Rapport de stage
Problème de localisation et de dimensionnement de site de concentration en contexte incertain

Onur Celebi

6 septembre 2007

GSCOP - France Télécom

1 Introduction

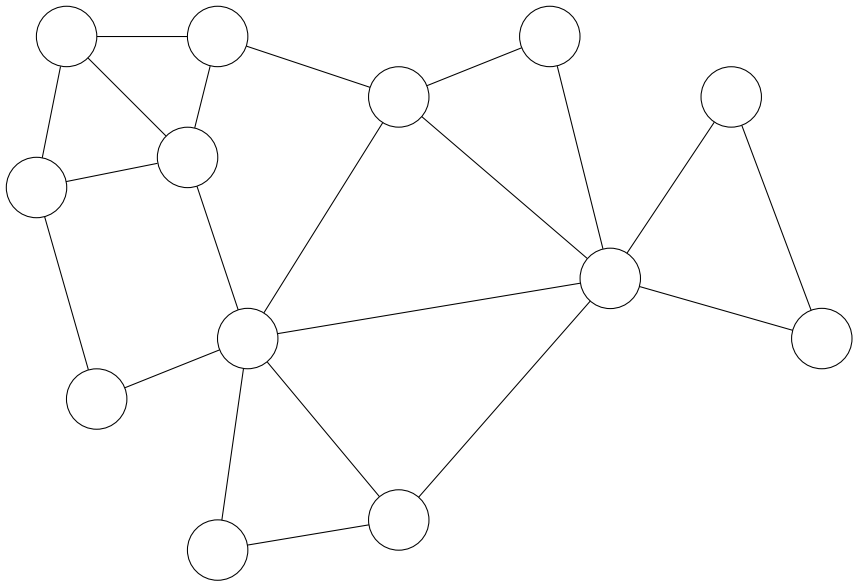
Ce stage traite le problème de décision associé au choix de localisation de concentrateurs dans un réseau de télécommunications. Il s'agit de mettre en place un modèle relativement détaillé qui, pour un graphe donné, est capable de déterminer le sous-ensemble de noeuds de ce graphe où des éléments de concentration (routeur, hub, etc..) peuvent être placés. Le modèle doit également pouvoir dimensionner les sites concentrateurs ainsi formés.

Etant donné un graphe représentant un réseau, le problème que nous traitons est d'affecter à certains noeuds des éléments de concentration - que nous appellerons "site concentrateur" dans la suite de ce document - de manière à ce que tout élément du réseau

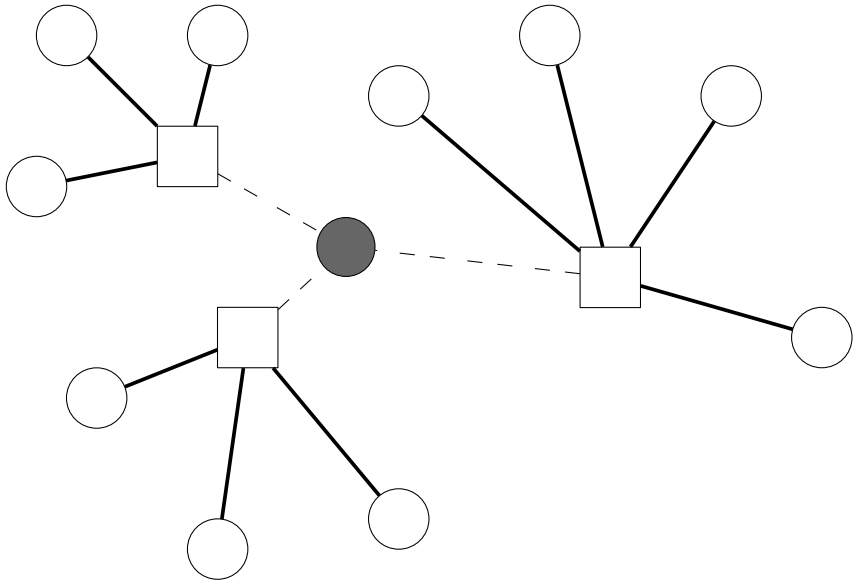
- soit connecté à un seul site concentrateur
- ou devienne un site concentrateur.

Les autres noeuds sont appelés "site client". L'ensemble de site clients affectés à un site concentrateur forme un graphe en étoile, qu'on appelle réseau d'accès ("access" en anglais). Puis, on doit relier les sites entre eux. Un noeud qui devient un site concentrateur est relié à des équipements qui sont déjà en place et dont on ne se préoccupe pas dans le modèle étudiée. Le réseau qui relie les sites concentrateurs entre eux s'apelle réseau coeur ("backbone" en anglais). On ne connait pas la forme exacte de ce réseau.

Bienque les sites concentrateurs ne sont pas physiquement reliés entre eux deux à deux, on represente un graphe complet avec des arêtes en pointillé pour pouvoir indiquer le prix par unité de débit de trafic qui circulerait entre chaque paire de site concentrateur.



devient



Les noeuds carrés sont ceux qui ont reçu des éléments de concentration. Les arêtes dans le graphe de départ représentées de manière plus épaisse ici correspondent aux liaisons client-site. On remarque qu'on a ainsi formé des graphes étoiles. Ces graphes sont appelés réseau accès. Il existe une infrastructure qui permet de relier les sites concentrateurs entre eux. Le réseau qui relie les sites concentrateurs entre eux est appelé réseau coeur. En réalité, on ne connaît pas la topologie de ce réseau, mais on considère que chaque site concentrateur est relié à une sorte de super concentrateur (dessiné en gris dans la figure). On suppose donc également une topologie de graphe étoile.

On remarque que le problème est fortement combinatoire. Ce qui nous intéresse d'abord, c'est le dimensionnement des sites concentrateurs. Le problème qui consiste à choisir les sites concen-

trateurs est appelé P-median dans la littérature. C'est un problème NP-complet. Ici, on aimerait également tenir compte du trafic qu'il va y avoir entre chaque paire de sites concentrateurs dans la solution.

2 Modèle

2.1 Explication du modèle

On suppose qu'on connaît le trafic entre chaque paires de sites clients. Le coût de connexion d'un client à un site est fixe, mais d'après les études technico-économiques, il est plus pertinent de considérer le coût par unité de débit pour une connexion entre deux sites (en pointillé sur la figure). Le trafic va donc intervenir dans le coût du réseau coeur et dans le dimensionnement des sites car on doit satisfaire la demande en trafic au niveau de chaque site.

Sur les sites, on peut placer deux types de cartes : des cartes de type "access" et des cartes de type "backbone". Elles contiennent respectivement des ports de type "access" et "backbone". Chaque type de port a une certaine capacité. On doit être en mesure de placer assez de ports de type "access" pour satisfaire le trafic dans le reseau accès du site en question, et de même, on doit placer assez de ports "backbone" pour satisfaire le trafic du côté du réseau coeur.

Chaque site est relié à un site concentrateur ou devient lui même un site concentrateur. On va donc avoir un certain nombre de variables booléennes associées à cette décision.

On note :

Les paramètres du problème et définitions :

- I l'ensemble des noeuds
- D l'ensemble de demandes (de trafic). Chaque élément $k \in D$ correspond à un couple $(o(k), d(k)) \in I^2$, avec $o(k)$ l'origine de la demande, et $d(k)$ la destination.
- K_a capacité d'un port "access"
- K_b capacité d'un port "backbone"
- N_a nombre de ports sur une carte "access"
- N_b nombre de ports sur une carte "backbone"

Les données en entrée du problème :

- t_{im} débit du trafic entre le client i et m
- A_{ij} coût de connexion d'un client i à un site j
- B_{jl} coût par unité de débit de trafic qui circule entre deux sites j et l.
- c_a coût d'une carte de type "access"
- c_b coût d'une carte de type "backbone"

Les variables de décision :

- x_{ij} variable booléenne representant la connexion du client i au site j.
- u_j^a nombre de cartes de type "access" sur le site j
- u_j^b nombre de cartes de type "backbone" sur le site j

Variables intermédiaires :

- z_{jl} débit circulant du site j au site l
- d_j^a nombre de ports de type "access" sur le site j
- d_j^b nombre de ports de type "backbone" sur le site j

Nous allons donc chercher à minimiser l'expression :

$$\sum_j (c_a.u_j^a + c_b.u_j^b) + \sum_j \sum_l B_{jl} z_{jl} + \sum_i \sum_j A_{ij} x_{ij}$$

Chaque site doit être relié à un site concentrateur ou doit être lui-même un site concentrateur. En prenant comme convention que le noeud j devient un site est exprimé par $x_{jj} = 1$, cette contrainte s'exprime de la manière suivante :
$\sum_j x_{ij} = 1 \qquad \forall i$

Un client i ne peut être relié à un site j que s'il est effectivement un site :
$x_{ij} \leq x_{jj} \qquad \forall i \forall j$

Les cartes "access" ont N_a ports et les cartes "backbone" ont N_b ports.
$N_a n_j^a \geq d_j^a \qquad \forall j$

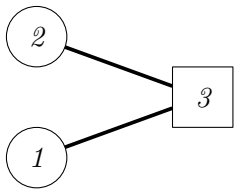
$N_b n_j^b \geq d_j^b \qquad \forall j$

Pour un site donné, le nombre de ports côté réseau accès doit être assez grand pour satisfaire le trafic qui circule dans le réseau accès de ce site. Pour chaque site j, on regarde chaque client i qui lui est connecté, et on fait la somme du trafic entre ce client et tous les autres clients du réseau.
$K_a d_j^a \geq \sum_{i \in I} \sum_{m \in I} t_{im}.x_{ij} \qquad \forall j$

Cependant, on ne peut pas agréger deux trafics venant de deux clients différents. Par exemple, s'il reste 300 Mb/sec de capacité non utilisé sur un port donné, on ne peut pas l'utiliser pour un autre client. C'est pourquoi, il est plus pertinent de modéliser comme suit :

$$d_j^a \geq \sum_{i \in I} \left\lceil \frac{\sum_{m \in I} t_{im}}{K_a} \right\rceil x_{ij} \qquad \forall j$$

Exemple 1 *Prennons l'exemple de deux sites clients (1) et (2) qui sont raccordés à un site concentrateur (3). On suppose que le trafic sortant de (1) vers (3) est de 500Mb/s, et que le trafic sortant de (2) vers (3) est de 500Mb/s également. On voit que même si on place un port de capacité 1000Mb/s sur le site, on ne pourra pas raccorder physiquement les deux clients sur ce même port. Il faudra donc au minimum 2 ports.*



On met un place un dimensionnement symétrique. C'est-à-dire qu'un port mis en place doit satisfaire le maximum parmi le débit entrant et le débit sortant, on a donc également :

$$d_j^a \geq \sum_{i \in I} \left\lceil \frac{\sum_{m \in I} t_{mi}}{K_a} \right\rceil x_{ij} \qquad \forall j$$

On peut exprimer ceci à l'aide de données calculées en prétraitement :
$t_i^{in} = \left\lceil \frac{\sum_{m \in I} t_{mi}}{K_a} \right\rceil$

