Rapport de projet 3A

Passage à l'échelle d'un réseau LoRaWAN: De la validation du modèle à l'optimisation topologique

Travail encadré par Laurent Ciarletta

réalisé par
Tom Mourot-Faraut

Table des matières

Introduction

Selon de nombreuses prévisions, environ 50 Milliards d'objets devraient être connectés d'ici à 2022. un tiers de ces objets continueront d'utiliser les protocoles GSM actuels, un tiers utiliseront des réseaux à très courte portée, notamment dans des usages domotiques et un tiers devraient être connectés à des réseaux bas débit et longue portée, soit un marché total adressable potentiel de 15 milliards d'objets connectés. Ces réseaux bas débit, longue portée et faible consommation sont les réseaux LPWAN pour Low Power Wide Area Network qui sont un ensemble de réseaux émergents très utilisés dans l'IoT permettant la communication de milliers d'objets connectés et qui jouent donc un grand rôle dans le développement futur de smart cities mais aussi dans le monitoring industriel ou encore l'agriculture. Cette technologie permet d'échanger des messages de très faible débit sur de longues portées pour une faible consommation électrique, permettant ainsi une autonomie de ces objets de plusieurs années. Les deux technologies les plus connues aujourd'hui sont SigFox et LoRa. SigFox dont la différence majeure avec son concurrent est son caractère propriétaire alors que LoRa est un réseau ouvert permettant à chacun de le développer et de l'exploiter du moment qu'il possède une puce homonyme.

L'objectif de ce projet est d'étudier le passage à l'échelle d'un tel réseau, c'est-à-dire, déterminer les facteurs qui permettent de conserver le bon fonctionnement d'un réseau LoRa lorsque le nombre de noeuds augmente.

1. Les réseaux LoRaWAN et le simulateur LoRaSIM

1.1 LoRaWAN

LoRaWAN est l'acronyme de Long Range Wide-area network que l'on peut traduire par « réseau étendu à longue portée ». Il s'agit d'un protocole de télécommunication de plus en plus utilisé dans le monde de l'IoT et qui représente une solution intéressante pour l'émergence des smart cities. Ce protocole utilise une technique de modulation par étalement de spectre de type Chirp Spread Spectrum. Il permet une communication peu énergivore et à bas débit par ondes radio entre objets connectés. La technologie de modulation liée à LoRaWAN est LoRa, née à la suite de l'acquisition de la startup grenobloise Cycléo par Semtech en 2012. Semtech promeut sa plateforme LoRa grâce à la LoRa Alliance, dont elle fait partie. Chacun des objets connectés communique directement avec une passerelle (sink node), ainsi, le fait qu'une grande superficie soit couverte et que le réseau dispose de peu de passerelles fait que les objets doivent partager un même medium ce qui induit des problèmes de collisions. Cependant, afin de palier à ce problème LoRa fait en sorte que chaque émetteur dispose de 5 paramètres de communication qui sont :
Puissance de transmission (TP) : entre -4dBm et 20dBm
Fréquence porteuse (CF) : entre 137MHz et 1020MHz
Facteur d'étalement (SF) : Le rapport entre le chip rate et le symbole rate. Il varie entre 5 et 12
Bande passante (BW) : Un réseau LoRa opère généralement avec une bande passante de 125, 250 ou 500kHz
Taux de codage (CR) : La proportion du flux de données utile c'est-à-dire non redondant. Il permet de rendre le signal plus robuste. Sa valeur peut être 4/5, 4/6, 4/7 ou 4/8.

Les multiples combinaisons possibles de ces paramètres peuvent être orthogonales et peuvent ainsi permettre des communications sans collisions. Cependant il existe bien évidemment des limites au nombre de communications qu'un réseau LoRa peut supporter en fonction du nombre de noeuds, du nombre de passerelles ainsi que du nombre de communications par heure. Etant données ces 3 données, il est possible d'améliorer le bon fonctionnement d'un réseau LoRa en modifiant dynamiquement les paramètres de communication de chaque noeud ou alors en optimisant le placement des passerelles, c'est-à-dire la topologie du réseau.

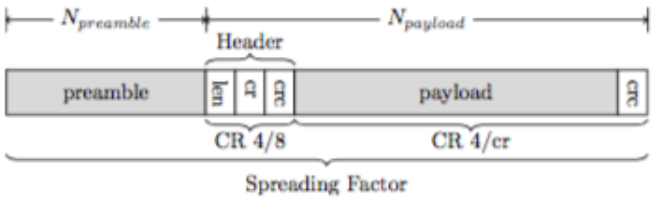

FIGURE 1 – Structure d'un paquet LoRa

1.2 Simulation d'un réseau LoRaWAN : LoRaSim

Etant donné les coûts d'infrastructure pour mettre en place un réseau LoRa complet comprenant des passerelles et des centaines de noeuds, l'utilisation d'un simulateur pour étudier le passage à l'échelle est nécessaire.
SimPy est un framework de simulation d'évènements discrets, développé par le MIT, basé sur python et disponible en tant que logiciel open source sous license qui permet d'implémenter des systèmes multi-agents. LoRaSim est un simulateur d'événements discrets basé sur SimPy dont le rôle est de simuler des collisions dans un réseau LoRa afin d'en analyser le passage à l'échelle.
LoRaSim a été développé par une équipe de recherche de l'université de Lancaster fin 2016 dans le cadre de recherches sur le passage à l'échelle des réseaux LoRa. Il est composé de 4 scripts python : un simulateur avec une seule passerelle, un avec plusieurs passerelles et enfin deux autres concernant l'utilisation d'antennes directionnelles. Dans le cadre de notre travail sur l'optimisation topologique, le script qui nous intéresse est celui qui simule un réseau LoRa possédant plusieurs passerelles : loraDirMulBs.py. Le simulateur permet de placer N noeuds, ainsi que M passerelles dans une grille à 2 dimensions qui représente un quartier. Dans le code originel, le placement des M passerelles est fixe et ne dépend pas de la topologie du réseau, c'est-à-dire des positions des noeuds.
Dans notre cas d'utilisation on choisit de fixer les paramètres de communication de la manière récapitulée par le tableau suivant :

Parameter	SN^1
TP (dBm)	14
CF (MHz)	868
SF	12
BW (kHz)	125
CR	4/8

FIGURE 2 – Paramètres de communication utilisés pour l'étude de l'optimisation topologique

Ces paramètres correspondent à ceux utilisés dans la plupart des réseaux LoRa et ont la particularité de rendre les signaux particulièrement robustes et de ce fait augmentent leur temps d'émission. Par ailleurs, dans notre simulation chaque paquet est composé de 20 octets.
Pour simuler le fonctionnement d'un réseau LoRa, nous allons modéliser l'atténuation d'un signal en fonction de la distance avec le modèle log-distance path loss. Une transmission est bien reçue si la puissance du signal émis Prx dépasse le seuil de sensibilité du récepteur Srx.

$$Prx = Ptx + Gtx - Ltx - Lpl - Lm + Grx - Lrx \tag{1}$$

Prx est la puissance reçue en dB, Ptx la puissance émise en dB, Gtx est le gain d'antenne de l'émetteur en dBi, Ltx est les pertes de l'émetteur en dB, Lpl est l'affaiblissement de propagation (path loss en anglais) en dB et qui dépend de l'environnement, Lm représente les pertes diverses en dB, Grx est le gain d'antenne du récepteur en dBi et Lrx représente les pertes du récepteur en dB.
Afin de pouvoir modéliser le fonctionnement du réseau LoRa on simplifie l'équation de la

façon suivante en nommant GL la somme de tous les gains et pertes :

$$Prx = Ptx + GL - Lpl \tag{2}$$

Pour les besoins de notre étude on fixe GL = 0.
De nombreux modèles existent pour décrire l'affaiblissement de propagation mais le modèle log-distance path loss convient habituellement pour les environnements densément peuplés, ce qui est cohérent dans le cas de l'étude d'une smart city. L'équation du modèle est la suivante :

$$Lpl(d) = Lpl(d0) + 10\gamma log(d/d0) \tag{3}$$

Avec Lpl(d) l'affaiblissement de propagation en dB, Lpl(d0) l'affaiblissement de propagation à la distance de référence d0, et γ le facteur d'affaiblissement de propagation. Pour notre simulation les valeurs utilisées sont d0 = 40m, Lpl(d0) = 127.41dB et γ = 2.08.

D'après la documentation de Semtech, le comportement du récepteur peut-être résumé par l'équation du seuil de réception suivante :

$$Srx = -174 + 10log(BW) + NF + SNR \tag{4}$$

Le premier terme désigne le bruit thermique et ne dépend que de la température du récepteur. NF est le facteur de bruit qui dépend du hardware et SNR et le rapport signal sur bruit (signal to noise ratio) et est proportionnel au SF.

	Bandwidth (kHz)		
SF	125	250	500
7	−126.50	−124.25	−120.75
8	−127.25	−126.75	−124.00
9	−131.25	−128.25	−127.50
10	−132.75	−130.25	−128.75
11	−134.50	−132.75	−128.75
12	−133.25	−132.25	−132.25

FIGURE 3 – Seuils de sensibilités du récepteur en fonction des paramètres BW et SF

Le tableau de données précédent nous donne les valeurs mesurées du seuil de réception en fonction de BW et SF. Pour nos simulations, nous utilisons SF = 12 et BW = 125kHz, ainsi notre seuil de réception est de -133.25.

Pour résumer, la condition à vérifier afin de s'assurer qu'un noeud puisse communiquer avec une passerelle est la suivante :

$$R = \begin{cases} 1, & Prx > Srx, \\ 0, & else \end{cases}$$

2. Vérification du modèle

Le modèle log-distance path loss et donc la portée maximale de communication dépendent fortement de l'environnement et il n'est pas évident que les valeurs utilisées dans le modèle soient les mêmes quel que soit l'environnement. Dans cette partie, nous allons donc tenter de valider le modèle de la portée de communication dans un réseau LoRa décrit dans la partie précédente.
Pour cela on utilise un matériel similaire à celui utilisé dans le cadre des travaux qui ont permis l'établissement du modèle. Le LORIA dispose d'un module LoPy et un module FiPy de Pycom qui permettent tous les deux l'émission ou la réception de paquets LoRa. Le module LoPy est composé d'un émetteur-récepteur SX1272 de Semtech qui est exactement le même que celui utilisé par l'université de Lancaster.
La validation du modèle se décompose en deux temps :
- Relevé des puissances reçues (Log-distance path loss model) en fonction de la distance entre l'émetteur et le récepteur.
- Détermination du seuil de sensibilité du récepteur.

