Université Libre de Bruxelles
Master informatique parcours “optimisation en recherche opérationnelle (ORO)”
Année académique 2018-2019

Rapport de Projet

Single Period Coverage Facility Location Problem

A.Gontier, C.Pelhâtre, M.Ocquident, M.Latif 12 août 2020

Résumé

Le projet que nous allons présenter dans ce rapport porte sur la résolution du Single Period Coverage Facility Location Problem (SPCFLP). Nous allons présenter différents algorithmes implémentés pour la résolution de ce dernier. La résolution exacte de ce problème d'affectation pouvant être coûteuse en terme de calcul, nous allons nous intéresser à des méthodes dites de relaxation permettant d'obtenir des bornes inférieures pour les solutions optimales, mais également à des bornes supérieures que nous obtiendrons à l'aide d'heuristiques de construction.
Nous finirons ce rapport en présentant une étude comparative de l'efficacité des solutions mises en place en prenant en compte les temps de calculs mais également la qualité des solutions obtenues.

Table des matières

1 Formulation du SPCFLP

Le SPCFLP est constitué de trois ensembles de variables :

I **:** un ensemble de clients indexés par $i \in I$

J **:** un ensemble de facilités indexées par $j \in J$

T **:** un ensemble de périodes indexées par $t \in T$

Nous posons les données suivantes pour notre problème :

H **:** le nombre maximal de clients i pouvant être assignés à une facilité j à la période t si cette dernière est ouverte

p^t **:** le nombre minimum de facilités à ouvrir sur la période $t \in T$

c_{ij}^t **:** le coût d'affectation de la facilité j au client i sur la période t $\forall i \in I$, $\forall j \in J$, $\forall t \in T$.

f_j^t : le coût d'ouverture de la facilité j sur la période t $\forall j \in J$, $\forall t \in T$.

Enfin, nous définissons les variables de décisions suivantes $\forall i \in I$, $\forall j \in J$, $\forall t \in T$:

$$x_{ij}^t = \begin{cases} 1 & \text{Si le client } i \text{ est servi par la facilité } j \text{ sur la période } t \\ 0 & \text{sinon.} \end{cases}$$

$$y_j^t = \begin{cases} 1 & \text{Si la facilité } j \text{ est ouverte sur la période } t \\ 0 & \text{sinon.} \end{cases}$$

Le problème se modélise de la manière suivante :

$$\min \sum_{t \in T} \sum_{j \in J} (f_j^t y_j^t + \sum_{i \in I} c_{i,j}^t x_{i,j}^t) \tag{1}$$

$$\text{s.c} \sum_{t \in T} \sum_{j \in J} x_{ij}^t = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I} x_{ij}^t \leq H y_j^t \qquad \forall j \in J, \forall t \in T \tag{3}$$

$$\sum_{j \in S} y_j^t \geq p^t \qquad \forall t \in T \tag{4}$$

$$x_{ij}^t \in \{0,1\} \forall i \in I, \forall j \in J, \forall t \in T \tag{5}$$

$$y_j^t \in \{0,1\} \forall j \in J, \forall t \in T. \tag{6}$$

Le problème consiste en l'affectation des clients sur un nombre minimum de facilités ouvertes sur chaque périodes et ceci en respectant les contraintes de capacités de chaque facilités.
La fonction objectif minimise le coût total (coûts d'affectations c_{ij}^t et coûts d'ouvertures f_t^j des facilités). Les contraintes (2) assurent d'affecter chaque client à une facilité à une période. Les contraintes (3) représentent la capacité des facilités si cette dernière est ouverte. Enfin les contraintes (4) assurent l'ouverture d'un nombre minimum de facilités sur chaque périodes.

2 Relaxation linéaire

La relaxation linéaire consiste à relâcher les contraintes d'intégrité des variables x et y qui deviennent :

$$0 \leq x_{ij}^t \leq 1 \qquad x_{ij}^t \in \mathbb{R}; \forall i \in I, \forall j \in J, \forall t \in T \tag{7}$$

$$0 \leq y_j^t \leq 1 \qquad y_j^t \in \mathbb{R}; \forall j \in J, \forall t \in T \tag{8}$$

Dans les solutions obtenues, on remarque que les variables x_{ij}^t prennent des valeurs entières. Cela s'explique par le fait que la matrice d'affectation est totalement unimodulaire (TUM) [1].

