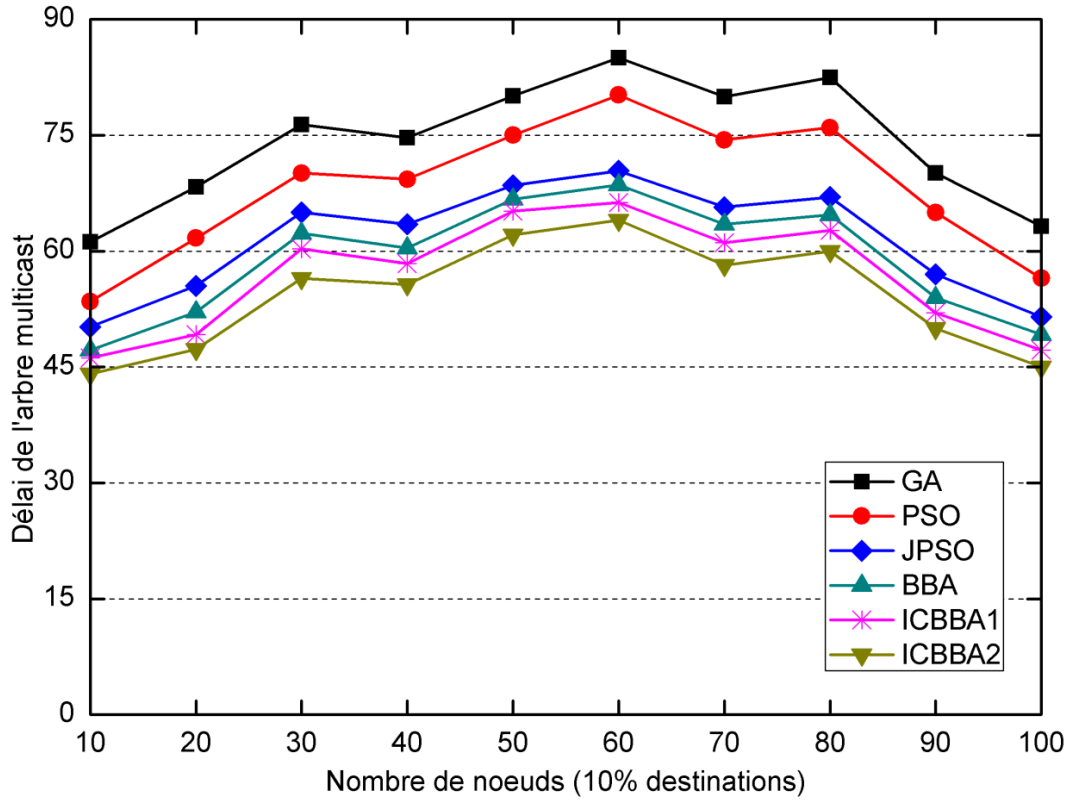


Figure 3.6: Délai de l'arbre multicast en variant le nombre de nœuds.

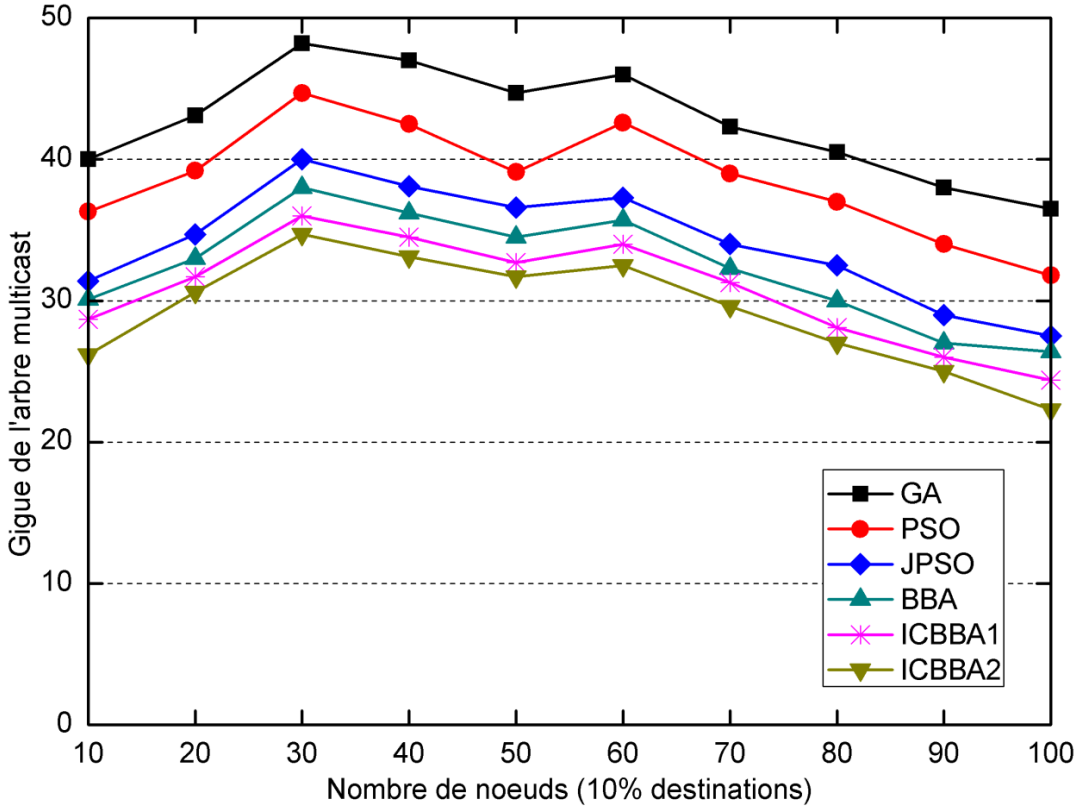


Figure 3.7: Gigue de l'arbre multicast en variant le nombre de nœuds.

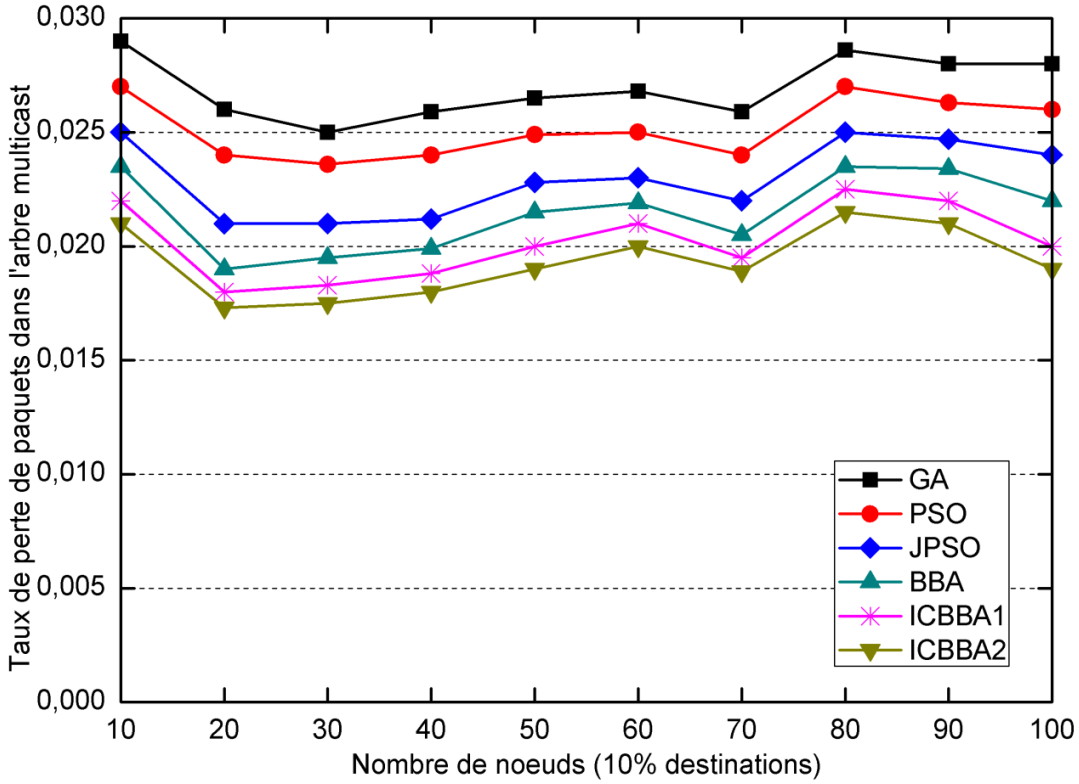


Figure 3.8: Taux de perte de paquets de l'arbre multicast en variant le nombre de nœuds.

Dans le deuxième scénario, afin d'évaluer les performances de nos algorithmes proposés en termes du coût de l'arbre multicast obtenu et du temps de convergence, nous avons varié le nombre de nœuds destinations de 10 % à 70% dans un réseau fixe de 100 nœuds. Les résultats sont donnés dans les figures 3.9 et 3.10.

La figure 3.9 représente le coût de l'arbre multicast généré par chaque algorithme en variant le nombre de nœuds destinations dans un réseau fixe de 100 nœuds. Nous constatons que le coût de l'arbre multicast pour chaque algorithme augmente avec l'augmentation du nombre de nœuds destinations. Nous observons également que les arbres de diffusion multicast générés par les algorithmes proposés ont les coûts les plus bas par rapport à BBA, GA, PSO, et JPSO.

La figure 3.10 montre le temps de convergence de chaque algorithme en variant le nombre de nœuds destinations de 10% à 70% dans un réseau de 100 nœuds. Les résultats montrent clairement que le temps de convergence de chaque algorithme augmente avec l'augmentation du nombre de nœuds. ICBBA2 consomme le moins de temps de calcul par rapport à ICBBA1, BBA, GA, PSO et JPSO.

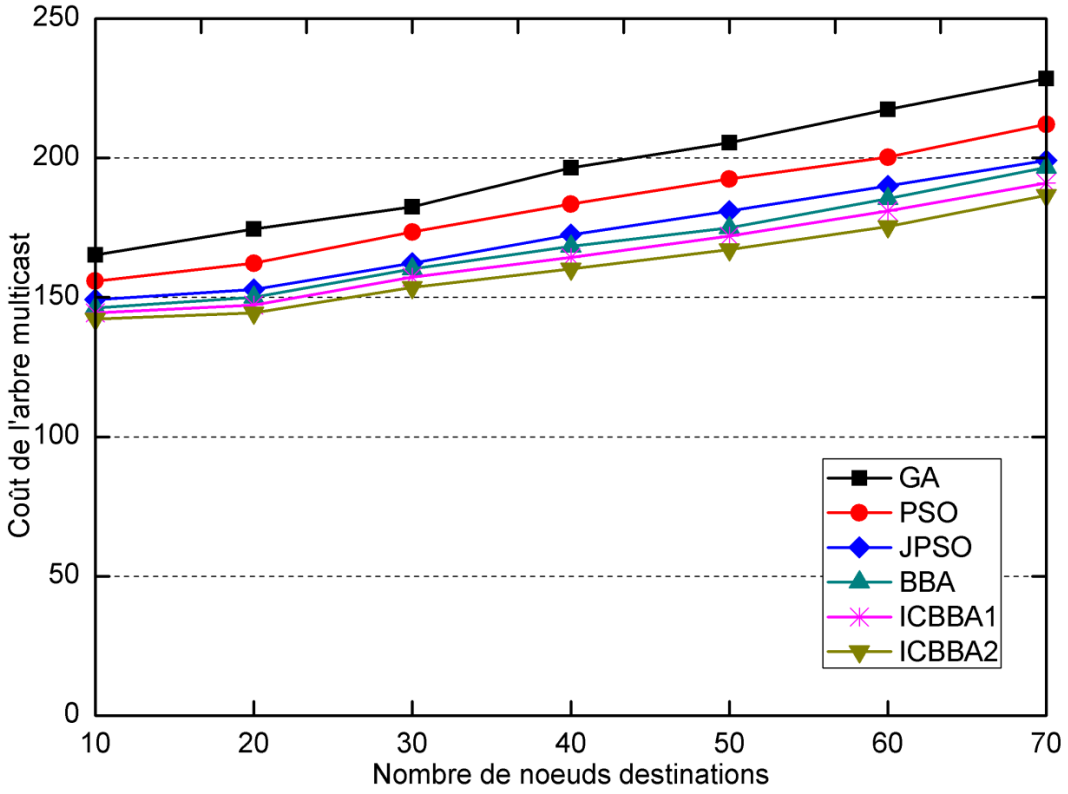


Figure3.9: Coût de l'arbre multicast en variant le nombre de nœuds destinations.

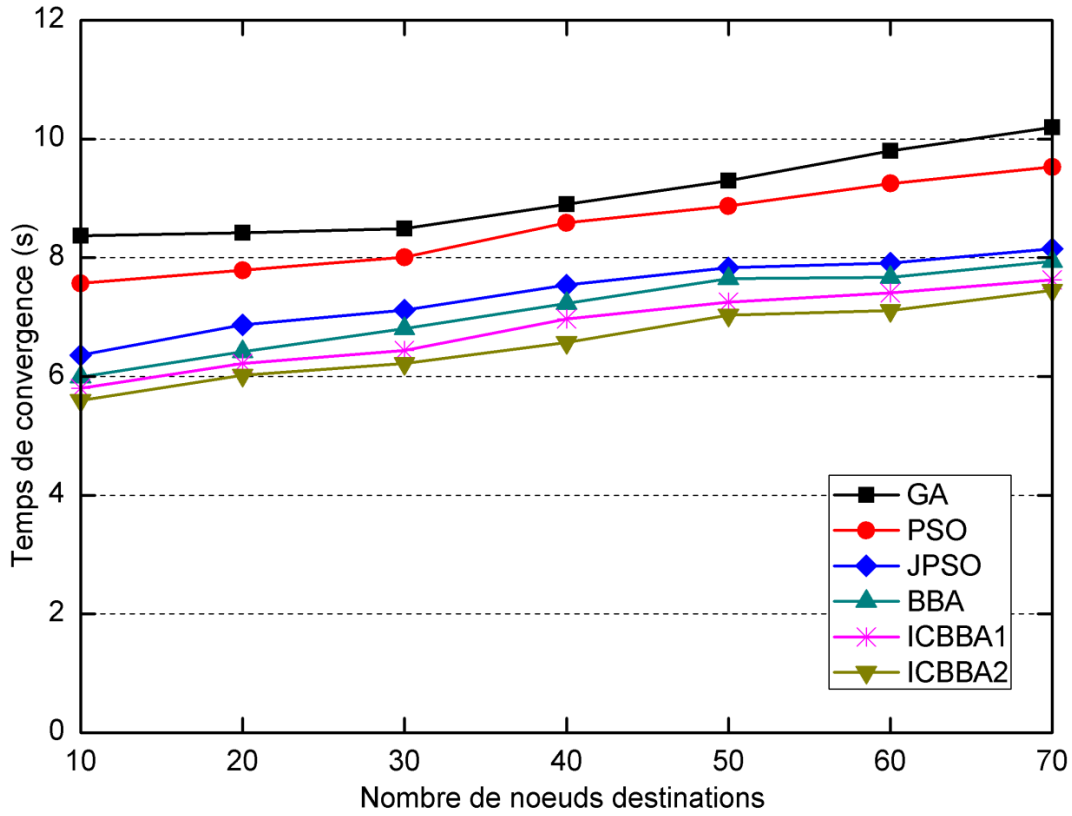


Figure 3.10: Temps de convergence de chaque algorithme en variant le nombre de nœuds destinations.

3.5.2 Application pour les VANETs

Comme dans les WMNs, nous allons étudier les performances de nos algorithmes proposés dans deux scénarios différents

Dans le premier scénario, nous avons varié le nombre de véhicules de 5 à 50 par incrément de 5 véhicules et avec 10% de véhicules comme destinations afin d'évaluer les performances des algorithmes proposés en termes du coût, temps de convergence, délai, gigue et taux de perte de paquets en les comparant avec d'autres algorithmes qui existent dans la littérature. Les résultats sont donnés dans les figures 3.11 à 3.15.

La figure 3.11 montre le coût de l'arbre de diffusion multicast généré par chaque algorithme en faisant varier le nombre de véhicules dans le réseau de 5 à 50 avec 10% de véhicules comme destinations. Les résultats révèlent clairement que le coût de l'arbre multicast augmente pour chaque algorithme en augmentant le nombre de véhicules. Les algorithmes proposés ont les coûts les plus faibles par rapport à GA, PSO, JPSO et BBA.

Le temps de convergence de chaque algorithme en faisant varier le nombre de véhicules du réseau de 5 à 50 avec 10 % des véhicules comme destinations est représenté dans la figure 3.12. Il est observé que ICBBA2 a le temps de calcul le plus faible par rapport à ICBBA1 et les algorithmes existants.

