Viven BERNARD-NICOD 3MICF Pol-Elouan BRIENT

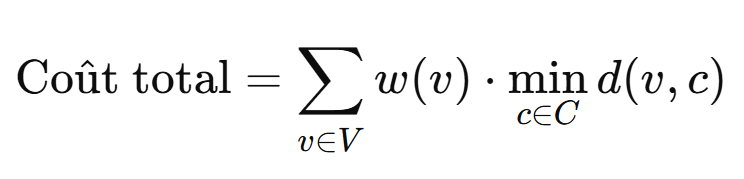
**BE GRAPHES**

**PROBLEME OUVERT**

Sujet :

Pour ce BE, nous avons opté pour le problème “centres de graphes” dans lequel nous plaçons k points sur la carte de telle sorte que la pire distance/pire temps d’un sommet à un des points soit minimale.

Globalement, nous cherchons à optimiser :



Où :

V = ensemble de sommets,

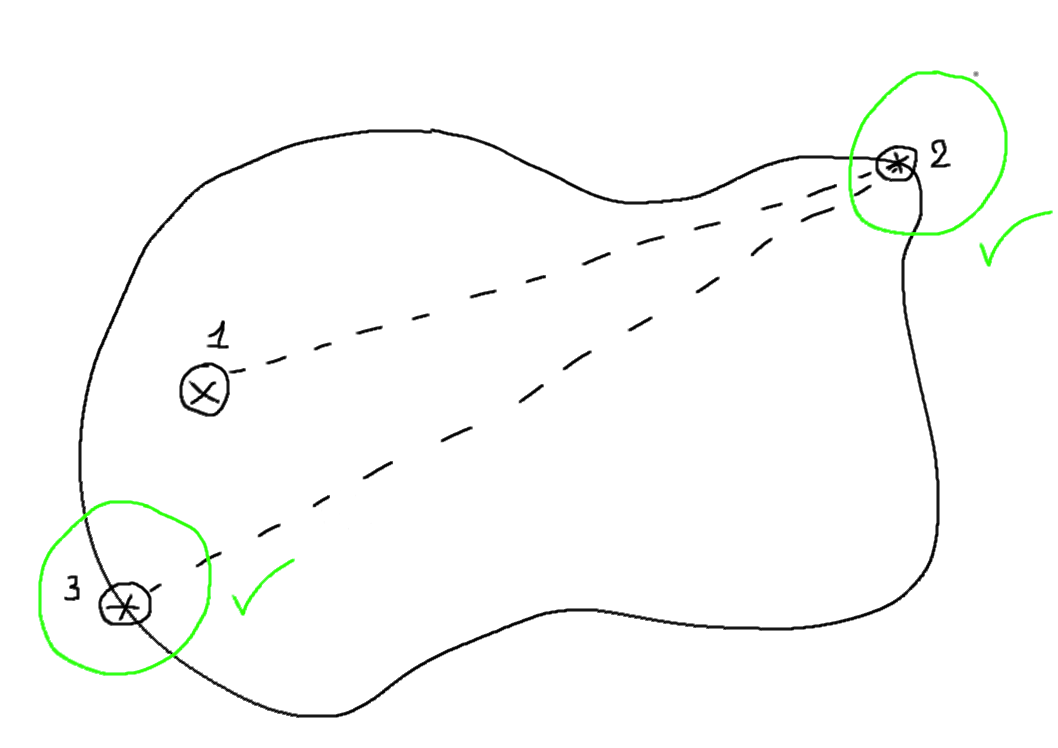
C = ensemble des centres à trouver

d(v,c) = distance/temps entre le sommet et un centre

w(v) = population par sommet (non utilisé pour l’instant donc à 1 par défaut => chaque sommet = poids de 1)

Première idée :