Démonstration. La preuve est laissée au lecteur à titre d'exercice □

Seules les variables y_j^t prennent des valeurs entre 0 et 1, ce qui rend la solution non admissible pour le problème initial. De ce fait la valeur de la solution optimale de la relaxation linéaire est meilleure que la solution optimale du problème initiale, C'est une borne inférieure.

Pour obtenir une première solution admissible on peut mettre à 1 les variables y_j^t fractionnaires. En effet si on augmente les y_j^t, les contraintes (3) et (4) seront toujours respectées. On obtient donc une borne supérieure.

3 Heuristique

L'heuristique choisie pour trouver une solution admissible est Reactiv GRASP (Greedy Randomized Adaptative Search Procedure). Pour expliquer notre heuristique nous allons d'abord parler de GRASP, puis de Reactiv GRASP.

3.1 GRASP : construction de solution admissible

GRASP est une heuristique qui fait un compromis entre algorithme glouton et aléatoire. C'est une heuristique de construction qui choisit une à une les variables à faire entrer dans la solution. Comme les variables y_j^t sont définies à partir des variables x_{ij}^t, on a seulement besoin de déterminer les variables x_{ij}^t dans notre solution. Ces variables sont choisies selon une fonction d'utilité définie comme suit :

$$U(x_{ij}^t) = f_j^t + c_{ij}^t$$

Cette dernière correspond au coût d'une affectation. L'algorithme va favoriser les variables les moins chères, mais avec une marge aléatoire de tolérance. L'aléatoire dans l'heuristique de construction est contrôlée par un paramètre α. Ce paramètre limite la qualité des nouvelles variables qui seront choisies pour entrer dans la solution, les obligeant à ne pas dépasser le seuil d'utilité limite :

$$Limit = U_{min} + \alpha \times (U_{max} - U_{min})$$

avec U_{min}, respectivement U_{max}, l'utilité minimum et maximum parmi l'ensemble des variables. L'algorithme consiste à choisir itérativement une variable de manière aléatoire dont l'utilité ne dépasse pas la limite. Voici le pseudo-code de l'algorithme :

1. Une matrice est totalement unimodulaire (TUM) si le déterminant de toutes ses sous-matrices carrés est à valeur dans $\{-1, 0, 1\}$

Algorithme 1 : GRASP construction

Entrées : réel α, instance du SPCFLP I
Sorties : une solution admissible

$C \leftarrow$ Liste des candidats x_{ij}^t
$solution \leftarrow \{\}$
while *nonVide(C)* **do**
  $Limit \leftarrow U_{min} + \alpha \times (U_{max} - U_{min})$
  $RCL \leftarrow \{x_{ij}^t \in C, U(x_{ij}^t) \leq Limit\}$
  $e \leftarrow$ choix d'un candidat aléatoire parmis la RCL
  $solution \leftarrow solution \cup e$
  $C \leftarrow$ mise à jour de l'ensemble des candidats suivant l'instance I
end
retourner *solution*

Pour s'assurer d'obtenir une solution admissible, il faut bien mettre à jour l'ensemble des candidats à chaque itération, de manière à respecter toutes les contraintes. En effet, lors de l'ajout d'un candidat à la solution, on doit en enlever certains parmi l'ensemble des candidats. Soit x_{ij}^t la variable ajoutée à la solution, voici les étapes à suivre pour mettre à jour C (non détaillées dans le pseudo-code par simplicité) :

-Pour respecter les contraintes (2), on enlève toutes les variables $x_{i'j'}^{t'}$ tel que $i' = i, \forall i' \in I, \forall j' \in J, \forall t' \in T$

-Pour respecter les contraintes (3), on garde en mémoire le nombre de clients affectés à une facilité j à la période t dans une matrice de taille $[T\text{x}J]$. Si lors de l'ajout d'une variable dans la solution ce nombre atteint H, alors on enlève les candidats qui ont le même indice t et j.

-Tant que les contraintes (4) ne sont pas respectées, on interdit les candidats qui utilisent une facilité déjà ouverte. Pour que toutes les périodes soient respectées, on mémorise le nombre de facilités ouvertes à la période t dans un vecteur. Lorsqu'une période atteint son quota minimum de facilités à ouvrir, on met tous les candidats de cette période en attente [2]. Une fois que C est vide, on recommence l'algorithme GRASP avec le vecteur d'attente comme liste de candidats.

3.2 Recherche locale

Une fois que l'on a obtenu une solution admissible avec GRASP, on améliore la solution avec une recherche locale. Le principe est le suivant : Entre les différents clients i de la solution, on essaye d'intervertir les t et les j pour voir si le coût est diminué.