La figure 3.13 montre le délai de l'arbre de diffusion multicast généré par chaque algorithme en variant le nombre de véhicules de 5 à 50 par incrément de 10 et avec 10% de véhicules comme destinations. Les résultats montrent que ICBBA2 a le délai le plus faible par rapport au délai d'ICBBA1, BBA, GA, PSO et JPSO.

La figure 3.14 représente la gigue de l'arbre multicast générée par chaque algorithme dans un réseau de 50 véhicules avec 10 % de véhicules comme destinations. Les résultats obtenus confirment notre conclusion sur ceux du délai. Comme pour le délai, ICBBA2 a la gigue la plus faible tandis que GA a la gigue la plus élevée.
La figure 3.15 montre le taux de perte de paquets des six algorithmes testés dans un réseau de 50 véhicules avec 10 % de véhicules comme destination. On observe que les algorithmes proposés ont les plus faibles taux de perte de paquets par rapport à GA, PSO, JPSO, et BBA.

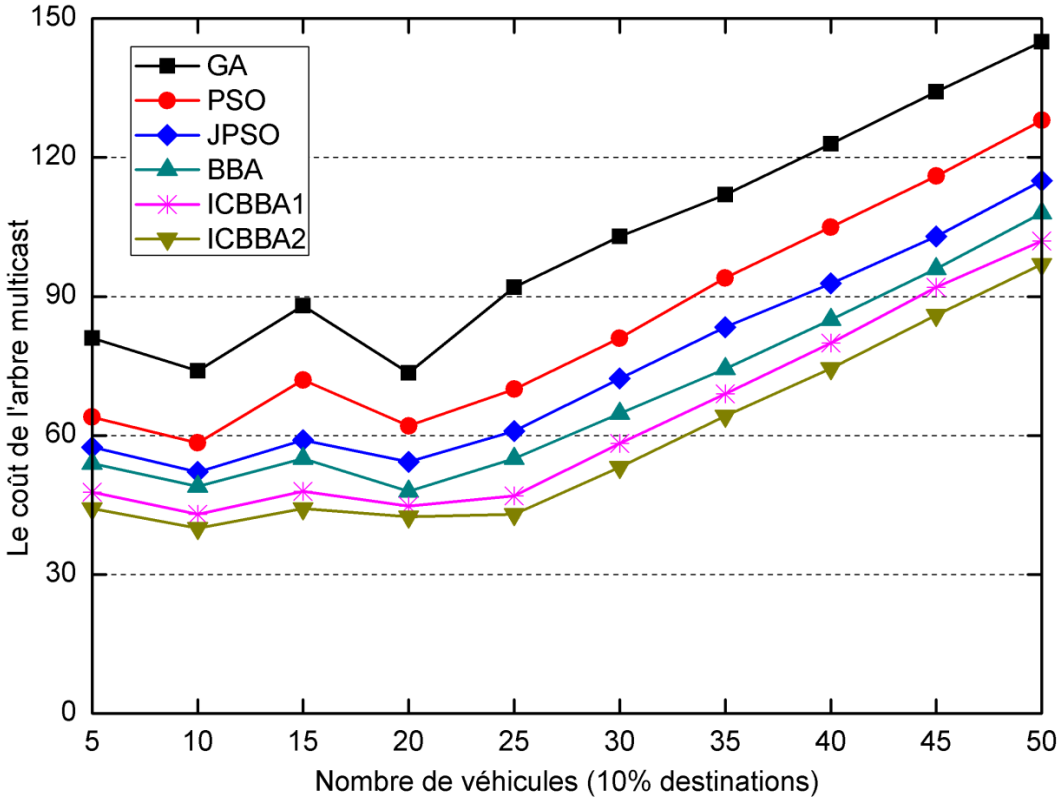


Figure3.11: Coût de l'arbre multicast en variant le nombre de véhicules.

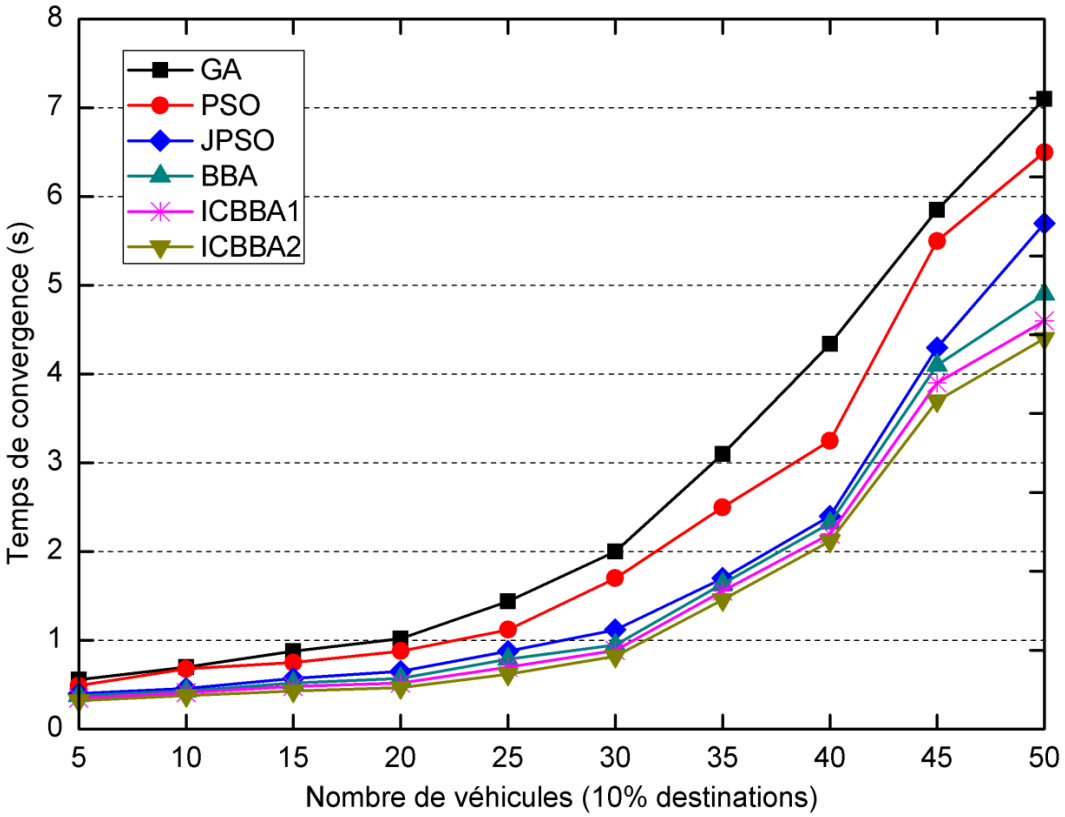


Figure 3.12: Temps de convergence de chaque algorithme en variant le nombre de véhicules.

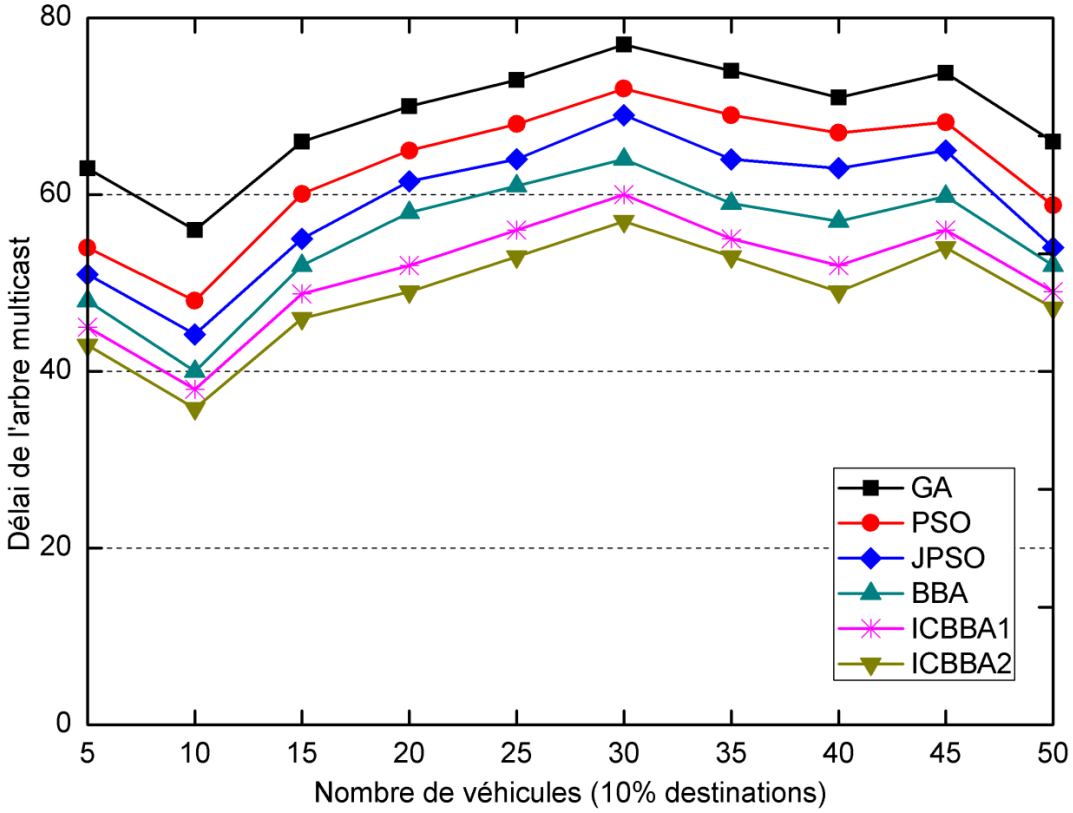


Figure 3.13: Délai de l'arbre multicast en variant le nombre de véhicules.

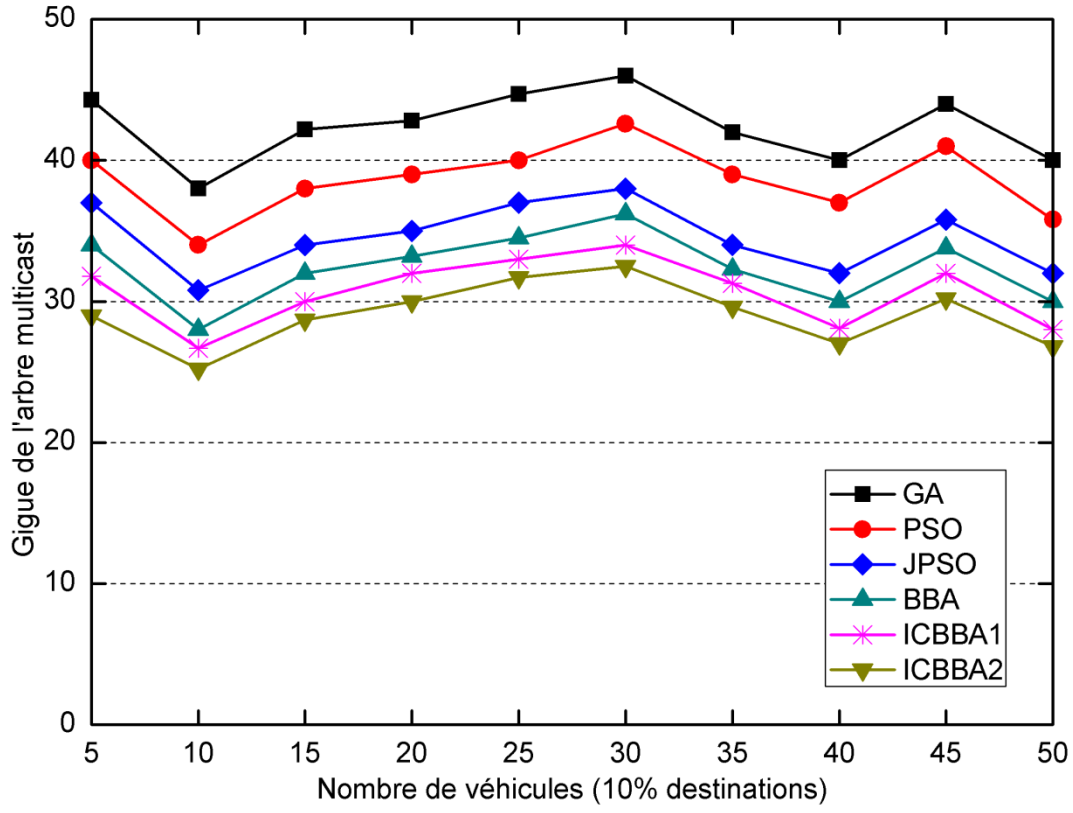


Figure 3.14: Gigue de l'arbre multicast en variant le nombre de véhicules

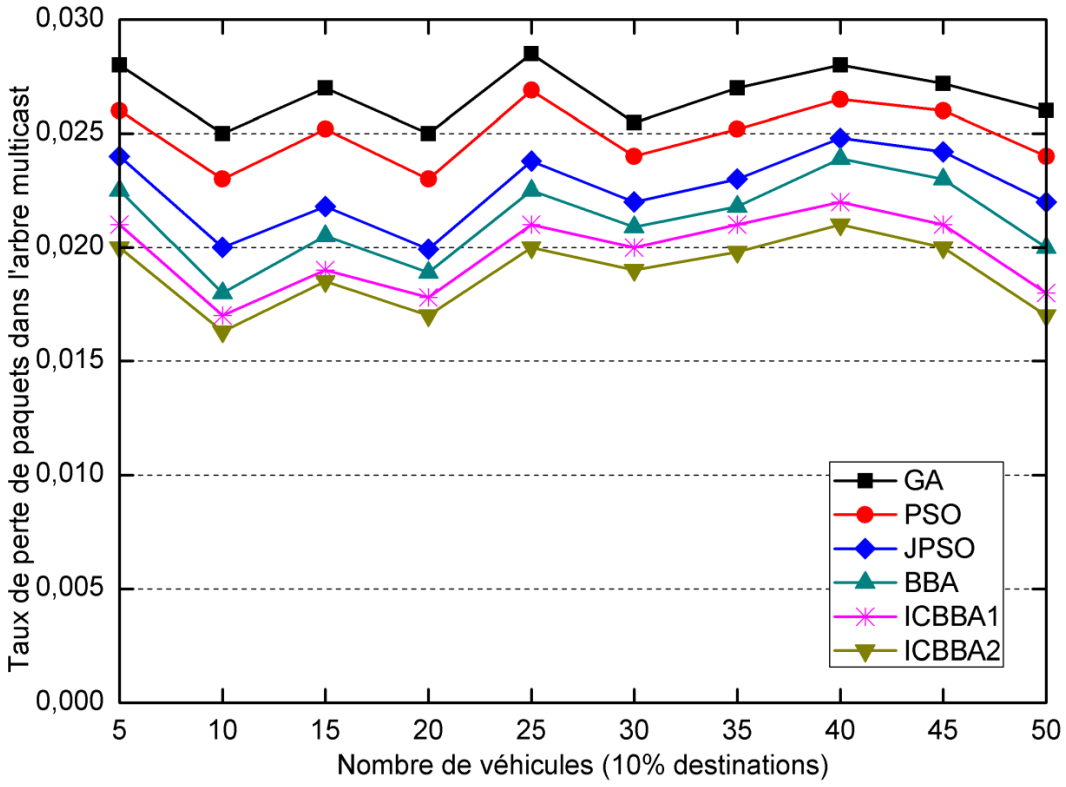


Figure 3.15: Taux de perte de paquets de l'arbre multicast en variant le nombre de véhicules.

Dans le deuxième scénario, nous avons varié le nombre de véhicules destinations de 10 % à 70% dans un réseau fixe de 50 véhicules afin d'évaluer les performances de nos algorithmes proposés en termes du coût de l'arbre multicast obtenu et du temps de convergence. Les résultats sont donnés dans les figures 3.16 et 3.17.

La figure 3.16 montre le coût de l'arbre multicast généré par chaque algorithme en faisant varier le nombre de véhicules destinations de 10 % à 70% dans un réseau fixe de 50 véhicules. Nous constatons que le coût de l'arbre multicast pour chaque algorithme augmente avec l'augmentation du nombre de nœuds destinations. Nous observons également que ICBBA2 a le coût le plus bas tandis que GA a le coût le plus élevé.

La figure 3.17 montre le temps de convergence de chaque algorithme en variant le nombre de véhicules destinations de 10% à 70% dans un réseau de 50 véhicules. Les résultats montrent clairement que le temps de convergence de chaque algorithme augmente avec l'augmentation du nombre de nœuds. ICBBA2 consomme le moins de temps de calcul par rapport à ICBBA1, BBA, GA, PSO et JPSO.