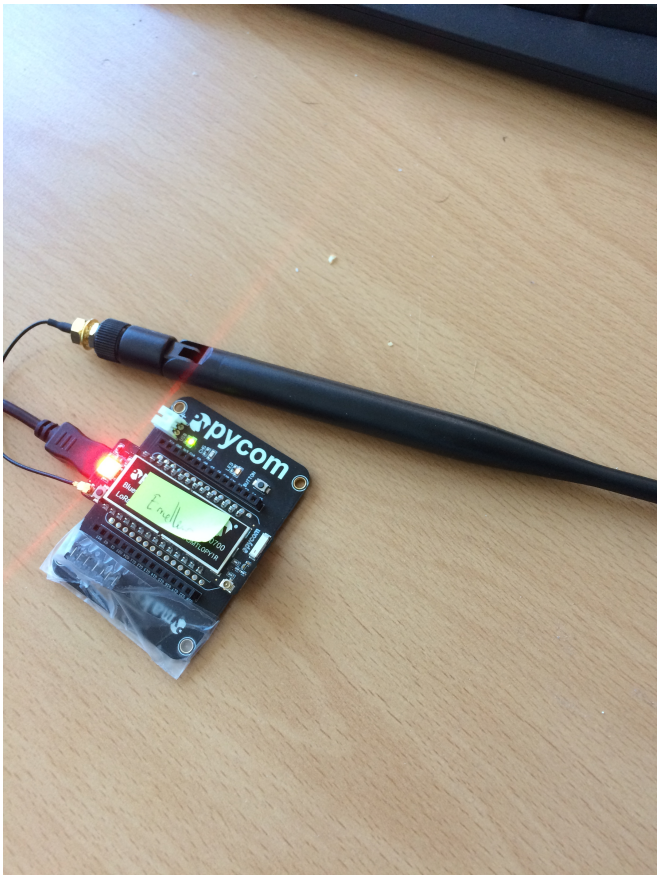

FIGURE 4 – Module LoPy de Pycom avec antenne

FIGURE 5 – Chemin parcouru pour les expérimentations

2.1 Puissance du signal reçu

Dans cette partie on cherche à relever la puissance reçue par le récepteur en fonction de la distance qui le sépare de l'émetteur.
Le protocole expérimental est le suivant :
En fixant TP et en faisant varier la distance on détermine le rssi (Received Signal Strength Indication) en fonction de la distance et on voit si les données sont cohérentes avec le modèle. L'expérience a eu lieu sur le plateau de Brabois qui offre une distance maximale entre le récepteur et l'émetteur de 700m. Etant donné que la distance maximale théorique de communication est 322m, la superficie est en principe suffisante. On fixe les 5 paramètres tels que données dans la figure n°2. Un module GPS dans le module LoPy permet de relever à chaque instant la position du récepteur en même temps qu'il reçoit les paquets de l'émetteur. Le récepteur écrit ainsi chaque couple (position GPS, rssi) dans un fichier .txt.

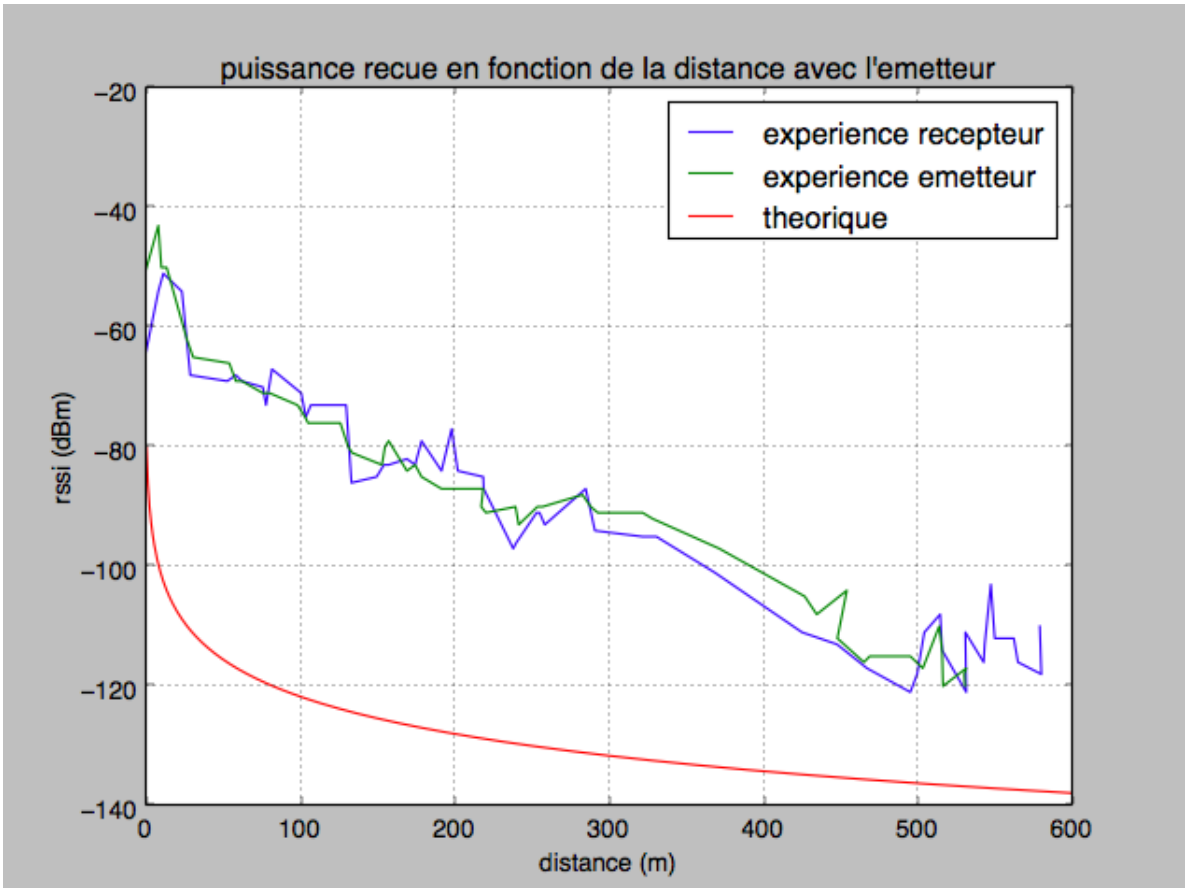

FIGURE 6 – Confrontation du modèle log-distance path loss avec l'expérience

On convertit les données GPS en distances par rapport au point d'origine. Les courbes expérimentales suivent globalement la courbe théorique bien que la puissance reçue soit plus élevée. Cependant l'étendue du plateau de Brabois n'a pas permis d'augmenter suffisamment la distance et de trouver le seuil de sensibilité. Pour cela on a réalisé la même expérience en abaissant la puissance de transmission à 2. Malheureusement, ceci ne nous permettra pas de déterminer expérimentalement si la distance maximale de communication pour TP = 14 est bien de 322m.

Les différences notées entre les deux courbes s'expliquent par le fait que le plateau de Brabois n'a aucun obstacle et ne correspond pas à un environnement densément peuplé où le modèle log-distance path loss conviendrait. Cependant il correspond au terrain d'expérimentation le plus rigoureux compte tenu des spécificités de chaque zone urbaine ayant un grand impact sur le rssi selon les matériaux des bâtiments etc. Une solution aurait été de faire plusieurs expérimentations dans une multitude de zones urbaines et faire une moyenne des résultats mais cela nécessiterait beaucoup plus de temps.

2.2 Seuil de sensibilité

Afin de déterminer expérimentalement le seuil de sensibilité du récepteur nous avons abaissé la puissance de transmission TP de 14 à 2 dans le but de réduire la distance à parcourir avant de trouver le seuil. Ceci nous a permis de déterminer à partir de quelle puissance reçue la connexion était coupée entre l'émetteur et le récepteur.