Prenons par exemple la variable x_{ij}^t et la variable $x_{i'j'}^{t'}$ qui font partie de la solution admissible. On peut calculer le coût des deux variables en sommant leur utilité : $U(x_{ij}^t) + U(x_{i'j'}^{t'})$. On regarde si $x_{ij'}^{t'}$ et $x_{i'j}^t$ coûtent moins chère : $U(x_{ij'}^{t'}) + U(x_{i'j}^t)$. Si c'est le cas, les variables deviennent $x_{ij'}^{t'}$ et $x_{i'j}^t$. On effectue cette comparaison entre tous les clients i.

Plus précisément, on peut définir un voisinage pour chaque variable x_{ij}^t appartenant à la solution initiale :
$voisinage(x_{ij}^t) = \{x_{ij'}^{t'}, \exists i', x_{i'j'}^{t'} = 1, \forall t' \in T \setminus \{t\}, \forall j' \in J \setminus \{j\}\}$
Ainsi un voisin est dit améliorant si $U(x_{ij'}^{t'}) + U(x_{i'j}^t) < U(x_{ij}^t) + U(x_{i'j'}^{t'})$. Notre recherche locale consiste alors à trouver un voisin améliorant pour chaque variable x_{ij}^t de notre solution initiale.

Le fait de ne pas modifier les valeurs de i nous assure de respecter les contraintes (2). Par ailleurs, les couples (t, j) ne sont pas modifiés (on les affecte juste à un autre client), ce qui nous assure de ne pas violer les contraintes (3) et (4).

2. on les enlève de C mais on les garde en mémoire dans un vecteur d'attente. Ainsi, les mises à jour évoquées précédemment se font aussi pour le vecteur d'attente

3.3 Reactiv GRASP : ajustement du paramètre α par apprentissage

Dans le cas d'un Grasp, le choix du paramètre α est complètement arbitraire. Pour le choisir plus intelligemment, l'objectif est d'essayer de le régler sur les données en introduisant une phase d'apprentissage durant laquelle on choisira un α qui correspond le mieux possible aux données. Pour ce faire, on choisit de discrétiser les valeurs possibles de α et de leur attribuer une probabilité égale (loi uniforme).

Après avoir laissé l'algorithme trouver quelques solutions, la probabilité de chaque α est recalculée en fonction de la qualité des solutions qu'ils ont engendrés.

On recommence ce cycle de modifications de probabilités puis de construction et recherche locale de solutions jusqu'à un temps/nombre d'itérations limite.

Algorithme 2 : Reactiv GRASP

Entrées : entier $n\alpha$, tableau réel $V\alpha$, instance du SPCFLP I,temp limite
Sorties : une solution admissible

$C \leftarrow$ Liste des candidats x_{ij}^t
$solution \leftarrow \{\}$
$p \leftarrow$ poids des α initialisés selon une loi uniforme
while *temp limite* **do**
  for $i \leftarrow 1$ **to** $n\alpha$ **do**
    $\alpha \leftarrow$ choix parmis le tableau V_α selon les poids p
    $sol_{glouton} \leftarrow$ GRASP(α, I)
    $sol_{locale} \leftarrow$ rechercheLocale($sol_{glouton}$)
    $z \leftarrow$ évaluation(sol_{locale})
    $sol_{best}, z_{best}, z_{worst} \leftarrow$ mise à jour de la meilleur sol et des bornes
  end
  $p \leftarrow$ mise à jour des poids p selon la qualité des solutions optenues
end
retourner *solution*

Ainsi, le paramètre du Grasp apprend et correspond de mieux en mieux aux données sur lesquelles il est appliqué. Grâce à cet apprentissage, le Reactiv GRASP est une heuristique globalement bonne sur toutes les instances.

Nous avons aussi eu l'idée d'une petite heuristique simple pour faire des comparaisons avec les autres. Cette heuristique consiste à remettre à 1 les variables y relâches lors de la relaxation linéaire. Cette action est possible car les contraintes (3) et (4) restent satisfaites si les variables y augmentent. On obtient alors une borne Sup.

