université de BORDEAUX

Collège Sciences et Technologies
UF Mathématiques et Interactions

Technicians and Interventions Scheduling for Telecommunications

Kea Horvath

CMI OPTIM

2022 – 2023

Challenge Algorithmique

Table des matières

Engagement de non plagiat

Je, Kea Horvath, déclare être pleinement conscient que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour produire et écrire ce rapport.

Fait à Talence le 6 janvier 2023

Signature

Kea Horvath

1 Introduction

Tous les deux ans depuis 1999, l'association ROADEF (Société française de recherche opérationnelle et d'aide à la décision) propose un challenge à application industrielle. En 2007, c'est l'opérateur France Télécom qui a soumis un sujet. Le but du sujet est de résoudre un problème de planification d'interventions de techniciens sur des infrastructures de télécommunications. Dans ce rapport, nous tenterons de trouver des solutions satisfaisantes au problème en créant des emplois du temps les plus efficaces possibles pour les jeux de données fournis.

Dans la suite du document, la section 2 présente la formalisation du problème. Dans la section 3, nous modélisons celui-ci comme un programme linéaire en nombre entiers. Nous introduisons un algorithme de résolution du problème dans la section 4. Enfin, nous analyserons les résultats de notre algorithme dans la section 5.

2 Formalisation du problème

Notre but est de créer un emploi du temps efficace pour réaliser toutes les interventions demandées par France Telecom.

Chaque intervention $i \in I = \{1, ..., n\}$ doit être réalisée par des techniciens. Pour ce faire, nous allons placer chaque technicien $t \in T = \{1, ..., m\}$ dans une équipe $g \in G = \{0, ..., m\}$ pour un jour $k \in K = \{1, ..., n\}$ donné (les équipes ne peuvent pas changer au cours d'une journée). Il est possible qu'un technicien ne soit pas toujours disponible pour travailler. On notera $\hat{K}_t \in K$ l'ensemble des jours durant lesquels le technicien $t \in T$ ne peut pas travailler. Les techniciens ne travaillant pas un jour $k \in K$ donné se trouvent dans l'équipe $g = 0$ ce jour-là.

Chaque intervention $i \in I$ requiert des compétences particulières pour être réalisée. Les compétences sont régroupées dans des domaines $d \in D = \{1, ..., q\}$ et chaque technicien $t \in T$ possède un niveau $l \in L = \{0, ..., p\}$ dans chacun des domaines. $s(t, d)$ représente le niveau de maîtrise du domaine de compétences $d \in D$ du technicien $t \in T$. $s'(t, d, l) \in \{0, 1\}$ vaut 1 si le technicien $t \in T$ a un niveau supérieur ou égal à $l \in L$ dans le domaine $d \in D$ et 0 sinon. $r(i, d, l)$ représente le nombre de techniciens de niveau au moins $l \in L$ dans le domaine de compétence $d \in D$ requis pour l'éxecution de l'intervention $i \in I$.

Chaque intervention $i \in I$ a une durée p_i. Chaque jour $k \in K$ comporte 120 unités de temps. La réalisation d'une intervention $i \in I$ ne peut pas chevaucher deux jours. Les interventions qui doivent être réalisées avant l'intervention $i \in I$ sont dans la liste $Pred(i)$.

Les interventions ont aussi la possibilité d'être sous-traitées par un organisme extérieur. Chaque intervention $i \in I$ a un coût de sous-traitance c_i. Le coût des interventions sous-traitées ne peut pas dépasser le budget maximal autorisé pour la sous-traitance B. De plus, si une intervention $i_2 \in Pred(i_1)$ est sous-traitée, alors l'intervention i_1 est obligatoirement sous-traitée aussi. Enfin, si une intervention $i \in I$ est sous-traitée, on considère qu'elle est réalisée à la première unité de temps et que sa durée est nulle.