$t_i^{out} = \left\lceil \frac{\sum_{m \in I} t_{im}}{K_a} \right\rceil$

Remarque 1 *Dans le cas des t_{ii}, on considère en réalité des sites colocalisés. Dans le modèle, tout se passe comme si on avait placé des éléments de concentration juste à côté d'un site client. On pose $t_{ii} = 0$, ceci résulte un coût de raccordement nul, puisque les éléments sont physiquement côte à côte.*

Ce qui donne l'expression générale des contraintes côté accès :

$$d_j^a \geq \sum_{i \in I} t_i^{in} x_{ij} \qquad \forall j$$

$$d_j^a \geq \sum_{i \in I} t_i^{out} x_{ij} \qquad \forall j$$

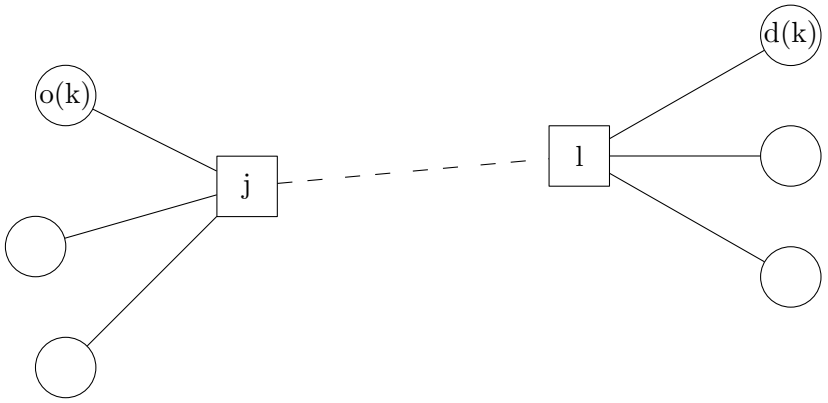
On ne connaît pas à l'avance le débit qui va circuler entre deux sites dans le réseau "backbone", puisqu'on ne sait pas quels noeuds vont devenir des sites. Le calcul du débit z_{jl} entre deux sites j et l est délicat. Cependant, on peut facilement exprimer le nombre de ports nécessaires de type "backbone" en fonction des z_{jl}. Pour un site j, on somme le débit circulant entre j et les autres sites l. De même, un port doit supporter le maximum parmi le débit entrant et sortant.

$$K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{jl} \qquad \forall j$$

$$K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{lj} \qquad \forall j$$

Pour calculer le débit entre deux sites j et l, on fait la somme du trafic y circulant si et seulement si les noeuds j et l ont été designés comme des sites de concentration dans la solution actuelle.

$$z_{jl} \geq \sum_{k \in D} t_{o(k)d(k)} x_{o(k)j} x_{d(k)l} \qquad \forall j \forall l$$



Cette expression étant quadratique, on veut linéariser cette contrainte. On va suivre une méthode utilisé dans [YAMGOU].

Lorsqu'on a un produit de deux variables booléennes $x_{o(k)j}.x_{d(k)l}$, on peut remplacer cette expression par une variable y_{kjl} qui obéirait aux contraintes $y_{kjl} \geq x_{o(k)j}+x_{d(k)l}-1$, $y_{kjl} \leq x_{o(k)j}$ et $y_{kjl} \leq x_{d(k)l}$ et $y_{kjl} \geq 0$. Ainsi on a bien $y_{kjl} = x_{o(k)j}.x_{d(k)l}$

Dans notre cas, seul les inégalités $y_{kjl} \geq x_{o(k)j} + x_{d(k)l} - 1$ et $y_{kjl} \geq 0$ interviennent et on peut finalement écrire la contrainte de la manière suivante :

$$\begin{array}{rll} & z_{jl} \geq \sum_{k\in D} t_{o(k)d(k)} x_{o(k)j} x_{d(k)l} & \forall j \forall l \\ \Leftrightarrow & \exists y \left\{ \begin{array}{ll} z_{jl} \geq \sum_{k\in D} t_{o(k)d(k)} y_{kjl} & \forall j \forall l \\ y_{kjl} \geq x_{o(k)j} + x_{d(k)l} - 1 & \forall k \forall j \forall l \\ y_{kjl} \geq 0 & \forall k \forall j \forall l \end{array} \right. \end{array}$$

Dans [YAMGOU], on montre que

$$\begin{array}{rll} & \exists y \left\{ \begin{array}{ll} z_{jl} \geq \sum_{k\in D} t_{o(k)d(k)} y_{kjl} & \forall j \forall l \\ y_{kjl} \geq x_{o(k)j} + x_{d(k)l} - 1 & \forall k \forall j \forall l \\ y_{kjl} \geq 0 & \forall k \forall j \forall l \end{array} \right. & \\ \Leftrightarrow & z_{jl} \geq \sum_{k\in D'} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1) & \forall j, \forall l, \forall D' \in D \end{array}$$

Ici, il y a un nombre exponentiel de contraintes car on considère chaque sous-ensemble de D. Mais pour j et l fixés, il existe un D^*, tel que ces contraintes seront équivalentes à $z_{jl} \geq \sum_{k\in D^*} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)$

D^* dépend clairement de j et l et on peut le noter D^*_{jl} et la définir ainsi : $D^*_{jl} = \{k \in D / x_{o(k)j} + x_{d(k)l} - 1 \geq 0\}$

Ce qui montre que pour x fixé, c'est-à-dire lorsqu'on connaît les sites, il est facile (faisable en temps polynomial) de déterminer la contrainte correspondante à chaque z_{jl}. De ce fait, on peut générer ces contraintes de manière dynamique.

2.2 Modèle déterministe

Le programme linéaire ainsi formé est le suivant :

$$
\begin{array}{lcll}
\min & \sum_j (c_a.u_j^a + c_b.u_j^b) + \sum_j \sum_l B_{jl} z_{jl} + \sum_i \sum_j A_{ij} x_{ij} & & \\
\text{s.c.} & x_{ij} \leq x_{jj} & \forall i \forall j & \\
& \sum_j x_{ij} = 1 & \forall i & \\
& d_j^a \geq \sum_{i \in I} t_i^{in} x_{ij} & \forall j & \\
& d_j^a \geq \sum_{i \in I} t_i^{out} x_{ij} & \forall j & \\
& z_{jl} \geq \sum_{k \in D_{jl}^*} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1) & \forall j \forall l & (1) \\
& K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{jl} & \forall j & \\
& K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{lj} & \forall j & \\
& N_a u_j^a \geq d_j^a & \forall j & \\
& N_b u_j^b \geq d_j^b & \forall j & \\
& z_{jl} \geq 0 & \forall j \forall l & \\
& x_{ij} \in \{0,1\} & \forall i \forall j & \\
& d_j^a \geq 0 & \forall j & \\
& d_j^b \in \mathbb{N} & \forall j & \\
& u_j^a \in \mathbb{N} & \forall j & \\
& u_j^b \in \mathbb{N} & \forall j &
\end{array}
$$

Il n'est pas nécessaire de contraindre la variable d_j^a à être un entier vu les contraintes auxquelles elle est soumise.

Les contraintes (1) sont génerées dynamiquement. Durant la résolution du problème en nombre entiers, le solveur effectue une suite d'étapes de Branch&Bound. Il obtient des solutions fractionnaires et à chaque solution fractionnaire, l'algorithme tente de trouver une solution entière proche de cette solution. Pour introduire les contraintes de manière dynamique, on fait éxecuter l'algorithme suivant : A chaque noeud de branchement, on regarde l'ensemble de paires de sites choisies dans la solution courante. Puis, pour chaque paire (j, l), on génère la contrainte correspondante, et si la contrainte est effectivement violée, on l'ajoute au problème en cours de résolution.

On veut mettre en place un modèle plus robuste, capable de tenir compte des aléas dans les données en entrée. On suppose donc qu'on ne connaît pas de manière exacte les coûts de connexion "access" et "backbone", ni le trafic entre chaque paire de client.

3 Résumé des modèles stochastiques dans la littérature

Il existe plusieurs façons de tenir compte de l'aléatoire dans les données de problèmes d'optimisation.

Un travail bibliographique concernant la stochastique dans les problèmes d'optimisation a été effectué. Il y a plusieurs façons de tenir compte de l'aspect aléatoire des données. En règle générale, on veut optimiser l'espérance dans le temps d'une fonction objectif.

On peut par exemple penser à un modèle où les décisions sont prises en plusieurs étapes, avant et après la réalisation d'un évenement aléatoire. Il s'agit de modèles dits "multi-étapes avec recours". Par exemple dans notre problème, on prendrait une première décision en plaçant

des sites, en reliant les clients à ces sites et en plaçant un certain nombre de cartes sur les sites, puis, on observerait le trafic réel et on modifierait la localisation et le dimensionnement des sites en fonction de l'évenement survenu. Et ceci, en optimisant en moyenne sur l'ensemble des éventualités qui peuvent survenir (l'ensemble de trafic et/ou de coût de connexion qu'on pourrait avoir). Il s'agit de planification multi-période.

Une autre façon de voir les choses est d'essayer d'avoir une solution robuste. C'est-à-dire que la solution fournie reste faisable quelque soit l'évenement réalisé. On suppose donc avoir une matrice A dont les éléments A_{ij} sont incertains et peuvent avoir des valeurs comprises entre $\underline{A_{ij}}$ et $\overline{A_{ij}}$ et on optimise sous la contrainte $Ax \leq b$. Si on veut une solution faisable quelque soit l'évenement qui arrive, on doit résoudre le programme linéaire avec le pire cas : $\overline{A}x \leq b$

Ceci donne bien sur des solutions dont le coût est très élevé par rapport aux solutions qu'on pourrait avoir si on ne tenait pas compte de l'aspect aléatoire et qu'on donnait en entrée la moyenne des données par exemple. Ceci est assez intuitif dans le sens où le programme va accorder une grande importance aux cas les plus défavorables, bien qu'ils arrivent rarement.

L'idée naturelle consiste à optimiser en négligeant les cas les plus défavorables, si possible de manière paramètrable. Il y a eu beaucoup de travaux qui ont été effectués dans ce sens.

3.1 Programmation stochastique à 2 étapes

3.1.1 Première Formulation

Dans ce genre de problème, on considère qu'on peut prendre les décisions en deux temps. Une première fois avant toute évenement, puis dans un deuxième temps, connaissant l'évenement arrivé. Ces modèles sont étudiés dans [BIRLOU].

Si on appelle x la variable de décision de la 1ère étape, et y la variable de décision de la 2ème étape, y dépendra de l'éventualité ω. Puisque y est décidé après x, il dépend aussi de x. On note alors $y(x, \omega)$.

Le problème est alors de chercher à optimiser l'ensemble en moyenne en fonction des éventualités.

