I. Introduction

Comme la gestion de l'énergie dans les réseaux de capteurs sans fil (RCSFs) est très exigeante, plusieurs chercheurs sont actuellement engagés non seulement dans le développement des technologies requises au niveau des différentes couches de la pile protocolaire des réseaux de capteurs, mais aussi pour la conception des méthodes et des algorithmes qui assurent une gestion meilleure.

Au cours de ces dernières années, les chercheurs ont arrivés à représenter un réseau sous forme d'un graphe G, où chaque capteur sera représenté par un nœud et la connexion entre deux nœuds est représentée par une arête. Néanmoins, cela ne résout pas les problèmes de routage pour la gestion d'énergie dans les RCSFs, alors ils ont pensé à la notion du plus court chemin dans G. Cette approche a donné naissance à un nouveau problème nommé le problème de ensemble dominant (Connected Dominating Set CDS).

Cette approche a été largement utilisée pour construire une ossature de routage dans les réseaux de capteurs [20] [39]. Elle se concentre sur les nœuds au lieu des arêtes afin de minimiser la consommation d'énergie. Tandis que l'énergie dissipée lors du routage des données est directement liée aux arêtes.

Le problème de l'arbre dominant (Dominating Tree Problem ou DTP) a donc été formulé dans le but de pallier aux lacunes du CDS [13] [14] .

Dans ce chapitre, nous présenterons le problème de l'arbre dominant, puis nous synthétiserons quelques travaux effectués pour sa résolution, enfin nous clôturerons ceci par une conclusion.

II. Définition du problème:

Le problème de l'arbre dominante nous incite à trouver un arbre de pondération de poids total minimal DT. Le principe est que chaque réseau sans fil sera représenter par un graphe pondéré non orienté G=(V,E,w), où V est l'ensemble des nœuds du réseau, E l'ensemble des liaisons de communication du réseau et w un poids non négatif attribué pour chaque arête e=(u,v). Chaque nœud $v \in V$, est soit dans DT, soit adjacent à un nœud déjà dans DT. La complexité de la construction d'un arbre dominant est Np-hard.

D'autre part, la résolution du problème DT peut générer un backbone de routage pour les protocoles de diffusion puisque chaque nœud n'a pas à construire son propre arbre de diffusion, alors ils utilisent le backbone virtuel pour réduire la surcharge de message et le poids du backbone qui représente la consommation d'énergie minimisée. Comment réglé le problème de l'arbre dominant (DTP)?

III. Complexité et approximation:

Le problème de l'arbre dominant (DTP) est une variante récente d'ensemble dominants en théorie des graphes. Il est prouvé qu'il est un problème N P-Hard [13, 14].

L'arbre de Steiner est souvent associé à des problème de conception de réseau. Le principe de conception de cet arbre est très proche du principe de conception de l'arbre dominant. Il consiste à construire un arbre de poids minimal r nommé DST, en veillant à ce qu'il y ait au moins un chemin allant de la racine à chaque nœud de l'ensemble D tel que D est un ensemble dominant minimal appartenant à V dans G.

Dans [13] ils ont montré que le problème DT peut être réduit au problème DST en un temps polynomial. Rappelons que l'algorithme de construction de DST avec la conservation du rapport de performance nous permet de l' appliquer au problème DT de la façons suivante:

- Dans le problème DT nous introduisons un sommet factrice v^* (un faux sommet) pour chaque sommet réel $v \in V$, puis nous ajoutons les arcs pour tous les voisins de v (y compris v lui-même) au nœud factice v, et fixons le poids à zéro pour toute arrête nouvellement ajoutées. De même, pour les arêtes originales, l'arête est bidirectionnelle et le poids identique à celui du graphe original. Dans le nouveau graphe orienté G' = (V', E', w'), si nous choisissons un $v' \in V'$ arbitraire comme root r, et que tous les sommets factices soient des terminaux, nous pouvons obtenir un DST basé sur un algorithme DST existant [14] et obtenir ainsi un DT de G. Il est clair que la réduction est réalisée en temps linéaire. Cette réduction est illustrée sur la Figure2.1

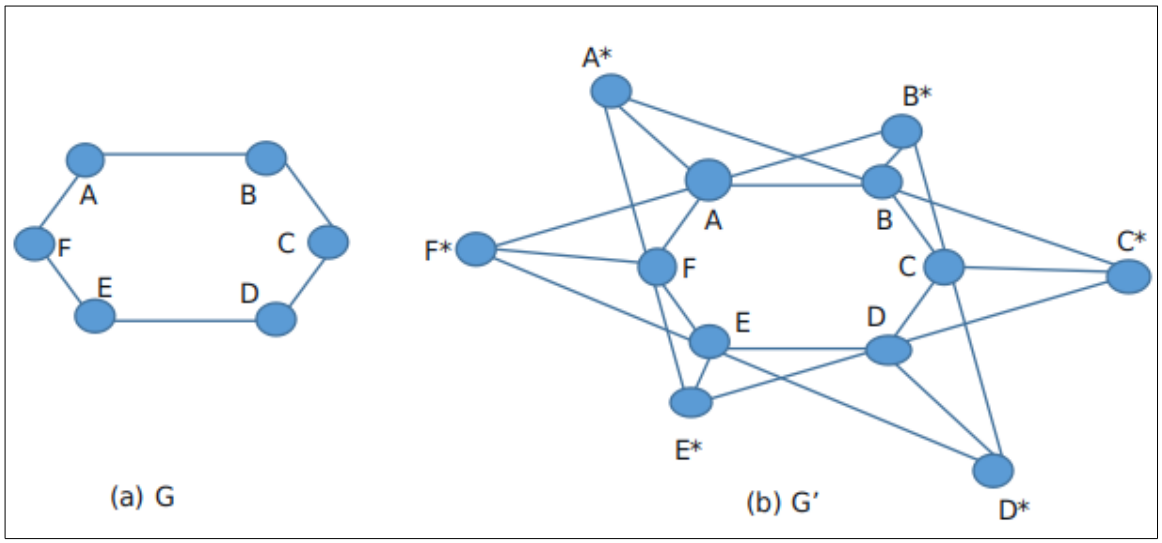

figure 3.1 Exemple de réduction du graphe G en G'