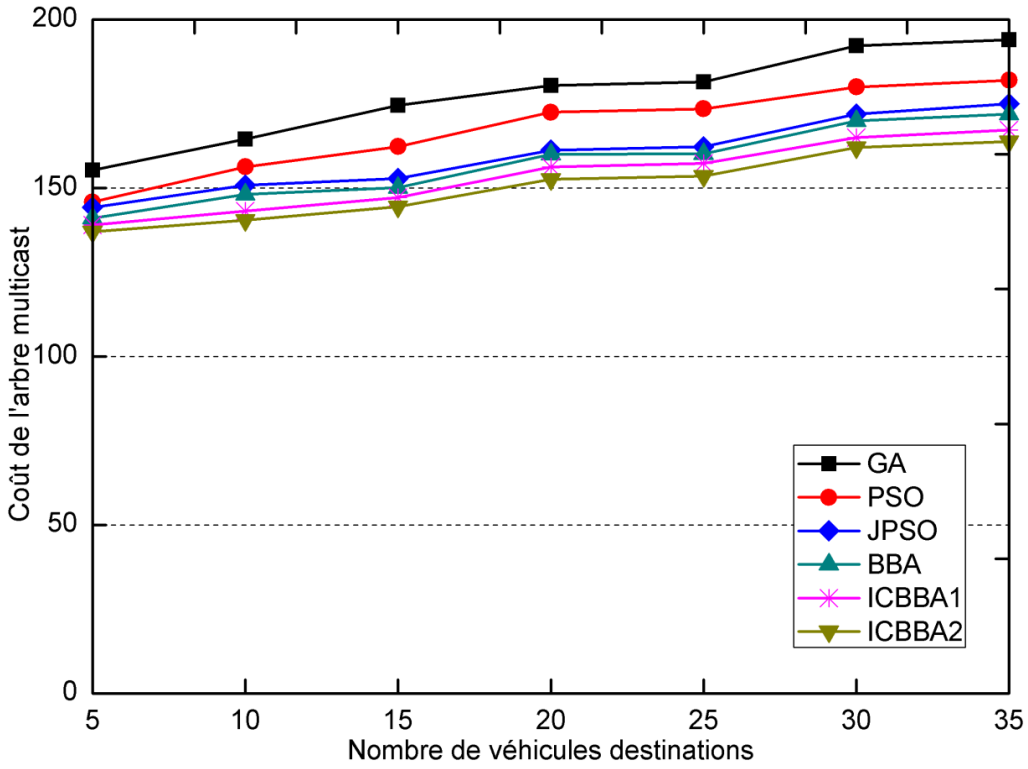


Figure3.16: Coût de l'arbre multicast en variant le nombre de véhicules destinations.

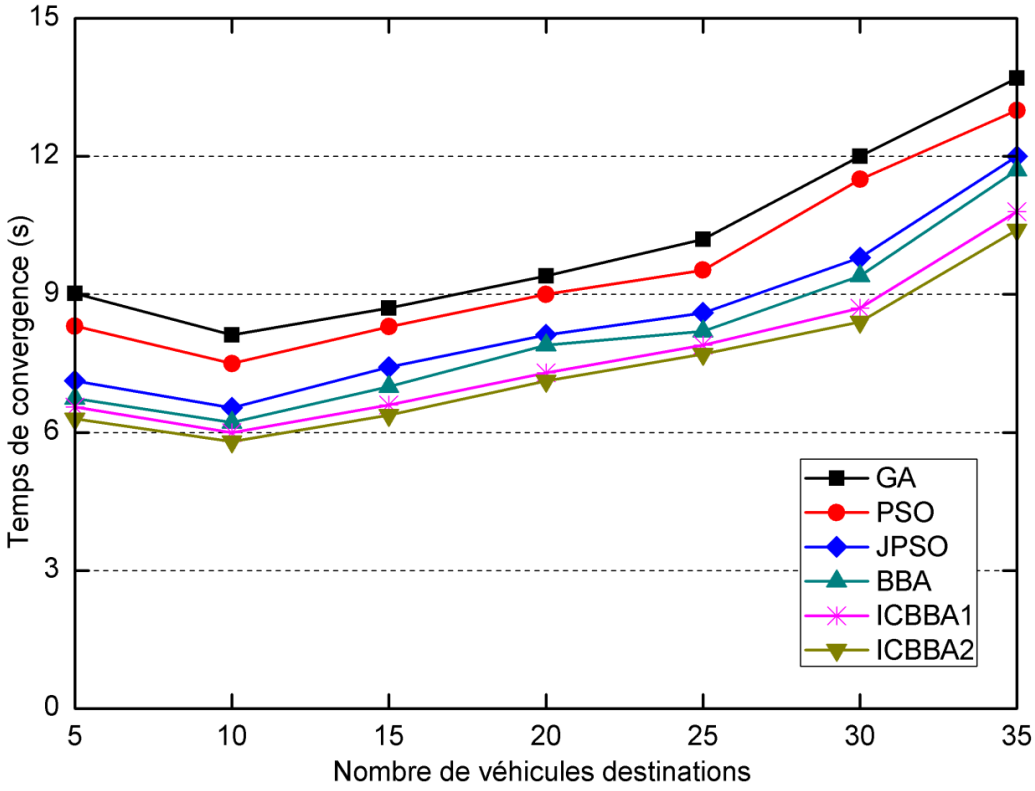


Figure 3.17: Temps de convergence de chaque algorithme en variant le nombre de véhicules destinations.

3.6 Conclusion

Dans ce chapitre, nous avons présenté deux nouveaux algorithmes proposés ICBBA1 (utilisant la fonction logistique pour déterminer β) et ICBBA2 (utilisant la fonction tent pour déterminer β). Ces algorithmes sont appliqués pour résoudre le problème de routage multicast avec QoS pour deux types de réseaux sans fil, à savoir les WMNs et les VANETs. Les résultats expérimentaux montrent que ICBBA2 est la meilleure solution dans la plupart des cas pour les deux types de réseaux sans fil.

Parmi les limitations des métaheuristiques nous citons le déséquilibre entre l'exploration et l'exploitation de l'espace de recherche des solutions, il est alors difficile de faire la distinction entre une solution optimale et une autre globale. L'hybridation des métaheuristiques permet d'avoir cet équilibre et assure la diversité des solutions.

Dans le chapitre suivant, nous allons proposer une nouvelle approche basée sur l'hybridation de ICBBA et l'algorithme Quantum évolutionnaire pour résoudre le problème de routage multicast avec QoS.

.

CHAPITRE 4

APPROCHES HYBRIDES POUR LE ROUTAGE MULTICAST: APPLICATION EN WMNs et VANETs

4.1 Introduction

Ce chapitre a pour objet de présenter une hybridation de l'algorithme des chauves-souris binaire BBA avec l'algorithme évolutionnaire quantique QEA pour résoudre le problème de routage multicast avec QoS.

Deux approches hybrides basées sur l'hybridation de BBA avec QEA ont été proposées pour améliorer les capacités d'exploration de l'espace de recherche et augmenter la diversité des solutions. La première approche, nommée BBAQEA1, est basée sur l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. La deuxième approche, nommée BBAQEA2, est basée sur le remplacement de l'opérateur évolutionnaire quantique de QEA par l'équation d'évolution de BBA.

Dans ce chapitre, nous présentons un rappel des méthodes et approches hybrides proposées dans la littérature pour résoudre le problème de routage multicast, nous décrivons ensuite la formulation du problème de routage multicast, nous présentons également en détail les deux approches hybrides BBAQEA1 et BBAQEA2 proposées pour résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETS pour enfin évaluer leurs performances et comparer leurs caractéristiques avec l'algorithme ICBBA2.

4.2 Approches d'optimisation hybrides pour le routage multicast avec QoS

Un certain nombre d'approches d'optimisation hybrides ont été proposées dans la littérature pour résoudre le problème de routage multicast. Li et al. [87] ont proposé une approche basée sur l'hybridation de l'algorithme génétique (GA) avec l'algorithme d'optimisation d'essaims particules (PSO), appelée HGAPSO, pour résoudre le problème de routage multicast. HGAPSO combine les fonctions de génération des nouveaux individus des deux algorithmes. Les résultats de simulation ont montré que HGAPSO a de meilleures performances par rapport à GA et PSO. Abdel-kader [76] a proposé une méthode basée sur l'hybridation de l'algorithme génétique avec l'algorithme d'optimisation d'essaims particules pour résoudre le problème de routage multicast avec contraintes multiples. La méthode combine GA avec PSO pour fournir une recherche efficace de l'espace de solutions et réaliser l'équilibre entre la sélection des individus et le partage de bonnes connaissances. Les résultats de simulation ont montré l'efficacité et la supériorité de la méthode proposée sur les algorithmes basés sur PSO et GA. Wang et al. [88] ont proposé une méthode d'optimisation basée sur l'hybridation de PSO avec l'algorithme d'optimisation basé sur l'arborescence pour résoudre le problème de routage multicast avec contraintes multiples. Les résultats expérimentaux ont révélé les meilleures performances de l'approche proposée en termes de recherche, temps de convergence et capacité d'adaptation par rapport à d'autres algorithmes existants dans la littérature. Xi-Hong et al. [89] ont proposé une approche basée sur l'hybridation de ACO avec PSO pour résoudre le problème de routage multicast. Dans cette approche, la mise à jour de la position de PSO est utilisée pour réguler la solution générée par ACO. Les résultats de simulation ont montré que l'algorithme proposé a de meilleures performances par rapport à d'autres algorithmes tels que GA, ACO et PSO. Zhang et al. [90] ont proposé une nouvelle approche basée sur l'hybridation de l'algorithme génétique avec l'algorithme du recuit simulé (SA) pour résoudre le problème de routage multicast avec QoS. Les résultats de simulation ont montré que l'algorithme proposé a le coût le plus bas et le temps de convergence le plus rapide par rapport à GA et SA. Xing et al. [91] ont proposé un nouvel algorithme évolutif parallèle basé sur l'hybridation de l'algorithme évolutionnaire et l'algorithme quantique, appelé MEQGA, pour résoudre le problème de routage multicast dans les réseaux WDM (Wavelength Division Multiplexing). MEQGA utilise le mécanisme d'évolution multi-granularité, la stratégie de rotation quantique et une mutation adaptative quantique pour éviter la recherche locale. Les résultats de simulation ont montré que MEQGA surpasse d'autres

algorithmes existants dans la littérature en termes de robustesse, taux de réception, convergence rapide et capacité de recherche globale. Tableau 4.1 présente une synthèse des travaux sur le routage multicast avec QoS basés sur des approches hybrides.

Algorithmes	**Références**	**Fonction objectif (coût)**	**Délai**	**gigue**	**Bande passante**	**Taux de perte**
HGAPSO	Li et al. [87]	*				
HACOPSO	Xi-Hong et al. [89]	*	*	*	*	*
PSOTREE	Wang et al. [88]	*	*		*	
MEQGA	Xing et al. [91]	*	*	*	*	*
HGASA	Zhang et al. [90]	*	*	*	*	
HPSOGA	Abdel-kader [76]	*	*	*	*	

Tableau 4.1: Synthèse des travaux sur le routage multicast avec QoS basés sur des approches hybrides.

4.3 Formulation du problème de routage multicast

Le réseau de communication est modélisé et présenté, comme dans le chapitre 3, par un graphe orienté et pondéré $G = (V, E)$, où V dénote l'ensemble des nœuds et E dénote l'ensemble des arcs du graphe représentant les liens sans fil entre les nœuds. $|V| = n$ est le nombre de nœuds et $|E| = l$ est l'ensemble d'arcs du réseau. Chaque arc $e = (i, j) \in E$ qui relie le nœud i avec le nœud j est associé à un coût de l'arc $Cost(e): E \rightarrow R^+$, délai de l'arc $Delay(e): E \rightarrow R^+$, gigue de l'arc $Jitter(e): E \rightarrow R^+$, bande passante de l'arc $Bandwidth(e): E \rightarrow R^+$, taux de perte de paquets de l'arc $LR(e): E \rightarrow R^+$, où R^+ désigne l'ensemble de tous les nombre réels positifs.

Ainsi, comme dans le chapitre 3, le problème de routage multicast avec qualité de service (QoS) peut être formulé comme suit :

$$Minimiser\ Cost\ (T(s, M)) \qquad (4.1)$$

Sous les contraintes:

$$Delay(P_T(s, m)) \leq D_{max} \quad (4.2)$$

$$Jitter(P_T(s, m)) \leq J_{max} \quad (4.3)$$

$$Min(Bandwidth(P_T(s, m))) \geq B_{min} \quad (4.5)$$

$$LR(P_T(s, m)) \leq LR_{max} \quad (4.6)$$

4.4 Hybridation de BBA avec QEA pour le routage multicast

Afin d'améliorer les capacités d'exploration de l'espace de recherche et assurer la diversité de la population, deux approches hybrides basées sur l'hybridation de BBA avec QEA ont été proposées. La première approche, nommée BBAQEA1, est basée sur l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. La deuxième approche, nommée BBAQEA2, est basée sur le remplacement de l'opérateur évolutionnaire quantique de QEA par l'équation d'évolution de BBA.

4.4.1 Algorithme évolutionnaire quantique (QEA: Quantum Evolutionary Algorithm)

L'algorithme évolutionnaire inspiré du quantique ou l'algorithme évolutionnaire quantique (QEA), proposé par Kuk Hyuan Han en 2002 [92], est un nouvel algorithme d'optimisation de recherche probabiliste basé sur le concept et le principe de la théorie du calcul quantique. Il utilise le bit quantique ou qubit pour représenter l'état probabiliste des individus. Le qubit est l'unité élémentaire d'information la plus petite, il peut être à l'état 0, l'état 1 ou dans une superposition des deux états (entre 0 et 1) [92-94]. L'état du qubit est décrit par l'équation (4.7) [95].

$$|\Psi >= \alpha|0 > +\beta|1 > \quad (4.7)$$

Où α et β sont des nombres complexes qui spécifient les amplitudes des états correspondants. La mesure d'un qubit peut donner la valeur 0 avec la probabilité $|\alpha^2|$ et peut aussi donner la valeur 1 avec la probabilité $|\beta^2|$ de telle sorte que $|\alpha^2| + |\beta^2| = 1$. Dans QEA, un individu peut être représenté par une chaine de *m* qubits en mesure de représenter 2^m états. L'i^{ème} qubit Q_i^t est donné par l'équation:

$$Q_i^t = \begin{bmatrix} \alpha_{i1}^t | & \alpha_{i2}^t | & \dots | & \alpha_{im}^t \\ \beta_{i1}^t | & \beta_{i2}^t | & \dots | & \beta_{im}^t \end{bmatrix} \quad (4.8)$$

Où *m* est le nombre de qubits représentant la longueur de l'individu Q_i^t.

Le pseudo-code de l'algorithme évolutionnaire quantique original (QEA) est illustré dans l'algorithme 4.1.

1 : *t := 0*
2 : Initialiser *Q(t)*
3 : Produire P(t) par observation de *Q(t)*
4 : Evaluer *P(t)*
5 : Stocker le meilleur individu (la meilleure solution) parmi *P(t)* dans *B(t)*
6 : **Tant que** (*critère d'arrêt non satisfait*) **faire**
7 : *t := t+1*
8 : Produire *P(t)* par observation de *Q(t-1)*
9 : Evaluer *P(t)*
10 : Mettre à jour *Q(t-1)* en utilisant la porte quantique *U(t)*
11 : Stocker la meilleure solution
12 : **Fin tant que**

Algorithme 4.1: Pseudo-code de l'algorithme évolutionnaire quantique.

QEA comporte quatre étapes principales : initialiser *Q(t),* observer *Q(t),* évaluer *Q(t)* et mettre à jour Q(t).

Premièrement, dans l'étape d'initialisation de *Q(t),*α_{ij}^t et β_{ij}^t, *j= 1, 2, ...m,* de tous les m qubits Q_i^t, *i=1, 2, ...,n* dans *Q(t)* sont fixés à $\frac{1}{\sqrt{2}}$, ce qui signifie qu'un individu qubit Q_i^t représente la superposition linéaire de tous les états possibles avec la même probabilité à 0 ou 1.

Deuxièmement, l'étape observer *Q(t)* produit un ensemble de solutions binaires *P(t)*= X_1^t, X_2^t, ..., X_n^t en observant l'état de *Q(t)* (*Q(t)*= Q_1^t, Q_2^t, ..., Q_n^t) à la génération t où n est la taille de la population. L'i^{ème} solution binaire X_i^t représentée par une chaine de longueur m est générée en fonction des probabilités de $|\alpha_{ij}^t|^2$ ou $|\beta_{ij}^t|^2$ respectivement avec *j= 1, 2, ..., m*.