Classiquement, le problème d'optimisation détérministe à une étape s'écrirait :

$$\begin{array}{ll} \min & cx \\ \text{s.c.} & Ax \geq b \\ & x \geq 0 \end{array}$$

Dans notre étude, on considère également y qui va intervenir dans le coût, pour ne pas perdre de généralité, on considère que le coût peut dépendre de l'éventualité w. Le coût des décisions de y s'écrit donc : $q(\omega)y(\omega)$. Les contraintes de y dépendent de la décision prise pour x et de l'évenement qui est survenu. Si ces éléments sont linéairement liés, on peut écrire : $T(\omega)x + Wy(\omega) \geq h(\omega)$ On suppose que W n'est pas stochastique, on dit alors qu'on est en recours fixé. De plus, on essaye de minimiser en moyenne. On prend alors l'éspérance en ω.

Le problème stochastique à 2 étapes avec recours fixé s'écrit donc :

$$\begin{array}{ll} \min & cx + E[q(\omega)y(\omega)] \\ \text{s.c.} & Ax \geq b \\ & T(\omega)x + Wy(\omega) \geq h(\omega) \\ & x \geq 0, y(\omega) \geq 0 \end{array}$$

Pour x et ω fixés, $y(\omega)$ est la solution du programme :

$$Q(x,\omega) = \min\{q(\omega)y | Wy \geq h(\omega) - T(\omega)x, y \geq 0\}$$

De plus, on note : $Q(x) = E[Q(x,\omega)]$.

On peut alors formuler notre problème de manière plus compacte :

$$\begin{array}{ll} \min & cx + Q(x) \\ \text{s.c.} & Ax \geq b \\ & x \geq 0 \end{array}$$

3.2 Autre façon de voir les choses

[KALWAL] donne une formulation plus générale des programmes stochastiques. Ceci permet de relier les modèles multi-étapes et les contraintes probabilistes avec une unique formulation :

$$\begin{array}{lll} \min & g_0(x,\omega) & \\ \text{s.c.} & g_i(x,\omega) \leq 0 & i = 1..m \\ & x \in X \subset \mathbb{R}^n & \end{array}$$

On pose :

$$g_i^+(x,\omega) = \begin{cases} 0 & \text{si } g_i(x,\omega) \leq 0, \\ g_i(x,\omega) & \text{sinon} \end{cases}$$

On dira alors que la $i$ème contrainte a été violée si $g_i^+(x,\omega) > 0$ pour une décision x et une éventualité ω. Donc, après observation, on peut fournir un recours ou une activité de deuxième étape $y_i(\omega)$ qui permettra de compenser, si besoin est, la contrainte i violée : $g_i(x,\omega) - y_i(\omega) \leq 0$. Pour rester réaliste, le fait de permettre une telle action doit avoir un coût, on associe alors à chaque y_i, un coût q_i. Le coût total du recours donné par la fonction de recours est exprimée par

$$Q(x,\omega) = \min_y \{\sum_{i=1}^{m} q_i y_i(\omega) | y_i(\omega) \geq g_i^+(x,\omega), i = 1, ..., m\} \tag{2}$$

De manière plus générale, on peut prendre un vecteur $y \in Y \subset \mathbb{R}^{\bar{n}}$, une matrice W (matrice de recours) de taille $m \times \bar{n}$ et un vecteur coût $q \in \mathbb{R}^{\bar{n}}$, avec $\bar{n}$ arbitraire et Y un polyèdre dans $\mathbb{R}^{\bar{n}}$. La fonction de recours devient :

$$Q(x,\omega) = \min_y \{qy | Wy \geq g_i^+(x,\omega), y \in Y\} \tag{3}$$

Une illustration de ce cas est l'exemple d'une usine qui doit satisfaire des demandes de m types de produits. Une fois les produits fabriqués, (une fois x décidé), la demande peut etre aléatoire (dépendre de ω). Le terme $g_i^+(x,\omega)$ peut etre interpreté comme le manque du produit i. Si $g_i^+(x,\omega) > 0$, l'entreprise doit trouver un moyen de fournir le produit i autrement. On peut alors penser qu'elle a la possibilité d'acheter des paquets à d'autres entreprises. Ces paquets contiennent des produits qu'elle est censée fournir pour satisfaire les demandes. On suppose qu'elle a le choix parmi $\bar{n}$ paquets. Le nombre de produits de type i dans le paquet j est donné par W_{ij}. L'achat du paquet j implique un coût de q_j avec $j \in 1...\bar{n}$

Un cas particulier pourrait etre de prendre $W = I$ la matrice identité, comme c'est exprimé dans 2. Dans ce cas, chaque "paquet" de produit corresponderait à un et un seul produit, autrement dit cela signifierait que l'entreprise achète ailleurs les produits à fournir pour satisfaire la demande.

3.3 Programmation sous contraintes probabilistes

Dans certains problèmes, trouver la meilleure solution "en moyenne" peut conduire à des résultats très peu efficaces particulièrement pour certaines éventualités.

Parfois, on préfère optimiser la fonction objectif en considérant la possibilité de ne pas satisfaire toutes les demandes. On met alors une condition faisant intervenir une mesure probabiliste sur les contraintes. Les problèmes de programmation sous contraintes probabilistes s'écrivent ainsi :

$$\begin{array}{ll} \min & \mathbb{E}[c(\omega)x] \\ \text{s.c.} & \mathbb{P}(A(\omega)x \geq b(\omega)) \geq \alpha \\ & x \geq 0 \end{array}$$

avec $\alpha \in]0, 1]$.

Ici, il s'agit d'un modèle à contrainte de probabilités jointes. Ce type de modèle est difficilement traitable. Si on veut un paramètre α_j différent pour chaque contrainte, on écrit le modèle avec contraintes de probabilités séparées :

$$\begin{array}{lll} \min & \mathbb{E}[c(\omega)x] & \\ \text{s.c.} & \mathbb{P}(A_j(\omega)x \geq b_j(\omega)) \geq \alpha_j & \forall j \\ & x \geq 0 & \end{array}$$

avec $\alpha_j \in]0, 1]$.

3.4 Approche robuste

Dans l'optimisation robuste, la loi des probabilités des variables aléatoires n'est pas nécessairement connue, on exprime leur caractère incertain soit par un ensemble de scénarios dénombrable, soit par un support. Et on cherche à minimiser le coût ou le regret du plus mauvais cas.

- optimisation du coût min-max : On tente de minimiser le coût dans le pire scénario.

$$\begin{array}{lll} \min & \max_k c_k x & \\ \text{s.c.} & A_k x \geq b_k & \forall k \in \{1, ..., K\} \\ & x \in X & \end{array} \quad \Leftrightarrow \quad \begin{array}{lll} \min & z & \\ \text{s.c.} & c_k x \leq z & \forall k \in \{1, ..., K\} \\ & A_k x \geq b_k & \forall k \in \{1, ..., K\} \\ & x \in X & \end{array}$$

- optimisation du regret. Pour chaque scénario k, on trouve la meilleure solution z_k. L'optimisation du regret consiste alors à minimiser la différence entre la valeur de la décision sans connaissance des évenements incertains et z_k.

$$\begin{array}{lll} \min & \max_k c_k x - z_k & \\ \text{s.c.} & A_k x \geq b_k & \forall k \in \{1, ..., K\} \\ & x \in X & \end{array} \quad \Leftrightarrow \quad \begin{array}{lll} \min & z & \\ \text{s.c.} & c_k x \leq z_k + z & \forall k \in \{1, ..., K\} \\ & A_k x \geq b_k & \forall k \in \{1, ..., K\} \\ & x \in X & \end{array}$$

3.5 Algorithmes de résolution

3.5.1 Programmation stochastique à deux étapes

Pour résoudre les problèmes de programmation stochastique à deux étapes, on utilise la décomposition de Benders.

La décomposition de Benders est une méthode très efficace pour la résolution des problèmes qui ont la forme suivante :

$$\begin{array}{lllll} T_1x & +W_1y_1 & & & = h_1 \\ T_2x & & +W_2y_2 & & = h_2 \\ T_3x & & & +W_3y_3 & = h_2 \\ & & & & \vdots \\ T_Kx & & & +W_Ky_K & = h_K \end{array} \tag{4}$$

Les phases de l'algorithme sont :

Etape 1 : Initialisation
$v := 1$ { n° d'itération }
$UB := +\infty$ { borne supérieure }
$LB := -\infty$ { borne inférieure }
Résoudre le problème initial :

$$\begin{array}{ll} \min & cx \\ \text{s.c.} & Ax = b \\ & x \geq 0 \end{array}$$

On pose $\bar{x}^v := \bar{x}^1 := x^*$ avec x^* la solution du problème.

Etape 2 : Les sous-problèmes
Pour toute éventualité $\omega \in \Omega$ faire {
Résoudre le sous-problème :

$$\begin{array}{ll} \min & q_\omega y_\omega \\ \text{s.c.} & W_\omega y_\omega = h_\omega - T_\omega \bar{x}^v \\ & y_\omega \geq 0 \end{array}$$

On affecte :
$\bar{y}^v_\omega := \bar{y}^*_\omega$ La valeur optimale du problème
$\bar{\pi}^v_\omega := \bar{\pi}^*_\omega$ La valeur optimale du dual

} fin pour.
$UB := \min\{UB, c\bar{x}^v + \sum_{\omega\in\Omega} p_w d_w \bar{y}^v_\omega\}$

Etape 3 : Test de convergence
si $\frac{UB-LB}{1+LB} \leq TOL$:
STOP. On a atteint la précision souhaitée. Renvoyer la valeur $\bar{x}^v$.

Etape 4 : Problème principal
Résoudre le problème principal :

$$\begin{array}{lrl} \min & cx & +\theta \\ \text{s.c.} & Ax & = b \\ & \theta & \geq \sum_{\omega\in\Omega} p_\omega(-\bar{\pi}^l_\omega[T_\omega x + W_\omega \bar{y}^l_\omega - h_\omega]), l = 1, ..., v-1 \\ & x & \geq 0 \end{array}$$

$\bar{x}^v := x^*$ (la meilleure solution)
$\bar{\theta}^v := \theta^*$
$LB := c\bar{x}^v + \bar{\theta}^v$
Aller à l'étape 2.

Dans un premier temps, en phase d'initialisation, on trouve une solution optimal sur les variables de décision de première étape au problème dépourvu de son aspect stochastique.

Puis avec la meilleure solution de x qu'on a en main, on regarde ce qui se passe dans chaque scénario pour résoudre la minimisation du coût des décisions de la deuxième étape. Ceci nous permet d'avoir une borne supérieure pour le problème global.

On évalue également la borne inférieure en utilisant les $\bar{\pi}^l_w$: valeurs optimales du dual du sous-problème.

Lorsqu'on a atteint une précision arbitrairement fixée, on arrete. On connait alors les décisions à prendre avant toute évenement stochastique (les variables de décision de première étape) et les décisions à prendre pour toute éventualité qui peut survenir, et le tout en ayant minimisé la valeur moyenne de coût de l'ensemble des opérations.

