PROJET DE RECHERCHE OPERATIONNELLE

Cléménce Barillot, Louise Résillot, Charbel-Raphaël Segerie

Janvier 2019

Contents

1 La douzième machine

1.1 Modélisation sous la forme d'un PLNE

Notations et variables

On dispose de 12 machines pour produire une quantité journalière fixée de 9 biens. On note d_b la demande quotidienne en bien b. Le but est d'utiliser le moins de machines possible pour satisfaire cette production. On note $\mathscr{B} = [\![1;9]\!]$ l'ensemble des biens et $\mathscr{M} = [\![1;12]\!]$ l'ensemble des machines.

On introduit la variable suivante :

$$x_m = \begin{cases} 1 & \text{si la machine m est utilisée pendant la journée} \\ 0 & \text{sinon} \end{cases}$$

Notre but est de minimiser $\sum_{m=1}^{12} x_m$.

On introduit également les variables entières suivantes :

$$y_{b,m} = \begin{cases} 1 & \text{si la machine m produit du bien b} \\ 0 & \text{sinon} \end{cases}$$

Ensuite, il nous fallait également une variable reliée à la notion de temps. Introduire $t_{b,m}$, temps passé par la machine m à produire du bien b ne convient pas car ce n'est pas une variable entière. En observant l'égalité $n_{b,m} = t_{b,m} * q_{b,m}$, on a donc préférer utiliser $n_{b,m} \in \mathbb{N}$ le nombre de biens b produits par la machine m lors de la journée. Ce nombre est entier car la production d'un bien ne peut pas être décomposée sur plusieurs machines.

Contraintes du PLNE

-La demande quotidienne de chaque bien doit être satisfaite :

$\sum_{m=1}^{12} n_{b,m} = d_b \ \forall$ b $\in \mathscr{B}$

- La durée d'utilisation de la machine est d'une journée maximum. On suppose que si une machine produit une certain nombre de biens b, elle les produit à la suite, car c'est plus efficace. Le temps passé à la production d'un bien b par la machine m est $\frac{n_{b,m}}{q_{b,m}}$. Changer de bien produit prend un temps δ à la machine, et le nombre de changements de biens produits vaut le nombre de biens produits moins 1. On obtient donc la contrainte suivante :

$\sum_{b=1}^{9} (\frac{n_{b,m}}{q_{b,m}} + \delta y_{b,m}) - \delta \leqslant 1 \ \forall$ m $\in \mathscr{M}$

-On ne peut produire un bien b avec la machine m que si la machine m est utilisée, ce qui se traduit par l'inégalité : $y_{b,m} \leqslant x_m \ \forall$ m $\in \mathscr{M}$, $\forall$ b $\in \mathscr{B}$

- Enfin, il nous faut borner le nombre de biens b que peut produire une machine m.

Tout d'abord, un nombre non nul de biens b est produit par une machine m si et seulement si elle produit de ce bien b : $n_{b,m} > 0 \iff y_{bm} = 1$.

L'implication $y_{bm} = 1 \Rightarrow n_{b,m} > 0$ (au moins un bien b est produit par la machine m si cette machine m produit le bien b) se traduit par l'inégalité $n_{bm} \geqslant y_{bm}$. On a obtenu une borne inférieure sur $n_{b,m}$.

Pour trouver la borne supérieure sur $n_{b,m}$, on utilise la contraposée de la condition $n_{b,m} > 0 \Rightarrow y_{bm} = 1$, ie $y_{bm} = 0 \Rightarrow n_{b,m} = 0$ et le fait que le nombre de biens b produits par une machine m ne peut pas excéder le nombre total de biens b que peut produire la machine m si elle ne produit que ce bien toute la journée. En combinant ces 2 conditions, on obtient l'inégalité suivante :

