HAL
open science

Un algorithme GRASP pour le problème de planification de techniciens et d'interventions pour les télécommunications

Sylvain Boussier, Hashimoto Hideki, Michel Vasquez, Christophe Wilbaut

To cite this version:

Sylvain Boussier, Hashimoto Hideki, Michel Vasquez, Christophe Wilbaut. Un algorithme GRASP pour le problème de planification de techniciens et d'interventions pour les télécommunications. RAIRO - Operations Research, 2009, 43 (4), pp.387-407. 10.1051/ro/2009027 . hal-00842407

HAL Id: hal-00842407

https://hal.science/hal-00842407v1

Submitted on 7 Aug 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Un algorithme GRASP pour le problème de planification de techniciens et d'interventions pour les télécommunications

Sylvain Boussier Hideki Hashimoto Michel Vasquez
Christophe Wilbaut

Résumé

Le problème de planification de techniciens et d'interventions pour les télécommunications (TIST pour *Technicians and Interventions Scheduling Problem for Telecommunications*) comprend la planification d'interventions et l'affectation d'équipes de techniciens à ces interventions. Chaque intervention est caractérisée, entre autres, par une priorité. L'objectif de ce problème est de séquencer les interventions en tenant compte de leur priorité tout en satisfaisant un ensemble de contraintes comme l'ordre d'exécution de certaines interventions et le nombre minimum de techniciens d'un niveau de compétence donné à affecter à chaque intervention. La résolution de ce problème est centrée sur un algorithme GRASP (*Greedy Randomized Adaptive Search Procedure*) caractérisé par une mise à jour dynamique des critères de choix des interventions. Pour évaluer la qualité des résultats obtenus par cette approche heuristique, nous présentons également un calcul de bornes inférieures.

1 Introduction

Dans cet article, nous décrivons une approche heuristique pour résoudre un problème de planification de techniciens et d'interventions pour les télécommunications que nous abrégeons par TIST pour *Technicians and Interventions Scheduling Problem for Telecommunications*. Ce problème [3] a été proposé par *France Telecom* pour le 5eme challenge de la Société Française de Recherche Opérationnelle et d'aide à la décision (ROADEF) [1].

Les interventions sont caractérisées par des critères spécifiques comme une priorité de planification et une durée d'exécution. Certaines interventions doivent être réalisées avant d'autres, ce qui constitue un ensemble de contraintes de précédence qui doivent être satisfaites par l'ordonnancement. Les interventions sont également composées de différentes tâches qui nécessitent un certain nombre de techniciens d'un certain niveau de compétence dans un domaine donné. Les techniciens sont spécialisés dans différents domaines avec différents niveaux de com-

1. `http://www.g-scop.fr/ChallengeROADEF2007/` ou `http://www.roadef.org/`

pétence et chaque technicien a une liste de jours d'indisponibilités. Par ailleurs, chaque intervention a un coût donné si elle est sous-traitée par une entreprise extérieure. Le coût total de ces interventions sous-traitées ne peut dépasser un certain budget. Notons qu'une intervention ne peut être sous-traitée que si ses successeurs, selon les contraintes de précédence, ont également été sous-traités. L'objectif du TIST est d'ordonnancer un ensemble d'interventions en minimisant une fonction scalaire qui attribue une pénalité plus grande aux interventions les plus prioritaires. Ce problème a deux aspects combinatoires distincts : l'ordonnancement des interventions (qui dépend des contraintes de précédence) et la construction des équipes de techniciens (qui dépend du jour courant).

Le coeur de notre approche est constitué d'un algorithme GRASP (pour *Greedy Randomized Adaptive Search Procedure*). Les critères de sélection des interventions pour l'algorithme glouton sont mis à jour durant la recherche : les nouvelles solutions sont construites en utilisant l'information fournie par les solutions précédentes. Nous essayons ensuite d'améliorer ces solutions avec un algorithme de recherche locale. Globalement, notre approche est divisée en trois phases : premièrement, une phase de prétraitement qui sélectionne les interventions à sous-traiter ; deuxièmement, une phase d'identification des coefficients initiaux pour l'algorithme glouton qui cherche les deux meilleurs ordres d'insertion des interventions ; troisièmement, une phase de recherche de solutions par la procédure GRASP qui est composée de la *phase gloutonne* qui génère des solutions initiales suivie de la *phase de recherche locale* qui tente d'améliorer ces solutions.

