INSTITUT POLYTECHNIQUE DE PARIS

Le Cnam

Institut Polytechnique de Paris

Groupe 1

Optimisation du placement de fonctions de service dans les réseaux télécoms

---

Auteur :

Justine De Sousa

Natalia Jorquera

Yue Zhang

Superviseurs:

Alain Faye

---

le cnam

31 mars 2022

Introduction

Il s'agit de résoudre un problème de flot à commodité multiple. Chaque commodité souhaite transférer une quantité b_k de données d'un noeud source s_k vers un noeud terminal t_k tout en effectuant un certain nombre de traitements sur ces données en respectant un ordre précis et une latence maximale pour la transmission des données. Les traitements peuvent être effectués sur des machines qui constitueront les noeuds du graphe. Les commodités ont également certaines requêtes d'exclusivité qui empêche d'effectuer deux traitements sur une même machines.

On propose différentes méthodes de résolution ou d'approximation de ce problème : un modèle exact en PLNE, son relâché continu, deux décompositions de Dantzig-Wolfe et un algorithme de recuit simulé.

1 Modélisation PLNE

Pour résoudre ce problème, le modèle PLNE suivant est proposé, dont les données, la fonction objectif et les contraintes sont expliquées ci-dessous.

1.1 Ecriture du modèle

Données :

- Graphe $G = (V, E)$ orienté pondéré par $(l_{ij})_{(ij)\in E}$
  - V : un ensemble de machines dénotées par $i \in V \subset \mathbb{N}$
  - E : un arc (ij) représente une communication possible entre les machines i et j
  - l_{ij} : temps de parcours entre le noeud i et le noeud j
- $|V|$ machines
  - c_i : coût d'ouverture du noeud i
  - $\kappa_N(i)$: nombre de fonctions maximum installables au noeud i
- F fonctions installées sur les noeuds $i \in V$
  - c_{fi} : coût d'installation de f sur le noeud i
  - $\kappa_F(f)$: capacité de la fonction f en quantité de données traitées
- K commodités
  - b_k : quantité de données transférées par k
  - s_k : noeud de départ pour k
  - t_k : noeud d'arrivée pour k
  - l_k : temps de parcours maximum autorisé pour k
  - C_k : nombre de fonctions à utiliser par la commodité k
  - $C^k(f)$: couche de la fonction f pour la commodité k
  - $\mathcal{F}_k \subset \{(f, f') | f, f' \in \{1, ..., F\}\}$: couples de fonctions qui ne peuvent être utilisées au même noeud par k

Variables :

- $\boxed{x_{i,j}^{k,c} \in \{0,1\}}\ \forall\, k \in \{1 \dots K\}, \forall\, i \in V, \forall\, j \in V, \forall\, c \in C_k$ (= 1 si le client k emprunte l'arc (ij) à la couche c)

  Remarque : Lorsque $\mathbf{i = j}$, $\mathbf{x_{i,i}^{k,c} = 1}$ signifie qu'une fonction est placée au sommet i et

qu'on passe à la couche suivante

- $\boxed{y_{fi} \in \mathbb{N}}\ \forall\, i \in V, \forall\, f \in \{1 \dots F\}$ (= nombre de fonctions f installées au sommet i)
- $\boxed{u_i \in \{0,1\}}\ \forall\, i \in V$ (= 1 si au moins une fonction est installée au sommet i)

Objectif :

$$\min_{u_i, y_{if}} \quad \underbrace{\sum_{i \in V} c_i \cdot u_i}_{\text{coût d'activation des noeuds}} + \sum_{i \in V} \sum_{f=1}^{F} \underbrace{c_{fi} \cdot y_{fi}}_{\text{coût d'installation de } f \text{ en } i}$$

Contraintes :

Conservation de flux de données :

Premièrement, on utilise un modèle de flot pour établir un chemin de s_k à t_k pour chaque commodité k. Pour chaque commodité, un chemin est décomposé en plusieurs couches. Chaque couche peut s'interpréter comme un étage sur lequel la commodité k parcourt un chemin jusqu'à la prochaine fonction, disons f. Dès lors que f est atteinte (disons au noeud i), alors k passe à l'étage supérieur (où le graphe est répliqué) et y effectue un chemin de i jusqu'au noeud qui contient la fonction suivante. Il faut donc prendre en compte toutes les couches pour écrire le modèle de flot global.

- (Flot de 1 sortant de s_k)

$$\sum_{c=1}^{C_k} \left(\sum_{j \in \delta^+(s_k)} x_{s_k j}^{k,c} - \sum_{j \in \delta^-(s_k)} x_{j s_k}^{k,c} \right) = 1 \quad \forall\, k \in \{1 \dots K\}$$

- (Flot de 1 entrant en t_k)

$$\sum_{c=1}^{C_k} \left(\sum_{j \in \delta^+(t_k)} x_{t_k j}^{k,c} - \sum_{j \in \delta^-(t_k)} x_{j t_k}^{k,c} \right) = -1 \quad \forall\, k \in \{1 \dots K\}$$

- (Conservation du flot en chaque noeud)

$$\sum_{c=1}^{C_k} \left(\sum_{j \in \delta^+(i)} x_{ij}^{k,c} - \sum_{j \in \delta^-(i)} x_{ji}^{k,c} \right) = 0 \quad \forall\, k \in \{1 \dots K\}, i \in V \setminus \{s_k, t_k\}$$

Conservation du flux en chaque couche :

On doit conserver le flot d'une couche à la suivante sur le saut de couche ($x_{ii}^{k,c}$), afin de s'assurer la connexité de flot intra-couches .

- (Conservation du flot en chaque noeud de la couche 1)

$$\sum_{j \in \delta^+(i)} x_{ij}^{k,1} - \sum_{j \in \delta^-(i)} x_{ji}^{k,1} = \begin{cases} x_{ii}^{k,1} - 1, \text{ if } i = s_k \\ x_{ii}^{k,1}, \text{ sinon} \end{cases} \quad \forall\, i \in V, \forall\, k \in \{1 \dots K\}$$

- (Conservation de flot en chaque noeud des couches 2 à C^k)

$$\sum_{j\in\delta^+(i)} x_{ij}^{k,c+1} - \sum_{j\in\delta^-(i)} x_{ji}^{k,c+1} + x_{ii}^{c+1} = x_{ii}^{c} \quad \forall\, i \in V, \forall\, k \in \{1\dots K\}, \forall\, c \in \{1\dots C^k - 1\}$$

Contraintes du problème (capacité, latence) :

Pour chaque commodité k, on s'assure que le temps total de parcours ne dépasse pas à la latence maximum autorisée et qu'il y ait suffisamment de fonctions f au sommet i pour traiter le volume total de données de toutes les commodités passants par i.