Pour finir, chaque intervention $i \in I$ a un niveau de priorité $w_i \in \{1, 2, 3\}$. Notre but va être de minimiser une fonction qui dépend des dates de fin des interventions. Plus précisément, il s'agit de minimiser la fonction suivante :

$$28C_1 + 14C_2 + 4C_3 + C$$

C_1, C_2 et C_3 représentent la date de fin la plus tardive des interventions de priorité 1, 2 et 3 (respectivement). C représente la date de fin la plus tardive des interventions (indépendamment de sa priorité).

3 Modélisation du problème

Afin de modéliser le problème sous forme d'un programme linéaire en nombres entiers, nous avons besoin de variables de décision.

On va avoir besoin de quatre variables binaires. On pose $x(t, g, k)$ qui vaut 1 si le technicien $t \in T$ est dans l'équipe $g \in G$ le jour $k \in K$ et 0 sinon. La variable binaire $y(i, g, k)$ vaut 1 si l'équipe $g \in G$ réalise l'intervention $i \in I$ le jour $k \in K$ et 0 sinon. La variable o_i vaut 1 si l'intervention $i \in I$ est sous-traitée et 0 sinon. Enfin, la variable $u_{i,i'}$ vaut 1 si l'intervention $i \in I$ a lieu avant l'intervention $i' \in I$ et 0 si c'est l'inverse.

La variable b_i représente la date de début de l'intervention $i \in I$.

Enfin, les variables qui nous permettent d'évaluer une solution sont C qui représente la date de fin la plus tardive des interventions et les variables C_i qui représentent la date de fin la plus tardive des interventions de priorité $i \in 1, 2, 3$.

Un programme linéaire en nombre entiers pour le problème de "Technicians and Interventions Scheduling for Telecommunications" est le suivant :

$$\min\ 28C_1 + 14C_2 + 4C_3 + C \quad (1)$$

$$sc\ \sum_{g\in G} x(t,g,k) = 1 \qquad \forall t \in T, \forall k \in K \quad (2)$$

$$x(t,0,k) = 1 \qquad \forall t \in T, \forall k \in \hat{K}_t \quad (3)$$

$$y(i,0,k) = 0 \qquad \forall i \in I, \forall k \in K \quad (4)$$

$$\sum_{t\in T} x(t,g,k)s'(t,d,l) \geqslant r(i,d,l) \times y(i,g,k) \qquad \forall i \in I, \forall d \in D, \forall l \in L, \forall g \in G, \forall k \in K \quad (5)$$

$$b_i \geqslant 120 \times \sum_{g\in G}\sum_{k\in K} y(i,g,k)(k-1) \qquad \forall i \in I \quad (6)$$

$$b_i \leqslant 120 \times \sum_{g\in G}\sum_{k\in K} y(i,g,k)k - p_i \times \sum_{g\in G}\sum_{k\in K} y(i,g,k) \qquad \forall i \in I \quad (7)$$

$$b_i + p_i \leqslant b_{i'} + \sum_{i''\in I} p_{i''} \times (1 - u_{i,i'}) \qquad \forall i, i' \in I, i \neq i' \quad (8)$$

$$u_{ii'} + u_{i'i} \geqslant y(i,g,k) + y(i',g,k) - 1 \qquad \forall i, i' \in I, i \neq i', \forall g \in G, \forall k \in K \quad (9)$$

$$\sum_{g\in G}\sum_{k\in K} y(i,g,k) + o_i = 1 \qquad \forall i \in I \quad (10)$$

$$b_j + p_j \leqslant b_i + \sum_{i''\in I} p_{i''} \times o_i \qquad \forall i \in I, \forall j \in Pred(i) \quad (11)$$

$$o_j \leqslant o_i \qquad \forall i \in I, \forall j \in Pred(i) \quad (12)$$

$$\sum_{i\in I} c_i o_i \leqslant B \quad (13)$$

$$C \geqslant b_i + p_i \sum_{g\in G}\sum_{k\in K} y(i,g,k) \qquad \forall i \in I \quad (14)$$

$$C_1 \geqslant b_i + p_i \sum_{g\in G}\sum_{k\in K} y(i,g,k) \qquad \forall i \in I, \forall w_i = 1 \quad (15)$$

$$C_2 \geqslant b_i + p_i \sum_{g\in G}\sum_{k\in K} y(i,g,k) \qquad \forall i \in I, \forall w_i = 2 \quad (16)$$

