République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia

Faculté des Sciences Exactes

Département Informatique

Mémoire de fin de cycle

Pour l'obtention du diplôme de Master en Administration et Sécurité des Réseaux

Option :

Administration et sécurité des réseaux

Thème

Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil

Présenté par :

- **FERROUDJ Manel**
- **GASMI Elaldja**

Encadré par : **Mr KABYL Kamal**

Examiné par : **Mr SADI Mustapha**

Mr OUZEGGANE Redouane

Bejaia, 2020/2021.

Remerciements

Ce travail est le fruit de la combinaison d'efforts de plusieurs personnes. On remercie tout d'abord le DIEU tout puissant qui, par sa grâce nous a permis d'arriver au bout de nos efforts en nos donnant la santé, la force, le courage , sans Lui rien de tout cela n'aurait pu être et en nos faisant entourer des merveilleuses personnes dont on remercie.

Nous remercions nos chers parents pour leur grand soutien et leur encouragement.

Notre encadrant k.kabyl pour son encadrement sans faille, son soutien moral, sa rigueur au travail, ses multiples conseils, ses orientations et sa disponibilité malgré ses multiples occupations ;

Les membres de juryMr SADI Mustapha et Mr OUZEGGANE Redouane Pour avoir accepté à évaluer notre travail.

Notre cher prof sidali.abdlatif pour son aide, et sa présence.

Tous les enseignants de département informatique de l'université ABDERRAHMANE MIRA, pour leurs enseignements de qualité et leurs conseils qui nous ont permis de poursuivre notre itinéraire académique jusqu'à présent ; Nos camarades, amis et connaissances Tous ceux qui de près ou de loin ont contribué à l'accomplissement de ce travail.

Dédicace

Je dédie ce travail à ma Mère, ma raison d'être DJ.Bara pour la sollicitude qu'elle m'a toujours apportée.

Pour mon père A.Gasmi qui est toujours là pour me soutenir, À mon très cher N.bouhafs, pour sa présence dans ma vie.

À mes frères et sœurs, Mes nièces Mouma, Assil, Aridj, Anfal, Amina, et ma petite Maria.

À mes amies : Farah, Thiziri, Ahlem, Manel, Nesrine , Aïcha, Wissam. À tout personne qui m'a soutenu durant ce parcours.

GASMI Elaldja

Je dédie ce travail :

À mes très chères parents qui sont toujours là pour moi , qui m'ont soutenu et encouragé tous le long de mon parcours, que Dieu les protèges et les prêts bon santé et longue vie.

À mes deux adorables frères Adel et Nassim qui sont toujours à mes cotés, et pour leurs soutien et encouragements et leurs conseils , et à toute ma famille

À tous mes chères enseignants qui m'ont enseigné

À Tous mes chères amis et mes proches sans exception qui m'ont donné le soutien pour réaliser ce travail

FERROUDJ Manel

Table des matières

Table des Figures 5
List des tableaux 6

1 Concepts de base de la théorie des graphes **9**
1.1 Introduction 9
1.2 Définitions et concepts de base 9
1.2.1 Concept de graphe 9
1.2.2 Graphe orienté 9
1.2.3 Graphe non orienté 10
1.3 Extrémité initiale et terminale (successeur et prédécesseur) 10
1.4 Degré d'un sommet 11
1.5 Boucle 11
1.6 Sommets adjacents 11
1.7 Les voisins d'un sommet 11
1.8 Graphe simple 11
1.9 Graphe partiel et sous-graphe 12
1.10 Stable 13
1.11 Clique 13
1.12 Notion d'adjacence entre sommets 14
1.13 Chaînes, Chemins, Cycles et Circuits 14
1.13.1 Chaîne 14
1.13.2 Chemin 14
1.13.3 Cycle 14
1.13.4 Circuit 14
1.14 Connexité dans les graphes 15
1.14.1 Connexité et forte connexité 15
1.14.2 Composantes connexes 15
1.14.3 Forte connexité 15
1.14.4 Composantes fortement connexes 16
1.14.5 Graphe fortement connexe 16
1.15 Quelques types de graphes 17
1.15.1 Graphe complet 17
1.15.2 Graphe biparti 17
1.15.3 Graphe biparti complet 18
1.15.4 Arbres 18
1.15.5 Arborescence 19
1.15.6 Graphe valué : 19

1.16 Représentation matricielle d'un graphe 20
1.16.1 Matrice d'adjacences 20
1.16.2 Listes d'adjacences 21
1.16.3 Matrice d'incidence 22
1.17 Opérations sur les graphes 22
1.17.1 Subdivision de graphe 22
1.17.2 Somme Cartésienne de deux graphes 22
1.17.3 Produit Cartésien de deux graphes 23
1.17.4 Morphisme de graphe 24
1.17.5 Homomorphisme de graphe 24
1.17.6 Isomorphisme de graphe 24
1.18 Implémentation des graphes pondérés 25
1.19 Conclusion 25

2 Les réseaux de capteurs sans fil et leurs applications 26
2.1 Introduction 26
2.2 Les réseaux de capteurs 26
2.2.1 Définition d'un capteur 26
2.2.2 Définition des réseaux de capteurs 27
2.3 Architecture de capteur sans fil 27
2.4 Modèles de communication dans les RCSF 28
2.4.1 Modèle en couche 29
2.4.2 Plan de gestion de l'énergie 30
2.4.3 Plan de gestion de la mobilité 30
2.4.4 Plan de gestion des tâches 31
2.5 Applications des réseaux de capteurs sans fil 31
2.5.1 Application militaire 31
2.5.2 Application médicale 31
2.5.3 Application d'agriculture 32
2.5.4 Application commerciale 32
2.5.5 Application environnementale 33
2.6 Risques liés aux réseaux de capteurs sans fil 33
2.6.1 Manque de sécurité 33
2.6.2 War-driving 33
2.6.3 Risques en matière de sécurité 33
2.7 Caractéristiques des réseaux des capteurs 34
2.7.1 Système d'exploitation TinyOS 34
2.7.2 Propriétés de OS 34
2.8 Energie de capteur sans fils 35
2.8.1 Problèmatique 35
2.9 Ordonnancement de réseaux de capteur 36
2.10 Gestion d'ordonnancement 36
2.11 Déploiement de capteur 37
2.12 Conclusion 37

3 Quelques Techniques d'optimisations dans les réseaux de Capteurs **38**
3.1 Introduction . . . 38
3.2 Modélisation de réseaux de capteur . . . 38
3.3 Solution proposée . . . 39
3.4 Hypothèses . . . 39
3.5 Problèmes d'optimisations dans les réseaux . . . 39
3.5.1 Problème de plus court chemin . . . 39
3.6 Détail de la solution proposée . . . 41
3.6.1 Objectif de notre modélisation . . . 41
3.6.2 Problème d'arbre couvrant de poids minimum [4] . . . 43
3.6.3 Routage dans le RCSF . . . 47
3.7 Conclusion . . . 48

4 Application **49**
4.1 Introduction . . . 49
4.2 Introduction à l'environnement de travail (Eclipse) . . . 49
4.3 Exécutez le code . . . 51
4.4 Présentation de Langage utilisé « JAVA » . . . 51
4.5 Quelques capteurs de code source . . . 52
4.6 Cas d'optimisation d'énergie : . . . 53
4.7 Conclusion . . . 57

Table des figures

1.1 Graphe orienté . . . 10
1.2 Graphe non orienté . . . 10
1.3 Graphe simple . . . 11
1.4 Graphe G à 6 sommets et 6 arrêtes . . . 12
1.5 Graphe partiel de G . . . 12
1.6 Sous-graphe de G . . . 13
1.7 Graphe à deux stables . . . 13
1.8 Clique à 3 sommets . . . 14
1.9 Graphe avec 2 composantes connexes . . . 15
1.10 Relation de la forte connexité . . . 16
1.11 Graphe à deux Composante fortement connexe . . . 16
1.12 K_5 . . . 17
1.13 Graphe biparti . . . 17
1.14 Graphe $K_{3,2}$ . . . 18
1.15 Arbre . . . 18
1.16 Arborescence de racine a . . . 19
1.17 Graphe valué . . . 19
1.18 Graphe pondéré . . . 20
1.19 Graphe G et Sa matrice d'adjacence . . . 21
1.20 Graphe G et sa liste d'adjacence . . . 21
1.21 Représentation d'un graphe par la matrice d'incidence . . . 22
1.22 Somme cartésienne de deux graphes . . . 23
1.23 Produit cartésien de deux graphes . . . 23
1.24 Homomorphisme de graphe . . . 24
1.25 Isomorphisme de graphe . . . 25

2.1 Un capteur sans fil . . . 26
2.2 Les différents composants d'un capteur sans fil . . . 27
2.3 L'architecture d'un réseau sans fil . . . 29
2.4 Modèle en couche pour la communication dans les RCSF . . . 29
2.5 Application militaire . . . 31
2.6 Application médicale . . . 32
2.7 Réseau de capteur pour l'agriculture. . . . 32
2.8 Surveillance de l'environnement . . . 33
2.9 Système d'exploitation TinyOS . . . 34
2.10 Exemple d'un circuit de capteur . . . 36

2.11 Gestion d'ordonnancement 37

3.1 Un réseau du capteur 38
3.2 Réseau de capteur d'ordre 6 41
3.3 Exemple d'application 44
3.4 Arbres couvrants de poids minimum 45
3.5 L'application de l'algorithme de prim 47

4.1 « Eclipse » 49

Liste des tableaux

3.1 Initialisaion 42
3.2 Itération 1 42
3.3 Itération 5 43

Introduction générale

Depuis leur création, les réseaux sans fil ne cessent de connaitre un succès croissant au sein des communautés scientifiques et industrielles. Grâce à leurs divers avantages, cette technologie a pu s'instaurer comme acteur incontournable dans les architectures réseaux de communication actuels. En effet, grâce à leur support de transmission qui est le média hertzien, ces réseaux présentent plusieurs avantages qui sont entre autres le coût réduit des équipements, la facilité d'installation et l'ubiquité de l'information. Au cours de leurs évolutions, les réseaux sans fil ont donné naissance à diverses architectures telles que les réseaux cellulaires, les réseaux locaux sans fil, les réseaux Ad hoc, etc. [1]

Durant ces deux dernières décennies, une nouvelle architecture appelée Réseau de Capteur Sans Fil (RCSF) a vu le jour. Ce type de réseau est né de la fusion entre les systèmes embarqués et les communications sans fil. Un RCSF ("WSN : Wireless Sensor Network" en Anglais) est un réseau Ad hoc composé d'un grand nombre de nœuds qui sont des micro-capteurs communicant via des liaisons sans fil par des ondes radioélectriques de façon autonome. Ces nœuds encore appelés capteurs sont capables de récolter plusieurs paramètres sur l'environnement qui les entoure, appelé généralement zone de captage (ou zone de surveillance). Ensuite, ils doivent si nécessaire traiter les données capturées et les transmettre à un ou plusieurs nœuds de collecte appelés station de base, collecteurs, centres de traitements (ou "sink" en Anglais).[1]

Un réseau de capteur sans fil consiste en un nombre important de nœuds capteurs intelligents de petites tailles, à faible coût de puissance limitée et multifonctionnels. Ces nœuds sont déployés dans un espace d'intérêt et sont intégrés pour collaborer à travers un réseau sans fil. Ils sont alimentés par une source d'énergie (batterie) de capacité limitée. Ils sont capables de capter (ou collecter) des grandeurs physiques de l'environnement telles que la température, vitesse du vent, humidité relative, ...etc. Ils sont également capables de détecter les évènements du monde réel tels que les feux de forêt, traiter les données et communiquer entre eux en employant typiquement des canaux sans fil en fréquence radio (RF : radio Frequency) pour faire aboutir les informations collectées à un point de collecte appelé puits ou Sink. Ces informations sont ensuite transmises via un réseau de transport (Internet, réseau cellulaire type GSM, RNIS vers un centre de traitement final. [2]

Nous nous intéressons dans notre travail à la définition ainsi que "la programmation de quelques méthodes d'optimisation dans les graphes ", dont le but est de résoudre un problème réel en utilisant la théorie des graphes et plus particulièrement l'optimisation. Et pour cela, on a réparti notre mémoire en quatre chapitres, une conclusion et une introduction :

Dans le premier chapitre, nous avons introduit la notion des graphes, le deuxième chapitre quant à lui présente les réseaux de capteurs, en particulier, l'architecture d'un réseau de capteur, ses caractéristiques ainsi que ses applications ont été présentées. Au troisième et dernier chapitre, nous aborderons quelques situations réelles qui résument le but de notre travail, à savoir l'optimisation de la consommation de l'énergie dans le réseau, qui vont être résolus par la suite en utilisant

les algorithmes de Djikstra et Kruskal, de la consommation de l'énergie, ce qui conduira à élaborer des résultats pour chaque problème, obtenus à travers l'exécution de l'algorithme de Dijkstra et Kruskal implémenté sous le logiciel Eclipse IDE pour JAVA. Enfin, le mémoire s'achève par une conclusion, dans laquelle notre contribution a été résumée et les perspectives de notre travail ont été soulignés.

Chapitre 1

Concepts de base de la théorie des graphes

1.1 Introduction

Un graphe est un schéma constitué de sommets, dont certains sont reliés par des liens La notion de graphe est une structure combinatoire permettant de représenter de nombreuses situations rencontrées dans des applications faisant intervenir des mathématiques discrètes et nécessitant une solution informatique. Circuits électriques, réseaux de transport (ferrés, routiers, aériens), réseaux d'ordinateurs, ordonnancement d'un ensemble de tâches sont les principaux domaines d'application où la structure de graphe intervient.

1.2 Définitions et concepts de base

1.2.1 Concept de graphe

Un graphe est composé de points appelés sommets ou nœuds. Certains de ces points sont reliés entre eux par des arêtes. Ces arêtes sont souvent représentées par des segments mais elles peuvent très bien être courbes. De façon plus formelle, un graphe est défini par un couple G = (S, A) tel que S est un ensemble fini de sommets, A est un ensemble de couples de sommets $(s_i; s_j)\epsilon\ S^2$.[25]

1.2.2 Graphe orienté

Un graphe oriente est un couple (X ;U), ou X est l'ensemble des sommets du graphe et U, l'ensemble de ses arcs. X et U sont finis. L'arc est une relation entre deux sommets, dotée d'une orientation : Si u = (x ;y) est un arc de U alors avec x ; $y \in X$ la relation est orientée de x vers y. Le graphe G est note G =(X ;U).[25]

La figure 1.1 montre un graphe orienté à 5 sommets et 9 arcs.

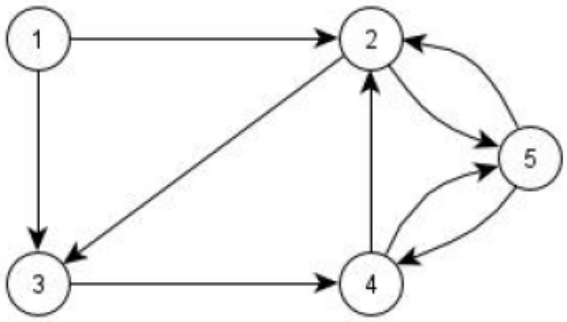


FIGURE 1.1 – Graphe orienté

1.2.3 Graphe non orienté

Un graphe non orienté G est un couple (X ;E), ou X est un ensemble dont les éléments sont appelés sommets et E un sous-ensemble de parties de X contenant chacune au plus 2 éléments et dont les éléments sont appelés arêtes.

La figure 1.2 illustre un graphe non orienté a 5 sommets et 7 arêtes.

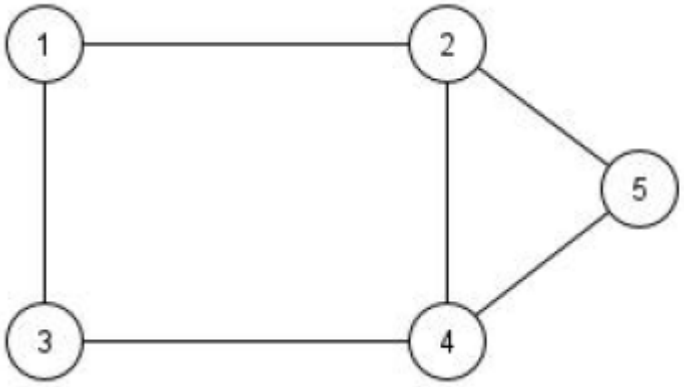


FIGURE 1.2 – Graphe non orienté

1.3 Extrémité initiale et terminale (successeur et prédécesseur)

Soit un arc (i, j) ; i est dit extrémité initiale de (i, j) et j extrémités terminale de (i, j). On dit aussi que j est un successeur de i, et i un prédécesseur de j. L'arc (i, j) est dit incident vers l'extérieur en i et vers l'intérieur en j.

Définition : Un arc de graphe orienté G de la forme (i, i) est appelé une boucle. Pour un arc $u = (i, j)$, le point i est son extrémité initiale, et le point j son extrémité terminale.

On dit que j est un successeur de i s'il existe un arc ayant son extrémité initiale en i et son extrémité terminale en j. l'ensemble des successeurs de i se note $\Gamma_G^+(i)$

De même, on dit j est un prédécesseur de i s'il existe un arc de la forme (j, i). L'ensemble des prédécesseurs de i se note $\Gamma_G^-(i)$

L'ensemble des sommets voisins de i se note $\Gamma_G(i) = \Gamma_G^+(i) \cup \Gamma_G^-(i$ On note $\Gamma^+(i)$ l'ensemble des successeurs de i, $\Gamma^-(i)$ l'ensemble des prédécesseurs de i.

1.4 Degré d'un sommet

Dans un graphe non-orienté, le degré d'un sommet est le nombre d'arêtes incidentes à ce sommet (une boucle comptant pour 2). Dans le cas d'un graphe simple, on aura d(s) = |Adj(s)|.

Dans un graphe orienté, le demi-degré extérieur ou demi-degré sortant d'un sommet s, noté $d^+(s)$, est le nombre d'arcs partant de s, i.e. de la forme $(s; v)$ avec $v \in S$. Dans le cas d'un graphe, on aura $d^+(s) = |\text{Succ(s)}|$. tel que : S est un ensemble fini de sommets,[17]

1.5 Boucle

Une boucle est une arête dont son extrémité initiale et son extrémité terminal coïncident.

1.6 Sommets adjacents

On dit que deux sommets, x et y, sont adjacent si et seulement s'il existe une arrête le reliant.

1.7 Les voisins d'un sommet

Les voisins d'un sommet x d'un graphe G sont les sommets qui admettent une arrête avec x,

1.8 Graphe simple

Un graph simple est un graphe sans boucles ni arcs (arêtes) multiple.[25]
La figure 1.3 montre un graphe simple.

