LENARTOWICZ Sacha
MARTIN Sébastien
TISSIER Alexandre

Projet Optimisation Stochastique

-

Document technique

Sommaire

Introduction

Comme indiqué dans l'énoncé fourni, le problème de location de vélo en libre-service (VLS), tel le service Vélib' de Paris, consiste à minimiser le coût total d'acquisition des vélos et des coûts supplémentaires. Les coûts supplémentaires pris en compte ici sont le manque à gagner dû à un manque de vélo dans la station et le manque à gagner causé par la perte de temps liée à l'attente d'une place libre pour déposer son vélo. Vous retrouverez les différentes variables utilisées dans le sujet.

I - Description du problème mathématique

A – Présentation du projet

Les données du problème

- $\mathbf{B} = \{1, \dots, B\}$ l'ensemble des stations de vélos (stations vélib par exemple à Paris).
- $\mathbf{S} = \{1, \dots, S\}$ l'ensemble de scénarios de demande de vélos, qui est une variable aléatoire.

Paramètres déterministes

- $c_i \in \mathbf{R}^+$: coût d'acquisition d'un vélo à la station i, $i \in \mathbf{B}$.
- $v_i \in \mathbf{R}^+$: coût lié à un manque de vélo dans la station i, $i \in \mathbf{B}$.
- $w_i \in \mathbf{R}^+$: coût lié au temps perdu par un utilisateur qui ne trouve pas de place à la station i, $i \in \mathbf{B}$ pour rendre son vélo.
- $k_i \in \mathbf{Z}^+$: capacité en nombre de vélos de la station i, $i \in \mathbf{B}$.

Paramètres stochastiques

Le paramètre stochastique est le paramètre qui introduit la notion d'aléa au problème.

- $\xi_{ijs} \in \mathbf{Z}^+$: demande stochastique de vélos pour aller de la station i à la station j tel que $i, j \in \mathbf{B}, s \in \mathbf{S}$. Pour ce paramètre et pour les suivants portant sur deux stations, on considère, pour simplifier le modèle, que les deux stations i et j doivent être différentes.

Variables de premier niveau

- $x_i \in \mathbf{Z}^+$: nombre de vélos à affecter à la station $i \in \mathbf{B}$ à l'ouverture du service de location.

Variables de deuxième niveau

- $\beta_{ijs} \in \mathbf{Z}^+$: nombre de vélos loués pour aller de la station i à la station j pour le scénario s tel que $i, j \in \mathbf{B}$, s $\in \mathbf{S}$.
- $I_{is}^+ = \left(x_i - \sum_{j=1}^{B} \beta_{ijs}\right)^+$: surplus de vélos dans la station i pour le scénario s. Cette valeur est positive dans le cas où il y a déficit, ou nulle dans le cas où la station n'est pas en situation de surplus de vélos.
- I_{ijs}^- : déficit de vélos dans la station i pour le scénario s pour chaque parcours entre la station i et la station j. Cette valeur est également positive ou nulle. Elle représente le déficit (le manque) de vélo pour un certain trajet entre deux stations i et j d'un scénario s.

Objectif

Il faut minimiser le coût total composé de la fonction du premier niveau et celle du problème de recours. On retrouve la fonction suivante :

$$min \sum_{i=1}^{B} c_i x_i + \sum_{s=1}^{S} p_s \sum_{i=1}^{B} \left\{ \sum_{j=1}^{B} v_i I_{ijs}^- + w_i O_{is}^- \right\}$$

Cette fonction représente le coût total qui se décompose en trois parties distinctes :

- le coût d'acquisition des vélos pour l'ensemble des stations, en rouge.
- le manque à gagner engendré par un manque de vélo dans l'ensemble des stations, en vert.
- le manque à gagner engendré par le manque de temps dû au temps perdu par un utilisateur pour trouver une borne de libre (station pleine). Dans ce cas, l'utilisateur attendra qu'une borne se libère pour pouvoir poser son vélo. Ce manque à gagner correspond à la partie en bleu. La nouvelle variable O_{is}^- est décrite dans la partie suivante.

L'objectif est donc de minimiser le coût et le manque à gagner lié à l'exploitation des stations disponibles. Pour ce faire, il nous faudra mettre en place plusieurs algorithmes pour résoudre le problème de VLS déterministe et VLS stochastique (décris plus loin).