$n_{b,m} \leqslant y_{b,m} * q_{b,m}$

Programme linéaire en nombres entiers

On obtient donc le programme linéaire en nombres entiers suivant :

$$\text{Min} \sum_{m=1}^{12} x_m$$

$$\text{sc} \sum_{m=1}^{12} n_{b,m} = d_b \ \forall \ \text{b} \in \mathscr{B}$$

$$\sum_{b=1}^{9} (\frac{n_{b,m}}{q_{b,m}} + \delta y_{b,m}) - \delta \leqslant 1 \ \forall \ \text{m} \in \mathscr{M}$$

$\forall$ m $\in \mathscr{M}$, $\forall$ b $\in \mathscr{B}$:

$y_{b,m} \leqslant x_m$

$\text{n}_{bm} \geqslant y_{bm}$

$\text{n}_{b,m} \leqslant y_{b,m} * q_{b,m}$

$\text{n}_{b,m} \in \mathbb{N}$

$x_m, y_{b,m} \in \{0; 1\}$

1.2 Résolution à l'aide d'un solveur

Voici le tableau donnant $q_{b,12}$:

Bien b	1	2	3	4	5	6	7	8	9
Lettre	C	L	E	M	E	N	C	E	L
Index i	3	12	5	13	5	14	3	5	12
$q_{b,12}$	34	61	40	64	40	67	34	40	61

On utilise Julia et le solveur Cbc qui permet de résoudre des problèmes d'optimisation linéaire en nombres entiers (voir code en annexe). On obtient alors une solution optimale pour le PLNE posé précédemment, de 5 machines, utilisées comme le montre le tableau suivant qui donne les $n_{b,m}$:

	Machine											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	14	0	0	20	0	0	0	0	0	0
2	0	0	21	0	0	0	0	0	0	0	0	0
3	0	0	22	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	67	0	0	0	0	0
Bien 5	0	0	0	0	0	0	0	0	0	0	54	0
6	0	0	0	0	0	0	0	0	0	0	10	0
7	0	0	0	0	0	36	0	0	0	0	7	0
8	0	0	0	0	0	7	0	0	21	0	0	0
9	0	0	0	0	0	0	17	0	0	0	0	0

On vérifie ainsi que la demande en chaque bien est bien assurée. On note de plus que les machines sont utilisées quasiment toute la journée, ce qui est également une preuve de l'optimalité de la solution:

Temps d'utilisation de chaque machine

Machine	3	6	7	9	11
Temps d'utilisation	0.9991	0.9852	0.99947	0.4118	0.9963

2 Déploiement de fibres optiques

Le but de ce problème est de proposer une architecture de coût minimale, c'est-a-dire qui minimise la longueur totale de fibre utilisée pour desservir un ensemble d'antennes d'une ville en fibre optique.

Les antennes peuvent être reliées à un point de distribution de 2 façons:

- faire partie d'une boucle structurante: boucle qui comprend le point de distribution et compte au plus 30 antennes
- faire partie d'une chaîne de collecte : chaîne de maximum 5 antennes qui part d'une antenne appartenant à une boucle structurante

2.1 Définition des hypothèses principales

H1 : Il y autant de boucles structurantes que de distributeurs. Autrement dit, chaque boucle structurante contient un seul distributeur.

L'énoncé sous-entend cette hypothèse, et en effet, c'est logique d'exploiter tous les points de distribution. En mettre 2 dans la même boucle structurante est inutile.

H2: Chaque antenne a appartient au réseau associé au distributeur d le plus proche de cette antenne.

En effet, on peut supposer que toutes les antennes sur le réseau sont "proches" du distributeur, puisque le nombre de liaisons maximal entre une antenne du réseau prise au hasard et le distributeur associé à ce réseau est de 20 ((30/2)+5). Dit grossièrement, le "diamètre du réseau est borné", ce qui justifie de travailler localement.