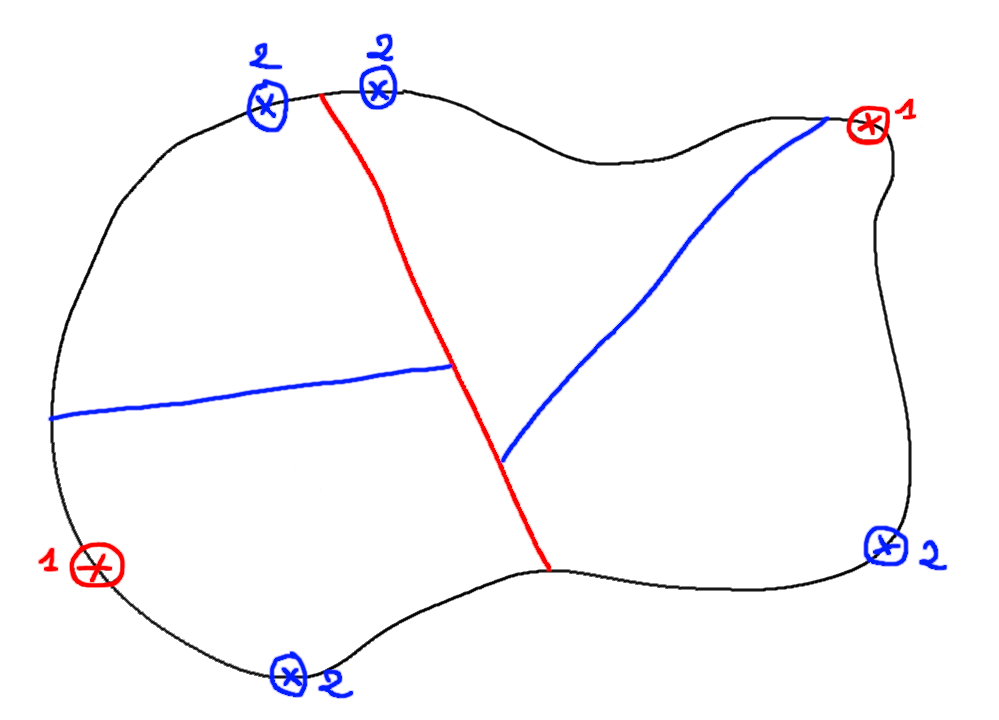
Nous avons d’abord pensé simple, avec deux centres. Nous avons eu l’idée de prendre un point au hasard sur la carte, faire la distance maximale entre ce point et tous les sommets du graphes. Nous avions alors une extrémité de la carte. Ensuite il suffisait de refaire une distance maximale depuis ce nouveau point et nous avions les 2 centres d’un bout et l’autre de la carte.



Cette solution ne couvrant que 2 points, il nous fallait trouver une autre solution, nous avons alors pensé à la découpe de carte....

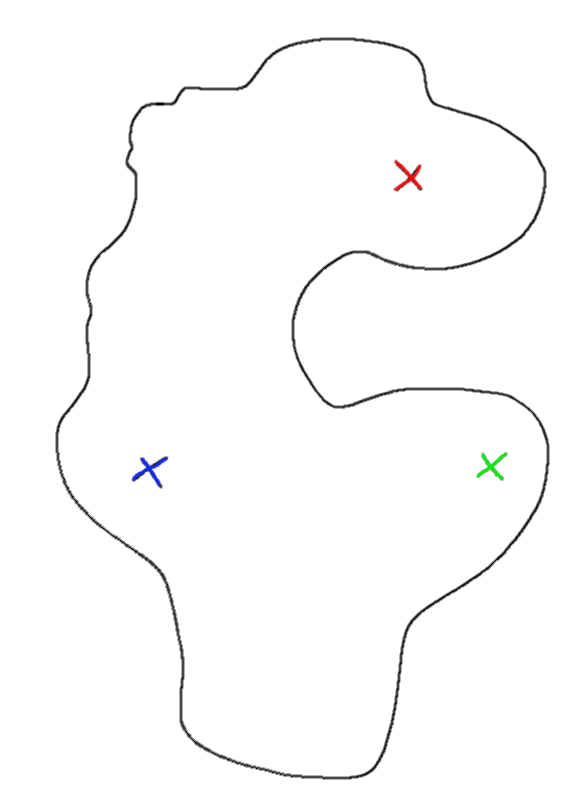
2ème idée :

Le principe était de reprendre algorithme précédent, mais en formant deux groupes à partir des deux centres créés, de cela on pouvait recommencer cet algorithme récursivement dans chaque groupe pour créer des sous groupes qui créent 2 centres distincts, etc.. voici un exemple pour deux itérations :