$$C_3 \geqslant b_i + p_i \sum_{g\in G}\sum_{k\in K} y(i,g,k) \qquad \forall i \in I, \forall w_i = 3 \quad (17)$$

$$x(t,g,k) \in \{0,1\} \qquad \forall t \in T, g \in G, k \in K \quad (18)$$

$$y(i,g,k) \in \{0,1\} \qquad \forall i \in I, g \in G, k \in K \quad (19)$$

$$o_i \in \{0,1\} \qquad \forall i \in I \quad (20)$$

$$u_{ii'} \in \{0,1\} \qquad \forall i, i' \in I \quad (21)$$

$$b_i \geqslant 0 \qquad \forall i \in I \quad (22)$$

L'objectif (1) consiste à minimiser une fonction qui dépend des dates de fin des interventions. La contrainte (2) assure que les équipes ne puissent pas changer au cours d'une journée. La contrainte (3) assure qu'un technicien qui n'est pas disponible ne travaille pas. La contrainte (4) assure que l'équipe 0 ne réalise pas d'intervention. La contrainte (5) vérifie que chaque intervention a assez de techniciens qualifiés. Les contraintes (6) et (7) assurent qu'une intervention ne puisse pas chevaucher deux jours. Les contraintes (8) et (9) vérifient qu'une équipe ne réalise pas plusieurs interventions en même temps. La contrainte (10) vérifie qu'une intervention est soit réalisée, soit sous-traitée. La contrainte (11) vérifie que tous les prédécesseurs de i soient terminés avant de commencer i. La contrainte (12) assure que si i est sous-traitée, alors les successeurs de i sont également sous-traités. La contrainte (13) vérifie que la somme du coût de toutes les sous-traitances ne dépasse pas le budget. Les contraintes (14), (15), (16) et (17) donnent une valeur aux C. Les contraintes (18) (19) (20) (21) (22) définissent le domaine des variables de décision.

4 Algorithmes de résolution

4.1 Première Heuristique

Nous allons réaliser un premier algorithme qui donne une solution réalisable au problème.

Etape 1 Nous commençons par trier les interventions selon un ou plusieurs critères précis(voir section 5.2). Puis nous nous plaçons dans le premier jour.

Etape 2 Nous regardons d'abord si toutes les interventions à réaliser peuvent être sous-traitées. Si c'est le cas, alors la solution est trouvée et on sort de l'algorithme.

Etape 3 Nous prenons la première intervention réalisable de la liste. On modifie d'abord la variable des domaines de cette intervention afin de supprimer toutes les informations inutiles. Par exemple, si dans un domaine, l'intervention requiert 2 techniciens du niveau 1 et 3 techniciens du niveau 2, alors on peut dire que l'intervention requiert 0 techniciens du niveau 1 et 3 du niveau 2.

Etape 4 On va ensuite chercher la "meilleure équipe" qui peut réaliser cette intervention. Pour chaque domaine d et chaque niveau l du domaine, si l'intervention a besoin d'au moins un technicien, nous allons parcourir la liste de tous les techniciens encore disponibles pour cette journée. Si le technicien est qualifié (il a au moins le niveau l dans le domaine d), alors on augmente de 1 sa valeur dans une table de hachage. Ainsi, après avoir parcouru tous les domaines et tous les niveaux, cette table contiendra pour chaque technicien le nombre de fois où il serait utile pour cette intervention. On trouve ensuite le technicien le plus "utile" en prenant tous les techniciens qui ont la valeur maximale dans la table de hachage, puis en prenant, parmi ceux-ci, celui qui a le moins de niveaux inutiles (qui ne servent pas pour cette intervention). On place ensuite ce technicien dans l'équipe (le retirant de la liste des techniciens disponibles) et on met à jour la variable des domaines de l'intervention. On répète ce processus jusqu'à ce que l'intervention soit réalisable par l'équipe (jusqu'à ce que sa variable des domaines ait uniquement des valeurs égales à 0). Enfin, on supprime cette intervention de la liste des interventions à réaliser.