La notion de proximité peut s'implémenter de deux manières différentes: soit on prend la norme 2 à l'aides des coordonnées géographiques fournies, soit on utilise les distances associées aux fourreaux. Nous avons retenu cette dernière approche, puisque les antennes du réseau sont reliées par les fourreaux. La première étape de notre code est de "clusteriser" le graphe, ie on affecte chaque antenne au point de distribution le plus proche. Les clusters sont donc une partition de l'ensemble des sommets.

H3: Dorénavant, ces clusters sont fixés : on sait à l'avance pour chaque antenne dans quelle réseau elle sera incorporée.

C'est une hypothèse assez forte, mais qui est en fait très puissante, car elle permet maintenant d'optimiser cluster par cluster. C'est le fameux: **diviser pour régner**. A présent, à la place de créer une architecture complète, et de comparer le coût total architecture par architecture, on peut travailler par cluster indépendamment, trouver une bonne approximation de la solution optimale pour chacun des clusters, pour finalement réunir toutes ces bonnes approximations en une seule architecture.

2.2 Idée générale de l'algorithme

On présente ici l'algorithme utilisé en utilisant les chiffres de Paris. Les parties suivantes serviront à detailler les différentes étapes de cet algorithme.

Création des clusters d'ordre 0:
Rattacher chaque antenne au distributeur le plus proche

Pour passer des clusters d'ordre 0 aux clusters d'ordre 1:
Pour chaque cluster (de taille notée N), répéter 500 fois la boucle primaire :

> créer 1 000 réseaux aléatoires

> sélectionner la meilleure boucle structurante créée parmi les 1 000.

> *répéter 5 000 fois (ou 20 000 si la boucle est prometteuse au bout de 5000 itérations):*
* modifier la boucle structurante par insertion ou substitution d'une antenne
O(N)
* compléter aléatoirement la boucle structurante. $O(N^3)$ qui se comporte comme un $O(N^2)$
* si le score est meilleur que pour la boucle structurante précédente, accepter cette boucle structurante

> récupérer le réseau en sortie de boucle, et le sauvegarder s'il est meilleur que ceux calculés précédemment par la boucle primaire

La boucle primaire s'execute en deux secondes pour des clusters de taille 50. On l'exécute sur les 11 clusters, ce qui prend environ 10*500*2 =10 000 s, soit **3h de calcul**.

Pour créer le réseau final:
Pour chaque cluster:

> On prend la meilleure boucle structurante issue de la boucle primaire

> On considère la ville de manière globale et en supprimant toutes les chaînes de collectes

> On recrée les chaînes de collecte en minimisant la distance *antenne seule-antenne de la boucle de collecte*, cette fois-ci sur tout le graphe et non pas seulement cluster par cluster. Une antenne qui était alors sur une chaîne de collecte liée à la boucle structurante du premier cluster peut désormais se retrouver sur une chaîne de collecte du second cluster.

On considère maintenant toutes les boucle structurantes:

> On supprime toutes les chaînes de collecte créé précédemment.

> Parmi toutes les antennes de la ville qui ne sont pas sur une boucle structurante, on sélectionne celle qui est la plus proche d'une des antennes sur une boucle structurante, sans tenir compte des clusters initiaux, et on la place. Et ainsi de suite.

On est ainsi passé des clusters d'ordre 0 à des cluster d'ordre 1.
On procède de même pour accéder aux clusters d'ordre 2, qui nous donneront notre solution finale.

Au total, l'algorithme s'exécute sur l'instance Paris en 6h, car il faut 3h pour passer des clusters d'ordre 0 à 1, et 3h de plus pour passer de l'ordre 2 à l'ordre 3.

2.3 Stratégie de résolution détaillée

Cette partie reprend en détail la partie précédente en expliquant plus en profondeur chaque étape. On suppose les 3 hypothèses H1 H2 et H3 définies précédemment.

Pourquoi clusteriser alors que l'on perd du même coup l'optimalité? Explication: pour un cluster donné, si on crée 1000 réseaux de manière aléatoire. Si on sélectionne le meilleur réseau. Puis si on réunit tous les meilleurs réseaux de chaque cluster, on créer alors une architecture excellente.