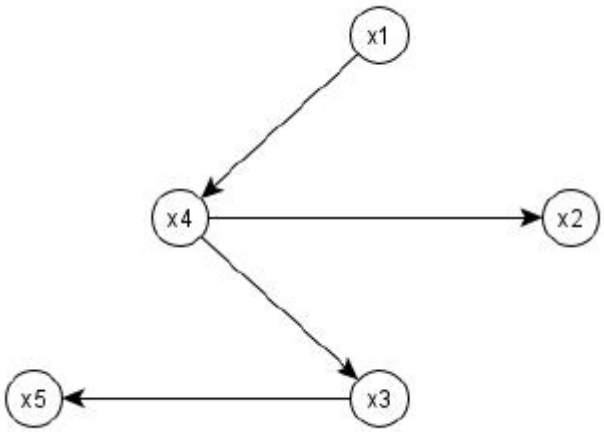


FIGURE 1.3 – Graphe simple

1.9 Graphe partiel et sous-graphe

Soit G = (V, E) un graphe . Le graphe $G\prime$ = (V,$E\prime$) est un graphe partiel de G, si $E\prime$ est inclus dans E . Autrement dit, on obtient $G\prime$ en enlevant une ou plusieurs arêtes au graphe G. Pour un sous-ensemble de sommets A inclus dans V , le sous-graphe de G induit par A est le graphe G_A =(A,E(A)) dont l'ensemble des sommets est A et l'ensemble des arêtes E(A) est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G_A en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.[22]

La figure 1.4 montre un graphe G à 6 sommets et 6 arêtes

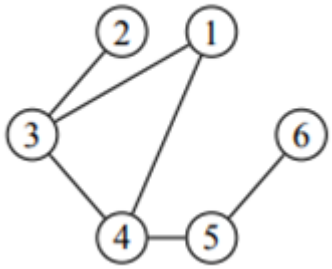


FIGURE 1.4 – Graphe G à 6 sommets et 6 arrêtes

La figure 1.5 montre un graphe partiel de G

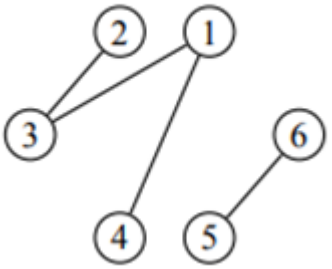


FIGURE 1.5 – Graphe partiel de G

La figure 1.6 montre un Sous-graphe de G engendre par $A = \{1, 2, 3, 5, 6\}$

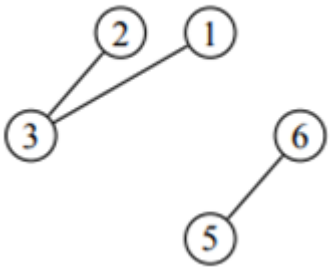


FIGURE 1.6 – Sous-graphe de G

1.10 Stable

Un stable est un ensemble de sommets de G deux à deux non-adjacents. Le graphe de la Figure 1.7 admet deux stables i : $S1 = \{1; 3; 6; 8\}$ et $S2 = \{2; 4; 5; 7\}$

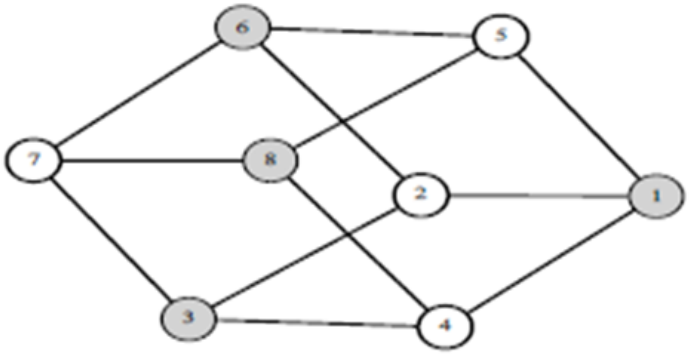


FIGURE 1.7 – Graphe à deux stables

1.11 Clique

Une clique dans un graphe est un ensemble de sommets deux à deux adjacents. W(G) : la taille de la plus grande clique dans un graphe G.

La Figure 1.8 représente clique à 3 sommets .

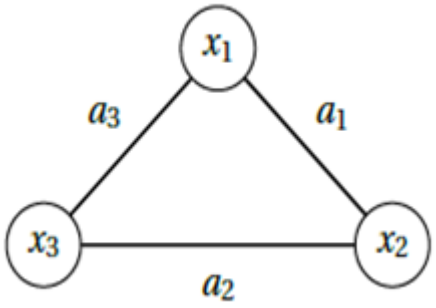


FIGURE 1.8 – Clique à 3 sommets

1.12 Notion d'adjacence entre sommets

Dans un graphe non orienté, un sommet i est dit adjacent à un autre sommet j s'il existe une arête entre i et j .

1.13 Chaînes, Chemins, Cycles et Circuits

1.13.1 Chaîne

Une chaîne joignant deux sommets x_0 et x_k dans un graphe G est une suite de sommets relié par des arêtes tel que deux sommets successifs ont une arête commune. On la note $(x_0; x_1; :::; x_k)$.

– Le premier et le dernier sommet sont appelés extrémités de la chaîne.

– La longueur de la chaîne est égale au nombre d'arêtes qui la composent.

– Une chaîne qui passe une seule fois par ses arêtes est dite chaine simple et celle qui passe une fois par chaque sommet est dite chaîne élémentaire.

1.13.2 Chemin

Un chemin de x_0 à x_k dans un graphe est une suite de sommets tel que $\forall i \in \{_0, ...,_k\}$

x_i est relié à x_{i+1} par une arcs (x_i, x_{i+1}).

– Un chemin dit simple s'il passe une et une seul fois par ses arcs et il dit chemin élémentaire s'il passe une et une seul fois par ses sommets.

– On appelle distance de sommet x_i à un autre sommet x_j , la longueur du plus court chemin de x_i à x_j.

– Le diamètre de graphe G est alors la plus grand distance entre deux sommet de G.

1.13.3 Cycle

un cycle est une chaîne simple qui se ferme sur elle-même.

1.13.4 Circuit

un circuit est un chemin simple qui se ferme sur lui-même.

1.14 Connexité dans les graphes

1.14.1 Connexité et forte connexité

Un graphe G est dit connexe s'il y a une chaîne entre n'importe quelle paire de sommets distincts de G.

On dit aussi un graphe G est connexe si ses sommet deux à deux vérifient la relation suivante : Il existe une chaîne entre un paire quelconque de sommet (x,y) ou bien x=y.[25]

1.14.2 Composantes connexes

On appelle composante connexe un ensemble de sommets qui ont deux a deux la relation de connexité, de plus tout sommet en dehors de la composante n'a pas de relation de connexité avec cette composante.

La Figure 1.9 montre un graphe avec 2 composantes connexes.

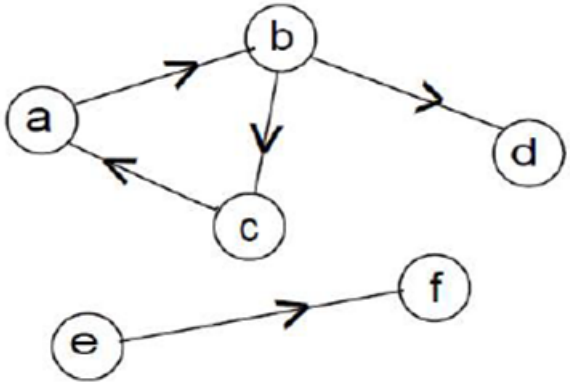


FIGURE 1.9 – Graphe avec 2 composantes connexes

Les sommets a, b, c, d ont deux à deux la relation de connexité, donc l'ensemble a,b,c,d forme ainsi la première composante connexe, on la note C1.

- L'ensemble $\{e, f\}$ forme la deuxième composante connexe, on la note C2.

On constate que les sommets de C1 n'ont pas de relation de connexité avec les sommets de C2

1.14.3 Forte connexité

On définit la forte connexité dans un graphe par une relation entre deux sommets de la manière suivante : deux sommets x et y ont une relation de forte connexité, il existe un chemin de x vers y et un chemin de y vers x ou bien x = y.[25] On considère la figure suivante :

-On a un chemin reliant le sommet x_1 et x_3 et un chemin reliant le sommet x_3 au sommet x_1 alors x_1 et x_3 ont une relation de forte connexité.

-On a un chemin reliant le sommet x_4 à x_3, mais on n'a pas de chemin reliant le sommet x_3 au sommet x_4 alors x_4 et x_3 n'ont pas de relation de forte connexité.

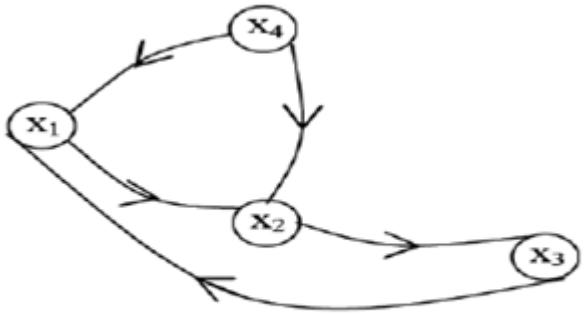


FIGURE 1.10 – Relation de la forte connexité

1.14.4 Composantes fortement connexes

On appelle composante fortement connexe un ensemble de sommets, qui ont deux a deux la relation de forte connexité, de plus tout sommet en dehors de la composante n'a pas de relation de forte connexité avec aucun élément de cette composante. On considère la figure suivante :

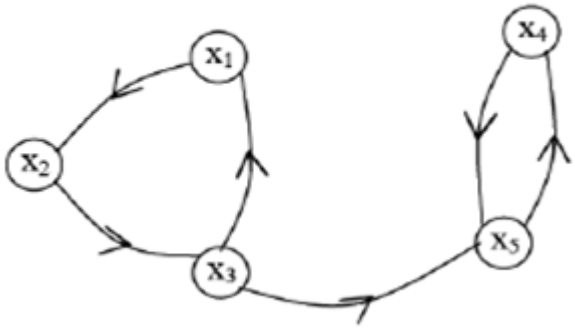


FIGURE 1.11 – Graphe à deux Composante fortement connexe

-Les sommets x_1 , x_2 , x_3 ont deux à deux la relation de forte connexité, donc l'ensemble $\{x_1, x_2, x_3\}$ forme ainsi la première e composante fortement connexe, on la note C_1.

- L'ensemble $\{x4, x5\}$ forme la deuxième composante fortement connexe, on la note C_2. On constate que les sommets de C_1 n'ont pas de relation de forte connexité avec les sommets de $C2$.

1.14.5 Graphe fortement connexe

Un graphe G est dit fortement connexe si tous ses sommets ont deux à deux la relation de forte connexité, autrement dit si G contient une seule composante fortement connexe.

1. Le graphe G précédant contient deux composantes fortement connexes $C_1 = \{x_1, x_2, x_3\}$ et $C_2 = \{x_4, x_5\}$, le graphe n'est pas fortement connexe.

2. Si on ajoute l'arc (x_5, x_3) au graphe précédant, le graphe obtenu contient une seule Composante fortement connexe $C = \{x_1, x_2, x_3, x_4, x_5\}$, alors il est fortement connexe.

1.15 Quelques types de graphes

1.15.1 Graphe complet

Un graphe G(V,E) est dit complet si chaque sommet du graphe est relié directement à tous les autres sommets[22].

Un graphe complet simple à n sommets est notée K_n.

La Figure 1.12 montre le graphe K_5

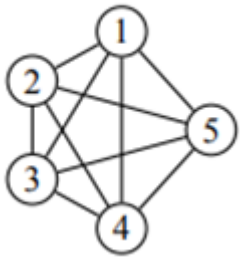


FIGURE 1.12 – K_5

1.15.2 Graphe biparti

Un graphe G(V,E) est dit biparti si l'ensemble de ses sommets V est partitionné en deux classes X et Y tel que deux sommets de la même classe ne sont jamais reliés. Les arêtes de ce graphe sont celle qui relient un sommet de X a un sommet de Y.[25]

La Figure 1.13 représente un graphe biparti .

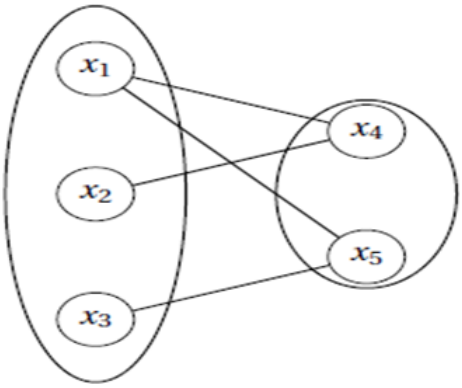


FIGURE 1.13 – Graphe biparti

1.15.3 Graphe biparti complet

Un graphe G($X \cup V$,E) biparti est dit complet (ou encore est appelé une biclique) si chaque sommet de X est reliè à chaque sommet de Y.[25]

Un graphe biparti complet simple à n×m sommets est noté par $K_{n,m}$.

La Figure 1.14 montre le graphe $K_{3,2}$

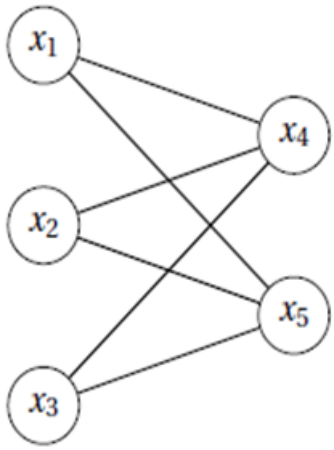


FIGURE 1.14 – Graphe $K_{3,2}$

1.15.4 Arbres

Un arbre est un graphe non orienté G qui vérifie une des conditions équivalentes suivantes [17] :

- G est connexe et acyclique
- G est sans cycle et possède $n-1$ arêtes
- G est connexe et admet $n-1$ arêtes
- G est sans cycle, et en ajoutant une arête, on crée un et un seul cycle élémentaire,
- G est connexe, et en supprimant une arête quelconque, il n'est plus connexe,
- Il existe une chaîne et une seule entre 2 sommets quelconques de G.

La Figure 1.15 montre un arbre

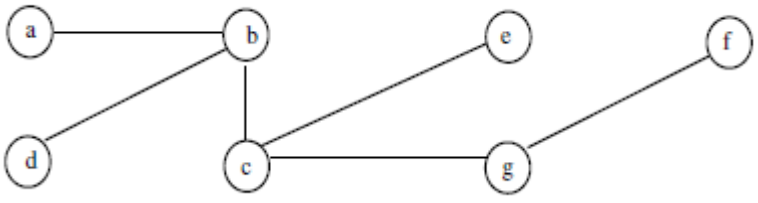


FIGURE 1.15 – Arbre

1.15.5 Arborescence

Une arborescence est un graphe orienté sans circuit admettant une racine $s_0 \in S$ telle que, pour tout autre sommet $s_i \in S$, il existe un chemin unique allant de s_0 vers s_i. Si l'arborescence comporte n sommets, alors elle comporte exactement $n - 1$ arcs.[17]

La Figure 1.16 montre une arborescence de racine a

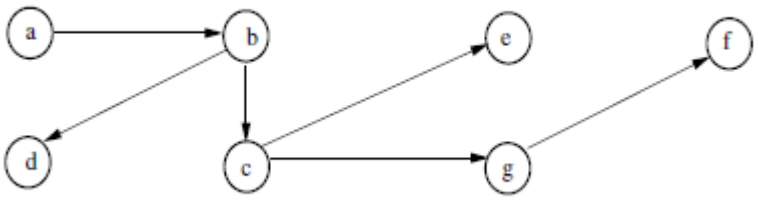


FIGURE 1.16 – Arborescence de racine a

1.15.6 Graphe valué :

Un graphe valué est un graphe pour lequel chaque arête est associée à un nombre réel appelé poids. Si ce nombre est positif, on parle alors de graphe pondéré.

La Figure 1.17 montre un graphe valué :

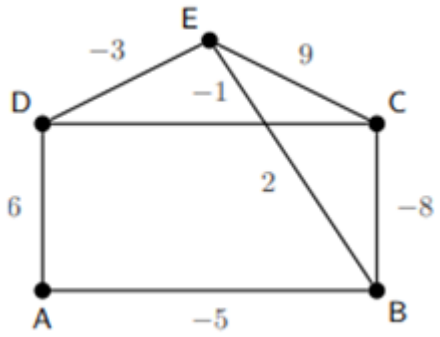


FIGURE 1.17 – Graphe valué

La Figure 1.18 montre un graphe pondéré :

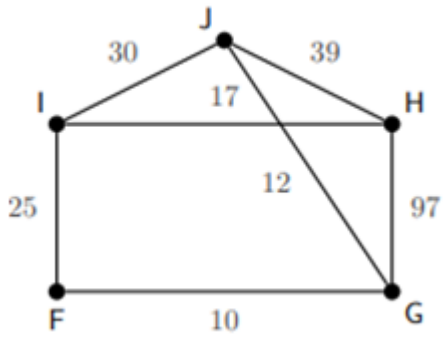


FIGURE 1.18 – Graphe pondéré

L'un des problèmes classiques des graphes pondérés est celui de recherche d'un trajet routier le plus court (en termes de temps ou de kilomètres).

1.16 Représentation matricielle d'un graphe

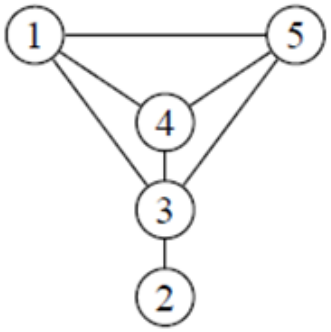
Il existe deux façons classiques de représenter un graphe en machine : par une matrice d'adjacence , par un ensemble de listes d'adjacence, où bien par la matrice d'incidence.

1.16.1 Matrice d'adjacences

On peut représenter un graphe simple par une matrice d'adjacences. Une matrice (n×m) est un tableau de n lignes et m colonnes. (i, j) désigne l'intersection de la ligne i et de la colonne j. Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe. Un « 1 » à la position (i, j) signifie que le sommet i est adjacent au sommet j .[22]

Cette matrice est donné par

$$M_{ij} = \begin{cases} 1 \ si \ i \ et \ j \ sont \ \ adjacent \\ 0 \ si \ non \end{cases}$$



$$M = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

FIGURE 1.19 – Graphe G et Sa matrice d'adjacence

Cette matrice a plusieurs caractéristiques :

1. Elle est carrée : il y a autant de lignes que de colonnes.

2. Il n'y a que des zéros sur la diagonale allant du coin supérieur gauche au coin inférieur droit. Un « 1 » sur la diagonale indiquerait une boucle.

3. Elle est symétrique : mi j =mji . On peut dire que la diagonale est un axe de symétrie.

4. Une fois que l'on fixe l'ordre des sommets, il existe une matrice d'adjacences unique pour chaque graphe. Celle-ci n'est la matrice d'adjacences d'aucun autre graphe.