Procédure produire *P(t)*

1: *r=0*
2: **Début**
3: **Pour** *i* allant de 1 à *n* **faire**
4: **Tant que** (*i* < *m*) **faire**
5: *i=i+1*
6: **Si** rand(0,1) > $|\alpha_{ji}^t|^2$ **alors**
7: $x_{ji}^t = 1$
8: **Sinon**
9: $x_{ji}^t = 0$
10: **End**
11: **End**
12: **End**

Troisièmement, dans l'étape d'évaluation de *P(t)*, chaque solution binaire X_i^t est évaluée pour donner la valeur de la fitness et la meilleure solution X_{best}^t parmi les solutions binaires *P(t)* est stockée dans *B(t)*. Quatrièmement, dans l'étape de mise à jour de *Q(t)*. L'individu qubit est mis à jour en utilisant des portes quantiques appropriées *U(t)* définies comme des opérateurs de variation de QEA de telle sorte que les états des individus qubits sont générés. Les portes quantiques les plus couramment utilisées sont la porte de négation, la porte de négation contrôlée, la porte de rotation, la porte de Hadamard [96,97]. La porte de rotation appliquée à de nombreuses métaheuristiques [92, 97-100] est adoptée dans notre étude, elle est donnée par l'équation suivante:

$$U(\theta_i^t) = \begin{bmatrix} \cos(\theta_i^t) & -\sin\theta_i^t \\ \sin\theta_i^t & \cos(\theta_i^t) \end{bmatrix} \tag{4.9}$$

Où θ_i^t est l'angle de rotation de Q_i^t et peut être défini par différentes méthodes. La valeur du i^{ème} qubit $(\alpha_{ij}^t, \beta_{ij}^t)$ de Q_i^t est mise à jour par l'équation (4.10).

$$\begin{bmatrix} \alpha_{ij}^{t+1} \\ \beta_{ij}^{t+1} \end{bmatrix} = U(\theta_i^t) \begin{bmatrix} \alpha_{ij}^t \\ \beta_{ij}^t \end{bmatrix} = \begin{bmatrix} \cos(\theta_i^t) & -\sin(\theta_i^t) \\ \sin(\theta_i^t) & \cos(\theta_i^t) \end{bmatrix} \begin{bmatrix} \alpha_{ij}^t \\ \beta_{ij}^t \end{bmatrix} \tag{4.10}$$

Ce qui donne :

$$\begin{cases} \alpha_{ij}^{t+1} = \alpha_{ij}^{t} \cos(\theta_i^t) + \beta_{ij}^{t} \sin(\theta_i^t) \\ \beta_{ij}^{t+1} = \beta_{ij}^{t} \cos(\theta_i^t) - \alpha_{ij}^{t} \sin(\theta_i^t) \end{cases} \quad (4.11)$$

Le processus est répété jusqu'à ce que la condition soit satisfaite.

4.4.2 BBA intégré avec QEA (BBAQEA1)

L'idée principale de la première approche proposée, appelée BBAQEA1, est l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. Dans BBAQEA1, l'angle de rotation des portes quantiques utilisé pour mettre à jour *Q(t)* est ajusté par l'équation d'évolution de BBA. Donc la méthode de la table de consultation look-up classique n'est pas nécessaire. Un ajustement correct de l'angle de rotation joue un rôle très important dans la mise à jour de *Q(t)*. La nouvelle méthode qui consiste à ajuster l'i^{ème} angle de rotation θ_i^t peut être décrite par l'équation (4.12).

$$\theta_{ij}^{t} = \Delta\theta * (v_{ij}^{t-1} + (x_{ij}^{t-1} - x_{Gbest}) f_{ij} \quad (4.12)$$

$\Delta\theta$ représente la grandeur de l'angle de rotation qui diminue d'une manière monotone de $\Delta\theta_{max}$ à $\Delta\theta_{min}$. La procédure de mise à jour de *Q(t)* peut être alors redéfinie par l'équation (4.13).

$$\begin{cases} \theta_{ij}^{t} = \Delta\theta * (v_{ij}^{t-1} + (x_{ij}^{t-1} - x_{Gbest}) f_{ij} \\ \begin{bmatrix} \alpha_{ij}^{t+1} \\ \beta_{ij}^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i^t) & -\sin(\theta_i^t) \\ \sin(\theta_i^t) & \cos(\theta_i^t) \end{bmatrix} \begin{bmatrix} \alpha_{ij}^{t} \\ \beta_{ij}^{t} \end{bmatrix} \end{cases} \quad (4.13)$$

Le pseudo-code de l'approche hybride BBAQEA1 est illustré dans l'algorithme 4.2. Les principaux changements par rapport à l'algorithme QEA sont soulignés.

1 : *t := 0*

2 : Initialiser *Q(t)*

3 : Produire *P(t)* par observation de *Q(t)*

4 : Evaluer *P(t)*

5 : Réparer les solutions

6 : Stocker le meilleur individu (la meilleure solution) parmi *P(t)* dans *B(t)*

7 : **Tant que** (*critère d'arrêt non satisfait*) **faire**

8 : *t := t+1*

9 : Produire *P(t)* par observation de *Q(t-1)*

10 : Evaluer *P(t)*

11 : Réparer les solutions

12 : Mettre à jour *Q(t-1)* en utilisant l'équation (4.13)

13 : Stocker la meilleure solution

14 : **Fin tant que**

Algorithme 4.2: Pseudo-code de l'approche hybride BBAQEA1.

4.4.3 BBA remplacé par QEA (BBAQEA2)

L'idée principale de la deuxième approche proposée, appelée BBAQEA2, est que l'opérateur évolutionnaire quantique de QEA est remplacé par l'équation d'évolution de BBA. Afin de permettre à BBA de mettre à jour l'individu qubit automatiquement, nous allons utiliser l'angle de rotation pour coder le qubit. Les nombres complexes α_{im}^t et β_{im}^t peuvent être exprimés en $\sin(\theta_{im}^t)$ et $\cos(\theta_{im}^t)$ respectivement. Alors le qubit $\begin{bmatrix}\alpha_{im}^t \\ \beta_{im}^t\end{bmatrix}$ est présenté par $\begin{bmatrix}\sin(\theta_{im}^t) \\ \cos(\theta_{im}^t)\end{bmatrix}$. L'i^{ème} individu qubit $Q_i^t = \begin{bmatrix}\alpha_{i1}^t|\ \alpha_{i2}^t|\ \ldots|\ \alpha_{im}^t \\ \beta_{i1}^t|\ \beta_{i2}^t|\ \ldots|\ \beta_{im}^t\end{bmatrix}$ est remplacé par $\begin{bmatrix}\sin(\theta_{i1}^t)|\ \sin(\theta_{i2}^t)|\ \cdots\ |\sin(\theta_{im}^t) \\ \cos(\theta_{i1}^t)|\ \cos(\theta_{i2}^t)|\ \cdots\ |\cos(\theta_{im}^t)\end{bmatrix}$. La stratégie d'évolution de BBA est utilisée pour mettre à jour l'angle de rotation à la place des portes quantiques traditionnelles. L'angle de rotation est mis à jour en utilisant l'équation (4.14).

$$\theta_{ij}^{t+1} = (\theta_{ij}^t + (x_{ij}^t - x_{Gbest})f_{ij} \qquad (4.14)$$

Donc, la procédure de mise à jour de *Q(t)* peut être redéfinie comme suit :

$$\begin{cases} \theta_{ij}^{t+1} = (\theta_{ij}^{t} + (x_{ij}^{t} - x_{Gbest}) f_{ij} \\ \begin{bmatrix} \alpha_{ij}^{t+1} \\ \beta_{ij}^{t+1} \end{bmatrix} = \begin{bmatrix} \sin(\theta_{i1}^{t})| & \sin(\theta_{i2}^{t})| & \cdots & |\sin(\theta_{im}^{t}) \\ \cos(\theta_{i1}^{t})| & \cos(\theta_{i2}^{t})| & \cdots & |\cos(\theta_{im}^{t}) \end{bmatrix} \end{cases} \quad (4.15)$$

Afin d'éliminer la recherche répétitive et améliorer l'efficacité de BBAQEA2, θ_{ij}^{t} est limité dans le premier quadrant. La raison est que la plage des valeurs de α_{ij}^{t} est la même dans les quatre quadrants. θ_{ij}^{t} est dans l'intervalle $[\theta_{min}, \theta_{max}]$, avec $\theta_{min}, \theta_{max} \in [0, \frac{\pi}{2}]$

Le pseudo-code de l'approche hybride BBAQEA2 est illustré dans l'algorithme 4.3. Les principaux changements par rapport à l'algorithme QEA sont soulignés.

1 : *t := 0*
2 : Initialiser *Q(t)*
3 : Produire *P(t)* par observation de *Q(t)*
4 : Evaluer *P(t)*
5 : Réparer les solutions
6 : Stocker le meilleur individu (la meilleure solution) parmi *P(t)* dans *B(t)*
7 : **Tant que** (*critère d'arrêt non satisfait*) **faire**
8 : *t := t+1*
9 : Produire *P(t)* par observation de *Q(t-1)*
10 : Evaluer *P(t)*
11 : Réparer les solutions
12 : Mettre à jour *Q(t-1)* en utilisant l'équation (4.15)
13 : Stocker la meilleure solution
14 : **Fin tant que**

Algorithme 4.3: Pseudo-code de l'approche hybride BBAQEA2.

4.4.4 Représentation de la solution

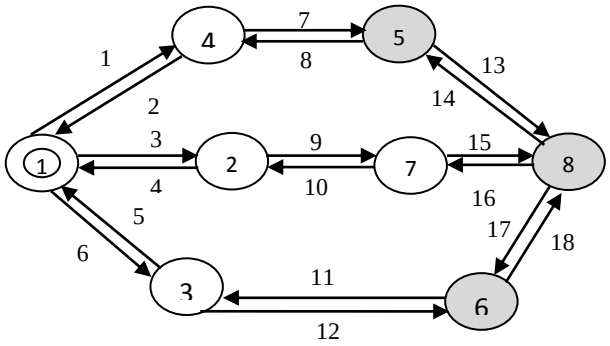
La représentation de l'ième individu qubit est définie par un vecteur réel $Q_i^t = (q_{i1}^t, q_{i2}^t, \ldots, q_{il}^t)$, *i=1,2,…,l* , avec *l* représente la longueur du vecteur qui est égal au nombre de liens dans le réseau. Le vecteur qubit Q_i^t est représenté par l'équation (4.16).

$$Q_i^t = \begin{bmatrix} \alpha_{i1}^t | \, \alpha_{i2}^t | \, \dots | \, \alpha_{il}^t \\ \beta_{i1}^t | \, \beta_{i2}^t | \, \dots | \, \beta_{il}^t \end{bmatrix} \tag{4.16}$$

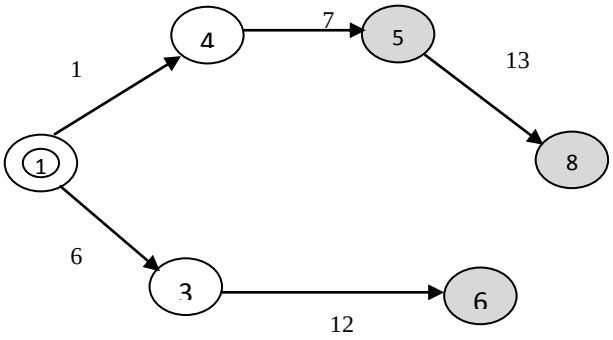
La représentation binaire de la solution, qui est un arbre de diffusion multicast, est définie par un vecteur binaire $X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{il}^t)$, *i=1,2,…,l* , avec x_{ij}^t peut prendre la valeur 0 ou 1. Le vecteur binaire est défini comme suit :

$$x_{ij} = \begin{cases} 1, \text{si rand}(0,1) > \left|\alpha_{ij}^t\right|^2 \text{ (} i \text{ est sélectionné pour construire l'arbre multicast)} \\ 0, \text{ sinon} \end{cases} \tag{4.17}$$

La figure 4.1(d) montre une représentation binaire correspondant à la représentation qubit (figure 4.1(c)) de l'arbre multicast (solution) (figure 4.1(b)) obtenu à partir d'un graphe donné (figure 4.1(a)), où 1 représente le nœud source et 5, 6 et 8 représentent les nœuds destinations.



(a) Graphe original.



(b) L'arbre multicast obtenu (solution).

$\frac{\pi}{10}$	$\frac{\pi}{3}$	$\frac{2\pi}{5}$	$\frac{3\pi}{8}$	$\frac{2\pi}{7}$	$\frac{\pi}{6}$	$\frac{2\pi}{9}$	$\frac{3\pi}{11}$	$\frac{4\pi}{10}$	$\frac{5\pi}{14}$	$\frac{4\pi}{13}$	$\frac{\pi}{9}$	$\frac{\pi}{8}$	$\frac{3\pi}{10}$	$\frac{4\pi}{9}$	$\frac{5\pi}{12}$	$\frac{3\pi}{8}$	$\frac{3\pi}{7}$

(c) Représentation qubit de la solution.

1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0

(d) Représentation binaire de la solution.

Figure 4.1: Représentation de la solution.

4.4.5 Initialisation de la population

Avant de générer la population initiale, nous devons supprimer tous les liens dont la bande passante est inférieure à la bande passante minimale requise B_{min}. L'algorithme de recherche

aléatoire Depth-First est utilisé pour générer une solution initiale aléatoire. Nous commençons à partir du nœud source, et nous le connectons aléatoirement à l'un de ses successeurs. Si ce nœud constitue une destination, le chemin est valide, sinon on refait le même processus jusqu'à atteindre toutes les destinations. Le même processus est utilisé pour générer *m* arbres multicast initiaux [T_1, T_2, ..., T_m]. Afin d'éviter la génération des boucles ou cycles, une vérification est effectuée à chaque fois qu'un nœud est ajouté.

4.4.6 Réparation de la solution

Nous avons utilisé le même algorithme de réparation de la solution présenté dans la section 3.4.2.3. La procédure de réparation de l'arbre de diffusion multicast est donnée comme suit :

1) Supprimer tous les arcs qui n'existent pas dans le graphe original.
2) Connecter le nœud source avec tous les nœuds destinations
3) Eliminer toutes les boucles qui existent dans le réseau multicast
4) Supprimer les nœuds feuilles qui ne constituent pas de destinations ainsi que leurs arcs adjacents
5) Remplacer, si possible, les chemins indirects par les chemins directs. Un chemin d'accès direct est un chemin qui connecte deux nœuds directement, tandis qu'un chemin indirect est un chemin qui relie deux nœuds indirectement par d'autres nœuds intermédiaires.

4.5 Résultats de simulation

Afin d'évaluer l'efficacité et la performance de nos approches hybrides BBAQEA1 et BBAQEA2, nous avons mené une série de tests où nous avons comparé ces deux méthodes avec l'algorithme ICBBA2 présenté dans le chapitre 3. Les trois algorithmes ont été implémentés en utilisant Visual C++. Les simulations sont exécutées sur un PC doté d'un processeur Core I3 2.27GHz, une RAM de 4Go et un système d'exploitation Windows 7. Les positions des nœuds sont fixées aléatoirement dans un rectangle de dimension 4000Km x 2400Km. Le modèle de topologie WAXMAN est utilisé afin de générer le graphe aléatoirement. Le coût, le délai, la gigue, la bande passante et le taux de perte de chaque lien sont répartis uniformément dans les intervalles [2, 10], [0, 30], [40, 80] et [0.0001, 0.01] respectivement. D_{max}, J_{max}, et LR_{max} sont fixés à 120 ms, 60 ms et 0.05 respectivement. La bande passante minimale requise est générée de façon aléatoire dans l'intervalle [40, 80]. Le nœud source et les nœuds destinations sont sélectionnés aléatoirement. Le coût moyen de

l'arbre multicast est obtenu en effectuant 50 exécutions de chacun de ces algorithmes. Nous avons appliqué nos algorithmes pour deux types de réseaux sans fil, à savoir les WMNs et les VANETS

4.5.1 Application pour les WMNs

Nous avons étudié les performances de nos algorithmes proposés dans deux scénarios différents

Dans le premier scénario, nous avons varié le nombre de nœuds de 10 à 100 par incrément de 10 nœuds et avec 10% de nœuds comme destinations afin d'évaluer les performances des algorithmes proposés en termes du coût, temps de convergence, délai, gigue et taux de perte de paquets en les comparant avec l'algorithme ICBBA2. Les résultats sont donnés dans les figures 4.2 à 4.6.

La figure 4.2 montre le coût de l'arbre de diffusion multicast généré par chaque algorithme en variant le nombre de nœuds de 10 à 100 par incrément de 10 et avec 10% de nœuds comme destinations. Les résultats montrent clairement que le coût de l'arbre multicast augmente pour chaque algorithme en augmentant le nombre de nœuds. BBAQEA2 a le coût le plus bas tandis que ICBBA2 a le coût le plus élevé.

