Cours C9-1 — projet « GOTIC »

vianney.boeuf@inria.fr
Année 2015

1 Consignes

— *Date de rendu du modèle frontal (écrit)* : 28 septembre 2015
— *Date limite de rendu du rapport et des sources* : 19 octobre 2015
— *Soutenance* : 19 octobre 2015

Ce projet compte pour moitié dans votre évaluation du cours C9-1. Vous aurez à rendre par groupes de deux :

— Le modèle frontal (Question 1) (noté 2pts/10) ;
— Un rapport, basé sur les questions de ce document ;
— Vos sources, rendues suffisamment lisibles, avec une note explicative pour pouvoir compiler et exécuter ;
— Des fichiers solutions pour les différentes instances, au format spécifié en partie 5.2.

2 Présentation informelle

Le sujet de ce TD est issu du problème « GOTIC » de Gestion Optimale des tournées des Techniciens d'Intervention Clients (TIC) à France Télécom. Ce problème a été adapté et simplifié pour l'enseignement, sous plusieurs versions renouvelées, par Thierry Defaix, Maurice Diamantini, Éric Gourdin, Nicolas Grebille et Olivier Klopfenstein.

Les TIC sont chargés de répondre aux demandes des clients d'un opérateur de télécommunication en intervenant physiquement chez le client (par exemple pour l'installation d'une ligne téléphonique) ou sur le réseau (par exemple pour la connection d'une nouvelle ligne ADSL). Les demandes d'interventions sont recueillies par différents canaux (centres d'appel, agences, etc...) et sont centralisées au niveau du service qui doit planifier l'activité des TIC. Chaque intervention est caractérisée par un lieu géographique, une plage horaire où l'intervention doit avoir lieu et un type de compétence requis pour pouvoir effectuer l'intervention. Le plan de charge des TIC est élaboré toutes les semaines, en fonction des techniciens disponibles, puis revu tous les jours en fonction des aléas. L'objectif du planificateur est de satisfaire toutes les interventions dans les plages horaires prévues et ce, en minimisant la somme des longueurs des tournées.

Le problème de planification des tournées des TIC s'apparente au problème académique du VRP/TW (*Vehicule Routing Problem with Time Windows*), dans lequel il faut planifier des tournées partant d'un dépot central et visitant chaque client dans une fenêtre de temps. Dans le VRP/TW, les tournées sont réalisées dans l'objectif de livrer ou de collecter des marchandises : les véhicules ont des capacités, et chaque client demande une certaine quantité ou un certain volume de marchandises. Le problème que l'on considère ici est différent du VRP/TW sur plusieurs aspects. En premier lieu, il n'y a pas de capacités ni de demandes : les véhicules n'ont pas pour rôle de transporter des marchandises ; en revanche, chaque intervention doit être réalisée par un technicien ayant une compétence spécifique. D'autre part, le « dépôt » n'est pas unique : contrairement au VRP où toutes les tournées partent du même noeud (le dépôt central), chaque TIC a son propre noeud de départ et d'arrivée. En pratique, certains TIC partent directement de chez eux et d'autres partent d'une base (qui n'est pas nécessairement unique).

3 Données et notations

L'horizon de temps considéré est d'une journée, et tous les temps seront exprimés en minutes entières ; par exemple, $t = 480$ représente la minute débutant à 8h et $t = 1080$ celle débutant à 18h (les temps et les durées sont donc des entiers compris entre 0 et 1439).

Les données géographiques (des bases des techniciens et des lieux d'interventions) sont données en coordonnées entières dans le carré $[0, 100]^2$ (l'unité est le kilomètre). La distance d entre deux points sera calculée à partir de la distance euclidienne *arrondie à l'entier le plus proche*. Par exemple, pour calculer la distance entre les deux points $A = (10, 48)$ et $B = (63, 25)$, on commence par calculer la distance euclidienne :

$$\|A - B\| = \sqrt{(63 - 10)^2 + (25 - 48)^2} \approx 57.8$$

et on arrondit *au plus proche voisin*, ce qui donne donc $d(A, B) = 58$ (exprimée en km).

La vitesse des techniciens sera supposée toujours égale à une moyenne V, exprimée en km/h. Le temps nécessaire pour parcourir la distance d'un point A a un point B sera calculé à partir de la distance *arrondie* $d(A, B)$ entre les deux points, et ensuite de nouveau arrondi *à la minute entière la plus proche*. Dans l'exemple précédent, si V est de 50km/h, alors un technicien parcourra le trajet entre A et B en $58/50 = 1.16$ heures, soit 69.6 minutes, ce que l'on arrondit à 70 minutes.