1.16.2 Listes d'adjacences

On peut aussi représenter un graphe simple en donnant pour chacun de ses sommets la liste des sommets auxquels il est adjacent. Ce sont les listes d'adjacences.[22]

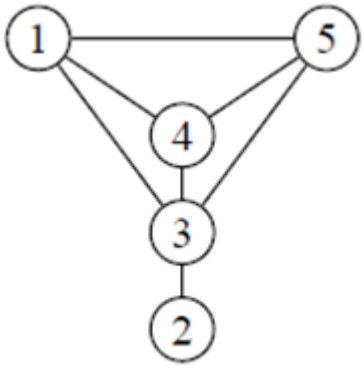


FIGURE 1.20 – Graphe G et sa liste d'adjacence

1 : 3, 4, 5
2 : 3
3 : 1, 2, 4, 5
4 : 1, 3, 5
5 : 1, 3, 4

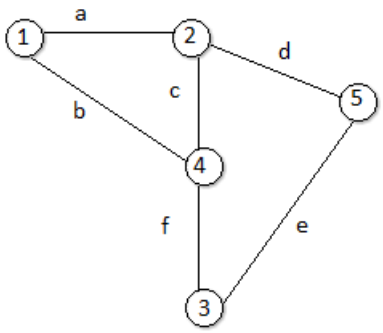
1.16.3 Matrice d'incidence

Un graphe non orientée à n sommets numérotés et p arcs numérotés peut être représenté par une matrice (n ; p)d'entiers [] :

$\forall i \in \{1, ..., n\}$, $j \in \{1, ..., p\}$

$$M_{ij} = \begin{cases} 1 & si\ l'extremite\ initial\ de\ l'arc\ j\ et\ le\ sommet\ i \\ -1 & si\ l'extremite\ terminal\ de\ l'arc\ j\ et\ le\ sommet\ i \\ 0 & sinon \end{cases}$$

La Figure 1.21 montre une matrice d'incidence d'un graphe non orienté .



	a	b	c	d	e	f
1	1	1	0	0	0	0
2	1	0	1	1	0	0
3	0	1	0	0	1	1
4	0	0	1	0	0	1
5	0	0	0	1	1	0

FIGURE 1.21 – Représentation d'un graphe par la matrice d'incidence

1.17 Opérations sur les graphes

1.17.1 Subdivision de graphe

En remplaçant les arêtes d'un graphe G =(X,A) par des chaines disjointes intérieurement , on obtient une subdivision de G. Une subdivision d'une arête UV s'obtient en remplaçant UV par un chemin $\{Ux_1, x_1, x_2, ..., x_k, x_k, x_{kv}\}$

1.17.2 Somme Cartésienne de deux graphes

On appelle somme cartésienne de deux graphes G = (X(G),A(G)) et H = (X(H),A(H)), notée G H, le graphe dont l'ensemble des sommets est le produit cartésien X(G)*X(H) et ou deux sommets (x, $x\prime$) et (y, $y\prime$) sont adjacents si et seulement si l'une des propriétés suivantes est vérifiée [27] :

-x = y et $x\prime y\prime \in A(H)$

ou

-x $y \in A(G)$ et $x\prime$ $y\prime$.

La Figure 1.22 montre la somme cartésienne de deux graphes :

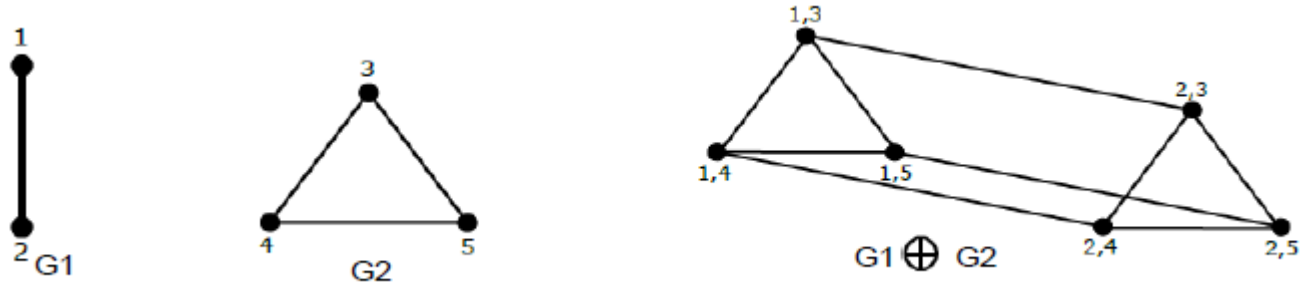


FIGURE 1.22 – Somme cartésienne de deux graphes

On note que le nombre de sommets dans G * H est |X(G) |.|X(H)|, et que le nombre d'arêtes est :

|X(G) |.|A(H)|+|X(H) |.|A(G)|.

1.17.3 Produit Cartésien de deux graphes

Le produit cartésien de deux graphes G= (X(G),A(G)) et H= (X(H),A(H)) est le graphe noté G*H où X(G*H) =X(G)*X(H). Deux sommets (x ;y) et ($x\prime$; $y\prime$) sont adjacents si et seulement si x$x\prime \in A(G)$ et y $y\prime \in A(H)$.[27]

La Figure 1.23 montre le produit cartésien de deux graphes.

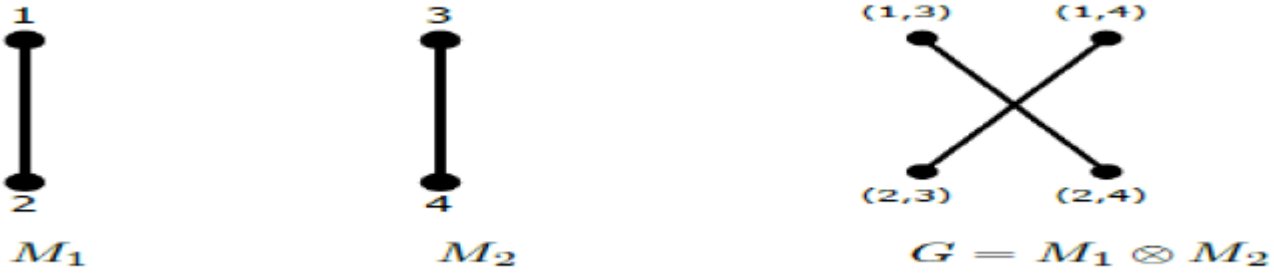


FIGURE 1.23 – Produit cartésien de deux graphes

1.17.4 Morphisme de graphe

Graphes non orientées

Soit G = (X,A) et H = (T,B) deux graphes non orientés. On dit qu'une Application ϕ de X dans T réalise un morphisme du graphe G vers le graphe H si elle vérifie la propriété suivante :

$(y \in A)$–> $\phi\ (y) \in B$

Si ϕ est une application injective, on dit que le graphe G s'injecte dans le graphe H. On dit aussi parfois que le graphe H contient le graphe G.

1.17.5 Homomorphisme de graphe

Soient G = (X, A) et $G\prime$ = ($x\prime$,$A\prime$) deux graphes. Une application f : X –>$x\prime$ tel que : est un homomorphisme de G dans $G\prime$ Si : $(x; y) \in A)(f(x), f(y)) \in A\prime$

On considère la Figure 1.24 , avec les graphes G et G‘ on voit facilement qu'on a un homomorphisme de G dans $G\prime$ mais pas de $G\prime$ dans G.

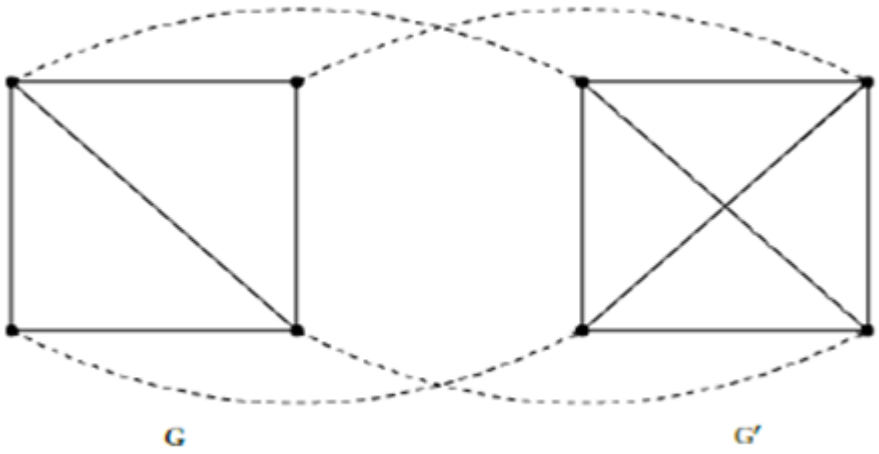


FIGURE 1.24 – Homomorphisme de graphe

1.17.6 Isomorphisme de graphe

Soient G= (X,A) et $G\prime$ = (X,$A\prime$) deux graphes. S'il existe une relation univoque m :X–>$x\prime$
Tel que $(x1, x2) \in A <=> (m(x1), m(x2)) \in A\prime$, alors G et $G\prime$ sont isomorphes.[26]
La Figure 1.25 montre deux graphes isomorphes.

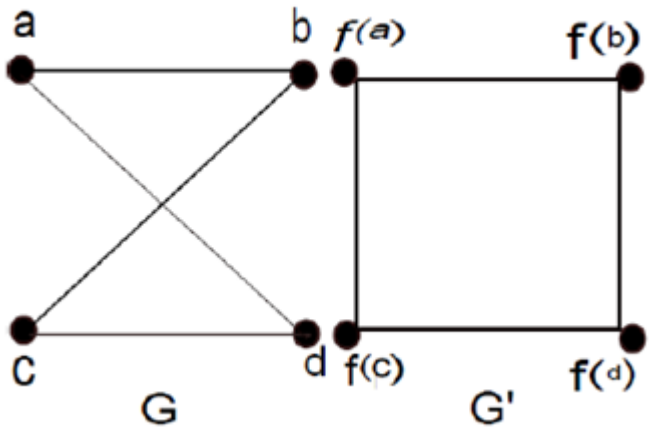


FIGURE 1.25 – Isomorphisme de graphe

1.18 Implémentation des graphes pondérés

Il n'y a pas grand-chose à modifier aux classes déjà implémentées pour prendre en compte les poids : si on utilise une matrice d'adjacence, n'importe quelle valeur de la matrice peut servir de poids et il n'y a donc rien à modifier ; si l'on utilise une liste ou un dictionnaire d'adjacence, on utilise un couple (v, poids) pour décrire l'arête uv d'un poids donné. On supposera donc l'existence d'une classe Graphe Pondéré, disposant des méthodes déjà couvertes dans le cas des graphes non pondérés et qui devront être légèrement modifiées [19] :

— Ajouter - arête(u, v, poids) prend maintenant en paramètre le poids de l'arête u, v ;

— Ajouter - arêtes(séquence) suppose maintenant qu'on lui fournit des triplets plutôt que des couples ;

— Sous-graphe-induit(sequence) renvoie maintenant un graphe pondéré. On supposera également l'éxistence des nouvelles méthodes suivantes :

— Arêtes-incidentes(sommet), qui renvoie l'ensemble des arêtes incidentes à un sommet donné sous la forme de triplets ;

— Poids-arête(u, v), qui renvoie le poids de l'arête u, v si elle existe, NUL sinon.

1.19 Conclusion

Dans ce premier chapitre nous avons fait un rappel sur les concepts de base de la théorie des graphes en présentant les graphes non orientés et les graphes simples et partiels et sous graphe ainsi que les chaine et cycle avec une notion de connexité .Nous avons donné une représentation graphique et nous avons cité d'autres types de graphe. Ensuite nous avons défini les deux matrice d'adjacence et d'incidence pour terminer avec une implémentation.

Chapitre 2

Les réseaux de capteurs sans fil et leurs applications

2.1 Introduction

Depuis quelque années, le marché des réseaux et des applications sans fil s'est considérablement dévloppé.De ce constat, une nouvelle branche s'est créée pour offrir des solutions économiquement intéressantes pour la surveillance à distance et le traitement des données dans les environnements complexes et distribués : les réseaux de capteurs sans fil ([1]

RCSF ou WSN : Wireless Sensor Networks).

2.2 Les réseaux de capteurs

2.2.1 Définition d'un capteur

Les capteurs sont des dispositifs de taille extrêmement réduite avec des ressources très limitées, autonomes, capable de traiter des informations et de les transmettre, via les ondes Radio, à une autre entité (capteur, unité de traitement. . .) sur une distance limitée à quelques mètres. la figure 2.1 montre un capteur sans fil.[3]

FIGURE 2.1 – Un capteur sans fil

1. RCSF :les réseaux de capteurs sans fil

2.2.2 Définition des réseaux de capteurs

Wireless Sensor Networks (WSN) sont considérés comme un type spécial de réseaux ad hoc. Les nœuds de ce type de réseaux consistent en un grand nombre de micro-capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène capté.[1]

2.3 Architecture de capteur sans fil

Un nœud capteur contient quatre unités de base : l'unité de captage, l'unité de traitement, l'unité de transmission, et l'unité de contrôle d'énergie. Il peut contenir également, suivant son domaine d'application, des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire). On peut même trouver des micro-capteurs, un peu plus volumineux, dotés d'un système mobilisateur chargé de déplacer le micro-capteur en cas de nécessité.

la figure 2.2 montre les différents composants d'un capteur sans fil.[3]

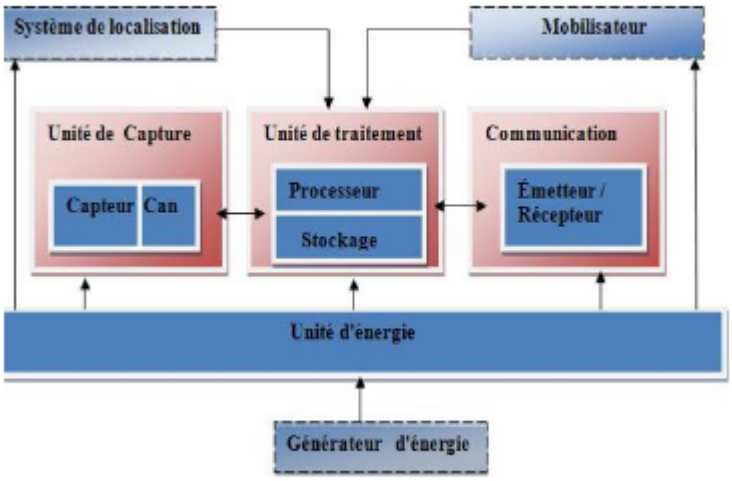


FIGURE 2.2 – Les différents composants d'un capteur sans fil

chaque groupe de composants de la figure 2.2 possède son propre rôle :

- **Unité de traitement**

L'unité de traitement est généralement composée d'une mote, d'un processeur, d'une mémoire RAM et Flash. On appelle généralement mote la carte physique utilisant le système d'exploitation pour fonctionner. Celle-ci a pour cœur le bloc constitué du processeur et des mémoires RAM et flash. Cet ensemble est la base 10 du calcul binaire et du stockage, temporaire pour les données et définitif pour le système d'exploitation. Cette unité est chargée d'exécuter les protocoles de

communications qui permettent de faire collaborer le nœud avec les autres entités du réseau. Elle peut aussi analyser les données captées pour alléger la tâche du nœud puits.

- **Unité de transmission**

Radio et antenne : Les équipements étudiés sont donc généralement équipés d'une radio ainsi que d'une antenne. Cette unité est responsable d'effectuer toutes les émissions et réceptions des données sur un medium sans fil. Elle peut être de type optique (comme dans les nœuds Smart Dust), ou de type radiofréquence. Les communications de type optique sont robustes vis-à-vis des interférences électriques. Néanmoins, elles présentent l'inconvénient d'exiger une ligne de vue permanente entre les entités communicantes. Par conséquent, elles ne peuvent pas établir de liaisons à travers des obstacles.

- **Unités de captage**

On retrouve donc des équipements de différents types de détecteur et d'autre entrée. Le capteur est généralement composé de deux sous-unités : le récepteur (reconnaissant l'analyse) et le transducteur (convertissant le signal du récepteur en signal électrique). Le capteur est responsable de fournir des signaux analogiques, basés sur le phénomène observé, au convertisseur Analogique/Numérique. Ce dernier transforme ces signaux en un signal numérique compréhensible par l'unité de traitement.

- **Unités de control d'énergie**

Batterie : Un micro-capteur est muni d'une ressource énergétique (généralement une batterie de type AAA) pour alimenter tous ses composants. Cependant, en conséquence de sa taille réduite, la ressource énergétique dont il dispose est limitée et généralement irremplaçable. Cette unité peut aussi gérer des systèmes de rechargement d'énergie à partir de l'environnement observé telles que les cellules solaires, afin d'étendre la durée de vie totale du réseau.

2.4 Modèles de communication dans les RCSF

Pour la gestion de la communication dans les RCSF, c'est l'approche qui gère celle dans les réseaux conventionnels qui a été adoptée. Cette approche est basée sur une subdivision de la communication en couches dans laquelle chacune d'elles a un rôle particulier à jouer. Le fonctionnement de cette pile de communication ainsi que le rôle de chaque couche sont détaillés dans la section ci-dessous.

la figure 2.3 montre l'architecture d'un réseau sans fil.[1]

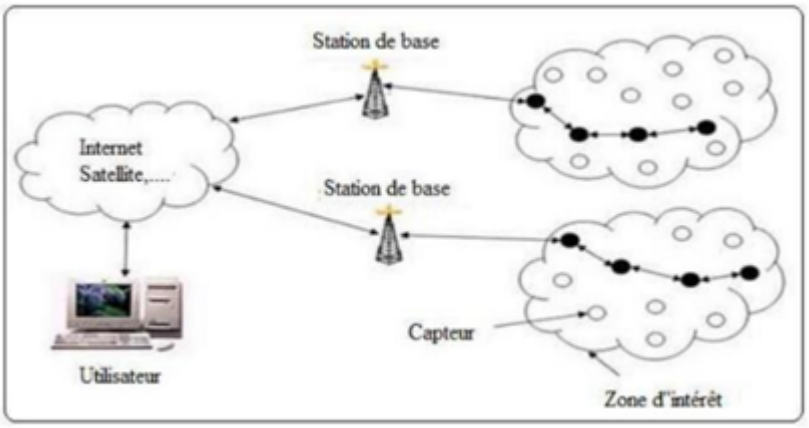


FIGURE 2.3 – L'architecture d'un réseau sans fil

2.4.1 Modèle en couche

La communication dans les RCSF est régie suivant le modèle en couche schématisé sur la figure 2.4 Ce modèle décrit les différentes couches de la communication, leurs rôles ainsi que les différents plans de gestions de l'énergie, de la mobilité et des tâches. Ce modèle a pour rôle de standardiser la communication entre les différents composants du réseau afin que les matériels de différents constructeurs puissent être compatibles. A l'instar du modèle de référence OSI (Open System Interconnexion), ce modèle comporte cinq couches qui sont la couche physique, la couche liaison de donnée, la couche réseau, la couche transport et la couche application.