B – Modèle mathématique

Enoncé du problème

Le problème de location de vélos en libre-service stochastique peut être formulé à l'aide du programme linéaire en variables binaires suivant :

$$min \sum_{i=1}^{B} c_i x_i + \sum_{s=1}^{S} p_s \sum_{i=1}^{B} \left\{ \sum_{j=1}^{B} v_i I_{ijs}^{-} + w_i O_{is}^{-} \right\}$$

s.t.

$$x_i \leq k_i, \forall i \in \mathbf{B} \quad (1a)$$

$$\beta_{ijs} = \xi_{ijs} - I_{ijs}^{-}, \forall i, j \in \mathbf{B}, \forall s \in \mathbf{S} \quad (1b)$$

$$I_{is}^{+} - \sum_{j=1}^{B} I_{ijs}^{-} = x_i - \sum_{j=1}^{B} \xi_{ijs}, \forall i \in \mathbf{B}, \forall s \in \mathbf{S} \quad (1c)$$

$$O_{is}^{+} - O_{is}^{-} = k_i - x_i + \sum_{j=1}^{B} \beta_{ijs} - \sum_{j=1}^{B} \beta_{jis}, \forall i \in \mathbf{B}, \forall s \in \mathbf{S} \quad (1d)$$

$$x_i, \beta_{ijs}, I_{is}^{+}, I_{ijs}^{-}, O_{is}^{+}, O_{is}^{-} \in \mathbf{Z}^{+}, \forall i, j \in \mathbf{B}, \forall s \in \mathbf{S} \quad (1e)$$

Nous voyons ici que deux nouvelles variables, O_{is}^{+} et O_{is}^{-} sont utilisés. Voici ce à quoi elles correspondent :

- O_{is}^{+} : capacité résiduelle de la station i dans le scénario s. La formule permettant de calculer cette capacité est la suivante :

$$O_{is}^{+} = \left(k_i - x_i + \sum_j \beta_{ijs} - \sum_j \beta_{jis}\right)^{+}$$

- O_{is}^{-} : surplus de la station i pour le scénario s.

$$O_{is}^{-} = \left(-k_i + x_i - \sum_j \beta_{ijs} + \sum_j \beta_{jis}\right)^{+}$$

Ces deux valeurs sont toujours supérieures ou égales à 0. Si O_{is}^{+} est strictement supérieur à 0, alors le nombre de place(s) restante(s) est supérieur à 0, donc le surplus en vélos est nul. Si au contraire, O_{is}^{-} est supérieur à 0, il ne reste plus de place dans la station ($O_{is}^{+} = 0$). Les deux valeurs sont nulles si le nombre de vélos correspond au nombre de places dans la station. En faire la somme (ici une soustraction) nous permet d'éliminer le fait d'avoir des variables strictement positives, et nous ajoute une contrainte essentielle pour compléter le modèle mathématique.

Explication des contraintes

(1a) Désigne le fait que chaque station a un nombre maximum de vélo stockable qu'elle ne peut dépasser : sa capacité k_i.

(1b) Le nombre de vélos loués pour aller de la station i à la station j correspond à la différence entre la demande stochastique de vélo pour aller de i à j moins le nombre de vélo nécessaire dans la station i pour compléter la demande vers la station j.

(1c) Assure l'équilibre entre la demande stochastique et la pénurie. En effet, pour chaque station, deux ensembles doivent s'égaler : le nombre de vélos à affecter soustrait de l'ensemble des demandes vers les autres stations et le surplus de vélos après demande soustrait de l'ensemble des déficits de vélos en destination de n'importe quelle autre station.

Afin de mieux visualiser, nous pouvons illustrer ceci par un exemple :

- nombre de vélos affecté à la station A : 8.
- demande vers les autres stations : {station B : 6, station C : 3}.
- surplus de vélos : 0 (car 8-(6+3) = -1 < 0).
- déficit de vélos : {station B : 1, station C : 0} (on aurait pu inverser ces valeurs).

Résultat : 0 - (1+0) = 8 - (6+3) = -1.

(1d) Assure l'équilibre entre la capacité résiduelle et le surplus qui entraîne alors un coût. Là encore, nous avons une égalité pour chaque station entre :

- la capacité en vélos, - les vélos attribués, + la somme des vélos sortants vers les autres stations et - la somme des arrivées de vélos pour des trajets depuis une autre station. Ces demandes peuvent engendrer un manque de place.
- la capacité résiduelle en vélo ou bien le nombre de vélos en trop. Si cette somme est négative, sa valeur absolue donne le nombre de places restantes. Si elle est négative, elle représente le nombre de vélos en trop (surplus).