Lemme 1: Si r est dans le DT optimal, alors le DT introduit par DST aura le même poids optimal. En outre, l'utilisation de l'algorithme d'approximation pour le problème DST aura le même rapport d'approximation pour le problème DT.[13]

Preuve: Supposons qu'il existe une arborescence dominante optimale DT^*, nous pouvons faire en sorte que tous les nœuds de DT^* apparaissent dans le DST. Puisque r est dans DT^*, pour chaque nœud de DT, au moins un de ses voisins doit être dans DT, et pour tout terminal, nous pouvons ajouter un front dirigé depuis ce voisin. Et nous savons que ces arêtes sont pondérées à zéro, de sorte que le poids de cette DST est égal à $w(DT^*)$. Supposons qu'il existe un DST optimal ST^*, alors nous avons $w(DT^*) \geq w(ST^*)$. En outre, pour un DST optimal, nous pouvons éliminer toutes ces arêtes pondérées par zéro. Étant donné que tous les nœuds terminaux ont un chemin allant de r à eux, chaque nœud aura au moins un de ses voisins figurant dans cet arbre, ce qui signifie qu'il s'agit d'un DT. Nous avons donc $w(ST^*) \geq w(DT^*)$. En conclusion, nous avons $w(DT^*) = w(ST^*)$. Cela implique à son tour que nous pouvons obtenir le même rapport de performances pour le problème DT que pour le DST.[13]

Une approximation poly-logarithmique a été fourni par les auteurs dans [15]. L'algorithme qui nous permet d'obtenir le rapport de performance est le suivant :

Algorithme 1: approximation de DT

---

1. **Entrées**: un graphe général pondéré non orienté G=(V, E, w)
2. **Sortie**: Un DT de G
3. Initialiser une liste pour sauvegarder le DT et son poids
4. Trouver un sommet factice u en G avec un degré minimum
5. Transformer G en G' = (V', E', w') en utilisant la technique de transformation
6. **pour** chaque nœud v où (u, v) ∈ E **faire**
7. Exécution de l'algorithme DST [15] dans G et définir v comme racine r pour obtenir un DT
8. Sauvegarder le DT et son poids dans la liste
9. **Fin pour**
10. Retourner le DT avec poids minimum dans la liste

L'algorithme 1 nous permet de transformer le problème DT en DST. Puis nous pouvons appliquer l'algorithme dans [15] qui va nous permettre par la suite de résoudre le problème DST et de trouver une solution pour DT. Cela n'est possible que si le sommet racine r choisie dans l'algorithme 1 soit le meilleur choix (r doit être dans le DT optimal). Au final, nous pouvons revenir à l'état initial de notre graphe de départ en supprimant les nœuds factrices ajoutés ainsi que leurs arêtes.

IV. Notions de base:

IV. 1. Arbre couvrant de poids minimal (MST):

un graphe pondéré non orienté G=(V,E,w), où V est l'ensemble des nœuds du réseau, E l'ensemble des liaisons de communication du réseau et W un poids non négatif attribué sur chaque arête e=(u,v).

a) Qu'est-ce qu'un arbre couvrant (Spanning Tree ST)?

un arbre couvrant d'un graphe G est un arbre qui s'étend à inclure tous les sommets de G.

b) Qu'est-ce qu'un arbre couvrant de poids minimal (MST)?

Le coût de l'arbre couvrant est la somme des poids de toutes les arêtes de l'arbre. Il peut y avoir beaucoup d'arbres couvrant. L'arbre couvrant de poids minimal est l'arbre couvrant dont le coût est minimal parmi tous les arbres couvrant. Il peut également y avoir beaucoup d'arbres couvrant minimum.

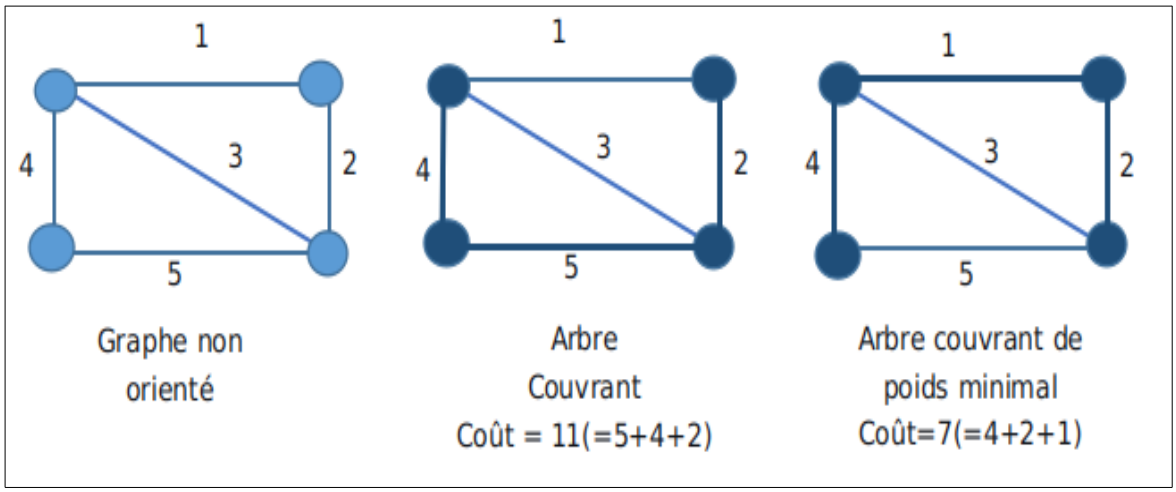

Figure 3.2: Exemple d'un arbre couvrant de poids minimal

Il existe deux algorithmes les plus réputé pour l construction d'un arbre couvrant de poids minimal : **Kruskal** et **Prim**. Nous allons se contenter de l'algorithme de Prim car d'après [40], c'est algorithme le plus efficace.

Algorithme de Prim:

C'est un algorithme glouton qui tente à construire un arbre couvrant de poids minimal dans un graphe non orienté, connexe. Nous allons présenter dans ce qui suit l'algorithme de Prim ainsi qu'un exemple qui illustre le déroulement de ce dernier.