3.5.2 Equivalent déterministe du problème stochastique sous contraintes probabilistes

Dans cette section, nous allons essayer de transformer les contraintes de type $\mathbb{P}(Ax \leq b) \geq \alpha$ en un ensemble de contraintes détérministes. Ceci est possible lorsque :

- les variables aléatoires suivent une loi stable par la somme,
- ou lorsqu'on connait le support fini et la moyenne des variables aléatoires.

De plus, comme les variables de décision sont booléennes, on peut linéariser les expressions obtenues.

Cas de variables gaussiennes :

On suppose que les coefficients de la matrice de contraintes A sont normalement distribués et qu'ils sont indépendants. Nous allons étudier le cas des contraintes séparées. Ainsi, dans les formules qui suivent on ne considère que la ligne j de la matrice.

On note μ_i (resp. σ_i) l'espérance (resp. l'écart-type) de la variable A_i.

$$\sum_i A_i x_i \rightsquigarrow \mathcal{N}\big(\sum_i \mu_i x_i, \sum_i \sigma_i^2 x_i^2\big)$$

on a donc

$$\frac{\sum_i A_i x_i - \sum_i \mu_i x_i}{\sqrt{\sum_i \sigma_i^2 x_i^2}} \rightsquigarrow \mathcal{N}\big(0,1\big)$$

et

$$\frac{\sum_i \mu_i x_i - \sum_i A_i x_i}{\sqrt{\sum_i \sigma_i^2 x_i^2}} \rightsquigarrow \mathcal{N}\big(0,1\big) \qquad \text{par symétrie}$$

On note Φ la fonction de répartition de la loi normale standard $\mathcal{N}(0,1)$.

$$
\begin{array}{ll}
& \mathbb{P}(\sum_i A_i x_i \geq h) \geq \alpha \\
\Leftrightarrow & \mathbb{P}\big(\frac{\sum_i A_i x_i - \sum_i \mu_i x_i}{\sqrt{\sum_i \sigma_i^2 x_i^2}} \geq \frac{h - \sum_i \mu_i x_i}{\sqrt{\sum_i \sigma_i^2 x_i^2}}\big) \geq \alpha \\
\Leftrightarrow & \mathbb{P}\big(\frac{\sum_i \mu_i x_i - \sum_i A_i x_i}{\sqrt{\sum_i \sigma_i^2 x_i^2}} \leq \frac{\sum_i \mu_i x_i - h}{\sqrt{\sum_i \sigma_i^2 x_i^2}}\big) \geq \alpha \\
\Leftrightarrow & \Phi\big(\frac{\sum_i \mu_i x_i - h}{\sqrt{\sum_i \sigma_i^2 x_i^2}}\big) \geq \alpha \\
\Leftrightarrow & \sum_i \mu_i x_i - h \geq \Phi^{-1}(\alpha)\sqrt{\sum_i \sigma_i^2 x_i^2} \\
\Leftrightarrow & \left\{ \begin{array}{l} (\sum_i \mu_i x_i - h)^2 \geq (\Phi^{-1}(\alpha))^2 . \sum_i \sigma_i^2 x_i^2 \\ \sum_i \mu_i x_i - h \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} \sum_i \mu_i^2 x_i^2 + \sum_i \sum_{j \neq i} \mu_i \mu_j x_i x_j - 2\sum_i \mu_i x_i h + h^2 \geq (\Phi^{-1}(\alpha))^2 . \sum_i \sigma_i^2 x_i^2 \\ \sum_i \mu_i x_i - h \geq 0 \end{array} \right.
\end{array}
$$

On travaille avec des variables booléennes, d'où $x_i^2 = x_i$. Pour les multiplications $x_i x_j$ on insère de nouvelles variables $y_{ij} = x_i x_j$ en définissant $y_{ij} \leq x_i$, $y_{ij} \leq x_j$, $y_{ij} \geq x_i + x_j - 1$ et $y_{ij} \geq 0$. Ce qui nous donne :

$$
\mathbb{P}(\sum_i A_i x_i \geq h) \geq \alpha \quad \Leftrightarrow \quad \exists y \text{ tq } \left\{ \begin{array}{l} \sum_i (\mu_i^2 - 2\mu_i h) x_i + \sum_i \sum_{j \neq i} \mu_i \mu_j y_{ij} + h^2 \geq (\Phi^{-1}(\alpha))^2 . \sum_i \sigma_i^2 x_i \\ \sum_i \mu_i x_i - h \geq 0 \\ y_{ij} \leq x_i, y_{ij} \leq x_j, y_{ij} \geq 0 \qquad i \neq j \end{array} \right.
$$

Donc, dans le cas de distributions gaussiennes, on arrive à donner une nouvelle formulation déterministe linéaire du problème stochastique en insérant un nombre polynomial de nouvelles variables continues.

Cas de variables de support borné :

Dans le cas où la matrice de contraintes A a pour élément des variables aléatoires dont on ne connait pas les lois de probabilité mais seulement leurs supports et leurs moyennes, on peut approcher le résultat grace à la borne de Hoeffding.

Théorème 1 *Soit $X_1, X_2, ..., X_n$, des variables aléatoires indépendantes telles que $\forall i \in \{1, ..., n\}, \mathbb{P}(X_i \in [\alpha_i, \beta_i]) = 1$. Alors, en notant $S = \sum_i X_i$,*

$\forall \tau \geq 0, \mathbb{P}(S \geq \mathbb{E}(S) + \tau) \leq exp\left(-\frac{2\tau^2}{\sum_i (\beta_i - \alpha_i)^2}\right)$.

On travaille sur une ligne donnée de la matrice.

Notons $\mu(x) = \mathbb{E}[\sum_{i \in I_u} A_i x_i]$, avec I_u : l'ensemble des colonnes ayant pour élément une variable aléatoire.

On suppose que les variables aléatoires $\{A_i\}_{i \in I_u(j)}$ sont indépendantes. Le théorème de Hoeffding nous permet d'écrire :

$$
\mathbb{P}(\sum_i A_i x_i \geq h) \leq \exp(-\frac{2d(x)^2}{\sum_{i \in I_u} \delta_i^2 x_i^2})
$$

où $d(x) = h - \sum_{i \notin I_u} A_i x_i - \mu(x)$.

3.5.3 Heuristiques pour la programmation sous contraintes probabilistes

On considère les programmes sous contraintes probabilistes :

$$\max \left\{ cx \mid P(Ax \leq b) \geq 1 - \varepsilon,\, x \in \{0,1\}^n \right\} \tag{5}$$

On suppose ici que certains éléments de la matrice A ne sont pas connues de manière exacte. On suppose connaitre le support des éléments : pour un élément A_{ji}, on suppose qu'il est dans l'intervalle : $[\underline{A}_{ji}, \overline{A}_{ji}]$ avec $\overline{A}_{ji} > \underline{A}_{ji}$. De plus, pour une ligne j, on note $I_u(j)$ l'ensemble de numéros de colonnes correspondantes à des éléments de A uncertains. On note également $\delta_{ji} = \overline{A}_{ji} - \underline{A}_{ji}$ J est l'ensemble des lignes et I est l'ensemble des colonnes.

Nous définissons le problème RILP(γ) où γ est un paramètre du problème :

$$\begin{array}{lll} \max & cx & \\ \text{s.t.} & \sum_{i \in I_u(j)} \underline{A}_{ji} x_i + \Delta_j(x) + \sum_{i \notin I_u(j)} A_{ji} x_i \leq b_j, & \forall j \in J \\ & x \in \{0,1\}^n & \forall i \in I \end{array} \tag{6}$$

avec $\Delta_j(x) = \min \left\{ \sum_{i \in I_u(j)} \delta_{ji} x_i, \gamma_j . \sum_{i \in I_u(j)} \delta_{ji} \right\}$

Nous allons voir comment résoudre de manière approchée et en temps raisonnable le programme 5. Il s'agit donc d'un problème de programmation en nombres entiers mais les résultats restent valide et les algorithmes sont facilement adaptables si une partie des variables est réelle.

De meme, on étudie le modèle avec les contraintes de probabilités jointes. Le modèle avec contraintes de probabilités séparées consiste à choisir un ε_j différente pour chaque ligne j de la matrice A. Dans ce cas, les méthodes proposées sont également facilement adaptables.

Un résultat montre que les problèmes 5 et 6 sont équivalents. De plus le théorème de Hoeffding permet d'avoir une borne inférieure pour la mesure de probabilité de la contrainte.

Un lemme important est utilisé par l'algorithme rapide.

Lemme 1 *Soit x^* une solution optimale de RILP(γ). Alors, x^* est aussi une solution optimale de RILP(γ'), où pour tout $j \in J$:*

- *si* $\sum_{i \in I_u(j)} \delta_{ji} x_i^* \leq \gamma_j \sum_{i \in I_u(j)} \delta_{ji}$ *:* $\gamma'_j = 1$,
- *sinon :* $\gamma'_j = \left(b_j - \sum_{i \in I_u(j)} \underline{A}_{ji} x_i^* - \sum_{i \notin I_u(j)} \underline{A}_{ji} x_i^* \right) / \sum_{i \in I_u(j)} \delta_{ji}$.

Cela veut dire qu'avec une solution x^* de RILP(γ), on peut trouver un autre parametre γ' qui garde la meme solution x^*. L'idée est donc de résoudre RILP(γ), et d'augmenter γ jusqu'à satisfaire la contrainte de probabilité. On veut donc introduire une $\gamma' \geq \gamma$

Algorithme rapide	
Etape 0 :	Affecter $\gamma = 0$.
Etape 1 :	Résoudre RILP(γ), on note x la solution optimale.
Etape 2 :	Calculer (ou evaluer une borne de) $P(Ax \leq b)$.
Etape 3 :	Si $P(Ax \leq b) < 1 - \varepsilon$:
	Calculer $\gamma' \geq \gamma$ comme décrit par le lemme 1, et affecter $\gamma \leftarrow \gamma'$;
	Choisir $j \in J$ tel que $\gamma_j = \min_{k \in J} \gamma_k$;
	Affecter $\gamma_j \leftarrow \gamma_j + 1 / \sum_{i \in I_u(j)} \delta_{ji}$ et aller à l'étape 1.
Etape 4 :	Si $P(Ax \leq b) \geq 1 - \varepsilon$: STOP.

Cet algorithme a comme inconvénient d'etre lent en pratique à cause de la résolution exacte de RILP.

Il existe une heuristique permettant d'approcher la résolution de RILP.

Résolution par heuristique de RILP(γ)

Etape 1 : Soit x' la solution optimale de :

$$\begin{array}{lll} \max & cx & \\ \text{s.t.} & \sum_{i\in I_u(j)} \underline{A}_{ji}x_i + \gamma_j . \sum_{i\in I_u(j)} \delta_{ji} + \sum_{i\notin I_u(j)} A_{ji}x_i \leq b_j, & \forall j \in J \\ & x \in \{0,1\}^n & \forall i \in I \end{array}$$

Etape 2 : Avec x', calculer γ' comme décrit par le lemme 1.

