SCIENCES SORBONNE UNIVERSITÉ

Master Données, Apprentissage et Connaissances-DAC

RAPPORT PROJET DAC

Clustering pour les infrastructures sans fils

Realisé par :

Hanane Djeddal
Liticia Touzari

Encadré par :

Anastasios Giovanidis

Résumé

L'augmentation croissante du trafic de données a posé de grands défis aux opérateurs mobiles pour augmenter leur capacité de traitement des données, ce qui entraîne une consommation d'énergie et des coûts de déploiement importants sans avoir necéssairement une croissance dans le chiffre d'affairs vu que l'utilisateur attend qu'il paye moins pour plus de données. Avec l'emergence de l'architecture Cloud Radio Access Network (C-RAN) les unités de traitement des données peuvent désormais être centralisées et partagées entre les stations de base, chose qui réduit les coûts de déploiement et offre une architecture de base qui facilite l'implémentation des algorithmes et des solutions pour des problèmes divers. Le partage des unités de traitement se fait en clusterisant les stations de base et en mappant chauqe cluster à une unité de traitement de données. Les schémas de trafic des stations de base étants très dynamiques à différents moments et endroits, par exemple le trafic dans une région résidentielle durant la journée n'est pas le même durant la nuit, L'idée est de créer des cluster de stations de base avec des schéma de trafic complémentaires afin que l'unité de traitement peut être pleinement utilisée à différentes périodes de temps, et la capacité requise à déployer devrait être inférieure à la somme des capacités d'une seule base stations. Cependant, il est difficile de prévoir et de caractériser les schémas de trafic à l'avance pour réaliser des schémas de regroupement optimaux. Dans ce rapport, nous abordons ces problèmes en étudiant les solutions déjà proposées dans le cadre d'optimisation C-RAN basé sur l'apprentissage en profondeur. Premièrement, nous implémentons les algorithmes déjà existants, nous procédons par la suite à évaluer leur performances en utilisant des dataset fournit par Orange. Nous exposons aussi des différents alogorithmes de clustering, principalement K-means, et nous essayons à les adapter à notre problème. Nous terminons par comparer les performances des différentes méthodes.

Mots clés: C-RAN, RAN Cloudification, Clustering, K-means

Contents

1 État de l'art **3**

1.1 L'architecture D-RAN 3
1.2 L'architecture C-RAN 4
1.3 Méthodes de Clustering 5
1.3.1 K-means Clustering 6
1.3.2 Clustering Hierarchique 8
1.4 L'algorithme DCCA 9
1.4.1 Définitions 10

2 Conception **13**

2.1 Analyse des données 13
2.1.1 Données Géographiques 13
2.1.2 Données de Trafic 14
2.2 Application des algorithmes 20
2.2.1 Clustering sur les Données géographique 20
2.2.2 Clustering sur les Données du trafic 22
2.3 K-means Vs DCCA 27

1. État de l'art

Dans cette partie, on va introduire les technologies et concepts principaux dans le projet. D'un part, on parle de l'architecture traditionnelle des réseaux sans fils et son évolution à l'architecure C-RAN. D'autre part, on présente deux méthodes de clustering: K-means et le clustering héirarchique dans leur version la plus générale. À la fin, on introduit la methode de clustering proposée dans l'artcile [3] qu'on va implémenter et evaluer dans la suite du rapport.

1.1 L'architecture D-RAN

Dans l'architecture traditionnelle Distributed Radio Access Network (D-RAN), le site de chaque cellule (eNodeB) contient deux compsants: une unité de traitement de bande de base (BBU) au pied de la tour et une tête radio à distance (Remote Radio Head, RRH) au sommet. Les deux conposants sont reliés par un cable en fibre optique. Le RRH s'occupe des fonctionalités radio telles que conversion des fréquences, amplifications, A/D et D/A conversion etc. Quant à la BBU, elle effectue les traitements de la bande de base, des packets, etc et assure le fonctionnement de la station.

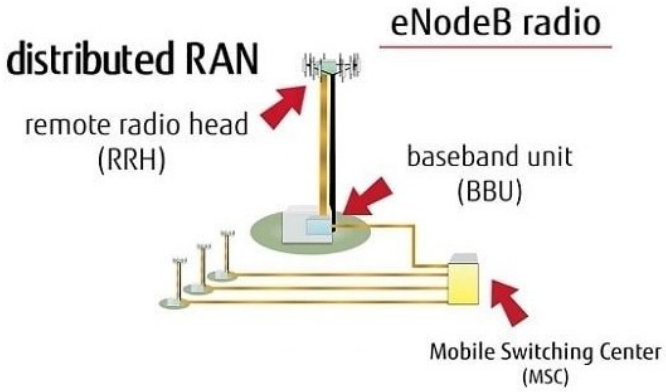

Figure 1.1: D-RAN

Une solution pour accomomder le nouveau volume du trafic est de deployer plus de cellules de petite taille et reutiliser les fréquences. Cependant, cette approche necessite des coûts imporants d'installation et crée un problème d'interférence entre les cellules.
Un autre problème que engendre cette architecture est la consomation d'energie. En effet, les stations de base consomme le plus d'energie dans les réseaux sans fils et augmenter le nombre de cellules c'est augmenter les coûts d'exploitation et l'emission du gaz carbonique, qui, bien évidemment, a un effet négative sur l'environement.
Une nouvelle architecture doit être capable d'offrir une solution à ces problèmes tout en garadant un revenue positif.

1.2 L'architecture C-RAN

L'architecture Cloud Radio Access Network (C-RAN) est un concept qui combine des technique de Centralisation, Collaboration et de Virtulaisation pour offrir une performance aémlioré avec moins de coûts et moins de consomation d'energie (Clean RAN).

L'idée de C-RAN est de centraliser les différentes ressources de traitement de bande de base (les BBUs) pour créer un 'pool' qui gère dynamiquement l'allocation de ressources. Les composants de base dans une architecture C-RAN sont :

1. BBU pool : regroupe l'ensemble BBUs dans un centre et permet l'allocation dynamique et le reconfiguration basée sur des données en temps-réel.
2. RRH : Comme dans les architectures traditionnelle, les RRHs sont distribués dans les différentes station de base et assurent les mêmes fonctionalités de couverage des signaux.
3. Réseau de transmission : une interconnexion entre une instance de BBU et un RRH.

Ce concept, simple et direct, offre plusieurs avantages. La centralisation des BBUs dans un seul pool avec des interconnexions qui relient les différents noeuds

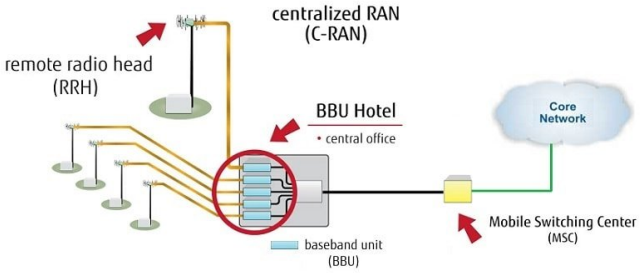

Figure 1.2: C-RAN

avec une bande passante elevée et une faible latance, permet la communication et l'echange d'information et par conséquence plusieurs technologies telles que Joint Processing et cooperative multiPoint (CoMP), difficile à implémenter dans l'architecture traditionnelle, seront facitement intégrées. En plus, contrairement à l'architecture traditionnelle où les ressources d'une BBU sont limitées à la station de base où elle est installée, dans le contexte C-RAN, les ressources sont aggrégées dans un pool (ressources cloudification) et peuvent être allouées sur demande, ce qui réduit la consomation d'energie et optimise l'utilisation des ressources. Aussi, due à sa nature basée sur le concept de Cloud et centralisation, C-RAN est caractérisée par sa flexibilité et scalabilité qui sont necessaires pour l'évolution des systèmes 5G.

1.3 Méthodes de Clustering

Le clustering fait référence à un ensemble très large de techniques pour rechercher des sous-groupes, ou clusters, dans un ensemble de données. Lorsque nous regroupons les observations d'un ensemble de données, nous cherchons à les diviser en groupes distincts afin que les observations au sein de chaque groupe soient assez similaires les unes aux autres, tandis que les observations dans différents groupes sont assez différentes les unes des autres. Bien sûr, pour concrétiser cela, nous devons définir ce que signifie que deux ou plusieurs observations soient similaires ou différentes. En effet, il s'agit souvent d'une considération spécifique au domaine qui doit être faite sur la base de la connaissance des données étudiées. Le clustering étant populaire dans de nombreux domaines, il existe un grand nombre de méthodes de clustering. Nous nous concentrons sur les deux approches de clustering les plus connues: le clustering K-means et le clustering hiérarchique. Dans le clustering K-means, nous cherchons à

partitionner les observations en un nombre prédéfini de clusters. En revanche, dans le clustering hiérarchique, le nombre de clusters n'est pas prédéfini, nous nous retrouvons avec une représentation visuelle arborescente des observations, appelée dendrogramme, qui permet de visualiser immédiatement les regroupements obtenus pour chaque nombre possible de regroupements, de 1 à n. En général, nous pouvons regrouper des observations sur la base des caractéristiques afin d'identifier des sous-groupes parmi les observations, ou nous pouvons regrouper des caractéristiques sur la base des observations afin de découvrir des sous-groupes parmi les caractéristiques. [3]

1.3.1 K-means Clustering

Le clustering K-means est une approche simple et élégante pour partitionner un ensemble de données en K clusters distincts qui ne se chevauchent pas. Pour effectuer le clustering K-means, nous devons d'abord spécifier le nombre souhaité de clusters K; alors l'algorithme K-means assignera chaque observation à exactement l'un des K clusters. La figure ci-dessous montre les résultats obtenus en déroulant l'algorithme sur l'ensemble des RRHs de Lille (ville Française) avec 88 emplacement différents.