Le temps de convergence de chaque algorithme en faisant varier le nombre de nœuds dans le réseau de 10 à 100 avec 10% de nœuds comme destinations est représenté dans la figure 4.3. Les résultats révèlent clairement que le temps de convergence de chaque algorithme augmente en augmentant le nombre de nœuds. Il en ressort que BBAQEA2 a le temps de calcul le plus faible par rapport à BBAQEA1 et ICBBA2.

La figure 4.4 montre le délai de l'arbre multicast généré par chaque algorithme en faisant varier le nombre de nœuds du réseau avec 10 % des nœuds comme destinations. Il est observé que le délai généré par BBAQEA2 est le plus bas par rapport à BBAQEA1 et ICBBA2.

La figure 4.5 représente la gigue de l'arbre multicast générée par chaque algorithme dans un réseau de 100 nœuds avec 10 % de nœuds comme destinations. Les résultats obtenus confirment notre conclusion sur ceux du délai. Comme pour le délai, BBAQEA2 a la gigue la plus faible par rapport à la gigue de BBAQEA1 et ICBBA2.

La figure 4.6 montre le taux de perte de paquets des six algorithmes testés dans un réseau de 100 nœuds avec 10 % de nœuds comme destination. On observe que BBAQEA2 a le plus faibles taux de perte de paquets par rapport à BBAQEA1 et ICBBA2.

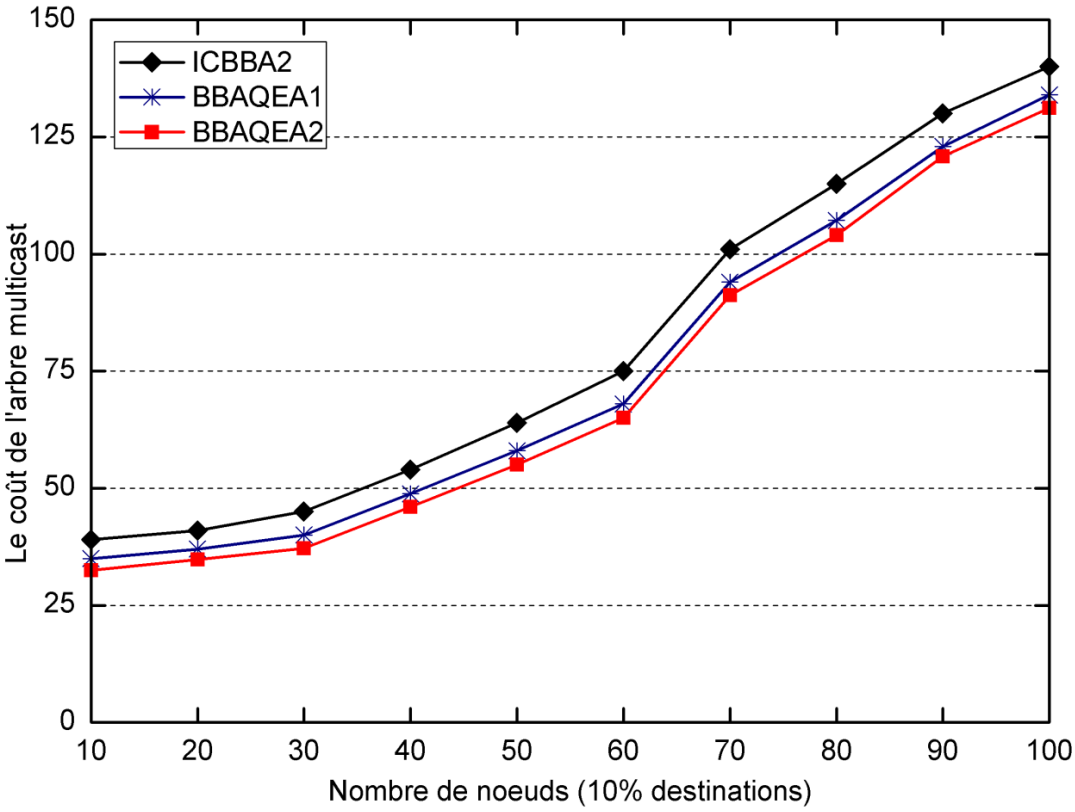


Figure 4.2: Coût de l'arbre multicast en variant le nombre de nœuds.

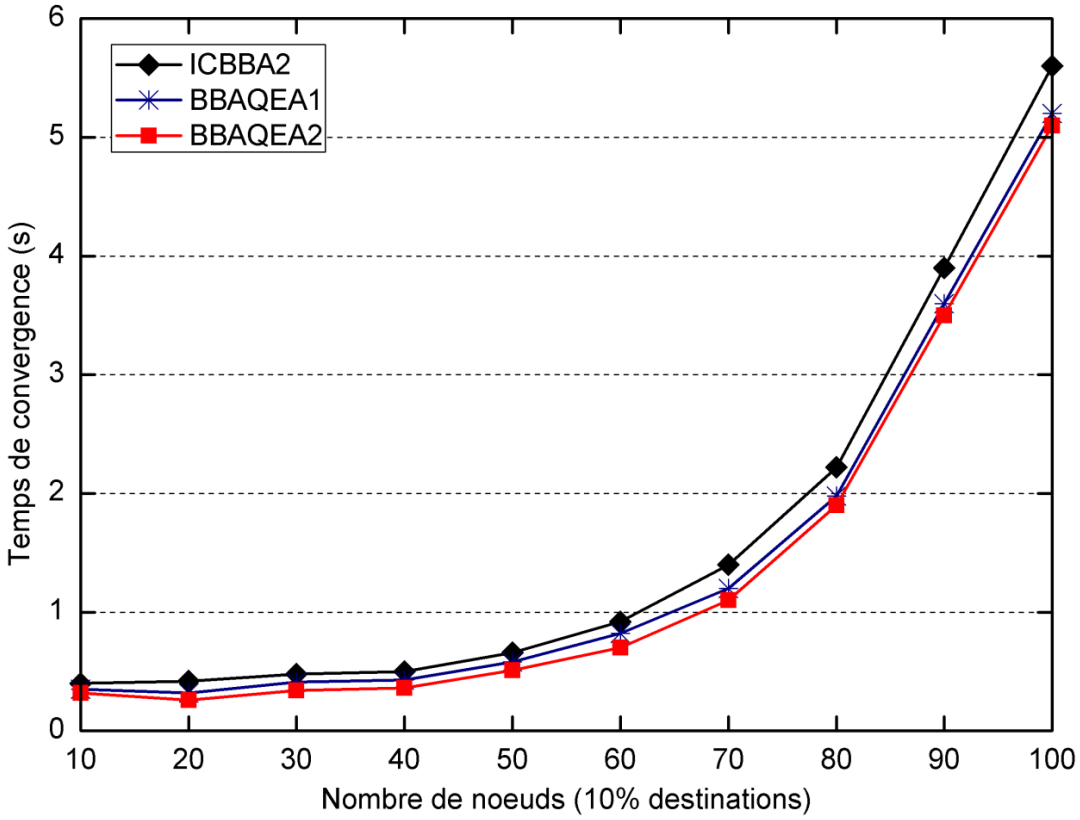


Figure 4.3: Temps de convergence de chaque algorithme en variant le nombre de nœuds.

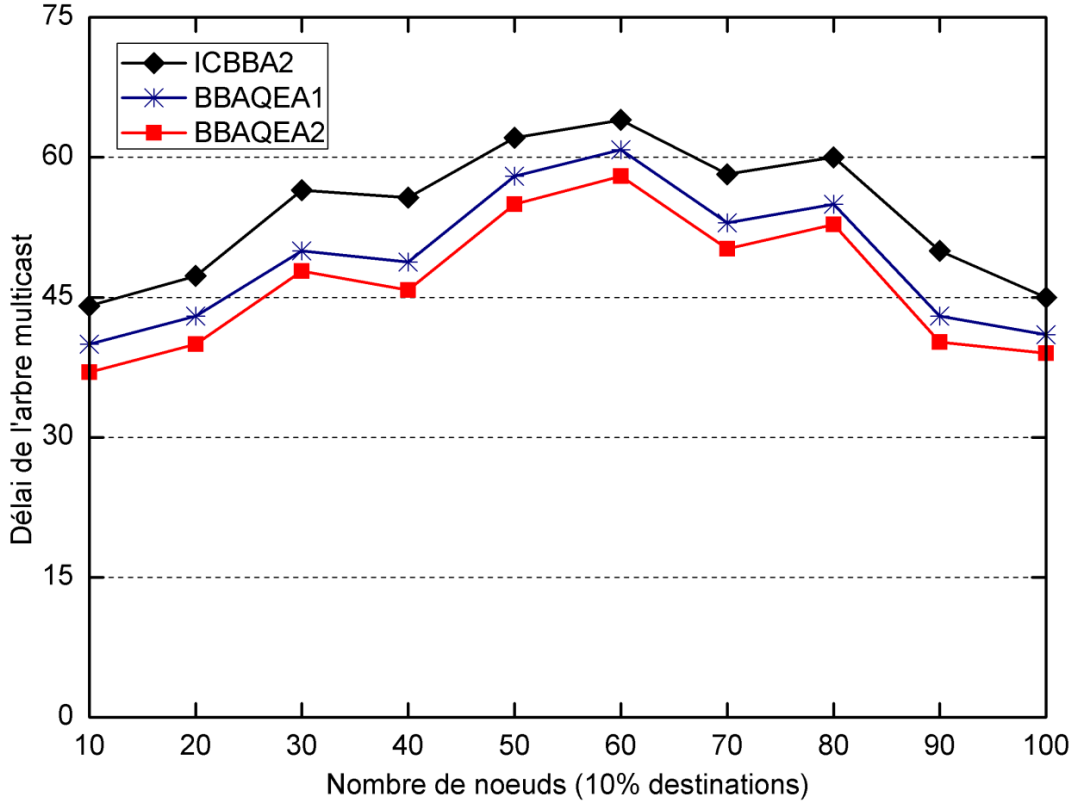


Figure 4.4: Délai de l'arbre multicast en variant le nombre de nœuds.

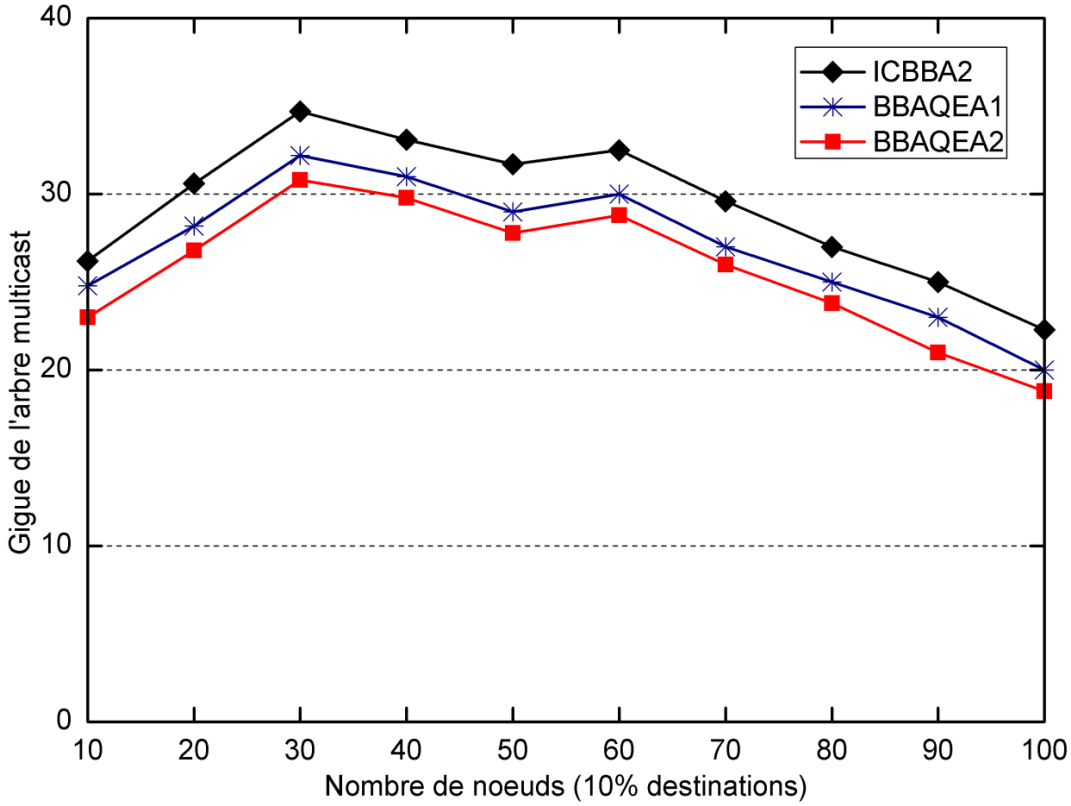


Figure 4.5: Gigue de l'arbre multicast en variant le nombre de nœuds.

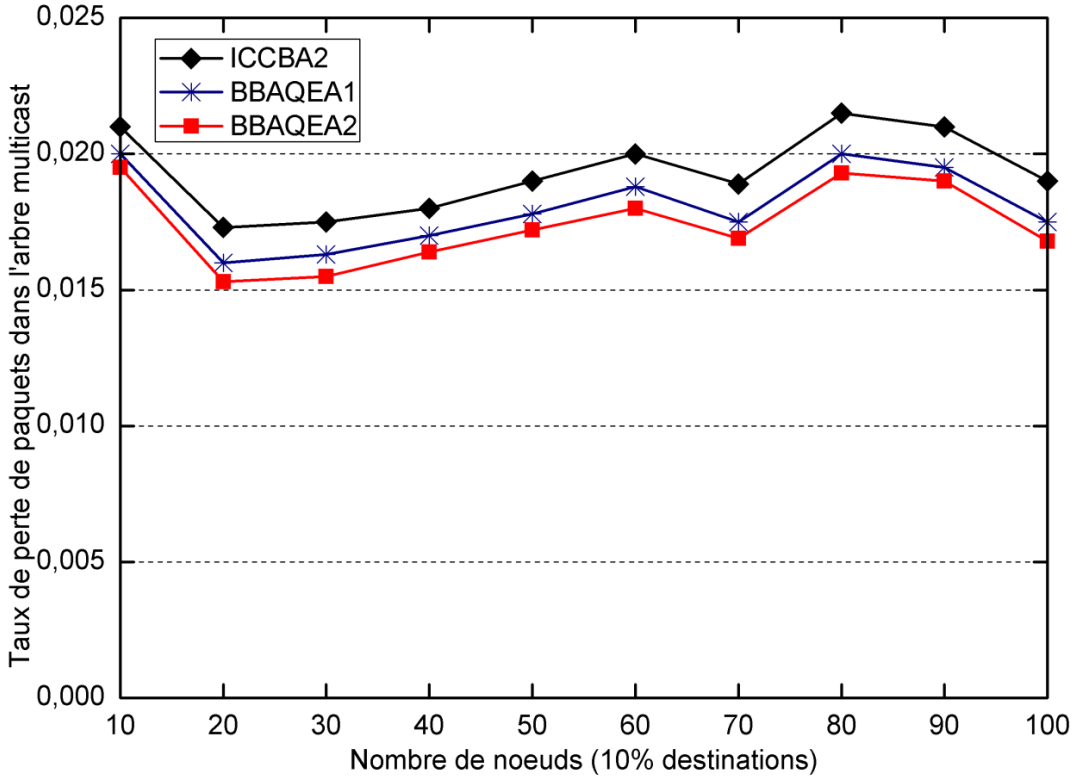


Figure 4.6: Taux de perte de paquets de l'arbre multicast en variant le nombre de nœuds.

Dans le deuxième scénario, afin d'évaluer les performances de nos algorithmes proposés en termes du coût de l'arbre multicast obtenu et du temps de convergence, nous avons varié le

nombre de nœuds destinations de 10 % à 70% dans un réseau fixe de 100 nœuds. Les résultats sont donnés dans les figures 4.7 et 4.8.

La figure 4.7 montre le coût de l'arbre multicast généré par chaque algorithme en variant le nombre de nœuds destinations dans un réseau fixe de 100 nœuds. Nous constatons que le coût de l'arbre multicast pour chaque algorithme augmente avec l'augmentation du nombre de nœuds destinations. Nous observons également que l'arbre de diffusion multicast généré par BBAQEA2 a le coût le plus bas par rapport à BBAQEA1 et ICBBA2.

La figure 4.8 montre le temps de convergence de chaque algorithme en variant le nombre de nœuds destinations de 10% à 70% dans un réseau de 100 nœuds. Les résultats montrent clairement que le temps de convergence de chaque algorithme augmente avec l'augmentation du nombre de nœuds. BBAQEA2 consomme le moins de temps de calcul par rapport à BBAQEA1 et ICBBA2.