Algorithme 2: Prim

1. **Entrées**: un graphe général pondéré non orienté G=(V, E, w)
2. **Sortie**: Un arbre couvrant de poids minimal T
3. sommets : un sommet de G qu'on choisit } **Initialisation de T**
4. arêtes : aucune
5. **Répéter**
6. Trouver toutes les arêtes de G qui relient un sommet de T et un sommet extérieur à T
7. Sélectionner l'arête ayant le plus petit poids parmis l'ensemble des arêtes trouvées.
8. Ajouter à T cette arête et le sommet correspondant.
9. **jusqu'à** tous les sommets de G soient dans T
10. Retourner T

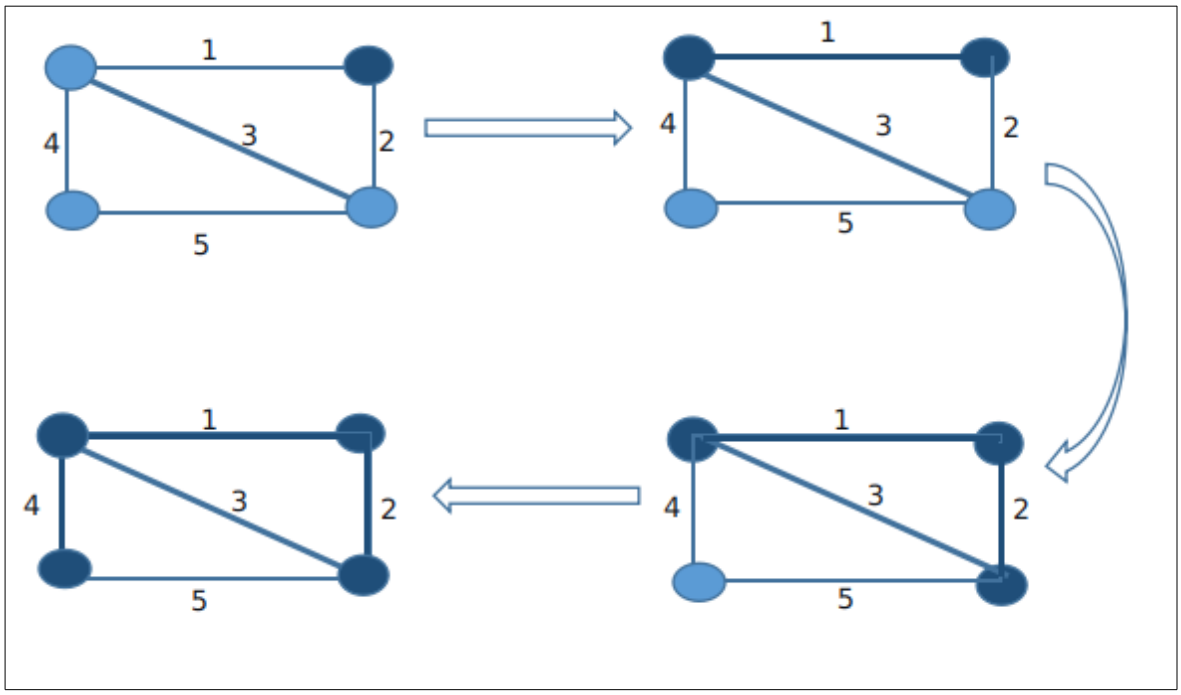

Figure 3.3: illustration de l'algorithme de Prim.

IV. 2. L'élagage (Pruning):

Cette méthode est apparue pour la première fois dans [13,14] et a été améliorée dans [16]. Dans [16], la fonction de pruning consiste à vérifié si un sommet $v \in DT$ de degré un peut être élagué sans que DT soit non connexe. Si cela est vérifiée, ce nœud alors va devenir non dominant et l'arête incidente peut également être retirer de DT, ce qui réduit par la suite le poids de DT. Dans un second lieu, cette fonction vérifie tous les sommets de DT dans l'ordre FIFO (First In First Out). Si le degré d'un sommet dominant v est égale à un, il est vérifié si tous les sommets adjacents à v sont également adjacents à d'autres sommets

dominants dans DT. Si c'est le cas supprimer ce sommet de DT et l'arête incidente aussi. Cette méthode de pruning est largement différente que celle utilisée dans [13] dans laquelle un arbre couvrant est d'abord construit et, par conséquent, tous les sommets de feuille peuvent être élagués sans autre vérification.

V. Les travaux relatifs

Le problème de l'arbre dominant est apparu récemment (2008) introduit par [14] pour la construction d'un backbone de routage pour les réseaux sans fil avec une consommation d'énergie minimal, ce qui justifie le manque de la littérature pour ce problème.

La consommation d'énergie est prise en compte qu'au niveau des nœuds et non les arêtes. En réalité, l'énergie consommée par chaque bord affecte directement la consommation d'énergie d'un système de routage. Par conséquent, pour minimiser la consommation d'énergie d'un système de routage, nous devons prendre en compte l'énergie consommée par chaque bord [16].

Au meilleur de notre connaissance, aucune formulation de programmation en nombres entiers n'a été étudiée dans la littérature pour le DTP autre que celle classique introduite dans [13].

En 2010, puisque le problème de l'arbre dominant est NP-hard, Yilin Shen · My T. Thai ont fait l approximation du problème DT au problème DST . Cependant, la complexité temporelle est assez élevée, du fait que la construction d'un DST entraîne généralement une longue durée d'exécution. De ce point de vue, une heuristique peu complexe en temps est très attendue ce qui a rendu la littérature sur le DTP portent principalement sur des algorithmes heuristiques, tels que la recherche par voisinage variable (VNS) et des algorithmes méta-heuristiques, tels que l'optimisation génétique, la colonie d'abeilles artificielles et les colonies de fourmis.

Nous allons considérer pour la suite un graphe G = (V, E, w), non orienté, pondéré, où : V est l'ensemble des sommets et E est l'ensemble des arêtes du graphe G.