- (Temps de parcours maximum)

$$\sum_{c=1}^{C_k} \sum_{(ij)\in A} l_{ij} \cdot x_{ij}^{k,c} \leqslant l_k \quad \forall\, k \in \{1\dots K\}$$

- (Capacité des fonctions)

$$\sum_{k=1}^{K} \left(x_{ii}^{k,C^k(f)}\right) \cdot b_k \leqslant \kappa_F(f) \cdot y_{fi} \quad \forall\, i \in V, \forall\, f \in \{1\dots F\}$$

- (Capacité des machines + ouverture du sommet i)

$$\sum_{f=1}^{F} y_{fi} \leqslant \kappa_N(i) \cdot u_i \quad \forall\, i \in V$$

Variable d'installation des fonctions :

Si aucune fonction n'est installée au sommet i, alors il n'est pas nécessaire d'ouvrir le sommet i.

$$u_i \leqslant \sum_{f=1}^{F} y_{fi} \quad \forall\, i \in V$$

Ordre d'utilisation des fonctions :

Si deux fonctions sont exclusives pour la commodité k, elle ne peut pas emprunter le noeud i sur deux couches différentes. être installées sur le même noeud. Sur chaque couche, il existe exactement un sommet sur lequel est installé une fonction.

- (Contraintes d'exclusivité)

$$x_{ii}^{k,c'} \leqslant 1 - x_{ii}^{k,c} \quad \forall\, (f,f') \in \mathcal{F}_k; (c,c') \in C^k(f') \times C^k(f); \forall\, i \in V, \forall\, k \in \{1\dots K\}$$

- (Une fonction par couche)

$$\sum_{i\in V} x_{ii}^{k,c} = 1 \quad \forall\, k \in \{1,...,K\}, \forall\, c \in \{1\dots C_k\}$$

1.2 Vérification de la faisabilité

Après avoir obtenu la solution de solveur Cplex, il est nécessaire de vérifier que la solution de notre modèle **PLNE** est bien réalisable. Tout d'abord on vérifie que pour toute commodité k, il existe bien un chemin de s_k à t_k (par l'algorithme parcours en largeur) et que la latence maximum est bien respectée. On vérifie ensuite que les fonctions sont installées correctement (dans le bon

ordre et en respectant les contraintes d'exclusivité en chaque noeud). Finalement, on vérifie les contraintes de capacités.

2 Génération de colonne : méthode 1

2.1 Algorithme de génération de colonne

Pour avoir une meilleure borne inférieure du modèle PLNE présenté dans la section précédente, un modèle de décomposition de Dantzig-Wolfe est proposé. L'idée principale est de générer les chemins réalisables ayant le coût réduit minimal pour les k commodités dans le sous-problème en utilisant la même **modélisation par couche** que dans le PLNE. Dans le problème maître, on considère les chemins réalisables comme étant des données et on déplace les fonctions de manière à minimiser le coût total. A chaque itération, on récupère les variables duales associées aux variables de chemins et on passe au sous-problème afin de chercher la solution réalisable de coût réduit minimum. L'algorithme s'arrête soit lorsqu'on a atteint le nombre d'itérations maximum imposé soit lorsque que toutes les commodités ont un coût réduit positif ou nul.

2.2 Solutions initiales

Pour générer les solutions initiales réalisables, on voudrait avoir un ensemble des solutions diverses. Pour cela, on propose les solutions initiales ci-dessous :

- En changeant l'objectif de notre modèle **PLNE** par une constante, résoudre la nouvelle problème par CPLEX donne une solution réalisable.
- Comme notre **sous-problème** renvoie la solution réalisable avec le coût réduit, en changeant son objectif, on peut avoir les différente solution réalisables :
  - Le sous-problème revoie une solution réalisable en remplaçant l'objectif par un constant.
  - On cherche une solution réalisable qui évite d'installer trop de fonctions à un noeud.
  - On cherche une solution réalisable parcourant le moins d'arcs.
  - On cherche une solution réalisable parcourant le plus d'arcs.

2.3 Problème maître (restrictif relâché)

Données :

- $\mathcal{P}^k$: l'ensemble des chemins réalisables pour la commodité k. $\mathcal{P}^k$ contient l'ensemble vecteurs binaires d'arcs [x].

Variables :

- $\boxed{\rho_p^k \in [0,1]}$ (=1 si le chemin p de la commodité k est choisi (i.e. $\mathcal{P}^k[p]$ est choisi))
- $\boxed{y_{fi} \in \mathbb{R}_+}$ $\forall\, i \in V, \forall\, f \in \{1 \dots F\}$ (= nombre de fonctions f installées au sommet i)
- $\boxed{u_i \in [0,1]}$ $\forall\, i \in V$ (= 1 si au moins une fonction est installée au sommet i)

Problème maître :

Fonction objectif :

L'objective n'est pas changé, on veut toujours minimiser le coût total.

$$\min_{u_i, y_{if}, \rho_p^k} \sum_{i \in V} c_i \cdot u_i + \sum_{i \in V} \sum_{f=1}^{F} c_{fi} \cdot y_{fi}$$

Contraintes de convexité :

Pour chaque commodité, on ne choisi qu'un seul chemin.

$$\sum_{p \in \mathcal{P}^k} \rho_p^k = 1 \quad \forall\, k \in K$$

Contraintes de capacités des fonctions :

On doit installer un nombre suffisant de fonctions f au sommet i pour traiter le volume total de données de toutes les commodités passants par i.

$$\sum_{k=1}^{K} \sum_{p \in \mathcal{P}^k} \sum_{c \in C^k(f)} \left(x_{ii}^c \cdot \rho_p^k\right) \cdot b_k \leqslant \kappa_F(f) \cdot y_{fi} \quad \forall\, i \in V, \forall\, f \in \{1 \dots F\}$$