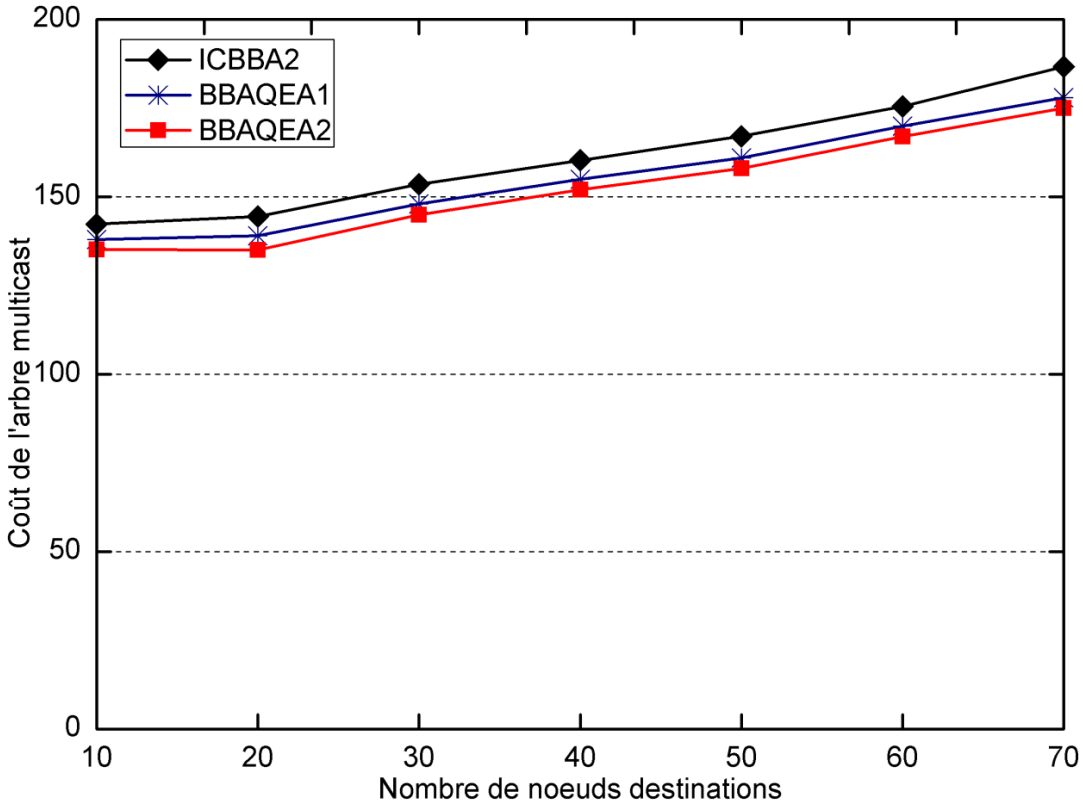


Figure 4.7: Coût de l'arbre multicast en variant le nombre de nœuds destinations.

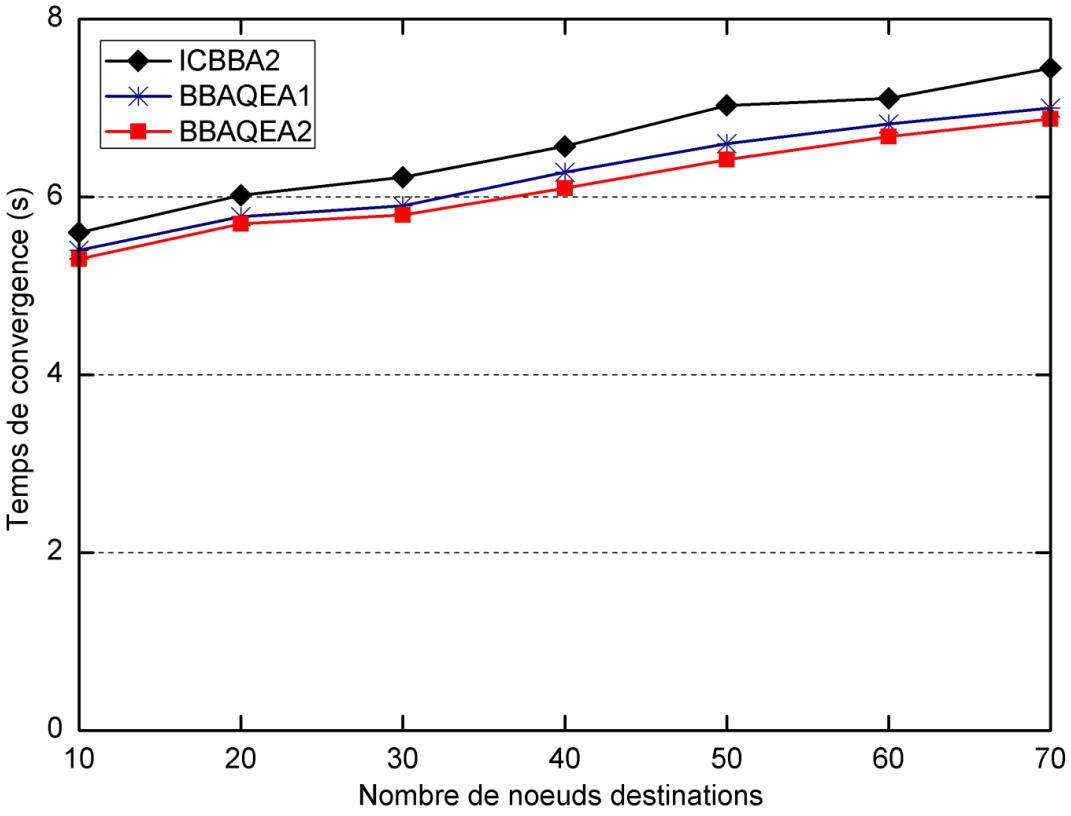


Figure 4.8: Temps de convergence de chaque algorithme en variant le nombre de nœuds destinations.

4.5.2 Application pour les VANETs

Comme dans les WMNs, nous allons étudier les performances de nos algorithmes proposés dans deux scénarios différents

Dans le premier scénario, nous avons varié le nombre de véhicules de 5 à 50 par incrément de 5 véhicules et avec 10% de véhicules comme destinations afin d'évaluer la robustesse des algorithmes proposés en termes du coût, temps de convergence, délai, gigue et taux de perte de paquets en les comparant avec l'algorithme ICBBA2. Les résultats sont donnés dans les figures 4.9 à 4.13.

La figure 4.9 montre le coût de l'arbre de diffusion multicast généré par chaque algorithme en faisant varier le nombre de véhicules dans le réseau de 5 à 50 avec 10% de véhicules comme destinations. Les résultats montrent que le coût de l'arbre multicast augmente pour chaque algorithme en augmentant le nombre de véhicules. BBAQEA2 a le coût le plus faible par rapport à BBAQEA1 et ICBBA2.

La figure 4.10 montre le temps de convergence de chaque algorithme en faisant varier le nombre de véhicules du réseau de 5 à 50 avec 10 % des véhicules comme destinations. Il est observé que BBAQEA2 a le temps de calcul le plus faible par rapport à BBAQEA1 et ICBBA2.

Le délai de l'arbre de diffusion multicast généré par chaque algorithme en variant le nombre de véhicules de 5 à 50 par incrément de 10 et avec 10% de véhicules comme destinations est représenté dans la figure 4.11. Les résultats montrent que BBAQEA2 a le délai le plus faible par rapport au délai de BBAQEA1 et ICBBA2.

La figure 4.12 représente la gigue de l'arbre multicast générée par chaque algorithme dans un réseau de 50 véhicules avec 10 % de véhicules comme destinations. Les résultats obtenus confirment notre conclusion sur ceux du délai. Comme pour le délai, BBAQEA2 a la gigue la plus faible tandis que ICBBA2 a la gigue la plus élevée.
La figure 4.13 montre le taux de perte de paquets des six algorithmes testés dans un réseau de 50 véhicules avec 10 % de véhicules comme destination. On observe que les algorithmes proposés ont les plus faibles taux de perte de paquets par rapport à GA, PSO, JPSO, et BBA.

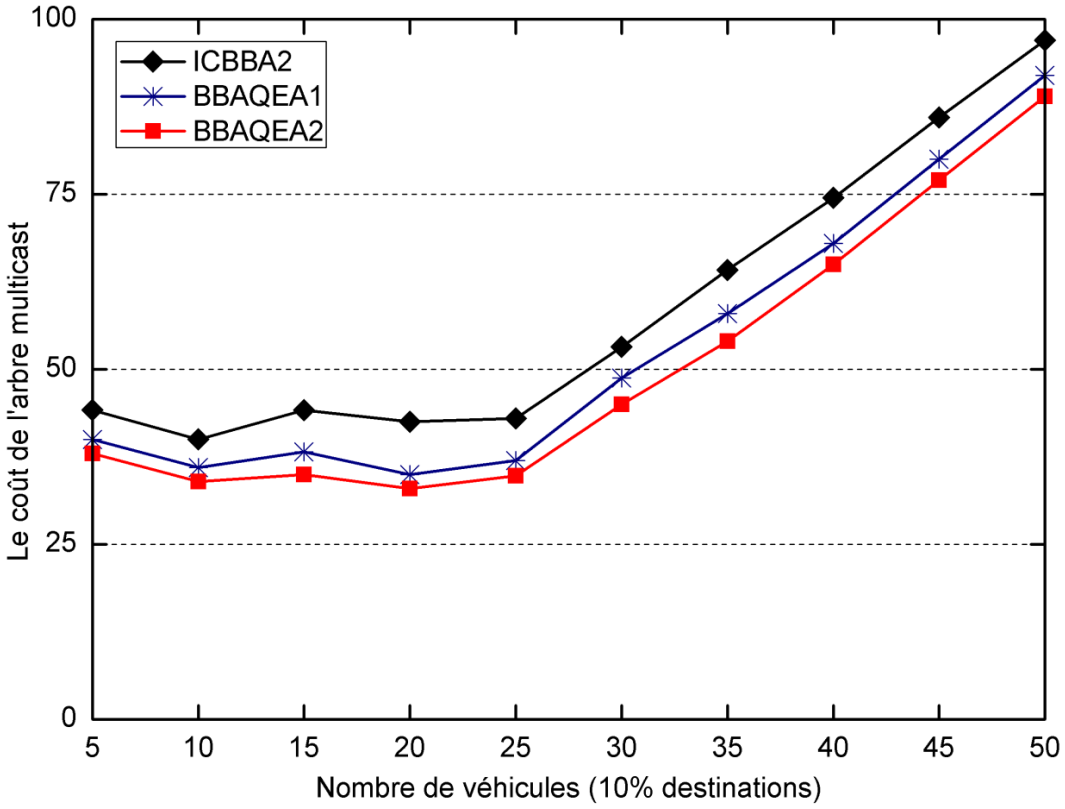


Figure 4.9: Coût de l'arbre multicast en variant le nombre de véhicules destinations.

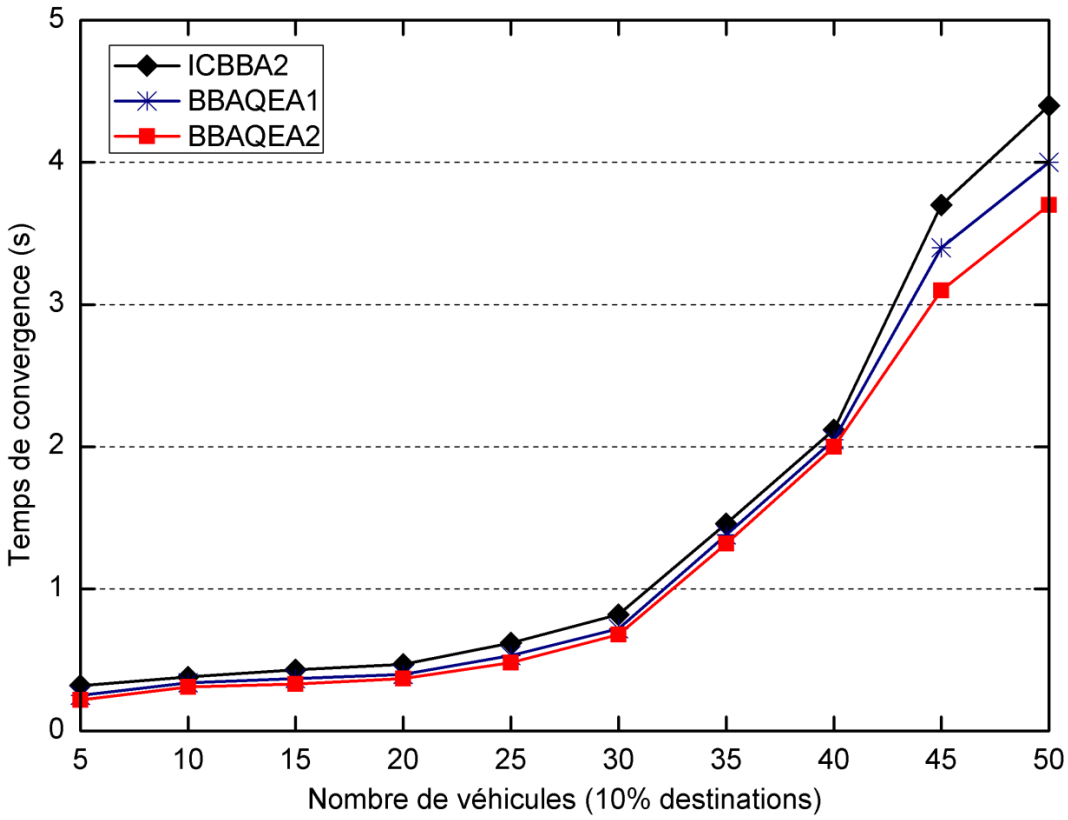


Figure 4.10: Temps de convergence de chaque algorithme en variant le nombre de véhicules.

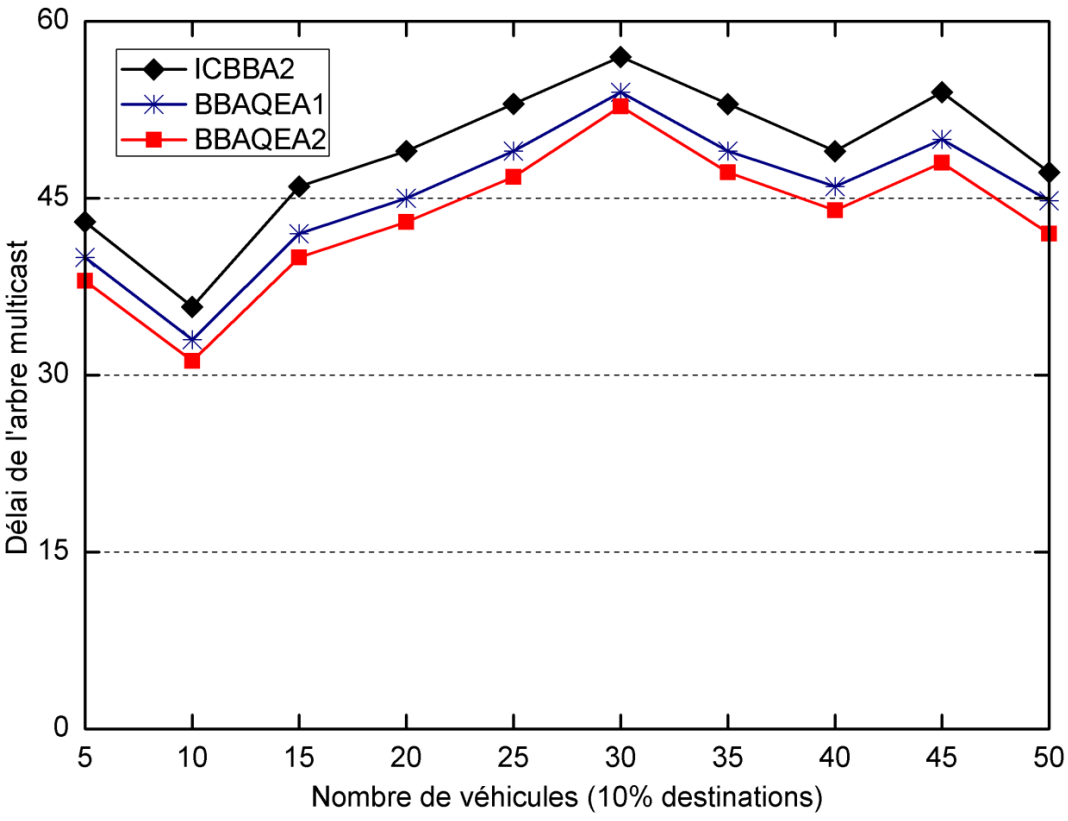


Figure 4.11: Délai de l'arbre multicast en variant le nombre de véhicules.