Si on avait créé 1000 architectures (ce qui prend autant de temps que la stratégie qui précède), et qu'on s'était contenté de sélectionner la meilleure architecture, la probabilité pour que l'on tombe sur une solution meilleure que celle de notre stratégie est très faible.

2.3.1 Génération d'un réseau pour un cluster donné

Il faut maintenant expliquer plus en détail le processus de génération d'un premier réseau pour un cluster donné.

Ce processus se décompose en deux étapes:

1) Création de la boucle structurante

2) Création des chaînes de collectes.

1.Création de la boucle structurante

Pour créer la boucle structurante, on peut utiliser une stratégie aléatoire:on choisit un nombre aléatoire size_boucle entre 0 et min(30,nombre d'autres antennes que le point de distribution dans le cluster) (size_boucle=0 signifie que la boucle ne contient que le distributeur), puis on choisit itérativement size_boucle antennes successives de manière aléatoire. On ne choisit pas les autres antennes au plus proche du dernier point placé parce que ça ne nous donnerait qu'un seul réseau. Or notre idée est de créer un grand nombre de réseaux différents puis de les modifier pour aboutir à une bonne solution.

2.Création des chaînes de collecte

La boucle structurante étant fixée, on crée de manière itérative et déterministe les chaînes de collectes. C'est le caracère déterministe de cette étape qui nous permettra dans la suite de définir une topologie et des voisinages efficaces. Pour chaque antenne a du cluster qui n'est pas sur la boucle, on trouve l'antenne b (sur la boucle ou en bout de chaîne de collecte) qui réalise la distance minimale avec a. On retient cette distance, puis on procède de même avec les autres antennes.

Mathématiquement, on cherche (a*,b*)=arg min(d(a,b)) avec a l'indice de l'antenne qui n'est pas encore reliée au réseau, et b une antenne déjà placée sur la boucle ou en dernière position sur une chaîne. Puis on lie a* et b*.

Cette manière de faire construit des chaînes de collecte de manière presque optimale pour une boucle fixée. Certes, ce n'est pas tout à fait optimal, mais c'est très rapide à calculer avec notre méthode.

2.3.2 Détail de l'étape de création des chaînes de collecte sur un exemple et calcul de complexité

Il est primordial d'avoir un algorithme qui tourne vite pour la création des chaînes de collecte car on l'appelle un grand nombre de fois dans notre code. Nous avons donc réfléchi à une façon de réduire le temps de calcul de cette partie du code. Nous détaillerons le calcul de complexité pour cette étape, pour lequel on prend en compte le nombre de calculs de distances effectué.

1.Méthode

Prenons un exemple simple pour illustrer notre méthode.

On se place dans un cluster constitué des indices {0,1,2,3,4,5,6}. La boucle structurante est déjà fixée: boucle={0,1}.

La première étape consiste à générer le tableau contenant pour chaque antenne à raccorder au réseau l'indice de l'antenne déjà placée la plus proche ainsi que cette distance. On obtient par exemple le tableau suivant :

Etape i=0

Indice à placer	2	3	4	5	6
$dmin_0(a)$ = min(d(a,0),d(a,1))	1	2	2	4	3
b*(a,i)	0	1	0	1	1

$dmin_i(a)$ est la distance minimale entre a et une antenne déjà placée (sur une boucle ou sur l'extremité d'une chaîne de collecte).

b*(a,i) est l'indice de l'antenne b déjà placée (sur la boucle ou en extremité d'une chaîne de collecte) la plus proche de a à l'étape i.

On note aussi a_i l'antenne raccordée au réseau à l'étape i.

On raccorde au réseau l'antenne qui minimise la distance avec le réseau en place. Ici, on lie donc l'antenne 2 et l'antenne 0. Le réseau devient r={0-2, 1}.