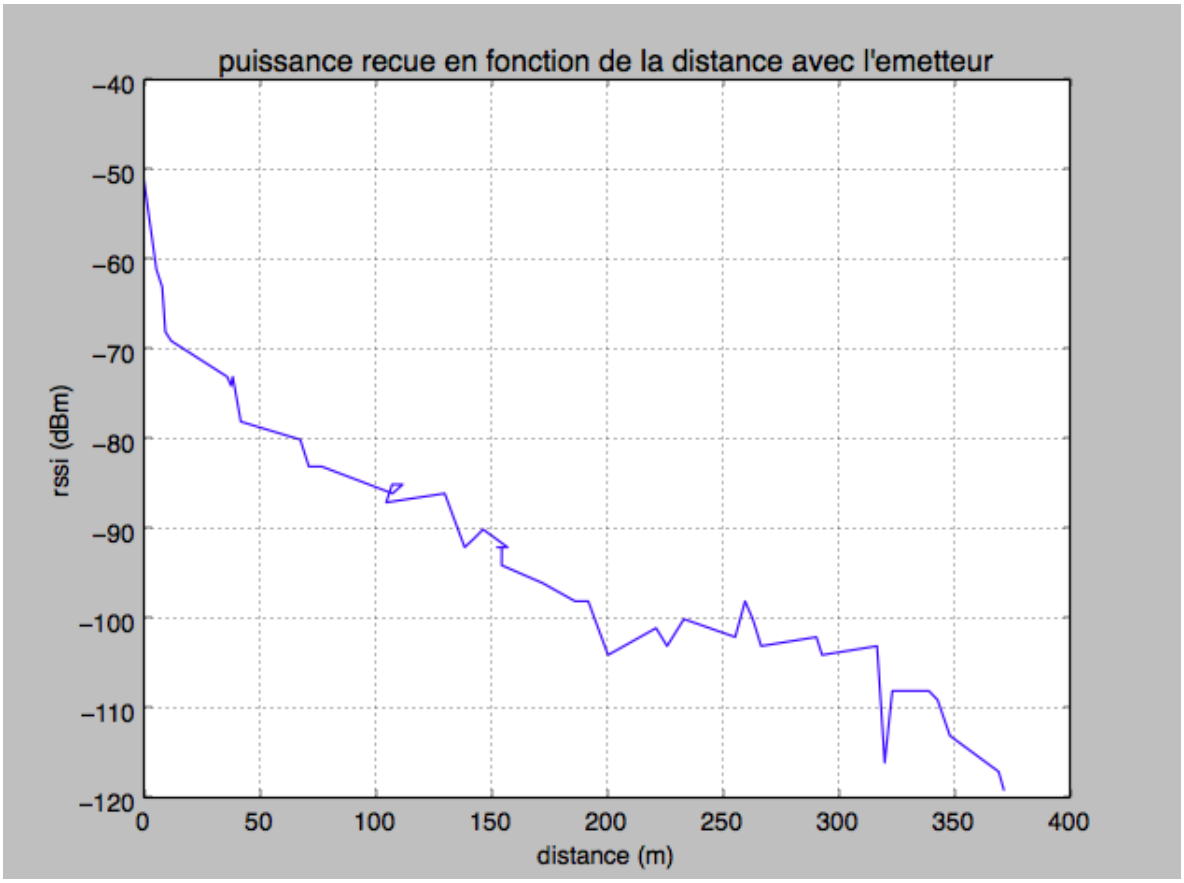

FIGURE 7 – Détection du seuil de réception

L'expérience a permis de montrer que le seuil de réception était d'environ -119 dBm. C'est un seuil de sensibilité plus élevé qu'en théorie qui est de -133 dBm. L'équation (4) montre que cette valeur dépend de l'implémentation du hardware, ce qui explique la différence. Par ailleurs on peut voir que la connexion s'est coupée une fois 370m passés. La distance maximale de communication expérimentale avec TP = 2 dB est plus grand que la distance maximale théorique qui est de 322m, alors que cette valeur est calculée en prenant TP = 14 dB. La différence devrait s'expliquer encore une fois par l'absence d'obstacle sur le plateau de Brabois qui permet une communication à plus longue portée.

3. Optimisation topologique

Dans le but d'améliorer le fonctionnement d'un réseau LoRa, il est possible de jouer sur plusieurs facteurs étant donnés un nombre de noeuds et un nombre de passerelles.

Tout d'abord l'optimisation paramétrique. Chaque communication LoRa est modulée par 5 paramètres qui sont pour rappel : TP, CR, SF, BW et CF. En cas de collision entre deux signaux LoRa, si ces deux signaux ne possèdent pas les mêmes paramètres, il se peut qu'ils soient orthogonaux entre eux. Par exemple deux signaux qui n'ont pas la même fréquence porteuse n'interfèrent généralement pas, de même que deux signaux qui n'ont pas le même facteur d'étalement. La question a été étudiée et des simulations ont été réalisées afin d'étudier le passage à l'échelle d'un réseau LoRa dont les paramètres des noeuds sont dynamiques. Les résultats de la simulation montrent que le passage à l'échelle est satisfaisant mais la réalisation en pratique d'un tel réseau est problématique. En effet l'intervention de network manager est nécessaire afin de garantir le bon paramétrage de chaque noeud à chaque instant et il n'existe pas encore de tel algorithme. Par ailleurs, la simulation ne prend pas en compte la faible robustesse du fait d'un faible CR de certains signaux. Enfin à l'heure actuelle il n'existe qu'un seul ensemble de paramètres utilisés en réalité dans les réseaux LoRa, exceptée la fréquence porteuse qui change en fonction du continent du fait des législations en vigueur.

Vient finalement la solution de l'optimisation topologique. Lorsqu'augmenter le nombre de passerelles est impossible pour raisons budgétaires, dans le but d'augmenter le rapport DER (voir ci-dessous) il convient de trouver l'emplacement optimal de chaque passerelle en fonction de la position des noeuds. On fait cette hypothèse en partant du principe que plus les noeuds sont séparés, moins ils ont de chances d'interférer. En effet, du fait que tous les paramètres sont les mêmes, en cas de chevauchement temporel, on assiste à un chevauchement en fréquence, effet de capture etc. qui vont faire que les signaux vont interférer avec pour résultat qu'aucun paquet ne sera reçu par la passerelle. Afin d'étudier l'influence de l'optimisation de placement des passerelles dans un réseau LoRa, on va exécuter une simulation pour chaque algorithme grâce à l'outil LoRaSim. Afin de mesurer le bon fonctionnement d'un réseau LoRa on utilise le DER pour data extraction rate.

$$DER = \frac{nombre\ de\ paquets\ recus}{nombre\ de\ paquets\ emis} \tag{5}$$

3.1 K-means

Le partitionnement en K-moyennes (K-means) est une méthode de partitionnement de données qui étant données des points et un entier K, divise les points en K groupes (clusters) de façon à minimiser une certaine fonction. La finalité de notre algorithme est de diviser les noeuds en K clusters et de placer une passerelle en leur barycentre. La fonction à minimiser que l'on utilise est la distance euclidienne dans R2. L'algorithme se présente sous la forme suivante :
1 : On sélection K classes S_t
2 : On place les K centres c_t au hasard

3 : On détermine la classe de chaque point en déterminant de quel centre chaque point est le plus proche et donc à quel cluster il appartient. Pour cela on minimise la distance euclidienne

$$t_i = \underset{1 \leq t \leq K}{argmin}(p_i - c_t) \tag{6}$$

4 : Pour chaque cluster, on calcule son barycentre et on déplace le centre de chaque cluster dessus.

$$c_t = \frac{1}{Card(S_t)} \sum_{p_i \in S_t} p_i \tag{7}$$

5 : Si les centres ont bougé, on réitère à partir de l'étape n°3

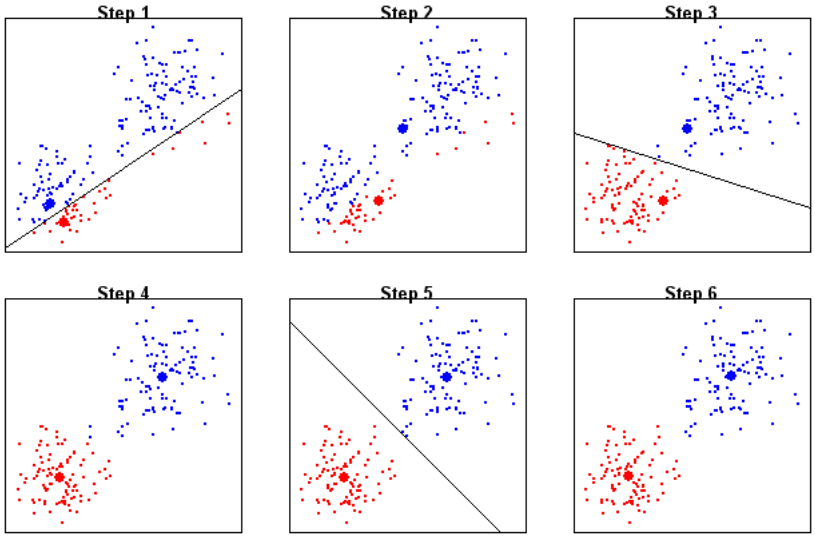

FIGURE 8 – Schématisation de l'algorithme K-means pour K = 2

Cet algorithme a l'avantage d'être relativement simple. Par ailleurs, chaque itération réduit la fonction de coût, ce qui permet d'affirmer que l'algorithme converge toujours en temps fini, bien que ce temps puisse être long du fait que cet algorithme soit gourmand en mémoire. Cependant le temps que prend le code pour placer les passerelles par rapport aux noeuds est non significatif par rapport au temps d'exécution de la simulation qui peut être très long et qui augmente rapidement avec le nombre de noeuds.

