Optimisation de la consommation énergétique dans les réseaux sans fil fixes

David Coudert, Napoleão Nepomuceno, Issam Tahiri

Optimisation de la consommation énergétique dans les réseaux sans fil fixes. †

D. Coudert[1] and N. Nepomuceno[2] and I. Tahiri[1]

[1] *MASCOTTE, INRIA, I3S (CNRS/Univ. Nice Sophia Antipolis), France.*
[2] *IMADA, Syddansk Universitet, Denmark.*

Nous étudions le problème d'optimisation énergétique dans les réseaux sans fil fixes dans le cas d'une faible demande de trafic par rapport à la capacité du réseau. Nous proposons un programme linéaire pour résoudre le problème, puis nous présentons une heuristique permettant de trouver rapidement une bonne solution.

Keywords: Réseau de collecte, routage, programmation linéaire, densité d'une coupe.

1 Introduction

Notre recherche porte sur l'optimisation de la consommation énergétique dans les réseaux sans fil fixes à travers la mise en veille d'une sélection de ressources réseau. Le réseau comprend des sites distants, chacun d'eux étant servi par une station radio de base (RBS). Ces dernières sont connectées par des liens radio micro-ondes à haut débit. Chaque lien radio bidirectionnel connectant deux RBS requiert une paire dédiée d'unités externes ("outdoor unit", ODUs), chacune couplée directement avec une antenne directionnelle. Peu importe le débit, dès qu'elle est allumée, cette ODU présente une consommation énergétique qui a un impact considérable sur les dépenses opérationnelles du réseau. Dans ce contexte, le réseau a souvent des liens et des chemins redondants afin d'assurer une tolérance aux pannes et pour garantir une capacité suffisante pour répondre au trafic variant de manière non négligeable. Cependant, ceci induit des pertes d'énergie importantes lorsque le besoin de ces ressources supplémentaires n'a pas lieu. Dans ce travail nous considérons le problème d'optimisation qui consiste à décider de la configuration optimale et le routage associé pour minimiser l'énergie totale consommée, tout en répondant aux demandes de trafic. Plus exactement, nous entendons par configuration le choix des ressources à mettre en veille. Nous présentons le programme linéaire permettant une résolution exacte du problème. Il repose sur le modèle de conception d'un réseau à capacités de coûts fixes. Ce type de problèmes étant difficile [3], nous proposons aussi une heuristique pour répondre à la question de manière approchée mais rapide. Finalement nous comparons à travers des simulations les résultats des deux approches.

2 Modélisation du problème et formulation linéaire

La topologie du réseau est modélisée par un graphe orienté $H = (V,E)$ où chaque noeud $v \in V$ représente une station de base et chaque arc $vw \in E$ représente un lien radio. Chaque lien a une capacité c_{vw} et peut être soit actif (il consomme de l'énergie) ou pas. Les demandes de trafic seront définies par un nombre $|D|$ de paires (s^d, t^d), avec $s^d, t^d \in V$ et par un volume moyen de la demande h^d. Par ailleurs, le graphe H est supposé symétrique puisque dans le type de réseau étudié, les liens radio sont toujours symétriques. Ceci implique que pour chaque $v \in V$ le voisinage entrant est le même que le voisinage sortant : $\delta^+(v) = \delta^-(v) = \delta(v)$. Aussi, le coût d'un lien actif est considéré constant quelque soit le volume du trafic qu'il écoule : il est égal à CL. Nous faisons également l'hypothèse (forte) que le trafic d'une demande d peut être routé selon différentes routes entre s^d et t^d. Ceci n'est pas toujours vrai en pratique.

†Financé par le projet APRF RAISOM (PACA & FEDER), l'ANR DIMAGREEN et Villum Kann Rasmussen foundation.