Pour remplir le nouveau tableau aux étapes ultérieures, il n'est pas nécessaire de calculer pour chaque antenne à placer les distances à toutes les antennes placées pour savoir quel point a est le plus proche d'un point déjà placé.

On se place à l'étape i (il y en aura a_0 (a_0=le nombre d'antenne à placer initialement dans le cluster) en tout). Pour les antennes a pas encore placées qui vérifient b*(a,i-1) $\neq b*$(a$_{i-1}$,i-1), il suffit de calculer la distance entre a antenne et a_{i-1} et de comparer à la distance entre a et b*(a,i-1). On obtient ainsi b*(a,i).

Mathématiquement, si b*(a,i-1) $\neq b*$ (a$_{i-1}$,i-1) Alors b*(a,i)=min(b*(a,i-1),d(a_{i-1},a)) ou bien b*(a,i)=b*(a,i-1) si a_{i-1} est le 5ieme élement d'une chaine de collecte.

Dans notre exemple, il s'agit des antennes 3,5 et 6. Pour l'antenne 3, on calcule d(1,3) et d(2,3) et on remplace dans le tableau l'indice b*(3) du point du réseau le plus proche du point 3 (1 ou 2) ainsi que la valeur de cette distance. On fait de même pour les antennes 5 et 6.

Par contre si b*(a,i-1) = b*(a_{i-1},i-1) alors il faut recalculer le minimum entre a et toutes les antennes déjà placées.

Car pour les antennes dont l'antenne la plus proche à l'étape i-1 a été reliée à l'antenne placée à l'étape i-1, dans le pire des cas, ie si cette antenne b*(a,i-1) est dans une chaîne de collecte, on ne peut plus y relier d'antenne. On doit donc recalculer les distances entre l'antenne a et toutes les antennes du réseau disponibles (ie sur la boucle structurante, ou en bout de chaîne de collecte sans être la 5ème). Cela occasionne un nombre de calculs de l'ordre de b_i, avec b_i le nombre d'antennes déjà placées à l'étape i. En vérité, ce nombre est b_i retranché du nombre d'antennes qui sont dans une chaîne de collecte pas en dernière position, ou en 5ème position, mais l'ordre de grandeur b_i est suffisant pour le calcul de complexité. Ici, ce cas ne se passe pas car 4 est le plus proche de 0, qui a certes été liée à 2 mais est une antenne de la boucle donc peut être le point de départ de plusieurs chaînes. Donc en l'occurrence, il suffit de calculer d(2,4) et comparer avec d(0,4). Le cas où il faut recalculer des distances se présentera aux étapes ultérieures, qu'on ne présente pas ici. Ensuite, on place dans le réseau l'antenne qui est à a distance la plus faible d'un point des réseaux, et on réitère cette étape de

calcul du nouveau tableau de distances.

2.Calcul de la complexité pour construire toutes les chaînes de collecte d'un réseau

La complexité pour générer le tableau de l'étape 0 est en $a_0 * b_0$ avec a_0 le nombre d'antennes pas encore placées à l'étape 0 et b_0 le nombre d'antennes déjà placées à l'étape 0.
On note :

$\boldsymbol{a_i}$: le nombre d'antennes qu'il reste à placer dans le réseau à l'étape i. $\boldsymbol{a_i = a_0 - i}$

$\boldsymbol{b_i}$: le nombre d'antennes déjà placées dans le réseau à l'étape i. $\boldsymbol{b_i = b_0}$ **- i**

$\boldsymbol{k_i}$: le nombre d'antennes dont l'antenne la plus proche à l'étape i-1 a été reliée à l'antenne insérée dans le réseau à l'étape i-1. On suppose que les antennes sont uniformément réparties sur le graphe, ainsi à chaque étape, il y a $\frac{a_i}{b_i}$ arêtes qui veulent se relier à une des b_i antennes placées. Ainsi,le nombre d'antennes qui se sont fait "devancer" par l'antenne précédente pour rallier une antenne du réseau à l'étape i est de $\boldsymbol{k_i \approx \frac{a_i}{b_i}}$.