V.1 L'heuristique pour le problème de l'arbre dominant (H_DT)

En 2013, Shyam Sundar et Alok Singh proposent un nouvel algorithme heuristique (H_DT) pour la construction d'un arbre dominant DTP. Entre autre, un arbre couvrant minimal est construit sur le sous-graphe de G induit par l'ensemble des sommets dominants de DT à l'aide de l'algorithme de Prim [40]. De cette façon, le poids total de DT peut être réduit davantage. La raison est que même après pruning, le poids total de DT peut ne pas être minimum en raison de la sélection d'arêtes incorrectes lors de la construction de DT. Sur un ensemble donné de sommets dominants, de nombreux arbres dominants peuvent être construits en G. De toute évidence, un arbre dominant obtenu à l'aide de l'algorithme de Prim aura un coût minimum parmi tous ces arbres dominants. Après cela, la procédure d'élagage est à nouveau appliquée, puis un arbre couvrant minimal est à nouveau construit sur le sous-

graphe de G induit par l'ensemble des sommets dominants de DT à l'aide de l'algorithme de Prim. Dans ce qui suit nous allons donner le pseudo-code de l'heuristique H_DT et expliquer brièvement son fonctionnement:

Algorithme 3: pseudo-code H_DT

---

1. **Entrées**: un graphe pondéré, non orienté et connecté G=(V, E)
2. **Sortie**: sortie: un arbre dominant DT$\subseteq$ E
3. Initialise DT $\leftarrow \varnothing$
4. **Pour** chaque sommet i $\in$ V **faire**
5. **Mark [i] $\leftarrow$ 0;**
6. **Fin** pour
7. Calcule les chemins les plus courts entre toutes les paires de sommets de G;
8. Triez toutes les arêtes de E en ordre non décroissant en fonction de leur poids.
9. **Tant que** tous les sommets de V ne sont pas marqués **faire**
10. Sélectionnez une arête non sélectionnée e_{ij} avec un coût minimal, dont au moins un point d'extrémité n'est pas marqué
11. **Pour** chaque sommet v adjacent à i **faire**
12. Mark[v] $\leftarrow$ 1
13. **Fin** pour
14. **Pour** chaque sommet v adjacent à j **faire**
15. Mark[v] $\leftarrow$ 1
16. **Si** DT =$\varnothing$ ou e_{ij} est adjacent à une arête dans DT **alors**
17. DT $\leftarrow$ DT $\cup$ {e_{ij}}
18. **Sinon**
19. Trouver le chemin le plus court ST reliant DT et {i, j}
20. DT $\leftarrow$ DT $\cup$ ST $\cup$ {e_{ij}}
21. **Pour** chaque sommet v en ST **faire**
22. Mark [v] $\leftarrow$ 1
23. **Pour** chaque sommet k adjacent à v **faire**
24. Mark [k] $\leftarrow$ 1
25. **Fin** Pour,
26. **Fin** pour,
27. **Fin** tant que
28. Appliquer la procédure d'élagage sur DT
29. Reconnectez les sommets dominants de DT en y construisant un arbre couvrent de poids minimal.
30. Appliquer l'élagage sur le DT;

31. Reconnectez les sommets dominants de DT en y construisant un arbre couvrent de poids minimal.
32. **Retourner** DT;

Cette heuristique se décompose en deux phases principales. La première phase est la phase d'initialisation, où le chemin le plus court entre toute paire de nœuds est calculé.
L'ensemble des sommets est étiqueté à "non marqué" et l'ensemble d'arêtes constituant DT est égale à l'ensemble vide. Ensuite, une phase itérative commence, où, à chaque itération, une arête e_{ij} de poids minimal avec au moins une extrémité "non marquée" est choisie. Les voisins des extrémités de l'arête ajoutée seront "marqués". Une vérification est ensuite effectuée pour déterminer si l'arête e_{ij} est adjacente à DT ou non. Si ce n'est pas le cas, alors un chemin de coût minimal est déterminé pour établir un lien entre le DT partiellement construit et l'arête sélectionnée. Les arêtes constituant ce dernier sont ajoutés à DT et les nœuds appartenant à ce chemin ainsi que leurs voisins "non-marqués" seront "marqués". Dès que tous les sommets dans V seront marqués (condition d'arrêt vérifiée), la procédure d'élagage suivit de l'algorithme de Prim sont appliqués.
En outre, Shyam et Alok travaillent sur deux approches méta-heuristiques qui donnent des résultats satisfaisants , et ils ont essayé d'améliorer cet heuristique dans les années qui suivent.

V.2 Colonie d'abeilles artificielle pour le DTP (ABC_DT))

L'algorithme de colonie d'abeilles artificielles ou Artificial Bee Colony (ABC) a été introduit par Dervis Karaboga en 2005 [41] et développé depuis 2005 par Karaboga et Basturk [42] pour les problèmes d'optimisation continue. C'est un algorithme à population, d'inspiration naturaliste, basé sur le butinage des abeilles.

a) Génération de la solution initiale :

C'est un processus itératif pour générer des solutions initiales (arbres dominants) faisables, où à chaque itération, pour une probabilité P, une arête de poids minimal est choisie pour le nœud de départ qui a été choisi lui aussi aléatoirement sinon l'arête est choisie aléatoirement selon la méthode de sélection de roulette (wheel) afin de maintenir un équilibre entre qualité et diversité de la solution. A ce stade, un arbre dominant de graphe G est construit. Puis,la procédure d'élagage (décrite dans la section IV.2) est appliquée à DT de manière répétée jusqu'à ce qu'il ne soit plus possible d'élaguer un sommet dominant avec un degré un, ainsi que la "procédure MST" (de la section IV.1), est appliquée sur la solution obtenue.

b) Probabilité de sélectionner une source alimentaire :

C'est à travers la méthode de sélection de "tournoi" dite "Binary Tournament Selection Method"(BTS), que les abeilles Onlooker (spectateurs) choisissent une solution de la

population. Avec la probabilité b t, la meilleure est choisie, sinon la pire solution est sélectionnée avec probabilité 1-bt.