Considérons, pour représenter l'état du lien, la variable de décision binaire u_{vw}, ayant la valeur 1 si le lien vw est actif et 0 sinon. En considérant aussi que x_{vw}^d représente la partie du flot de la demande d qui passe sur l'arc vw, notre problème d'optimisation peut être formulé sous forme d'un programme linéaire à variables mixtes. La fonction objective (1) représente l'énergie totale consommée, et que l'on veut minimiser. Elle compte pour chaque lien le coût CL si et seulement si ce dernier est actif. (2) assurent, pour chaque lien, le fait que la capacité "effective" de ce dernier est suffisante pour écouler la totalité du flot passant par lui. Les contraintes de conservation du flot (3) fournissent les routes pour chaque demande. Enfin, (4) reflètent la symétrie au niveau des arcs du réseau.

$$\min \sum_{vw \in E} CL \cdot u_{vw} \tag{1}$$

$$s.t. \sum_{d=1}^{D} x_{vw}^d \leq c_{vw} u_{vw} \qquad \forall vw \in E \tag{2}$$

$$\sum_{w \in \delta(v)} x_{wv}^d - \sum_{w \in \delta(v)} x_{vw}^d = \begin{cases} -h^d, & \text{si } v = s^d, \\ h^d, & \text{si } v = t^d, \\ 0, & \text{sinon} \end{cases} \qquad \forall v \in V, \forall d = 1, \ldots, |D| \tag{3}$$

$$u_{vw} = u_{wv} \qquad \forall v \in V, \forall w \in \delta(v) \tag{4}$$

$$x_{vw}^d \in [0, h^d] \qquad \forall vw \in E, \forall d = 1, \ldots, |D| \tag{5}$$

$$u_{vw} \in \{0, 1\} \qquad \forall vw \in E \tag{6}$$

Ceci est un programme entier mixte puisqu'il contient des variables réelles et des variables binaires. Ce type de programme peut être résolu par un solveur linéaire tel que "Cplex". Une telle approche donne une solution exacte, mais le temps d'exécution ainsi que la mémoire nécessaire peuvent être énormes. Après avoir essayé en vain différentes formulations du programme linéaire, nous avons conclu que cette méthode ne peut être utilisée raisonnablement que pour des réseaux de petite taille ou des réseaux avec de très lentes évolutions du trafic. Nous avons alors conçu une autre méthode de résolution (voir section suivante) qui ne garantit pas une solution optimale mais qui n'est pas gourmande en terme de mémoire ni en temps.

Dans les simulations, nous avons utilisé une formulation où le trafic est agrégé par source avec utilisation de contraintes adéquates pour extraire la bonne quantité de flot à destination. Ceci permet de réduire le nombre de variables de flots et les contraintes associées.

3 Heuristique basée sur les coupes les moins denses

Nous proposons une heuristique pour choisir de manière plus rapide les liens à mettre en veille. Cette solution suit une approche hybride puisqu'elle combine une heuristique pour décider à chaque itération d'un lien candidat à rajouter aux liens désactivés, ainsi qu'un programme linéaire similaire au programme précèdent ou les u_{vw} ne sont plus des variables mais plutôt des constantes choisies par l'heuristique. Le programme linéaire n'ayant plus que des variables réelles, devient très rapide à résoudre. Cette méthode se base sur les coupes les moins denses d'un graphe. Grosso modo, une coupe n'est pas considérée dense si elle a une charge moyenne élevée par lien. De manière plus formelle : Soit $G = (V, E)$ un graphe non orienté. Soit D un ensemble de paires de noeuds. Nous définissons la capacité c d'un sous-ensemble d'arêtes comme étant la somme des capacités de toutes les arêtes dans ce sous-ensemble. Nous définissons également le volume h d'un sous-ensemble de D comme étant la somme des volumes des demandes présentes dans ce sous-ensemble. Pour chaque coupe séparant S (un sous ensemble de V) de $\bar{S}$, notons $\alpha(S)$ la densité de cette coupe avec : $\alpha(S) = c(E(S, \bar{S}))/h((S * \bar{S}) \cap D)$. Si nous considérons la version uniforme dans laquelle toutes les arêtes ont capacité 1 et à chaque des paires de noeuds correspond une demande de volume unitaire, la densité est alors $\alpha(S) = |E(S, \bar{S})|/|S| * |\bar{S}|$.