Etape 3 : Pour tout $j \in J$, si $\gamma'_j = 1$, remplacer la contrainte j du problème précédent par :

$$\sum_{i\in I_u(j)} \overline{A}_{ji}x_i + \sum_{i\notin I_u(j)} A_{ji}x_i \leq b_j$$

Etape 4 : Résoudre le nouveau problème obtenu, et affecter à x^* la solution obtenue.

4 Prise en compte de l'incertain dans notre problème

C'est avec les contraintes probabilistes qu'on se propose de résoudre le problème d'optimisation en tenant compte de l'aspect aléatoire dans les données. On va donc tenter de réduire l'espérance du coût, en satisfaisant les contraintes liées à la satisfaction des demandes en débit avec une certaine probabilité α.

4.1 Localisation et dimensionnement dans un contexte incertain

Le variables aléatoires seront notés avec des tildes : $\widetilde{t_{im}}, \widetilde{A_{ij}}, \widetilde{B_{jl}}, \widetilde{c_a}, \widetilde{c_b}$.

On veut minimiser :

$$\mathbb{E}\big[\sum_j (\widetilde{c_a}.u_j^a + \widetilde{c_b}.u_j^b) + \sum_j \sum_l \widetilde{B_{jl}} z_{jl} + \sum_i \sum_j \widetilde{A_{ij}} x_{ij}\big]$$

par la linéarité de l'espérance, cela revient à minimiser :

$$\sum_j (\mathbb{E}[\widetilde{c_a}].u_j^a + \mathbb{E}[\widetilde{c_b}].u_j^b) + \sum_j \sum_l \mathbb{E}[\widetilde{B_{jl}}] z_{jl} + \sum_i \sum_j \mathbb{E}[\widetilde{A_{ij}}] x_{ij}$$

Les données $\widetilde{c_a}, \widetilde{c_b}, \widetilde{A_{ij}}$ et $\widetilde{B_{jl}}$ n'intervenant que dans la fonction objectif, on peut simplement remplacer ces variables aléatoires par leur moyennes qu'on va noter respectivement c_a, c_b, A_{ij} et B_{jl}.

De plus, on n'utilisera plus les définitions suivantes :
$t_i^{in} = \left\lceil \frac{\sum_{m\in I} t_{mi}}{K_a} \right\rceil$

$t_i^{out} = \left\lceil \frac{\sum_{m\in I} t_{im}}{K_a} \right\rceil$

puisqu'elles font intervenir de la non-linéarité.

Ainsi, le programme linéaire du problème d'optimisation en contexte incertain est :

$$
\begin{array}{lcll}
\min & \sum_j (c_a.u_j^a + c_b.u_j^b) + \sum_j \sum_l B_{jl} z_{jl} + \sum_i \sum_j A_{ij} x_{ij} & & \\
\text{s.c.} & x_{ij} \leq x_{jj} & \forall i \forall j & \\
& \sum_j x_{ij} = 1 & \forall i & \\
& \mathbb{P}(d_j^a \geq \frac{1}{K_a} \sum_{i \in I} \sum_{m \in I} \tilde{t}_{mi} x_{ij}) \geq \alpha_a & \forall j & \\
& \mathbb{P}(d_j^a \geq \frac{1}{K_a} \sum_{i \in I} \sum_{m \in I} \tilde{t}_{im} x_{ij}) \geq \alpha_a & \forall j & \\
& \mathbb{P}(z_{jl} \geq \sum_{k \in D'} \tilde{t}_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1) \quad \forall D' \subseteq D) \geq \alpha_t & \forall j \forall l & (7) \\
& K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{jl} & \forall j & \\
& K_b d_j^b \geq \sum_{l \in I \setminus \{j\}} z_{lj} & \forall j & \\
& N_a u_j^a \geq d_j^a & \forall j & \\
& N_b u_j^b \geq d_j^b & \forall j & \\
& z_{jl} \geq 0 & \forall j \forall l & \\
& x_{ij} \in \{0,1\} & \forall i \forall j & \\
& d_j^a \in \mathbb{N} & \forall j & \\
& d_j^b \in \mathbb{N} & \forall j & \\
& u_j^a \in \mathbb{N} & \forall j & \\
& u_j^b \in \mathbb{N} & \forall j &
\end{array}
$$

Dans la version probabiliste, on est obligé d'avoir une contrainte d'integrité pour la variable d_j^a.

4.1.1 Cas de variables gaussiennes

Nous allons donner un équivalent déterministe au problème de contraintes probabilistes ainsi formé.

On suppose que

$$\tilde{t}_{im} \rightsquigarrow \mathcal{N}(\mu_{im}, \sigma_{im}^2)$$

Pour alléger les notations, dans cette partie on note :

$$\sum_m \tilde{t}_{im} \rightsquigarrow \mathcal{N}(\mu_i, \sigma_i^2)$$

On peut donner un équivalent détérministe aux contraintes probabilistes de satisfaction de demande côté réseau accès :

Pour un site j :

$$
\begin{array}{ll}
& \mathbb{P}(\sum_i \sum_m \tilde{t}_{im} x_{ij} \leq K_a d_j^a) \geq \alpha_a \\
\Leftrightarrow & \mathbb{P}(\frac{\sum_i \sum_m \tilde{t}_{im} x_{ij} - \sum_i \mu_i x_{ij}}{\sqrt{\sum_i \sigma_i^2 x_{ij}^2}} \leq \frac{K_a d_j^a - \sum_i \mu_i x_{ij}}{\sqrt{\sum_i \sigma_i^2 x_{ij}^2}}) \geq \alpha_a \\
\Leftrightarrow & \Phi(\frac{K_a d_j^a - \sum_i \mu_i x_{ij}}{\sqrt{\sum_i \sigma_i^2 x_{ij}^2}}) \geq \alpha_a \\
\Leftrightarrow & K_a d_j^a - \sum_i \mu_i x_{ij} \geq \Phi^{-1}(\alpha_a) \sqrt{\sum_i \sigma_i^2 x_{ij}^2} \\
\Leftrightarrow & \left\{ \begin{array}{l} (K_a d_j^a - \sum_i \mu_i x_{ij})^2 \geq (\Phi^{-1}(\alpha_a))^2 . \sum_i \sigma_i^2 x_{ij}^2 \\ K_a d_j^a - \sum_i \mu_i x_{ij} \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} {K_a}^2 d_j^{a2} - 2 K_a d_j^a \sum_i \mu_i x_{ij} + \sum_i {\mu_i}^2 {x_{ij}}^2 + \sum_i \sum_{r \neq i} \mu_i \mu_r x_{ij} x_{rj} \geq (\Phi^{-1}(\alpha))^2 \sum_i \sigma_i^2 x_{ij}^2 \\ K_a d_j^a - \sum_i \mu_i x_{ij} \geq 0 \end{array} \right. \\
\Leftrightarrow & \exists y \text{ tq } \left\{ \begin{array}{ll} (\sum_i {\mu_i}^2 - 2 K_a d_j^a \sum_i \mu_i - (\Phi^{-1}(\alpha))^2 \sum_i \sigma_i^2) x_{ij} + \sum_i \sum_{r \neq i} \mu_i \mu_r y_{irj} \geq -{K_a}^2 d_j^{a2} & \\ K_a d_j^a - \sum_i \mu_i x_{ij} \geq 0 & \\ y_{irj} \leq x_{ij}, y_{irj} \leq x_{rj}, y_{irj} \geq 0 & \forall i \neq r \end{array} \right.
\end{array}
$$

Le problème est qu'on est obtient une expression quadratique d_j^{a2}.

4.1.2 Travaux sur les contraintes du trafic dans le réseau "backbone"

Les contraintes probabilistes pour détérminer le trafic circulant entre deux sites sont :

$$\mathbb{P}(z_{jl} \geq \sum_{k \in D'} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1) \quad \forall D' \subseteq D) \geq \alpha_t \qquad \forall j \forall l \tag{8}$$

Ce sont donc des contraintes de probabilités jointes qu'on ne sait pas bien traiter. Nous aimerions avoir :

$$\mathbb{P}(z_{jl} \geq \sum_{k \in D'} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)) \geq \alpha_t \qquad \forall j \forall l \forall D' \subseteq D \tag{9}$$

Nous allons essayer de voir le lien entre (8) et (9).
Notons

$$D^*(x) = \{k \in D / (x_{o(k)j} + x_{d(k)l} - 1) \geq 0\}$$

Pour (8), l'égalité est clairement atteint pour ce $D^*(x)$ et on peut écrire l'équivalence :

$$(8) \Leftrightarrow \mathbb{P}(z_{jl} \geq \sum_{k \in D^*(x)} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)) \geq \alpha_t \qquad \forall j \forall l \text{ avec } t \geq 0$$

D'autre part :

$$
\begin{array}{ll}
& \mathbb{P}(z_{jl} \geq \sum_{k \in D'} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)) \geq \alpha_t \qquad \forall j \forall l \forall D' \subseteq D \\
\Rightarrow & \mathbb{P}(z_{jl} \geq \sum_{k \in D^*(x)} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)) \geq \alpha_t \qquad \forall j \forall l
\end{array}
$$

On a donc (9) $\Rightarrow$ (8).
Montrons que (8) $\Rightarrow$ (9).
Pour $D' \subseteq D^*$,
$\sum_{k \in D'} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1) \leq \sum_{k \in D^*} t_{o(k)d(k)} (x_{o(k)j} + x_{d(k)l} - 1)$ avec $t \geq 0$

et donc

$$\mathbb{P}(\sum_{k\in D'} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1) \leq z_{jl}) \geq \mathbb{P}(\sum_{k\in D^*} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1) \leq z_{jl})$$

$$\begin{array}{rl} & \mathbb{P}(\sum_{k\in D^*} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1) \leq z_{jl}) \geq \alpha_t \\ \Rightarrow & \mathbb{P}(\sum_{k\in D'} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1) \leq z_{jl}) \geq \alpha_t \end{array}$$

Pour $D' \nsubseteq D^*$, on peut écrire $D' = {D_1}' \cup {D_2}'$ avec ${D_1}' \cap {D_2}' = \oslash$ et tel que ${D_1}' \subseteq D^*$ et ${D_2}' \subseteq D \backslash D^*$

$\sum_{k\in D'} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1) = \sum_{k\in {D_1}'} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1)+\sum_{k\in {D_2}'} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1)$

Or $\sum_{k\in {D_1}'} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1) \leq \sum_{k\in D^*} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1)$ et $\sum_{k\in {D_2}'} t_{o(k)d(k)}(x_{o(k)j}+x_{d(k)l}-1) \leq 0$ par définition de $D^*(x)$.

Donc on a également :
$\sum_{k\in D'} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1) \leq \sum_{k\in D^*} t_{o(k)d(k)}(x_{o(k)j} + x_{d(k)l} - 1)$ avec $t \geq 0$

Ainsi (8) $\Rightarrow$ (9).
On peut donc faire sortir l'expression $\forall D' \subseteq D$ de la probabilité sans aucune perte de généralité.

4.1.3 Equivalent déterministe des contraintes probabilistes du trafic dans le réseau "backbone"

Comme pour les contraintes de satisfaction du trafic dans le réseau accès, nous allons tenter de donner un équivalent déterministe des contraintes probabilistes concernant le réseau backbone.