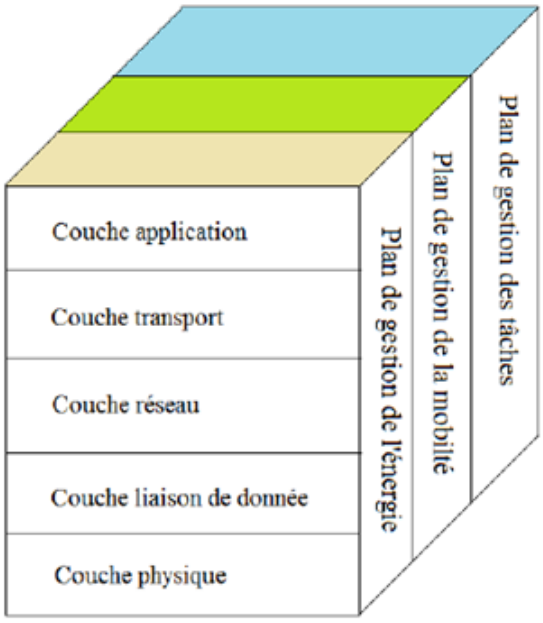


FIGURE 2.4 – Modèle en couche pour la communication dans les RCSF

• **Couche physique :**

La couche physique est responsable du choix de la fréquence, de la génération de la fréquence porteuse, de la détection du signal, de la modulation et du chiffrage des données

• **Couche liaison de données :**

Beaucoup de recherches dans le domaine des réseaux de capteurs entrent dans le cadre de l'optimisation de la couche de liaison. Elle manipule toutes les issues de communication entre les nœuds voisins. Dans les réseaux sans fil, l'accès au médium commun (la fréquence) doit être contrôlé. Ceci est appelé le contrôle d'accès au Medium (MAC : Medium Access Control). La tâche principale de cette couche est d'interdire l'accès simultané au canal dans la même marge de fréquence radio

• **Couche réseau :**

La couche réseau en ce qui concerne les réseaux de capteurs est habituellement conçue selon les principes suivants :
L'efficacité d'énergie est toujours une considération importante.
Les réseaux de capteurs sont la plupart du temps data-centric
L'agrégation de données est une fonctionnalité majeure.

• **Couche transport :**

La couche transport fournit un service de communication de bout en bout, fiable pour l'application. Elle manipule la segmentation des grands paquets. Elle effectue le contrôle des flots de données de bout en bout afin d'éviter la surcharge du récepteur ou du réseau.

• **Couche application :**

Selon les tâches de capture, il existe différents types de logiciels qui peuvent être installés et employés pour la couche application.

2.4.2 Plan de gestion de l'énergie

Le plan de gestion de l'énergie contrôle l'utilisation de la batterie. Par exemple, après la réception d'un message, le module radio du nœud récepteur peut être éteint afin que ce dernier puisse économiser son énergie. En outre, si le niveau d'énergie d'un capteur donné devient très faible alors ce nœud peut diffuser une alerte à l'ensemble de ses voisins afin de les informer qu'il ne peut plus participer au routage, ce qui peut renforcer la tolérance aux pannes. L'énergie résiduelle de ce nœud pourra ainsi être réservée pour d'autres fonctions, par exemple le captage.

2.4.3 Plan de gestion de la mobilité

Le plan de gestion de la mobilité permet de détecter et d'enregistrer le mouvement d'un nœud capteur donné dans la zone d'intérêt. Un retour arrière vers l'utilisateur est toujours conservé et le nœud peut garder des traces sur l'ensemble ses nœuds voisins. Avec les informations sur l'état

de ses voisins, les nœuds capteurs peuvent mieux coordonner et gérer l'utilisation de leur énergie pour la réalisation de leurs différentes tâches.

2.4.4 Plan de gestion des tâches

Le plan de gestion des tâches permet de faire l'ordonnancement des différentes tâches de captage des données dans une zone de surveillance donnée. Dans certaines topologies de déploiements denses où les champs de captages sont souvent redondants, il n'est pas nécessaire que tous les nœuds capteurs d'une zone donnée effectuent en même temps une tâche donnée. Certains nœuds capteurs peuvent donc effectuer les tâches de captage au moment où d'autres nœuds vont se mettre dans le mode éteint afin de sauvegarder leur énergie.

2.5 Applications des réseaux de capteurs sans fil

La miniaturisation des micro-capteurs, le coût de plus en plus faible, la large gamme des types de capteurs disponibles (thermique, optique, vibrations, etc.) ainsi que le support de communication sans fil utilisé, permettent l'application des réseaux de capteurs dans plusieurs domaines parmi lesquels[7] :

2.5.1 Application militaire

Comme pour de nombreuses autres technologies, le domaine militaire a été le moteur initial pour le développement des réseaux de capteurs. Le déploiement rapide, le coût réduit, l'auto-organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui font de ce type de réseaux un outil appréciable dans un tel domaine. Actuellement, les RCSFs peuvent être une partie intégrante dans le commandement, le contrôle, la communication, la surveillance, la reconnaissance, etc.

la figure 2.5 montre un cas d'une application militaire

FIGURE 2.5 – Application militaire

2.5.2 Application médicale

Les réseaux de capteurs sont également largement répandus dans le domaine médical. Cette classe inclut des applications comme : fournir une interface d'aide pour les handicapés, collecter

des informations physiologiques humaines de meilleure qualité, facilitant ainsi le diagnostic de certaines maladies, surveiller en permanence les malades et les médecins à l'intérieur de l'hôpital.

la figure 2.6 montre une image d'une application médicale

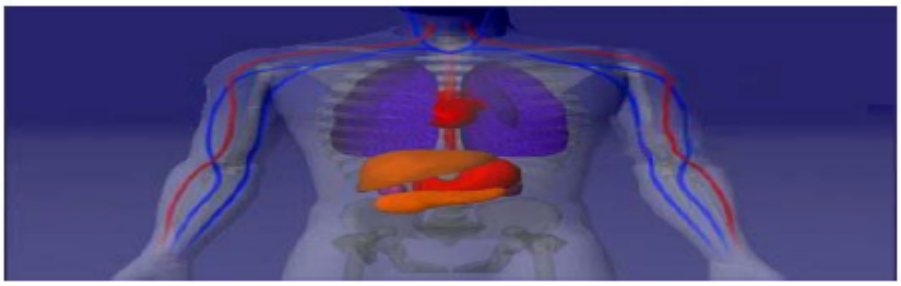


FIGURE 2.6 – Application médicale

2.5.3 Application d'agriculture

Les RCSFs peuvent améliorer le domaine de l'agriculture. Ils offrent un support important qui permettra la gestion précise des ressources, le suivi de développement des maladies, la prédiction de moment de récolte. Ils nécessitent un coût d'installation réduit, et ils sont faciles à déployer sur le terrain, puisque la mise en place des capteurs ne demande pas un câblage spécial. Les capteurs collectent des données sur la qualité de sol et les transmettent à une station centrale dans la ferme.

la figure 2.7 montre un réseau de capteur pour l'agriculture

FIGURE 2.7 – Réseau de capteur pour l'agriculture.

2.5.4 Application commerciale

Parmi les domaines dans lesquels les réseaux de capteurs ont aussi prouvé leur utilité, on trouve le domaine commercial. Dans ce secteur on peut énumérer plusieurs applications comme : la surveillance de l'état du matériel, le contrôle et l'automatisation des processus d'usinage, etc

2.5.5 Application environnementale

Dans ce domaine, les capteurs peuvent êtres exploités pour détecter les catastrophes naturelles (feux de forêts, tremblements de terre, etc.), détecter des émanations de produits toxiques (gaz, produits chimiques, pétrole, etc.) dans des sites industriels tels que les centrales nucléaires ou pétrolières.

la figure 2.8 montre une image de la surveillance de l'environnement

FIGURE 2.8 – Surveillance de l'environnement

2.6 Risques liés aux réseaux de capteurs sans fil

2.6.1 Manque de sécurité

Les ondes radioélectriques ont intrinsèquement une grande capacité à se propager dans toutes les directions avec une portée relativement grande. Il est ainsi très difficile d'arriver à confiner les émissions d'ondes radio dans un périmètre restreint. La propagation des ondes radio doit également être pensée en trois dimensions. Ainsi les ondes se propagent également d'un étage à un autre (avec de plus grandes atténuations). La principale conséquence de cette "propagation sauvage" des ondes radio est la facilité que peut avoir une personne non autorisée d'écouter le réseau, éventuellement en dehors de l'enceinte du bâtiment où le réseau sans fil est déployé.

2.6.2 War-driving

Etant donné qu'il est très facile d'écouter des réseaux sans fils, une pratique venue tout droit des Etats-Unis consiste à circuler dans la ville avec un ordinateur portable (voire un assistant personnel) équipé d'une carte réseau sans fil à la recherche de réseaux sans fils, il s'agit du war driving (parfois noté wardriving ou war-Xing pour "war crossing"). Des logiciels spécialisés dans ce type d'activité permettent même d'établir une cartographie très précise en exploitant un matériel de géolocalisation (GPS : Global Positionning System).

2.6.3 Risques en matière de sécurité

Les risques liés à la mauvaise protection d'un réseau sans fil sont multiples :

a) Interception de données consistant à écouter les transmissions des différents utilisateurs du réseau sans fil

b) Détournement de connexion dont le but est d'obtenir l'accès à un réseau local ou à internet

c) Brouillage des transmissions consistant à émettre des signaux radio de telle manière à produire des interférences

d) Dénis de service rendant le réseau inutilisable en envoyant des commandes factices

2.7 Caractéristiques des réseaux des capteurs

2.7.1 Système d'exploitation TinyOS

Présentation

TinyOS est un système d'exploitation Open Source pour les réseaux des capteurs, conçu par l'université américaine de BERKELEY. Le caractère open source permet à ce système d'être régulièrement enrichie par une multitude d'utilisateurs. Sa conception a été entièrement réalisée en NesC, langage orienté composant syntaxiquement proche du C. Il respecte une architecture basée sur une association de composants, réduisant ainsi la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les capteurs, pourvus de ressources très limités dues à leur miniaturisation. Pour autant, la bibliothèque des composants de TinyOS est particulièrement complète, puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs, et des outils d'acquisition de données. Un programme s'exécutant sur TinyOS est constitué d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application à laquelle il sera destiné (mesure de température, du taux d'humidité...). TinyOS s'appuie sur un fonctionnement évènementiel, c'est à dire qu'il ne devient actif qu'à l'apparition de certains évènements, par exemple l'arrivée d'un message radio. Le reste du temps, le capteur se trouve en état de veille, garantissant une durée de vie maximale connaissant les faibles ressources énergétiques des capteurs. Ce type de fonctionnement permet une meilleure adaptation à la nature aléatoire de la communication sans fil entre capteurs.

La figure 2.9 montre un système d'exploitation TinyOS.[3]

FIGURE 2.9 – Système d'exploitation TinyOS

2.7.2 Propriétés de OS

TinyOS est basé sur quatre grandes propriétés qui font que ce système d'exploitation, s'adapte particulièrement bien aux systèmes à faible ressources :

Evénementiel : Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements qui se déclenche. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'évènements, ceux-ci ayant la plus forte priorité. Ce fonctionnement évènementiel (event driven) s'oppose au fonctionnement dit temporel (time driven), où les actions du système sont gérées par une horloge donnée.

Non préemptif : Le caractère préemptif d'un système d'exploitation précise si celui-ci permet l'interruption d'une tâche en cours. TinyOS ne gère pas ce mécanisme de préemption entre les tâches, mais donne la priorité aux interruptions matérielles. Ainsi, les tâches entre elles ne s'interrompent pas mais une interruption peut stopper l'exécution d'une tâche.

Pas de temps réel : Lorsqu'un système est dit « temps réel » celui-ci gère des niveaux de priorité dans ses tâches, permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel. TinyOS se situe au-delà de ce second type, car il n'est pas prévu pour avoir un fonctionnement temps réel.

Consommation d'énergie : TinyOS a été conçu pour réduire au maximum la consommation en énergie du capteur. Ainsi, lorsqu'aucune tâche n'est pas active, il se met automatiquement en veille.

2.8 Energie de capteur sans fils

Introduction

Les capteurs sans fil sont des éléments indépendants les uns des autres, comme leur nom l'indique. Par conséquent, ils doivent également disposer d'une alimentation autonome. Leur durée de vie est limitée par la durée de vie de leur batterie. Cette contrainte forte a une influence majeure sur l'ensemble des techniques, mises en place pour le déploiement de tels réseaux. Un effet majeur de cette limitation énergétique est la limitation maximale des transmissions par voie hertzienne, très coûteuses. Il est donc primordial d'effectuer tant que possible le traitement de l'information localement au niveau du nœud. L'enjeu est donc d'étendre la durée de vie du système et sa robustesse, en cas de chute de certains nœuds seulement. Les problématiques sont donc très éloignées de celles des réseaux classiques, telle la maximisation du débit. Dans les réseaux de capteur sans fils, il faut assurer une consommation répartie de l'énergie au sein du réseau. Cette énergie est consommée par les diverses fonctionnalités de réseaux qui sont donc triés par ordre décroissant de consommation d'énergie

Radio (Communication)
Protocoles (MAC, routage)
CPU (calcul, agrégation)
Acquisition

2.8.1 Problèmatique

L'enjeu d'énergie est capital dans les réseaux de capteur, pour augmenter l'autonomie de capteur il faut agir sur plusieurs paramètres : Sur quels paramètres est-il possible d'agir ? Point de vue « circuit » L'interdépendance des paramètres est un casse-tête pour réaliser une optimisation

- Si on augmente le taux de transmission
  * La probabilité de collision diminue
  * Le taux d'erreur augmente
  * La consommation augmente
- Si on augmente la puissance de la correction d'erreur
  * Le taux d'erreur diminue

* La probabilité de collision augmente
* La consommation augmente
• Si on augmente la puissance d'émission
* Le taux d'erreur diminue
* La probabilité de collision augmente
* La consommation augmente

La consommation énergétique du module de surveillance dépend énormément du matériel employé et de la nature du phénomène observé. L'économie d'énergie obtenue par la mise en veille de certains nœuds pour l'observation est donc très variable.

La figure 2.10 montre un circuit de capteur

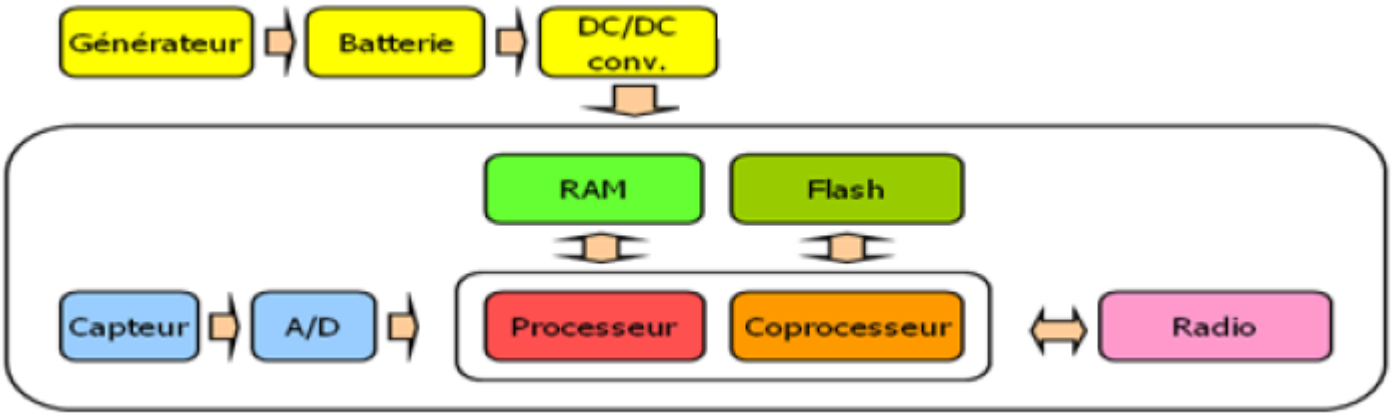


FIGURE 2.10 – Exemple d'un circuit de capteur

2.9 Ordonnancement de réseaux de capteur

Les réseaux de capteurs sont généralement denses et redondants. En effet, suivant l'application, on déploiera plus ou moins des capteurs dans un souci d'allongement de la durée de vie de l'application. À tout moment, il existe donc des capteurs qui observent une même portion de la zone de déploiement. Cette redondance est exploitée par l'ordonnancement d'activité : Ordonnancer l'activité dans un réseau de capteurs consiste à alterner les charges de façon à épuiser les nœuds équitablement. Pendant qu'une partie participe à l'application, les autres sont dans un mode passif, économisant ainsi leur énergie.

2.10 Gestion d'ordonnancement

L'ordonnancement d'activité peut se faire de diverses façons. Nous distinguons ici les approches centralisées (où une entité centrale connaît chaque nœud est capable d'influer sur chacun pour lui assigner ses tâches) des approches hiérarchiques (une vision hiérarchisée du réseau où l'autorité centrale est démultipliée selon plusieurs entités responsables d'une sous-partie du réseau) et des approches localisées, par conséquent totalement décentralisées, dans lesquelles un comportement global cohérent doit être obtenu à partir de décisions prises localement.

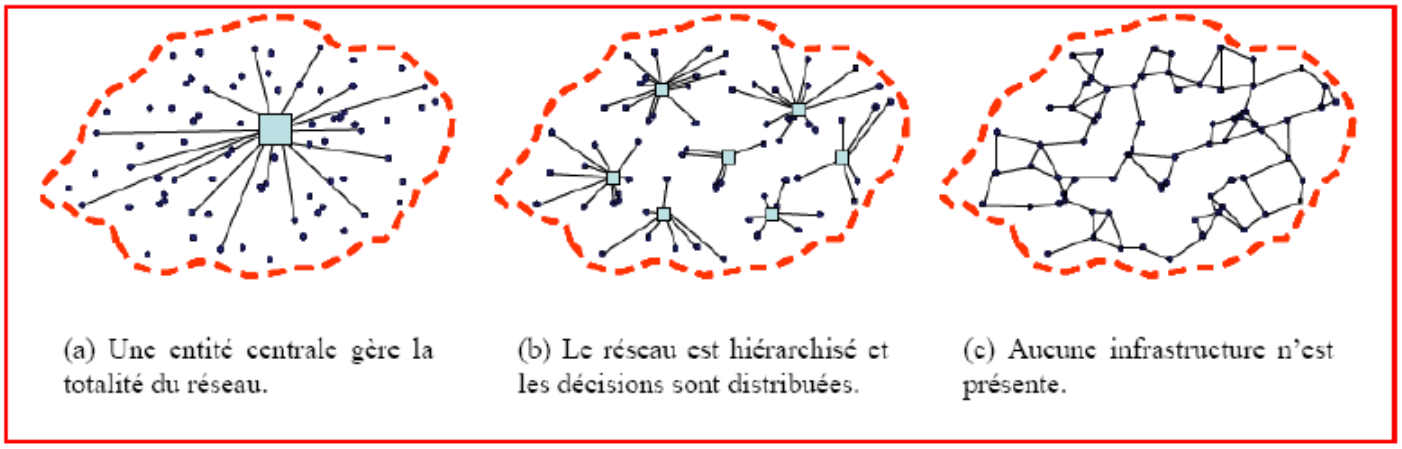


FIGURE 2.11 – Gestion d'ordonnancement

2.11 Déploiement de capteur

Les capteurs sont au préalable déployés sur une zone à surveiller. Pour satisfaire de nouvelles contraintes ou pour pallier des pannes, un déploiement de nœuds supplémentaires, dit itératif, peut être requis. Différents modes de déploiement sont envisageables et dépendent essentiellement de l'application de surveillance. Une fois déployés, nous supposons que les capteurs sont statiques.