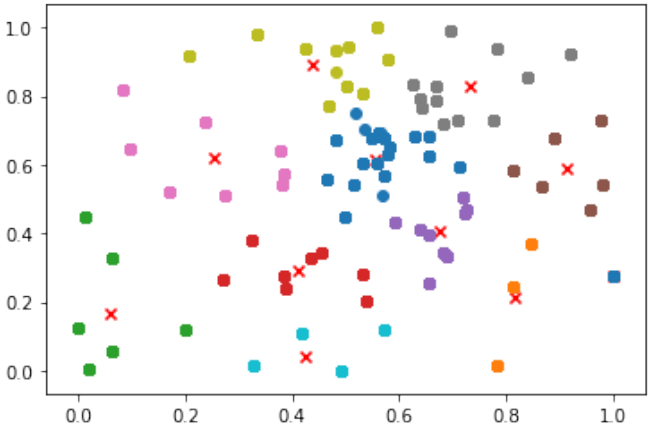

Figure 1.3: K-means clustering

La procédure de clustering K-means résulte d'un problème mathématique simple et intuitif. Nous commençons par définir une notation. Soit C1,. . . , CK désignent des ensembles contenant les indices des observations dans chaque cluster. Ces ensembles satisfont deux propriétés:

1. $C_1 \cup C_2 \cup ... \cup C_k = \{1, ..., n\}$ chaque observation appartient à au moins

l'un des K clusters.

2. $C_k \cap C_{k'} = \varnothing$ aucune observation n'appartient à plus d'un cluster.

Par exemple, si la ième observation se trouve dans le kième groupe, alors $i \in C_k$. L'idée derrière le clustering K-means est qu'un bon clustering est celui pour lequel la variation intra-cluster est aussi petite que possible. La variation intra-cluster pour le cluster C_k est une mesure $W(C_k)$ de la différence entre les observations au sein d'un cluster. Par conséquent, nous voulons résoudre le problème : $\min_{C_1,C_2,...C_K} \sum_{k=1}^{K} C_k$. [3]

En termes, cette formule dit que nous voulons partitionner les observations en K clusters de telle sorte que la variation totale intra-cluster, additionnée sur tous les K clusters, soit aussi petite que possible.
Il s'agit en fait d'un problème très difficile à résoudre avec précision, car il existe presque K^n façons de partitionner n observations en K clusters. Néanmoins, il existe un algorithme très simple pour fournir un optimum local - une assez bonne solution - au problème d'optimisation K-means. Cette approche est présentée dans le pseudo l'algorithme suivant :

Algorithme K-means Clustering

---

1. Attribuez au hasard un numéro, de 1 à K, à chacune des observations. Celles-ci servent d'initialisations.
2. Itérez jusqu'à ce que les affectations de cluster cessent de changer:
   (a) Pour chacun des K clusters, calculer le centroïde du cluster.
   (b) Attribuez chaque observation au cluster dont le centroïde est le plus proche (où le plus proche est défini en utilisant la distance euclidienne par exemple).

Parce que l'algorithme K-means trouve une optimisation locale plutôt que globale, les résultats obtenus dépendront de l'affectation initiale (aléatoire) de chaque observation à l'étape 1 de l'algorithme. Pour cette raison, il est important d'exécuter l'algorithme plusieurs fois à partir de différentes configurations initiales aléatoires. Ensuite, en sélectionner la meilleure solution, c'est-à-dire

celle pour laquelle l'objectif est le plus petit.
Comme vu précédemment, pour effectuer un clustering K-means, il faut définir le nombre de clusters K dés le départ. Le problème de la sélection de K est loin d'être simple.

1.3.2 Clustering Hierarchique

Un inconvénient potentiel de l'algorithme K-means est qu'il faut pré-spécifier le nombre de clusters K. Le clustering hiérarchique est une approche alternative qui ne nécessite pas un choix particulier de K. Le résultat du clustering est souvant traduit par une représentation arborescente attrayante des observations, appelée dendrogramme.

Le dendrogramme du clustering hiérarchique est obtenu via un algorithme extrêmement simple. Commençant par définir une sorte de mesure de dissimilarité entre chaque paire d'observations. Le plus souvent, la distance euclidienne est utilisée. L'algorithme se déroule de manière itérative. Partant du bas du dendrogramme, chacune des n observations est traitée comme son propre cluster. Les deux clusters qui se ressemblent le plus sont ensuite fusionnées pour qu'il y ait n - 1 clusters. Ensuite, les deux clusters qui se ressemblent le plus sont fusionnés à nouveau, de sorte qu'il en reste n - 2 clusters. L'algorithme procède de cette manière jusqu'à ce que toutes les observations appartiennent à un seul cluster et que le dendrogramme soit terminé.
L'algorithme de clustering hiérarchique est donné comme suit:

Algorithme Clustering Hiérarchique

---

1. Commencez par n observations et une mesure (telle que la distance) et traitez chacun observation comme un cluster.
2. Pour i= n, n-1, ..., 2:
   (a) Examinez toutes les dissemblances inter-cluster par paires parmi les i clusters et identifiez la paire de clusters qui sont les moins dissemblables (c'est-à-dire les plus similaires). Fusionnez ces deux clusters. La dissimilarité entre ces deux groupes indique la hauteur dans le dendrogramme

à laquelle la fusion doit être placée.

(b) Calculez les nouvelles dissemblances inter-cluster par paire parmi les i-1 clusters restants.

Le concept de dissimilarité entre une paire d'observations doit être étendu à une paire de groupes d'observations. Cette extension est obtenue en développant la notion de lien, qui définit la dissimilarité entre deux groupes d'observations. Les quatre types de liens les plus courants - complet, moyen, unique et centroïde - sont brièvement décrits dans le tableau ci-dessous:

Linkage	Description
Complet	Différenciation intercluster maximale. Calculez toutes les disparités par paires entre les observations du cluster A et les observations du cluster B, et retenir la plus grande de ces différences.
Unique	Dissimilarité intercluster minimale. Calculez toutes les disparités par paire entre les observations du cluster A et les observations du cluster B et noter la plus petite de ces différences. Un couplage unique peut entraîner des clusters étendues dans lesquelles des observations uniques sont fusionnées une par une.
Moyen	Dissimilarité intercluster moyenne. Calculez toutes les disparités par paires entre les observations du cluster A et les observations du cluster B et notez la moyenne de ces différences.
Centroïde	La dissimilarité entre le centroïde du cluster A et le centroïde du cluster B. La liaison centroïde peut entraîner des inversions indésirables.

1.4 L'algorithme DCCA

Pour optimiser l'utilisation des ressources, et maximiser l'utilié des unités de traitement de base, l'association BBU-RRH doit prendre en considération les variation du trafic. En effet, la demande de trafic de données n'est pas uniformément distribuée sur les différentes régions et périodes du temps (voir figure 4). Donc, c'est important de regrouper des RRHs avec des schémas de trafic complémentaire afin que l'unité de traitement peut être pleinement utilisée. L'algorithme Distance-Constrained Complementarity-Aware (DCCA)

est une méthode proposée par [3] qui permet de trouver des schéma de clustering optimaux pour maximiser l'utilité de la capacité et réduire les coûts.