Exemple (assez de place dans la station):

- capacité en vélos et vélos attribués pour la station A : $k_A = 12$ et $x_A = 10$.
- somme des demandes entrantes : {de station B : 3, de station C : 3}.
- somme de vélos sortants : {vers station B : 3, vers station C : 5}.
- O_A^+ : capacité résiduelle.
- $O_A^+ - O_A^-$ = 12 - 10 + (3+5) - (3+3) = 4, O_A^+ = 4 donc il y a une capacité résiduelle de 4 (4 places sont encore disponibles à la fin).

Exemple (pas assez de place, trop d'arrivées):

- capacité en vélos et vélos attribués pour la station A : $k_A = 12$ et $x_A = 10$.
- somme des demandes entrantes : {de station B : 8, de station C : 6}.
- somme de vélos sortants : {vers station B : 3, vers station C : 5}.
- O_A^- : surplus de vélos.

- $O_A^+ - O_A^-$ = 12 - 10 + (3+5) - (8+6) = -4, O_A^- = 4 donc il y a un manque de 4 places dans la station (4 personnes ont attendu une place libre).

Le coût total peut se décrire comme étant le coût d'achat de l'ensemble des vélos (pour toutes les stations), en plus du manque à gagner pour chaque station engendré par le manque et le surplus de vélos pour toutes les stations et scénarios (à chaque scénario étant associé une probabilité de réalisation).

(1e) Les indices i et j correspondent aux indices de 2 stations différentes [1,B] et selon un scénario s parmi d'autre[1,S]. Un scénario se définissant par un exemple de flux des vélos et des paramètres initialisés tels que le nombre de vélos par station, la capacité des stations, la demande stochastique et les autres coûts.

II – Un exemple concret

Afin d'illustrer la valeurs des différentes variables décrites précédemment, nous allons illustrer le problème à l'aide d'un système composé de 3 stations, en indiquant la valeur de chacune des valeurs et respectant toutes les contraintes présentées.