Algorithme 4: Pseudo-code de ABC DT

1. Générer n e solutions aléatoires E_1 , E_2 , ..., E_{ne}
2. Meilleure_sol ← La meilleure solution parmi E_1 , E_2 , ..., E_{ne}
3. Tant que La condition de terminaison est non satisfaite Faire
4. Pour (Chaque abeille employée i dans ne) Faire
5. E' ← determination_d_ une_solution_voisine (E_i) ;
6. Si (E' est meilleure que E_i) Alors
7. $E_i \leftarrow$ E' ;
8. Sinon
9. Si (E_i n'a pas changé après un nombre limité d'itérations) Alors
10. Remplacer E_i avec une solution générée aléatoirement ;
11. Fin Si
12. Fin Si
13. Si (E' est meilleure que Meilleure_sol) Alors
14. Meilleure_sol $\leftarrow E_i$;
15. Fin Si
16. Fin Pour
17. Pour (Chaque abeille onlooker i dans n_0) Faire
18. $s_i \leftarrow$ méthode de sélection de tournoi binaire (E_1 , E_2 , ..., E_{ne})
19. $O_{ni} \leftarrow$ determination_d_une_solution_voisine (E_{si})
20. Si (O_{ni} est meilleure que Meilleure_sol) Alors
21. Meilleure_sol $\leftarrow O_{ni}$;
22. Fin Si
23. Fin Pour
24. Pour (Chaque abeille onlooker i dans n_0) Faire
25. Si (O_{ni} est meilleure que E_{si}) Alors
26. $E_{si} \leftarrow O_{ni}$;
27. Fin Si
28. Fin Pour
29. Fin Tant que

c) Détermination du voisinage d'une source alimentaire:

La détermination d'une solution voisine se fait selon deux procédures qui s'excluent mutuellement. Après qu'une solution s est crée une copie de cette dernière est affectée à s'. Après cela, Avec une probabilité P, une solution voisine est générée en appliquant la méthode PDE, qui supprime une arête de façon aléatoire sinon la méthode PANDV est appelée pour ajouter un sommet aléatoirement. L'application de ces deux méthodes fait que la solution devient infaisable, ce qui fait que chacune d'entre elles procède différemment pour la rendre faisable. Le pseudo-code de l'algorithme de détermination de voisinage d'ABC_DT est représenté dans l'Algorithme 5 ci-dessous.

Algorithme 5: Pseudo-code de détermination_d_une_solution_voisine

---

1. **Entrées** : Une solution s ;
2. **Sorties** : Une solution voisine s ;
3. Créer une copie s' de s ;
4. **Si** (u 01 < q 1) **Alors**
5. Appliquer la procédure PDE sur s ;
6. **Sinon**
7. Appliquer la procédure PANDV sur s ;
8. Fin Si
9. Appliquer la procédure pruning sur s ;
10. Reconnecter les sommets dominants de s' en construisant un arbre de poids minimum sur eux (MST) ;
11. Appliquer la procédure pruning sur s ;
12. Reconnecter les sommets dominants de s' en construisant un arbre de poids minimum sur eux (MST) ;

V.3 Algorithme génétique stationnaire :

C'est un algorithme génétique, dit "stationnaire" (Steady State Genetic Algorithm SSGA)[43], SSGA travaille sur la méthode de remplacement de la population à l'état d'équilibre [44]. Cet algorithme commence itérativement par la sélection de deux parents, exécuter le croisement et la mutation pour générer une solution (enfant) qui remplace le pire individu de la population. Dans ce qui suit, nous allons présenter les principales caractéristiques de cet algorithme.

a) Génération de la solution initiale:

C’est un processus itératif, qui procède par l’ajout des arêtes de manière aléatoire dans le but d’avoir un arbre dominant. Après que cet arbre soit créé les deux procédures "pruning" de la section IV.2 et "MST" (voir la section IV.1) sont appliquées.

b) Fitness:

La Fitness d’une solution est la même que la fonction objectif d’une solution, telle que la valeur de la fonction objective est la somme des poids des arcs de la solution. L'aptitude de chaque solution est calculée, où la fonction d'aptitude d'une solution est identique à la fonction objective d'une solution.

c) La sélection:

La méthode de sélection de tournoi binaire. Cette méthode est utilisée pour la sélection d’un individu pour la mutation. Elle est utilisée aussi deux fois de façon consécutive pour la sélection de deux chromosomes (individus) comme parents pour l’opération de croisement (cross-over); avec la probabilité P, l’individu avec une meilleure fitness est sélectionné, sinon, c’est le plus mauvais qu’il l’est (avec la probabilité 1−P).

d) Le croisement (cross-over):

Afin de générer une solution enfant C, qui est initialement vide, deux parents p1 et p2 sont sélectionnés à travers la méthode BTS.

Les sommets constituant la solution enfant seront choisis, en alternance, des deux ensembles des nœuds dominants des deux parents. En effet, un sommet ne sera rajouté que s’il n’existe pas déjà dans C. Maintenant, il y a une vérification sur l’existence d’une arête reliant le sommet choisi à l’arbre partiellement construit. Si c’est le cas, alors il est ajouté à C et est étiqueté à marqué ainsi que tous ses voisins. Sinon, un chemin optimal, appelé chemin potentiel, est calculé en fonction du nombre de nœuds non couverts appartenant au chemin sur le coût de celui-ci. En effet, ce chemin vise à maximiser le nombre de nœuds à rajouter dans C. De plus, cela permet d’éviter un opérateur réparateur et d’assurer d’avoir directement un arbre dominant de bonne qualité. Ce processus itératif est répété jusqu’à ce que tous les sommets soient "marqués".

e) Mutation:

Cette partie commence par la sélection d’une solution initiale en utilisant la méthode BTS. Un petit ensemble V_m de sommets non dominants est sélectionné aléatoirement de V\C. Itérativement, un sommet $v_s \in V_m$ est de degré 1 dans G et ayant un chemin de coût

minimal avec un sommet $v_c \in C$. Cette procédure se poursuit jusqu'à ce que V_m devient vide. L'algorithme de Prim est ensuite appliqué à l'ensemble de sommets dominants de C.

f) Politique de remplacement:

Dans cette politique de remplacement, l'unicité de la solution enfant C, nouvellement créée, est examinée par rapport aux individus de la population. Dans le cas où C se révèle être différent de tous les individus de la population, alors il remplace le plus mauvais d'entre eux, sinon, il est rejeté.