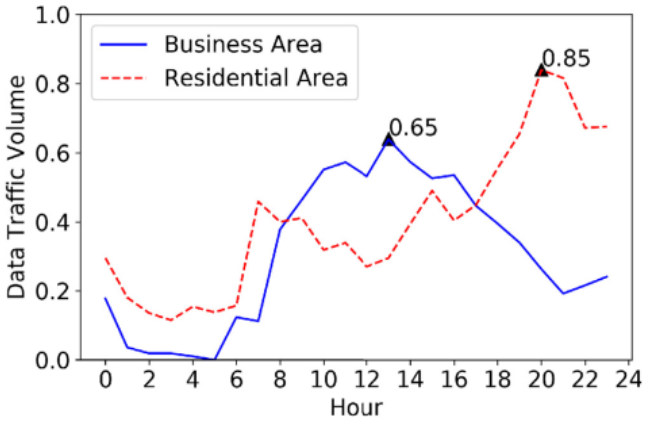

Figure 1.4: Volume de trafic

L'algorithme introduit une mesure de complémentarité entre les RRHs utilisée pour calculer la connéctivité entre un RRH et un cluster. L'objectif de DCCA est d'avoir une connectivité entre un RRH 'r' et son cluster qui est supérieure à la connectivité entre'r' et tout autre cluster.
$\forall v \in C_k$ Con(v,C) $\geq$ max $Con(v, C_l), C_l \in P$
Dans la méthode proposée, une étape de prédiction du trafic précède l'application de DCCA. Pour chaque RRH, un pattern de trafic est prédit pour une duré du temps future basé sur l'historique du trafic du RRH. Un model Multivariate Long Short-Term Memory (MuLSTM) est utilisé pour générer une matrice F. L'algorithme MuLSTM prend un F_i et retourne un F_{i+1} tel que Fi est une matrice de dimension [Nt,Nr] avec, Nt: nombre de time slots et Nr: nombre des RRHs.
Ce modèle est utilisé pour prédire le trafic heure par heure pour le jour suivant. Le clustering des RRHs sera mit à jour dynamiquement selon ce trafic prédit.
L'étape suivante est l'application de DCCA. Avant d'introduire l'algorithme, on va définir quelque mesures.

1.4.1 Définitions

1. **TimeSlot :** On divise la journée en tranches du temps où on considère le trafic. Par exemple si on prends un TimeSlot de 10 minutes, ça veut dire

qu'on recupère la valeur du trafic chaque 10 minutes. Dans ce projet, on considère un TimeSlot d'une heure.

2. **Distribution de peak hours :** Pour un cluster donné C =$\{r_1, r_2, ..., r_l\}$, on recupère les peak hours des RRHs du cluster, soit T.
   C'est à dire, pour chaque RRH du cluster, on recupère la liste des heures considérées comme peak hours pour ce RRH, c'est T[ri].
   On calcule par la suite l'entropie de shannon sur les probabilité d'avoir un peak-hour dans le cluster.
   H(C) = - $\sum_{k=1}^{K} p_k \log p_k$
   Où K=|T(C)| qui correspond au nombre total des peak hours dans le cluster et p_k est la propabilité d'observer le peak hour correspondant dans l'ensemble T(C).
   Une grande valeur de l'entropie implique une grande uncertitude ce qui veut dire une grande dispersion entre les peak hours dans le cluster.

3. **L'utilité de la capacité :** Le trafic aggréger des RRHs du cluster doit être proche à la capacité de la BBU du cluster sans la dépasser.
   $U(C) = \left(\frac{meanf(C)}{|B|}\right)^{\ln \frac{meanF(C)}{|B|}}$
   où f(C) = $\sum_{i=1}^{n} f(r_i)$ correspond au trafic agrégé des RRHs du cluster.

4. **Complémentarité :** $M(C) = U(C) * H(C)$

5. **Matrice de complémentarité :** Il faut prendre en considération la distance entre les RRHs pour que les délais de propagation entre BBU et RRH respectent les contraintes de qualité de service, et aussi pour permettre la communication entre RRHs. Donc on définit un τ tel que les RRHs qui sont séparés par une distance $> \tau$ ne sont pas regroupés ensemble. La matrice de complémentarité a la forme [Nr,Nr] et associe à chaque couple(ri,rj) la valeur : $w(ri, rj) = M(ri, rj) * a_ij$ tel que $a_ij = \begin{cases} 1 & \text{si } dist(r_i, r_j) < \tau, \\ 0 & \text{sinon.} \end{cases}$

6. **Connectivité :** Elle représente la mesure de distance qui permet d'affecter un RRH à un cluster: $con(v, C) = \sum_{v' \in C} w_{vv'}$
   Il faut prendre en considération de plus la distance entre le RRH et les

autres clusters, on definit donc :
$value(v, C) = con(v, C) * \log\left(\frac{\tau}{maxdist(v,v')}\right)$

7. **Clusters adjacents :** $\mathbb{C}(v) = \{C | con(C, v) > 0, C \in \mathbb{P}\}$
   représente l'ensemble des clusters à qui le RRH v peut appartenir. C'est à dire la contraine de la distance est respectée et donc la connectivité est strictement positive.

L'algorithme DCCA peut être, donc, décrit comme suit :

Algorithme DCCA

---

1. Initialement, attribuer à chaque RRH un numéro (label) unique de cluster. Chaque RRH représente un cluster.
2. Itérez jusqu'à ce que les affectations de cluster cessent de changer, ou on atteint le nombre max d'itérations:
   (a) Parcourir les RRHs dans un ordre aléatoire.
      i. Pour chaque RRH, on recupère l'ensemble de clusters adjacents, soit AC.
      ii. Parmi les clusters de AC, on choisit celui qui a la value(v,C) max, soit newCluster.
      iii. Si newCluster est different de l'ancient cluster de RRH (différents labels), on reaffecte RRH au nouveau cluster.

---

2. Conception

2.1 Analyse des données

Dans cette section on va analyser les données de géo-localisation et de trafic fournies par l'opérateur Orange pour les villes française: Paris, Nantes, Lille et Lyon.

2.1.1 Données Géographiques

Les données géographiques représentent les positions des RRHs sur un plan 2D (coordonnee x et y).

Analyse des données pour la ville Lille

- Nombre de RRHs est 1394 et le nombre de régions est de 88 RRHs (avec des RRHs à la même position).

- Cellules géographiques : le digramme de Voronoi ci-dessous permet de délimiter les zones géographiques dont est responsable chaque RRH et ainsi calculer la superficie de la zone :

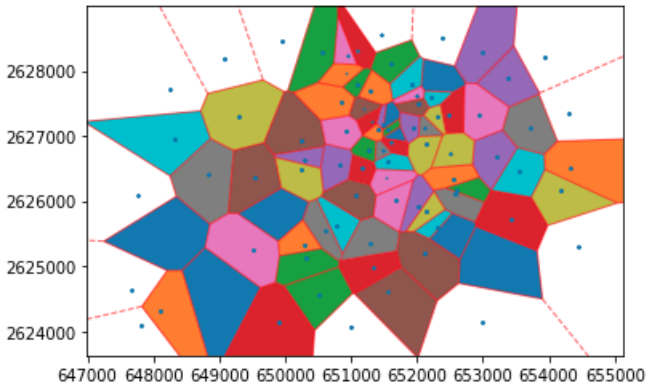

Figure 2.1: Diagramme de Voronoi pour Lille

- Pour chaque RRH on évalue le nombre RRHs à distance variante de celui-ci: Exemple : Pour le RRH à la position (649540, 2626350) on obtient les graphes suivants avec un pas de 500 m:

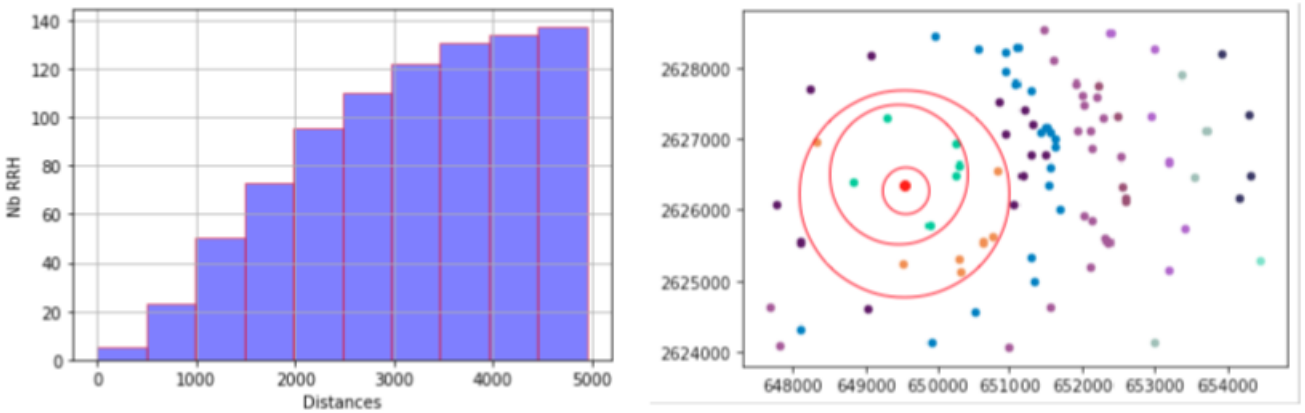

Figure 2.2: Histogramme du nombre de RRHs par distance d'un point

2.1.2 Données de Trafic

Analyse des données pour la ville Lille

Les données de trafic renseignent pour chaque RRH le nombres de bytes up et bytes down pour des timeslot de 10min entre les mois de mars et juin 2019 ainsi que le maximum et minimum des bytes en up et down pour les RRHs tel que:
Min traffic up : 0.0 Max traffic up : 7249226387.374723

Min traffic down : 0.0 Max traffic down : 15828358767.955057
Les courbes suivantes représentent le trafic up et down pour un RRH donné:

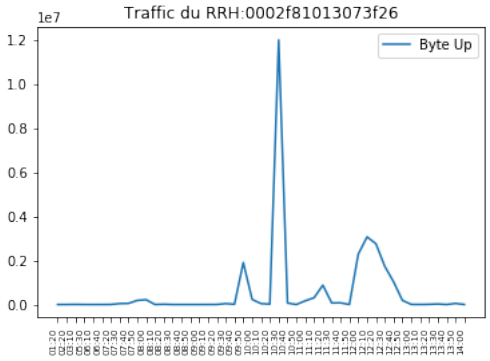

Figure 2.3: Byte Up matin

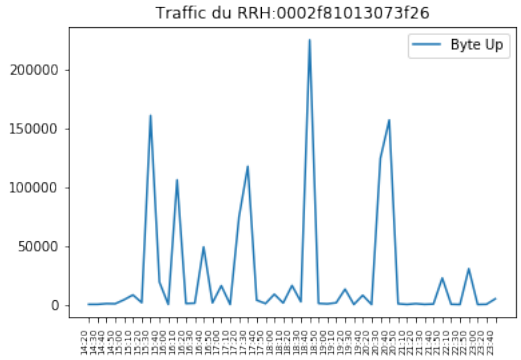

Figure 2.4: Byte Up aprés 14h

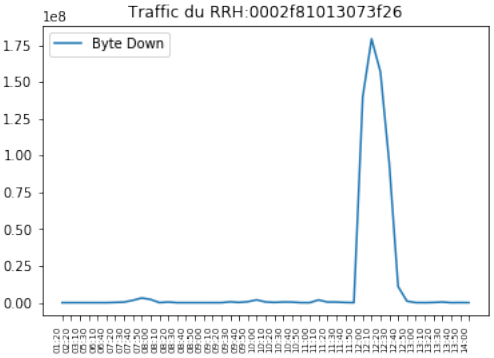

Figure 2.5: Byte Down matin

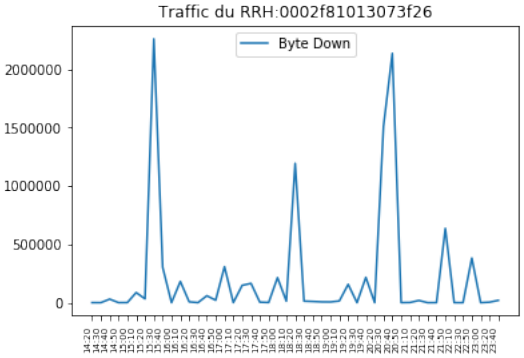

Figure 2.6: Byte Down aprés 14h

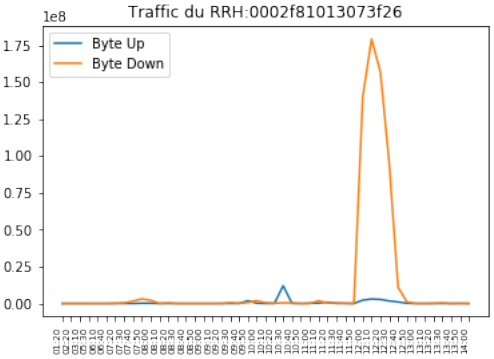

Figure 2.7: Comparaison Byte Up et Byte Down matin

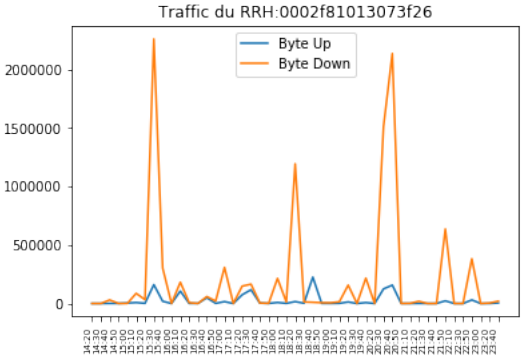

Figure 2.8: Comparaison Byte Up et Byte Down aprés 14h

On remarque bien que le trafic en down est beaucoup plus élevé qu'en up, et on peut facilement repérer les pics de trafic.

Les figures ci-dessous représente pour des périodes et jours différents le trafic RRH dans les régions du digramme de voronoi avec un code couleur.

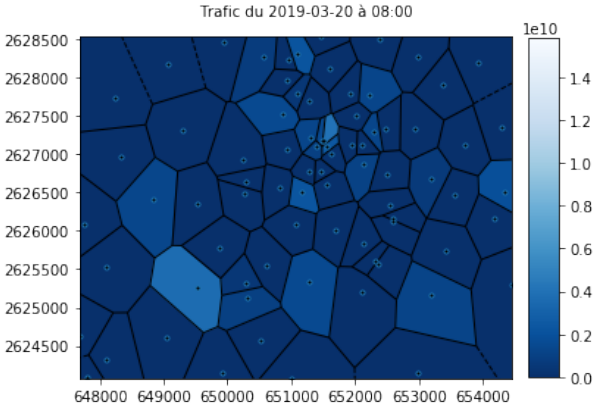

Figure 2.9: Jour de semaine

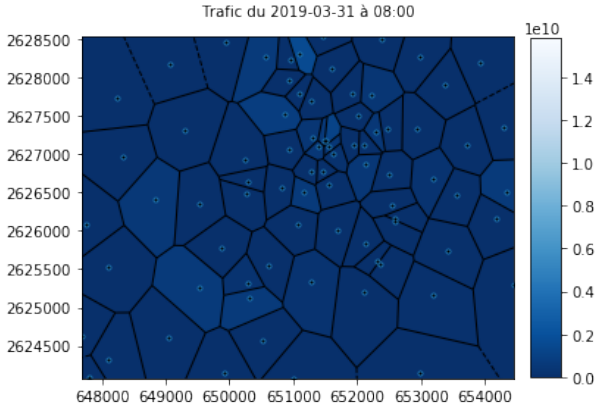

Figure 2.10: Weekend

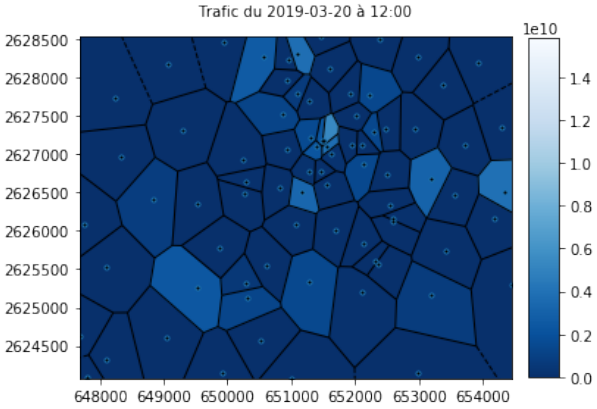

Figure 2.11: Jour de semaine

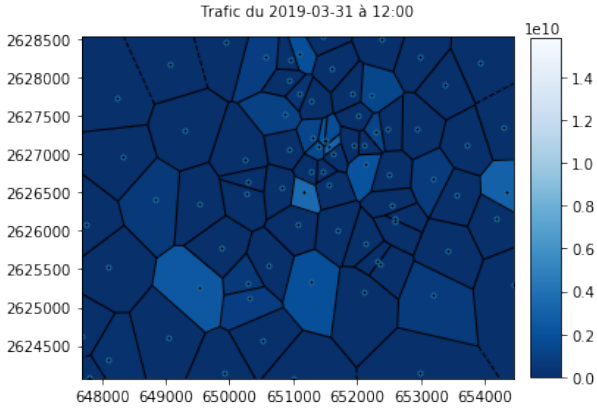

Figure 2.12: Weekend

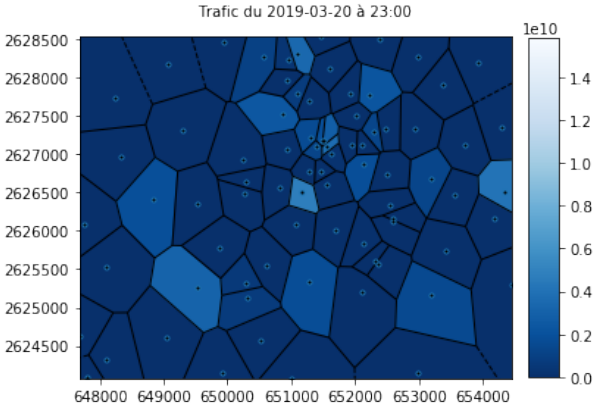

Figure 2.13: Jour de semaine

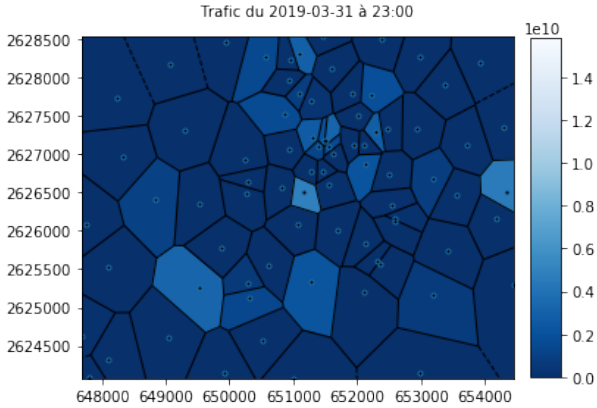

Figure 2.14: Weekend

Les régions à trafic élevé

On considère les régions qui atteignent un trafic supérieur à la moyenne du trafic en général comme étant des régions à trafic élevé, la figure qui suit sépare les régions de Lille en régions à trafic élevé et régions à trafic réduit, on obtient pratiquement les même régions avec un seuil de deux fois la moyenne du trafic pour les régions à trafic élevé et un seuil d'un demi de la moyenne du trafic pour les régions à trafic réduit:

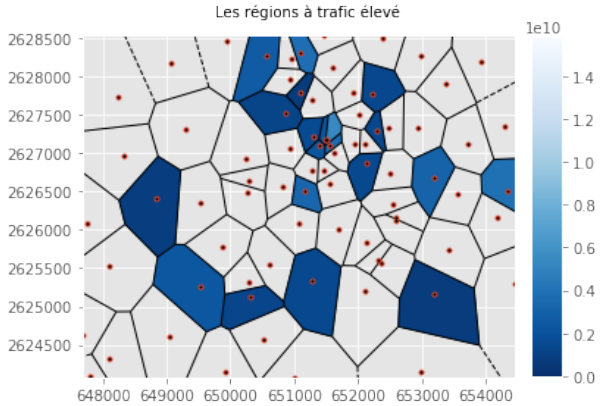

Figure 2.15: Régions avec un trafic supérieur à la moyenne

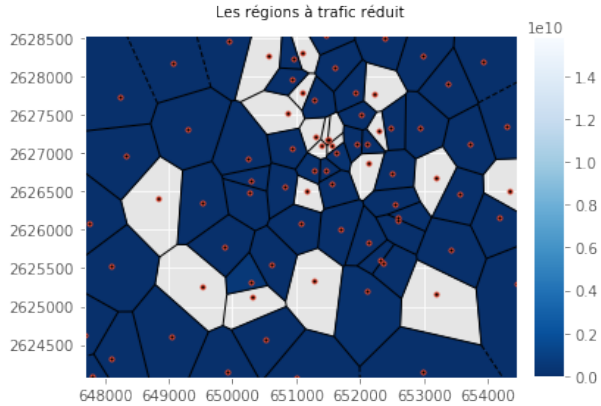

Figure 2.16: Régions avec un trafic inférieur à la moyenne

On remarque que les régions à trafic élevé correspondent plus ou moins aux régions dont le trafic varie le plus.

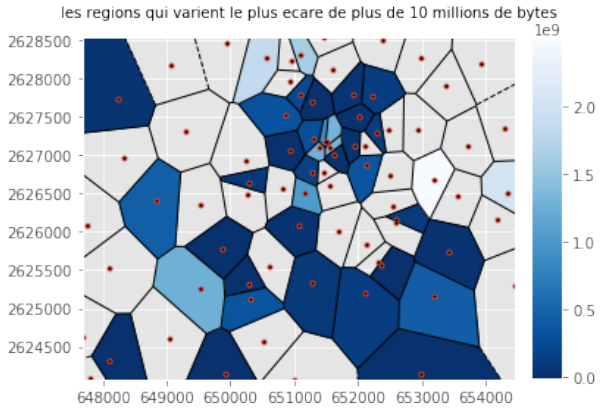

Figure 2.17: Les régions dont le trafic varie le plus

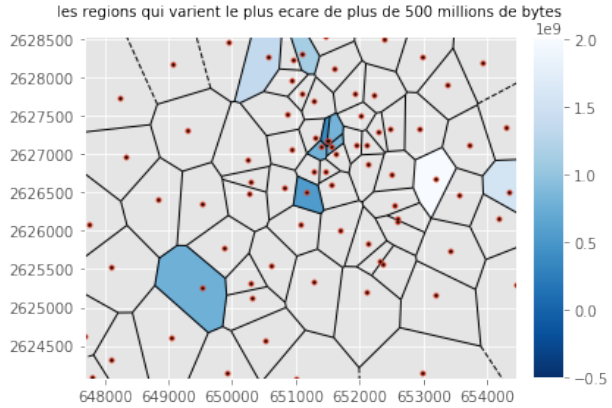

Figure 2.18: Les régions avec très forte variation écart de plus de 500 millions de bytes entre périodes différentes

Corrélation des deux types de trafic:

La corrélation générale en concaténant pour tous les RRHs les deux trafic

donne une valeur de 0.7228, qui correspond à une forte corrélation. Cependant, l'étude de chaque RRHs révèle des corrélations qui varient. La figure ci-dessous représente les régions dont les trafics up et down sont très corrélés (corrélation supérieure à 0.5), dont la plupart font parti des régions à trafic élevé.

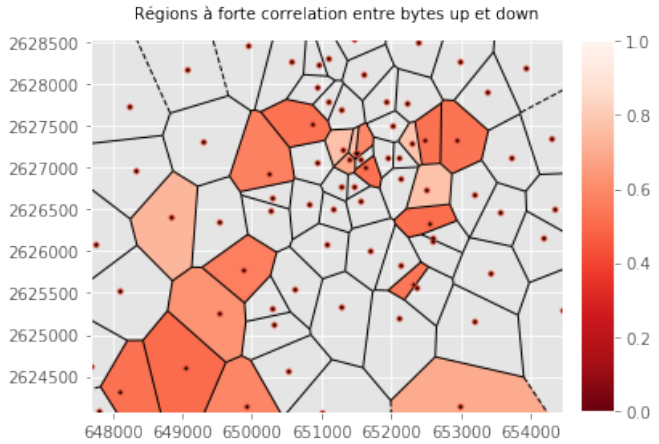

Figure 2.19: Régions à forte corrélation entre byte up et down

La plupart des régions à forte corrélation représentent des régions à trafic élevé.

Capacité de la BBU:
Pour determiner la capacité de la BBU, on peut suivre deux approche.

1. Le maximum du trafic:
La capacité d'une BBU satisfait le trafic du RRH auquel elle y est rattachée. Il suffirait donc de prendre la valeur maximum du trafic comme capacité, de plus puisque le trafic en down a des valeurs bien plus élevées par rapport au trafic en up et qu'il y a une certaine corrélation entre les deux, on envisage dans la suite du projet de dérouler les algorithmes de clustering sur les bytes en down.

Pour la ville de Lille: la valeur maximum du trafic relevé sur les mois de mars jusqu'à juin 2019 est de 15 828 358 767.955057, on peut donc poser comme capacité BBU la valeur 15 828 358 800

La figure suivante représente la région de l'un des RRH avec les régions dont

les RRHs lui sont complémentaires en terme de capacité (somme des trafics inférieure à la capacité BBU) pour une date donnée, ici 2019-06-10:

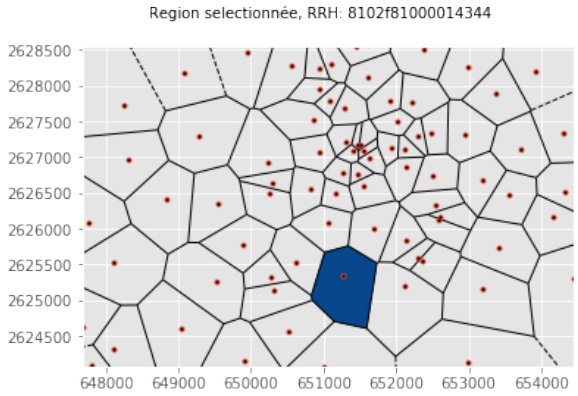

Figure 2.20: Example de région

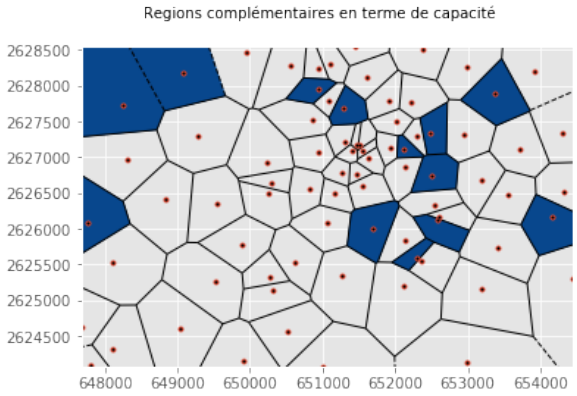

Figure 2.21: RRHs complémentaires

Cette méthode génère une surcharge due au fait qu'on considère le maximum du trafic généré par un seul RRH. Or, les RRHs étants regroupés, il y aura un moment (le TimeSlot où le trafic max est atteint) où la valeur de la capacité sera dépassée. Si le clustering est bien fait, la surcarge ne doit pas être considérable et on peut se permettait cette approximation.