- Déploiement déterministe

Lorsque l'environnement est accessible ou connu, il est possible de placer précisément les nœuds sur la zone. Dans le problème que nous étudions, il est alors possible de programmer leurs activités au préalable. C'est ainsi, par exemple, que sont mis en place les capteurs chargés de réguler la climatisation d'un immeuble ou de surveiller les constantes médicales de malades à distance. On parle alors de déploiement déterministe.

- Déploiement aléatoires

L'utilisation des capteurs dans des zones inaccessibles ou sensibles rend impossible un déploiement déterministe, au cours duquel chaque objet serait placé à une position prédéterminée. Les nœuds peuvent alors être déployés à l'aide de moyens divers. Il a souvent été question d'un déploiement aléatoire des capteurs, effectué comme un jeté de graines. Il semble pourtant difficile, vu la fragilité des capteurs existants, d'envisager un déploiement par avion par exemple. Néanmoins, un déploiement aléatoire peut être obtenu à partir d'une distribution de capteurs à des individus. Dans ce travail, nous supposerons des réseaux déployés aléatoirement. Une fois disséminés, il est couramment admis que les capteurs sont statiques.

2.12 Conclusion

Dans ce deuxième chapitre, nous avons fait une présentation générale sur les RCSF en décrivant leurs fonctionnements, Les architectures et les modèles de communications et, leurs domaines d'applications, les risques liés aux réseaux de capteurs sont également abordés.

Chapitre 3

Quelques Techniques d'optimisations dans les réseaux de Capteurs

3.1 Introduction

La théorie des graphes a été largement étudiée et de nombreux problèmes concrets ont été modélisés par les graphes et résolus via les techniques d'optimisations dans les graphes. Dans ce chapitre nous allons présenter quelques problèmes d'optimisations pour lesquels nous allons utiliser certaines techniques de la théorie des graphes pour les résoudre.

3.2 Modélisation de réseaux de capteur

Un réseau considéré est un ensemble de nœuds déployés aléatoirement dans une zone de capteur, ou chaque nœud capteur envoie ses données à la station de base par un intermédiaire de ses nœuds voisins.

La Figure 3.1 montre un réseau de capteur sans fil.

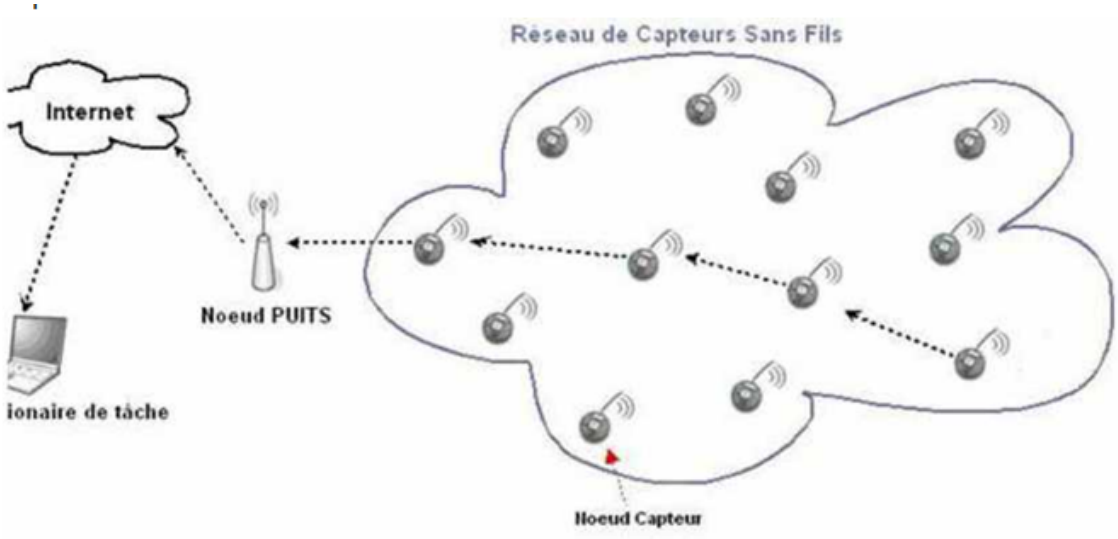


FIGURE 3.1 – Un réseau du capteur

3.3 Solution proposée

la modélisation de notre problème qui est la minimisation d'énergie dans les RCSF, nous a conduit à l'étude de sommets et d'arêtes, sachant que les nœuds capteurs sont dotés d'une petite batterie. Dans ce fait, nous avons utilisé l'un des algorithmes de théorie des graphes qui est l'algorithme de dijkstra, qui nous permettra de déterminer les routes à emprunter avec un chemin de plus faible distance (métrique).

Remarque :
Pour un chemine (p)allant de nœuds source S au nœuds puits(Station de base),on a
W= $\sum\ C(e_i)$ avec $e_i \in p$ et $C(e_i)$est la distance attribuée à l'arête e_i

3.4 Hypothèses

a. Supposons que les capteurs de réseau sont numérotés par les entier de l'ensemble $\{1, 2, \ldots, n\}$ et deux capteurs à distance i et j sont en communication directe si la distance dij entre i et j et inférieur $k \in N^*$.

b. L'énergie consommée par un capteur émetteur, notée et qui est proportionnelle a la taille du paquet envoyé est la distance qui sépare le capteur émetteur et le capteur récepteur.

c. L'énergie consommée par un capteur récepteur notée Er est proportionnelle à la taille de paquet reçu comme le capteur récepteur consomme une énergie proportionnelle à la taille du paquet, alors on va prendre en considération la distance qui sépare un capteur émetteur et un capteur récepteur.

Une fois les distances ont été déterminées on va chercher le plus court chemin de la station de base vers tous les autres capteurs. Pour ce faire nous allons utiliser l'algorithme de Djikstra.

3.5 Problèmes d'optimisations dans les réseaux

Il y a plusieurs problèmes d'optimisations dans les graphes valués connexes (Réseaux), dans notre travail nous allons étudier le problème de cheminements, le problème de routage et problème d'arbre de poids minimum.

3.5.1 Problème de plus court chemin

Les problèmes de cheminements sont des problèmes classiques de la théorie des graphes. L'objectif est de calculer une route entre les sommets d'un graphe qui minimise ou maximise une certaine fonction économique. Le problème le plus classique consiste à chercher le chemin qui minimise la somme des évaluations des arêtes traversées. Il existe des algorithmes pour le résoudre, comme l'algorithme de Dijkstra, Bellman, Ford... On peut s'intéresser à la recherche d'un plus court chemin dans un graphe :

P1 – entre un sommet source et puits.
P2 - entre tous les couples de sommets.
P3 – d'un sommet à tous les autres sommets.

Notons qu'il n'y a pas de solution spécifique au problème consistant à chercher un plus court chemin entre deux sommets x et y particuliers d'un graphe, il se résout en résolvant (P1) : partant de x en s'arrêtant en y.

L'algorithme de DIJKSTRA est sans doute le plus utilisé car il est aisé à mettre en œuvre, efficace en temps d'exécution et bien adapté aux situations courantes, c'est pourquoi nous en donnerons une écriture détaillée.

Algorithme de DIJKSTRA

Cet algorithme n'est utilisable que dans le cas où les coûts des arcs sont tous positifs ou nuls. Il calcule le plus court chemin entre une source « s » et tous les sommets accessibles depuis « s ». On obtient une arborescence, de racine « s » formée par ces plus courts chemins.

Principe

Soit G = un graphe de « n » capteurs. Soit s le capteur qui va émettre, et G un graphe dont on veut trouver les plus courts chemins aux autres sommets. Cet algorithme doit donc déterminer pour chaque sommet m le chemin le plus court entre s et m dans le graphe G. L'algorithme de Dijkstra est un algorithme de type glouton1 : à chaque nouvelle étape, on traite un nouveau sommet. Reste à définir le choix du sommet à traiter, et le traitement à lui infligé. Tout au long du calcul, on va donc maintenir deux ensembles :

• C, l'ensemble des sommets qui restent à visiter ; au départ $C = S - \{source\}$.

• D, l'ensemble des sommets pour lesquels on connaît déjà leur plus petite distance à la source ; au départ,$D = \{source\}$.

L'algorithme se termine bien évidemment lorsque C est vide.

• Dijkstra peut trouver le plus court chemin d'un nœud source à tous les autres.

• Dijkstra peut trouver le plus court chemin entre 2 sommets, en lançant l'algorithme depuis un nœud source bien défini « x », jusqu'à ce qu'il atteigne un point « y » spécifique.

• Dijkstra peut trouver le plus court chemin entre tous les nœuds d'un graphe en le lançant sur tous ces nœuds. [3]

Algorithmique

Algorithme de Dijkstra [4]

Données : R=(S,A,V), s :racine

Résultats :π :Les Plus courtes distances , B : Arborescence , C : Sous-ensemble de sommets accessibles à partir de s.

a) Initialisation : $C = \{s\}$, $\pi(s) = 0$, $B = \phi$, $\pi(j) = \{v_{sj} si (s,j) \in A, +\infty sinon\ \}$

b) Procédure de calcul :

(.) Chercher un sommet $i \in (S - c)$
vérifiant $\pi(i) = min\{\pi(j)\}\ j \in (S - c)$

(..) Poser :$C = C \cup \{i\}$ et $\pi(j) = min\{\pi(j), \pi(i) + v_{ij}\}$ $j \in \Gamma i \cap (S - c)$

c) Test d'arrêt : Si |C| = |S| , terminer,
Sinon : retourner en (b)

3.6 Détail de la solution proposée

3.6.1 Objectif de notre modélisation

Modélisation de système par un graphe.

Pour cela nous avons modélisé notre problème avec un graphe. L'algorithme de Dijkstra permet de déterminer la distance minimale et d'acheminer notre information d'un nœud capteur vers une station de base en minimisant l'énergie consommée.Dans le but de déterminer le chemin minimal d'acheminement de l'information d'un nœud capteur à une station de base, nous avons fait appel à l'algorithme de Dijkstra.

L'application de l'algorithme de Djikstra au réseau de la figure 3.2 est comme suit :

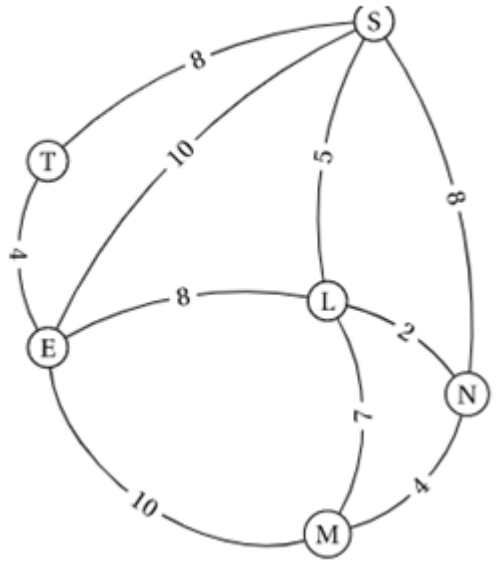


FIGURE 3.2 – Réseau de capteur d'ordre 6

Initialisaion

On construit un tableau ayant pour colonnes chacun des sommets du graphe. On ajoute à gauche une colonne qui recensera les sommets choisis à chaque étape (cette colonne est facultative mais facilitera la compréhension de l'algorithme).

Puisque l'on part du sommet M, on inscrit, sur la première ligne intitulée « Départ », 0-$\{M\}$ 0_M dans la colonne M et ∞ dans les autres colonnes.

Cela signifie qu'à ce stade, on peut rejoindre M en 0 minute et on n'a rejoint aucun autre sommet puisque l'on n'a pas encore emprunté de chemin...

	E	L	M	N	S	T
Départ	∞	∞	0_M	∞	∞	∞

TABLE 3.1 – Initialisaion

Itération 1

On sélectionne le plus petit résultat de la dernière ligne. Ici, c'est « 0-$\{M\}$ 0_M » qui correspond au chemin menant au sommet M en 0 minute.

• On met en évidence cette sélection (nous l'écrirons en rouge mais il est également possible de la souligner, de l'entourer, etc.).

• On inscrit le sommet retenu et la durée correspondante dans la première colonne (ici on écrit M(0)).

• On désactive les cases situées en dessous de notre sélection en les grisant par exemple. En effet, on a trouvé le trajet le plus court menant à M ; il sera inutile d'en chercher d'autres.

	E	L	M	N	S	T
Départ	∞	∞	0_M	∞	∞	∞
M (0)						

TABLE 3.2 – Itération 1

appliquant les procédures de l'algorithme et on obtient le résultat final à l'itération 5.

Itération 5

On sélectionne le plus petit résultat. C'est « 11-$\{L\}$ 11L » qui correspond au chemin menant au sommet S en 11 minutes. On a trouvé le trajet le plus court menant à S : il dure 11 minutes. Comme c'est la question posée dans l'énoncé, il est inutile d'aller plus loin et le tableau est terminé !

Il reste toutefois à reconstituer le trajet qui correspond à cette durée de 11 minutes. En pratique, il est plus facile de trouver le trajet en sens inverse en « remontant » dans le tableau de la façon suivante :

	E	L	M	N	S	T
Départ	∞	∞	0_M	∞	∞	∞
M (0)	10_M	7_M		4_M	∞	∞
N (4)	10_M	6_M		4_M	12_N	∞
L (6)	10_M				11_L	∞
E (10)					11_L	14_E

TABLE 3.3 – Itération 5

• On part de notre point d'arrivée : S

• On recherche la cellule marquée en rouge de la colonne S ; elle contient color $\{11 - \{L\}\}$ 11L. On note la lettre écrite en indice : L.

• On recherche la cellule marquée en rouge de la colonne L ; elle contient color $\{6 - \{N\}\}$ 6N On note la lettre écrite en indice : N.

• On recherche la cellule marquée en rouge de la colonne N ; elle contient color $\{4 - \{M\}\}$ 4M On note la lettre écrite en indice : M.

On est arrivé à notre point de départ M après être passé par N et L et S (liste obtenue en listant les sommets en ordre inverse).

Le trajet optimal est donc M - N - L - S.

Enfin, on peut vérifier sur le graphe que ce trajet est correct et dure 11 minutes ! [5]

3.6.2 Problème d'arbre couvrant de poids minimum [4]

En théorie des graphes, étant donné un graphe non orienté connexe dont les arêtes sont pondérées. L'arbre couvrant de poids minimum est aussi connu sous certains autres noms, tel qu'arbre couvrant minimum ou encore arbre sous-tendant minimum. L'arbre couvrant minimum peut s'interpréter de différentes manières selon le type de graphe. De manière générale si on considère un réseau où un ensemble d'objets doivent être reliés entre eux, l'arbre couvrant minimum est la façon de construire un tel réseau en minimisant un coût représenté par le poids des arêtes. Pour la recherche d'arbre couvrant de poids minimum, nous utiliserons l'un des algorithmes suivants :

Algorithme de Kruskal

Description simplifiée de l'algorithme de Kruskal [6]
1. Trier les m arêtes du graphe par valeurs croissantes.
2. Soit a_0 une des plus petites arêtes : $F = \{a_0\}$.
3. Soit p le nombre d'arêtes déjà placées dans le graphe partiel : p = 1.
4. Soit n le nombre de points du graphe (X, E, V).
5. Tant que p inférieur ou égal à n-2
et que toutes les arêtes n'ont pas été examinées
faire
Soit a_0 la plus petite arête non encore examinée.
Si l'ajout de a_0 à F ne crée pas de cycle dans le graphe partiel
alors
ajouter a_0 à F ;

p = p + 1 ;
finsi
Fin Tant que
6. Si p = n-1 alors
F contient les arêtes d'un arbre de recouvrement minimal
sinon
le graphe initial n'était pas connexe et il n'est pas possible de trouver un arbre de recouvrement minimal.
finsi

L'application de l'algorithme de Kruskal au réseau de la figure 3.6 est comme suit : Appliquons l'algorithme au graphe suivant :

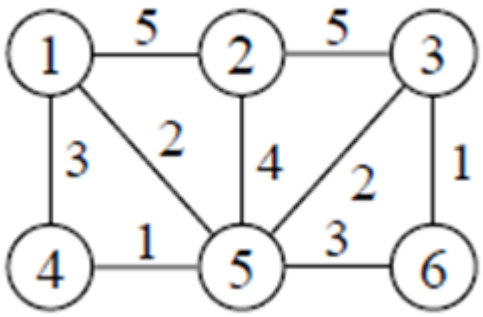


FIGURE 3.3 – Exemple d'application

Les arêtes de poids 3 n'ont pas pu être placées, car elles auraient formé un cycle. L'algorithme s'est arrêté dès que cinq arêtes ont été placées. Toute arête supplémentaire aurait créé un cycle. S'il y a plusieurs arêtes de même poids, il peut y avoir plusieurs arbres couvrants de poids minimum : tout dépend de l'ordre dans lequel ces arêtes ont été triées.

Nous avons alors obtenu l'arbrede la figure 3.7 :

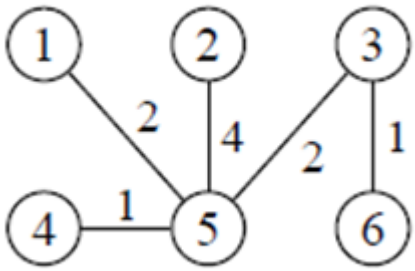


FIGURE 3.4 – Arbres couvrants de poids minimum

Algorithme de Prim : [7]

Algorithme de recherche d'un arbre de poids minimum dans un graphe (The Shortest Spanning Tree of the graphe)(SST)

Etape 1 :
soit $T_s = \{Xs\}$,ou x_s est un sommet choisit arbitrairement, $A_s = \phi$ (As est l'ensemble des arêtes qui forment le (SST)).