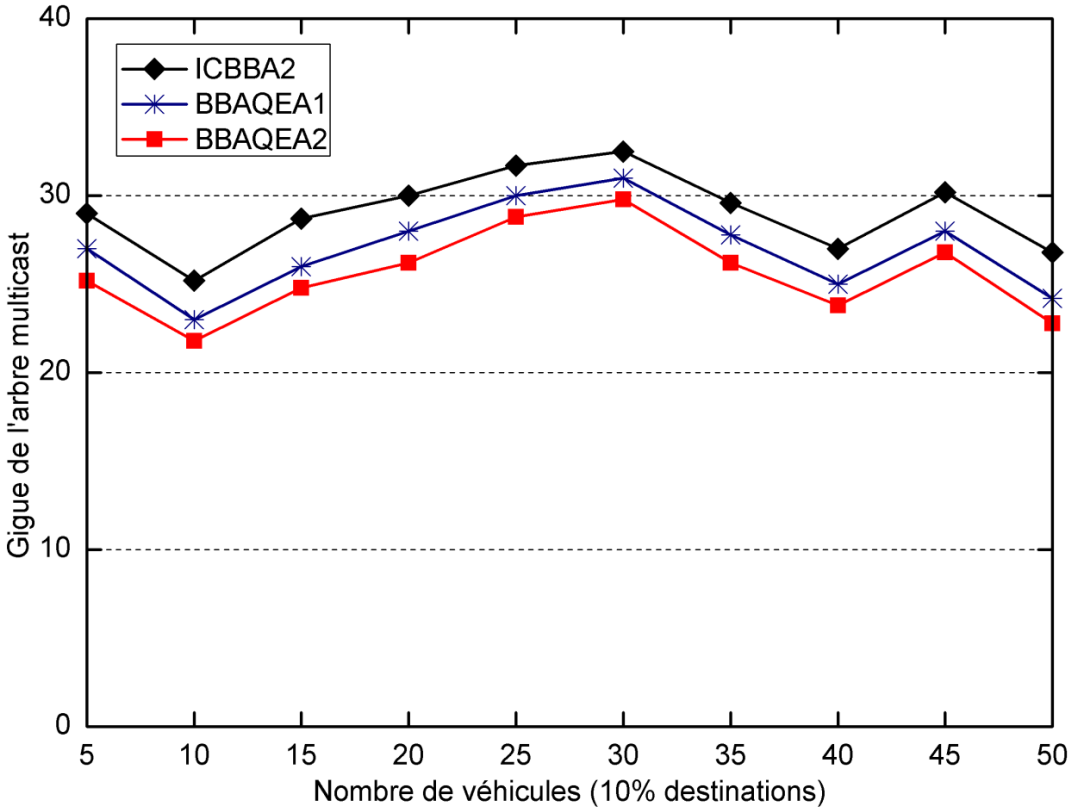


Figure 4.12: Gigue de l'arbre multicast en variant le nombre de véhicules

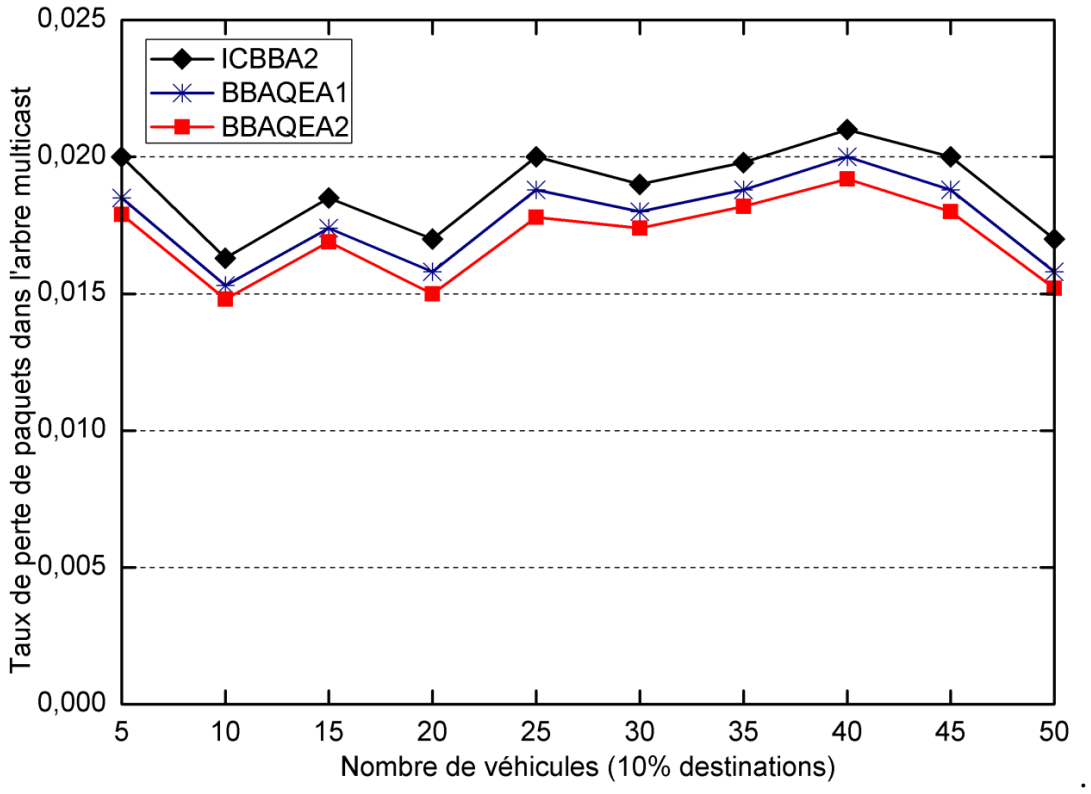


Figure 4.13: Taux de perte de paquets de l'arbre multicast en variant le nombre de véhicules.

Dans le deuxième scénario, nous avons varié le nombre de véhicules destinations de 10 % à 70% dans un réseau fixe de 50 véhicules afin d'évaluer les performances de nos algorithmes proposés en termes du coût de l'arbre multicast obtenu et du temps de convergence. Les résultats sont donnés dans les figures 4.14 et 4.15.

La figure 4.14 représente le coût de l'arbre multicast généré par chaque algorithme en faisant varier le nombre de véhicules destinations de 10 % à 70% dans un réseau fixe de 50 véhicules. Nous constatons que le coût de l'arbre multicast pour chaque algorithme augmente avec l'augmentation du nombre de nœuds destinations. Nous observons également que BBAQEA2 a le coût le plus bas tandis que ICBBA2 a le coût le plus élevé.

La figure 4.15 montre le temps de convergence de chaque algorithme en variant le nombre de véhicules destinations de 10% à 70% dans un réseau de 50 véhicules. Les résultats montrent clairement que le temps de convergence de chaque algorithme augmente avec l'augmentation du nombre de nœuds. BBAQEA2 consomme le moins de temps de calcul par rapport à BBAQEA1 et ICBBA2.

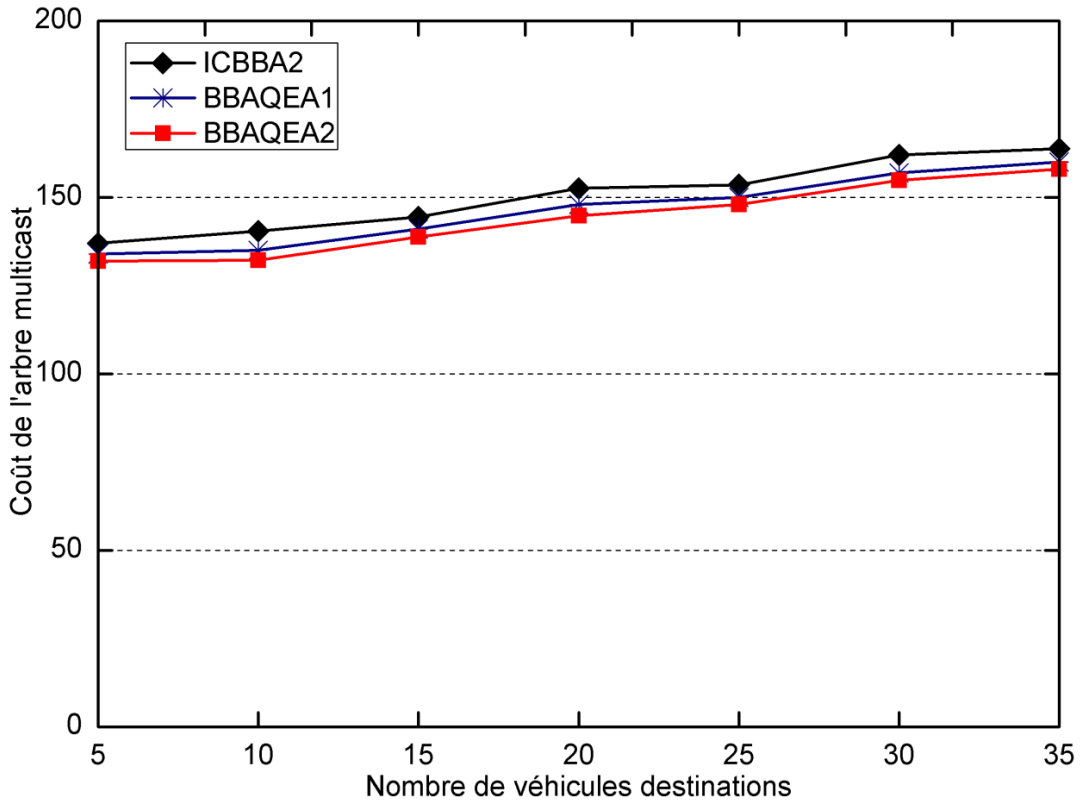


Figure 4.14: Coût de l'arbre multicast en variant le nombre de véhicules destinations.

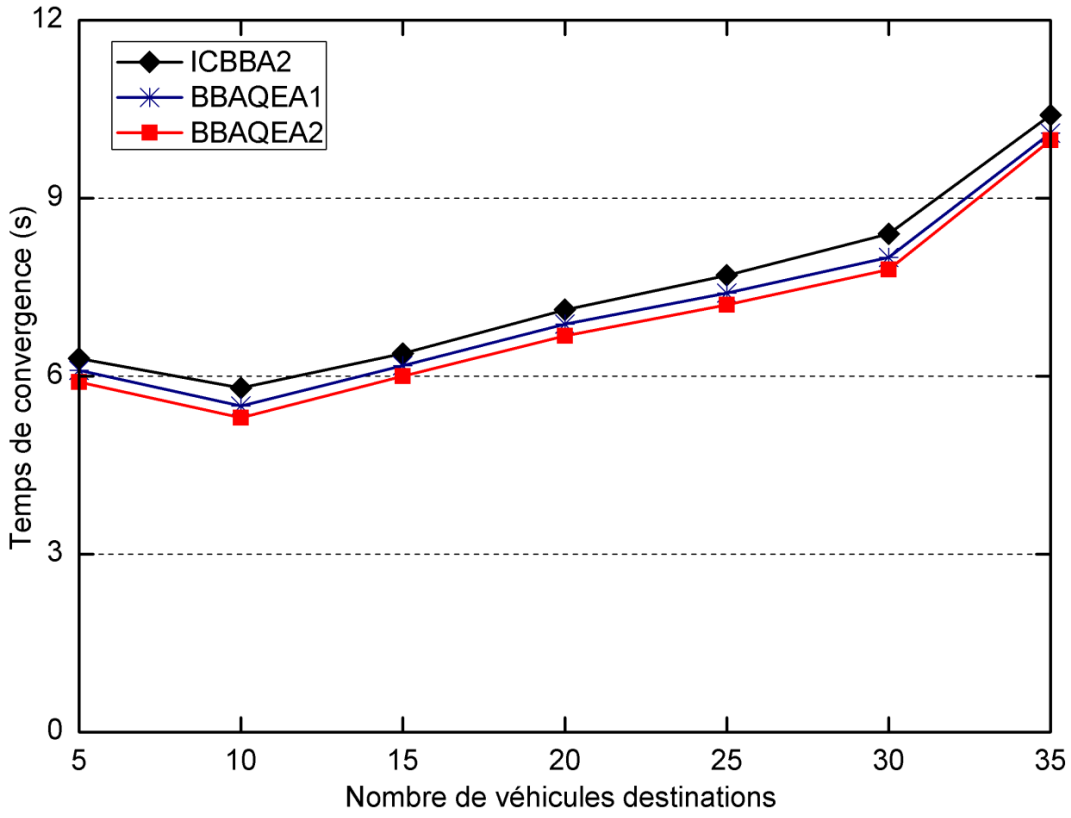


Figure 4.15: Temps de convergence de chaque algorithme en variant le nombre de véhicules destinations.

4.6 Conclusion

Dans ce chapitre, nous avons présenté deux nouvelles approches basées sur l'hybridation de BBA avec QEA. L'idée principale de la première approche proposée, appelée BBAQEA1, est l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. Tandis que l'idée principale de la deuxième approche proposée, appelée BBAQEA2, est que l'opérateur évolutionnaire quantique de QEA est remplacé par l'équation d'évolution de BBA Ces algorithmes sont appliqués pour résoudre le problème de routage multicast avec QoS pour deux types de réseaux sans fil, à savoir les WMNs et les VANETS. Les résultats expérimentaux montrent que BBAQEA2 est la meilleure solution dans la plupart des cas pour les deux types de réseaux sans fil.

CONCLUSION GÉNÉRALE

Les WMNs et les VANETs jouent un rôle clé dans les réseaux sans fil de nouvelles générations. Cela est dû à la fois à leur facilité de déploiement, à leur faible coût de déploiement ainsi qu'aux nombreux domaines d'application. Néanmoins, une des contraintes de ces réseaux est le problème d'acheminement de données depuis une source vers un groupe de destinations avec satisfaction de garantie de QoS.

L'objectif de cette thèse est de résoudre le problème de routage multicast avec QoS dans les WMNs et les VANETs afin de trouver un arbre multicast à moindre coût et satisfaire les garanties de QoS en termes de délai, de gigue, de bande passante et de taux de perte des paquets.
Nous avons proposé deux contributions majeures:

Dans notre première contribution, nous avons présenté une amélioration de l'algorithme des chauves-souris BA où nous avons proposé deux variantes ICBBA1 (utilisant la fonction logistique pour déterminer β) et ICBBA2 (utilisant la fonction tent pour déterminer β) pour résoudre le problème de routage multicast avec QoS. Les résultats expérimentaux ont montré la supériorité et efficacité des algorithmes proposés par rapport à d'autres méthodes existantes dans la littérature tels que GA, PSO, JPSO et BBA. ICBBA2 est la meilleure solution dans la plupart des cas pour les deux types de réseaux sans fil.

Dans la deuxième contribution, pour palier au problème de déséquilibre entre l'exploration et l'exploitation de l'espace de recherche des solutions et assurer leur diversité, nous avons proposé deux approches hybrides basées sur l'hybridation de l'algorithme des chauves-souris binaire BBA avec l'algorithme évolutionnaire quantique QEA. La première approche, nommée BBAQEA1, est basée sur l'intégration de l'équation d'évolution de BBA dans l'opérateur quantique de QEA. La deuxième approche, nommée BBAQEA2, est basée sur le remplacement de l'opérateur évolutionnaire quantique de QEA par l'équation d'évolution de BBA. Les résultats des expérimentations ont montré que BBAQEA2 obtient les meilleurs résultats en termes de temps de convergence et de qualité de solution dans la plupart des cas pour les deux types de réseaux sans fil.

En perspectives, nous proposons d'étudier le même problème en variant le nombre total de nœuds dans le réseau et le nombre de nœuds destinations simultanément ainsi en variant le nombre d'itérations.

Nous suggérons également d'appliquer les algorithmes proposés pour résoudre d'autres problèmes d'optimisation tels que le problème de planifications dans les réseaux sans fil maillés et le problème de détection d'intrusion.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] T. Lemlouma and D.N. BADACHE "Le routage dans les réseaux mobiles Ad Hoc". Université des Sciences et de la Technologie Houari Boumediene, Institut d'Informatique (2000).

[2] S. Rimour "Généralités sur les réseaux: chapitre 1". IUT C.F département informatique (2002).

[3] J. Defaye, "les différents types de réseaux sans fil". Conservatoire des arts et métiers Rhône-Alpes Centre de Lyon (2007).

[4] A. OUNI "Optimisation de la capacité et de la consommation énergétique dans les réseaux maillés sans fil". Thèse de doctorat. Lyon, INSA. (2013).

[5] I.F. Akyildiz, X. Wang and W. Wang, "Wireless mesh networks: a survey". *Computer networks*, 47(4) (2005) pp. 445-487.

