Changez de CAP et prenez des couleurs, Méthodes d'optimisation combinatoire appliquées au Channel Assignment Problem

Tristan Roussillon

31 mars 2021

1 Définition du problème

Le problème d'allocation (ou assignation ou attribution) de fréquences consiste à répartir un ensemble de fréquences sur les stations de base d'un réseau tout en minimisant le nombre de fréquences utilisées. Il est habituellement appelé *Channel assignment problem* (CAP) ou encore *minimum span frequency assignment problem* (MSFAP) en anglais, mais possède plusieurs variantes. Le but de cette section est de poser une version de ce problème de manière rigoureuse [1].

Nous considérons un ensemble de noeuds V. Ils représentent des émetteurs. Nous avons une fonction $p : V \to \mathbb{N}$ qui associe à chaque noeud, un nombre entier appelé *demande*. C'est le nombre de fréquences distinctes souhaité pour un émetteur. Il y a un ordre total sur les fréquences de sorte qu'elles peuvent être représentées par un ensemble d'entiers consécutifs $\{1, \ldots, t\}$. Nous avons de plus une fonction $l : V \times V \to \mathbb{N}$ qui associe à chaque paire de noeuds, un nombre entier qu'on peut appelé *poids* ou *longueur*. Cette fonction décrit les contraintes d'interférence entre les émetteurs : les fréquences utilisées par deux émetteurs v_1 et v_2 sont contraintes d'être distantes de $l(v_1, v_2)$ ou plus afin d'éviter toute interférence. Remarquons enfin que V et l définissent implicitement un graphe $G(V, E)$, où E est défini comme l'ensemble des paires de poids non nul $\{(v_1, v_2) \in V \times V \mid l(v_1, v_2) > 0\}$.

Une *solution* est une fonction $\phi : V \to \mathcal{P}(\{1, \ldots, t\})$ telle que le nombre d'éléments de $\phi(v)$ est égal à $p(v)$ pour chaque $v \in V$ [2]. Une solution est *faisable* si pour toute paire $(v_1, v_2) \in V \times V$, tout $f_1 \in \phi(v_1)$ et tout $f_2 \in \phi(v_2)$, on a $|f_1 - f_2| \geq l(v_1, v_2)$. L'intervalle (*span*) du problème, noté $\text{span}(V, p, l)$, est le plus petit t tel qu'il existe une solution faisable. Étant donnés V, p, l, le problème (CAP) consiste à déterminer $\text{span}(V, p, l)$ ainsi qu'une solution faisable ϕ.

Exemple 1. *G est un cycle de 3 noeuds, qui ont chacun une demande de* 1. *Les 3 arêtes sont de poids* 3. *Le span est* 7.

Exemple 2. *G est un cycle de 4 noeuds, qui ont chacun une demande de* 1. *Les 4 arêtes sont de poids* 3. *Le span est* 4.

1. Vous avez dans ce document dont nous reprenons plusieurs éléments, une description un peu plus approfondie des aspects de télécommunication mis en jeu.

2. La notation $\mathcal{P}$ désigne l'ensemble des parties d'un ensemble. Pour chaque $v \in V$, $\phi(v)$ est donc un sous-ensemble des fréquences $\{1, \ldots, t\}$

Exemple 3. *G est un cycle de 5 noeuds avec en plus une boucle sur chaque noeud. Chaque noeud a une demande de* 2*. Les boucles ont un poids de* 2*, tandis que toutes les autres arêtes ont un poids de* 1*. Le span est* 5*.*

Exercice 1.1. Expliquez comment on peut transformer tout (CAP) faisant intervenir des demandes supérieures à 1 en un (CAP) dont la demande est 1 pour chaque noeud. Illustrez avec l'exemple 3.

Grâce à cette transformation, nous pouvons maintenant considérer sans perdre en généralité que la demande est 1 pour chaque noeud.

Exercice 1.2. La coloration de graphe consiste à trouver le plus petit nombre de couleurs permettant d'attribuer une couleur à chacun des noeuds d'un graphe, tout en garantissant que deux noeuds reliés par une arête sont de couleur différente. Expliquez sous quelles conditions (CAP) correspond à un problème de coloration de graphe. Illustrez avec un exemple de votre choix.

La coloration de graphe est un problème NP-complet. Or, (CAP) étant au moins aussi difficile que la coloration de graphe d'après la question précédente, on conclut que (CAP) est aussi un problème NP-complet.

2 Résolution du problème

Même si (CAP) est NP-comlet, on peut résoudre certaines instances de ce problème. Vous avez par exemple pu résoudre des instances de (CAP) proposées dans les exemples 1, 2 et 3.