4 Relaxation Lagrangienne

Nous allons à présent présenter les différentes relaxations Lagrangiennes développées pour la résolution du SPCFLP. Nous avons décidé d'appliquer cette relaxation sur deux contraintes pour obtenir deux problèmes relâchés que nous allons décrire dans la section ci-dessous :

La contrainte n° 2 : Affectation des clients aux facilités

La contrainte n° 3 : Capacité maximale d'une facilité si cette dernière est ouverte/

4.1 Relaxation de la contrainte d'affectation des clients

Nous allons présenter les détails des calculs permettant d'obtenir une relaxation lagrangienne pour le problème SPCFLP en relâchant la contrainte d'affectation des clients. La contrainte concernée s'écrit de la manière suivante :

$$\sum_{t\in T}\sum_{j\in J} x_{ij}^t = 1 \ \forall i \in I$$

Le choix de cette contrainte est motivée par le fait que nous allons pouvoir obtenir un ensemble de facilités ouvertes $\{y_j^t\}$ admissibles ; Dès lors, nous pouvons sélectionner des valeurs de $\{x_{ij}^t\}$ basées sur les facilités ouvertes pour chaque période.
Nous allons démontrer la transformation réalisée afin d'obtenir cette relaxation. Réécrivons la contrainte choisie :

$$\sum_{t\in T}\sum_{j\in J} x_{ij}^t = 1 \ \forall i \in I \Longleftrightarrow 1 - \sum_{t\in T}\sum_{j\in J} x_{ij}^t = 0 \ \forall i \in I$$

Nous pouvons voir que le quantificateur universel ne porte que sur les clients, on peut donc en déduire que le coefficient de Lagrange $\lambda = (\lambda_i)_{i\in|I|}$ sera un vecteur de dimension $(1 \times |I|)$.

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J}\left(f_j^t y_j^t + \sum_{i\in I} c_{i,j}^t x_{i,j}^t\right) + \lambda\left(1 - \sum_{t\in T}\sum_{j\in J} x_{ij}^t\right) \tag{9}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J}\left(f_j^t y_j^t + \sum_{i\in I} c_{i,j}^t x_{i,j}^t\right) + \sum_{i\in I}\lambda_i\left(1 - \sum_{t\in T}\sum_{j\in J} x_{ij}^t\right) \tag{10}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J} f_j^t y_j^t + \sum_{t\in T}\sum_{j\in J}\sum_{i\in I} c_{i,j}^t x_{i,j}^t + \sum_{i\in I}\lambda_i - \sum_{t\in T}\sum_{j\in J}\sum_{i\in I}\lambda_i x_{i,j}^t \tag{11}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J} f_j^t y_j^t + \sum_{t\in T}\sum_{j\in J}\sum_{i\in I} c_{i,j}^t x_{i,j}^t - \lambda_i x_{i,j}^t + \sum_{i\in I}\lambda_i \tag{12}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J} f_j^t y_j^t + \sum_{t\in T}\sum_{j\in J}\sum_{i\in I} c_{i,j}^t x_{i,j}^t - \lambda_i x_{i,j}^t + \sum_{i\in I}\lambda_i \tag{13}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J} f_j^t y_j^t + \sum_{t\in T}\sum_{j\in J}\sum_{i\in I}\left(c_{i,j}^t - \lambda_i\right) x_{i,j}^t + \sum_{i\in I}\lambda_i \tag{14}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J}\left(f_j^t y_j^t + \sum_{i\in I}\left(c_{i,j}^t - \lambda_i\right) x_{i,j}^t\right) + \sum_{i\in I}\lambda_i \tag{15}$$

Le problème du SPCFLP relâché devient donc :

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t\in T}\sum_{j\in J}\left(f_j^t y_j^t + \sum_{i\in I}\left(c_{i,j}^t - \lambda_i\right) x_{i,j}^t\right) + \sum_{i\in I}\lambda_i \tag{16}$$

$$\text{s.c} \ \sum_{i\in I} x_{ij}^t \leq H y_j^t \qquad \forall j \in J, \forall t \in T \tag{17}$$

$$\sum_{j\in S} y_j^t \geq p^t \qquad \forall t \in T \tag{18}$$

$$x_{ij}^t \in [0,1] \ \forall i \in I, \forall j \in J, \forall t \in T \tag{19}$$

$$y_j^t \in \{0,1\} \ \forall j \in J, \forall t \in T. \tag{20}$$

4.2 Relaxation de la contrainte de capacité maximale des facilités

Nous allons présenter les détails des calculs permettant d'obtenir une relaxation lagrangienne pour le problème SPCFLP en relâchant la contrainte de capacité maximale des facilités si ces dernières sont ouvertes. La contrainte concernée s'écrit de la manière suivante :

$$\sum_{i \in I} x_{ij}^t \leq H y_j^t \ \forall j \in J \ , \forall t \in T$$

Le choix de cette contrainte est motivée par le fait qu'il nous sera possible d'effectuer une séparation du SPCFLP en deux sous problèmes, un pour chaque variable de décision, que nous pourrions résoudre indépendamment. La résolution peut également être réalisée en combinant dans un même modèle les deux sous problèmes, comme nous l'avons fait dans l'implémentation.