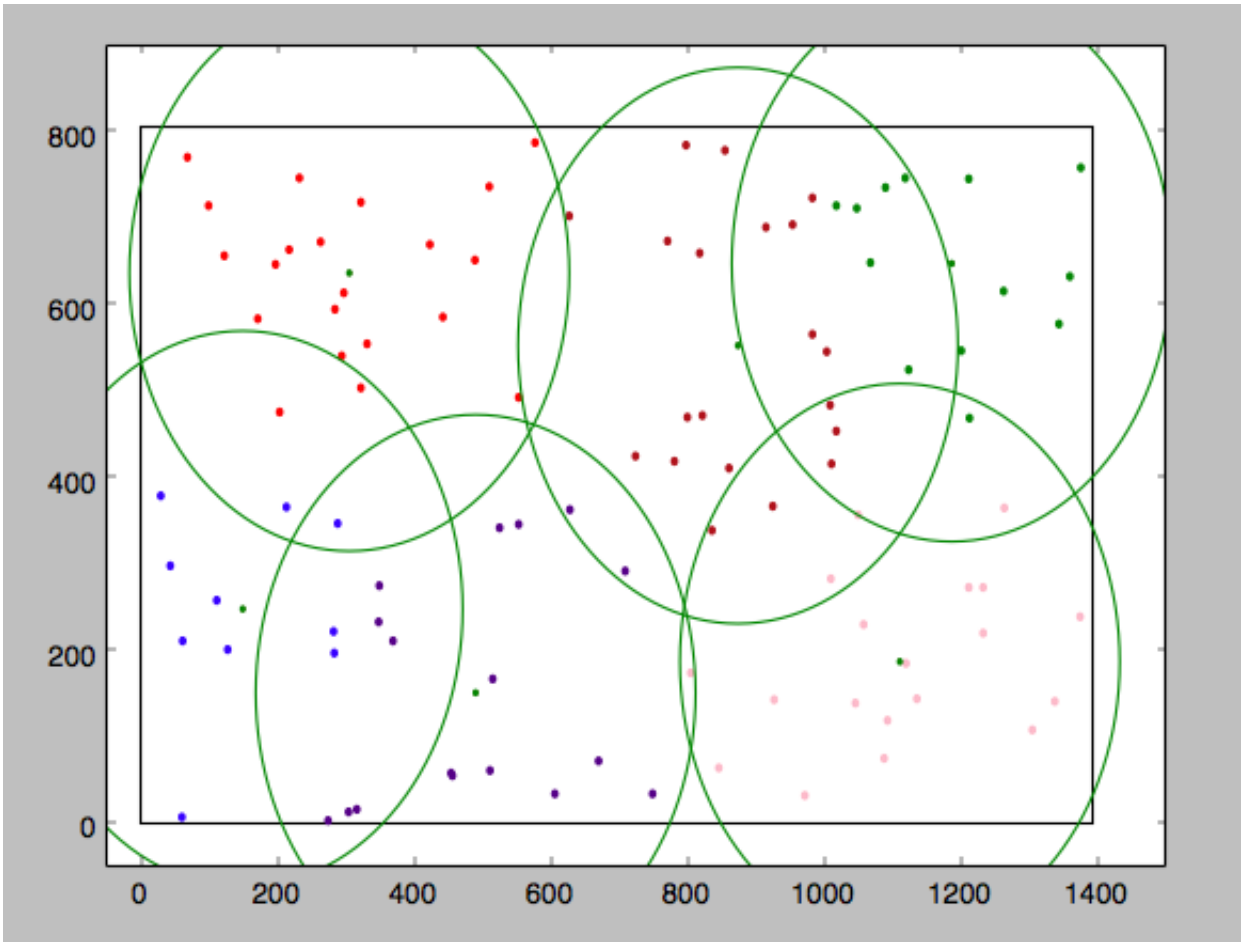

FIGURE 9 – Simulation d'un réseau LoRa avec l'algorithme K-means. nrNodes = 100 et DER = 0.95

En ce qui concerne les réserves sur cet algorithmes, il nécessite d'abord de choisir K a priori. Puisque l'algorithme fournit un minimum local il est assez sensible à l'initialisation : tout dépend du placement originel aléatoire des centres (passerelles). Enfin cet algorithme est sensible aux points aberrants, ce qui nous amène à l'étude du deuxième algorithme utilisé : K-medoids.

3.2 K-medoids

K-medoids est un algorithme de partitionnement dérivé de K-means où la différence est que le centre d'un cluster n'est plus sa moyenne mais un de ses points que l'on appelle medoid. Ce medoid est le point le plus au centre du cluster. L'algorithme se présente sous la forme suivante :
1 : On sélectionne K medoids au hasard parmi l'ensemble des points
2 : On associe à chaque point son medoid le plus proche, ce qui correspond à placer ces points dans des clusters
3 : Pour chaque cluster, on choisit un autre point en tant que nouveau medoid et on vérifie s'il est plus au centre que l'ancien. Pour cela on additionne l'ensemble des distances entre le medoid et les autres points du cluster. Si cette somme est inférieure à la précédente lorsque l'on avait l'ancien medoid, on définit ce medoid comme nouveau medoid sinon on revient en arrière et on ne change rien
4 : Si la somme des distances change on réitère à partir de l'étape n°2

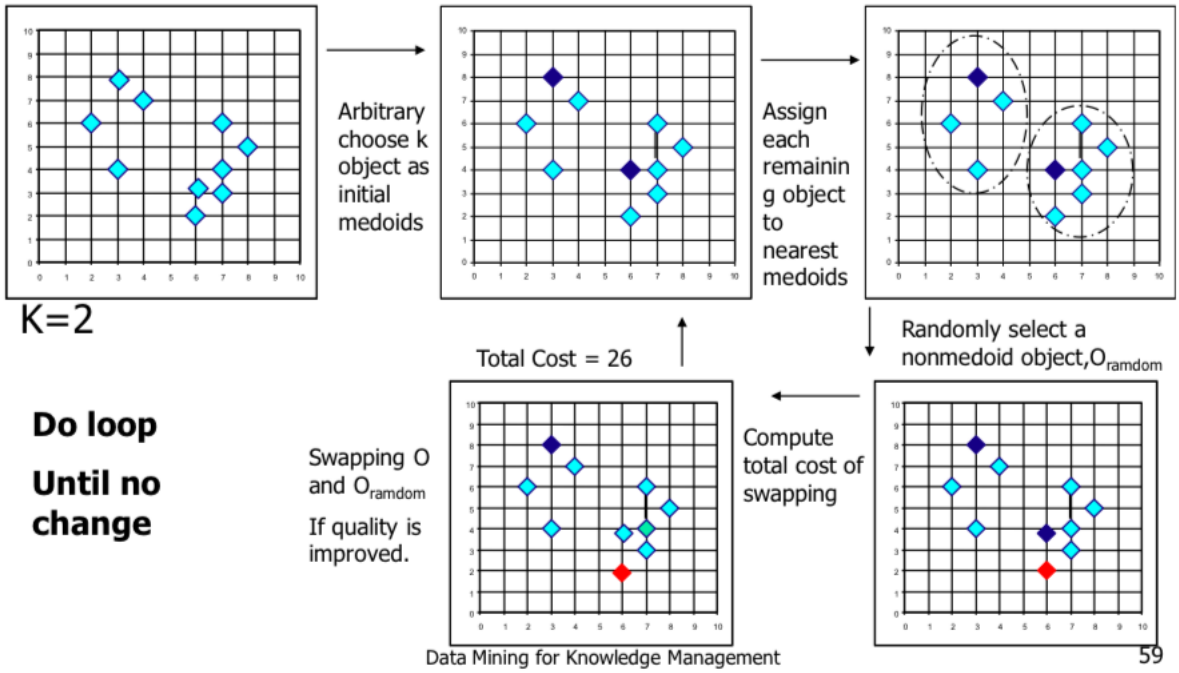

FIGURE 10 – Schématisation de l'algorithme K-medoids pour K = 2 et nrNodes = 10

L'avantage de cette méthode est qu'un medoid est beaucoup moins sensible aux valeurs aberrantes qu'une moyenne ce qui rend l'algorithme plus robuste face à ces dernières. Cependant l'algorithme pseudo-aléatoire de placement des noeuds tend à répartir uniformément ces derniers sur toute la zone. Pour cette raison on ne perçoit que rarement des valeurs aberrantes dans notre simulation.

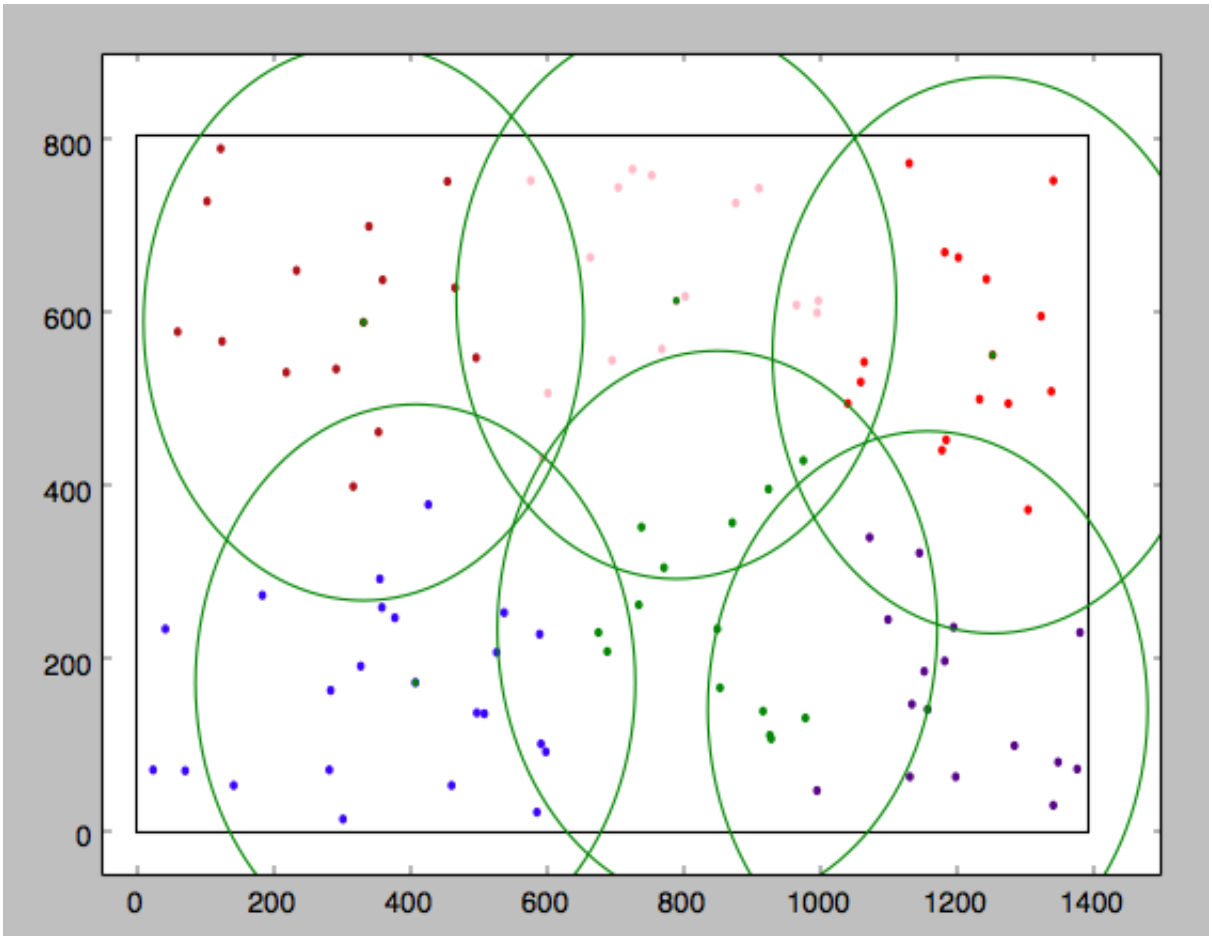

FIGURE 11 – Simulation d'un réseau LoRa avec l'algorithme K-medoids. nrNodes = 100 et DER = 0.92

On peut repérer que quelques noeuds sont mis à l'écart. Ceci est dû à la faible sensibilité de l'algorithme K-medoids face aux valeurs aberrantes. Ceci explique le fait que dans le DER soit en moyenne plus faible que l'algorithme K-means.