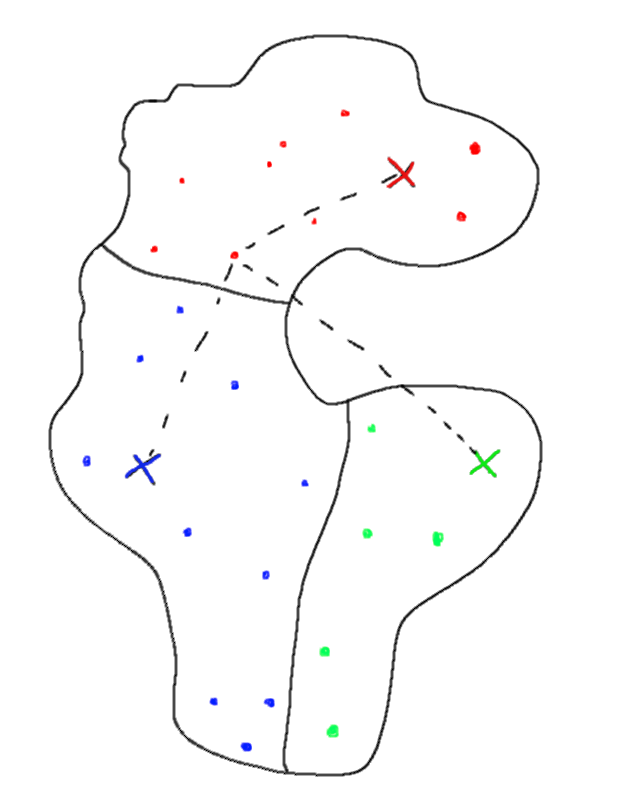
Mais cette solution n’était toujours pas la bonne : nous voyons que cette réalisation, en plus de ne couvrir que des puissances de 2 (2^k magasins), couvrait un nombre maximum de zones de la carte sans prendre en compte la densité de population.

Nous nous somme alors rabattus sur une autre solution, inspirée du cours du premier semestre sur l’analyse de données: le clustering k-means.

Implémentation finale

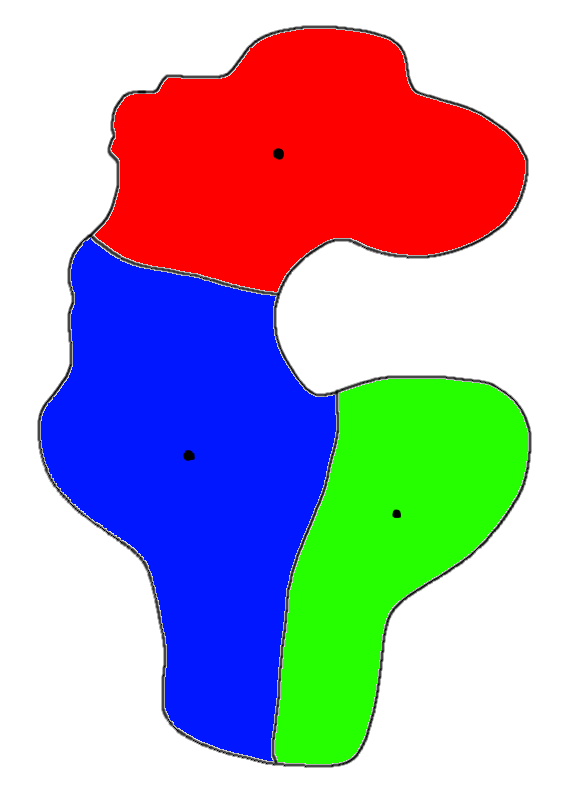
• Initialisation:

Le principe est assez simple: Nous commençons avec un graphe quelconque et prenons k nœuds arbitrairement sur ce graphe, pour l’exemple nous prendrons ici 3 nœuds (qui seront nos centres) correspondant à nos magasins :



• Première itération :

Ensuite, pour tous les points sur le graphe, nous allons regarder quel est le centre le plus proche de lui (en utilisant un Dijkstra ou A\* par exemple) et le mettre dans un groupe correspondant à son centre. Nous aurons donc k groupes :



• Réarrangement :

Après avoir créé nos groupes, nous cherchons le point central dans chaque groupe, c’est à dire le point qui minimise un algorithme de parcours en largeur (en ayant le moins de couches possible) :

• Répétition :

Nous allons faire ce test en boucle jusqu’à ce que la somme du coût du parcours en largeur des k groupes soit égale (avec un seuil de tolérance, plus ou moins grand en fonction du temps que l’on a). C’est à dire qu’a chaque itération, on regarde individuellement le coût que cela prend d’aller d’un centre k à tous les autres nœuds dans son groupe et on somme ces coûts pour chaque groupe. On arrête le programme quand les sommes des coûts sont égales entre tous les groupes (avec un delta de, par exemple, 5%).

• Coût :

Cet algorithme, a implémenter, serait assez coûteux et nous en avons conscience. Nous pensons qu’il y a des améliorations à faire à ce niveau, cependant de part son utilisation, qui est un algorithme a utiliser une seule fois pour pouvoir savoir où placer des magasins..., ce problème de coût n’est pas forcément trop impactant.