Proposition 1. *Propriété des matrices TUM*
Pour le problème d'affectation, si la matrice de coefficients est TUM, alors la solution au problème relâché sera entière.

Cette propriété nous permet d'effectuer la relaxation de la contrainte d'intégrité de la variable $x_{ij}^t \ \forall i \in I \ \forall j \in J \ \forall t \in T$.
Nous allons démontrer la transformation réalisée afin d'obtenir cette relaxation. Réécrivons la contrainte choisie :

$$\sum_{i \in I} x_{ij}^t \leq H y_j^t \ \forall j \in J \ , \forall t \in T \Longleftrightarrow \sum_{i \in I} x_{ij}^t - H y_j^t \leq 0 \ \forall j \in J \ \forall t \in T$$

Nous pouvons voir que les quantificateurs universels ne portent sur les facilités ainsi que les périodes, on peut donc en déduire que le coefficient de Lagrange $\lambda = \left(\lambda_j^t\right)_{j \in |J|, t \in |T|}$ sera un vecteur de dimension $(|J| \times |T|)$.

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t y_j^t + \sum_{i \in I} c_{i,j}^t x_{i,j}^t \right) + \lambda \left(\sum_{i \in I} x_{ij}^t - H y_j^t \right) \tag{21}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t y_j^t + \sum_{i \in I} c_{i,j}^t x_{i,j}^t \right) + \sum_{t \in T} \sum_{j \in J} \lambda_j^t \left(\sum_{i \in I} x_{ij}^t - H y_j^t \right) \tag{22}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} f_j^t y_j^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} c_{i,j}^t x_{i,j}^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} \lambda_j^t x_{ij}^t - \sum_{t \in T} \sum_{j \in J} \lambda_j^t H y_j^t \tag{23}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} f_j^t y_j^t - \lambda_j^t H y_j^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} c_{i,j}^t x_{i,j}^t + \lambda_j^t x_{ij}^t \tag{24}$$

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t - \lambda_j^t H \right) y_j^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} \left(c_{i,j}^t + \lambda_j^t \right) x_{i,j}^t \tag{25}$$

Le problème du SPCFLP relâché devient donc :

$$z_{\mathcal{L}}(\lambda) = \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t - \lambda_j^t H \right) y_j^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} \left(c_{i,j}^t + \lambda_j^t \right) x_{i,j}^t \tag{26}$$

$$\text{s.c} \sum_{t \in T} \sum_{j \in J} x_{ij}^t = 1 \qquad \forall i \in I \tag{27}$$

$$\sum_{j \in S} y_j^t \geq p^t \qquad \forall t \in T \tag{28}$$

$$x_{ij}^t \in [0,1] \forall i \in I, \forall j \in J, \forall t \in T \tag{29}$$

$$y_j^t \in \{0,1\} \forall j \in J, \forall t \in T. \tag{30}$$

Comme nous l'avons évoqué précédemment, il est possible de diviser le SPCFLP en deux sous problèmes de la manière suivante :

$$z_{\mathcal{L}}(\lambda)^1 = \min \sum_{t\in T}\sum_{j\in J}\sum_{i\in I}\left(c_{i,j}^t + \lambda_j^t\right) x_{i,j}^t \tag{31}$$

$$\text{s.c} \sum_{t\in T}\sum_{j\in J} x_{ij}^t = 1 \qquad \forall i \in I \tag{32}$$

$$x_{ij}^t \in [0,1] \forall i \in I, \forall j \in J, \forall t \in T \tag{33}$$

$$z_{\mathcal{L}}(\lambda)^2 = \min \sum_{t\in T}\sum_{j\in J}\left(f_j^t - \lambda_j^t H\right) y_j^t \tag{34}$$

$$\text{s.c} \sum_{j\in S} y_j^t \geq p^t \qquad \forall t \in T \tag{35}$$

$$y_j^t \in \{0,1\} \forall j \in J, \forall t \in T. \tag{36}$$

4.3 Résolution SPCFLP et recherche du dual lagrangien

Nous allons présenter la méthode mise en place pour résoudre les différentes relaxations lagrangiennes présentées précédemment.