Etape 5 Puis, tant que des interventions sont réalisables le jour donné avec les techniciens encore disponibles, on répète les étapes 3 et 4.

Etape 6 Maintenant que toutes les équipes de cette journée sont faites, on prend chaque équipe une par une et on lui fait faire autant d'interventions possibles.

Etape 7 Enfin, on passe au jour suivant et on repasse à l'étape 2 jusqu'à ce que toutes les interventions soient réalisées et/ou sous-traitées.

4.2 Recherche locale

Afin d'améliorer les résultats obtenus avec notre heuristique, on peut envisager la réalisation d'une recherche locale. Pour ce faire, je conçois deux méthodes possibles pour trouver un voisinage.
La première consisterait à effectuer des permutations d'interventions. Puis un mouvement serait accepté que si les contraintes sont toujours respectées, si les valeurs des différents C (C1, C2, C3 et C) ne changent pas et si les deux interventions sont réalisables par leurs nouvelles équipes. On pourrait ensuite essayer de voir si un des techniciens des équipes pourrait être retiré. Cela permettrait à ce technicien d'aller dans une nouvelle équipe et de peut-être réaliser d'autres interventions qui sont pour le moment réalisées plus tard. Afin d'évaluer les solutions voisines, et ainsi décider quel voisin choisir, on pourrait essayer de voir si l'échange des interventions entraîne la libération d'un technicien. Si c'est le cas, alors ce voisin est accepté et on essaie de faire faire des interventions qui ont pour l'instant lieu plus tard par ce nouveau technicien libre.
La deuxième méthode serait, d'un jour donné, échanger deux techniciens qui sont dans des équipes

différentes, ou de supprimer un technicien d'une équipe. Cette méthode me parait cependant plus complexe, puisque ces modifications des équipes provoqueraient beaucoup de changements et les voisins seraient donc beaucoup durs à évaluer puisque très différents de la solution de départ.
J'ai essayé de mettre en oeuvre ces méthodes, mais je n'ai pas réussi à trouver comment convertir mes idées en code. Les expérimentations de la prochaine section seront donc uniquement basées sur la première heuristique.

5 Expérimentations et résultats

5.1 Implémentation du programme linéaire en nombres entiers

Nous avons commencé par implémenter un solveur du programme linéaire en nombres entiers avec l'aide de Gurobi Optimizer. La Table 1 présente un résumé des résultats de nos expérimentations.

TABLE 1 – Meilleures solutions trouvées par gurobi optimizer en 600s

dataSetA	MSC	Obj	Gap	Temps(s)
data1	2340	2340	0%	0.021
data2	4755	4755	0%	0.036
data3	11880	11880	6.57%	600
data4	13452	13452	0%	340
data5	28845	-	-	600

MSC - Meilleure solution connue

Notre implémentation semble juste, puisqu'on obtient les mêmes résultats que les meilleures solutions connues pour les premiers jeux de données. En revanche, on peut voir que trouver la solution optimale peut mettre très longtemps (le solveur n'a pas trouvé de solution réalisable pour le jeu de données data5 en 600s par exemple). C'est pour cela que l'on a implémenté une heuristique, qui ne donnera peut-être pas la solution optimale, mais qui sera bien plus rapide.

5.2 Heuristique

Nous effectuons les expérimentations avec les instances du datasetA et du datasetB. La Table 2 présente un résumé des résultats de l'expérimentation de notre heuristique avec 7 tris différents. Pour chacun des tris, nous commençons par trier les interventions par priorité croissante. Puis selon la version, nous ajoutons un ou plusieurs tris dans l'ordre croissant ou décroissant parmi les trois critères suivants : la durée, le nombre minimum de techniciens requis pour l'intervention et le nombre de niveaux demandé par l'intervention. Les expérimentations vont nous permettre de déterminer si les tris sont utiles, et si un tri sort du lot.

