Implémentation d'une formulation étendue pour la conception d'un réseau de distribution d'électricité

Christian Frantzen

Université Libre de Bruxelles, Bruxelles, Belgique
christian.frantzen@ulb.ac.be

Abstract

a

Introduction

La transition énergétique pose beaucoup de nouveaux défis pour l'industrie. Souvent de grandes centrales de production d'électricité sont remplacées par plusieurs centrales plus petites. Ceci nécessite un changement important dans la configuration du réseau. La configuration d'un réseau de distribution d'électricité a un impact important sur des paramètres physiques comme perte d'énergie de de tensionainsi que la charge des points-clés d'un réseau comme les transformateurs. Pour prendre en compte tous ces facteurs mais en gardant un niveau d'abstraction élevé, (Rossi et al., 2012) introduisent un paramètre de distance maximale D_{max}. On considère qu'un client alloué à un fournisseur à une distance inférieure à D_{max} n'est pas soumis à des phénomènes de perte d'énergie ou de tension trop importants.

Le but des algorithmes présentés dans leur article est de maximiser la marge minimale des différentes sources du réseau. La marge d'une source est la différence entre sa capacité de production d'énergie et la demande totale des clients connectés à cette source.

Avoir une grande marge est une caractéristique de robustesse d'un réseau face à des augmentations spontanées de la demande.

Approche

Définition du problème

Pour résoudre le problème, on représente le réseau par un graphe non-dirigé $G(V, E)$, où V est l'ensemble des sommets et E est l'ensembles des arrêtes. L'ensemble des sommets V est l'union des ensembles disjoints des clients V_c et des fournisseurs V_f. L'ensemble E représente les différents connexions entre les sommets. $n_f = |V_f|$ est le nombre de fournisseurs dans le réseau et $n_c = |V_c|$ est le nombre de clients. Chaque sommet j dans V_f a un poids pow_j qui représente sa capacité maximale de production d'énergie. Chaque sommet i dans V_c a un poids dem_i qui représente a demande d'énergie.La distance minimale entre 2 sommets i et j dans G est donnée par $d_{i,j}$ est représente un nombre de sauts. Comme G est non-dirigé, $d_{i,j} = d_{j,i} \forall (i, j) \in V \times V$. L'ensemble des clients pour lesquelles la distance minimale entre eux et le fournisseur j est inférieure à D_{max} est noté $N_j \ \forall j \in V_f$. N_j est l'ensemble des clients qui peuvent potentiellement être connectés au fournisseur j. S'il existe un client qui ne se retrouve dans aucun ensemble N_j, la résolution est infaisable.

Contraintes

Une configuration est admissible si elle respecte les contraintes suivantes :

1. Contrainte de demande : La capacité d'un fournisseur doit être suffisante pour satisfaire la demande des clients auxquels il est connecté.
2. Contrainte de connectivité : pour des raisons techniques, deux fournisseurs ne doivent pas être connectés entre eux, ce que implique que chaque client doit être connecté à exactement un fournisseur. Chaque sous-graph de G doit être non-cyclique.
3. Contrainte de distance : Pour minimiser les phénomènes de perte d'énergie et de tension, les clients ne doivent pas être alloués à des fournisseurs à une distance supérieure à D_{max}.

Conditions de faisabilité

Pour que la résolution du problème soit faisable, il existe 2 conditions nécessaire pour garantir la faisabilité :

— La somme des capacités des fournisseurs doit satisfaire la demande totale des clients :

$$\sum_{j \in V_f} pow_j = \sum_{i \in V_c} dem_i$$

— Pour chaque client il existe un ou plusieurs fournisseurs à une distance inférieure ou égale à D_{max} :

$$\forall i \in V_c : \exists j \in V_f \mid dist(j, i) \leq D_{max}$$

Si ces deux conditions ne sont pas satisfaites, on peut démontrer que la résolution du problème est infaisable.
Pour rendre la résolution plus facile, dans notre configuration, on attribue à chaque fournisseur la capacité de satisfaire tous les clients :

$$\forall j \in V_f : pow_j = \sum_{i \in V_c} dem_i.$$

Méthode implémentée

Formulation orientée sommets

Dans l'article, il y a 2 types de formulations utilisées pour le problème. Une qui ne prend en compte que les sommets du graphe et ne décide que si un client i est alloué ou pas à un fournisseur j et l'autre formulation ne prend en compte que l'état des différentes arrêtes du graphe et décide s'il y a un flux de courant électrique qui la traverse ou pas.