Contraintes de capacités des machines :

- $\sum_{f=1}^{F} y_{fi} \leqslant \kappa_N(i) \cdot u_i \quad \forall\, i \in V$
- $u_i \leqslant \sum_{f=1}^{F} y_{fi} \quad \forall\, i \in V$

Variables duales :

Les variables duales du problèmes maître sont notées :

- $\alpha_k, \forall k \in \{1 \dots K\}$ pour les contraintes convexité
- $\beta_{if}, \forall\, i \in V, \forall\, f \in \{1 \dots F\}$ pour les contraintes capacités de fonctions

Le coût réduit pour chaque commodité k est alors

$$-\alpha_k - \sum_{i \in V} \sum_{f \in F} \sum_{c \in C^k(f)} \beta_{if} \cdot x_{i,i}^c \cdot b_k$$

2.4 Sous-problème pour k fixé (PLNE)

Dans ce sous-problème pour chaque commodité k, on génère un chemin réalisable qui respecte une contrainte de latence maximum. Grâce à la modélisation par couches, on place aussi les fonctions ordonnées en vérifiant les contraintes d'exclusivité également.

Variables :

- $\boxed{x_{i,j}^c \in \{0,1\}}\ \forall\, i \in V, \forall\, j \in V, \forall\, c \in C_k$ ($= 1$ si le client emprunte l'arc (ij) à la couche c)

Sous-problème pour la commodité k :

Fonction objectif :

On cherche une solution réalisable pour la commodité k ayant un coût réduit minimum.

$$\min_x -\alpha_k - \sum_{i\in V}\sum_{f\in F}\sum_{c\in C^k(f)} \beta_{if}\cdot x^c_{i,i}\cdot b_k$$

Contraintes de conservation du flot :

- $\sum_{j\in\delta^+(s_k)} x^c_{s_kj} - \sum_{j\in\delta^-(s_k)} x^c_{js_k} = 1$
- $\sum_{j\in\delta^+(t_k)} x^c_{t_kj} - \sum_{j\in\delta^-(t_k)} x^c_{jt_k} = -1$
- $\sum_{j\in\delta^+(i)} x^c_{ij} - \sum_{j\in\delta^-(i)} x^c_{ji} = 0 \quad \forall\, i \in V\backslash\{s_k,t_k\}$

Conservation du flot sur chaque couche :

- $\sum_{j\in\delta^+(i)} x^1_{ji} - \sum_{j\in\delta^-(i)} x^1_{ij} = \begin{cases} x^1_{ii} - 1, \text{ if } i = s_k \\ x^1_{ii}, \text{ sinon} \end{cases} \quad \forall\, i \in V$
- $\sum_{j\in\delta^+(i)} x^{c+1}_{ij} - \sum_{j\in\delta^-(i)} x^{c+1}_{ji} + x^{c+1}_{ii} = x^c_{ii} \quad \forall\, i \in V, \forall\, c \in \{1\dots C^k - 1\}$

Contraintes de latence et d'exclusivité :

- $\sum_{c=1}^{C_k}\sum_{(ij)\in A} l_{ij}\cdot x^c_{ij} \leqslant l_k$
- $x^{c'}_{ii} \leqslant 1 - x^c_{ii} \quad \forall\,(f,f') \in \mathcal{F}_k : (c,c') \in C^k(f') \times C^k(f), \forall\, i \in V$
- $\sum_{i\in V} x^c_{ii} = 1 \quad \forall\, c \in \{1\dots C_k\}$

3 Génération de colonne : méthode 2

3.1 Algorithme de génération de colonne

Ici on propose un modèle différent utilisant des variables de chemins. Au lieu de générer des chemins réalisable avec des fonctions ordonnées, on peut chercher un chemin réalisable dans le sous-problème et déterminer le placement des fonctions pour les commodités dans le problème maître.

3.2 Solutions initiales

Une approche possible est de générer des solutions initiales en utilisant une heuristique. Ici, on se contente de changer la fonction objectif du sous-problème afin d'obtenir différentes solutions réalisables :

- Une solution réalisable quelconque remplaçant la fonction objectif par une fonction constante

- Une solution réalisable parcourant le moins d'arcs
- Une solution réalisable parcourant le plus d'arcs

3.3 Problème maître (restrictif relâché) :

Données :

- $\mathcal{P}^k$: l'ensemble des chemins réalisables pour la commodité k. $\mathcal{P}^k$ contient l'ensemble vecteurs binaires d'arcs [x].

Variables :

- $\boxed{\rho_p^k \in \{0,1\}}$, $\forall\, k \in \{1 \dots K\}, \forall\, p \in \mathcal{P}^k$ (=1 si le chemin p du commodité k est choisi)
- $\boxed{u_i \in \{0,1\}}$, $\forall\, i \in V$ (=1 si un sommet i est ouvert)
- $\boxed{y_{fi} \in \mathbb{N}}$, $\forall\, i \in V, \forall\, f \in \{1 \dots F\}$ (= le nombre de fonctions f installée au sommet i)
- $\boxed{b_{if}^k \in \{0,1\}}$, $\forall\, i \in V, \forall\, f \in \{1 \dots F\}$ (=1 si f est installée avant ou au sommet i)
- $\boxed{a_{if}^k \in \{0,1\}}$, $\forall\, i \in V, \forall\, f \in \{1 \dots F\}$ (=1, si f est installé exactement au sommet i)

Problème maître :

Fonction objectif :

La fonction objectif n'est pas changée, on veut toujours minimiser le coût total.

$$\min_{u_i, y_{if}, \rho_p^k} \sum_{i \in V} c_i \cdot u_i + \sum_{i \in V} \sum_{f=1}^{F} c_{fi} \cdot y_{fi}$$

Contraintes de convexité :

Pour chaque commodité, on ne choisi qu'un seul chemin.

$$\sum_{p \in P^k} \rho_p^k = 1 \quad \forall\, k \in \{1 \dots K\}$$

Contraintes logique de placement des fonctions :

Pour chaque commodité k,

- (On ne déplace pas de fonction avant le sommet s_k)

$$b_{s_k f}^k = 0 \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}$$

- (On ne déplace pas de fonction après le sommet t_k)

$$b_{t_k f}^k = 1 \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}$$

- (Chaque fonction doit être installée)

$$\sum_{i \in V} a_{if}^k \geqslant 1 \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}$$