TABLE 2 – Solutions trouvées par l'algorithme selon le tri effectué

Instance	MSC	V0	V1	V2	V3	V4	V5	V6	V7	MST
A-data1	2340	6510	2700	3690	3690	2700	3690	2550	3690	2550
A-data2	4755	4755	4755	4755	4755	4755	5940	5940	4755	4755
A-data3	11880	18420	17040	21360	16440	21960	15540	15540	16440	15540
A-data4	13452	23460	18420	13620	14040	24060	17952	16740	15480	13620
A-data5	28845	46830	40380	46605	36945	51660	37365	40260	39930	36945
A-data6	18795	34710	27570	27795	28215	23220	29250	27570	29475	23220
A-data7	30540	55380	56340	56340	35250	56340	35640	33360	35250	33360
A-data8	16920	32895	29835	26504	25800	34560	27540	26700	24720	24720
A-data9	27348	43515	33960	33855	36350	40680	34380	33780	34200	33780
A-data10	38296	76305	50820	49440	48300	55920	49140	45240	49970	45240
B-data1	33900	139350	63570	66600	65040	68400	59640	60960	56550	56550
B-data2	15870	54720	29460	29385	33615	40440	28620	28530	27645	27645
B-data3	16005	66000	31080	31320	34680	39840	30960	30960	27090	27090
B-data4	23775	90195	80685	70185	68820	87510	70590	67140	79470	67140
B-data5	88680	237000	143640	144600	143040	144960	145560	148080	143280	143040
B-data6	26955	73425	48420	45960	45840	54180	45240	45810	41955	41955
B-data7	31620	74520	45240	48240	51900	52200	47640	47640	41040	41040
B-data8	33030	103320	42600	44280	43440	42600	41520	43320	41040	41040
B-data9	28080	44640	41760	44040	41760	41760	36720	40080	38400	36720
B-data10	34680	61560	56400	58680	58680	55320	53040	49680	49680	49680
Gap avec MSC	0 %	51%	34%	35%	32%	39%	32%	31%	30%	25%
Ecart type		18%	15%	16%	16%	17%	13%	14%	14%	16%

MSC - Meilleure solution connue
MST - Meilleure solution tris
V0 - pas de tri
V1 - priorités croissantes
V2 - priorités croissantes -> durées décroissantes
V3 - priorités croissantes -> durées décroissantes -> nombre de techniciens minimum décroissant
V4 - priorités croissantes -> nombre de techniciens minimum croissant
V5 - priorités croissantes -> nombre de techniciens minimum décroissant
V6 - priorités croissantes -> nombre de niveaux décroissant
V7 - priorités croissantes -> durées décroissantes -> nombre de niveaux décroissant -> nombre de techniciens minimum décroissant

On peut voir que dans l'ensemble, il s'agit de la version 7 du tri qui donne les meilleurs résultats, puisqu'on obtient une différence d'écart avec les meilleures solutions connues (BKS) de 30% en moyenne et un écart type assez faible. De plus, cette version de tri donne le meilleur résultat dans 9 cas sur 20. On constate également que réaliser des tris est bénéfique puisque la version 0 (sans tri) produit dans l'ensemble de bien moins bons résultats (51% d'écart).
Cependant, afin de trouver la meilleure solution pour chaque jeu de données, on doit prendre les résultats de 6 versions différentes, tris parfois opposés (exemple : versions 4 et 5). On peut donc se demander si un tri aléatoire pourrait aussi améliorer nos résultats.
Nous allons donc répéter cette expérimentation en réalisant un grand nombre d'itérations de tri aléatoire et en gardant la meilleure solution (10000 itérations pour le datasetA et 1000 pour le datasetB car ce dernier met beaucoup plus de temps à être résolu). Puisque nous avons vu que les tris paraissent bénéfiques, nous effectuerons ensuite la même démarche en effectuant une version des tris vus précédemment après avoir effectué le tri aléatoire. Nous prendrons les versions 1 (priorités croissantes) et 7 (priorités croissantes -> durées décroissantes -> nombre de niveaux décroissant -> nombre de techniciens minimum décroissant) qui nous a donné les meilleurs résultats dans l'ensemble.