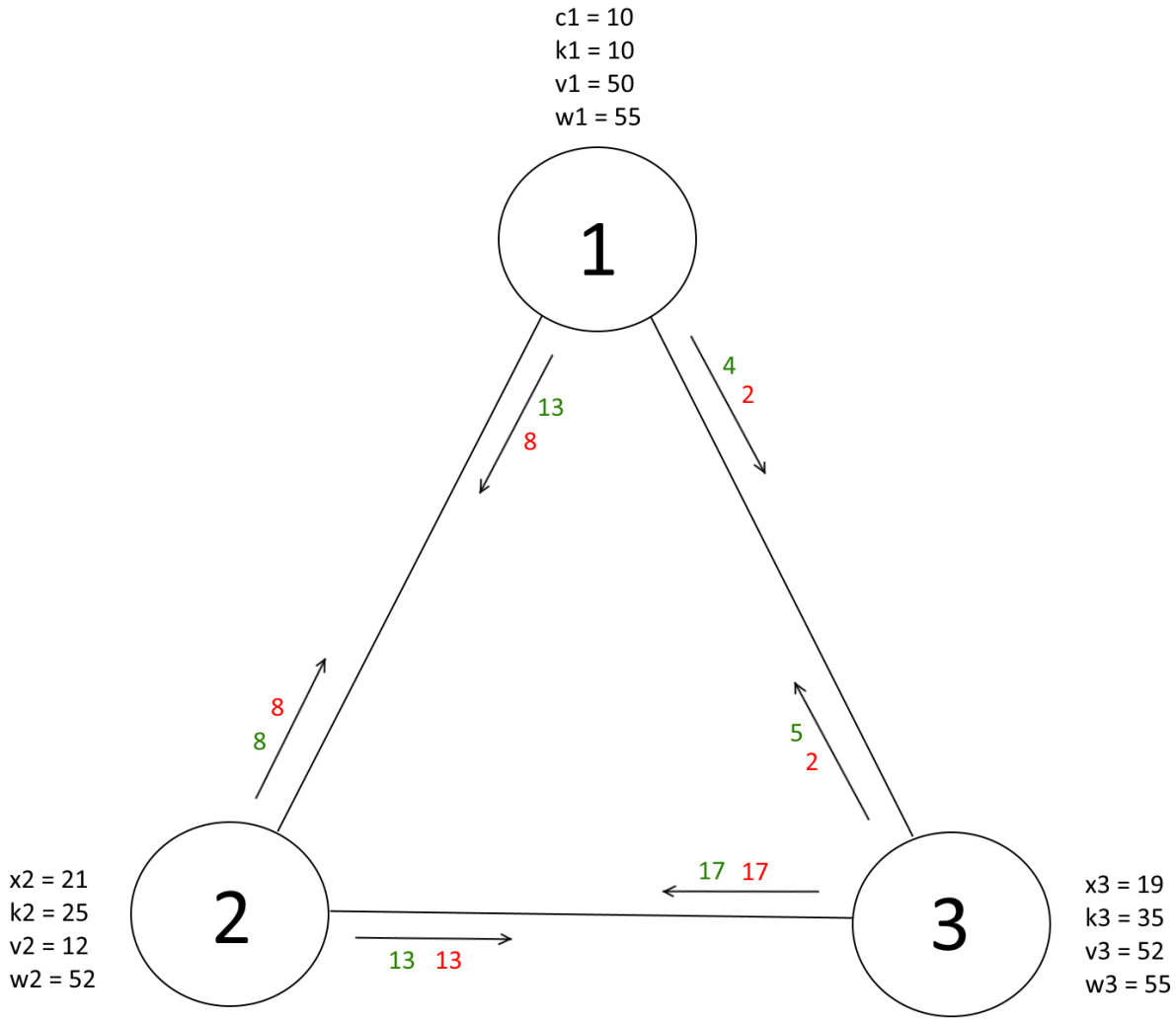

Figure 1 : schéma représentant un exemple du modèle VLS

Les valeurs indiquées en vert correspondent aux ξ_{ij} et celles en rouge aux β_{ij}. Toutes les autres variables O et I valent 0 dans cet exemple, excepté 4 variables :

- $I_{12}^{-} = 5$, $I_{13}^{-} = 2$ et $I_{31}^{-} = 3$: demande pas entièrement satisfaite car manque de vélo)
- $O_3^{+} = 20$: il reste 20 places dans la station 3 à la fin. Les autres stations sont parfaitement complètes.

La fonction objectif a pour valeur 1238€.

III - Algorithmes utilisés

A - Résolution du problème VLS avec CPLEX

L'outil Cplex, commercialisé par IBM, est principalement un solveur de programmes linéaires (PL) et programmes linéaires en nombres entiers (PLNE). Dans notre cas, il s'agit d'un PLNE. Il applique pour cela l'algorithme primal du simplexe (qui lui a donné son nom), l'algorithme dual du simplexe et l'algorithme de points intérieurs. Il est possible d'utiliser Cplex via un mode interactif (à l'aide d'un terminal), et via des API pour C++ et Java. Nous utiliserons donc l'API C++ (librairie nommée Concert Technologie et développée par IBM).

Le solveur Cplex, une fois le modèle correctement implémenté, se charge lui-même de son optimisation. La seule difficulté réside donc en la bonne intégration de tous les éléments qui compose le modèle à implémenter (pour nous, le VLS). Il est possible d'effectuer cette optimisation en demandant à Cplex, via le mode interactif, de lire un fichier au format lp dans lequel figure les différents paramètres (fonction objectif, contraintes, types des variables).

```
minimize 10 x1 + 12 x2 + 20 x3 + 50 I12_m + 50 I13_m + 58 I21_m + 58 I23_m + 52 I31_m + 52 I32_m + 55 O1_m + 52 O2_m + 55 O3_m

subject to

    k1 = 10
    k2 = 25
    k3 = 35

    x1 - k1 <= 0
    x2 - k2 <= 0
    x3 - k3 <= 0

    Ksi12 = 13
    Ksi13 = 4
    Ksi21 = 8
    Ksi23 = 13
    Ksi31 = 5
    Ksi32 = 17

    Beta12 + I12_m - Ksi12 = 0
    Beta13 + I13_m - Ksi13 = 0
    Beta21 + I21_m - Ksi21 = 0
    Beta23 + I23_m - Ksi23 = 0
    Beta31 + I31_m - Ksi31 = 0
    Beta32 + I32_m - Ksi32 = 0

    I1_p - I12_m - I13_m - x1 + Ksi12 + Ksi13 = 0
    I2_p - I21_m - I23_m - x2 + Ksi21 + Ksi23 = 0
    I3_p - I31_m - I32_m - x3 + Ksi31 + Ksi32 = 0

    O1_p - O1_m - Beta12 - Beta13 + Beta21 + Beta31 + x1 - k1 = 0
    O2_p - O2_m - Beta21 - Beta23 + Beta12 + Beta32 + x2 - k2 = 0
    O3_p - O3_m - Beta31 - Beta32 + Beta13 + Beta23 + x3 - k3 = 0

generals

    I1_p I2_p I3_p I12_m I13_m I21_m I23_m I31_m I32_m
    O1_p O2_p O3_p O1_m O2_m O3_m
    x1 x2 x3 k1 k2 k3
    Beta12 Beta13 Beta21 Beta23 Beta31 Beta32
    Ksi12 Ksi13 Ksi21 Ksi23 Ksi31 Ksi32

end
```