3.3 Mean shift

La différence principale de l'algorithme mean shift est que le nombre de clusters est détecté automatiquement. Il n'est plus nécessaire de le fixer avant d'exécuter la simulation. L'algorithme est aussi appelé le chercheur de mode, le mode étant la plus haute densité de points. L'algorithme se présente sous la forme suivante :
1 : Pour chaque point p on détermine quels sont ses voisins V(p)
2 : Pour chaque point on calcule son mean shift, c'est-à-dire son déplacement pondéré par la distance de ses voisins :

$$m(p) = \frac{\sum_{p_i \in V(p)} K(pi - p)pi}{\sum_{p_i \in V(p)} K(pi - p)} \tag{8}$$

Il s'agit d'une moyenne pondérée dont le poids est donnée par une fonction K appelée Kernel Gaussien :

$$K(x) = \frac{1}{b\sqrt{2\pi}} e^{-0.5x/b} \tag{9}$$

avec b une constante telle que b = 1000
3 : Le mean shift de chaque point est donné par m(p) - p
4 : On répète à partir de l'étape 1 pour n itérations ou jusqu'à ce que les points arrêtent de bouger. Dans notre cas on choisit n = 50 ce qui est plus que suffisant expérimentalement parlant.

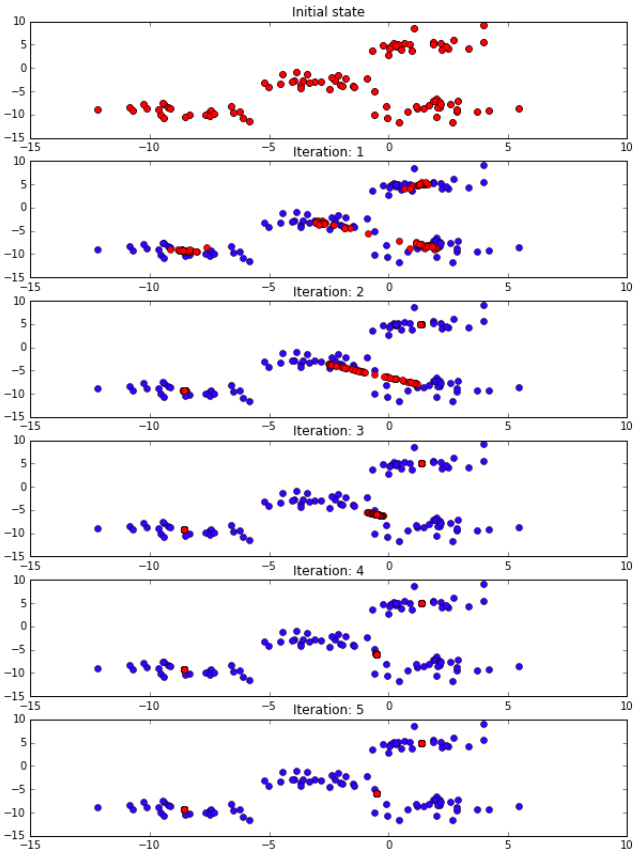

FIGURE 12 – Schématisation de l'algorithme mean shift

Etant donné que nous ne nous intéressons qu'aux simulations dont le nombre de passerelles est 6, on ne retient que les topologies de noeuds où l'algorithme détecte 6 clusters. On écarte de ce fait certaines simulations dont la topologie ne convient pas. L'autre solution aurait été de prendre chaque simulation peu importe le nombre de clusters détectés, cependant cela reviendrait à comparer des simulations avec des nombres de passerelles différents, ce qui donnerait des résultats inutilisables.

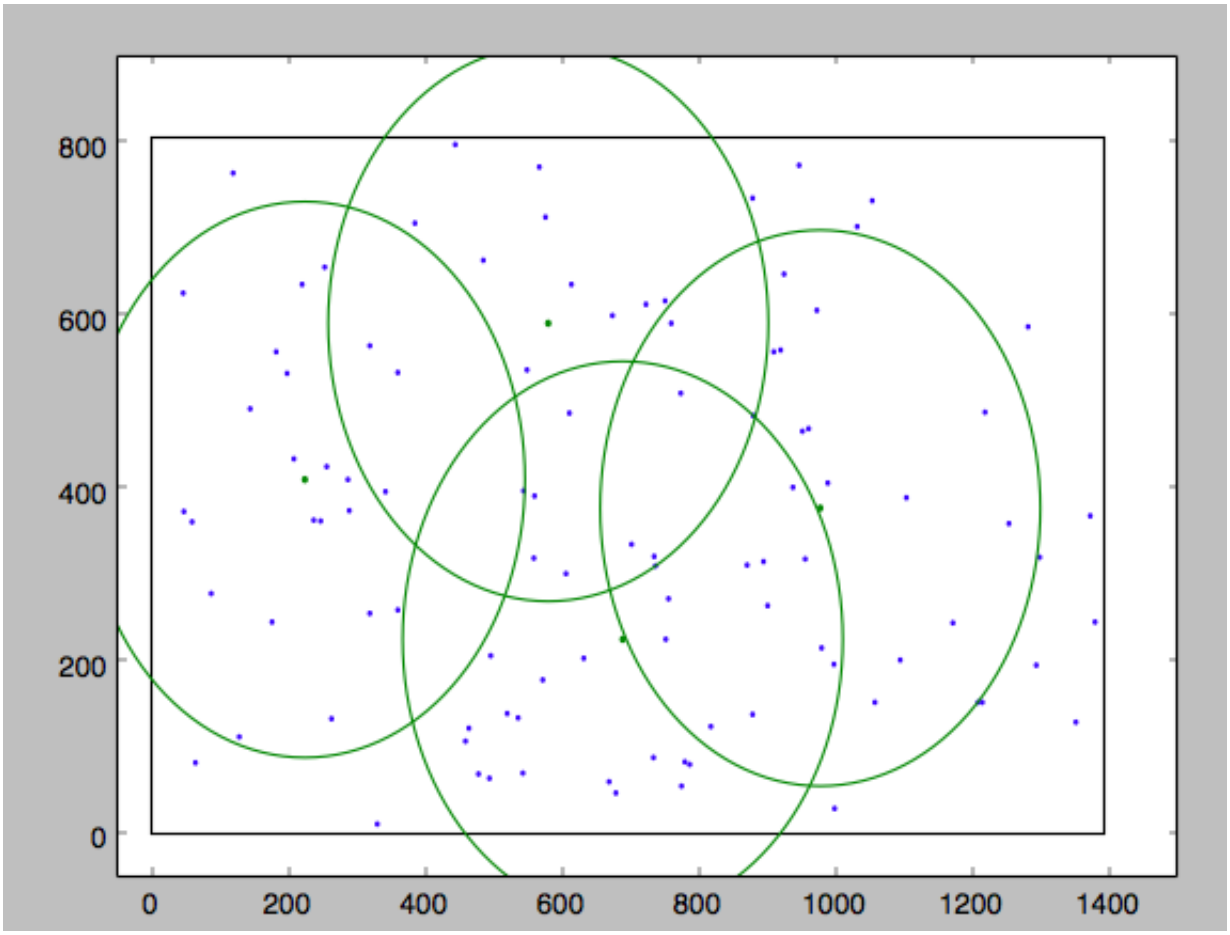

FIGURE 13 – Simulation d'un réseau LoRa avec l'algorithme Mean shift. nrNodes = 100 et DER = 0.79

3.4 Confrontation des résultats

Le protocole des simulations est le suivant : Pour chaque algorithme, on réalise des simulations de 58 jours avec une émission de chaque noeud toutes les 16.7 minutes, ce qui correspond au fonctionnement courant d'un réseau LoRa. On augmente également à chaque fois de 50 le nombre de noeuds présents et on réalise 50 simulations dont on extrait une moyenne et une variance. On compare les résultats obtenus entre eux et avec un placement manuel des passerelles initialement utilisé par l'équipe de recherche de l'Université de Lancaster. Les dimensions du "quartier" choisies sont en fonction de la distance maximale théorique de communication entre un noeud et une passerelles et permettent de couvrir la totalité de la zone avec 6 passerelles. Cependant pour des raisons de temps de calcul, au-delà de 500 noeuds, la simulation devient beaucoup trop lente pour la ré exécuter 50 fois. Afin de réduire le nombre d'exécutions à réaliser on mesure la variance :