Proposition 2. *Dual Lagrangien*
La meilleure relaxation possible par application de la méthode lagrangienne pour le SPCFLP

$$w_{LD} = \max\{z_{\mathcal{L}}(\lambda),\ \lambda \geq 0\}$$

Le SPCFLP cherchant à minimiser le coût total d'implantation des différentes facilités, l'application d'une relaxation lagrangienne va nous permettre d'obtenir une borne inférieure pour la valeur optimale du problème initial non relâché. Notre objectif est donc de trouver la meilleure borne inférieure possible. Pour ce faire, nous allons tenter de déterminer un vecteur $\lambda_1 \in \mathbb{R}^{|I|}$ pour la première relaxation et le vecteur $\lambda_2 \in \mathbb{R}^{|J|\times|T|}$ pour la seconde relaxation et ceci en résolvant le dual lagrangien énoncé précédemment.
La recherche du dual langrangien peut être réalisée grâce à l'algorithme du sous gradient que nous allons présenter ci-dessous.
La fonction que l'on souhaite minimiser sont concaves [3], linéaire par morceaux et non différentiable. Devant pareil cas, nous allons mettre en place l'algorithme du sous gradient pour espérer converger vers une borne inférieure pour le SPCFLP. Le sous gradient pour le problème relâché est donné pour :

la relaxation de la contrainte n° 2

$$\gamma(v) = (\gamma_i(v))_{i\in I} = 1 - \sum_{t\in T}\sum_{j\in J} x_{ij}^t(v)\ \forall i \in I$$

la relaxation de la contrainte n° 3

$$\gamma(v) = \left(\gamma_j^t(v)\right)_{j\in J, t\in T} = \sum_{i\in I} x_{ij}(v)^t - H y_j^t(v)\ \forall j \in J\ \forall t \in T$$

Où $(x_{ij}(v)^t, y_j^t(v))\ \forall i \in I\ \forall j \in J\ \forall t \in T$ est un couple de solutions optimales aux différents problèmes relâchés. ($v = (\lambda_i)_{i\in|I|}$ pour la première relaxation et $v = \left(\lambda_j^t\right)_{j\in|J|,t\in|T|}$ pour la seconde).
Afin d'obtenir une solution optimale notée $\lambda^\star$, nous devrions en théorie avoir recourt à des algorithmes de plans coupants mais le temps de résolution avec ces méthodes serait beaucoup trop long.
Pour obtenir des résultats pour ce problème dans un temps raisonnable, nous allons mettre en place une solution algorithmique d'approximation qui va permettre de converger vers la solution optimale plus ou moins rapidement suivant les valeurs de paramètres que nous expliciterons par la suite.

3. une fonction f est dite concave lorsque la fonction opposée $\breve{}f$ est convexe.

Dual lagrangien pour la relaxation de la contrainte n° 2

$$w_{LD} = \max \left\{ \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t y_j^t + \sum_{i \in I} \left(c_{i,j}^t - \lambda_i \right) x_{i,j}^t \right) + \sum_{i \in I} \lambda_i,\ \lambda \geq 0 \right\}$$

Dual lagrangien pour la relaxation de la contrainte n° 3

$$w_{LD} = \max \left\{ \min \sum_{t \in T} \sum_{j \in J} \left(f_j^t - \lambda_j^t H \right) y_j^t + \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} \left(c_{i,j}^t + \lambda_j^t \right) x_{i,j}^t,\ \lambda \geq 0 \right\}$$

Le pseudo code de l'algorithme du sous gradient générique est présenté ci-dessous :

Algorithme 3 : Algorithme du sous-gradient générique

Entrées : I instance du SPCFLP , t le temps limite, μ, ρ
Sorties : (x,y) : solution admissible du dual lagrangien

$u \leftarrow$ vecteur de réels de taille M // Pour la première R$\mathcal{L}$: $M \in \mathbb{R}^{|I|}$, pour la seconde : $M \in \mathbb{R}^{|J| \times |T|}$
$k \leftarrow 0$ // Compteur d'itération
// Initialisation
$u_k \leftarrow 0\ \forall k \in M$
// Mise à jour
Tant que *(le temps limite n'est pas atteint)* **faire**
    $(x(u_k), y(u_k)) \leftarrow \mathrm{R}\mathcal{L}(u_k)$ // Le couple de solutions optimales pour R$\mathcal{L}(u_k)$
    // Mise à jour du coefficient de Lagrange $\gamma(u_k)$
    $u_{k+1} \leftarrow \max\{u_k \pm \mu \times \gamma_i(u), 0\}\ \forall k \in M$
    $\mu \leftarrow \mu \times \rho$
    $k \leftarrow k + 1$
fin
retourner $(x(u_k), y(u_k))$// La meilleure solution obtenue au terme du temps limite.