Etape 2 :
Pour tout sommet $x_j \notin T_s$, chercher un sommet $a_j \in T_s$, tel que $C(x_i, x_j)$=$min[C(x_i, x_j)] = B_j$
$x_j \in T_s$
où $C(x_i, x_j)$ est le cout de l'arête x_i, x_j.
Si $N(x_j) \cap T_s = \phi$ alors $B_j = \infty$

Etape 3 :
Choisir un sommet x_j^*,tel que $B_j^* = min[B_j]$
$x_j \notin T_s$
$T_s = T_s \cup \{x_j^*\}$,$A_s = A_s \cup \{a_j x_j^*\}$
Si |Ts| = n,terminé,A_s forme le (SST)
Si |Ts| < n,aller en 4

Etape 4 :
Pour tout sommet $x_j \notin T_s$ et $x_j \in N(x_j^*)$,alors :
Si $B_j > C(x_j^*, x_j)$,$B_j = C(x_j^*, x_j)$,$a_j = x_j^*$ et aller en 3
Si$B_j < C(x_j, x_j)$ aller en 3

(S contient les sommets choisis, C les candidats possibles, d(y) représente la distance minimale courante de T à y et p(y) le voisin de y dans S correspondant à l'arête courante minimale en y). Appliqué à notre exemple l'algorithme de PRIM, à partir de 1, peut sélectionner les sommets dans l'ordre : 3 (par l'arête 13, on aurait pu choisir 2 par l'arête 12), 4 (par l'arête 34), 2 (par l'arête 42), 5 (par l'arête 45), 7 (par l'arête 57) et 6 (par l'arête 67, on aurait pu choisir 6 par l'arête 36). Tous les sommets ayant été atteints l'algorithme se termine avec un arbre recouvrant de poids 42, l'arbre obtenu n'est pas le même que le précédent, mais il est de même valeur

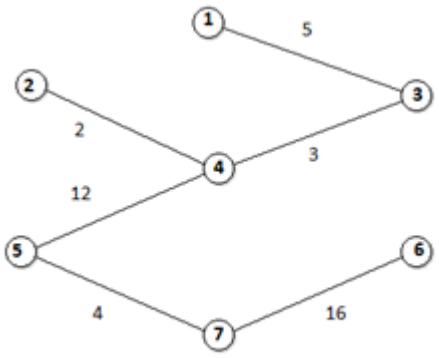


FIGURE 3.5 – L'application de l'algorithme de prim

3.6.3 Routage dans le RCSF

Le routage est un processus qui permet de sélectionner des chemins dans un réseau pour transmettre des données depuis un expéditeur jusqu'à un ou plusieurs destinataires. On parle de routage dans différents domaines : réseaux électronique (comme Internet), réseaux de transports. Le routage est le mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires. Le routage est une tâche exécutée dans de nombreux réseaux de transports. Sa performance est importante dans les réseaux décentralisés, c'est-à-dire où l'information n'est pas distribuée par une seule source, mais échangée entre des agents indépendants. Le terme routage désigne l'ensemble des mécanismes mis en œuvre dans un réseau pour déterminer les routes qui vont acheminer les paquets d'un terminal émetteur à un terminal récepteur. On distingue généralement deux entités : *L'algorithme de routage. *Le protocole de routage. [14]

3.6.3.1 Protocoles de routages

Dans les réseaux de capteurs, la conservation de l'énergie qui a une grande influence sur la durée de vie du réseau, est considérée, relativement, plus importante que les performances du réseau. Donc les protocoles de routages doivent permettre une consommation minimale de l'énergie sans dégrader considérablement les performances du réseau.

3.6.3.2 Protocoles de routages pour les réseaux Ad Hoc

Puisque les réseaux de capteurs font partie des réseaux ad hoc, il est intéressant de présenter quelques protocoles de routage de ces derniers. Selon l'information utilisée pour calculer les routes deux types de routage existent [9] :

Routage à état de liens : Maintenir dans chaque nœud une carte du réseau où figurent les nœuds et les liens entre eux aidant à construire les tables de routage. Par exemple : le protocole OLSR (Optimized Link State Routing) et le Protocole FSR (Fisheye State Routing).

Routage à vecteur de distance : Plutôt que de maintenir une carte complète du réseau, ces protocoles ne conservent que la liste des nœuds du réseau et l'identité du voisin par lequel on

passe pour atteindre la destination par le chemin le plus court. Par exemple : le protocole AODV (Ad-hoc On Demand Distance Vector).

Selon la méthode utilisée pour construire une route entre un noeud source et une autre de destination trois classes de protocole se figure [9] :

•**L'approche réactive :** Ces protocoles ne cherchent à calculer une route qu'à la demande d'une application. Cela permet d'économiser de la bande passante et de l'énergie. Par exemple : le protocole DSR (Dynamic Source Routing).

•**L'approche proactive :** Le principe de base est de maintenir à jour les tables de routage périodiquement par l'échange de trames de contrôle, de sorte que lorsqu'un nœud désire envoyer un paquet à un autre, une route sera immédiatement connue. Par exemple : le protocole DSDV, le protocole OLSR et le Protocole FSR.

•**Les protocoles hybrides :** Le principe est de connaître notre voisinage de manière proactive jusqu'à une certaine distance (par exemple trois ou quatre sauts), et d'effectuer une recherche réactive à l'extérieur de cette zone. L'un des protocoles de ce type est le Protocole ZRP (Zone Routing Protocol).

3.6.3.3 Routage des données

L'acheminement des paquets constitue la fonction principale d'un réseau. Cette fonction influe sur l'énergie de calcul et de communication. Un nœud qui prend la décision d'envoyer un paquet à un de ses voisins, doit calculer une fonction de coût. Ce calcul consomme une énergie qui s'ajoute à celle de la communication. L'énergie de routage totale représente la somme des énergies consommées par les nœuds capteurs tout le long du chemin [10]. La perte d'énergie due à un mauvais acheminement des paquets de données à un impact sur la durée de vie du réseau et peut conduire au partitionnement de ce dernier (dissipation totale de l'énergie des capteurs sur certaines routes).

3.7 Conclusion

Dans ce chapitre, nous avons présenté les différentes techniques d'optimisations dans les RCSFs, nous avons présenté les problèmes d'optimisations dans les graphes commençant par introduire le problème de cheminements utilisons l'algorithme de Djikstra pour établir le plus court chemin avec un exemple détaillé, puis on a fait une brève description de problème d'arbre couvrant de poids minimum en définissant l'algorithme de Kruskal et l'algorithme de Prim, pour terminer le chapitre par le routage dans les RCSFs.

Chapitre 4

Application

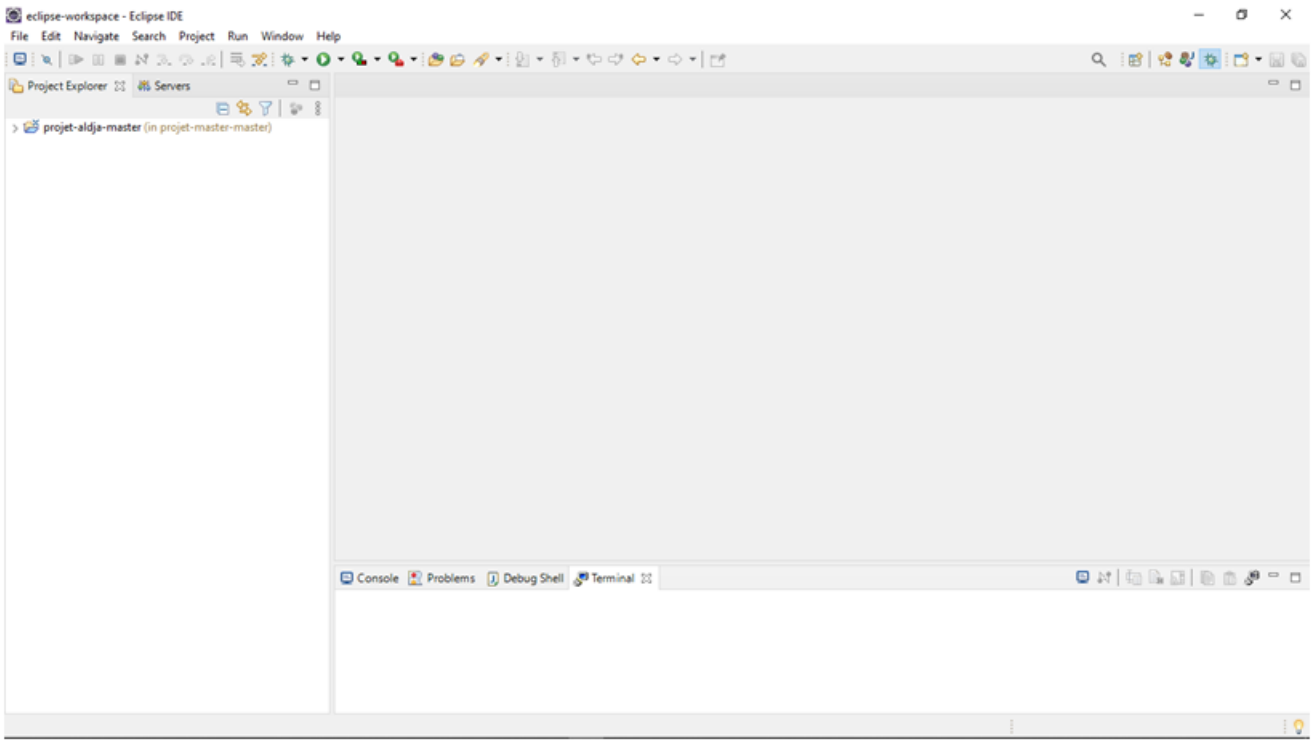
4.1 Introduction

Dans ce chapitre, nous allons présenter quelques situations concrètes que nous allons modéliser sous forme de graphe afin d'optimiser la consommation d'énergie avec des algorithmes cités précédemment

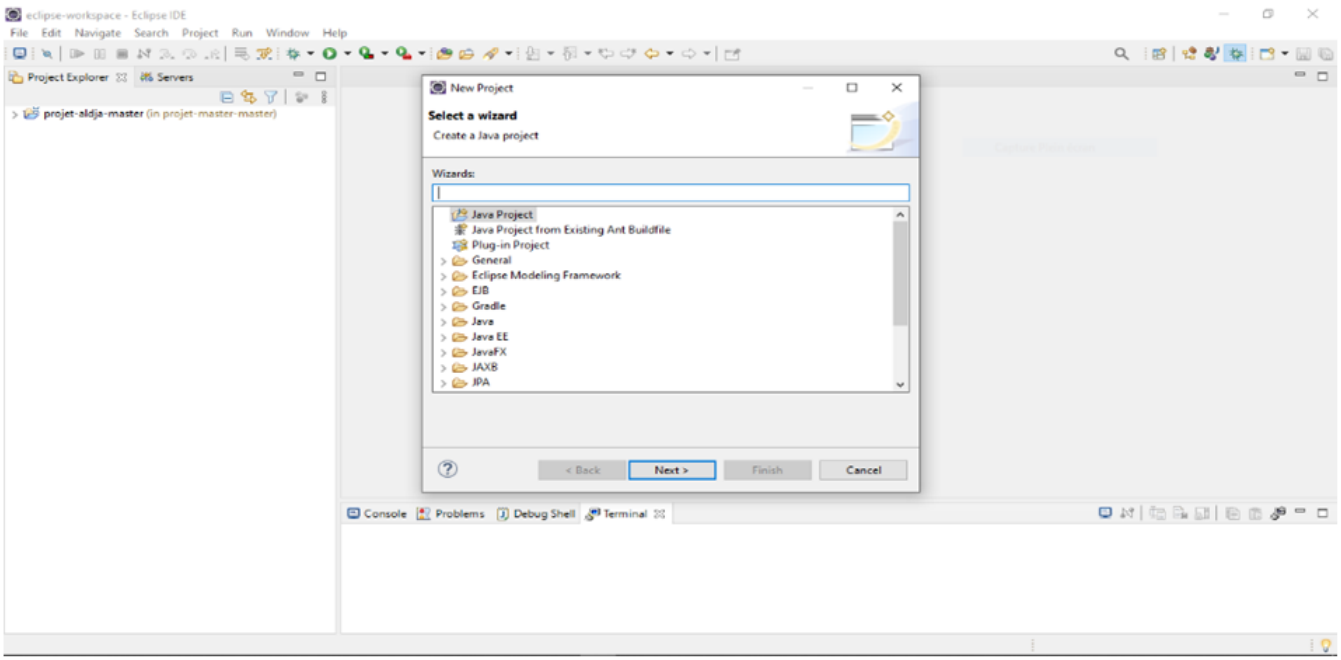
4.2 Introduction à l'environnement de travail (Eclipse)

FIGURE 4.1 – « Eclipse »

Eclipse est un environnement de développement intégré (IDE) utilisé dans la programmation informatique. Contient un espace de travail de base et un système de plug-in extensible pour personnaliser l'environnement. Eclipse est écrit principalement en Java et son utilisation principale est le développement d'application Java, mais il peut également être utilisé dans d'autres langages de programmation.

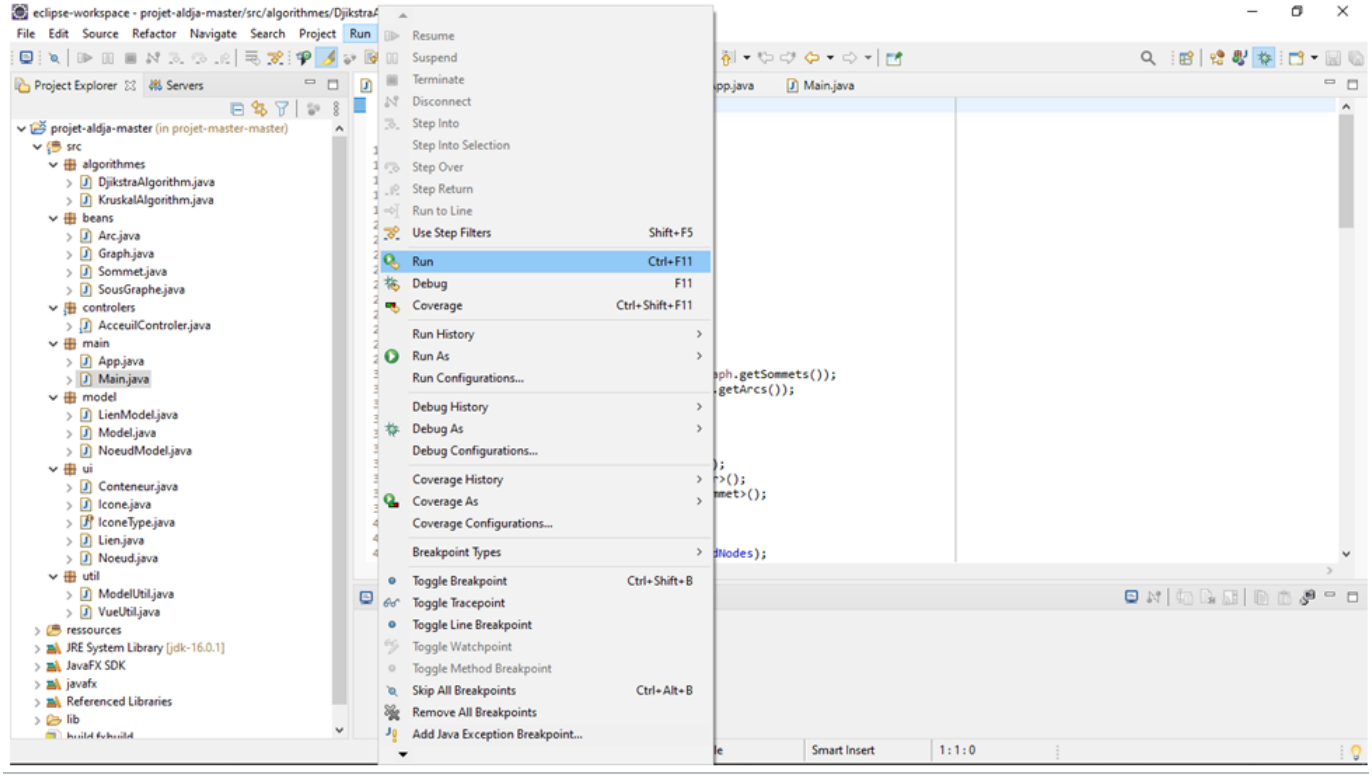


Eclipse fonctionne par projet. Pour créer un projet faites File→New→Java Project..., ou dans la fenêtre Package Explorer (sur la gauche) faites clic droit puis New, etc. Choisissez un nom de projet. Dans la partie Project layout (dans la seconde moitié de la fenêtre) vous pouvez demander à distinguer les dossiers pour les fichiers de sources et de classes. C'est évidemment ce qu'il faut faire. Si ce n'est pas déjà fait, sélectionnez donc Create separate source and output folders, puis cliquez sur Configure Defaults... et dans les Folders nommez les dossiers de source et de classes (par exemple src et classes). Cliquez Ok.



4.3 Exécutez le code

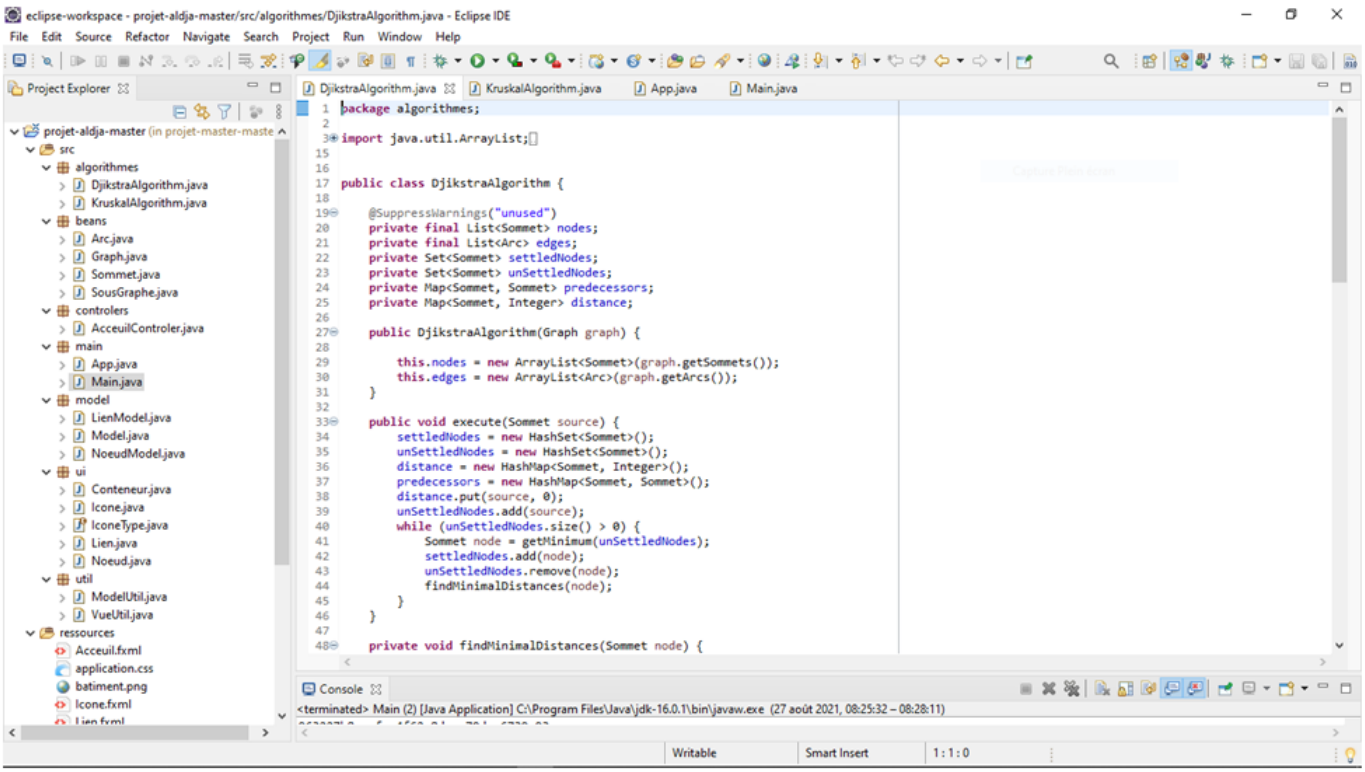
Créez une classe Main et dans la fenêtre de création cochez la création de la méthode main, ensuite complétez le code de cette méthode. Sélectionnez dans le menu Run→Run Configurations, choisissez Java Application. Cliquez sur le bouton New. Le champ Project : doit être déjà à jour. Cliquez sur Search... , les classes du projet contenant un main sont proposées (ici il n'y en a qu'une, elle a donc dû être proposée par défaut). Validez et cliquez Run. Le programme est alors exécuté. La trace apparaît dans une fenêtre console dans la partie inférieure de la fenêtre de l'IDE.