Soit W l'ensemble des techniciens (*workers*), S l'ensemble des compétences (*skills*), et J l'ensemble des tâches à effectuer (*jobs*).

Chaque technicien $w \in W$ est caractérisé par :

- la position géographique B_w de sa base, c'est-à-dire le lieu où il doit débuter et finir sa tournée ;
- un ensemble de compétences $S_w \subseteq S$;
- les instants T_w^{start} et T_w^{end} où il commence et termine sa journée, exprimée en minutes (entières).

Le technicien w doit impérativement partir de sa base après T_w^{start} (inclus), et doit y être retourné avant T_w^{end} : aucun déplacement n'est possible durant le pas de temps T_w^{end}.

Les caractéristiques d'une tâche $j \in J$ sont :

- sa position géographique (x_j, y_j) ;
- la compétence $s_j \in S$ nécessaire pour la réaliser ;
- la durée D_j de l'intervention, en minutes (entières) ;
- un booléen r_j qui indique si l'intervention se fait sur rendez-vous ;
- un créneau $[t_j^{\min}, t_j^{\max}]$ qui donne la fenêtre de temps dans laquelle l'intervention doit impérativement débuter (où $t_j^{\min}$ et $t_j^{\max}$ sont également donnés sous forme d'entiers indiquant un numéro de minute dans la journée). Si l'intervention est « sans rendez-vous », il ne faut pas tenir compte de la fenêtre de temps, et cette fenêtre de temps sera donc de la forme $[0, 1439]$) ;
- une pénalité P_j, qu'on n'utilisera pas dans ce projet.

Un technicien w ne peut réaliser la tâche j que s'il a la compétence nécessaire ($s_j \in S_w$), s'il reste suffisamment longtemps au point $L_j = (x_j, y_j)$ au cours de sa tournée pour réaliser la tâche, et s'il arrive dans le bon créneau de rendez-vous : si t^{in} et t^{out} sont les instants auxquels le technicien w arrive et repart de L_j, alors la condition $t^{\text{out}} - t^{\text{in}} \geq D_j$ doit être satisfaite. De plus, si l'intervention se fait sur rendez-vous ($r_j = 1$), alors la condition $t_j^{\min} \leq t^{\text{in}} \leq t_j^{\max}$ doit être satisfaite.

Enfin, toutes les tâches doivent être réalisées.

L'objectif du problème est de trouver un ensemble de tournées qui minimise la somme des distances parcourues par chaque technicien.

Chaque instance sera intégralement décrite dans un unique fichier texte dont le format est décrit en Section 5.

4 Résolution exacte

L'objectif de ce projet est de mettre en œuvre des méthodes de résolution exactes de ce problème en utilisant un solveur, d'en évaluer les limites et de pousser leur performances le plus loin possible. Il est demandé de modéliser, de mettre en oeuvre numériquement et de tester deux types d'approches.

4.1 Approche frontale

Il s'agit de modéliser directement le problème (dans son ensemble) comme un problème linéaire en nombres entiers (plne).

Question 1. Proposer un modèle *linéaire* en nombres entiers valide pour le problème. On prendra soin de bien expliquer le rôle de chaque variable ainsi que son type, et la signification de chaque contrainte.
Un pré-rapport noté répondant à cette question est à rendre le 28 septembre.

Question 2. Mettre en œuvre ce modèle en utilisant l'api en C++ de cplex, et tenter de résoudre les instances proposées.

4.2 Approche par relaxation lagrangienne

Il s'agit d'utiliser une méthode de relaxation lagrangienne pour améliorer la qualité de la solution.

Question 3. Le principe de la relaxation lagrangienne consiste à relaxer un sous-ensemble de contraintes de manière à former un pseudo-Lagrangien dont la résolution en variables primales est plus aisée. Montrer qu'il existe un ensemble de contraintes qui peut être relaxé de manière à former un pseudo-Lagrangien que l'on peut découpler selon l'ensemble W : autrement dit, minimiser le pseudo-Lagrangien revient à minimiser en x pour chaque technicien w une fonction $l^w(x, \lambda)$ sous contraintes, où les λ sont les multiplicateurs de Lagrange associé aux contraintes relaxées.