Dans ce travail, on se concentre sur la formulation avec les sommets, ainsi, on a les variables de décision suivantes : $\forall (i,j) \in V_c \times V_f$, $x_{i,j}$ est mise à 1 si et seulement si le client i est alloué au fournisseur j, sinon $x_{i,j} = 0$. Les contraintes de la formulation sont décrites ici :

$$Maximiser\ M_{min} \tag{1}$$

$$pow - \sum_{i \in Nj} x_{i,j} dem_i \geq M_{min}\ \forall j \in V_f \tag{2}$$

$$\sum_{i \in V_c} x_{i,j} = 1\ \forall j \in V_f \tag{3}$$

$$x_{i,j} = 0\ \ \forall j \in V_f, \forall i \notin N_j \tag{4}$$

$$\text{Les clients alloués au fournisseurs j forment une composante connexe } \forall j \in V_f \tag{5}$$

$$\text{La distance d'un client i à son fournisseur est inférieure ou égale à } D_{max}\ \forall i \in V_c \tag{6}$$

$$x_{i,j} \in \{0,1\}\ \forall (i,j) \in V_c \times V_f \tag{7}$$

$$M_{min} \geq 0 \tag{8}$$

(1) est la fonction objectif : maximiser la marge minimale pour tous les fournisseurs. (2) définit la marge minimale pour tous les fournisseurs. La contrainte (3) assure que chaque client soit alloué à 1 fournisseur et (4) assure qu'un client ne soit pas alloué à un fournisseur situé à une distance supérieure à D_{max}. Les contraintes (5) et (6) sont difficile à imposer, elles seront expliquées en détail dans la section à suivre. La contrainte (7) assure l'intégralité de solution et (8) la non-négativité.

Le modèle de programmation linéaire principal se compose des contraintes (1)-(4), (7) ,(8) ainsi que des *layer inequalities* (inégalités des couches), qui seront expliqués plus tard. Les *layer inequalities* n'aident pas seulement à atteindre la solution optimale, mais aussi à converger plus rapidement.

Layer inequalities

Les layer inequalities restreignent le problème principal pour qu'il n'admet pas de solutions qui ne seront jamais bonnes. Les layer inequalities fonctionnen comme suit : les clients se trouvant à une distance λ au fournisseur j dans le graphe appartiennent à la couche $L_{j,\lambda}$. Ainsi $\forall j \in V_f$ et $\forall \lambda \in \{1, ..., D_{max}\}$, tout client $i \in L_{j,\lambda+1}$ peut être alloué au fournisseur j si et seulement si il y a au moins un client $k \in L_{j,\lambda}$ qui lui aussi est alloué au fournisseur j. Cette contrainte peut s'écrire :

$$x_{i,j} \leq \sum_{k \in L_{j,\lambda}} x_{k,j} \forall i \in L_{j,\lambda+1},$$
$$\forall \lambda \in \{1, ..., D_{max} - 1\}, \forall j \in V_f$$

Ces inégalités imposent des conditions nécessaires pour la connectivité des la solution ainsi que partiellement prenant en compte les exigences de distance. Elles peuvent être rendues plus stricte en ajoutant la condition que pour tout client i dans $L_{j,\lambda+1}$, seul des clients k dans $L_{j,\lambda}$ qui se trouvent à une distance plus petite ou égale à $D_{max} - \lambda$ de i avec tous les clients alloués sur le chemin entre i et k se trouvant à une distance strictement plus grande que λ de j peuvent assurer la connectivité de la solution.S'il n'y a pas de chemin entre k et i d'une longueur inférieur ou égale à $D_{max} - \lambda$, alors le client n'est pas aidant pour connecter i à j. De plus, si sur les chemins entre k et i se trouvent un client $v \neq i$ pour lequel la distance à j est plus petite ou égale à λ, alors le client i peut être connecté à j à travers v et on n'a pas besoin du client k.