- (On n'installe pas de fonction s'il elle ne passe pas par ce sommet)

$$a_{if}^k \leqslant \sum_{p \in \mathcal{P}^k} \sum_{a \in A \text{ t.q. } i \in a} x_a^{k,p} \cdot \rho_p^k \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, i \in \{1 \dots V\} \quad (*)$$

- (Si f est installée après le sommet i, alors $f' > f$ aussi)

$$b_{if'}^k \leqslant b_{if}^k \quad \forall\, k \in \{1 \dots K\}, \forall\, i \in \{1 \dots V\}, \forall\, f < f' \in \left\{1 \dots F^k\right\}$$

- (Si f est installée après le sommet i, alors elle n'est pas installée en i)

$$a_{if}^k \leqslant b_{if}^k \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, i \in \{1 \dots V\}$$

- (contraintes d'exclusivité)

$$a_{if}^k + a_{if'}^k \leqslant 1 \quad \forall\, k \in \{1 \dots K\}, \forall\, (f, f') \in \mathcal{F}_k, \forall\, i \in \{1 \dots V\}$$

Contraintes placement de fonction :

Pour chaque commodité k, une fonction f peut être installée au sommet j si seulement si la commodité k passe par l'arc (i, j) et que la fonction f est installée après i ($b_{if}^k = 0$) et avant j ($b_{jf}^k = 1$).

$$\left(\sum_{p \in \mathcal{P}^k} x_{ij}^{kp} \cdot \rho_p^k\right) - 1 + (b_{jf}^k - b_{if}^k) \leqslant a_{jf}^k \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, (i, j) \in A$$

Contraintes capacités des fonctions et des machines :

- (Si le sommet i est fermé, la commodité k n'installe pas de fonction en i)

$$a_{if}^k \leqslant u_i \quad \forall\, k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, i \in \{1 \dots V\}$$

- (Contraintes de capacité des fonctions)

$$\sum_{k \in K} b_k \cdot a_{if}^k \leqslant capa(f) \cdot y_{fi} \quad \forall\, f \in \{1 \dots F\}, \forall\, i \in \{1 \dots V\}$$

- (Contraintes de capacité des machines)

$$\sum_{f \in F} y_{fi} \leqslant capa(i) \cdot u_i \quad \forall\, i \in \{1 \dots V\}$$

Variables duales :

Les variables duales du problèmes maître associées aux variables chemins sont notées :

- $\alpha_k \quad \forall k \in \{1 \dots K\}$ pour les contraintes de convexité
- $\mu_{f,a}^k \quad \forall k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, a \in A$ pour les contraintes de placement des fonctions
- $\omega_{f,i}^k \quad \forall k \in \{1 \dots K\}, \forall\, f \in \left\{1 \dots F^k\right\}, \forall\, i \in V$ pour les contraintes d'interdiction d'installation de fonction $(*)$

Le coût réduit pour chaque commodité k est alors :

$$-\alpha_k - \sum_{f \in F^k} \sum_{a \in A} \mu_{f,a}^k \cdot x_a + \sum_{f \in F^k} \sum_{i \in V} \sum_{a \in A \text{ t.q. } i \in a} \omega_{f,i}^k \cdot x_a$$

3.4 Sous-problème pour k fixé (PLNE)

Dans ce sous-problème pour chaque commodité k, on génère un chemin réalisable de coût réduit minimum qui respecte uniquement la contrainte de latence maximum.

Variables :

- $\boxed{x_{i,j} \in \{0,1\}}\ \forall\ (i,j) \in A$ (= 1 si le client emprunte l'arc (i,j))

Sous-problème pour la commodité k :

Fonction objectif :

On cherche une solution réalisable pour la commodité k ayant un coût réduit minimum.

$$\min_{x_{i,j}} -\alpha_k - \sum_{f \in F^k} \sum_{a \in A} \mu^k_{f,a} \cdot x_a + \sum_{f \in F^k} \sum_{i \in V} \sum_{a \in A \text{ t.q. } i \in a} \omega^k_{f,i} \cdot x_a$$

Contraintes de flot :

- (Un chemin de s_k à t_k)

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = \begin{cases} 0, \forall i \in V \backslash \{s_k, t_k\} \\ -1, \text{ si } i = t_k \\ 1, \text{ si } i = s_k \end{cases}$$

Contrainte de latence :

$$\sum_{(i,j) \in A} x_{i,j} \cdot l_{i,j} \leqslant l_k$$

4 Solution heuristique : le recuit simulé

Pour obtenir une borne supérieure sur la solution de ce problème, nous proposons l'utilisation de la métaheuristique "Recuit simulé" illustrée dans l'algorithme 1. Celle-ci est appliquée sur une version du problème relaxé, où les contraintes associées à l'ordre des fonctions ne sont pas considérées. Pour initialiser cet algorithme, une première solution est trouvée en deux étapes :

- Trouver des **plus courts chemins** à l'aide d'une heuristique inspirée des algorithmes Dikjstra et $A^\star$. À chaque noeud est attribué un score qui correspond à la latence totale du plus court chemin depuis la source jusqu'à ce sommet. A chaque itération, on considère un noeud puis on met à jour les scores de ses voisins. Une fois qu'on a traité un sommet et tous ses voisins, il est placé dans une liste appelée `closed_list` qui signifie que son étiquette est définitive et le sommet utilisé pour calculer son coût est sauvegardé pour reconstituer le chemin en sortie d'algorithme. On démarre l'algorithme en considérant le noeud source et à chaque début d'itération, on sélectionne le noeud de score minimum qui n'est pas dans `closed_list`. Notons que, comme on minimise le coût, le chemin renvoyé respecte la contrainte de latence maximum autorisée. On trouve indépendemment un plus court chemin pour chacunes des commodités.
- Assigner les **fonctions** aux nœuds selon un principe glouton. On assigne d'abord les fonctions aux nœuds les plus utilisés par les commodités jusqu'à ce que leur capacité maximale soit atteinte.