Question 4. Écrire le problème de maximisation qui doit être résolu en fonction de λ. Proposer un sur-gradient de la fonction objectif au point λ.

On a maintenant tous les éléments pour résoudre la relaxation lagrangienne en maximisant la fonction duale par un algorithme de sur-gradient.

Question 5. Écrire le détail d'un algorithme de sur-gradient. Proposer une série de pas positifs $(\rho_k)_k$ convergeant vers 0 respectant le critère de la série divergente.

Question 6. Avant d'implémenter numériquement l'algorithme de descente de gradient, tenter de minimiser « à la main » la fonction $d(\lambda_k)$ en testant plusieurs valeurs de λ_k.

Question 7. Implémenter numériquement l'algorithme de descente de gradient à l'aide de l'api de cplex, et trouver une série de pas positifs $(\rho_k)_k$ qui permet d'obtenir des résultats satisfaisants.

Question 8. Expliquer pourquoi l'algorithme mis en oeuvre ne permet pas d'obtenir directement une solution réalisable du problème initial. Proposer une méthode pour obtenir une solution admissible à partir du résultat de l'algorithme obtenu précédemment. On comparera les résultats obtenus par relaxation lagrangienne avec les résultats de la méthode frontale.

Question 9. Comparer la valeur de la borne obtenue par relaxation lagrangienne avec la valeur du relaxé continu du problème initial. Est-ce attendu ? Discuter.

Question 10. Bonus : proposer des idées d'amélioration de cet algorithme.

5 Format des données

Votre programme devra lire les données et écrire les fichiers de solution au format décrit ci-dessous.

Les lignes vides et les lignes commençant par '`#`' (lignes de commentaires) sont ignorées. Les formats sont spécifiés par lignes (terminées par un caractère de fin de ligne '`\n`'), dont les champs sont délimités par des espaces (espace ou tabulation). On pourra utiliser les fonctions `std::getline` et `std::isspace` en C++.

5.1 Fichiers d'instance

Les fichiers d'instances sont au format suivant :

```
#                       name     w    j    s    v
instance                tiny     2    2    3   80

#          id    x    y      Ts     Te    skills
tic    tic_00   84   39     480   1080   1 2
tic    tic_01   19   33     480   1080   1

#          id     x    y    tmin     tmax      s     d    r       p
job    job_000    47   62   660      1079      1    60    1    1000
job    job_001    95   91   840      1019      2    60    1    1000

end
```

La première ligne décrit les données générales de l'instance. Après le mot-clé instance, les champs sont, dans l'ordre :

name	Le nom de l'instance
w	Le nombre de techniciens
j	Le nombre de tâches
s	Le nombre de compétences
v	La vitesse des techniciens

Les lignes suivantes donnent les caractéristiques des techniciens :

id	Le nom du technicien
x, y	Les coordonnées de la base
Ts, Te	Les temps de début et de fin de la journée
skills	La liste des compétences (de 0 à s-1)

Ensuite, les lignes suivantes décrivent les tâches :

id	Le nom de la tâche
x, y	Les coordonnées géographiques
tmin, tmax	L'intervalle admissible de début de la tâche
s	La compétence nécessaire
d	La durée de la tâche
r	L'existence d'un rendez-vous (booléen)
p	La pénalité en cas de non-réalisation (à oublier ici).

Enfin, le fichier se termine par une ligne contenant le mot-clé `end`.

5.2 Fichiers de solution

Les fichiers de solution seront obligatoirement écrit au format suivant :

```
# solution of 'tiny.dat'
# exact solution, from frontal method, in 0s

#         instance    cost
solution  tiny        153

tic_00   job_000   513
tic_00   job_001   980
```

Après le mot-clé `solution`, la première ligne donne successivement le nom de l'instance et le coût de la solution.

Ensuite, chaque ligne contient, dans l'ordre : le nom d'un technicien, le nom d'une tâche et le moment auquel la tâche débute. Attention, les tâches successives d'un même technicien doivent apparaître *dans l'ordre* dans ce fichier.

L'utilisation de ce format vous permettra d'utiliser le programme de validation fourni pour afficher le détail du trajet (ce qui vous sera utile pour développer votre programme) et vérifier que la solution est correcte. Taper '`./gotic_valid -h`' pour obtenir de l'aide sur l'utilisation de cet outil. Attention, *une solution qui ne sera pas validée par l'outil fourni ne sera pas prise en compte dans vos résultats.*