Pour construire les layer inequalities, on a besoin de deux ensembles, $R_{k,j}$ et $P_{i,j}$. Soit k un client dans $L_{j,\lambda}$. L'ensemble des clients pour lesquels la distance jusqu'à j est strictement plus grande que λ et pour lesquels la distance jusqu'à k est plus petite ou égale à $D_{max} - \lambda$ est noté $R_{k,j}$. Cet ensemble peut être considéré comme l'ensemble des clients qui peuvent être connectés à j à travers k on satisfaisant la contrainte de distance. Soit i un client dans la couche $L_{j,\lambda+1}$. L'ensemble des prédécesseur potentiels dans $L_{j,\lambda}$ est noté $P_{i,j}$. Ainsi au moins un client dans $P_{i,j}$ doit être alloué au fournisseur j pour que i soit connecté à j.

Plus précisément :

$$R_{k,j} = \{v | \lambda < d_{v,j} \wedge d_{v,k} \leq D_{max} - \lambda\}$$
$$\forall k \in L_{j,\lambda}, \forall \lambda \in \{1, ..., D_{max} - 1\}, \forall j \in V_f$$
$$P_{i,j} = \{k \in L_{j,\lambda}\ i \in R_{k,j}\} \forall i \in L_{j,\lambda+1},$$
$$\forall \lambda \in \{1, ..., D_{max} - 1\}, \forall j \in V_f$$

Ainsi les inégalités des couches sont définies comme suit :

$$x_{i,j} \leq \sum_{k \in P_{i,j}} x_{k,j} \forall i \in L_{j,\lambda+1},$$
$$\forall \lambda \in \{1, ..., D_{max} - 1\}, \forall j \in V_f \tag{9}$$

Comme tout client n'apparaît qu'au plus une fois dans une couche $L_{j,\lambda}$ pour tous les fournisseurs $j \in V_f$,le nombre d'inégalités des couches est au plus $n_c \times n_f$.

Méthode des plans coupants

Coupes de connectivité Les inégalités des couches sont efficaces mais elles ne suffisent pas pour imposer la connectivité de la solution, c'est la raison pourquoi des coupes de connectivité sont ajoutées au problème principal.

Chaque fois que le problème principal est résolu, la composante connexe associée avec le fournisseur j, notée CC_j est construite pour tout j dans V_f. L'ensemble des clients qui sont alloués au fournisseur j mais qui ne sont pas dans CC_j est noté O_j.

Si O_j est non vide alors les clients alloués au fournisseur j ne forment pas une composante connexe, la solution n'est pas admissible et une coupe de connectivité est requise. Soit CS_j l'ensemble des clients adjacent à CC_j mais qui sont alloués à un fournisseur différent de j. CS_j peut être appelé ensemble de coupe (*cut set*) : enlever ses clients de G isolerait les clients dans CC_j du reste du graphe, particulièrement de O_j. Ainsi, pour que les clients dans O_j soient atteignables depuis j, au moins un client dans CS_j doit être alloué au fournisseur j. On peut l'écrire comme suit :

$$\sum_{i \in O_j} x_{i,j} \leq |O_j| \sum_{i \in CS_j} x_{i,j} \; \forall j \in V_f \tag{10}$$

L'inégalité (10) représente une coupe de connectivité. En théorie, une solution connectée peut être atteinte en intégrant les coupes de connectivité au modèle principal, or ce serait inefficace et requerrait un grand nombre d'itérations. Ainsi, à la place de générer une seule coupe de connectivité pour le fournisseur j, le nombre de coupes de connectivité à ajouter au problème principal est égal au nombre de composants connectés dans O_j. Ceci requiert la construction de C_j, l'ensemble de tous les composants connectés dans O_j. Soient CC un composant connecté dans C_j et CS l'ensemble des clients adjacents à CC qui ont un fournisseur différent de j. Pour qu'un client dans CC puissent être connecté au fournisseur j, au moins un client dans CS doit aussi être alloué à j. La version simplifié d'une coupe de connectivité peut être écrite comme suit :

$$x_{i,j} \leq |O_j| \sum_{k \in CS} x_{k,j} \; \forall i \in CC,$$
$$\forall CC \in C_j, \forall j \in V_f$$