Pour les $a_i - k_i$ antennes qui peuvent théoriquement encore être réliées à b*(a,i-1), il faut un calcul de distance avec la dernière arête placée. Et pour les k_i arêtes qui se sont fait "devancer", il faut dans le pire des cas de l'ordre de b_i calculs de distances.

D'où la complexité globale suivante :

$$\mathrm{C} = \sum_{i=0}^{a-1} [(a_i - k_i) + k_i b_i]$$

$$\mathrm{C} = \sum_{i=0}^{a-1} [a_i + k_i(b_i - 2)]$$

En utilisant le fait que $k_i \approx \frac{a_i}{b_i}$ et $a_i = a - i$,on obtient finalement :

$\mathbf{C = O(a^2)}$

Cette complexité est meilleure que pour la méthode naïve qui est en $O(a^3)$.

2.3.3 Amélioration du réseau aléatoire

Maintenant que l'on sait générer une "graine aléatoire de réseau", on peut l'améliorer en explorant "son voisinage". Deux solutions se présentent à nous:

- **le recuit simulé**, qui ne marche pas très bien ici car il est difficile de trouver les bonnes température pour chaque réseau...
- une **variante du recuit simulé** qui consiste à changer la boucle structurante seulement si le réseau qui en découle est aussi bon.

Pourquoi utiliser la variante du recuit simulé plutôt que le recuit simulé lui-même?

Le recuit ne prend pas le temps d'explorer en profondeur le voisinage, il gambade trop loin dans l'espace des boucles structurantes possible, et ne va pas au bout de la recherche du minimum local.

On choisit donc **de passer d'une boucle structurante à une autre seulement si le réseau qui en découle est meilleur**, afin de vraiment aller au bout de la recherche du minimum local. Comme on part de différentes graines aléatoires de réseau, on a bon espoir de trouver une solution optimale avec cette méthode.

En pratique, pour un cluster donné:

- on génère **1000 graines aléatoires**,
- on **sélectionne la meilleure** parmi les 1000 (pour ne pas partir d'une graine trop mauvaise)

- puis **on itère 10 000 fois la variante du recuit simulé** présentée précédemment. (10 000 est un compromis d'efficacité). On peut même améliorer encore l'algorithme en itérant 30 000 au lieu de 10 000 si la solution est "prometteuse" au bout des 10 000 premières itérations.

Comment calculer le voisinage?

Pour utiliser cette méthode, il faut avant tout définir ce qu'est un voisinage pour ce problème.

Si on considère l'ensemble des antennes à placer dans le cluster $A = \{a_1, \ldots, a_{|A|}\}$, et une boucle $B = (b_0, b_1, \ldots, b_{|B|})$ (par convention b0 est le distributeur, il est fixé à tout jamais).

On peut alors dans le cas général faire trois opérations:
1) intervertir un a_k avec un b_k si $|A| > 0$;
2) insérer un a_k dans la boucle si elle n'est pas déjà trop grande;
3) supprimer un b_k si la boucle n'est pas trop petite.

Un voisinage consiste donc à effectuer aléatoirement par équiprobabilité une de ces 3 opérations.

2.3.4 Résultats obtenus par cette méthode

Le recuit ainsi modifié permet d'atteindre un score de 35 000 (Score de Paris Nice et Grenoble cumulés) facilement (5 min de calcul).

2.3.5 Dernière amélioration

La dernière amélioration consiste à supprimer l'hypothese 2:

Maintenant que l'on a construit d'excellentes boucles structurantes pour chaque cluster, on supprime toutes les chaînes de collecte, et on les reconstruit avec cette fois-ci $A = \{$*antennes déjà placées sur l'ensemble de l'architecture* $\}$ et $B = \{$*antennes qui reste à placer dans toute la ville* $\}$.