Figure 2 : exemple de modèle Cplex

Ceci est un exemple de fichier .js contenant un modèle Cplex. Remarque : il n'est pas possible d'utiliser des variables pour c_i, v_i et w_i et de leur affecter une valeur, celles-ci

figurant dans la fonction objectif. Les valeurs entrées dans ce modèle correspondent aux valeurs utilisées dans l'exemple précédemment traité. La valeur de la solution objectif et des autres variables sont les mêmes que ceux indiqués dans ce même exemple.

B - Résolution du problème VLS par l'algorithme du recuit simulé

Description du recuit déterministe

La solution initiale au problème sera définit par nous et devra respecter les conditions du problème mathématique. Pour simplifier, nous prendrons une solution initiale nulle.

Nous aurons un tableau avec les valeurs de température et la première condition sera d'atteindre le pallier de température à laquelle on souhaite arrêter l'algorithme.

La deuxième condition dépends du nombre d'itérations que l'on veut mettre, il sera choisi en fonction de la puissance de nos ordinateurs pour avoir un nombre d'itération assez grand mais aussi du temps que le programme mettra à répondre. Il faudra prendre en compte le fait que pour un seul scénario déterministe nous aurons à faire tourner le programme qu'une seule fois tandis pour que les scénarios stochastiques nous devrons utiliser l'algorithme sur tous les scénarios ce qui prendra plus du temps.

Pour chaque itération de la boucle principale, on commence par choisir X' qui est un voisinage à la solution précédente notée X. Lors de la première itération, la solution précédente est la solution initiale. Nous aurons une fonction de voisinage pour choisir X' qui de façon aléatoire définira la nouvelle solution (expliquée plus tard).

Il faudra faire attention, lors du calcul le voisinage, à ce que le nombre de vélos d'une station ne puisse pas descendre en dessous de 0 mais aussi que les différentes contraintes imposées par le modèle soient toutes respectées. Tant que le voisinage ne répond pas aux conditions, nous devons recalculer un nouveau voisinage.

Une fois un voisinage correcte déterminé, on calcule la différence entre le résultat de la fonction objective pour notre solution X' avec le résultat de la fonction objective de notre solution X.

Si la différence est inférieur à 0 alors on met le voisinage X' dans la solution courante X.

Si le résultat de la fonction objective de la nouvelle solution X est inférieur au résultat de la fonction objective minimale alors on remplace la solution minimale par X et le résultat de la fonction objective minimale par le nouveau résultat de la fonction objective.

Si en revanche la différence du résultat des fonctions objectif pour X' et X n'est pas inférieur à 0, on tire une probabilité p dans [0,1]. Si cette probabilité est inférieur ou égale à la

probabilité de Boltzman, alors on remplace la solution X par la solution voisine X'. La probabilité de Boltzman se calcule ainsi : $\exp(\frac{-\ différence\ des\ fonctions\ objectif}{Température})$.

Lorsque les itérations de la boucle interne sont finies, on fait décroître la température T. L'algorithme peut se schématiser ainsi :