Algorithme 6: Pseudo-code de l'algorithme SSGA

---

1. Générer une population (pop_{size}) de solutions s_1 , s_2 , ..., s_{pop_size} aléatoirement ;
2. Meilleure_sol ← La meilleure solution de la population ;
3. Tant que (La condition de terminaison est non satisfaite) Faire
4. Si ($u_{01} < p_c$) Alors
5. p1 ← méthode de sélection de tournoi binaire s_1, s_2 , ..., s_{pop_size} ;
6. p2 ← méthode de sélection de tournoi binaire s_1 , s_2 , ..., s_{pop_size} ;
7. Enfant ← Opérateur de Croisement (p1, p2) ;
8. Sinon
9. p1 ← méthode de sélection de tournoi binaire s_1 , s_2 , ..., s_{pop_size} ;
10. Enfant ← Opérateur de Croisement (p1) ;
11. Fin Si
12. Si (Enfant est un DT partiel) Alors
13. Appliquer procédure de réparation sur Enfant ;
14. Fin Si
15. Appliquer la procédure pruning sur le DT de Enfant ;
16. Appliquer l'algorithme de Prim pour construire un arbre de poids minimal sur le sous-graphe de G induit par l'ensemble des sommets dominants de DT ;
17. Appliquer la procédure pruning sur la DT de Enfant ;
18. Si (Enfant est meilleure que Meilleure_sol) Alors
19. Meilleure_sol ← Enfant ;
20. Fin Si
21. Appliquer la politique de remplacement ;
22. Fin Tant que
23. Retourner Meilleure_sol ;

V.4 L’algorithme évolutionnaire avec mutation guidée EA/G

L’algorithme évolutionnaire avec mutation guidée (Evolutionary algorithm with guided mutation [45] est relativement un nouveau membre de la classe des algorithmes évolutionnaires. Il a été développé par Zhang et al. (2005) pour surmonter les inconvénients de deux algorithmes évolutifs, à savoir algorithmes génétiques (GAs) et estimation d'algorithmes de distribution (EDAs). Il est dit hybride car il utilise les caractéristiques des deux algorithmes cités précédemment.

En effet, Les Gas génère une progéniture à partir des parents sélectionnés en appliquant les opérateurs génétiques tels que le croisement et la mutation. Ces algorithmes n’utilisent que les informations de localisation des solutions et ne font pas usage des informations globales sur l’espace de recherche qui peuvent être collectées en gardant la trace de toutes les solutions générées depuis le début de l’algorithme. Dans ce contexte, "l’information de localisation d’une solution" revient à dire,l’information qui peut identifier de façon unique une solution dans l’espace de recherche de toutes les solutions. Par exemple, dans le cas du DTP, l’ensemble des arêtes formant le DT constitue son information de localisation car cette dernière peut l’identifier de façon unique.

Par ailleurs, les EDAs ne s’appuient que sur un modèle de probabilité pour générer une solution.

Le modèle de probabilité caractérise la distribution des solutions potentielles dans l’espace de recherche. Ce dernier, est mis à jour à chaque génération en utilisant des informations globales sur l’espace de recherche extraites des individus de la génération courante. Une solution est générée en se basant sur ce modèle de probabilité. Contrairement aux GA, les EDA n'utilisent pas directement les informations de localisation des solutions [46].

Pour conclure EA / G utilise un opérateur de mutation, appelé mutation guidée (GM), pour générer des descendants (solutions). La mutation guidée génère une nouvelle solution tenant compte à la fois des informations de localisation relatives à la solution mère ainsi que des informations statistiques globales relatives à l'espace de recherche. En d'autres termes, la génération d’une solution se fait par l’échantillonnage d’un modèle de probabilité caractérisant les informations statistiques globales et en copiant des éléments à partir de son parent.

EA/G hybride pour DTP (EA/G-MP)

Cette méthode s’inspire de l’approche de Zhang et al. (2005) pour le problème de la clique maximale (MCP) et est développée en 2014 par les mêmes chercheurs. Les résultats obtenues par EA/G ont été amélioré par cette nouvelle méthode grâce à l’utilisation des deux procédures à savoir la reconnexion des nœuds de la solution via un MST (minimum spanning

tree) et un opérateur d'élagage. Néanmoins, chacune de ces procédures est appliquée deux fois dans l'ordre MST → Pruning → MST → Pruning.

Pour commencer EA/G-MP, nous devant définir les voisins de chaque nœud v ∈ V et de calculer le chemin le plus court entre toute paire de nœuds. Avant de passer aux caractéristiques de l'algorithme, nous présentons dans ce qui suit un pseudo-code pour la procédure « pruning » pour cette approche suivit d'une explication pour ce dernier :

Algorithme 7: Pseudo-code de la procédure pruning

---

1. $R_n \leftarrow \{ v : v \in DN \text{ et } | ON(v) \cap DN | = 1 \text{ et } CN(v) \subseteq (U_{u \text{ in} DN \setminus \{v\}} ON(u))\}$;
2. Tant que ($R_n \neq \varnothing$) Faire
3. v ← Selectionner_Noeud (R_n) ;
4. DN ← DN \ { v } ;
5. $R_n \leftarrow \{ v : v \in DN \text{ et } | ON(v) \cap DN | = 1 \text{ et } CN(v) \subseteq (U_{u \in DN \setminus \{v\}} ON(u))\}$
6. Fin Tant que
7. Retourner DN ;

Puisque le rôle de la procédure d'élagage consiste à éliminer les nœuds redondants dans l'arbre dominant. Pour cela, Cet procédure commence tous d'abord à sélectionner tous ces nœuds où le degré de chaque nœud v dans DN est égal à un et tous les nœuds voisins non dominants du nœud v sont couverts par d'autres nœuds dominants (autres que le nœud v) dans DN. Puis, un processus itératif commence: lors de chaque itération, un nœud de l'ensemble R_n est sélectionné (la sélection se fait selon l'ordre FIFO) et est supprimé de DN ainsi que de R_n.