TABLE 3 – Solutions trouvées par l'algorithme selon le tri effectué après 1000 ou 10000 itérations

Instance	MSC	V0+	V1+	V7+	MST+	MS	Gap avec MST
A-data1	2340	2340	2550	3330	2340	2340	0%
A-data2	4755	4755	4755	4755	4755	4755	0%
A-data3	11880	13440	13560	16440	13440	13440	12%
A-data4	13452	14292	13620	15480	13620	13620	1%
A-data5	28845	33480	33480	37140	33480	33480	14%
A-data6	18795	23835	24015	29475	23835	23220	19%
A-data7	30540	32730	32520	32520	32520	32520	6%
A-data8	16920	22140	21900	24750	21900	21900	23%
A-data9	27348	30840	30840	34275	30840	30840	11%
A-data10	38296	42630	42720	48300	42630	42630	10%
B-data1	33900	116040	57000	55890	55890	55890	39%
B-data2	15870	44670	24660	24600	24600	24600	35%
B-data3	16005	57210	26820	25260	25260	25260	37%
B-data4	23775	67545	55170	75675	55170	55170	57%
B-data5	88680	176400	124320	136080	124320	124320	29%
B-data6	26955	68025	40470	39090	39090	39090	31%
B-data7	31620	62760	39960	40200	39960	39960	21%
B-data8	33030	97200	36360	40920	36360	36360	9%
B-data9	28080	36120	35040	38400	35040	35040	20%
B-data10	34680	50280	46320	49680	46320	46320	25%
Gap avec MSC	0 %	33%	21%	28%	20%	20%	
Ecart type		26%	15%	14%	15%	15%	

MS - Meilleure solution trouvée après toutes nos expérimentations

On voit qu'en effectuant un grand nombre d'itérations (plus ou moins) aléatoires, on améliore la solution pour presque tous les jeux de données. On est passés d'un gap moyen de 25% à un gap de 20%.
On peut remarquer la version 7 du tri, qui lors d'une seule itération donnait très souvent la meilleure solution, est ici moins intéressante. En effet, les quatre tris successifs sont sûrement trop restrictifs et ne permettent donc pas de parcourir un grand nombre de solutions. C'est la version 1 du tri (priorités croissantes) qui donne, et de loin, les meilleures solutions sur un grand nombre d'itérations (21% de gap). Cela parait cohérent puisque c'est un tri qui est indispensable pour réduire le coût total puisque notre but est réaliser les interventions de haute priorité le plus tôt possible. Mais contrairement à la version 7, ce tri n'est pas trop restrictifs et permet donc d'accéder à un nombre de solutions beaucoup plus important, ce qui explique le résultat obtenu.
Nos résultats sont très satisfaisants pour les petits jeux de données (notamment dans le dataset A). On obtient la solution optimale pour deux instances et un gap avec la meilleure solution trouvée de moins de 15% dans 9 instances sur 20. On obtient ainsi un gap de 20% en moyenne. En revanche, la différence avec les meilleures solutions connues reste très importante pour des instances avec beaucoup d'interventions. Cinq de nos meilleures solutions ont un gap de plus de 30% (on note en particulier le gap de 57% pour B-data4).
Vu le nombre de tris et d'itérations effectuées, il semble que cet algorithme ait atteint sa limite et qu'il ne puisse pas trouver de solutions significativement meilleures que celles déjà obtenues.

6 Conclusion

Ce projet m'a permis de mettre en pratique mes connaissances en optimisation mathématique en formalisant le problème proposé par France Télécom sous forme d'un programme linéaire en nombres entiers. J'ai ensuite pu consolider mes compétences de programmation Java en codant dans une première partie le programme linéaire à l'aide de Gurobi Optimizer, puis un algorithme de résolution.

Dans l'ensemble, les résultats obtenus semblent satisfaisants. En effet, on obtient un gap avec les meilleures solutions connues de 20% en moyenne. Il est tout de même clair que de nettes améliorations sont possibles, notamment pour les grandes instances, puisqu'on obtient par exemple un gap de 57% pour une des instances.

Une suite possible à ce projet serait d'essayer d'améliorer nos résultats en implémentant la recherche locale (4.2) puisque je n'ai pas encore réussi à le faire.