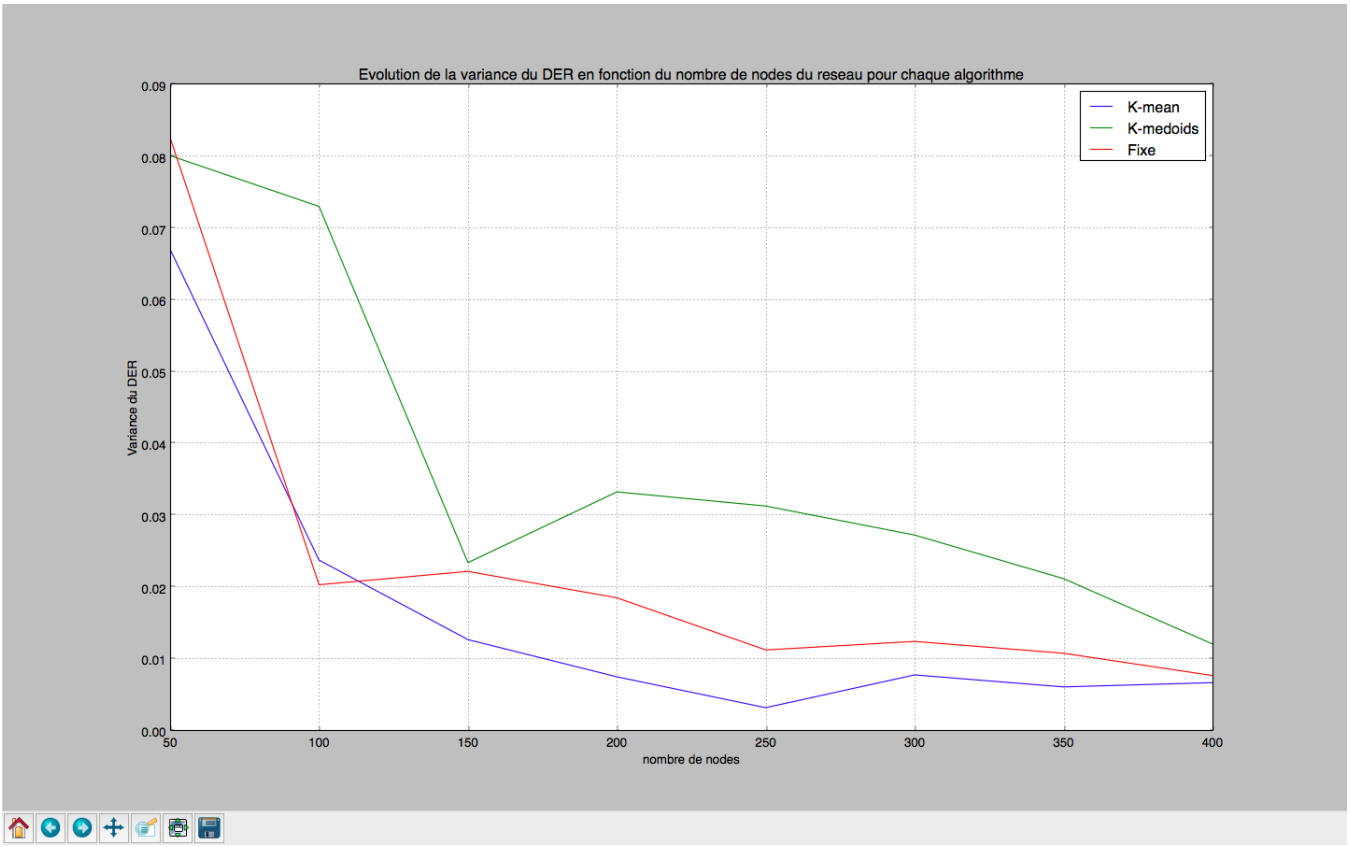

FIGURE 14 – Evolution de la variance du DER en fonction du nombre de nodes du reseau pour chaque algorithme

Etant donnée la faible variance mesurée (inférieure à 0.02 pour nrNodes $>$ 350) , il est légitime de ne faire que quelques tests pour des nombres de noeuds supérieurs. On explique cette faible variance par le fait que l'algorithme de placement des noeuds, bien qu'aléatoire, tend à répartir ces derniers uniformément sur toute la surface. Ainsi, pour chaque simulation, les topologies de noeuds sont similaires. Quoi qu'il en soit, on peut maintenant supposer que quelques simulations pour chaque expérience sont nécessaires et non plus 50. Cette décision permettra d'observer des résultats pour des simulations allant jusqu'à 700 noeuds.

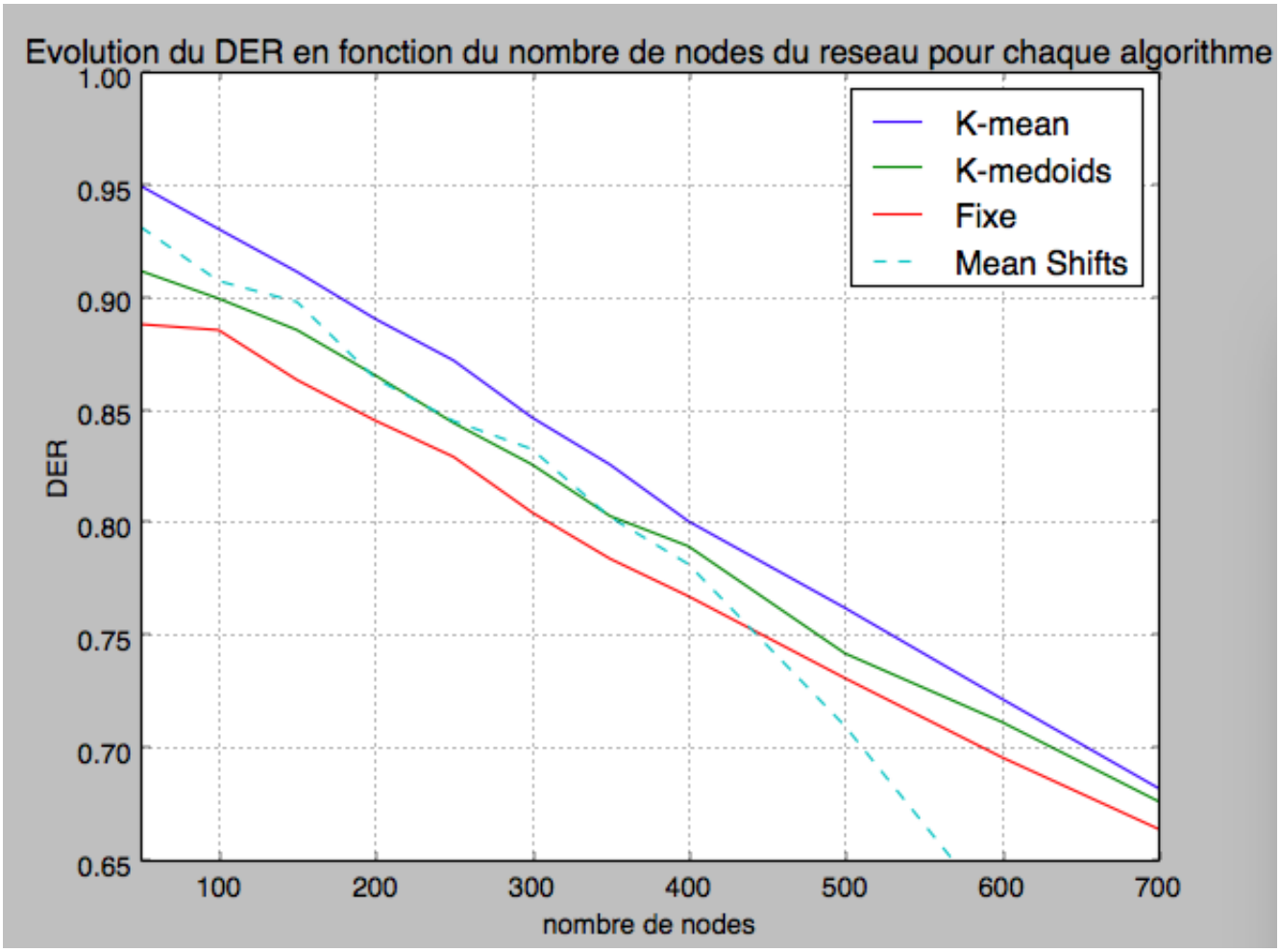

FIGURE 15 – Résultats des simulations

On voit qu'il y a un écart quasiment constant de 0.02 entre chaque algorithme. Notre simulation avec l'algorithme mean shift donne cependant des résultats peu concluants car ils ne reposent que sur quelques simulations pour les raisons expliquées précédemment. Si l'on considère qu'un réseau fonctionne si on a au minimum un DER de 0.9, l'algorithme K-means qui est le plus performant permet pour notre quartier d'avoir 180 noeuds au lieu de quelques uns pour un placement fixe, 100 pour l'algorithme K-medoids et 120 pour Mean Shifts.

4. Autres topologies et points mobiles

4.1 Autres topologies

Comme on a pu l'observer précédemment la variance entre chaque expérience est faible, ceci est dû au fait que l'algorithme de placement des noeuds n'est pas vraiment aléatoire. En effet il essaie de maximiser la distance entre les noeuds tout en les plaçant aléatoirement. Ceci a pour conséquence que chaque expérience se fait sur le même type de topologie. Dans cette partie nous allons essayer d'analyser les résultats sur un autre type de topologie. Nous allons simuler un réseau dans lequel existe un cluster contenant l'ensemble des points exceptés quelques uns disséminés aux extrémités de la zone de test afin de tester l'impact sur les algorithmes de placement.