Ca se passe moins bien pour les contraintes qu'on genere dynamiquement pendant la resolution. D'un point de vue informatique, l'ensemble D des demandes est un ensemble de paire de noeud. A chaque $k \in D$, on associe donc $(i, m) \in I^2$.

Pour j et l fixes, on a
$z_{jl} \geq \sum_{k\in D'} t_{o(k)d(k)}.(x_{o(k)j} + x_{d(k)l} - 1) \qquad \forall D' \subseteq D$
$\Leftrightarrow z_{jl} \geq \sum_{k\in D^*} t_{o(k)d(k)}.(x_{o(k)j} + x_{d(k)l} - 1)$
$\Leftrightarrow z_{jl} \geq \sum_{i,m \text{ avec } (i,m)\in I^*} t_{im}.(x_{ij} + x_{ml} - 1)$
avec $I^*(x) = \{(i, m) \in I^2 / x_{ij} + x_{ml} - 1 \geq 0\}$

*** Premier essai**
Ici, on donne la contrainte avec son expression informatique des le depart.
$z_{jl} \geq \sum_i \sum_m t_{im}.(x_{ij} + x_{ml} - 1)$ et $(i, m) \in I^*(x)$

On veut satisfaire cette inegalite avec une certaine probabilite.
$\mathbb{P}(z_{jl} \geq \sum_i \sum_m t_{im}.(x_{ij} + x_{ml} - 1)) \geq \alpha_t$

On tente de faire les memes demarches qu'avant :

$$
\begin{array}{ll}
& \mathbb{P}(z_{jl} \geq \sum_i \sum_m t_{im}.(x_{ij} + x_{ml} - 1)) \geq \alpha_t \\
\Leftrightarrow & \mathbb{P}(\frac{\sum_i \sum_m t_{im}.(x_{ij}+x_{ml}-1) - \sum_i \sum_m \mu_{im}(x_{ij}+x_{ml}-1)}{\sqrt{\sum_i \sum_m {\sigma_{im}}^2 (x_{ij}+x_{ml}-1)^2}} \leq \frac{z_{jl} - \sum_i \sum_m \mu_{im}(x_{ij}+x_{ml}-1)}{\sqrt{\sum_i \sum_m {\sigma_{im}}^2 (x_{ij}+x_{ml}-1)^2}}) \geq \alpha_t \\
\Leftrightarrow & \Phi(\frac{z_{jl} - \sum_i \sum_m \mu_{im}(x_{ij}+x_{ml}-1)}{\sqrt{\sum_i \sum_m {\sigma_{im}}^2 (x_{ij}+x_{ml}-1)^2}}) \geq \alpha_t \\
\Leftrightarrow & z_{jl} - \sum_i \sum_m \mu_{im}(x_{ij} + x_{ml} - 1) \geq \Phi^{-1}(\alpha_t) \sqrt{\sum_i \sum_m {\sigma_{im}}^2 (x_{ij} + x_{ml} - 1)^2} \\
\Leftrightarrow & \left\{ \begin{array}{l} (z_{jl} - \sum_i \sum_m \mu_{im}(x_{ij} + x_{ml} - 1))^2 \geq (\Phi^{-1}(\alpha_t))^2 . \sum_i \sum_m {\sigma_{im}}^2 (x_{ij} + x_{ml} - 1)^2 \\ z_{jl} - \sum_i \sum_m \mu_{im}(x_{ij} + x_{ml} - 1) \geq 0 \end{array} \right. \\
\Leftrightarrow & \{ \ ...
\end{array}
$$

On voit rapidement que ce modèle ne vas pas être traitable de manière efficace.

*** Deuxieme essai**

Une idee consiste a lineariser l'expression de la contrainte plus tard dans les calculs.
On part de :
$z_{jl} \geq \sum_i \sum_m t_{im}.(x_{ij} x_{ml})$

et on arrive a :

$$
\begin{array}{ll}
& \mathbb{P}(z_{jl} \geq \sum_i \sum_m t_{im} x_{ij} x_{ml}) \geq \alpha_t \\
\Leftrightarrow & \left\{ \begin{array}{l} (z_{jl} - \sum_i \sum_m \mu_{im} x_{ij} x_{ml})^2 \geq (\Phi^{-1}(\alpha_t))^2 . \sum_i \sum_m {\sigma_{im}}^2 {x_{ij}}^2 {x_{ml}}^2 \\ z_{jl} - \sum_i \sum_m \mu_{im} x_{ij} x_{ml} \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_i \sum_m \mu_{im} x_{ij} x_{ml} + (\sum_i \sum_m \mu_{im} x_{ij} x_{ml})^2 \geq (\Phi^{-1}(\alpha_t))^2 . \sum_i \sum_m {\sigma_{im}}^2 {x_{ij}}^2 {x_{ml}}^2 \\ z_{jl} - \sum_i \sum_m \mu_{im} x_{ij} x_{ml} \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_i \sum_m \mu_{im} X_{ijml} + (\sum_i \sum_m \mu_{im} x_{ij} x_{ml})^2 \geq (\Phi^{-1}(\alpha_t))^2 . \sum_i \sum_m {\sigma_{im}}^2 X_{ijml} \\ z_{jl} - \sum_i \sum_m \mu_{im} X_{ijml} \geq 0 \\ X_{ijml} \geq x_{ij} + x_{ml} - 1 \\ X_{ijml} \leq x_{ij}, X_{ijml} \leq x_{ml}, X_{ijml} \geq 0 \end{array} \right.
\end{array}
$$

Or :

$$
(\sum_i \sum_m \mu_{im} x_{ij} x_{ml})^2 =
$$

$$
\sum_i (\sum_m \mu_{im}^2 x_{ij}^2 x_{ml}^2 + \sum_m \sum_{n \neq m} \mu_{im} \mu_{in} x_{ij}^2 x_{ml} x_{nl}) + \sum_i \sum_{r \neq i} \sum_m \sum_n \mu_{im} \mu_{rn} x_{ij} x_{ml} x_{rj} x_{nl}
$$

Le produit $x_{ij} x_{ml} x_{rj} x_{nl}$ laisse supposer que c'est intraitable.

*** Troisième essai**

On repart de la definition avec les D.

Pour j,l fixes et x donné, on peut determiner facilement $D^*_{jl}(x)$: c'est l'ensemble des demandes d'origine zone arriere de j et de destination zone arriere de l.

On suppose que :

$$
t_{o(k)d(k)} \rightsquigarrow \mathcal{N}(\mu_k, \sigma_k^2)
$$

$$
\begin{array}{ll}
 & \mathbb{P}(\sum_{k\in D'} t_{o(k)d(k)}.(x_{o(k)j}+x_{d(k)l}-1)\leq z_{jl} \qquad \forall D'\subseteq D)\geq \alpha_t \\
\Leftrightarrow & \mathbb{P}(\sum_{k\in D^*} t_{o(k)d(k)}.(x_{o(k)j}+x_{d(k)l}-1)\leq z_{jl})\geq \alpha_t \\
\Leftrightarrow & \mathbb{P}(\frac{\sum_{k\in D^*} t_{o(k)d(k)}.(x_{o(k)j}+x_{d(k)l}-1)-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)}{\sqrt{\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2}}\leq \frac{z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)}{\sqrt{\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2}})\geq \alpha_t \\
\Leftrightarrow & \Phi(\frac{z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)}{\sqrt{\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2}})\geq \alpha_t \\
\Leftrightarrow & z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)\geq \Phi^{-1}(\alpha_t)\sqrt{\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2} \\
\Leftrightarrow & \left\{\begin{array}{l}(z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1))^2\geq (\Phi^{-1}(\alpha_t))^2.\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2 \\ z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)\geq 0\end{array}\right. \\
\Leftrightarrow & \left\{\begin{array}{l}{z_{jl}}^2-2z_{jl}\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1) \\ \qquad +(\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1))^2 \\ \qquad \geq (\Phi^{-1}(\alpha))^2\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2 \\ z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)\geq 0\end{array}\right. \\
\Leftrightarrow & \left\{\begin{array}{l}{z_{jl}}^2-2z_{jl}\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1) \\ \qquad +\sum_{k\in D^*}\mu_k^2(x_{o(k)j}+x_{d(k)l}-1)^2 \\ \qquad +\sum_{k\in D^*}\sum_{k'\in D^*}\mu_k\mu_{k'}(x_{o(k)j}+x_{d(k)l}-1)(x_{o(k')j}+x_{d(k')l}-1) \\ \qquad \geq (\Phi^{-1}(\alpha))^2\sum_{k\in D^*}\sigma_k^2(x_{o(k)j}+x_{d(k)l}-1)^2 \\ z_{jl}-\sum_{k\in D^*}\mu_k(x_{o(k)j}+x_{d(k)l}-1)\geq 0\end{array}\right.
\end{array}
$$

est-ce vrai que ?... On a $(x_{o(k)j}+x_{d(k)l}-1)^2=(x_{o(k)j}+x_{d(k)l}-1)$. Si oui, les seuls produits doubles sont :

* $x_{o(k)j}.x_{o(k')j}$
* $x_{o(k)j}.x_{d(k')l}$
* $x_{d(k)l}.x_{o(k')j}$
* $x_{d(k)l}.x_{d(k')l}$

Ce qui est tout de même trop pour être traité efficacement par un programme linéaire.

Quatrieme essai

On va travailler avec D a nouveau, mais on va garder les produits de variables le plus longtemps possible.