Comme la majorité des travaux portant sur la recherche de coupes les moins denses est effectuée dans le cadre des graphes non orientés, nous allons considérer le graphe non orienté sous-jacent en mettant une arête entre v et w si ces derniers sont voisins dans le graphe d'origine avec une capacité égale à $c_{vw} + c_{wv}$.

L'heuristique va donc supprimer des arêtes dans le graphe non orienté et donc des pairs d'arcs symétriques dans le graphe d'origine.

Dans un graphe G, le problème de trouver la coupe de densité minimum (*sparsest cut*) est un problème NP-complet bien connu. De nombreux algorithmes d'approximations ont été proposés [7, 5, 1, 4]. Le plus performant est basé sur la relaxation de la formulation en programme semi-défini positif (*SDP*), et permet d'approcher la meilleure solution du cas uniforme à un facteur $\sqrt{\log n}$ [2]. Nous utilisons donc cette méthode dans nos simulations. En plus, afin d'avoir non une mais plusieurs coupes non denses, nous rajoutons un caractère aléatoire à l'algorithme d'approximation puis nous l'exécutons plusieurs fois.

Algorithm 1 Maximiser le nombre des liens supprimés

Require: Le graphe initial $G_0 = (V,E)$, la capacité c de chaque lien, le volume h de chaque demande, et une liste des coupes non denses considérées *cutsList*.

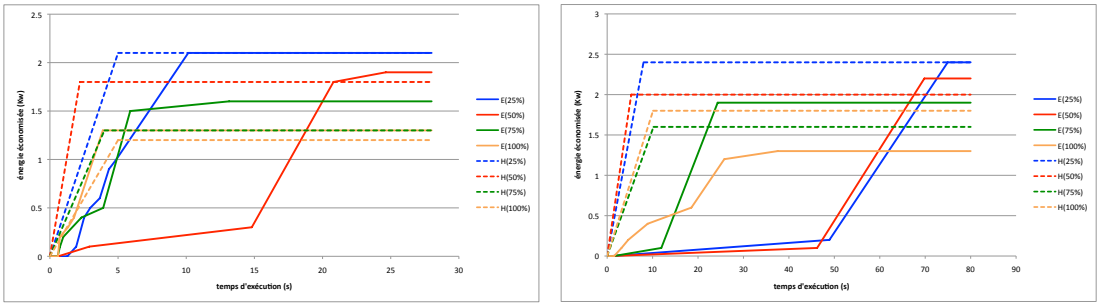
Ensure: Un sous-graphe $G = (V,E')$ où $E' \subseteq E$ permet d'assurer le routage de toutes les demandes.

1: $nonPotentialLinks \Leftarrow \emptyset$
2: $E' \Leftarrow E$
3: **while** $nonPotentialLinks \neq E$ **do**
4: SORT(*cutsList*) /* les coupes sont triées dans l'ordre croissant des densités */.
5: $potentialList \Leftarrow \emptyset$
6: **for** i from 1 to SIZE(*cutsList*) **do**
7: **for** *link* in $cutsList[i]$ and not in $potentialList$ **do**
8: $potentialList \Leftarrow potentialList|link$
9: $potentialList \Leftarrow potentialList|(E - (potentialList \cup nonPotentialLinks))$
10: **for** i from SIZE($potentialList$) to 1 /* dans un ordre LIFO */ **do**
11: **if** ROUTING_LP($E' - potentialList[i]$) is feasible **then**
12: $E' \Leftarrow E' - potentialList[i]$
13: $nonPotentialLinks \Leftarrow nonPotentialLinks + potentialList[i]$
14: **break**
15: **else**
16: $nonPotentialLinks \Leftarrow nonPotentialLinks + potentialList[i]$