Exercice 2.1. Implémentez [3] un sous-ensemble de ces algorithmes (par ordre de difficulté et points dans la note finale) afin de résoudre une instance de (CAP) :

1. *greedy* (glouton) : on calcule progressivement une solution en ne considérant qu'une nouvelle variable à la fois et sans remettre en cause les affectations précédentes. Bien sûr, il n'y a aucune garantie d'obtenir à la fin une solution optimale, mais c'est un début quand on s'attaque à un nouveau problème.
2. *backtracking* (retour sur trace) : à chaque fois qu'on choisit une nouvelle variable, et pour chaque affectation possible de cette variable, on teste récursivement si une solution faisable peut être construite à partir de cette affectation partielle. Si aucune solution n'est trouvée, on revient sur les affectations qui ont été faites précédemment (d'où le nom de retour sur trace).
3. *branch-and-bound* (séparation-évaluation) : à chaque fois qu'on choisit une nouvelle variable, et pour chaque affectation possible de cette variable (séparation), on détermine un minorant de la fonction objectif à partir de cette affectation partielle (évaluation), afin de ne tester que les cas où le minorant est suffisamment petit par rapport à ce qui a déjà été calculé. Le plus délicat est de définir la fonction d'évaluation, mais on a la garantie d'avoir à la fin l'optimum exact sans avoir tout énuméré.

3. les langages acceptés sont Python, Java, C, C++.

Exercice 2.2. Donnez le résultat de l'exécution de vos programmes (temps et solution) pour les exemples jouets 1, 2 et 3, ainsi que pour les 9 instances du jeu de données Philadelphia décrites également en annexe.

Vous devez rendre une archive contenant vos programmes et un court document répondant aux questions du sujet et expliquant vos choix d'implémentation.

A Jeu de données Philadelphia

Considérons un ensemble de 21 noeuds $V = \{v_1, \ldots, v_{21}\}$.

Considérons les demandes décrites par les listes suivantes (le i-ème élément est la demande pour le i-ème noeud) :

(L1) $(8, 25, 8, 8, 8, 15, 18, 52, 77, 28, 13, 15, 31, 15, 36, 57, 28, 8, 10, 13, 8)$

(L2) $(5, 5, 5, 8, 12, 25, 30, 25, 30, 40, 40, 45, 20, 30, 25, 15, 15, 30, 20, 20, 25)$

(L3) $(20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20)$

(L4) $16, 50, 16, 16, 16, 30, 36, 104, 154, 56, 26, 30, 62, 30, 72, 114, 56, 16, 20, 26, 16)$

(L5) $(32, 100, 32, 32, 32, 60, 72, 208, 308, 112, 52, 60, 124, 60, 144, 228, 112, 32, 40, 52, 32)$

Considérons les longueurs décrites par les matrices suivantes (à l'intersection de la i-ème ligne et j-ème colonne, on a la longueur de l'arête reliant v_i et v_j, 0 indiquant l'absence d'arête) :

(M1)
```
5 2 1 1 0 1 5 5 1 1 0 0 1 1 1 1 1 0 1 1 0
2 5 2 1 1 1 1 5 5 1 1 0 0 1 1 1 1 1 1 1 1
1 2 5 2 1 0 1 1 5 5 1 1 0 0 1 1 1 1 1 1 1
1 1 2 5 2 0 0 1 1 5 5 1 0 0 0 1 1 1 0 1 1
0 1 1 2 5 0 0 0 1 1 5 5 0 0 0 0 1 1 0 0 1
1 1 0 0 0 5 2 1 1 0 0 0 5 5 1 1 0 0 1 0 0
5 1 1 0 0 2 5 2 1 1 0 0 1 5 5 1 1 0 1 1 0
5 5 1 1 0 1 2 5 2 1 1 0 1 1 5 5 1 1 1 1 1
1 5 5 1 1 1 1 2 5 2 1 1 0 1 1 5 5 1 1 1 1
1 1 5 5 1 0 1 1 2 5 2 1 0 0 1 1 5 5 1 1 1
0 1 1 5 5 0 0 1 1 2 5 2 0 0 0 1 1 5 0 1 1
0 0 1 1 5 0 0 0 1 1 2 5 0 0 0 0 1 1 0 0 1
1 0 0 0 0 5 1 1 0 0 0 0 5 2 1 1 0 0 1 0 0
1 1 0 0 0 5 5 1 1 0 0 0 2 5 2 1 1 0 1 1 0
1 1 1 0 0 1 5 5 1 1 0 0 1 2 5 2 1 1 2 1 1
1 1 1 1 0 1 1 5 5 1 1 0 1 1 2 5 2 1 2 2 1
1 1 1 1 1 0 1 1 5 5 1 1 0 1 1 2 5 2 1 2 2
0 1 1 1 1 0 0 1 1 5 5 1 0 0 1 1 2 5 1 1 2
1 1 1 0 0 1 1 1 1 1 0 0 1 1 2 2 1 1 5 2 1
1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 2 2 1 2 5 2
0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 2 2 1 2 5
```