Pour diminuer le nombre de coupes de connectivité, une seule coupe de connectivité est générée pour chaque composante connexe CC dans O_j, pour tout $j \in V_f$. D'abord, un client $i_{cc} \in CC$ se trouvant à une distance de j qui est minimale entre tous les clients $i \in CC$ est choisi aléatoirement. Ensuite tous les clients k dans CS tels que $d_{k,j} + d_{k,i_{cc}} > D_{max}$ sont enlevés de CS. Ils ne sont pas utiles pour connecter i_{cc} à CC_j dans l'itération suivante de l'algorithme des plans sécants. Plus précisément, la coupe de connectivité associée à CC est

$$x_{i,j} \leq \sum_{\substack{k \in CS \\ d_{k,j}+d_{k,i_{cc}} \leq D_{max}}} x_{k,j} \quad \forall CC \in C_j, \forall j \in V_f \tag{11}$$

Coupes de distance Pour tout fournisseur $j \in V_f$, les contraintes de distance ne sont vérifiées que lorsque j et ses clients forment une composante connexe. Il peut paraître étonnant que cette contrainte puisse être violée si on n'alloue que des clients de N_j à j, or une situation peut arriver où des clients se trouvant sur les plus court chemins entre les clients alloués à j et j sont alloués à un fournisseur différent et peuvent ainsi bloquer l'utilisation du plus court chemin, ce qui allonge les chemins restants et rend possible le dépassage de la longueur maximale D_{max}.

Comme les contraintes de distance ne sont vérifiées que lorsque les contraintes de connectivité sont satisfaites pour un fournisseur j, O_j est vide mais l'ensemble CS_j est construit quand même. Soit F_j l'ensemble des clients alloués à j à une distance de $D_{max} + 1$ de j dans la solution courante.

Une version simplifée des coupes de distance dit que pour diminuer la distance entre j et un client i dans F_j, au moins un client dans CS_j doit être alloué à j :

$$x_{i,j} \leq \sum_{k \in CS_j} x_{k,j} \; \forall i \in F_j, \forall j \in V_f$$

À nouveau, on diminue le nombre de coupes de distance :

$$x_{i,j} \leq \sum_{\substack{k \in CS \\ d_{k,j}+d_{k,i} \leq D_{max}}} x_{k,j} \; \forall i \in F_j, \forall j \in V_f \tag{12}$$

Déroulement de l'algorithme

La figure 2 (à la fin du document) présente le déroulement de l'algorithme de la méthode des plans sécants. Comme déjà mentionné, le modèle de programmation linéaire principal se compose des contraintes (1)-(4) et (7) -(9). Après chaque résolution du problème principal on vérifie si on doit ajouter des contraintes supplémentaires au problème, si non cela veut dire que la solution courante satisfait toutes les contraintes et elle est finale.

Résultats expérimentaux

Génération de graphes

La génération des graphes tout comme le fichier des données pour l'algorithme est fait en Python3. Les graphes générés pour tester l'algorithme sont rectangulaires. Un fournisseur se trouve dans chaque n-ième ligne du graphe, leur

position dans la ligne étant déterminée aléatoirement. Le reste du graphe est rempli avec des clients aléatoires. La demande d'un client est un nombre entier entre 1 et 100. Chaque client est connecté à ses voisins horizontaux et verticaux. La distance maximale D_{max} est égale à $dim_y + 1$.