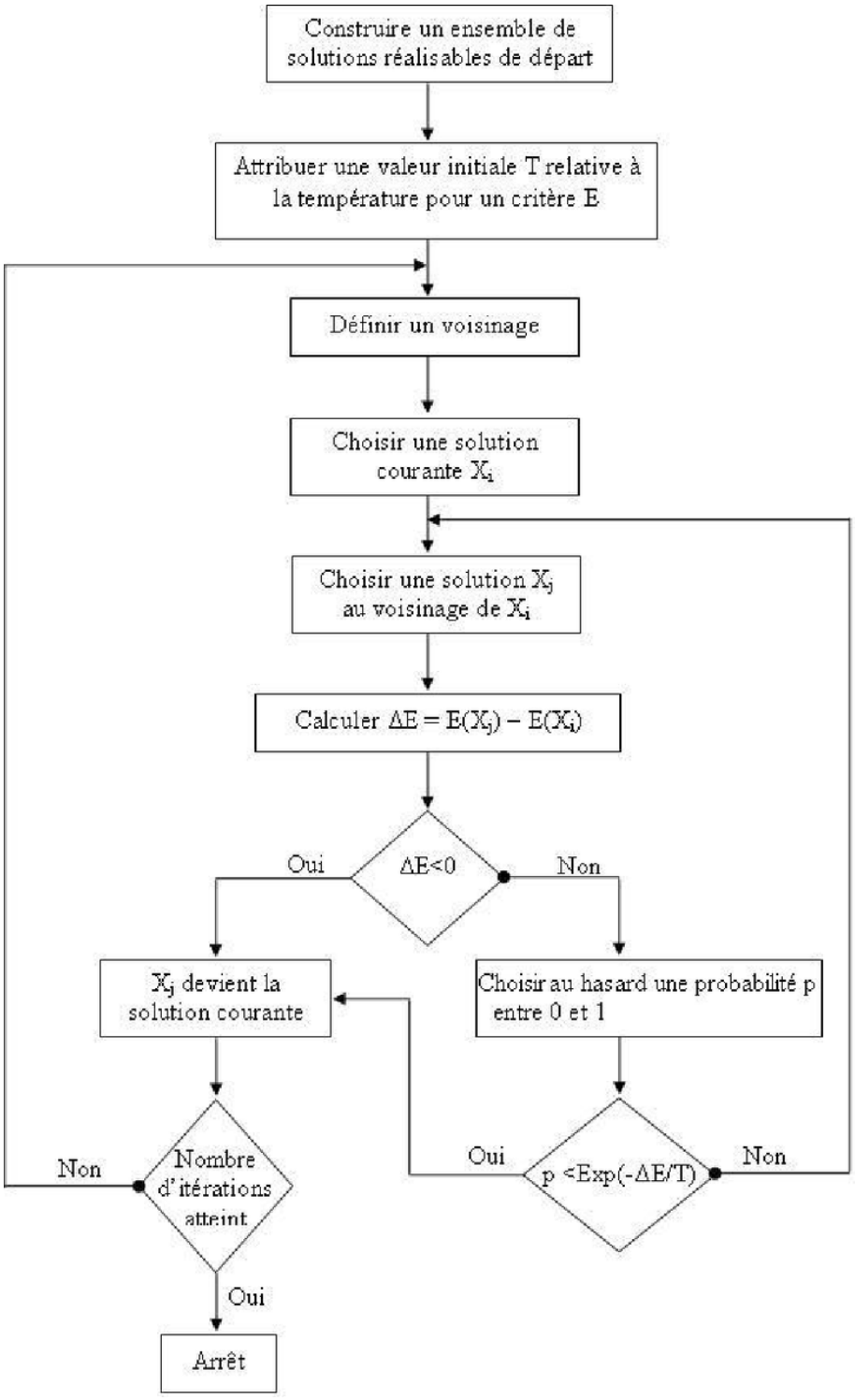

Figure 3 : représentation schématique du déroulement de l'algorithme du recuit simulé (source : www.researchgate.net)

Voisinage

Pour le voisinage à appliquer par notre algorithme du recuit simulé, nous avons décidé de procéder en deux étapes.

La première étape consiste à choisir équitablement entre 3 actions à réaliser qui sont :

- enlever 1 vélo à une station.
- ajouter 1 vélo à une station.
- transférer un vélo d'une station à une autre.

La deuxième étape consiste à choisir la ou les stations concernées. Pour les deux premiers cas, il suffit de tirer un nombre entre 1 et le nombre de stations S. Si nous avons choisi le premier cas il faut que la station possède au moins un vélo. Sinon si nous choisissons le deuxième cas il faudra vérifier que la capacité de la station n'est pas dépassée. Pour la dernière action, il faut tirer au sort deux stations, la première donnant un vélo (si elle en possède) à la deuxième station. Si la première station tirée ne possède pas de vélo, on recommence le tirage.

Exemple : Etat initial : station 1 a 3 vélos, station 2 en a 5.
Action : 3.
Choix stations : 1 et 2.
Etat final : station 1 a 2 vélos, station 2 en a 6.

Température

Nous avons choisi de fixer une température initiale (par exemple 15) avec un pallier pour faire descendre la température de 1 à chaque itération de cette boucle principale. Pour la boucle while interne, nous fixons également un certain nombre d'itération à effectuer (par exemple 600).