2. Expérimentalement:
Dans le contexte de l'algorithme DCCA, on considère la capacité de la BBU (CBBU) en terme d'unité de capacité, c'est à dire, elle prend des valeurs dans 1,2,3,..etc unités. Donc on peut dérouler l'algorithme sur le trafic normalisé en variant la valeur de la CBBU. On calcule l'écart entre la CBBU et la valeur maximale du trafic aggrégé de chaque cluster. L'idée est de trouver la valeur de CBBU qui permet de ne pas avoir un surcharge, c'est à dire, celle avec un ecart supérieure à 0 pour tous les clusters.

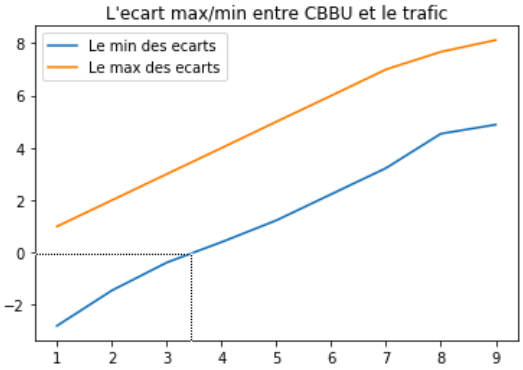

Figure 2.22: Valeur expérimentale de la CBBU

Comme la CBBU est un paramètre d'entrée de l'algorithme DCCA, elle doit être déterminer préalablement, donc le trafic utilisé pour générer le graphe ci-dessus doit être un trafic prédit, ou le trafic des jours passés. Cette méthode, est donc approximative et peut générer des surcharge, mais avec un trafic suffisament proche au trafic réel, on peut aboutir à des résultats satisfaisants.

2.2 Application des algorithmes

Dans cette partie, on va implémenter les diérents algorithmes. On commence par applique l'algorithme k-means et le clustering hiérarchique sur les données géographiques sans prendre en considération le trafic. Ensuite, on applique la version de DCCA proposée dans [3] sur les données du trafic après avoir déterminer les paramètres d'entrée.

2.2.1 Clustering sur les Données géographique

On applique l'algorithme de k-means, et le clustering hiérarchique sur les données géographiques des 4 villes: Paris, Lille, Nantes, Lyon.

K-means

La mesure de distance utilisée est la distance Euclidienne et on utilise les 2 critères suivants pour évaluer le resultat de clustering, c'est à dire la mesure de la "qualité" des clusters obtenus :

1. Compacité D(C_k) : qui représente la distance maximale enter deux RRHs du même cluster. Que l'en souhaite diminuer.
2. Séparabilité S(P) : qui représente la distance minimale entre les centroïdes des clusters. Que l'en souhaite augmenter.

Pour cela on utilise l'index de DUNN, qui prend en considération les deux critèresn :
I_{DNN} =D(P)/S(P) avec D(P)=max $D(C_k)$
Les résultats obtenus :

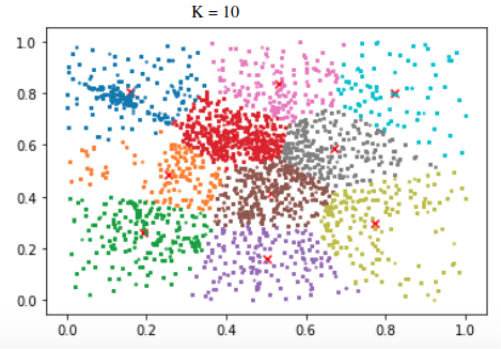

Figure 2.23: K-means Paris, K =20

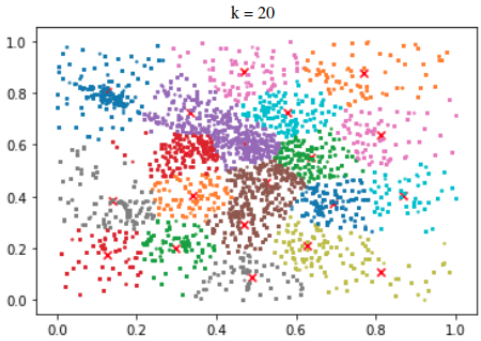

Figure 2.24: K-means Paris, K =10

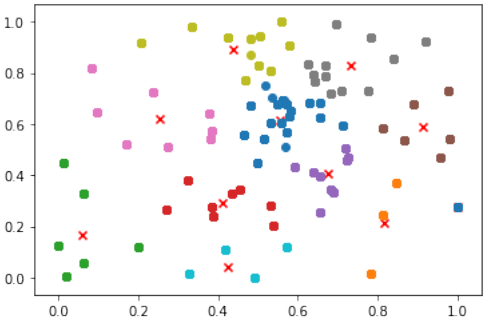
Figure 2.25: K-means Lille, K =10

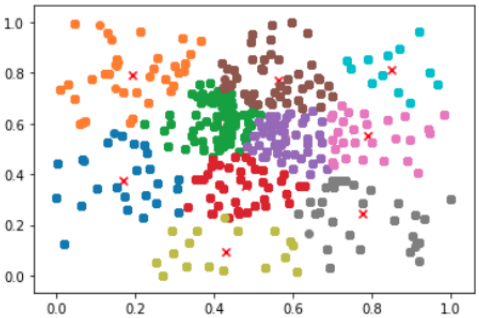
Figure 2.26: K-means Lyon, K =10

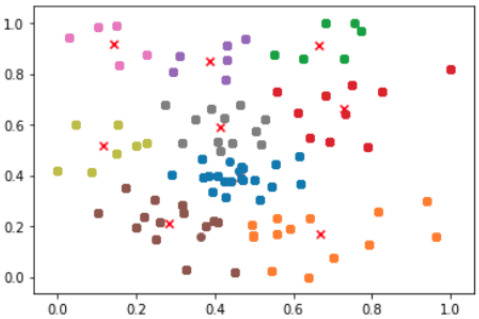
Figure 2.27: K-means Nantes, K=10

Clustering hiérachique

Après exécution de l'algorithme sur les positions de rrhs de la ville de Lille, on obtient une matrice de fusion successive des clusters.
La figure qui suit représente le dendrogramme traduisant cette matrice:

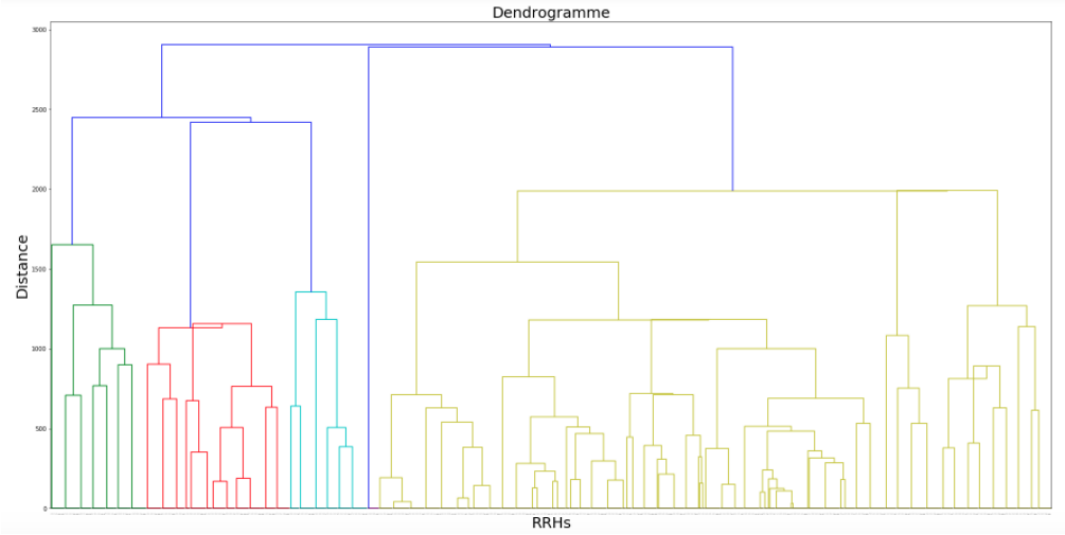

Figure 2.28: Dendogramme de la ville Lille

On peut obtenir le nombre de clusters voulu selon une coupure horizontale.

Les algorithmes précédents ne prennent pas en considération le trafic généré. Un cluster est déterminé par l'approximation géographique de ses RRHs. En général, les RRHs situés au même région ont des schémas du trafic similaire et donc non-complémentaires.

2.2.2 Clustering sur les Données du trafic

On applique DCCA sur le trafic downlink de la ville Lille avec des TimeSlots d'une heure. Pourcela on considère la moyenne du trafic par heure. Dans un premier temps, on applique le DCCA sur les 24 TimeSlots, c'est-à-dire on fait un seul clustering par jour. Cela va servir à determiner une valeur optimale de τ. Dans un deuxième temps, on considère des tranches du temps et on obtient un clustering par tranche. À la fin, on compare les clusters durant un jour de semaine et durant le weekend.

