~

TAKHI Kamil
kamil.takhi@imt-atlantique.net

SZENTES Kristof
kristof.szentes@imt-atlantique.net

PINSON Claire
claire.pinson@imt-atlantique.net

Rapport SER

Réseau Télécommunication - Groupe N1-3 - Sous-groupe 2

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1 Présentation du cas d'étude

Dans notre cas d'étude, nous nous sommes intéressés aux réseaux de télécommunications et leur optimisation. Les données du problème comprennent, un ensemble de clients, un ensemble de bureaux de liaison (end offices) et un ensemble de hubs numériques (digital hubs), tous positionnés sur un plan. On va chercher à créer un cycle de hubs numériques auxquels on va relier certains bureaux de liaisons auxquels on va relier les clients. Les bureaux de liaison on tous une capacité, ils ne peuvent pas être reliés à tous les clients et il en va de même pour les hubs numériques. Chaque liaison entre ces différents éléments et chaque choix de hub numérique a un coût, on va chercher à minimiser le coût total tout en desservant un taux supérieur ou égal à α de clients.

2 Modélisation sous la forme d'un MILP

On commence par définir les ensembles et les paramètres du problème:

- C,M et N sont respectivement les ensembles des clients, des end office et des hubs numériques. (On notera C,M et N les cardinaux de ces ensembles par la suite pour simplifier les notations)
- $h_{i,j}$ est le coût pour relier le client i au end office j.
- $c_{j,k}$ est le coût pour relier le end office j aux hub numérique k
- $g_{k,m}$ est le coût pour relier les hubs numériques k et m entre eux
- U_j^{max} est la capacité du end office j (c'est le nombre de clients qu'il peut servir)
- V_k^{max} est la capacité du hub numérique k (c'est le nombre de clients qu'il peut servir indirectement)

On ajoute à ces paramètres, des variables de décision qui permettent de formaliser les contraintes par la suite. Ils sont définis comme suit:

- $X_{i,j}$ qui vaut 1 si le client i est relié au end office j.
- $Y_{j,k}$ qui vaut 1 si le end office j est relié au digital hub k et 0 sinon.
- $Z_{k,m}$ qui vaut 1 si le hub numérique k est relié au hub numérique m et 0 sinon. On ne prend que chaque liaison une fois, si $Z_{k,m} = 1$ alors $Z_{m,k} = 0$.
- L_k qui vaut 1 si le hub numérique k est utilisé et 0 sinon.

La fonction objective devient alors:

$$Z = \underbrace{\sum_{i=0}^{C}\sum_{j=0}^{M} X_{i,j}h_{i,j} + \sum_{j=0}^{M}\sum_{j=0}^{N} Y_{j,k}c_{j,k}}_{\text{liaisons clients-bureaux de liaison et bureaux de liaison-hubs digitaux}} + \underbrace{\sum_{k=0}^{N} L_k f_k}_{\text{choix des hubs numériques}} + \underbrace{\sum_{i=0}^{N}\sum_{j=0}^{N} Z_{k,m}g_{k,m}}_{\text{laisions entre hubs numériques}}$$

Ensuite, on cherche à formaliser toutes les contraintes du problème:

- **Contrainte 1**: chaque client doit être lié au plus un seul end office

$$\forall j, \sum_{i=0}^{M} X_{i,j} \leq 1$$

- **Contrainte 2**: chaque end office doit être relié à un et un seul hub numérique

$$\forall j, \sum_{k=0}^{N} Y_{i,j} = 1$$

- **Contrainte 3**: Un end office ne peut être relié à un hub numérique que si celui-ci a été choisi.

$$\forall k, \forall j, Y_{j,k} \leq L_k$$

- **Contrainte 4**: Chaque hub numérique doit être relié à exactement deux autres hubs numériques (cela permet d'avoir une structure cyclique pour les hubs numériques)

$$\forall k, \sum_{m=0}^{N} (Z_{m,k} + Z_{k,m}) = 2L_k$$

- **Contrainte 5**: On doit avoir un seul cycle de hubs numériques.
Pour vérifier cela, on considère $\mathcal{N}$ l'ensemble des hubs et des $\mathcal{H}$ sous-ensembles ($\subset \mathcal{N}$ de hubs de tailles supérieures à 3). On essaie de voir à partir de là si y a une boucle formée dans $\mathcal{H}$. On doit avoir alors la somme liaisons dans H qui doit être inférieure au nombre de hubs sélectionnés dans $\mathcal{H}$ si il y a d'autre hubs sélectionnés non présents dans $\mathcal{M}$:

$$\forall l \in \mathcal{H}, \forall t \in \mathcal{N} - \mathcal{H}, \qquad \sum_{i,j \in H} Z_{i,j} \leq \sum_{k \in H - \{l\}} L_k + 1 - L_t$$

- **Contrainte 6**: Chaque end office a une capacité et ne peut accueillir qu'un certain nombre de clients.

$$\forall j, \sum_{i=0}^{C} x_{i,j} \leq U_j^{max}$$

- **Contrainte 7**: Chaque hub numérique a également un capacité maximale concernant les nombre de clients qu'il peut indirectement accueillir.

$$\forall k, \sum_{j=0}^{M} y_{j,k} \sum_{i=0}^{C} x_{i,j} \leq V_j^{max}$$

- **Contrainte 8**: Un cycle de hubs numériques doit en contenir au moins 3.

$$\sum_{k=0}^{N} L_k \geq 3$$

- **Contrainte 9**: On doit fournir une connection à au moins une part α de la population.

$$\frac{\sum_{j=0}^{M} \sum_{i=0}^{C} x_{i,j}}{C} \geq \alpha$$

- **Contrainte 10**: Les variables décisionnelles doivent toutes appartenir à $\{0, 1\}$.