On va voir ensuite comment lineariser, et exprimer les equations lineaires dans le cas fractionnaire.

$$
\begin{array}{ll}
 & \mathbb{P}(\sum_{k\in D} t_{o(k)d(k)} x_{o(k)j} x_{d(k)l} \leq z_{jl} \geq \alpha_t \\
\Leftrightarrow & \mathbb{P}(\frac{\sum_{k\in D} t_{o(k)d(k)} x_{o(k)j} x_{d(k)l} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l}}{\sqrt{\sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2}} \leq \frac{z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l}}{\sqrt{\sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2}}) \geq \alpha_t \\
\Leftrightarrow & \Phi(\frac{z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l}}{\sqrt{\sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2}}) \geq \alpha_t \\
\Leftrightarrow & z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \geq \Phi^{-1}(\alpha_t) \sqrt{\sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2} \\
\Leftrightarrow & \left\{ \begin{array}{l} (z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l})^2 \geq (\Phi^{-1}(\alpha_t))^2 . \sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2 \\ z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \\ \quad + (\sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l})^2 \\ \quad - (\Phi^{-1}(\alpha_t))^2 \sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2 \\ \quad \geq 0 \\ z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \geq 0 \end{array} \right. \\
\Leftrightarrow & \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \\ \quad + \sum_{k\in D} \mu_k^2 x_{o(k)j}^2 x_{d(k)l}^2 \\ \quad + \sum_{k\in D} \sum_{k'\in D} \mu_k \mu_{k'} x_{o(k)j} x_{d(k)l} x_{o(k')j} x_{d(k')l} \\ \quad - (\Phi^{-1}(\alpha_t))^2 \sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l})^2 \\ \quad \geq 0 \\ z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \geq 0 \end{array} \right.
\end{array}
$$

On peut omettre les carrées, car il s'agit de variables booléennes :

$$
\Leftrightarrow \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \\ \quad + \sum_{k\in D} \mu_k^2 x_{o(k)j} x_{d(k)l} \\ \quad + \sum_{k\in D} \sum_{k'\in D} \mu_k \mu_{k'} x_{o(k)j} x_{d(k)l} x_{o(k')j} x_{d(k')l} \\ \quad - (\Phi^{-1}(\alpha_t))^2 \sum_{k\in D} \sigma_k^2 (x_{o(k)j} x_{d(k)l}) \\ \quad \geq 0 \\ z_{jl} - \sum_{k\in D} \mu_k x_{o(k)j} x_{d(k)l} \geq 0 \end{array} \right.
$$

En posant $D^* = \{k \in D / x_{o(k)j} + x_{d(k)l} - 1 > 0\}$
et $Q = \{(k,k') \in D^2 / x_{o(k)j} + x_{d(k)l} + x_{o(k')j} + x_{d(k')l} - 3 > 0\}$

$$
\Leftrightarrow \left\{ \begin{array}{l} {z_{jl}}^2 - 2 z_{jl} \sum_{k\in D^*} \mu_k (\tilde{x}_{o(k)j} + \tilde{x}_{d(k)l} - 1) \\ \quad + \sum_{k\in D^*} \mu_k^2 (\tilde{x}_{o(k)j} + \tilde{x}_{d(k)l} - 1) \\ \quad + \sum_{(k,k')\in Q} \mu_k \mu_{k'} (\tilde{x}_{o(k)j} + \tilde{x}_{d(k)l} + \tilde{x}_{o(k')j} + \tilde{x}_{d(k')l} - 3) \\ \quad - (\Phi^{-1}(\alpha_t))^2 \sum_{k\in D^*} \sigma_k^2 (\tilde{x}_{o(k)j} + \tilde{x}_{d(k)l} - 1) \\ \quad \geq 0 \\ z_{jl} - \sum_{k\in D^*} \mu_k (\tilde{x}_{o(k)j} + \tilde{x}_{d(k)l} - 1) \geq 0 \end{array} \right.
$$

Malgré tout, nous avons des termes quadratiques dans les expression équivalentes déterministes de ces contraintes probabilistes. C'est pourquoi, il a été judicieux de choisir une heuristique pour résoudre les problèmes probabilistes.

4.2 Heuristique pour le problème stochastique

Les expressions analytiques obtenues pour les contraintes probabilistes ne sont pas linéaires. C'est pourquoi l'implémentation d'heuristiques de la version stochastique du problème semble

nécessaire.

On part de ce qui a été étudié dans le travail bibliographique, à savoir l'algorithme rapide. On utilise des termes de protection sur chaque contrainte de manière à faire augmenter la valeur de la variable correspondante à la prochaine résolution si la probabilité correspondante n'est pas atteinte. Une écriture de l'algorithme pour un programme général pourrait être :

Algorithme rapide	
Etape 0 :	Termes de protection $T_j = 0, \forall j$.
Etape 1 :	Résoudre le programme linéaire déterministe sous $A_j x \leq b_j - T_j, \forall j$
	on note x la solution optimale.
Etape 2 :	Calculer $P(A_j x \leq b_j)$. pour chaque ligne j
Etape 3 :	Si on n'a pas $P(A_j x \leq b_j) \geq \alpha_j, \forall j$:
	Choisir j^* tel que $j^* = \arg\max_j \{\alpha_j - P(A_j x \leq b_j)\}$
	Augmenter le terme de protection $T_{j^*} := T_{j^*} + 1$
	aller à l'étape 1.
Etape 4 :	Si $P(A_j x \leq b_j) \geq \alpha_j, \forall j$: STOP.

5 Implémentation

Les programmes linéaires ont été implémentés en C++ avec la librairie Cplex (www.ilog.com/products/cplex/) très célèbre dans le domaine de la recherche opérationnelle. Cependant, le temps de résolution du problème déterministe était très élevé, et plus encore pour le problème stochastique. C'est pourquoi, l'implémentation d'heuristiques même pour le problème déterministe était un choix naturel.

5.1 Le relaché linéaire

Une relaxation linéaire du problème en entier a donné de meilleures performances en temps d'execution. Le principe consiste à résoudre le problème en omettant les contraintes d'intégrité des variables x_{ij}. Une fois le problème résolu, on ajoute une contrainte de type $x_{i^*j^*} = 1$ avec $(i^*, j^*) = \arg\max_{(i,j)} \{x_{ij} / x_{ij} \neq 1\}$.

Cela permet de se passer des coupes exhaustives du branch & cut utilisées lorsqu'on fait résoudre le problème en nombre entiers de manière exacte. Cela a pour conséquence d'aller beaucoup plus vite puisque l'algorithme ne remet jamais en question un branchement effectué. Le désavantage est qu'on peut passer à côté de la meilleure solution. En effet, parfois il vaut mieux partir dans une direction qui ne semble pas optimum mais dont les noeuds de branchement fils contiennent la meilleure solution du problème.

Cependant, pour de petites instances (de l'ordre de graphes bipartis complets 4x4), cette méthode a donné une solution identique à l'algorithme de résolution exacte.

Rellin	
Etape 0 :	Initialisation : génération des contraintes z_{jl}
Etape 1 :	Résoudre $Ax \leq b$
	on note x la solution optimale.
Etape 2 :	Regarder s'il existe x_j non entier
Etape 3 :	Si oui, choisir x_{j^*} le plus proche de 1 mais $\neq 1$
	ajouter la contrainte $x_{j^*} = 1$
	aller à l'étape 1.
Etape 4 :	Si non : STOP.

Ici, dans la phase d'initialisation, on génère l'ensemble des contraintes sur les $z_j l$ dès le départ. On aimerait donc avoir :

$z_{jl} \geq \sum_i \sum_m t_{im} x_{ij}.x_{ml} \qquad \forall j, \forall l$

Comme le terme quadratique ne va pas pouvoir être résolu par le solveur, on ajoute des variables y_{ijml} pour remplacer les produits $x_{ij}.x_{ml}$.

$$\Leftrightarrow \begin{cases} z_{jl} \geq \sum_i \sum_m t_{im} x_{ij}.x_{ml} & \forall j, \forall l \\ z_{jl} \geq \sum_i \sum_m t_{im} y_{ijml} & \forall j, \forall l \\ y_{ijml} \geq x_{ij} + x_{ml} - 1 & \forall j, \forall l, \forall m, \forall i \\ y_{ijml} \leq x_{ij} & \forall j, \forall l, \forall m, \forall i \\ y_{ijml} \leq x_{ml} & \forall j, \forall l, \forall m, \forall i \\ y_{ijml} \geq 0 & \forall j, \forall l, \forall m, \forall i \end{cases}$$

On génère donc un nombre $O(n^4)$ de nouvelles variables et les contraintes y correspondantes. C'est pourquoi la phase d'initialisation prend un certain temps.

L'idée est donc de génerer ces contraintes dynamiquement, c'est-à-dire de les injecter au solveur au fur et à mesure qu'elles sont violées. Ce qui donne la variante "Rellin z_{jl} dynamiques".

Rellin z_{jl} dynamiques	
Etape 0 :	Initialisation.
Etape 1 :	Résoudre $Ax \leq b$
	on note x la solution optimale fractionnaire
Etape 2 :	Regarder s'il existe une contrainte z_{jl} qui est violée dans la solution courante
Etape 3 :	Si oui, insérer toutes les containtes violées dans le modèle
	aller à l'étape 1.
	Si non, aller à l'étape 4.
Etape 4 :	Regarder s'il existe x_j non entier
Etape 5 :	Si oui, choisir x_{j^*} le plus proche de 1
	ajouter la contrainte $x_{j^*} = 1$
	aller à l'étape 1.
Etape 6 :	Si non : STOP.

5.2 Le relaché linéaire et la résolution du problème stochastique

Pour résoudre le problème sous contraintes probabilistes, on utilise l'algorithme rapide. Dans l'étape correspondante à la résolution du problème déterministe, on fait appelle aux algorithmes qu'on vient de voir pour résoudre en temps raisonnable.

Sfa	
Etape 0 :	Initialisation : génération des contraintes z_{jl} et Termes de protection $T_j = 0, \forall j$.
Etape 1 :	Résoudre $A_j x \leq b_j - T_j, \forall j$
	on note x la solution optimale fractionnaire.
Etape 2 :	Regarder s'il existe x_j non entier
Etape 3 :	Si oui, choisir x_{j^*} le plus proche de 1
	ajouter la contrainte $x_{j^*} = 1$
	aller à l'étape 1.
Etape 4 :	Si non : Calculer $P(A_j x \leq b_j)$. pour chaque ligne j
Etape 5 :	Si on n'a pas $P(A_j x \leq b_j) \geq \alpha_j, \forall j$:
	Choisir j^* tel que $j^* = \arg\max_j \{\alpha_j - P(A_j x \leq b_j)\}$
	Augmenter le terme de protection $T_{j^*} := T_{j^*} + 1$
	aller à l'étape 1.
Etape 6 :	Si $P(A_j x \leq b_j) \geq \alpha_j, \forall j$: STOP.

6 Tests numériques

Les tests ont été effectués sur trois instances : $aretes5$, $aretes7$ et $opt10$. Les instances $aretes5$ et $aretes7$ sont des bipartis de 20 noeuds, avec 10 noeuds de chaque côté. Dans $aretes5$, chaque noeud d'en haut a un degré de 5 et dans $aretes7$, chaque noeud d'en haut a un degré de 7. $opt10$ est un graphe quelconque avec 10 noeuds.

6.1 Comparaison Exact/Heuristique

Une implémentation du code qui résoud le problème déterministe de manière a permis de comparer les résultats obtenus avec l'heuristique. Pour les instances $aretes5$ et $aretes7$, le solveur a mis énormément de temps à résoudre. C'est pourquoi, on mentionne seulement la valeur courante de la solution au bout d'un certain temps d'éxecution, avec le gap correspondant affiché par le solveur cplex.

TAB. 1 – Comparaison exact / heuristique, Valeur de la solution

Instances	Exact	Gap Exact	Heuristique	Heur. zjl debut
onur	344753	0,00%	443156	426507
aretes5	29306,8	3,36%	29594,1	29493,4
aretes7	28721,1	5,79%	28529,4	29072

TAB. 2 – Comparaison exact / heuristique, Temps d'éxecution en secondes

Instances	Exact	Heuristique	Heur. zjl debut
onur	2,27	0,16	0,64
aretes5	infini	16,79	12,19
aretes7	infini	38,26	42,15

On compare avec l'heuristique avec et sans l'option $-zjl$ qui impose au programme de mettre ou non les contraintes sur les variables z_{jl} dès le départ.