DCCA avec 24 TimeSlots

On applique DCCA sur les 24 TimeSlots et on varie la valeur de τ en fixant la capacité de la BBU à 10 unités.
Les figures suivantes montrent les clusters générés pour chaque valeurs de τ :

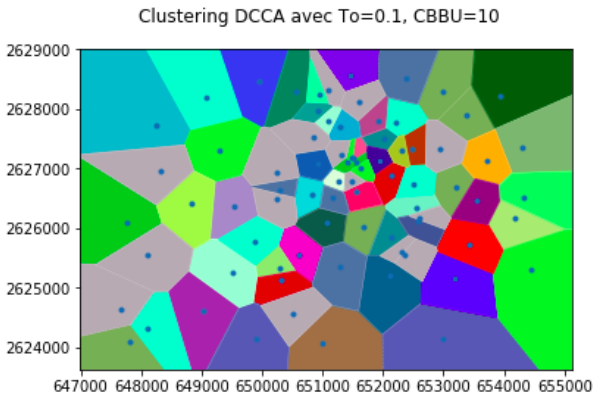

Figure 2.29: Clusters de la ville Lille (=106)

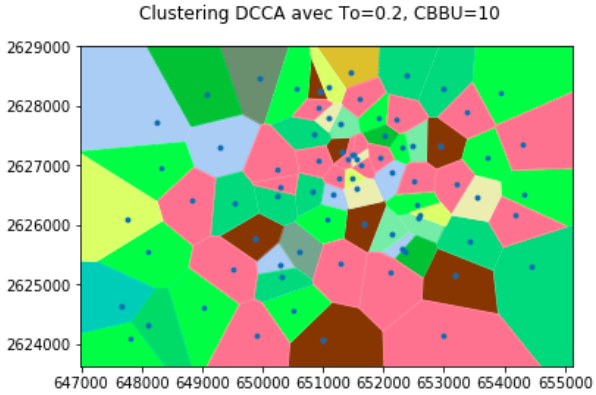

Figure 2.30: Clusters de la ville Lille (=31)

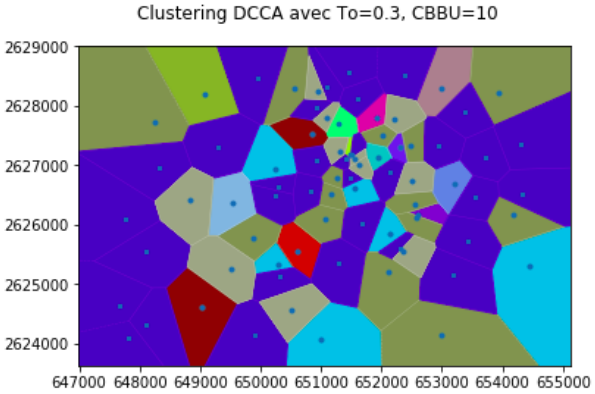

Figure 2.31: Clusters de la ville Lille (=16)

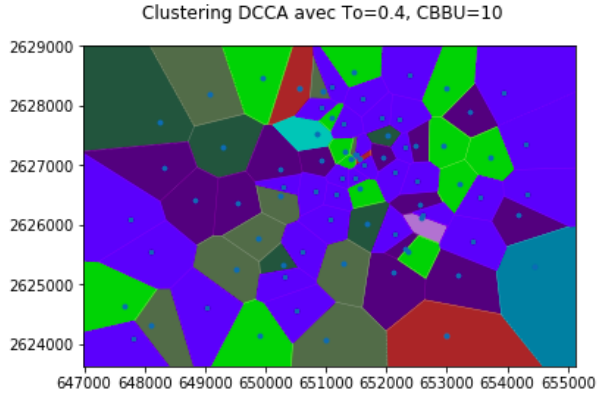

Figure 2.32: Clusters de la ville Lille (=13)

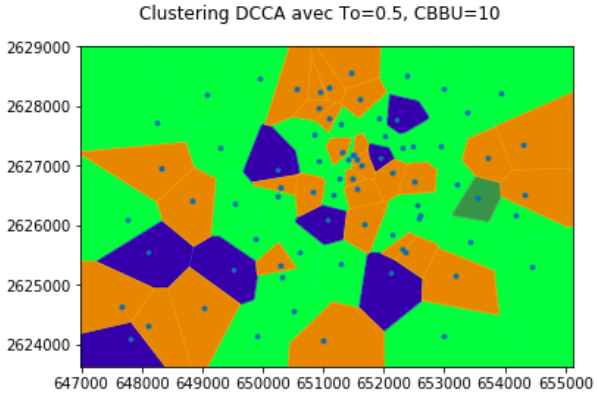

Figure 2.33: Clusters de la ville Lille (=7)

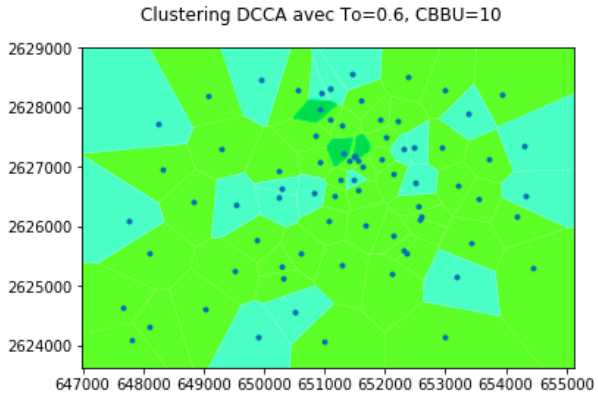

Figure 2.34: Clusters de la ville Lille (=6)

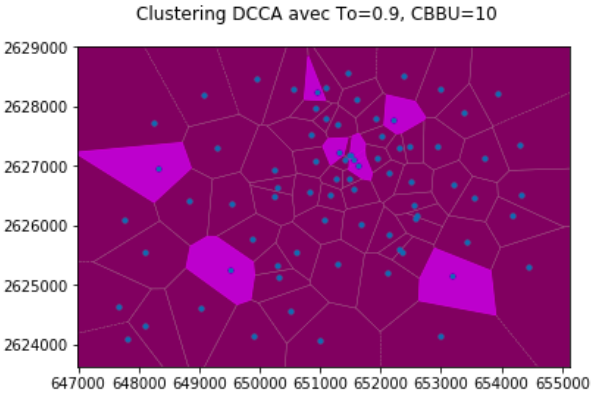

Figure 2.35: Clusters de la ville Lille (=5)

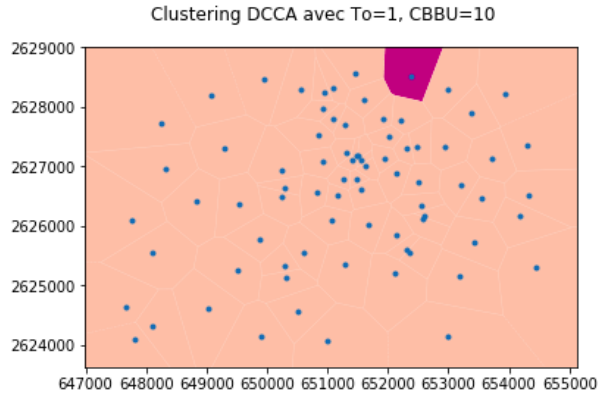

Figure 2.36: Clusters de la ville Lille (=3)

En augmentant la valeur de τ, on relache la contrainte sur la distance se qui parmet d'avoir plus de RRHs dans un seul cluster. Pour avoir un nombre accepltable de clusters et pour une meilleure visualisation, on fixe τ à 0.25 pour la suite de cette section.

Pour la Capacité de la BBU, en utilisant la deuxième méthode de la partie (2.1.2), on la fixe à CBBU = 8.

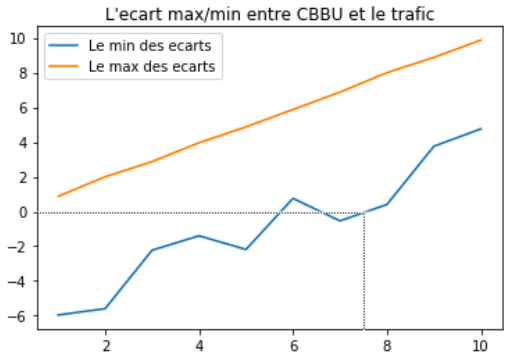

Figure 2.37: Valeur expérimentale de la CBBU

DCCA sur des tranches de 8 TimeSlots

Maintenant, on applique le DCCA sur les intervalles du temps suivants: [00:00,08:00[, [08:00,12:00[, [12:00,16:00[,[16:00,00:00[pour un jour de semaine et durant le weekend.