Algorithme 1 : Algorithme de recuit simulé

Entrées :

- Données du problème ;
- T_0, ϕ, $\bar{T}$, maxTime, nombreIt ;

```
Initialiser x_min = Solution trouvée par A* ; x ← x_min ;
Initialiser T = T_0, time=0 ;
tant que T >= T̄ and time<maxTime faire
    pour chaque i allant de 1 à nombreIt faire
        Construire un voisinage V(x) de x;
        Tirer au sort x̄ dans V(x);
        ΔE ← f(x̄) − f(x);
        si ΔE ⩽ 0 alors
            x ← x̄;
            si f(x̄) < f(x_min) alors
                x_min ← x
            fin
        sinon
            Tirer au sort un nombre q dans [0, 1];
            si q ⩽ e^(−ΔE/T) alors
                x ← x̄;
            fin
        fin
    fin
    T ← φ × T;
fin
si x_min n'est pas réalisable alors
    Récupération de la faisabilité
fin
```

Le **voisinage** $V(x)$ associé à une solution x est trouvé en modifiant de façon aléatoire une portion du chemin utilisé par chaque commodité dans x, puis pour chacune d'elles, le placement des fonctions est effectué de la même manière que dans la formation de la solution initiale. La fonction utilisée pour évaluer une solution comprend les coûts du problème ($C(x)$), plus une pénalité pour chaque contrainte qui n'est pas respectée en comptant le nombre de fonctions placées au même nœud qui ont une contrainte d'exclusivité ($excl(x)$), la latence supplémentaire que prend un chemin par rapport à la latence maximale autorisée ($extraLat(x)$) et le nombre de fonctions qui ne sont pas ordonnées ($nonOrd(x)$), ce qui donne la fonction suivante :

$$f(x) = C(x) + \alpha \cdot excl(x) + \beta \cdot extraLat(x) + \gamma \cdot nonOrd(x)$$

Enfin, une fois le processus de recuit simulé terminé, une heuristique de réparation est appliquée pour convertir la solution trouvée en une solution réalisable pour laquelle les fonctions placées dans les nœuds sont réorganisées de telle sorte que chaque commodité respecte son ordre établi.

Heuristique de réparation

Pour une commodité k,

- On considère f_1 la première fonction à placer pour la commodité k. On considère les noeuds parcourus par k dans l'ordre. Si f_1 est placée sur le noeud i, alors on passe à la fonction f_2 et on considère le même chemin mais seulement à partir de i.

- Si, pour une certaine fonction f, on a parcouru le chemin jusqu'à la fin sans la trouver, il est alors nécessaire de la placer. Pour choisir où la placer, on considère l'ensemble des noeuds déjà ouverts pour la solution x qui sont placés sur le chemin de k après i. S'il est possible de placer f en respectant les contraintes de capacité et d'exclusivité, on le fait et on passe à la fonction suivante dans l'ordre demandé par k.
- Si une des contraintes du problème n'est pas respectée lors du placement de f en i, alors on considère le prochain noeud situé sur le chemin de k.

5 Résultats numériques

Nous présentons les résultats obtenus dans le Tableau 1 avec les temps d'exécution de chaque méthode sur chacune des instances executées. Un GAP a également été calculé pour mesurer l'écart de la bonne obtenue à la valeur du PLNE, cette valeur nous permet de mesurer la qualité des bornes inférieures et supérieures qu'on a obtenues. On présente également un diagramme de performance Figure 1 qui compare les cinq méthodes de résolution/approximation utilisées. Sur ce diagramme on peut lire, pour chacune des méthodes, le nombre d'instances résolues en