3 Résolution du "petit" problème

La résolution du "petit" problème se fait sans soucis à l'aide de la libraire pulp de Python, notre code est disponible dans le fichier petit_probleme_pulp.py en annexe. Il ne fait que charger les éléments du fichier xlsx, définir les contraintes décrites ci-dessus et demander à pulp de résoudre le problème. On trouve avec cette méthode, comme solution optimale une solution pour laquelle la valeur de la fonction objective est 5519.

4 Choix pour la meta-heuristique ILS

Etant donné la taille des problèmes réels que l'on devra traiter, il est impossible (ou du moins bien trop long) d'utiliser la librairie pulp de python, ou des algorithmes à la main tel que le simplex. On doit donc opter pour de la méta-heuristique afin d'avoir un résultat satisfaisant le plus rapidement possible.
On a donc fait le choix d'une méta-heuristique ILS. Le but est de prendre une solution initiale aléatoire au début, puis à l'aide des fonctions d'intensification, chercher le miminum local de la fonction que l'on cherche à optimiser (le prix total). Une fois ce mimum trouvé (ou du moins approché suffisamment), on utilise des fonctions de diversification afin d'échapper au minima locaux (cf allégorie de la montagne). Il faut que ces changements soient suffisamment "violents" pour que l'on soit (presque) sûrs de changer de localité. On cherche alors de nouveau le minimum local, et ainsi de suite.

5 Description de l'implémentation de l'ILS

Voici la description, en pseudo-code, de notre algorithme ILS.
On utilise la structure indiqués dans le document localSearchTelecom.pdf, chaque solution consiste en 3 listes. La première (souvent appellée L_ce ou Lce) indicée par les clients, indique quel end office est utilisée par chaque client et la valeur 0 indique que le client en question n'est pas desservi.
La seconde (souvent appellée Led ou L_ed) indique pour chaque end office, le numéro du digital hub auquel il est relié.
La dernière (souvent appellée Ldd ou L_dd) indique quels digitals hubs sont utilisés (ceux qui ont une valeur non nulle associée) et pour chaque digital hub utilisé, elle indique le numéro du prochain dans le cycle.
Voici les fonctions qui constituent notre algorithme:

- init_random(): crée une solution initiale aléatoire
- score(Lce,Led,Ldd): calcule le score (la valeur de la fonction objective) pour la solution (Lce,Led,Ldd).
- verif(Lce,Led,Ldd): vérifie si les 3 solutions constituent bien une solution valable (sachant que beaucoup de contraintes sont toujours vérifiés de par notre représentation des solutions).
- swap_ce(Lce), swap_ed(Led), swap_dd(Ldd): ces fonctions effectuent des swaps sur les 3 listes.
- insertion_ce(Lce), insertion_ed(Lce,Led,Ldd) et insertion_dd(Lce,Led,Ldd): ces fonctions effectuent des insertions sur les 3 listes, elles remplacent un élément de la liste par un autre élément aléatoire et permettent ainsi de diversifier nos solutions.
- est_plein_end_office(k,Lce), est_plein_digital_hub(k,Led,Lce), est_utilise_digital_hub(k,Ldd) et liste_hub_utilises(Ldd) sont des fonctions utilitaires qui servent pour la programmation d'autres fonctions. Leur nom présente très bien à quoi elles servent.
- end_office_aleatoire(Lce) et digital_hub_aleatoire(Lce,Led): sont des fonctions utilitaires qui trouvent respectivement un end office ou un digital hub aléatoire que l'on a le droit d'insérer (en considérant simplement les capacités) dans les listes.
- neighbor(Lce,Led,Ldd) : cette fonction effectue un swap et une insertion sur chacune des 3 listes, il renvoie la nouvelle solution que si elle est valable et que son score est inférieur au score initial.
- intensification(Lce,Led,Ldd): cette fonction applique la fonction neighbor(Lce,Led,Ldd) jusqu'à ce qu'on atteigne un minimum local. On atteint un minimum local si notre solution ne s'améliore pas après 50 tentatives de neighbor(Lce,Led,Ldd).
- perturbation(Lce,Led,Ldd): Cette fonction effectue un certain nombre de swaps et d'insertions sans regarder si la solution s'améliore ou pas. Cela nous permet de s'échapper des minimums locaux. Dans notre version, on a perturbation_v1 qui effectue ceci, et perturbation_v2 qui en plus de ceci effectue des "double-bridge" (à l'aide d'une fonction double_bridge(Lce, Led, Ldd)) qui sont une autre facon de faire varier fortement nos solutions.
- main(): C'est la fonction principale, elle utilise les autres fonctions pour créer une solution initiale, chercher le minimum local, puis la perturber, puis chercher à nouveau un minimum local puis la reperturber ... etc. Elle effectue cela dix mille fois (nombre que l'on peut faire varier) avant de retourner la solution finale qui est la meilleure qu'elle a trouvée.

Le code de notre ils se trouve dans le fichier ils.py, mais le code pour lire le fichier excel se trouve dans read_excel.py, il faut donc l'avoir à côté du premier fichier pour que celui-ci fonctionne.

6 Résolution du "petit" et du "grand" problème

Pour le "petit" problème, on trouve systématiquement, en effectuant dix mille intensifications et perturbations, un résultat tel que l'écart entre la solution exacte (5119) et notre solution est inférieure à 100. On a donc **une erreur de moins de 2%** sur notre résultat en général.

Pour le "grand" problème,après plusieurs tentatives, le meilleur résultat que l'on trouve est de **11592**.