La motivation derrière l'utilisation des coupes non denses vient du fait que la densité d'une coupe n'est que l'inverse de la charge moyenne d'un lien de cette coupe. Ceci implique que les liens d'une coupe qui n'est pas dense auront tendance à être chargés à priori. Ainsi pour supprimer des arêtes du graphe tout en gardant une solution de routage pour les demandes, il est préférable de commencer par supprimer les liens qui n'apparaissent pas dans de telles coupes. Plus précisément, nous affecterons à chaque arête une estimation de sa charge qui sera égale à l'inverse de la densité de la coupe la moins dense couvrant cette dernière. L'heuristique essayera de supprimer les arêtes du graphe en suivant l'ordre croissant sur la charge estimée. A chaque itération (tentative de suppression d'une arête), la faisabilité du routage sera vérifiée par l'intermédiaire du programme linéaire validant ou pas la suppression de l'arête en question. Ensuite, et puisque les densités des coupes vont changer, une mise à jour est nécessaire avant de passer à l'itération suivante. Une implémentation efficace de cette heuristique est décrite par le pseudo-code 1 dont la complexité en temps est linéaire en le nombre d'arêtes.

4 Résultats des simulations

Nous avons considéré le même scénario puis comparé les deux approches expliquées auparavant : la formulation exacte (E) utilisant le programme linéaire ainsi que l'heuristique (H) basée sur les coupes de faible densité. Les simulations ont été réalisées sur deux topologies de SNDlib [6] représentant les réseaux coeur de la France (45 liens) et de la Norvège (51 liens). Malheureusement nous n'avons pas trouvé d'instance de réseaux fixes hertziens. Cela dit, les valeurs correspondent à un réseau sans fil fixe pour pouvoir estimer le gain en terme d'économie d'énergie. En ce qui concerne la matrice de trafic nous avons considéré le cas uniforme ; cela veut dire qu'une demande est associée à chaque paire de noeuds avec le



(a) France (b) Norvège

même volume. Pour chaque topologie ce volume prendra plusieurs valeurs $(100\%H, 75\%H, 50\%H, 25\%H)$, H étant le volume maximum réalisable dans un schéma uniforme.

Toutes les simulations ont été exécutées sur la même machine qui a un processeur dont la fréquence d'exécution est 3Ghz avec une limite sur la RAM utilisée de 2Go. Comme solveur de programmes linéaires mixtes, nous avons utilisé Cplex (12.1) auquel nous avons indiqué que la priorité est l'obtention d'une solution proche de l'optimale et non la preuve d'optimalité. Quant à la résolution des SDPs, nous avons utilisé le solveur CSDP. Sur les résultats affichés sur les figures (a) et (b), nous remarquons bien que la méthode exacte donne un meilleur résultat sur le long terme ; mais s'il faut trouver une bonne solution rapidement alors l'heuristique est à privilégier puisqu'elle fournit souvent de meilleurs résultats très rapidement. Il est intéressant aussi de noter que plus le réseau est grand plus la différence entre les deux méthodes devient importante.

5 Conclusion & Perspectives

Nous avons comparé deux méthodes pour résoudre le problème d'optimisation d'énergie dans des réseaux sans fil fixes. Notre conclusion est que même si la faisabilité d'une solution est l'existence d'un routage valide pour la totalité des demandes, les arguments de coupes suffiront souvent pour faire un choix judicieux sur les liens à mettre en veille. La décision pourra donc être rapide même si elle ne fournira pas toujours la solution optimale. L'une des voies d'amélioration possible, est de trouver une manière plus efficace de déterminer les k coupes les moins denses d'un graphe.

Références

[1] S. Arora, E. Hazan, and S. Kale. $O(p \log n)$ approximation to sparsest cut in $\tilde{o}(n^2)$ time. In *Proceedings 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 238–247, 2004.

[2] S. Arora, S. Rao, and U. Vazirani. Geometry, flows, and graph-partitioning algorithms. *Commun. ACM*, 51 :96–105, October 2008.

[3] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing routing energy consumption : from theoretical to practical results. In *International Conference on Green Computing and Communications (GreenCom'10)*, page 8p, Hangzhou, China, 2010.

[4] R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. In *Proceedings of the 38th annual ACM symposium on Theory of computing (STOC'06)*, pages 385–390, 2006.

[5] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6) :787–832, 1999.

[6] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. *Networks*, 55(3) :276–286, 2010.

[7] D.B. Shmoys. *Approximation Algorithms for NP-hard Problems*, chapter 5 : [Approximation algorithms for] Cut problems and their application to divide-and-conquer, pages 192–235. (D.S. Hochbaum, ed.) PWS, 1997.