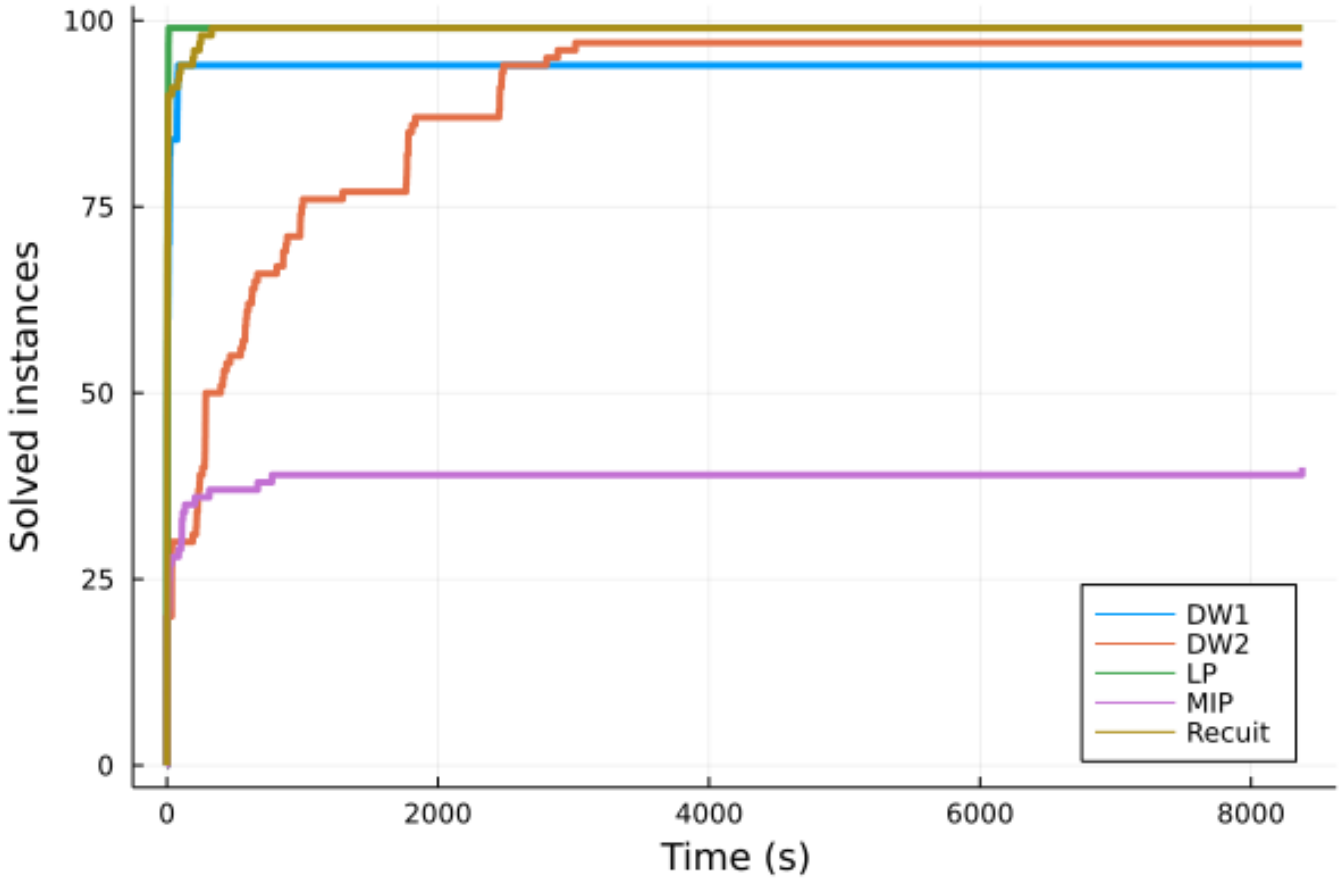

Figure 1 – Diagramme de performance

fonction du temps. On observe que la relaxation continue (**PL**) est très rapide pour résoudre la totalité des instances et l'heuristique est juste derrière. La première méthode par génération de colonnes (**DW1**) est également très rapide mais ne parvient pas à résoudre les instances *atlanta* 5 à 10. La deuxième méthode par génération de colonnes (**DW2**) est bien plus lente et parvient à résoudre 97 d'instances (donc une différence non significative et d'ailleurs, il peut être intéressant de noter que ce ne sont pas les mêmes instances qui ont faillies pour les deux méthodes. Enfin, la méthode par **PLNE** qui est la seule méthode qui donne une solution exacte est bien plus lente et pour des raisons de temps, plus de la moitié des instances n'ont pas été résolues.

5.1 Performance de modèle PLNE

Bien que l'implémentation du modèle PLNE soit performante, voire plus rapide que les méthodes d'approximation sur certaines instances telles que Pdh et di-yuan, dans la plupart des cas, l'exécution s'arrête par faute de place mémoire en RAM et ne peut donc pas trouver de solution.

5.2 Comparaison des deux modèles de génération de colonnes

Le premier modèle proposé (**DW1**, basé sur le **PLNE**) utilise moins de variables (de 3 types) mais la contrainte de capacité des fonctions est faible. On travaille sur le problème maître relâché. Les coefficients de y_{fi} étant positifs, il s'agit de les minimiser. Ainsi, les contraintes de capacité des fonctions sont toujours saturées donc les variables duales associées sont constantes. L'algorithme est donc rapide car peu de variables mais n'évolue pas beaucoup et la borne inférieure obtenue est assez proche de la borne obtenue par relaxation (**PL**).

Le deuxième modèle proposé (**DW2**) utilise plus de variables binaires et de contraintes logiques pour placer les fonctions dans l'ordre demandé mais ses contraintes sont plus fortes lors de la relaxation. Par conséquent, les bornes inférieure obtenue sont plus proches des valeurs du **PLNE**.

Par exemple, pour les instances de "pdh", les bornes inférieures du **PL** et de **DW1** sont assez proches et sont distantes d'environ 20%-30% de la solution optimale. Tandis que l'écart pour **DW2** se réduit à **6%-12%**.

En conclusion, **DW2** donne de meilleures bornes inférieures que **DW1** et que le **PL**.

5.3 Performance de la méta-heuristique

L'algorithme de recuit simulé a d'excellentes performances. Il réussi à trouver une solution réalisable assez vite et assez proche de la solution optimale.

Par exemple, pour les instances "atlanta", "di-yuan" et "pdh" l'écart à la solution optimale est entre **1%** et **16%**.

Conclusion et perspectives

- Étant donné que la méta-heuristique donne une solution réalisable de très bonne qualité, on pourrait l'utiliser afin d'initialiser le **PLNE** dans Cplex et ainsi accélérer le processus de branchement.
- Afin de renforcer les bornes données par le premier modèle en **PLNE**, on pourrait tenter d'ajouter des inégalités valides.
- Les chemins nécessaires au problème maître de **DW2** ne doivent respecter que la contrainte de latence maximale. Une autre approche en PLNE pourrait être de générer tous les chemins de s_k à t_k qui respectent cette contrainte de latence en pré-traitement. Puis d'appliquer le problème maître de **DW2** en nombres entiers.