La difficulté pour un tel algorithme réside dans le choix des paramètres ρ et μ afin d'assurer une convergence rapide vers une solution optimale pour le problème relâché.
Nous avons testé deux valeurs pour le coefficient μ ; Dans un premier temps, nous avons choisi une valeur constante $\mu_k = 1$. Dans un second temps nous avons tenté de faire varier ce paramètre tel que $\mu_k \leftarrow 1/k$.
Dans le second cas étudié, nous observons une convergence bien plus lente. Une variante serait de considérer le coefficient μ suffisamment grand au début de l'algorithme afin d'assurer une convergence *rapide*, et d'appliquer la multiplication par un coefficient ρ permettant de diminuer la valeur de μ comme présenté dans le sous gradient générique ci-dessus. Afin d'appliquer cette méthode, nous avons décidé d'initialiser les coefficients tels que $\mu = 3$ et $\rho = 0.95$.
L'algorithme que nous présentons dans cette section est adapté à partir de celui proposé dans le cours d'optimisation continue ; cependant nous sommes dans le cadre d'un problème de recherche du dual lagrangien en maximisation, nous souhaitons donc nous déplacer dans le sens du gradient $\gamma(u)$. Ainsi, nous mettons à jour le vecteur u tel que

$$u_{k+1} \leftarrow \max\{u_k + \mu \times \gamma_i(u), 0\}$$

5 Heuristique basée sur la solution précédente

Nous avons mis en place une heuristique à partir de la solution obtenue par la relaxation lagrangienne des contraintes (2). Cette relaxation nous donne une solution non admissible pour le problème initial, justement par le non-respect de ces contraintes. En revanche, les contraintes (3) et (4) sont respectées. C'est-à-dire qu'aucune facilité n'est surchargée, et à chaque période, il y a suffisamment de facilités ouvertes. Pour rendre la solution admissible, on doit donc faire en sorte de respecter les contraintes (2), autrement dit chaque client doit être affecté une seule fois. Pour cela nous allons procéder en deux étapes : d'abord enlever les clients qui sont affectés plus d'une fois, puis réutiliser Reactiv GRASP pour affecter les clients manquants.

Pour chaque client i, si il y a plus d'une variable x_{ij}^t à 1, on garde uniquement la moins chère [4]. Parallèlement, on construit une matrice [TxJ] dans laquelle on compte le nombre de clients affectés à une facilité j à une période t. On construit aussi un vecteur de taille T qui compte le nombre de facilités ouvertes à une période t. Enfin, on construit un vecteur contenant les variables x_{ij}^t dont la période t respecte déjà le nombre minimum de facilités à ouvrir (c'est le vecteur d'attente).

À la fin de cette étape on a donc les informations suffisantes pour utiliser GRASP, et ainsi affecter les clients non-affectés.

6 Instructions d'exécution

L'ensemble de nos implémentations ont été réalisées dans la dernière version stable de Julia : Julia 1.1.1 [5]. Depuis cette version de Julia, nous avons pu installer les packages :

— JuMP 0.19.1 [6]
— GLPK 1.1.1

Ces packages s'ajoutent avec la suite de commande suivante :

```
julia> using Pkg
julia> Pkg.add("JuMP")
julia> Pkg.add("GLPK")
```

L'ensemble de nos développements sont exécutables depuis le fichier main.jl à la racine du /src. L'exécution se fait sur l'ensemble des instances si l'appel du main est réalisé avec la valeur 0 sinon l'instance avec le numéro en paramètre sera la seule testée. Un grand nombre de paramètres d'exécution sont décrits au début du fichier dont certains permettant de modifier l'affichage.

Nous avons travaillé sur l'un de nos ordinateurs personnels possédant un processeur intel i7 7200 et 8go de Ram. Ainsi, les résultats obtenus avec notre temps d'exécution bridé à 5min divergeront avec ceux obtenus dans le cas d'une exécution sur les ordinateurs de l'ULB.