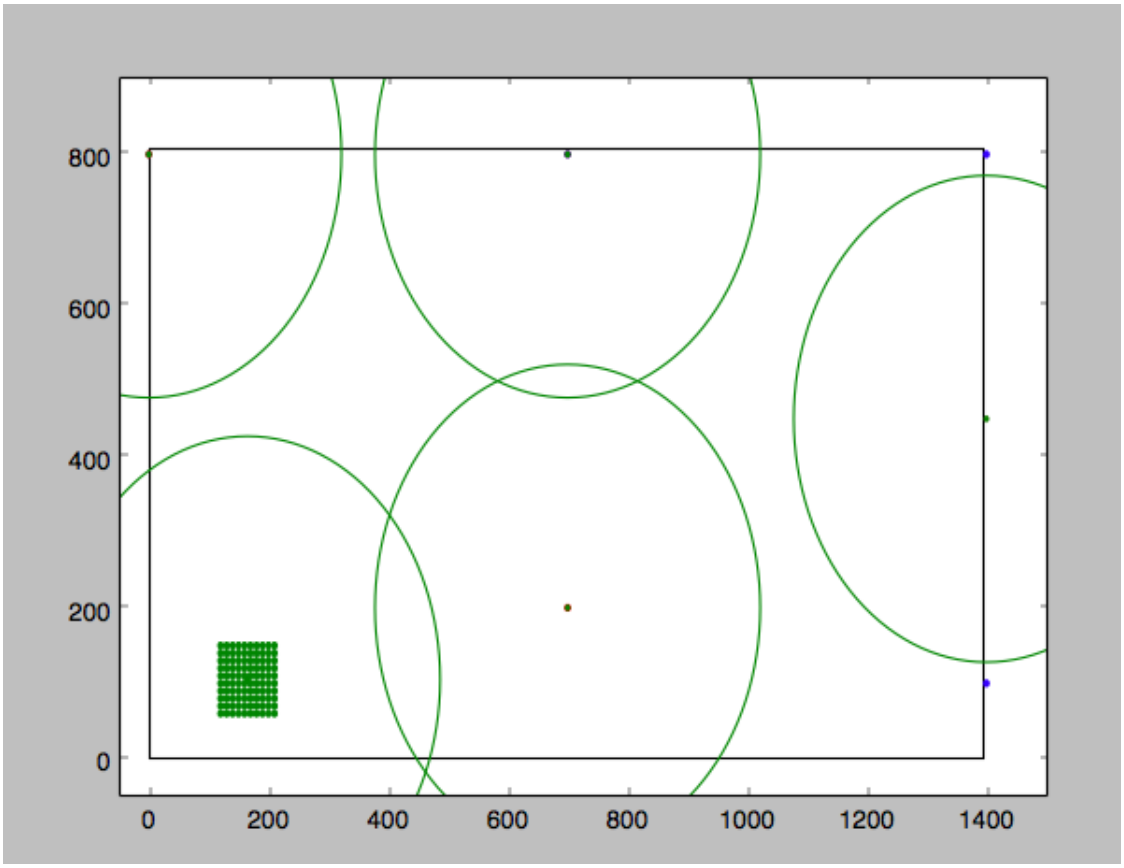

FIGURE 16 – Placement avec l'algorithme K-means

L'algorithme K-means étant sensible aux valeurs aberrantes, chaque point excentré possède sa propre passerelle ce qui réduit considérablement le DER étant donné que les noeuds du cluster ne disposent qu'un nombre réduit de passerelles avec lesquelles communiquer. Le résultat est qu'il y a beaucoup d'interférences.

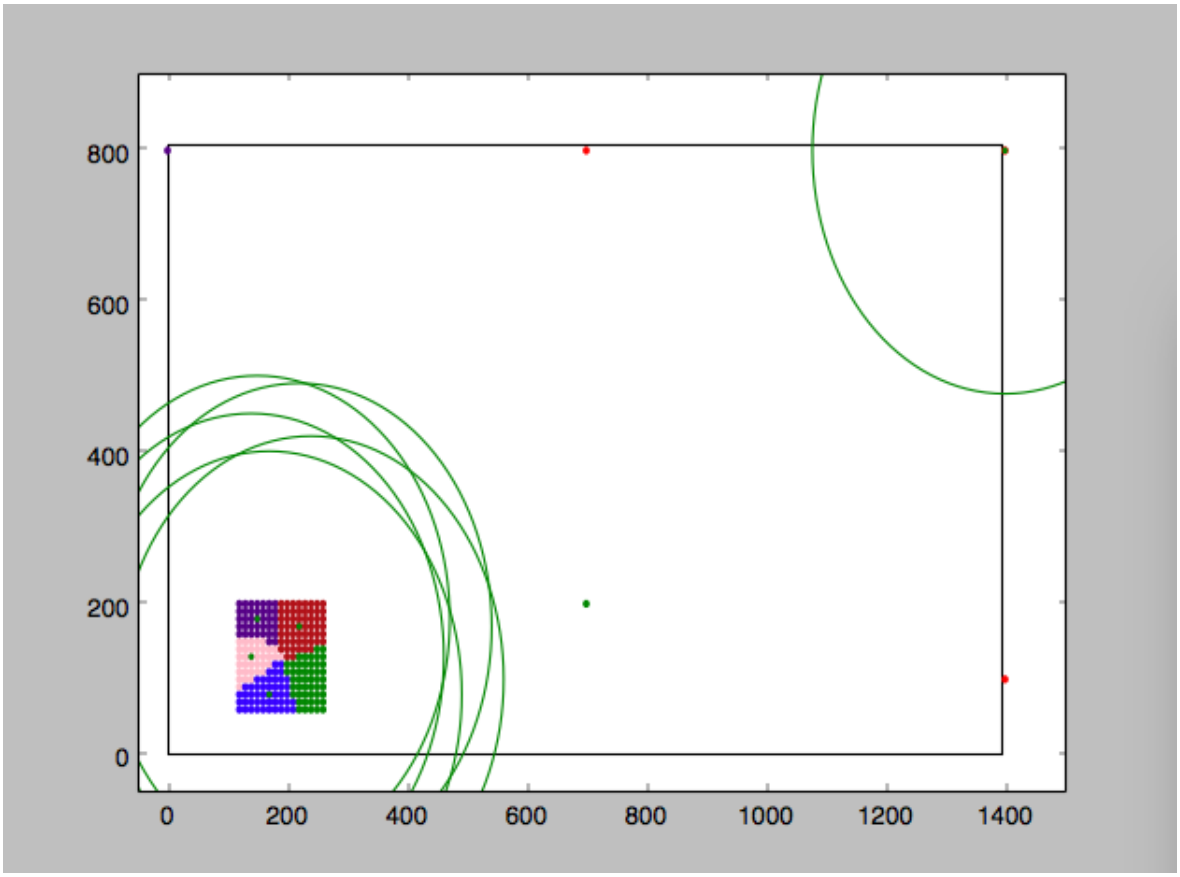

FIGURE 17 – Placement avec l'algorithme K-medoids

L'algorithme K-medoids, comme on le voit sur la figure ci-dessus, ignore les valeurs extrêmes, ce qui permet aux passerelles de se concentrer sur le cluster et de limiter par conséquent les interférences.

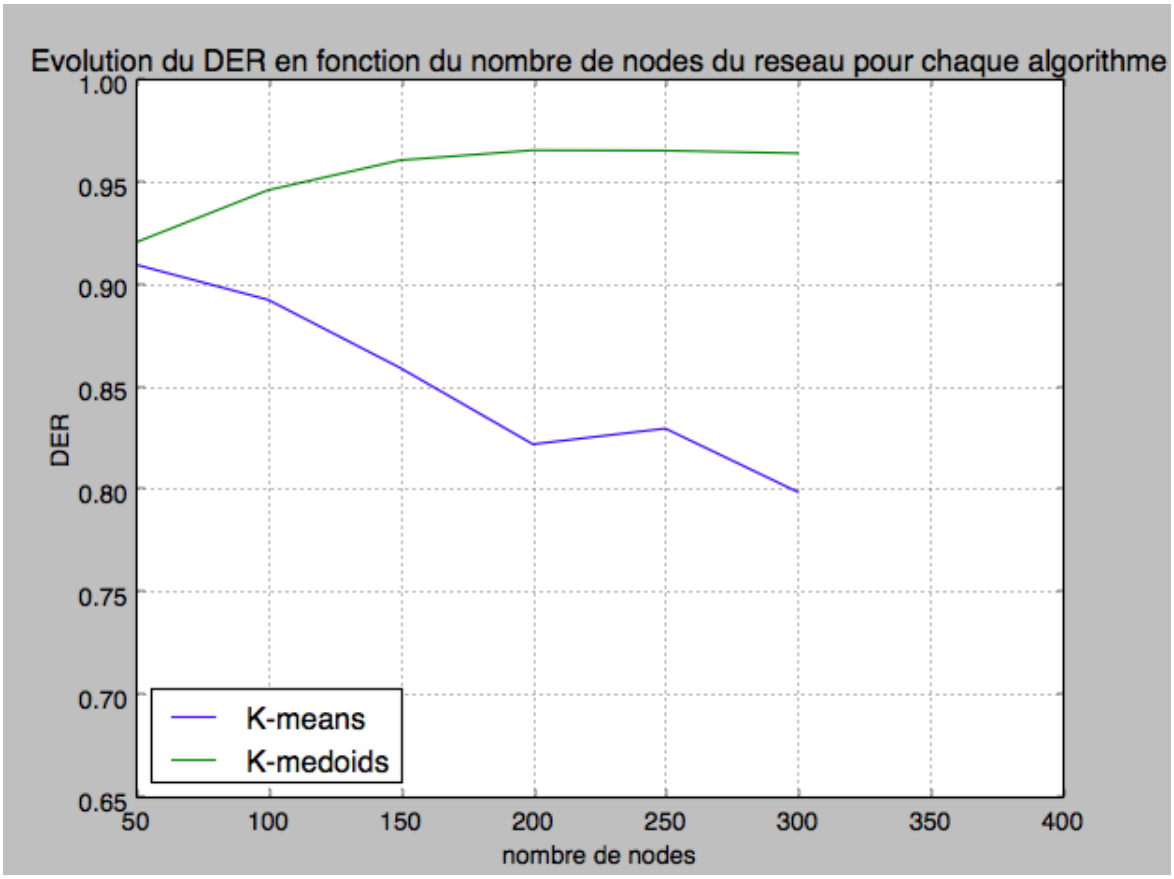

FIGURE 18 – Résultats des simulations

L'algorithme K-medoids permet comme prévu de meilleurs résultats que l'algorithme K-means dans ce type de topologie, étant donnée sa plus faible sensibilité aux valeurs aberrantes. La différence entre les deux algorithmes est nette dans ce type de topologie, cependant certains noeuds se retrouvent complètement incapables de communiquer avec l'algorithme k-medoids.

L'algorithme Mean Shifts n'a jamais convergé en revanche, aucun test n'a été possible.

4.2 Points mobiles

Une fois les passerelles placées en fonction de la topologie, il se peut que certains noeuds puissent disparaître, apparaître ou bien se déplacer. Afin de voir l'impact d'un déplacement de plusieurs noeuds dans un réseau sur son bon fonctionnement, on réalise l'expérience suivante : On place 50% des points sur le côté gauche et on les fait traverser la zone de test. On s'attend à ce qu'un déplacement massif de noeud sans repositionnement des passerelles ait un grand impact sur le DER pour un nombre de noeuds suffisamment grand.