Instances	PLNE		LP			DW1			DW2			Recuit Simulé		
	Temps(s)	Valeur	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP
pdh_1	0.16	63345.0	0.44	44278.48	30.1%	0.49	44278.48	30.1%	0.57	55318.28	12.67%	0.07	66105.0	4.36%
pdh_2	0.14	69472.0	0.57	46475.04	33.1%	0.54	46851.86	32.56%	0.33	63426.87	8.7%	0.07	74323.0	6.98%
pdh_3	0.33	64766.0	0.59	39904.13	38.39%	0.42	40487.4	37.49%	0.92	60353.48	6.81%	0.06	72551.0	12.02%
pdh_4	0.15	67689.0	0.47	41376.4	38.87%	0.47	41571.14	38.59%	0.33	61102.51	9.73%	0.09	75619.0	11.72%
pdh_5	0.15	61051.0	0.46	37152.77	39.14%	0.51	37496.59	38.58%	0.86	53735.23	11.98%	0.07	67680.0	10.86%
pdh_6	0.25	63558.0	0.37	40345.0	36.52%	0.4	40708.6	35.95%	0.3	56626.8	10.91%	0.07	71845.0	13.04%
pdh_7	0.12	64457.0	0.68	40881.8	36.58%	0.47	42108.04	34.67%	0.41	58304.98	9.54%	0.07	72289.0	12.15%
pdh_8	0.12	72216.0	0.53	50683.94	29.82%	0.42	51464.91	28.73%	0.31	61698.12	14.56%	0.09	77823.0	7.76%
pdh_9	0.15	72289.0	0.6	49746.66	31.18%	0.33	50528.01	30.1%	0.38	59438.92	17.78%	0.12	80115.0	10.83%
pdh_10	0.2	80203.0	0.68	62623.14	21.92%	0.44	63049.19	21.39%	0.32	73348.65	8.55%	0.07	82756.0	3.18%
di-yuan_1	0.63	60551.0	0.4	32940.75	45.6%	1.03	32940.75	45.6%	35.88	53268.61	12.03%	0.61	71055.0	17.35%
di-yuan_2	0.7	62934.0	0.51	37637.06	40.2%	0.86	37637.06	40.2%	35.23	57059.65	9.33%	0.41	67772.0	7.69%
di-yuan_3	0.59	58135.0	0.41	28434.08	51.09%	0.87	28513.63	50.95%	35.93	53039.3	8.77%	0.55	65421.0	12.53%
di-yuan_4	0.45	57695.0	0.49	27387.69	52.53%	0.83	27387.69	52.53%	35.08	51606.2	10.55%	0.4	63603.0	10.24%
di-yuan_5	0.4	58974.0	0.43	31954.41	45.82%	0.86	32080.37	45.6%	35.28	53231.51	9.74%	0.42	64045.0	8.6%
di-yuan_6	0.43	59446.0	0.42	28575.61	51.93%	1.06	28575.61	51.93%	35.15	51766.03	12.92%	1.02	66903.0	12.54%
di-yuan_7	0.4	64045.0	0.49	35581.8	44.44%	0.75	35581.8	44.44%	35.56	59481.7	7.13%	0.7	74259.0	15.95%
di-yuan_8	0.47	61392.0	0.42	37165.43	39.46%	0.88	37268.03	39.29%	36.43	55260.77	9.99%	0.83	70165.0	14.29%
di-yuan_9	0.4	60318.0	0.45	35403.87	41.3%	1.34	35404.45	41.3%	35.29	56669.64	6.05%	0.73	69340.0	14.96%
di-yuan_10	0.75	64767.0	0.44	34160.71	47.26%	1.1	34179.45	47.23%	38.17	56255.48	13.14%	0.53	61946.0	4.36%
dfn-bwin_1	313.2	71563.0	3.49	34215.19	52.19%	2.6	34726.57	51.47%	2.35	69966.31	2.23%	1.11	185159.0	158.74%
dfn-bwin_2	133.39	74466.0	3.1	38630.97	48.12%	1.54	39257.11	47.28%	1.92	64136.37	13.87%	0.78	184137.0	147.28%
dfn-bwin_3	86.49	94220.0	2.16	57394.01	39.09%	1.5	58473.74	37.94%	1.82	76935.29	18.35%	0.64	208796.0	121.6%
dfn-bwin_4	670.05	79039.0	2.6	42008.47	46.85%	1.58	42950.49	45.66%	2.2	73944.78	6.45%	1.84	194509.0	146.09%
dfn-bwin_5	34.49	114696.0	3.32	84644.88	26.2%	1.38	84993.6	25.9%	1.97	94567.66	17.55%	0.51	237819.0	107.35%
dfn-bwin_6	10.0	72915.0	2.44	33156.37	54.53%	1.42	34850.08	52.2%	1.86	68868.5	5.55%	1.61	196353.0	169.29%
dfn-bwin_7	206.81	73762.0	3.38	38647.49	47.61%	1.58	39043.31	47.07%	1.86	66927.6	9.27%	0.87	173125.0	134.71%
dfn-bwin_8	14.34	88450.0	3.13	55685.77	37.04%	1.93	58331.64	34.05%	1.87	87218.57	1.39%	0.82	204826.0	131.57%
dfn-bwin_9	775.27	78876.0	2.89	44836.98	43.16%	1.27	45740.14	42.01%	2.13	73622.62	6.66%	2.0	182932.0	131.92%
dfn-bwin_10	111.13	69495.0	2.74	33609.86	51.64%	1.51	34303.6	50.64%	1.99	65308.63	6.02%	1.68	185439.0	166.84%
atlanta_1	11.15	Inf	12.29	252742.01	-	25.07	253447.68	-	1771.7	301342.2	-	4.31	543167.0	-
atlanta_2	110.13	396057.0	10.17	56343.23	85.77%	17.46	56639.91	85.7%	1832.53	111201.86	71.92%	4.55	388225.0	1.98%
atlanta_3	15.08	389645.0	10.39	77569.07	80.09%	16.7	77452.64	80.12%	1764.43	122322.61	68.61%	10.23	362983.0	6.84%
atlanta_4	14.18	343076.0	11.02	54172.78	84.21%	16.42	54155.21	84.21%	1780.72	113578.15	66.89%	10.23	389942.0	13.66%
atlanta_5	117.55	370307.0	11.18	77304.66	79.12%	-	-	-	1781.1	130585.9	64.74%	3.4	393792.0	6.34%
atlanta_6	13.35	364516.0	11.16	55676.41	84.73%	-	-	-	1782.27	114469.91	68.6%	4.36	387314.0	6.25%
atlanta_7	110.56	410639.0	10.91	109224.87	73.4%	-	-	-	1771.72	164549.04	59.93%	3.18	441103.0	7.42%
atlanta_8	104.17	424439.0	7.97	86740.36	79.56%	-	-	-	1767.21	142824.23	66.35%	10.31	403536.0	4.92%
atlanta_9	18.67	335732.0	10.65	52096.24	84.48%	-	-	-	1769.82	110011.32	67.23%	3.99	390275.0	16.25%
atlanta_10	19.27	498054.0	13.85	206493.3	58.54%	-	-	-	1807.92	210758.24	57.68%	6.13	449670.0	9.71%