Application au recuit simulé stochastique

Pour résoudre le problème du VLS, nous pouvons également appliquer l'algorithme du recuit stochastique. Pour cela, nous avons besoin de plusieurs scénarios où les différents coûts (achat de vélos, surplus de vélo par rapport à la capacité de la station, et manque de vélo par rapport à la demande) et la capacité de la station seront identiques mais où la demande stochastique variera d'un scénario à l'autre autour d'une moyenne prédéfini.

La demande stochastique qui se définit le nombre de vélos demandés pour chaque trajet entre 2 stations i et j tel que i, j $\in B$ est en effet le paramètre stochastique. L'algorithme du recuit nous offre une possibilité d'améliorer une solution initiale respectant les contraintes du problème.

Le recuit stochastique se base sur le recuit déterministe. Il ne s'agit pas de faire une simple moyenne entre différentes fonctions objectives mais de rajouter une pénalité pour certaines solutions, jusqu'à obtenir la même solution pour tous les scénarios.

Il s'agit donc d'ajouter cette pénalité au calcul de la fonction objectif lors des étapes du recuit simulé. La pénalité prend en compte l'écart de la solution de l'étape du recuit avec la solution moyenne calculée lors de la première phase du recuit stochastique.

Nous allons vous expliquer via un exemple la façon dont nous allons résoudre le problème du VLS stochastique par l'algorithme du recuit simulé :

Imaginons un réseau composé de 2 stations A et B. Le nombre de scénarios générés pour le recuit stochastique est de 3.

Phase 1 :

Scénarios	Solution pour A (x_A)	Solution pour B (x_B)
1	4	4
2	2	5
2	0	6

La moyenne calculée pour x_A, notée $\overline{x_A}$, vaut 2, et $\overline{x_B}$ = 5.

Phase 2 :

On répète, pour chaque scénario, un recuit simulé, prenant cette fois-ci en compte dans le calcul de la fonction objectif une pénalité, calculée en multipliant un coefficient k à la somme des écarts de solution actuelle par rapport à la moyenne précédemment calculée. Le coefficient k choisit est identique pour tous les scénarios et constant pendant tout le recuit stochastique.

Le recuit va ajouter $400 \times e$ à la fonction objectif pour chaque étape du recuit, e étant la somme des écarts à la moyenne précédente. Par exemple, pour le scénario 1, le recuit travaille à un moment donné sur la solution $x_A' = 1$ et $x_B' = 4$. Ici,

$$e = |x_A - x_A'| + |x_B - x_B'| = (2 - 1) + (5 - 4) = 2$$

Le calcul de la fonction objectif pour cette étape du recuit ajoutera ainsi une pénalité de $2 \times 400 = 800$.

Au cours des phases suivantes, l'intégration de cette pénalité permet de tendre vers un ensemble de solutions de plus en plus proche, jusqu'à obtenir une phase n pour laquelle toutes les solutions sont identiques :

Phase n :

Scénarios	Solution pour A (x_A)	Solution pour B (x_B)
1	3	4
2	3	4
2	3	4

Arrivé à cette étape, l'algorithme est terminé. Les solutions de notre exemple sont $x_A = 3$ et $x_B = 4$.

Création de scénario

Pour calculer la valeur de la fonction objective, nous allons devoir avoir la probabilité de nos scénarios. On considère que la probabilité d'un scénario est la même quel que soit ce scénario. Si S est le nombre de scénario créé, alors la probabilité de chaque scénario sera $p_s = 1/S$. Les coûts engendrés pour la station i restent inchangés d'un scénario à l'autre et seront définis soit par la base de données de notre programme, soit par l'utilisateur. Comme expliqué précédemment, les demandes en vélos allant de la station i à la station j : $\beta_{ij}\ \forall i, j \in B$ est le paramètre stochastique qui doit varier d'un scénario à l'autre.

Pour pouvoir créer un scénario il nous faut un scénario de base qui nous a été fourni par le professeur. Ce scénario est considéré comme étant le scénario moyen.

Nous allons donc calculer les demandes à partir du scénario moyen via ce calcul : $\xi_{ij} \in [\xi_{ij_{moyen}} - 20\%, \xi_{ij_{moyen}} + 20\%]$, où $\xi_{ij_{moyen}}$ est la demande "moyenne" calculée à partir de notre scénario de base.

Par exemple si dans le scénario moyen il y a une demande de 30 vélos de la station i à la station j alors pour notre nouveau scénario la demande de vélo entre la station i et j sera entre ces valeurs [24,36]. Nous choisirons donc la nouvelle demande de vélo avec un chiffre aléatoire entre nos bornes et nous ferons pareil pour les autres demandes de vélo entre 2 stations.