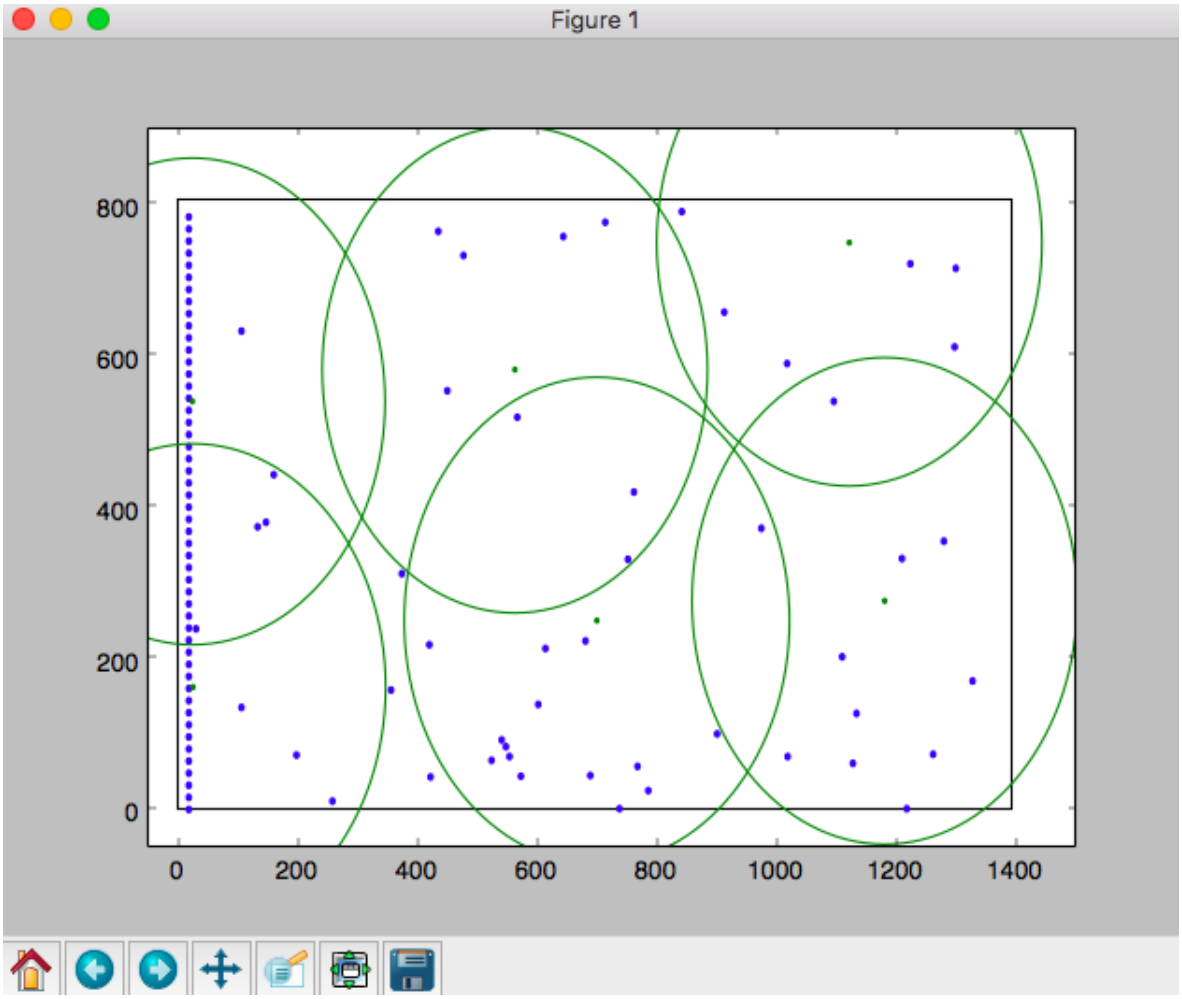

FIGURE 19 – Simulation d'un déplacement de noeuds

On repère ici une zone qui sera traversée et dépourvue de passerelle à proximité (à 400m), on peut penser qu'on assistera à une chute du DER à cet endroit précis.

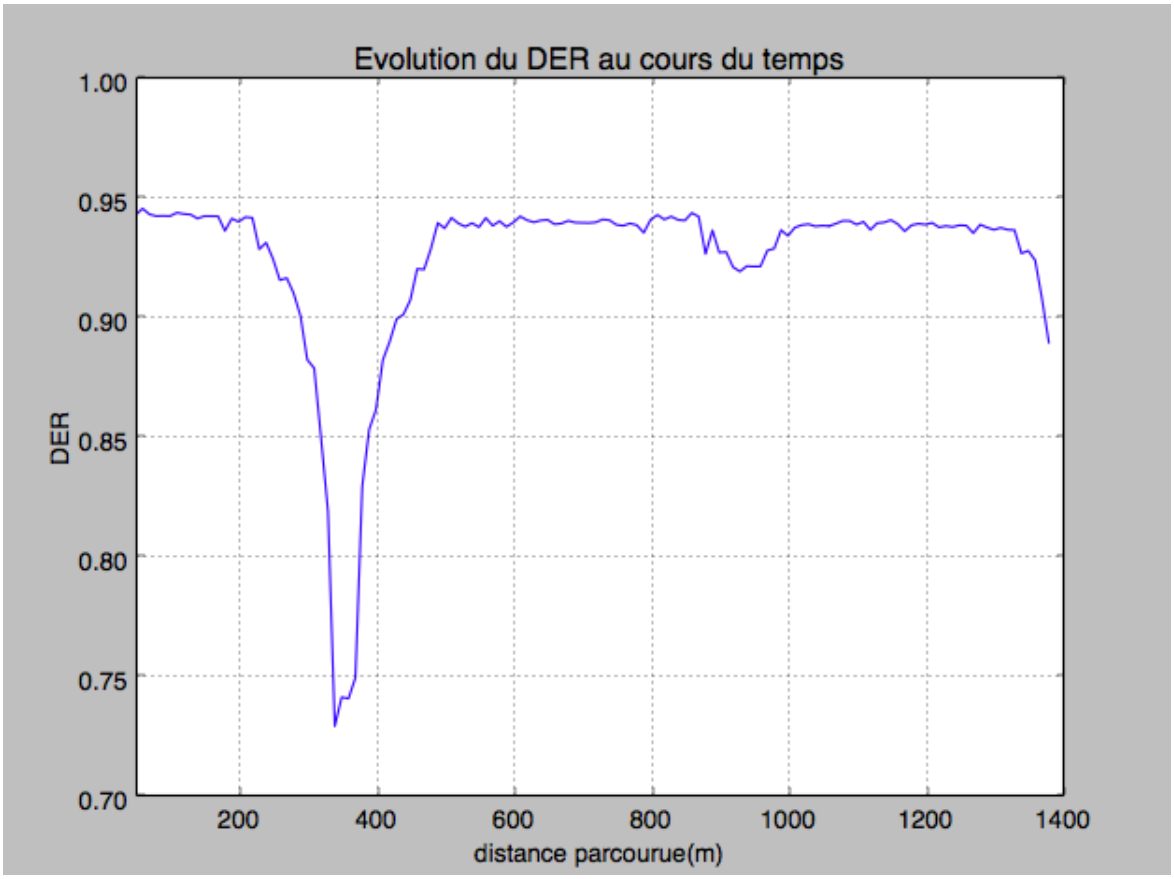

FIGURE 20 – Simulation d'un déplacement de noeuds

Les résultats confirment ce qui a été dit au-dessus, un déplacement de noeuds a un impact négatif sur le DER, et jamais positif étant donné que le placement était optimisé à l'origine.

Conclusion

L'ensemble des résultats a permis de montrer qu'une optimisation topologique est possible afin d'assurer le passage à l'échelle d'un réseau LoRa. L'ensemble des algorithmes utilisés donnent de meilleurs résultats qu'un placement qui ne prend pas en compte la topologie du réseau. On a vu que pour une zone de 112 hectares, en utilisant un algorithme K-means et 6 passerelles, il est possible de placer 180 noeuds tout en conservant le bon fonctionnement du réseau. D'une manière générale l'algorithme de clustering K-means donne les meilleurs résultats bien que l'on puisse trouver des cas de topologies extrêmes qui le rendent moins intéressant. Par ailleurs, un réseau n'est pas immuable et des noeuds peuvent disparaître, apparaître ou être remplacés. Dans ce cas de figure il conviendrait de mesurer le bon fonctionnement du réseau avant de songer à déplacer les passerelles. Enfin les résultats obtenus lors de ce projet peuvent être combinés au paramétrage dynamique et/ou à l'utilisation d'antennes directionnelles dans la mesure du possible et en tenant compte des législations en vigueur afin de s'assurer d'un meilleur passage à l'échelle d'un réseau LoRa.

Difficultés rencontrées

J'ai perdu un mois et demi à coder un simulateur de réseau LoRa en C++ avant de trouver le projet de recherche de l'université de Lancaster. La compréhension, la prise en main de LoRaSim et les modifications en profondeur pour l'adapter à mon étude ont également été chronophages. D'autant plus que le code initial était buggé et que le calcul de maxDist était faux à cause de l'oubli d'un facteur log(10) dans son calcul. Cependant l'utilisation d'un journal de bord m'a permis de m'y retrouver jour après jour et de réaliser mon projet de manière cohérente.

Références

[1] Martin Bor, Utz Roedig, Thiemo Voigt, Juan M.Alonso. *Do LoRa Low-Power Wide-Area Networks Scale ?* http ://www.lancaster.ac.uk/scc/sites/lora/lorasim.html, 2017.

Cours d'analyse d'image de 2A d'Erwan Kerrien :
https ://members.loria.fr/EKerrien/files/data/CETS8AH_6.pdf
LoRa et Sigfox :
https ://aruco.com/2016/03/strategie-reseaux-iot-sigfox-lora/
Matériel et formule du seuil :
https ://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf
K-means algorithm :
https ://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
K-medoids algorithm :
http ://disi.unitn.it/ themis/courses/MassiveDataAnalytics/slides/Clustering2-2in1.pdf
Mean shift algorithm :
http ://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/

Annexes

Github LoRaSIM 2.0 https ://github.com/Zinegit/LoRaSim2.0
Github LoRaSIM (C++) https ://github.com/Zinegit/LoRaSim