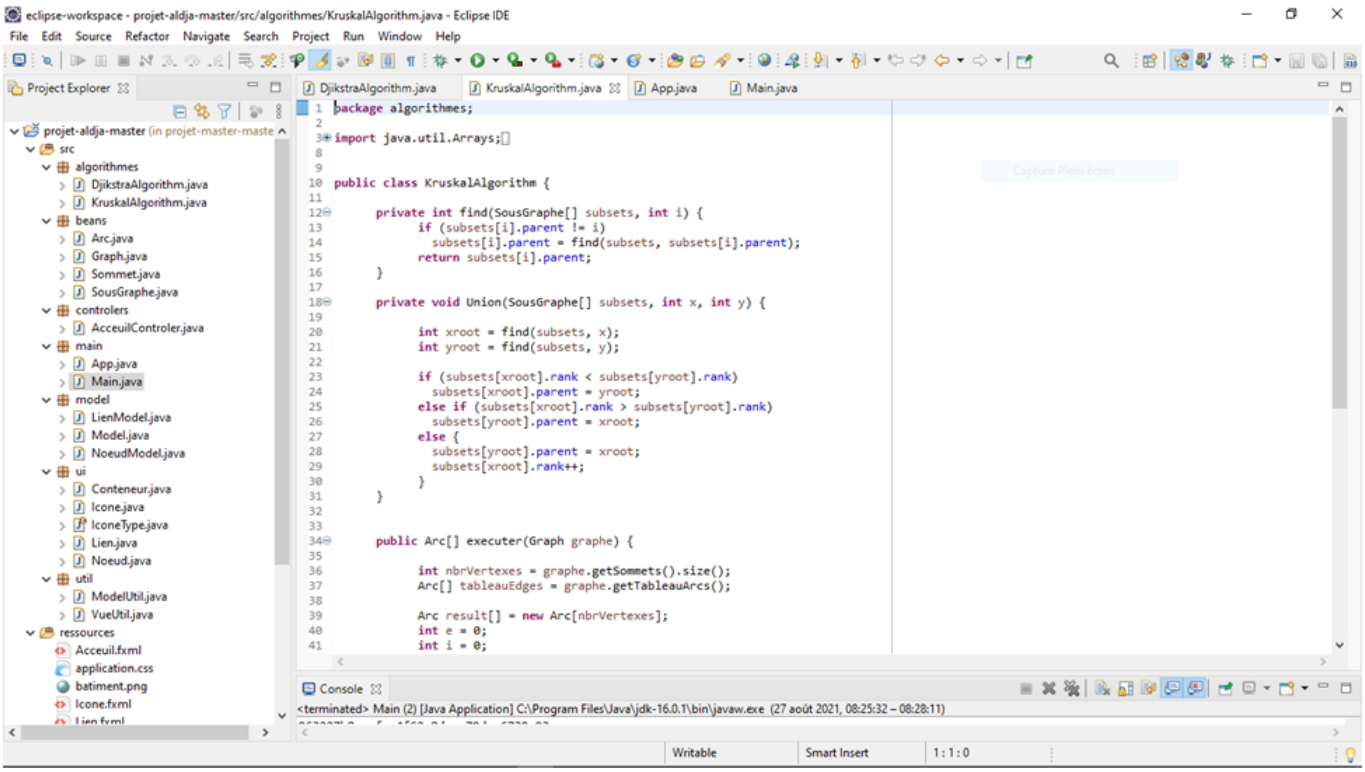


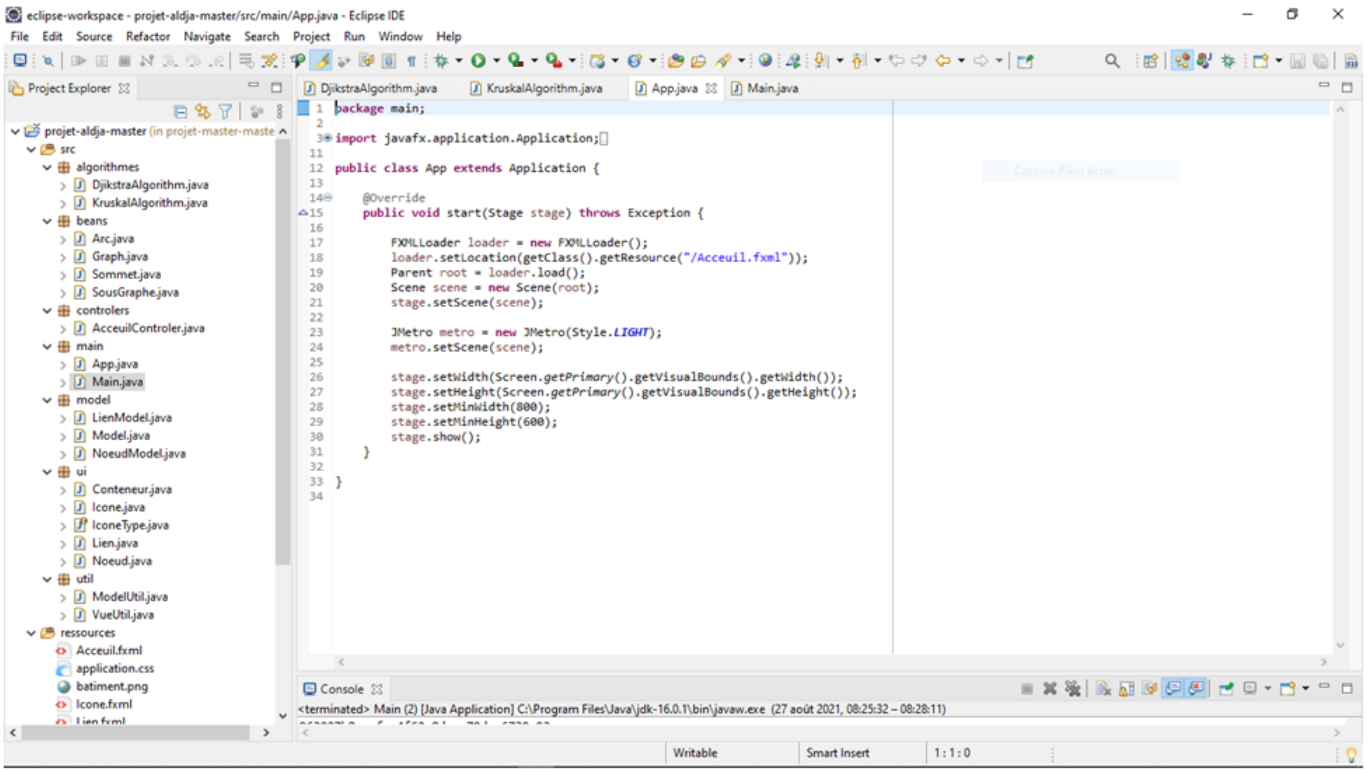
4.4 Présentation de Langage utilisé « JAVA »

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisé pour le développement d'applications d'entreprises et mobiles.

4.5 Quelques capteurs de code source

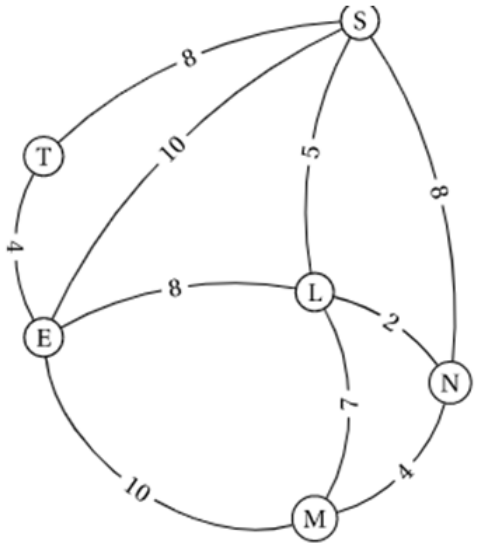




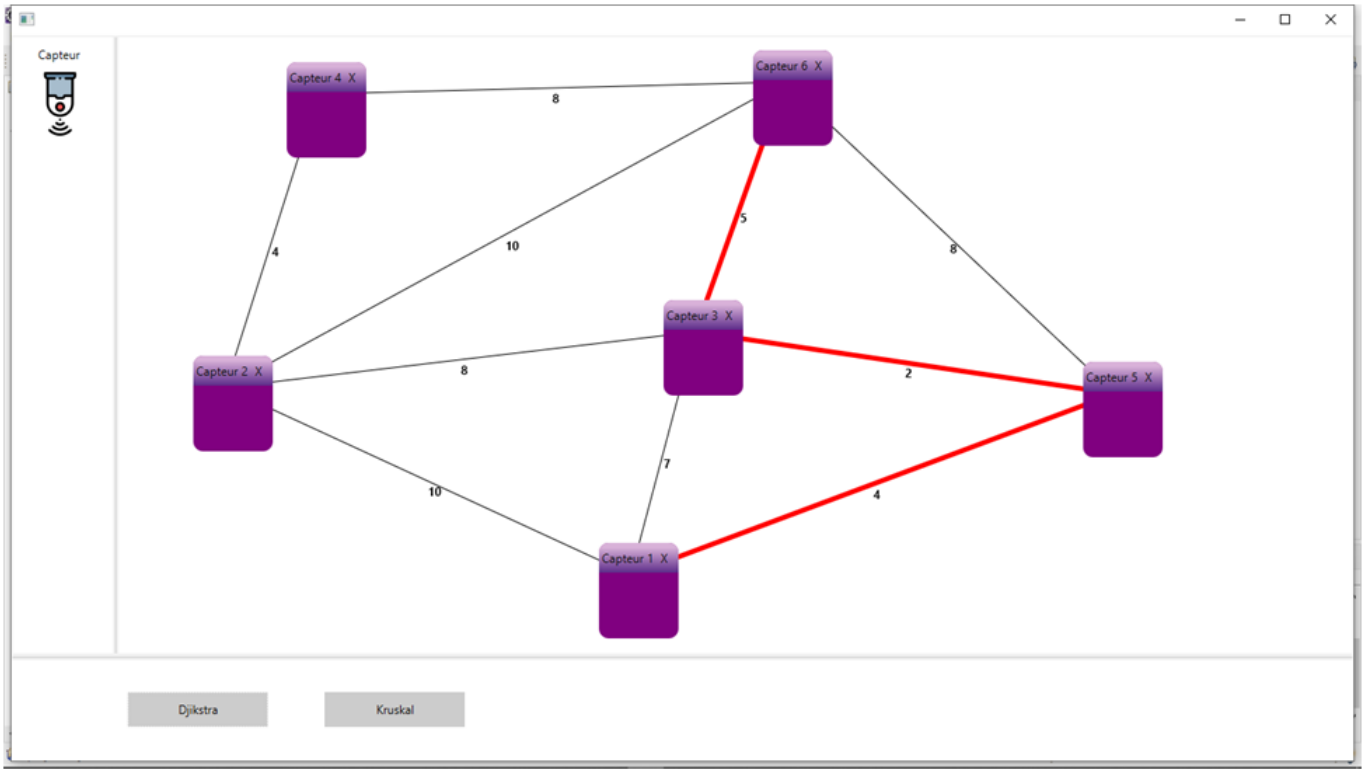


4.6 Cas d'optimisation d'énergie :

- **Application de notre implémentation au réseau ci-dessous**

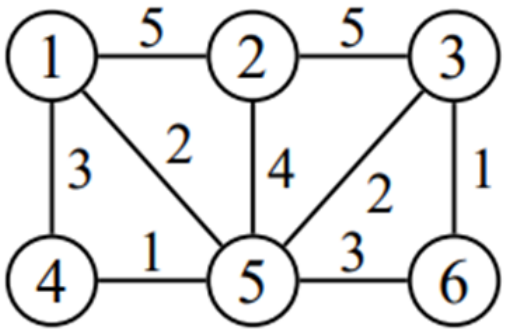


Résolution avec l'application, on obtient le résultat suivant

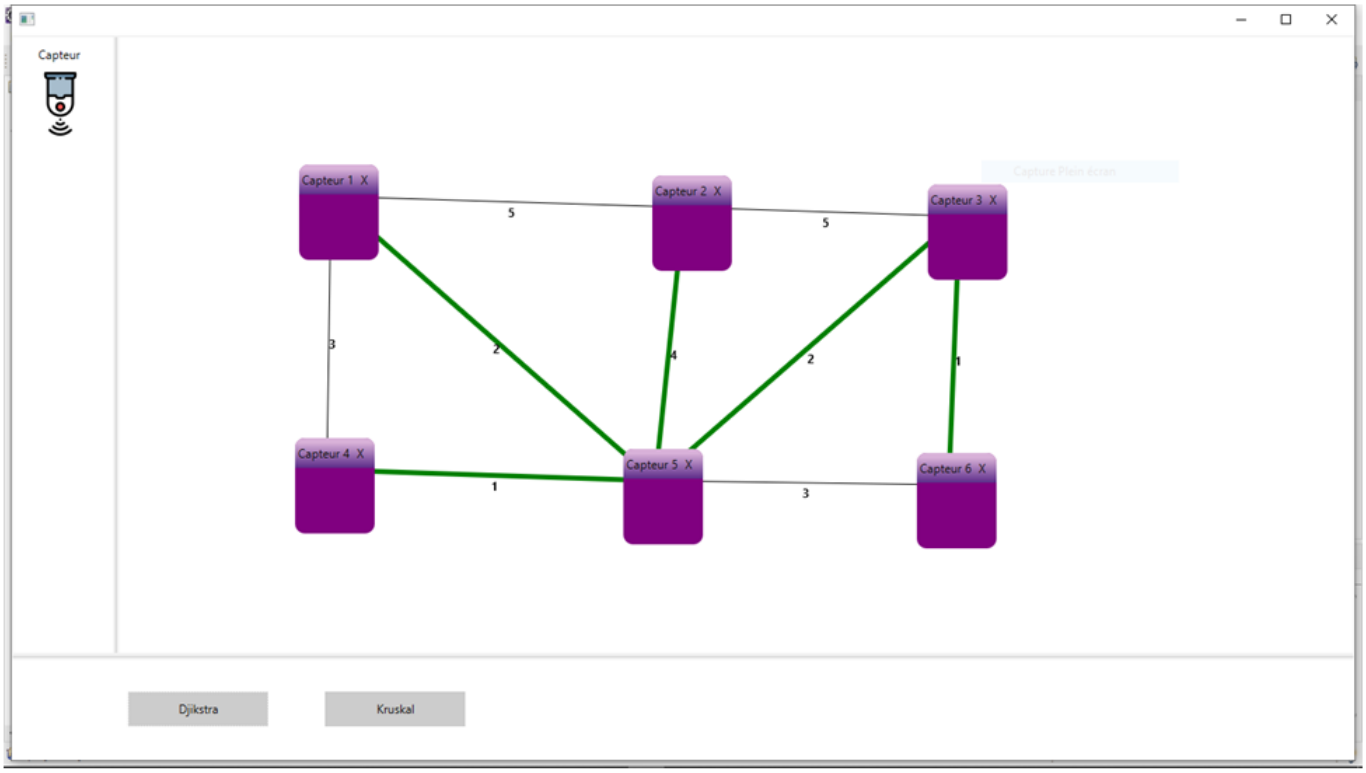


Soit le nœud S présente par le capteur « 6 » et le nœud M par le capteur « 1 » et ainsi de suite, voyons qu'on a eu les mêmes résultats, et la plus petite distance égale à 11.

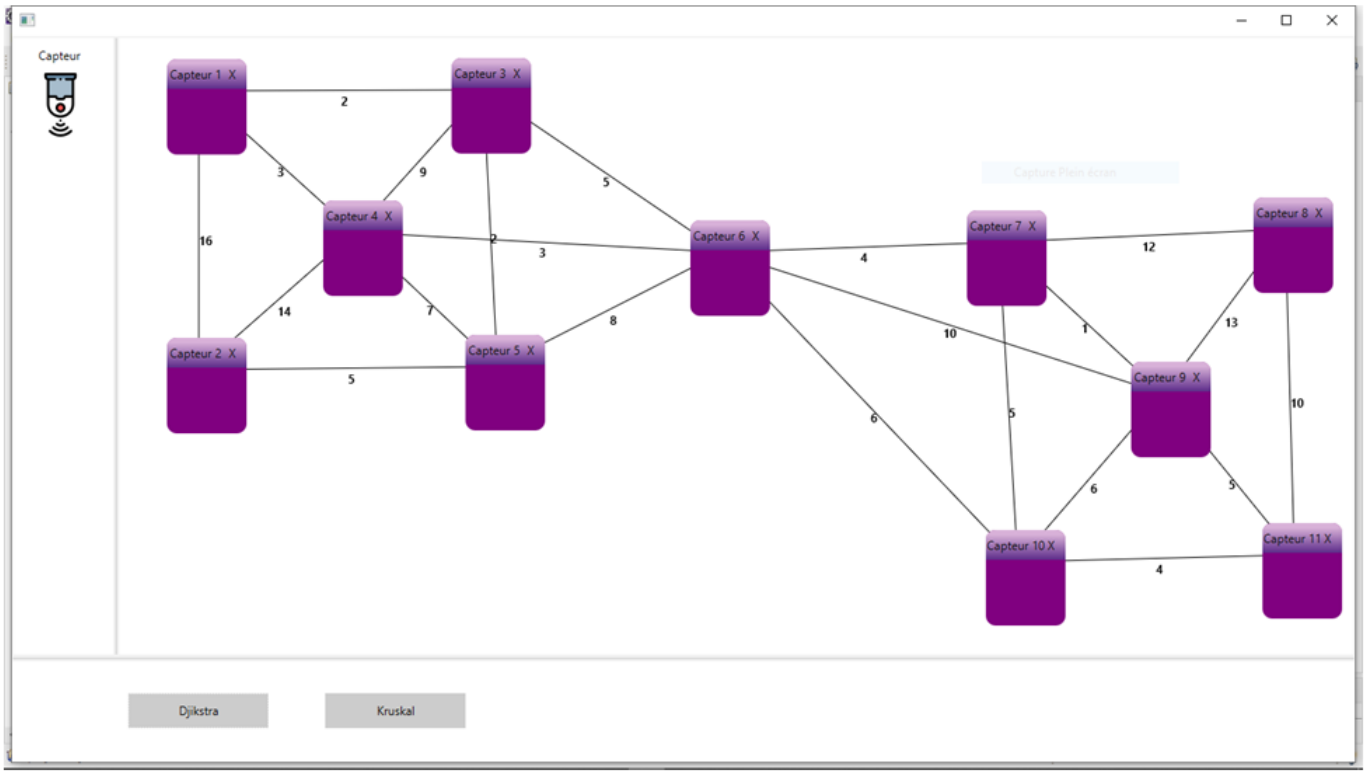
- **Application de notre implémentation au réseau ci-dessous**