On gomme ainsi l'approximation de H2, et un nouveau set de clusters, dit d'ordre 1 (les clusters construits "bêtement" avec H2 sont appelés d'ordre 0). On peut recommencer l'ensemble du processus avec les cluster d'ordre 1, ce qui permet d'obtenir un cluster d'ordre 2. Ces clusters offrant un très bon score, on s'arrête là.

2.4 Une borne inférieure possible

Voici une idée de PLNE pour obtenir une borne inférieure pour notre problème.

On note :

$d_{i,j}$ la distance entre les arêtes i et j, pour l'arête orientée de i vers j.
V l'ensemble des antennes et points de distribution du graphe du réseau

$$x_{i,j} = \begin{cases} 1 & \text{si il y a une arête entre les antennes i et j (orientée de i à j)} \\ 0 & \text{sinon} \end{cases}$$

On exploite l'idée que tout sommet du graphe qui n'est pas un distributeur a une et une seule arête entrante . On précise que l'arête entrant en un sommet ne peut pas provenir de ce même sommet. On relâche la contrainte sur le nombre d'arêtes sortantes (ce qui est faux en pratique pour les antennes des chaînes qui ont au plus une arête sortante).

Voici le PLNE proposé.

Min $\sum_{i,jinA} d_{i,j} x_{i,j}$

sc $\sum_{i} x_{i,j} = 1 \ \forall$ j $\in A$

$x_{i,i} = 0 \ \forall$ j $\in A$ (on ne relie pas un point à lui-même)

$x_{i,j} \in \{0,1\}$

En l'implémentant, on trouve les bornes inf suivantes:

Pour Grenoble: 1989 (effectivement, l'hypothèse qui (fait pourtant sens) des noeuds qui correspondent à des distributeurs qui doivent recevoir une arrête est violée). En réimplémentant le PNLE en libérant les contraintes sur les distributeurs on obtient: 1537

Pour Nice: 5502 (en ne relâchant pas les contraintes sur les distributeurs, ce n'est donc peut-être pas une borne inf, mais il parait tout de même stupide de n'avoir que des boucles structurantes de tailles 0...)

Remarque: Pour résoudre ce PNLE, il suffit d'implémenter le relaché linéaire, car la matrice est totalement unimodulaire.

Remarque: le PNLE est implémenté par scipy.optimize. Mais même avec la méthode du point intérieur, qui est optimisée pour des matrices sparces d'après la documentation, le problème ne tourne pas sur Paris.

3 Annexe

```julia
using JuMP
using Cbc
lpModel = Model(solver = CbcSolver(seconds = 3600))

@variable(lpModel, x[1:n_m] <=1, Int)
@variable(lpModel, y[1:n_b, 1:n_m] <=1, Int)
@variable(lpModel, n[1:n_b, 1:n_m] >=0, Int)

@objective(lpModel, Min, cost(x))

for b=1:n_b
    @constraint(lpModel, sum(n[b,m] for m=1:n_m) == d[b])
end

for m=1:n_m
    @constraint(lpModel,sum(n[b,m]/q[b,m]+δ*y[b,m] for b=1:n_b)-δ <=1)
    for b=1:n_b
        @constraint(lpModel, y[b,m]<=x[m])
        @constraint(lpModel, n[b,m]>=y[b,m])
        @constraint(lpModel, n[b,m]<= y[b,m]*q[b,m])
        @constraint(lpModel,y[b,m]>=0)
    end
    @constraint(lpModel, x[m] >=0)
end

solve(lpModel)

println("cost = ",getobjectivevalue(lpModel))

println("x = ",getvalue(x))

xopt = getvalue(x)
yopt = getvalue(y)
nopt = getvalue(n)

println("nopt = ", nopt)
```

Figure 1: Code Julia pour résoudre le PLNE