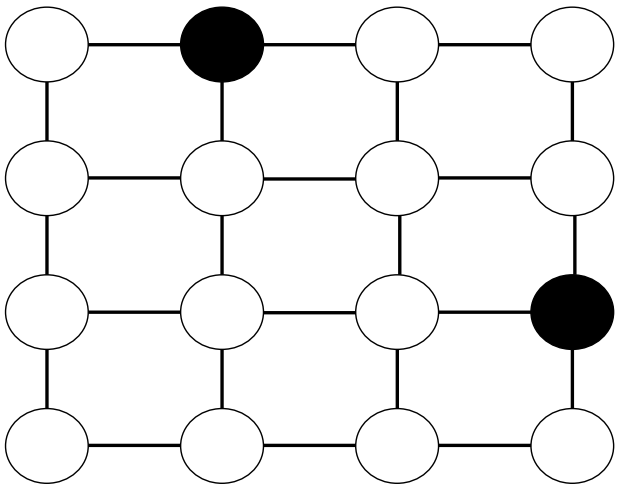

FIGURE 1 – Représentation d'un graphe 4×4. Les sommets en noirs sont les fournisseurs

Fichier des données Après la création du graphe, on génère les données nécessaires pour la résolution du problème, commençant avec la matrice comprenant tous les plus courts chemins du graphe, utilisant l'algorithme de Floyd [1]. Avec cette matrice, on peut établir les différentes couches $L_{j,\lambda}$, ainsi que les 2 ensembles R et P. Ensuite on crée les ensembles N_j. Pour finir on écrit tout dans un fichier de données dans le format AMPL (Fourer et al., 2003).

Résultats

Les expériences ont toutes été réalisées sur Ubuntu 14.04 LTS tournant dans VMware Workstation 10 tournant sur Windows 10, le processeur étant un Intel Core 4500U à 1.8 GHz avec 4GB de RAM pour la VM et 8GB pour l'ordinateur de base. Le solveur de programmation linéaire utilisé est CPLEX d'IBM. L'algorithme du problème principal et les méthodes des plans coupants ont été écrits en AMPL.

1. https://en.wikipedia.org/wiki/Floyd-Warshall_algorithm

dim	n	n_f	n_c	deg	temps	iter	inst
5x5	2	3	22	3.2	7.48	19.3	10
6x6	2	3	33	3.33	92.13	63.18	11
7x7	2	4	45	3.43	13245.44	110.8	5
5x5	3	2	23	3.2	1.48	16.3	10
6x6	3	2	34	3.33	10.34	79.27	11
6x6	4	2	34	3.33	5.90	53.82	11

La colonne *dim* indique les dimensions du graphe. La colonne *n* spécifie la valeur du pas de lignes lors de création du graphe pour l'insertion d'un fournisseur. Les colonnes n_f et n_c indiquent évidemment le nombre de fournisseurs et de clients dans le graphe. La colonne *deg* donne le degré moyen des sommets dans le graphe. La colonne *temps* donne la durée d'exécution moyenne du programme en secondes CPU, *iter* indique le nombre d'itérations moyen de l'algorithme. Le nombre d'instances de graphes considéré pour déterminé les valeurs est donné dans la colonne *inst*.

Discussion

Pour n=2, on voit très bien l'augmentation du temps de calcul en n'ajoutant que 11 clients aux graphes de 5×5 pour arriver aux graphes 6×6, le rapport est plus grand que 10. Le rapport pour n=3 reste proche de 10. Lorsqu'on diminue le nombre de fournisseurs dans le graphe, la solution est atteinte plus vite. Ceci est due à la diminution des combinations d'allocations : les différents set N_j ont moins d'intersections entre eux vue l'écart entre les fournisseurs. Si on avait mis une valeur moins élevée pour D_{max}, ce phénomène serait encore plus prononcé.

Pour les graphes 7×7, on a une augmentation énorme du temps de calcul en comparaison avec les graphes 6×6. Vue le temps de calcul minimum étant de 36.18 secondes CPU et le temps de calcul maximal étant de 62106.7 secondes CPU, ainsi que le nombre d'instances générées assez bas ne me permettent pas de donner une interprétation significative.

Conclusion et perspectives

Étant donné que l'algorithme traité dans ce travail est un algorithme exact, la solution trouvée est toujours optimale, donc la résolution du problème prend plus de temps qu'une heuristique. Vue que l'algorithme ne s'arrête que lorsque la méthode des plans coupants s'arrête, c'est à ce niveau là où on pourrait améliorer l'approche. Une version de la méthode des plans coupants combinée avec une heuristique ou une métaheuristique pourrait être une solution.