7 Résultats expérimentaux

Nous avons lancé les résolution des 20 premières instances avec comme limite temporelle 5 min pour chaque méthode (ce qui fait environ 30 minutes par instance). Les résultats obtenus sont dans le tableau suivant :

4. on reprend la fonction d'utilité $U(x_{ij}^t) = f_j^t + c_{ij}^t$
5. Site officiel du langage Julia : `https://julialang.org/`
6. Git JuliaOpt/JuMP : `https://github.com/JuliaOpt/JuMP.jl`

Nom d'instance	Resolution exacte	relax lineaire	heuristique Simple	Reactiv GRASP	relax (2) lagrangienne	grasp lagrange	relax (3) lagrangienne
data-100-10-4-3-0	4747.0	4747.0	4995.0	4754	4768.0	4756	4753.0
data-100-10-4-3-1	4418.0	4412.0	4764.0	4863	4423.0	4418	4419.0
data-100-10-4-3-2	4471.0	4466.0	4814.0	4868	4427.0	4479	4469.0
data-100-10-4-3-3	4876.0	4872.67	4994.0	5043	4880.0	4885	4875.0
data-100-10-4-3-4	4245.0	4173.0	4370.0	4667	3984.0	4293	4027.0
data-100-10-4-5-0	2784.0	2725.8	4046.0	4046	2712.0	3400	2712.0
data-100-10-4-5-1	3009.0	2940.6	4523.0	4071	2822.0	3776	2967.0
data-100-10-4-5-2	2980.0	2922.2	4547.0	4171	2842.0	3686	3000.0
data-100-10-4-5-3	3035.0	2982.0	4604.0	4187	2971.0	3518	3018.0
data-100-10-4-5-4	2887.0	2851.6	3985.0	3987	2882.0	3102	2874.0
data-150-10-4-5-0	4464.0	4422.0	5546.0	5263	4153.0	4808	4430.0
data-150-10-4-5-1	4472.0	4453.8	5108.0	5066	4495.0	4487	4459.0
data-150-10-4-5-2	4661.0	4622.0	5817.0	5550	4176.0	5050	4598.0
data-150-10-4-5-3	4409.0	4383.2	5250.0	5124	4354.0	4449	4401.0
data-150-10-4-5-4	4156.0	4132.2	4988.0	5106	3761.0	4508	4100.0
data-150-10-4-10-0	3112.0	2968.0	5683.0	4702	1980.0	4413	2955.0
data-150-10-4-10-1	2841.0	2697.6	4643.0	4595	2214.0	4090	2633.0
data-150-10-4-10-0	3112.0	2968.0	5683.0	4685	1980.0	4426	2955.0
data-150-10-4-10-1	2841.0	2697.6	4643.0	4596	2214.0	4053	2633.0

Remarques : À partir des données collectées pendant nos expérimentations numériques, nous pouvons constater les résultats suivants :

- La relaxation lagrangienne opérée sur la contrainte d'affectation (on rappelle la contrainte n° 2) entraîne des résultats inférieurs à ceux obtenus en relâchant la contrainte de capacité maximale des facilités ouvertes (la contrainte n° 3).
- L'utilisation de l'heuristique du Reactiv GRASP sur la relaxation lagrangienne permet d'obtenir de biens meilleures bornes inférieures que la même heuristique appliquée seule.
- L'application de l'heuristique *simple* permet d'obtenir parfois de meilleurs résultats que la méthode utlisant un apprentissage statistique, Reactiv GRASP. Notons de plus que dans certain cas, on observe que les relaxations lagrangiennes, censées fournir des bornes inférieures, retournent des valeurs supérieures à la solution optimale. Nous émettons l'hypothèse que cette incohérence peut provenir des imprécisions numériques accumulées durant les différentes étapes de la descente de gradient.

8 Conclusion

En conclusion on peut dire que l'idée d'utiliser une heuristique sur la solution d'une relaxation Lagrangienne est très performante. D'autant plus que le Reactiv grasp, de par sa fonctionnalité d'apprentissage, sera de plus en plus efficace si on le laisse tourner assez longtemps. Bien sur il existe encore des moyens pour réduire ce temps de calcul. On pourra penser notamment à une parallélisation des calculs mais aussi à une optimisation du code, par exemple, en se débarrassant du langage de modélisation JuMP pour se rapprocher du solveur.

9 Bibliographie

Publication : Yolanda Hinojosa, Elena Fernández, Maria Albareda-Sambola, Justo Puerto — The Single Period Coverage Facility Location Problem : Lagrangean heuristic and column generation approaches