De plus, tout en construisant un arbre dominant, la convention suivie, est celle où : un nœud prend la couleur BLANCHE s'il n'est pas un nœud dominant et n'est pas adjacent à un nœud dominant, GRISE si ce nœud est adjacent à un nœud dominant mais il ne l'est pas ,et un nœud appartenant à l'arbre est supposé avoir la couleur NOIRE. Initialement, tous les nœuds sont supposés avoir la couleur BLANCHE. Lorsque la construction de l'arbre dominant est terminée, les nœuds appartenant à l'arbre auront la couleur NOIRE et tous les autres nœuds (non dominants) auront la couleur GRISE.

a) Solution initiale:

Avant de présenter cette méthode commençons à définir quelques variables utilisées:

W_n : ensemble de nœuds BLANCs (W_n = V Initialement).

I_u : l'ensemble des nœuds non dominants.

n_b : Ensemble de nœuds BLANCs voisins à v i.e. : nb = $ON(v) \cap W_n$.

La méthode de génération de solution initiale est la même que celle utilisée dans Sundar et Singh (2013)(ABC_DT) en portant deux modifications : La première, est que seuls les arrêtes $e_{v,u}$ sont considérés où $u \in I_u$ a au moins un voisin BLANC, c'est-à-dire où $| ON (u) W_n | \geq 1$. La deuxième modification, est qu'avec la probabilité $1-\varphi$, nous sélectionnons les arrêtes uniformément au hasard au lieu d'utiliser la méthode de sélection de ABC_DT.

Algorithme 8: Pseudo-code de la solution initiale de EA/G-MP

---

1. Initialement tous les nœuds dans V sont colorés en BLANC ;
2. $W_n \leftarrow V$; $DT \leftarrow \varnothing$; $DN \leftarrow \varnothing$; $I_u \leftarrow \varnothing$;
3. $v \leftarrow$ random (W_n) ;
4. $DN \leftarrow DN \cup \{ v \}$;
5. Mettre v NOIR ;
6. $nb \leftarrow ON (v) \cap W_n$.
7. $W_n \leftarrow W_n \setminus (nb \cup \{ v \})$;
8. Mettre tous les noeuds $\in$ nb GRIS ;
9. $I_u \leftarrow nb$;
10. Tant que ($W_n \neq \varnothing$) Faire
11. Générer un nombre aléatoire u01 tel que $0 \leq u01 \leq 1$;
12. Si ($u_{01} < \varphi$) Alors
13. $(v, u) \leftarrow$ argmin $v \in DN, \{ u \in I_u | ON (u) \cap W_n | \geq 1 \} w (v, u)$;
14. Sinon
15. $v \leftarrow$ random (DN) ;
16. $u \leftarrow \{ u :$ random (I_u) et $| ON (v) \cap W_n | \geq 1 \}$;
17. Fin Si
18. $DT \leftarrow DT \cup \{ e_{v,u} \}$;
19. $DN \leftarrow DN \cup \{ u \}$;
20. Mettre u NOIR ;
21. $I_u \leftarrow I_u \setminus \{ u \}$;
22. $nb \leftarrow \{ u : u \in ON (u) \cap W_n \}$;
23. Mettre tous les noeuds $\in$ nb GRIS ;
24. $I_u \leftarrow I_u \cup nb$;
25. $W_n \leftarrow W_n \setminus nb$;
26. Fin Tant que
27. Appliquer la procédure pruning sur les nœuds dans DT ;
28. Reconnecter les nœuds dans DT via MST ;

29. Retourner DT ;

---

b) Initialisation et mise à jour du vecteur de probabilités

Le vecteur de probabilité est définit comme suit: p = {p1, p2, ..., p|V|} $[0, 1]^{|V|}$, où | V | est le nombre de nœuds dans l'ensemble V (nombre de nœuds dans le graphe G), p est la probabilité que le nœud v ∈ V soit présent dans un arbre dominant. Ce vecteur est initialisé avec le rapport entre le nombre de solutions initiales contenant ce nœud et le nombre total de solutions initiales. Le pseudo-code est présenté dans l'algorithme 9.

Algorithme 9: Pseudo-code de l'initialisation du vecteur de probabilités de EA/G-MP.

---

1. nv ← nombre de solutions initiales contenant le nœud v, ∀ v ∈ V ;
2. pv ← $\frac{nv}{Np}$, ∀ v ∈ V ;

A chaque génération g, un ensemble parent "parent (g)" est formé en sélectionnant les meilleures solutions L dans la population actuelle (g). Une fois que le parent (g) est formé, il est utilisé pour mettre à jour le vecteur de probabilité p. Le pseudo-code pour la mise à jour du vecteur de probabilité est donné dans l'algorithme 10:

Algorithme 10: Pseudo-code de la mise à jour du vecteur de probabilités de EA/G-MP.

---

1. nv ← nombre de solutions dans parent(g) contenant le nœud v, ∀ v ∈ V ;
2. pv ←(1 − λ) pv+ λ $\frac{nv}{L}$, ∀ v ∈ V ;

---

où: λ ∈ [0, 1] est le taux d'apprentissage ; plus λ est grand, plus la contribution des solutions dans parent (g) est grande.

L: Le nombre total de solution contenant dans le parent (g).

c) Opérateur de mutation guidée GM

L'opérateur GM utilise à la fois les informations statistiques globales stockées sous la forme du vecteur de probabilité p et les informations de localisation de la solution parent pour générer de nouveaux individus. Le pseudo-code de notre opérateur GM est présenté dans l'algorithme 11;

où β ∈ [0, 1] est un paramètre ajustable et DT est un nouveau arbre dominant construit par l'opérateur GM et dont les nœuds sont soit choisi aléatoirement selon le vecteur de probabilité p, soit ils sont copiés à partir de la solution mi de la population courante.

Le choix des nœuds à partir de mi peut aider à améliorer la qualité de la solution vu que celle-ci fait parti des meilleures solutions M, cependant, l'utilisation du vecteur de probabilités permet une meilleure diversification en ajoutant les nœuds les moins utilisés a travers les générations.