Comme déjà mentionné, avoir une grande marge est important pour le réseau pour pouvoir répondre aux augmentations spontanées de la demande. Les résultats expérimentaux montrent pourquoi : relancer l'algorithme pour trouver une nouvelle configuration admissible prend beaucoup trop de temps. Ici on n'a que des graphes très petits, en réalité les

dimensions des réseaux sont plus grandes. Les réseaux traditionnels n'ont que très peu de fournisseurs, or l'augmentation des modules solaire sur les toits d'immeubles n'est qu'un seul exemple où on introduit un nouveau fournisseur dans le réseau qui non seulement multiplie le nombre de configurations possibles, mais aussi déstabilise le réseau vu sa production non prédictible.

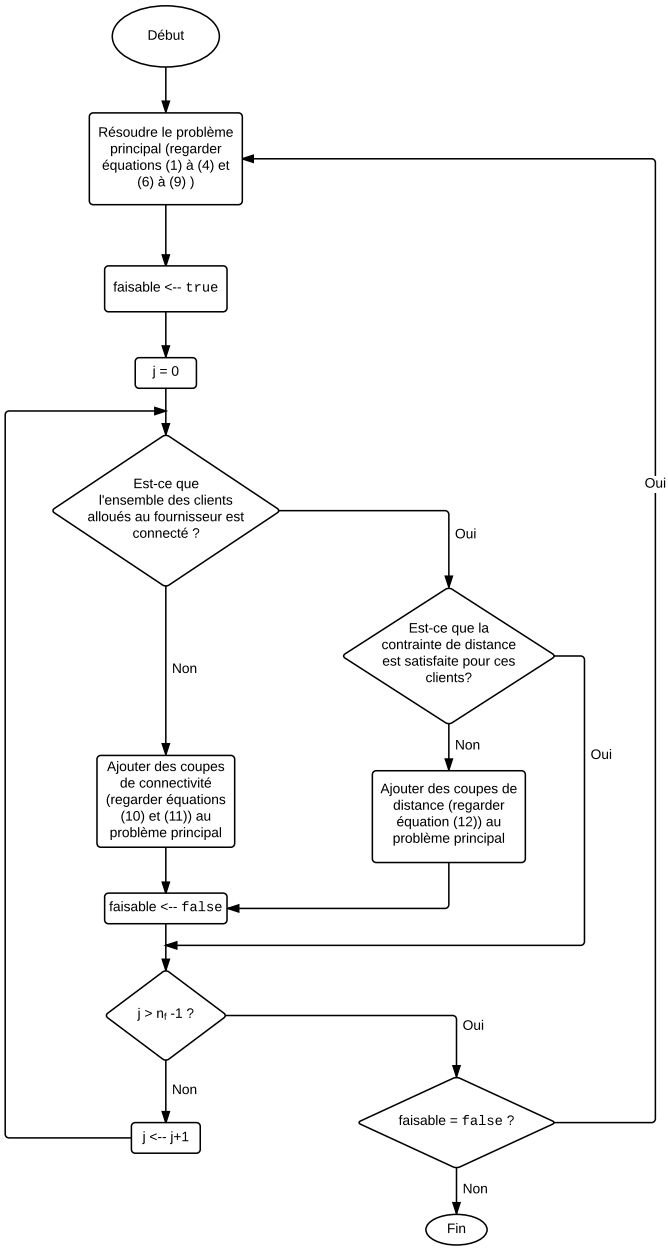

FIGURE 2 – Flowchart du *cutting plane* algorithme

References

Fourer, R., Gay, M. D., and Kernighan, W. B. (2003). *AMPL, A Modeling Language for Mathematical Programming*. Brooks/Cole - Thomson Learning.

Rossi, A., Aubry, A., and Jacomino, M. (2012). Connectivity-and-hop-constrained design of electricity distribution networks. *European Journal of Operational Research*, 218 :48–57.