C - Résolution du problème VLS stochastique par l'algorithme SAA

Description de l'algorithme SAA

L'algorithme SAA est l'algorithme de Sample Average Approximation.

Cet algorithme est utilisé pour résoudre des problèmes d'optimisation stochastique. Nous utiliserons cet algorithme sur notre modèle mathématique à 2 niveaux. Le principe de cette méthode est de trouver les solutions minimales sur tous les échantillons puis d'utiliser toutes ces solutions sur notre échantillon de référence et trouver la solution optimale.

Cet algorithme est séparé en différentes parties : tout d'abord l'initialisation où l'on génère N échantillons indépendants les uns des autres avec un ensemble de scénario. Puis on génère un autre échantillon suffisamment large pour pouvoir utiliser les solutions des résultats que l'on a trouvé précédemment.

Chaque échantillon contient un certain nombre de scénario, nous avons donc besoin de créer des scénarios pour pouvoir créer des échantillons puisqu'un échantillon est simplement un ensemble de scénario.

Pour chaque échantillon il faut résoudre notre problème mathématique à 2 niveaux puis récupérer la valeur minimale de la fonction objectif et sa solution optimale. Ensuite il faut calculer la moyenne pour tous les échantillons.

La 3ème étape a pour objectif d'estimer la vraie valeur de la fonction objective du problème de départ pour chaque échantillon en utilisant la solution optimale de premier niveau de l'étape 1. La dernière étape est de choisir la solution optimale avec la meilleure fonction objective.

Application de l'algorithme SAA

L'algorithme du SAA est utilisé en lien avec le recuit simulé. Effectivement nos échantillons sont un ensemble de scénario avec des demandes de vélos entre les stations qui sont aléatoires. Nous avons donc le recuit simulé stochastique qui va nous servir lors de l'algorithme du SAA. Il nous suffira d'utiliser plusieurs fois le recuit stochastique pour avoir notre algorithme SAA. Nous allons donc générer plusieurs scénarios qui créeront des échantillons et qui nous permettront de calculer des solutions optimales.

Pour cela nous avons une fonction qui permet de générer des échantillons en fonction d'une liste de scénario mais aussi du nombre d'échantillons voulus. Nous aurons donc une liste de liste de scénario qui sera l'ensemble des échantillons et pour récupérer un échantillon

il suffira de récupérer un objet de la liste de liste de scénario, ce qui nous donnera donc une liste de scénario.

Une fois cela fait nous allons récupérer chacune des solutions pour calculer la solution moyenne sur tous les échantillons. Il ne nous restera plus qu'à calculer la valeur de la fonction objective en fonction des solutions optimales de chaque échantillon sur notre échantillon de référence. Et de trouver la valeur objective minimale, ce qui nous donnera la solution optimale au problème.

IV – Résultats

Après avoir optimisé l'algorithme du recuit et le calcul de la fonction objectif, nous nous sommes rendu compte que le programme prenait un certain temps à calculer un recuit, en fonction du nombre de stations utilisées.

Nous avons cherché à obtenir les valeurs de la température initiale et du nombre d'itérations de la boucle intérieure de façon à ce qu'une augmentation de celles-ci n'améliore que sensiblement la valeur de la fonction objectif. En effet, comme nous partons d'une solution initiale nulle, il faut un certain temps avant lequel les stations ne sont toujours pas assez remplies.

Pour 100 stations, nous arrivons à obtenir un temps relativement acceptable d'environ 4 secondes, à partir duquel il n'est plus possible d'améliorer significativement la solution obtenue. Ce temps est obtenu avec une température initiale égale à 15 et 600 itérations internes. Les résultats présentés ci-dessous ont été obtenus avec ces deux valeurs, un coût d'achat de vélo de 5, un coût lié au surplus égale à 50 et un coup lié au déficit de 40 :

Algorithme	Nombre de stations	Valeur fonction objectif	Temps d'exécution (en secondes)	Nombre de scénario(s)	Nombre d'échantillon(s)
Recuit déterministe	10	6125	0,16	1	1
	50	19220	1,36	1	1
	100	31640	4,89	1	1
	200	97990*	19,03	1	1
	980	766520*	610,20	1	1
Recuit stochastique	10	6280	2,03	10	1
	50	19385	26,14	10	1
	100	nd	>3600	10	1
SAA	10	6080	10,49	10	5
	50	19350	151,31	10	5

(*) : valeurs biaisées par le fait que l'algorithme n'a pas effectué assez d'incrémentation du nombre de vélos par station.