Instances	PLNE		LP			DW1			DW2			Recuit Simulé		
	Temps(s)	Valeur	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP
polska_1	-	-	1.93	113770.6	-	2.83	114136.3	-	190.81	117798.08	-	0.43	171350.0	-
polska_2	-	-	1.81	136200.89	-	2.57	136918.19	-	240.76	145583.11	-	0.42	192309.0	-
polska_3	-	-	1.65	112363.06	-	3.76	112869.66	-	262.91	118467.51	-	0.37	159507.0	-
polska_4	-	-	2.27	124723.18	-	2.7	125147.68	-	233.44	130792.0	-	0.45	185336.0	-
polska_5	-	-	1.76	122354.82	-	2.6	123218.96	-	214.63	125926.17	-	10.03	170843.0	-
polska_6	-	-	1.77	127679.62	-	2.92	128050.67	-	227.03	132695.56	-	0.25	176121.0	-
polska_7	-	-	2.33	122313.08	-	2.99	122725.19	-	218.71	134880.03	-	0.35	169961.0	-
polska_8	-	-	1.82	131722.69	-	2.23	131998.79	-	228.4	136737.0	-	10.02	189861.0	-
polska_9	-	-	1.85	121823.07	-	2.88	122142.3	-	243.44	131888.54	-	10.03	169216.0	-
polska_10	-	-	1.67	90480.01	-	3.2	90582.89	-	220.66	105455.2	-	10.02	149403.0	-
dfn-gwin_1	-	-	4.31	101179.62	-	3.02	102224.63	-	285.68	128756.05	-	2.53	252449.0	-
dfn-gwin_2	-	-	2.59	146275.63	-	2.62	148037.42	-	286.88	161596.44	-	3.77	278482.0	-
dfn-gwin_3	-	-	5.31	85419.55	-	3.2	87133.84	-	285.16	116782.89	-	2.99	239544.0	-
dfn-gwin_4	-	-	4.23	85832.18	-	3.18	87894.76	-	278.42	106022.61	-	2.18	218655.0	-
dfn-gwin_5	-	-	2.05	154684.52	-	2.78	155481.53	-	283.44	167816.48	-	3.95	314431.0	-
dfn-gwin_6	-	-	3.72	127301.49	-	2.67	130024.81	-	283.74	154972.91	-	4.87	272630.0	-
dfn-gwin_7	-	-	5.09	141193.78	-	2.94	143674.07	-	280.26	151884.08	-	1.83	261905.0	-
dfn-gwin_8	-	-	2.36	173102.16	-	2.72	174090.2	-	282.07	180772.11	-	3.5	290686.0	-
dfn-gwin_9	-	-	4.73	119194.73	-	3.86	121161.54	-	281.53	137559.59	-	4.01	259539.0	-
dfn-gwin_10	-	-	6.23	157899.9	-	3.72	160028.1	-	284.57	174577.36	-	4.19	301919.0	-
nobel-germany_1	-	-	2.46	51748.13	-	9.95	52422.35	-	983.08	79825.3	-	0.86	235665.0	-
nobel-germany_2	-	-	6.23	29738.84	-	10.24	29990.63	-	1003.19	63911.56	-	0.72	213419.0	-
nobel-germany_3	-	-	2.4	89912.98	-	9.19	90524.59	-	994.21	110219.01	-	330.39	255917.0	-
nobel-germany_4	-	-	3.73	65829.56	-	8.78	65966.81	-	986.24	88774.78	-	240.39	261178.0	-
nobel-germany_5	-	-	2.75	89475.68	-	11.23	89697.83	-	1297.26	119537.38	-	10.1	253326.0	-
nobel-germany_6	-	-	3.0	85498.46	-	11.17	85938.68	-	857.84	112243.33	-	200.5	265850.0	-
nobel-germany_7	-	-	2.45	42950.36	-	10.05	43061.56	-	858.76	77784.98	-	100.33	244194.0	-
nobel-germany_8	-	-	2.29	56068.78	-	8.84	56705.32	-	878.1	83035.79	-	90.61	223897.0	-
nobel-germany_9	-	-	2.36	83724.08	-	10.94	83845.15	-	984.55	102467.4	-	340.38	244259.0	-
nobel-germany_10	-	-	2.41	39517.26	-	11.1	39713.71	-	887.42	72993.57	-	80.37	234784.0	-
nobel-us_1	8376.91	99407.0	2.31	58637.91	41.01%	4.89	59596.65	40.05%	420.33	86875.13	12.61%	0.36	215756.0	117.04%
nobel-us_2	-	-	2.25	44752.86	-	6.13	44906.8	-	397.46	71833.02	-	0.53	191762.0	-
nobel-us_3	-	-	2.28	118412.37	-	5.38	119432.03	-	-	-	-	0.38	263727.0	-
nobel-us_4	-	-	3.74	84863.24	-	4.9	85110.81	-	-	-	-	0.55	211075.0	-
nobel-us_5	-	-	3.25	153502.36	-	5.35	153772.36	-	-	-	-	0.54	284457.0	-
nobel-us_6	-	-	2.2	45258.23	-	4.48	45271.33	-	810.37	69794.76	-	0.45	181226.0	-
nobel-us_7	-	-	2.02	40786.17	-	6.16	40800.22	-	439.46	63406.94	-	0.6	196915.0	-
nobel-us_8	-	-	3.4	73504.45	-	6.11	73670.27	-	413.0	93526.87	-	0.36	199956.0	-
nobel-us_9	-	-	2.34	49446.39	-	4.57	49478.32	-	466.29	73369.54	-	0.46	199860.0	-
nobel-us_10	-	-	2.61	51332.14	-	5.08	51358.68	-	544.5	79787.33	-	0.71	192576.0	-

Instances	PLNE		LP			DW1			DW2			Recuit Simulé		
	Temps(s)	Valeur	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP	Temps(s)	Valeur	GAP
newyork_1	-	-	8.6	235152.62	-	18.23	235685.2	-	2455.12	286579.71	-	8.83	511920.0	-
newyork_2	-	-	10.45	261604.64	-	20.03	261806.36	-	2471.25	311365.37	-	7.07	543626.0	-
newyork_3	-	-	7.71	115484.11	-	17.42	115526.45	-	2455.4	173706.26	-	6.99	426057.0	-
newyork_4	-	-	8.16	201027.14	-	18.35	203399.33	-	2450.54	261670.11	-	9.17	500612.0	-
newyork_5	-	-	8.67	181245.12	-	18.92	181637.25	-	2457.56	212190.14	-	7.21	469129.0	-
newyork_6	-	-	6.94	89669.51	-	16.74	89849.3	-	2469.14	145094.27	-	8.1	406054.0	-
newyork_7	-	-	7.32	304738.48	-	24.79	307107.39	-	2482.78	359902.35	-	6.82	603021.0	-
newyork_8	-	-	6.5	238885.45	-	18.39	240139.12	-	2801.36	282772.75	-	7.62	510840.0	-
newyork_9	-	-	6.56	102711.68	-	20.42	102938.8	-	3014.57	152128.78	-	8.02	429004.0	-
newyork_10	-	-	7.73	212920.95	-	19.92	213612.8	-	2884.12	267834.47	-	8.11	515937.0	-

Table 1 – Comparaison entre les bornes obtenues et la valeur optimale