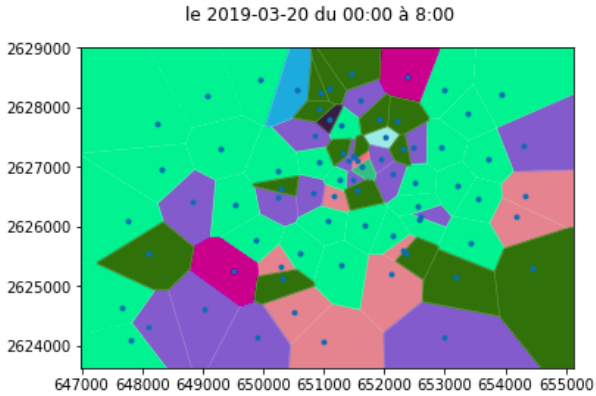

Figure 2.38: Jour de semaine

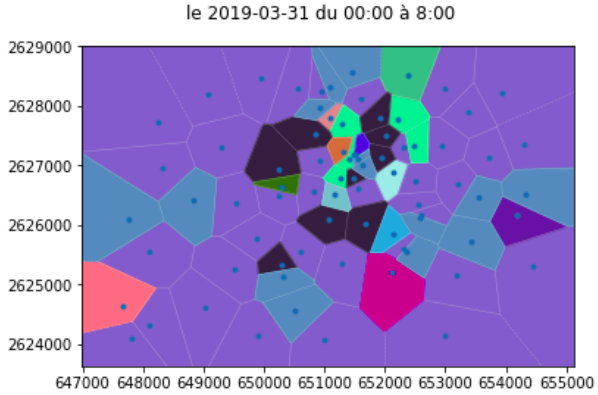

Figure 2.39: Weekend

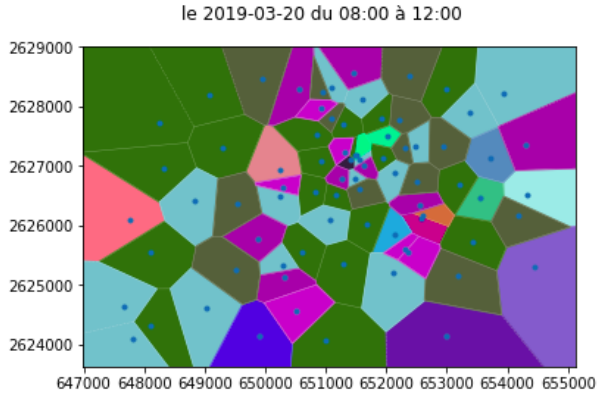

Figure 2.40: Jour de semaine

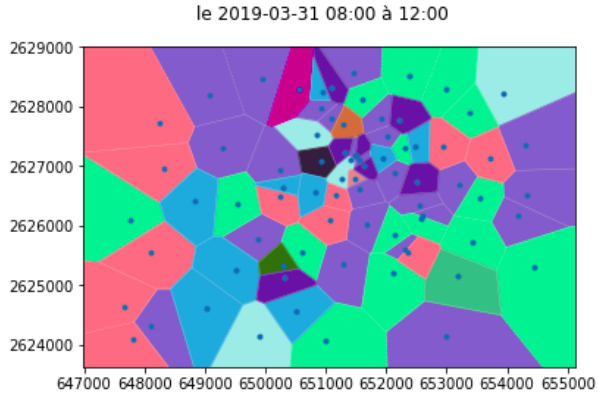

Figure 2.41: Weekend

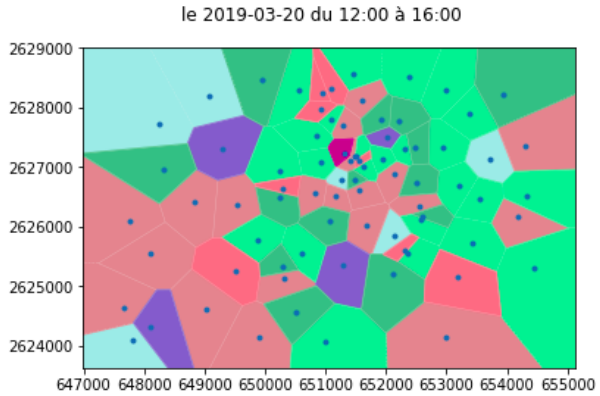

Figure 2.42: Jour de semaine

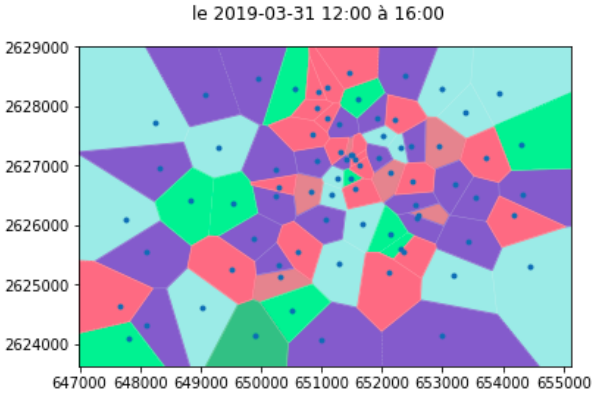

Figure 2.43: Weekend

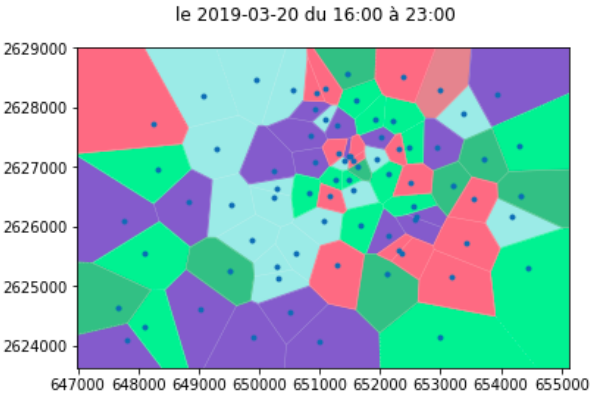

Figure 2.44: Jour de semaine

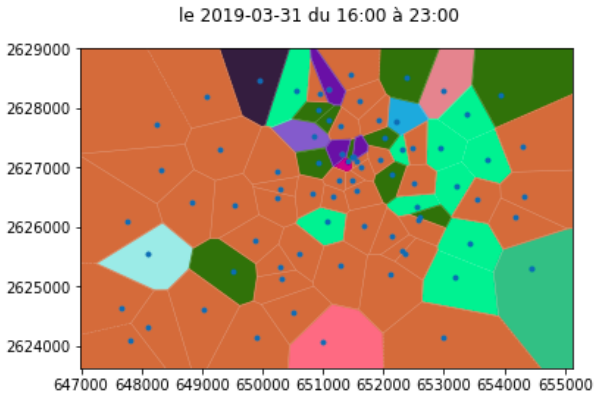

Figure 2.45: Weekend

2.3 K-means Vs DCCA

Après avoir appliqué les algorithmes DCCA et k-means sur les données de la ville de Lille, on compare leurs performances en terme d'utilité de la capacité de la BBU. Pour cela, on calcule le gain en multiplexage statique :
SMG $= \frac{\sum_{s=1}^{N} CBBU(C_s)}{CBBU(pool)}$
Tel que N est le nombre des stations de base, $CBBU(C_s)$ est la capacité de la BBU nécessaire pour soutenir la station de base s et CBBU(pool) est la capacité de la BBU nécessaire pour le pool des BBUs.
Sur le trafic du jour 2019-05-23, avec K=10 pour les K-means, le gain en multiplexage statique est de 1.25 tant que le DCCA donne un gain de 1.31, soit une amélioration de 4.8% . Les clusters formés par DCCA étants complémentaires, la capacité requise pour chaque cluster est donc inférieure à celle requise pour les clusters de k-means, ce qui permet cette amélioration.
En utilise une autre mesure pour comparer la charge du trafic dans les clusters. Pour cela, pour des valeurs différentes de la CBBU, on calcule le ratio:
$\frac{\max Trafic_i}{CBBU(pool)}$
tel que $Trafic_i$ est le trafic généré par le cluster i. Ce ratio permet d'evaluer la charge dans les clusters. Un ratio supérieur à 1 imaplique une surcharge; un ratio inférieur à 1 implique une sous-charge. L'idéal est d'avoir une valeur proche à 1.

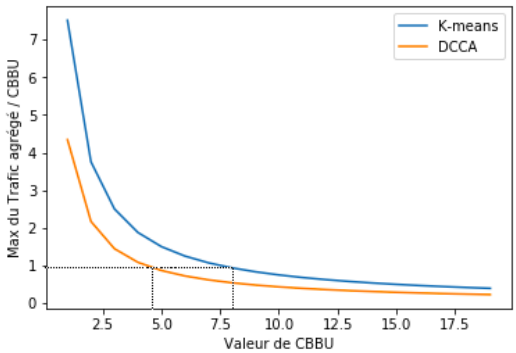

Figure 2.46: Evaluation de la charge

La figure ci-dessus montre que le capacité de la BBU nécessaire pour atteindre un équilbre est inferieure dans le clustering DCCA à celle de K-means.