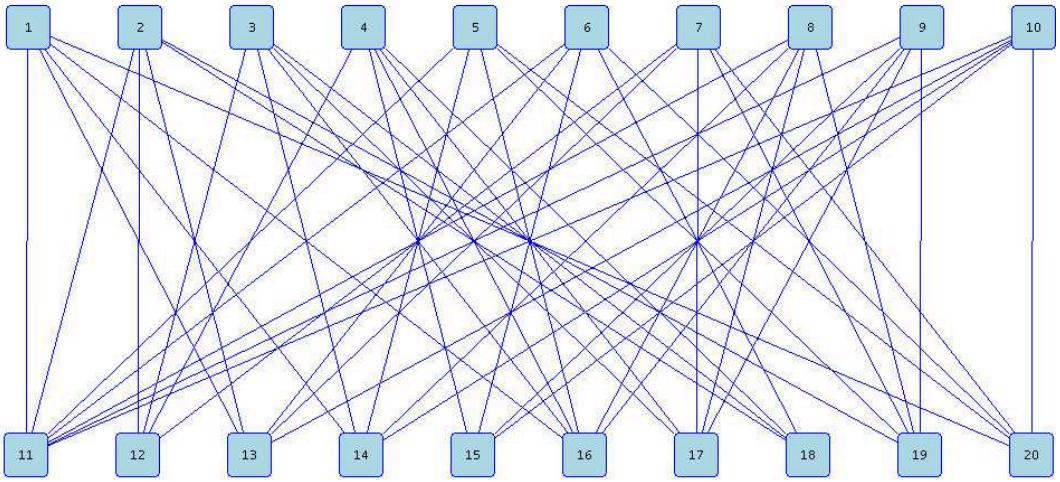


FIG. 1 – Instance *aretes*5

6.2 Influence du pas

On a vu dans l'implémentation de l'heuristique de la résolution du problème stochastique, que l'on fait augmenter un terme de protection à chaque résolution du problème déterministe. On peut jouer sur cette incrémentation pour avoir une solution plus ou moins fine en un temps plus ou moins important. Evidemment, plus le pas de l'incrémentation est grand, moins la solution est bonne, mais moins important est le temps d'éxecution également. Ces résultats résument l'influence du pas du terme de protection des contraintes z_{jl} sur la valeur de la solution et le temps d'éxecution.

TAB. 3 – Influence du pas, Valeur de la solution

Instances	Pas=1	Pas=5	Pas=10	Pas=50
onur	471667	471983	472501	478525
onur -z	448921	449231	449644	453380
aretes5	30032	30189,7	30197,3	30416,1
aretes5 -z	Memory	30586,9	30596,5	30666,9
aretes7	29007,8	29164,9	29174,2	29403,4
aretes7 -z	Memory	29463	29471,1	29550,5

On voit que lorsqu'on augmente le pas, la valeur de la solution se dégrade relativement peu.

TAB. 4 – Influence du pas, Temps de résolution

Instances	Pas=1	Pas=5	Pas=10	Pas=50
onur	62,61	28,29	21,73	19,18
onur -z	81,37	51,63	37,52	34,58
aretes5	764,01	175,77	102,08	48,99
aretes5 -z	Memory	195,46	101,94	46,54
aretes7	1040,29	265,57	161,29	89,05
aretes7 -z	Memory	333,21	184,25	113,35

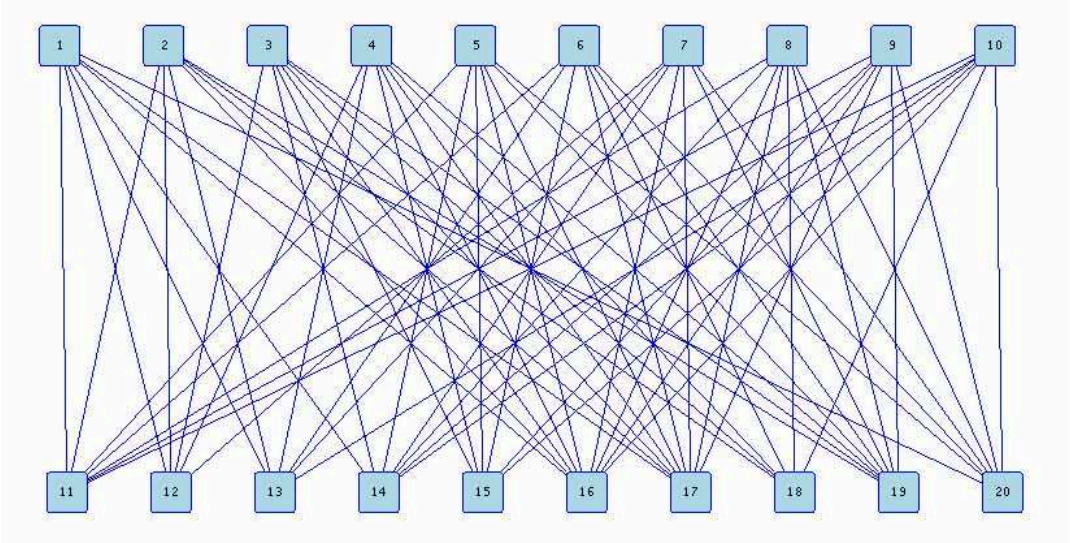


FIG. 2 – Instance *aretes*7

On voit que lorsqu'on augmente le pas, le temps d'éxecution baisse de manière importante.

7 Conclusion

Le problème de localisation et de dimensionnement est une généralisation de problème de hub-location largement étudié dans la littérature. Ici on se proposait également de tenir compte du trafic circulant au sein d'un réseau coeur formé dans une solution donnée. Ce qui nous a amené à une double prise de décision au niveau de chaque site de concentration : un dimensionnement coté accès et un dimensionnement coté coeur.

De plus, on voulait avoir une solution robuste face au non-déterminisme des données en entrée. Le modèle des contraintes probabilistes a été choisi et inseré au modèle de départ.

Ce stage a été effectué au sein de France Télécom R&D afin de répondre à un problématique industriel concret. Le modèle mis en place a été établi par une suite de questionnement auprès des responsables technico-économique pour justifier sa pertinence. Le problème déterministe est difficile à traiter en temps raisonnable à cause de son caractère très combinatoire. La volonté de tenir compte de l'aspect stochastique ne fait que rendre cette tâche plus difficile.

Néanmoins, les quelques résultats obtenus dans le stage, ont permis d'avoir des équivalences entre les contraintes probabilistes, et justifier que le traitement informatique de ces contraintes pouvait se faire sans perdre de généralité.

Malheureusement, dans les expressions de l'équivalent déterministe, il existe un terme quadratique. La mise en place d'heuristique a été nécessaire même pour le problème stochastique.

Les résultats montrent que les heuristiques mises en place sont relativement efficaces. L'utilisateur peut avoir un résultat plus ou moins rapide en modifiant la valeur du pas. La résolution est donc paramètrable et lorsqu'on augmente le pas, le temps d'éxecution baisse énormément tandis que la valeur de la solution est relativement peu élevé.

Autres pistes possibles pour traiter le problème est de proposer d'autres heuristiques, notamment pour résoudre le problème de manière approchée avec le terme quadratique de l'équivalent déterministe.

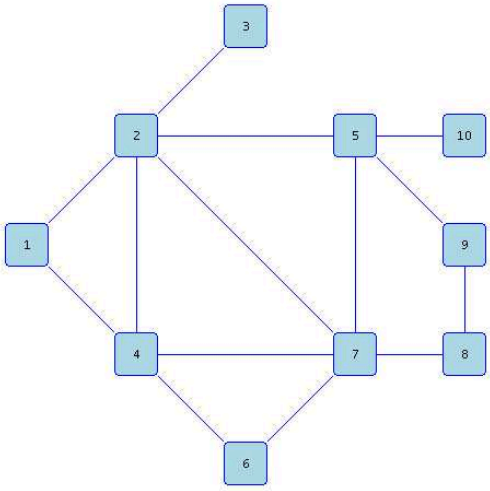


FIG. 3 – Instance *opt*10

Références

[YAMGOU] Hande Yaman, Martin Labbé, Eric Gourdin, *A Branch and Cut algorithm for Hub Location Problems with Sing Assignment*

[BIRLOU] John R. Birge, Francois Louveaux, *Introduction to Stochastic Programming*

[KALWAL] Peter Kall, Stein W. Wallace, *Stochastic Programming*

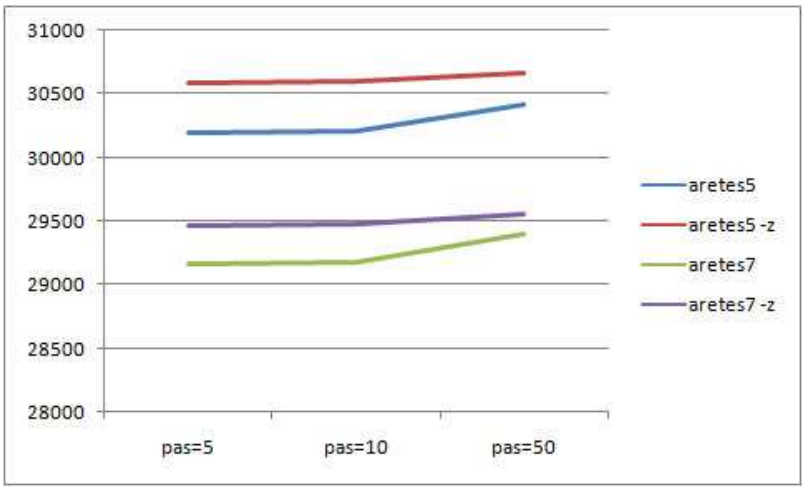


FIG. 4 – Influence du pas sur la valeur de la solution

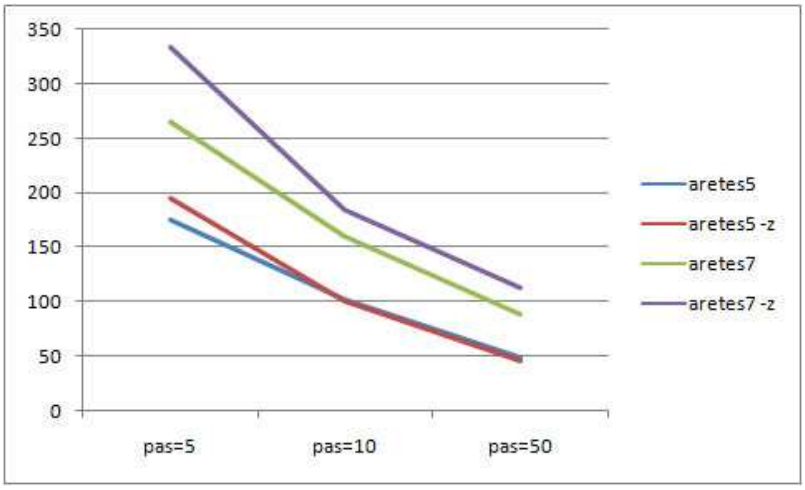


FIG. 5 – Influence du pas sur le temps d'éxecution