Algorithme 11: **Pseudo-code de GM.**

1. Mettre la couleur de tous les nœuds dans V à BLANC ;
2. DT $\leftarrow \varnothing$;
3. **Pour** (nœud v $\in$ V dans un ordre quelconque) **Faire**
4. Générer un nombre aléatoire r1 tel que $0 \leq r1 \leq 1$;
5. **Si** ($r1 < \beta$) **Alors**
6. Générer un nombre aléatoire r2 tel que $0 \leq r2 \leq 1$;
7. **Si** (r2 < p v Et ((v est BLANC) Ou (v est GRIS avec au moins un voisin BLANC))) **Alors**
8. Trouver le chemin le plus court SP entre le nœud v et un nœud u dans DT ;
9. Ajouter toutes les arêtes de SP dans DT ;
10. Colorer en NOIR tous les nœuds dans le chemin SP ;
11. Colorer en GRIS tous les nœuds voisins BLANCs des nœuds sur le chemin SP ;
12. **Fin Si**
13. **Sinon**
14. **Si** (v est un nœud dominant dans m i et v est GRIS avec au moins un voisin BLANC) **Alors**
15. Trouver le chemin le plus court SP entre le nœud v et un nœud u dans DT ;
16. Ajouter toutes les arêtes de SP dans DT ;
17. Colorer en NOIR tous les nœuds dans le chemin SP ;
18. Colorer en GRIS tous les nœuds voisins BLANCs des nœuds sur le chemin SP ;
19. **Fin Si**
20. **Fin Si**
21. **Fin Pour**
22. **Retourner DT ;**

L'opérateur de mutation ne garanti pas la faisabilité de la solution générée par l'opérateur de mutation GM. Pour cette raison, toutes les solutions infaisables générées par un opérateur GM sont traitées un par un par un opérateur de réparation de façon à ce qu'elles deviennent réalisables.

d) Opérateur de réparation:

L'opérateur de réparation s'applique uniquement sur une solution infaisable générée par l'opérateur GM. Car en appliquant ce dernier, peut en résulte des nœuds non couvrants. Pour résoudre ce problème, l'opérateur de correction procède comme suit:

soit Ucn l'ensemble des nœuds non couvrants (blancs) résultants. Tous d'abord, un nœud contenant le nombre le plus élevé de nœuds voisins blancs dans la liste Ucn est sélectionné. Si aucun des nœuds de l'ensemble Ucn n'a des nœuds voisins BLANCS, le nœud ayant l'indice le plus bas est sélectionné à partir de Ucn ; ce nœud est nommé i. Après cela, le chemin le plus court entre le nœud i et l'arbre partiellement construit DT est calculé, et toutes les arêtes de ce dernier sont ajoutées dans DT. De même, les nœuds de ce chemin sont recolorés en NOIR (si ce n'est pas déjà fait) et leurs nœuds voisins en GRIS. Tous ces nœuds NOIR et GRIS sont ensuite supprimés de l'ensemble Ucn . Ce processus itératif est répété jusqu'à ce que Ucn devient vide. Le pseudo-code de l'opérateur de réparation est donné dans l'Algorithme 12 suivant:

Algorithme 12: Pseudo-code de l'opérateur de réparation.

1:Tant que (U cn 6 = ∅) Faire
2:v ← − argmax u ∈ U cn (wd (u) > 0) ;
3:Si (v ≠ ∅) Alors
4:Sélectionner un nœud v avec le plus petit index de U cn ;
5:Fin Si
6:Trouver le chemin le plus court SP entre le nœud v et un nœud u dans DT ;
7:Ajouter toutes les arêtes de SP dans DT ;
8:Colorer en NOIR tous les nœuds dans le chemin SP ;
9:Colorer en GRIS tous les nœuds voisins BLANCs des nœuds sur le chemin SP ;
10:Enlever tous les nœuds NOIRs et GRIS de U cn ;
11:Fin Tant que
12:Retourner DT

Le pseudo-code de l'approche EA / G-MP pour DTP est donné dans l'algorithme 13.

Algorithme 12: EA/G-MP pour DTP.

1. A la génération g ← 0, une population initiale pop(g) qui consiste en N p solutions est générée aléatoirement ;

2. Initialiser le vecteur de probabilités p pour tous les nœuds en utilisant l'Algorithme 9 ;

3. Sélectionner les L meilleures solutions de pop(g) pour former un ensemble parent parent(g), puis mettre à jour le vecteur de probabilités p en utilisant l'Algorithme 10 ;

4.Appliquer l'opérateur de mutation guidée GM une fois sur chacune des M meilleures solutions. Un opérateur de réparation est appliqué sur chaque solution générée, si nécessaire, puis MST, l'opérateur de pruning, MST et l'opérateur de pruning sont appliqués sur chaque solution générée pour améliorer sa fitness. Ajouter toutes les M nouvelles solutions générées avec $N_p - M$ meilleures solutions à pop(g) pour former pop(g+1). Si le critère d'arrêt est satisfait, retourner l'arbre dominant avec le poids minimum trouvé jusqu'à présent ;

5. $g \leftarrow g + 1$;

6. **Si** la meilleure solution de la population ne s'améliore pas après S c générations, **alors** réinitialiser toute pop(g), sauf la meilleure solution, puis aller à l'étape 2 ;

7. Aller à l'étape 3

Conclusion

Dans ce chapitre nous avons présenté le problème de l'arbre dominant, qui est une des meilleures approches utilisé pour la résolution du problème de consommation de l'énergie dans les réseaux de capteurs sans fil. Comme la construction de DT est NP-hard, Shin et Thai ont proposé une approximation de ce dernier en DST pour optimiser le temps de construction de l'arbre dominant, et d'autre chercheurs ont opté à développer des méta-heuristiques pour la même raison. Cependant, on ne peut pas tout avoir dans une méta-heuristique (c'est à dire qu'une méta-heuristique peut donner des meilleurs résultat en terme temps de calcule et d'autre en terme qualité de solution) pourquoi ne pas penser à combiner des méta-heuristiques pour des résultats plus fiables?.