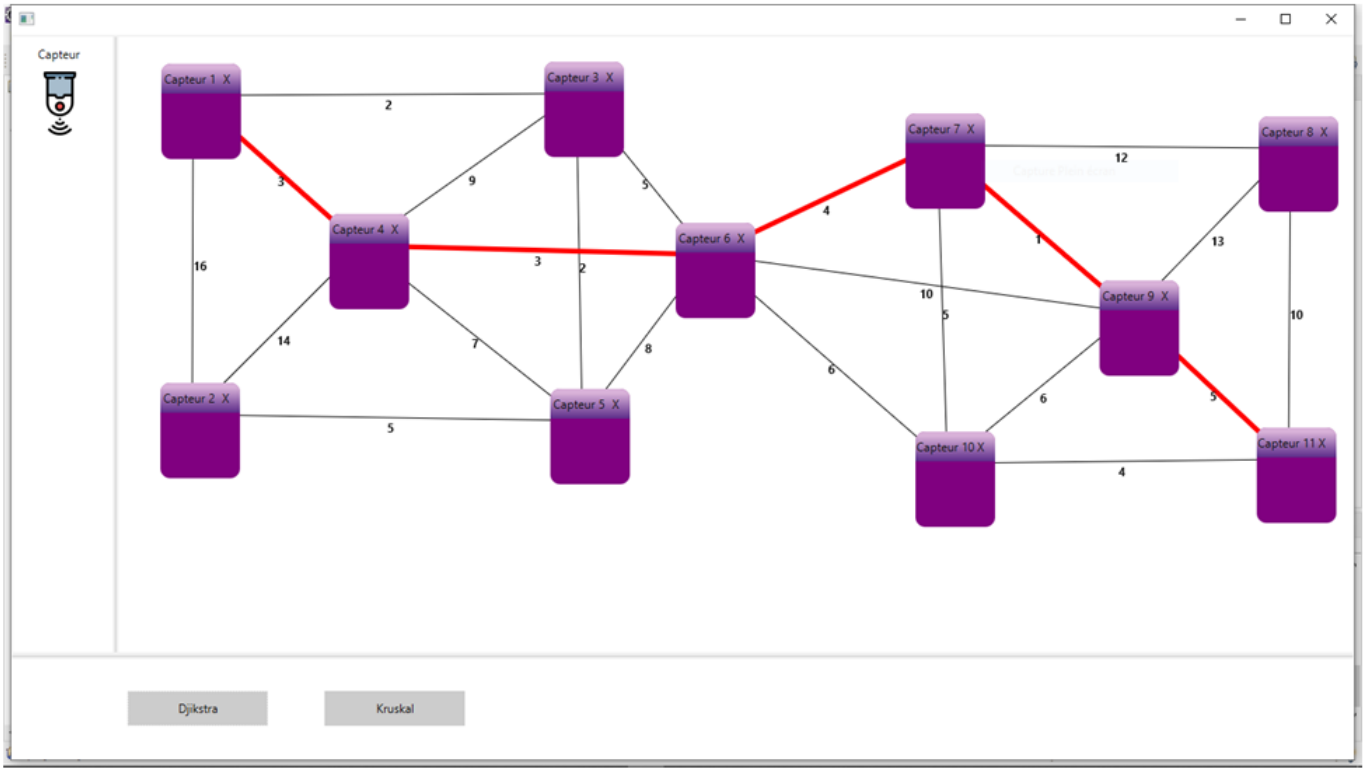
Illustrons l'exemple avec l'application pour construire l'arbre couvrant de poids minimum de graphe : on obtient l'arbdre couvrant de poids minimum comme le montre la figure suivante :



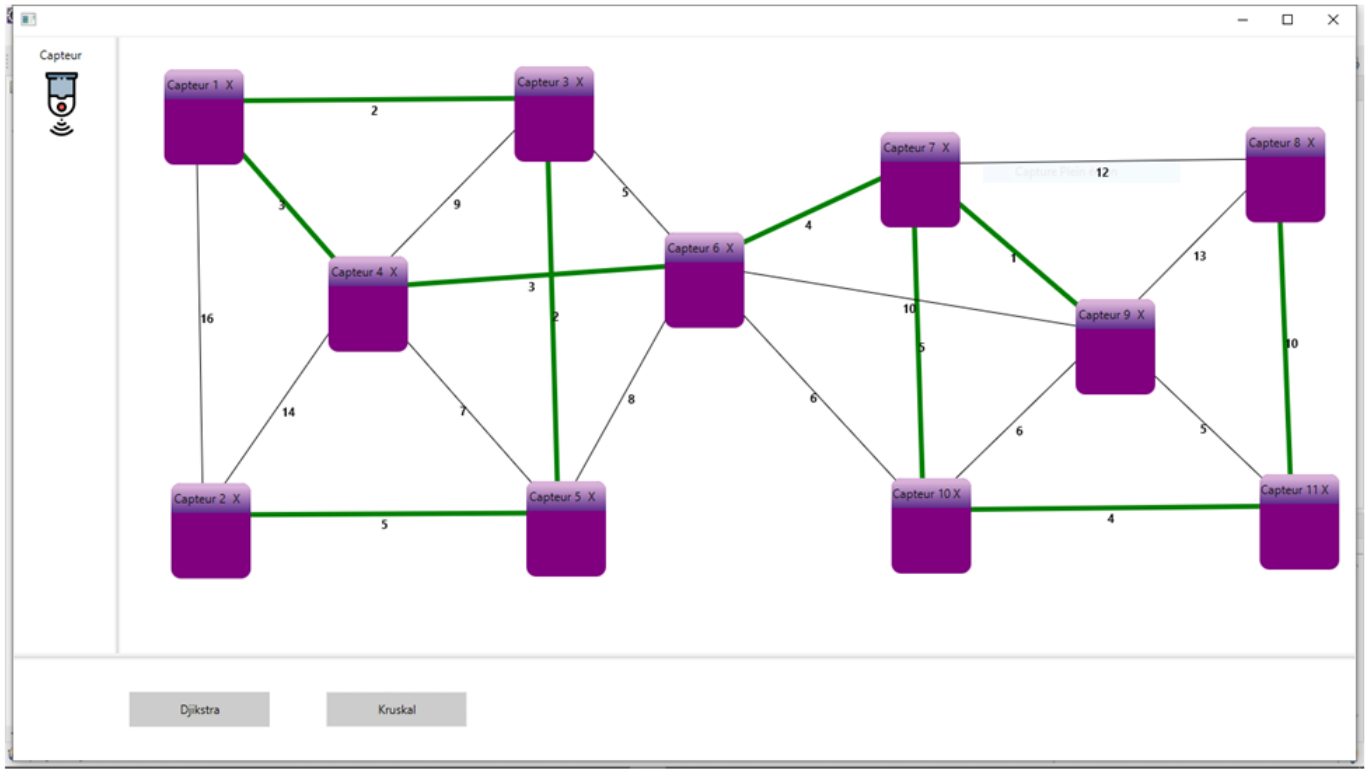
Traitons maintenant l'exemple suivant avec les deux algorithmes Djikstra et Kruskal, pour définir la plus petite distance entre le capteur « 1 » et le capteur « 11 », illustrer sur cette figure :



Le résultat avec l'application de l'algorithme de Djikstra est :



Le résultat avec l'application de l'algorithme de Kruskal est :



4.7 Conclusion

Ce dernier chapitre est consacré à la modélisation d'un réseau de capteur sans fil par un graphe, sachant que plus le nœud source s'éloigne du nœud puit, plus de consommation d'énergie, et dans le contexte de l'étude de l'énergie et de la façon de maintenir la durée de vie du réseau, on propose une programmation de l'algorithme de Dijkstra sur le langage Java qui calcule le meilleur chemin, ayant un poids minimal, entre la source et la destination comme une solution, et de l'algorithme de Kruskal qui construit l'arbre couvrant de poids minimum de graphe.

Résumé

Le réseau de capteur sans fil (“WSN : Wireless Sensor Network” en Anglais) est un réseau Ad hoc composé d'un grand nombre de nœuds qui sont des micro-capteurs qui peuvent être déployés de façon aléatoire ou déterministe dans une zone d'intérêt donnée. Ces nœuds capteurs sont capables de récolter plusieurs paramètres physiques sur l'environnement qui les entoure, appelé généralement zone de captage (ou zone de surveillance). Ensuite, ils doivent si nécessaire traiter les données capturées et les transmettre à un (ou plusieurs) nœud de collecte appelé station de base, centre de traitement (“sink” en Anglais). Beaucoup de domaines d'applications tels que le contrôle et suivi environnemental, le contrôle de production dans l'industrie, la surveillance de zone, le monitoring de l'habitat, l'agriculture intelligente, etc. sont basés sur les RCSF.

L'allongement de la durée de vie du réseau constitue un grand défi dans l'utilisation d'un RCSF à cause de la miniaturisation des capteurs nécessite des mécanismes de conservation d'énergie de ces derniers afin d'étendre la durée de vie du réseau. En effet notre travail est constitué de 4 chapitre, le premier chapitre a pour but de définir les concepts de base de la théorie des graphes, le deuxième chapitre porte sur des généralités sur les réseaux de capteur sans fil et leur application, et dans le chapitre 3 nous avons définit Quelques Techniques d'optimisations dans les réseaux de Capteurs, et en dernier chapitre nous avons présenté l'environnement de travail Eclipse et notre application codé avec Java des deux algorithmes Djikstra et Kruskl en traitant quelques exemples.

Mots clés : Réseau de capteurs sans fil (RCSF), optimisation d'énergie, routage, domaine d'application.

Abstract

Wireless Sensor Network (Wireless Sensor Network) is an Ad hoc network made up of a large number of nodes which are micro-sensors that can be deployed randomly or deterministically in a given area of interest. These sensor nodes are able to collect several physical parameters from the environment around them, generally called a catchment area (or monitoring area). Then, if necessary, they must process the captured data and transmit them to one (or more) collection node called base station, processing center ("sink"). Many application areas such as environmental monitoring and control, industrial production control, area surveillance, habitat monitoring, smart agriculture, etc. are based on WSN.

Extending the life of the network is a big challenge in the use of a SCN due to the miniaturization of the sensors requires energy conservation mechanisms of the latter in order to extend the life of the network. Indeed our work consists of 4 chapters, the first chapter aims to define the basic concepts of graph theory, the second chapter covers generalities on wireless sensor networks and their application, and in the chapter 3 we have defined Some Optimization Techniques in Sensor Networks, and in the last chapter we presented the Eclipse working environment and our application coded with Java of the two algorithms Djikstra and Kruskl by treating some examples.

Keywords : Wireless Sensor Network (WSN), energy optimization, routing, application area.

Conclusion générale

Dans ce travail, nous nous sommes intéressés à l'Optimisation dans les réseaux de capteurs sans fil. On a pu constater que les graphes constituent une méthode de pensée qui permet de modéliser une grande variété de problèmes concrets en se ramenant à l'étude de sommets et d'arcs. La théorie des graphes permet de générer des circuits optimisés et de gérer des réseaux (routiers, de communication, de transport, d'eau de gaz, . . .), d'ordonnancer des tâches et de gérer des plannings. Elle est la clé de l'intelligence artificielle avec la notion du " plus court chemin ". Ces nombreuses applications font de la théorie des graphes un outil appréciable d'aide à la décision (en recherche opérationnelle), en apparence, sa mise en œuvre est simple et ludique, voire enfantine.

Dans un premier lieu, nous avons introduit ce type de réseau, en particulier, nous avons présente l'architecture, les caractéristiques, ainsi que les domaines d'applications de ce genre de réseau. Il a été constaté que la recherche dans les réseaux de capteurs est beaucoup plus orientée vers la conservation de l'énergie.

Notre but est de prolonger la durée de vie du réseau de capteurs en minimisant la consommation d'énergie, pour cela De nombreux algorithmes, et protocoles ont été proposés dans la littérature pour traiter les problématiques de la minimisation de l'énergie dissipée par les capteurs on a implémenter un protocole de routage basée sur les algorithmes de djikIstra et kruskal pour trouver le meilleur chemin ayant un poids minimal entre la source et la destination.

Dans un dernier lieu on a élaboré une application pour la résolution du problème de telle sorte qu'elle soit aussi convivial que possible, sans perdre de vue son aspect pratique et encore moins sa performance, muni d'une interface claire et accessible, facilitant ainsi son utilisation. Finalement, notre travail consiste à étudier et implémenter de façon plus approfondie les algorithmes les plus adapté sous éclipse, afin de trouver la solution la plus adaptée pour l'utilisateur.

Bibliographie

[1] Diery NGOM (Mai 2016) Optimisation de la durée de vie dans les réseaux de capteurs sans fil sous contraintes de couverture et de connectivité, Informatique, 'Université de Haute Alsace (France) et de l'Université Cheikh Anta Diop de Dakar (Sénégal). Page 12-31
[2] KECHAR Bouabdellah, (2010), problématique de la consommation d'énergie dans les réseaux de capteurs sans fil,[2] KECHAR Bouabdellah, (2010), problématique de la consommation d'énergie dans les réseaux de capteurs sans fil, CAO/IAO et simulation. Page 12
[3] FARES Abdelfatah, (2008), développement d'une bibliothèque de capteurs, Master Informatique, France, Université de MONTPELLIER 2. Page 14-20-28-29-36
[4] Didier Müller, Introduction à la théorie des graphes. Page 36 [5] Algorithme de Dijkstra - Étape par étape - Maths-cours
[6] Théorie des graphes et algorithmes - Algorithme de Kruskal (unit.eu)
[7] Kaci Bader.Détectiond'intrusionsdanslesréseauxdecapteurssansfil.Cryptographieetsécurité [cs.CR]. 2010.page 33
[8] N.Mekki, K.Mohammed : Techniques de conservation d'énergie pour les réseaux de capteur sans fil, MEMOIRE DE MASTER OPTION : RISR, UNIVERSITE Dr. TAHAR MOULAY SAIDA, 2018.
[9] Dominique Dhoutaut. Etude du standard IEEE 802.11 dans le cadre des réseaux Ad hoc : de la simulation à l'expérimentation. Thèse pour obtenir le grade de Doctorat en Informatique. 2003.
[10] S. Singh, M. Woo, and C.S. Raghavendra. Power-aware routing in mobile ad hoc networks. ACM, MOBICOM 98, pages 181-190, 1998.
[11] V. Raghunathan, C Shurgers, S. Park, and M.B. Srivastava.Energy-aware wireless microsensor. IEEE Signal Process. Mag, 19(2) :40-50, May 2002.
[12] E. Shih and al.physical layer driven protocol and algorithm design for energy-efficient wireless. ACM/IEEE MOBICOM, pages 272-287, July 2001.
[13] M. Ilyas and I. Mahgoub.Handbook of sensor Network Compact Wireless and Wired Sesing System. Number 0-8493-1968-4. CRC PRESS LLS, USA, 2005.
[14] F.Zhao and Al.Collaborative signal and information processing : an information directed approach. IEEE, 8 :1199-1209, 2003.
[15] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. ACM, 43(5) : 51-58, May 2000.
[16] P. Santi, Topology control in wireless Ad Hoc and Sensor Networks. Hard-cover, July 2005. Z"
[17] https ://perso.liris.cnrs.fr/samba-ndojh.ndiaye/fichiers/App-Graphes.pdf . page 12-19-20
[18] http ://www.lirmm.fr/ leclere/enseignements/TER/2008/Rapport/19.pdf
[19] http ://igm.univ-mlv.fr/ alabarre/teaching/graphes/chap03-graphes-ponderes.pdf. page 26

[21] http ://www.uqac.ca/rebaine/8INF805/coursgraphe01.pdf
[22] https ://www.apprendre-en-ligne.net/graphes/graphes.pdf. page 13-18-21
[23] https ://dumas.ccsd.cnrs.fr/dumas-00530725/document
[24] https ://perso.liris.cnrs.fr/christine.solnon/polyGraphes.pdf
[25] O.Feyrouz, M.Sarah (2019) : Minimisation de l'énergie dans les réseaux de capteurs sans fil. page 11-12
[26] Optimisation dans les réseaux, Juin 2016.page 16-18
[27] Quelques méthodes d'optimisation et application dans les graphes,(2015). page 24

Résumé

Le réseau de capteur sans fil (“WSN : Wireless Sensor Network” en Anglais) est un réseau Ad hoc composé d’un grand nombre de nœuds qui sont des micro-capteurs qui peuvent être déployés de façon aléatoire ou déterministe dans une zone d’intérêt donnée. Ces nœuds capteurs sont capables de récolter plusieurs paramètres physiques sur l’environnement qui les entoure, appelé généralement zone de captage (ou zone de surveillance). Ensuite, ils doivent si nécessaire traiter les données capturées et les transmettre à un (ou plusieurs) nœud de collecte appelé station de base, centre de traitement (“sink” en Anglais). Beaucoup de domaines d’applications tels que le contrôle et suivi environnemental, le contrôle de production dans l’industrie, la surveillance de zone, le monitoring de l’habitat, l’agriculture intelligente, etc. Sont basés sur les RCSF.

L’allongement de la durée de vie du réseau constitue un grand défi dans l’utilisation d’un RCSF à cause de la miniaturisation des capteurs nécessite des mécanismes de conservation d’énergie de ces derniers afin d’étendre la durée de vie du réseau. En effet notre travail est constitué de 4 chapitre, le premier chapitre a pour but de définir les concepts de base de la théorie des graphes, le deuxième chapitre porte sur des généralités sur les réseaux de capteur sans fil et leur application, et dans le chapitre 3 nous avons défini quelques Techniques d’optimisations dans les réseaux de Capteurs, et en dernier chapitre nous avons présenté l’environnement de travail Eclipse et notre application codé avec Java des deux algorithmes Djikstra et Kruskl en traitant quelques exemples.

Mots clés : Réseau de capteurs sans fil (RCSF), optimisation d’énergie, routage, domaine d’application.

Abstract

Wireless Sensor Network (Wireless Sensor Network) is an Ad hoc network made up of a large number of nodes which are micro-sensors that can be deployed randomly or deterministically in a given area of interest. These sensor nodes are able to collect several physical parameters from the environment around them, generally called a catchment area (or monitoring area). Then, if necessary, they must process the captured data and transmit them to one (or more) collection node called base station, processing center (“sink”). Many application areas such as environmental monitoring and control, industrial production control, area surveillance, habitat monitoring, smart agriculture, etc. are based on WSN.\\

Extending the life of the network is a big challenge in the use of a SCN due to the miniaturization of the sensors requires energy conservation mechanisms of the latter in order to extend the life of the network. Indeed our work consists of 4 chapters, the first chapter aims to define the basic concepts of graph theory, the second chapter covers generalities on wireless sensor networks and their application, and in the chapter 3 we have defined Some Optimization Techniques in Sensor Networks, and in the last chapter we presented the Eclipse working environment and our application coded with Java of the two algorithms Djikstra and Kruskl by treating some examples.

Keywords: Wireless Sensor Network (WSN), energy optimization, routing, application area.