(M2)
```
5 2 1 0 0 1 5 5 1 0 0 0 0 1 1 1 0 0 0 0 0
2 5 2 1 0 0 1 5 5 1 0 0 0 0 1 1 1 0 0 0 0
1 2 5 2 1 0 0 1 5 5 1 0 0 0 0 1 1 1 0 0 0
0 1 2 5 2 0 0 0 1 5 5 1 0 0 0 0 1 1 0 0 0
0 0 1 2 5 0 0 0 0 1 5 5 0 0 0 0 0 1 0 0 0
```

1 0 0 0 0 5 2 1 0 0 0 0 5 5 1 0 0 0 0 0 0
5 1 0 0 0 2 5 2 1 0 0 0 1 5 5 1 0 0 1 0 0
5 5 1 0 0 1 2 5 2 1 0 0 0 1 5 5 1 0 1 1 0
1 5 5 1 0 0 1 2 5 2 1 0 0 0 1 5 5 1 1 1 1
0 1 5 5 1 0 0 1 2 5 2 1 0 0 0 1 5 5 0 1 1
0 0 1 5 5 0 0 0 1 2 5 2 0 0 0 0 1 5 0 0 1
0 0 0 1 5 0 0 0 0 1 2 5 0 0 0 0 0 1 0 0 0
0 0 0 0 0 5 1 0 0 0 0 0 5 2 1 0 0 0 0 0 0
1 0 0 0 0 5 5 1 0 0 0 0 2 5 2 1 0 0 1 0 0
1 1 0 0 0 1 5 5 1 0 0 0 1 2 5 2 1 0 2 1 0
1 1 1 0 0 0 1 5 5 1 0 0 0 1 2 5 2 1 2 2 1
0 1 1 1 0 0 0 1 5 5 1 0 0 0 1 2 5 2 1 2 2
0 0 1 1 1 0 0 0 1 5 5 1 0 0 0 1 2 5 0 1 2
0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 0 5 2 1
0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 2 5 2
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 2 5

(M3) 5 2 1 1 0 2 5 5 2 1 0 0 1 2 2 2 1 0 1 1 0
2 5 2 1 1 1 2 5 5 2 1 0 0 1 2 2 2 1 1 1 1
1 2 5 2 1 0 1 2 5 5 2 1 0 0 1 2 2 2 1 1 1
1 1 2 5 2 0 0 1 2 5 5 2 0 0 0 1 2 2 0 1 1
0 1 1 2 5 0 0 0 1 2 5 5 0 0 0 0 1 2 0 0 1
2 1 0 0 0 5 2 1 1 0 0 0 5 5 2 1 0 0 1 0 0
5 2 1 0 0 2 5 2 1 1 0 0 2 5 5 2 1 0 1 1 0
5 5 2 1 0 1 2 5 2 1 1 0 1 2 5 5 2 1 2 1 1
2 5 5 2 1 1 1 2 5 2 1 1 0 1 2 5 5 2 1 2 1
1 2 5 5 2 0 1 1 2 5 2 1 0 0 1 2 5 5 1 1 2
0 1 2 5 5 0 0 1 1 2 5 2 0 0 0 1 2 5 0 1 1
0 0 1 2 5 0 0 0 1 1 2 5 0 0 0 0 1 2 0 0 1
1 0 0 0 0 5 2 1 0 0 0 0 5 2 1 1 0 0 1 0 0
2 1 0 0 0 5 5 2 1 0 0 0 2 5 2 1 1 0 2 1 0
2 2 1 0 0 2 5 5 2 1 0 0 1 2 5 2 1 1 2 2 1
2 2 2 1 0 1 2 5 5 2 1 0 1 1 2 5 2 1 2 2 2
1 2 2 2 1 0 1 2 5 5 2 1 0 1 1 2 5 2 2 2 2
0 1 2 2 2 0 0 1 2 5 5 2 0 0 1 1 2 5 1 2 2
1 1 1 0 0 1 1 2 1 1 0 0 1 2 2 2 2 1 5 2 1
1 1 1 1 0 0 1 1 2 1 1 0 0 1 2 2 2 2 2 5 2
0 1 1 1 1 0 0 1 1 2 1 1 0 0 1 2 2 2 1 2 5

Les neuf instances $P1 \dots P9$ de ce jeu de données sont décrites à partir des listes et matrices précédentes ainsi :

(P1) $L1, M1$,
(P2) $L1, M2$,
(P3) $L2, M1$,
(P4) $L2, M2$,
(P5) $L3, M1$,
(P6) $L3, M2$,
(P7) $L4, M1$,
(P8) $L1, M3$,
(P9) $L5, M1$.