[6] I.F. Akyildiz and X. Wang "Wireless mesh networks". John Wiley & Sons, (2009).

[7] C. ROY "Amélioration de la performance des réseaux maillés sans fil cognitifs". Thèse de doctorat. École de technologie supérieure. Université du Québec. (2012).

[8] U. ASHRAF "Qualité de service et routage dans les réseaux maillés sans fil". Thèse de doctorat. Toulouse, INSA. (2010).

[9] C. MOLLE "Optimisation de la capacité des réseaux radio maillés". Thèse de doctorat. Université Nice Sophia Antipolis. (2009).

[10] M. BOUSHABA "Routage adaptatif et stabilité dans les réseaux maillés sans fil". Thèse de doctorat. Université de Montréal. (2013).

[11] J. Bernsen and D. Manivannan, "Unicast Routing Protocols for Vehicular Ad Hoc Networks: A Critical Comparison and Classification", *Elsevier Journal of Parvasive and Mobile Computing*, 5 (2009) pp. 1-18.

[12] R. Meraihi, M. Senouci and M. Djebri, "Réseau mobile Ad Hoc et réseaux de capteurs sans fil", *chapitre de livre Edition Hermès* (2006).

[13] Y. Khaled, H. Menouar and Y. Challal, "Reactive and adaptive protocol for inter-vehicle communications (RAP-IVC)", *In: Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on. IEEE*, (2004) pp. 63-64.

[14] A. Bachir and A. Benslimane, "A multicast protocol in ad hoc networks inter-vehicle geocast", *In Vehicular Technology Conference. The 57th IEEE Semiannual*, 4 (2009) pp. 2456-2460.

[15] G. Korkmaz, E. Ekici and F. Ozguner, "An efficient fully ad-hoc multi-hop broadcast protocol for inter-vehicular communication systems", *In: 2006 IEEE International Conference on Communications*, 1 (2006) pp. 423-428.

[16] K. Gökhan, "GPS based wireless communication protocols for vehicular ad-hoc networks". Thèse de doctorat. The Ohio State University (2006).

[17] M. Raya and J-P. Hubaux, "Securing vehicular ad hoc networks", *Journal of Computer Security*, 15(1) (2007) pp. 39-68.

[18] R. Meraihi, G. Le Grand, S. Tohmé and M. Riguidel, "Gestion multicouches de la qualité de service dans un réseau ad hoca cœur stable". *In: Actes du Colloque Francophone sur l'Ingénierie des Protocoles (CFIP), Evry, France*. (2003).

[19] R. Meraihi "Gestion de la qualité de service et contrôle de topologie dans les réseaux Ad Hoc". Thèse de doctorat. Ecole Nationale Supérieure des Télécommunications de Paris (2005).

[20] M. SEDRATI "Architecture pour le support de la qualité de service sans les réseaux ad hoc". Thèse de doctorat. Université de Batna. (2012).

[21] J.P. CHANET "Algorithme de routage coopératif à qualité de service pour des réseaux ad hoc agri-environnementaux". Thèse de doctorat. Université Blaise Pascal-Clermont-Ferrand II. (2007).

[22] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick, R. Nair and B. Rajagopalan, "A Framework for QoS-based Routing in the Internet", IETF RFC2386 (1998).

[23] M. Frodigh, P. Johansson and P. Larsson " Wireless ad hoc networking: the art of networking without a network. *Ericsson Review*, 4 (4) (2000) pp. 249.

[24] H. Moustafa " Routage unicast et multicast dans les réseaux mobiles Ad Hoc". Thèse doctorat. Ecole Nationale Supérieure des Télécommunications de Paris (2004).

[25] A. Naimi-Meraihi "Délai de routage dans les réseaux Ad Hoc 802.1". Thèse doctorat. INRIA Rocquencourt (2005).

[26] J. Vandermeerschen "Hybridation entre les modes Ad Hoc et infrastructure dans les réseaux de type Wifi-Fi". Rapport de DEA. (2006).

[27] M.E. LALAMI, "Contribution à la résolution de problèmes d'optimisation combinatoire: méthodes séquentielles et parallèles". Thèse de doctorat. Université de Toulouse, Université Toulouse III-Paul Sabatier. (2012).

[28] EL DOR, Abbas, "Perfectionnement des algorithmes d'optimisation par essaim particulaire: applications en segmentation d'images et en électronique". Thèse de doctorat. Université Paris-Est. (2012).

[29] C. VANARET, "Hybridation d'algorithmes évolutionnaires et de méthodes d'intervalles pour l'optimisation de problèmes difficiles". Thèse de doctorat. Université de Toulouse. (2015).

[30] H. HACHIMI, "Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications". Thèse de doctorat. INSA de Rouen. (2013).

[31] C. Yann and S. Patrick, "Optimisation Multiobjectif", *Eyrolles*, (2002).

[32] D. Simon, "Evolutionary optimization algorithms: Biologically-Inspired and Population Based Approaches to Computer Intelligence", *John Wiley & Sons, Inc., Hoboken, New Jersey*, (2013).

[33] I. BOUSSAID, "Perfectionnement de métaheuristiques pour l'optimisation continue". Thèse de doctorat. Université Paris-Est. (2013).

[34] I. Devarenne, "Etudes en recherche locale adaptative pour l'optimisation combinatoire". Thèse de Doctorat. Université de Franche-Comté. (2007).

[35] T. El-Ghazali, "Metaheuristics from design to implementation", *Wiley*, (2010).

[36] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems", *Econometrica*, 28 (1960) pp. 497-520.

[37] R. Bellman, "Some applications of the theory of dynamic programming, a review", *Operations Research*, 2 (1954) pp. 275-288.

[38] R. Bellman, "Dynamic Programming", *Princeton University Press*, (1957).

[39] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by simulated annealing", *Science*, 220 (4598) (1983) pp. 671–680.

[40] F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research*, 13(5) (1986) pp. 533–549.

[41] J. Dréo, A. Petrowski, É. D. Taillard and P. Siarry, "Métaheuristiques pour l'optimisation difficile", Eyrolles (Editions), (2003).

[42] J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, USA, (1975).

[43] L. J. Fogel, A. J. Owens and M. J. Walsh, "Artificial Intelligence through Simulated Evolution", *John Wiley, New York, USA*, (1966).

[44] I. Rechenberg, "Cybernetic solution path of an experimental problem", Rapport technique, Royal Air Force Establishment, (1965).

[45] I. Rechenberg, "Evolutions strategie: optimierung technischer systeme nach prinzipien der biologischen evolution", Frommann-Holzboog, Stuttgart, (1973).

[46] H.-P. Schwefel, "Numerical Optimization of Computer Models", John *Wiley & Sons, Inc., New York, NY, USA*, (1981).

[47] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems) ", *The MIT Press, 1edition*, (1992).

[48] E. Bonabeau, M. Dorigo and G. Theraulaz, "Swarm intelligence: from natural to artificial systems", *Oxford University Press, Inc., New York, NY, USA*, (1999).

[49] M. Dorigo, V. Maniezzo and A. Colorni, "Positive feedback as a search strategy", *Rapport technique*, Ecole polytechnique de Milan. (1991).

[50] M. Dorigo "Optimization, Learning and Natural Algorithms". Thèse doctorat. Ecole polytechnique de Milan. (1992).

[51] M. Dorigo, V. Maniezzo and A. Colorni, "The ant system: Optimization by a colony of cooperating agents", *IEEE Transactions On Systems, Man And Cybernetics-Part B*, 26(1) (1996) pp. 29-41.

[52] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey", *Theoretical Computer Science*, 344(2-3) (2005) pp. 243–278.

[53] J. L. Deneubourg, S. Aron, S. Goss and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant", *Journal of Insect Behavior*, 3(2) (1990) pp. 159–168.

[54] J. Kennedy and R. Eberhart, "Particle swarm optimization", *IEEE International Conference on Neural Networks*, 4 (1995) pp. 1942–1948.

[55] B. Alatas and A. Erhan, "Rough particle swarm optimization and its applications in data mining", *Soft Computing*, 12(12) (2008) pp. 1205-1218.

[56] J. Kennedy and R. Mendes, "Population structure and particle swarm performance", *Computational Intelligence, Proceedings of the World on Congress on*, 2 (2002) pp. 1671–1676.

[57] X.-S. Yang, "Firefly algorithms for multimodal optimization", *In Stochastic algorithms: foundations and applications*, (2009) pp. 169-178. Springer Berlin Heidelberg.

[58] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation", *International Journal of Bio-Inspired Computation*, 2(2) (2010) pp. 78-84.

[59] X.-S. Yang, S.S.S. Hosseini and A.H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect", *Applied Soft Computing*, 12(3) (2012) pp. 1180-1186.

[60] X.-S. Yang, "A new metaheuristic bat-inspired algorithm", *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Vol. 284 of Studies in Computational Intelligence (Springer Berlin/Heidelberg, 284 (2010) pp. 65-74.

[61] R. H. Hwang, W. Y. Do and S. C. Yang, "Multicast routing based on genetic algorithms", *J. Inf. Sci. Eng.* 16(6) (2000) pp. 885-901.

[62] A. T. Haghighat, K. Faez, M. Dehghan, A. Mowlaei and Y. Ghahremani, "GA-based heuristic algorithms for QoS based multicast routing", *Knowledge-Based Systems* 16(5) (2003) pp. 305-312.

[63] B. Sun, S. Pi, C. Gui, Y. Zeng, B. Yan, W. Wang and Q. Qin, "Multiple constraints QoS multicast routing optimization algorithm in MANET based on GA", *Progress in Natural Science* 18(3) (2008) pp. 331-336.

[64] Y. S. Yen, H. C. Chao, R. S. Chang and A. Vasilakos, "Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs", *Mathematical and Computer Modeling* 53(11) (2011) pp. 2238-2250.

[65] J. Liu, J. Sun and W.-B. Xu, "QoS multicast routing based on particle swarm optimization", *Intelligent Data Engineering and Automated Learning: IDEAL-2006*, Vol. 4224 of Lecture Notes in Computer Science (Springer Berlin/Heidelberg, 2006), pp. 936-943.

[66] J. Sun, W. Fang, X. Wu, Z. Xie and W. Xu, "QoS multicast routing using a quantum-behaved particle swarm optimization algorithm", *Engineering Applications of Artificial Intelligence* 24(1) (2011) pp. 123-131.

[67] R. Qu, Y. Xu, J. P. Castro and D. Landa-Silva, "Particle swarm optimization for the Steiner tree in graph and delay-constrained multicast routing problems", *Journal of Heuristics* 19(2) (2013) pp. 317-342.

[68] M. Shen, Z. H. Zhan, W. N. Chen, Y. J. Gong, J. Zhang and Y. Li, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks", *IEEE Transactions on Industrial Electronics* 61(12) (2014) pp. 7141-7151.

[69] S. Y. Tseng, C. C. Lin and Y. M. Huang, "Ant colony-based algorithm for constructing broadcasting tree with degree and delay constraints", *Expert Systems with Applications* 35(3) (2008) pp. 1473-1481.

[70] H. Wang, H. Xu, S. Yi and Z. Shi, "A tree-growth based ant colony algorithm for QoS multicast routing problem", *Expert Systems with Applications* 38(9) (2011) 11787-11795.

[71] N. Ghaboosi and A. T. Haghighat, "Tabu search based algorithms for bandwidth-delay-constrained least-cost multicast routing", *Telecommunication Systems* 34(3-4) (2007) pp. 147-166.

[72] H. Wang, X. Meng, M. Zhang and Y. Li, "Tabu search algorithm for RP selection in PIM-SM multicast routing", *Computer Communications* 33(1) (2010) pp. 35-42.

[73] R. F. Abdel-Kader, "Hybrid discrete PSO with GA operators for efficient QoS-multicast routing", *Ain Shams Engineering Journal* 2(1) (2011) pp. 21-31.

[74] S. Mirjalili, S. M. Mirjalili and X.-S. Yang, "Binary bat algorithm", *Neural Computing and Applications* 25(3-4) (2014) pp. 663-681.

[75] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization", *Swarm and Evolutionary Computation* 9 (2013) pp. 1-14.

[76] S. Saremi, S. Mirjalili and A. Lewis, "How important is a transfer function in discrete heuristic algorithms", *Neural Computing and Applications* 26(3) (2014) pp. 625-640.

[77] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, "BGSA: Binary gravitational search algorithm", *Natural Computing* 9(3) (2010) pp. 727-745.

[78] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm", *in IEEE Int. Conf. on Computational Cybernetics and Simulation* (1997), pp. 4104-4108.

[79] R. Caponetto, L. Fortuna, S. Fazzino and M. G. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, 7(3) (2003) pp. 289-304.

[80] L. Y. Chuang, C. H. Yang and J. C. Li, "Chaotic maps based on binary particle swarm optimization for feature selection", *Applied Soft Computing* 11(1) (2011) pp. 239-248.

[81] D. Zhao and Y. He, "Chaotic binary bat algorithm for analog test point selection", *Analog Integrated Circuits and Signal Processing* 84(2) (2015) pp. 201-214.

[82] A. A. Heidari, R. A. Abbaspour and A. R. Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks", *Neural Computing and Applications* (2015) pp. 1-29.

[83] X. Lei, M. Du, J. Xu and Y. Tan, "Chaotic fruit fly optimization algorithm", *Advances in Swarm Intelligence: ASI-2014*, Vol. 8794 of Lecture Notes in Computer Science (Springer Switzerland, 2014), pp. 74-85.

[84] A. Kanso and N. Smaoui, "Logistic chaotic maps for binary numbers generations", *Chaos, Solitons & Fractals* 40(5) (2009) pp. 2557-2568.

[85] L. dos Santos Coelho, J. G. Sauer and M. Rudek, "Differential evolution optimization combined with chaotic sequences for image contrast enhancement", *Chaos, Solitons & Fractals* 42(1) (2009) pp. 522-529.

[86] B. M. Waxman, "Routing of multipoint connections", *IEEE Journal on Selected Areas in Communications* 6(9) (1988) pp. 1617-1622.

[87] C. Li, C. Cao, Y. Li and Y. Yu, "Hybrid of genetic algorithm and particle swarm optimization for multicast QoS routing", *in Proc. IEEE Int. Conf. on Control and Automation (ICCA)* (2007), pp. 2355-2359.

[88] H. Wang, X. Meng, S. Li and H. Xu, "A tree-based particle swarm optimization for multicast routing", *Computer Networks*, 54(15) (2010) pp. 2775-2786.

[89] C. Xi-hong, L. Shao-wei, G. Jiao and L. Qiang, "Study on QoS multicast routing based on ACO-PSO algorithm", *In: Proceedings of 2010 IEEE international conference on Intelligent Computation Technology and Automation (ICICTA)*. (2010) pp. 534-537.

[90] L. Zhang, LB. Cai, M. Li and FH. Wang, "A method for least-cost QoS multicast routing based on genetic simulated annealing algorithm", *Computer Communications*, 32(1) (2009) pp. 105-110.

[91] H. Xing, X. Liu, X. Jin, L. Bai and Y. Ji, "A multi-granularity evolution based Quantum Genetic Algorithm for QoS multicast routing problem in WDM networks", *Computer Communications*, 32(2) (2009) pp. 386-393.

[92] KH. Han and JH. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization", *Evolutionary Computation, IEEE Transactions on* 6(6) (2002) pp. 580-593.

[93] KH. Han and JH. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem", *In Evolutionary Computation, 2000*. Proceedings of the 2000 Congress on 2 (2000) pp. 1354-1360).

[94] M. Ying, "Quantum computation, quantum theory and AI", *Artificial Intelligence* 174(2) (2010) pp. 162-176.

[95] T. Hey, "Quantum computing: an introduction", *Computing & Control Engineering Journal* 10(3) (1999) pp. 105-112.

[96] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines", *Journal of Statistical Physics* 22(5) (1980) pp. 563-591.

[97] LA. Wong, H. Shareef, A. Mohamed and AA. Ibrahim, "Novel quantum-inspired firefly algorithm for optimal power quality monitor placement", *Frontiers in Energy* 8(2) (2014) pp. 254-260.

[98] C. Patvardhan, S. Bansal and A. Srivastav, "Solving the 0-1 Quadratic Knapsack Problem with a competitive Quantum Inspired Evolutionary Algorithm", *Journal of Computational and Applied Mathematics* 285 (2015) pp. 86-99.

[99] X. Yuan, P. Wang, Y. Yuan, Y. Huang and X. Zhang, "A new quantum inspired chaotic artificial bee colony algorithm for optimal power flow problem", *Energy Conversion and Management*, 100 (2015) pp. 1-9.

[100] HL. Liu, "Acoustic partial discharge localization methodology in power transformers employing the quantum genetic algorithm", *Applied Acoustics* 102 (2016) pp. 71-78.