Le papier est organisé de la manière suivante : dans la section 2, nous décrivons le problème et introduisons les différentes notations ; la section 3 est consacrée à la présentation des différents composants de notre approche ; puis, dans la section 4, nous exposons un calcul de borne inférieure implémenté pour évaluer la qualité des résultats obtenus. Les résultats expérimentaux sont donnés en section 5.

2 Description du problème

Dans cette section, nous commençons par décrire le problème de manière informelle, puis nous présentons les différentes notations utilisées pour les données du problème. Nous concluons par une formulation mathématique du TIST dans laquelle les interventions sous-traitées ne sont pas prises en compte.

2.1 Description générale

Le problème traite d'interventions qui doivent être affectées à des équipes de techniciens. Les techniciens sont caractérisés par leurs jours de congés et leurs niveaux de compétence, et les interventions par leur priorité, leur temps d'exécution, leur liste de prédécesseurs (interventions qui doivent être traitées avant l'intervention) et par le nombre de techniciens requis pour chaque niveau et domaine de compétence. L'objectif est de construire des équipes de techniciens

pour chaque journée et d'affecter des interventions à ces équipes en respectant toutes les contraintes de la planification et en minimisant la fonction objective

$$28t_1 + 14t_2 + 4t_3 + t_4$$

où t_k est la date de fin de la dernière intervention de priorité k pour $k = 1, 2, 3$ et t_4 est la date de fin de l'ordonnancement.

Un ordonnancement doit satisfaire une liste de contraintes pour l'affectation des techniciens et pour l'affectation des interventions. Nous considérons que chaque journée de travail est dans l'intervalle de temps $[0, H_{\max}]$ et qu'il est impératif de respecter cet intervalle. En conséquence, une intervention ne peut être réalisée avant la date 0 ou après la date $H_{\max}$ et ne peut être réalisée sur plusieurs jours.

Une intervention doit être réalisée par une seule équipe à une unique date, plusieurs équipes ne peuvent se partager la même intervention. Il y a des jours de congés pour les techniciens et aucune intervention ne peut être affectée à un technicien en repos. Une contrainte forte est que les équipes ne peuvent changer durant la journée, ce qui implique que chaque technicien appartient à une seule équipe chaque jour. Cette contrainte est due au nombre limité de voitures disponibles et au temps que cela prendrait de rapatrier les voitures afin de former de nouvelles équipes.

Une équipe doit satisfaire les demandes de chaque intervention. Ainsi, pour chaque intervention, nous devons affecter suffisamment de techniciens qualifiés pour satisfaire toutes les demandes. Par exemple, une intervention qui nécessite un technicien de niveau 2 dans le domaine d1 peut être réalisée par un technicien de niveau 2, 3 ou 4 dans le domaine d1, mais ne peut être réalisée par deux techniciens de niveau 1 dans le domaine d1. Le nombre requis de techniciens d'un certain niveau pour une intervention est cumulable puisqu'un technicien d'un niveau donné est aussi qualifié pour tous les niveaux inférieurs pour le même domaine de compétence. Par exemple, si un technicien a un niveau de compétence de 3 dans un domaine donné, il peut également travailler sur des interventions qui nécessitent seulement un niveau de 2 dans ce domaine.

Finalement, il est possible de sous-traiter certaines interventions à une entreprise externe. Chaque intervention a un coût spécifique dans le cas où elle serait sous-traitée et le coût total de la sous-traitance ne peut excéder un budget donné. Notons que le modèle mathématique que nous exposons en section 2.3 ne prend pas en compte les interventions sous-traitées. En effet, cette partie du problème est réalisée dans une pré-procédure qui est décrite en section 3.1.

2.2 Notations

Dans cette section, nous introduisons un ensemble de notations utilisées. Nous introduisons premièrement les constantes du problème :

- $H_{\max}$ est la durée de temps de chaque jour ($H_{\max} = 120$ dans le sujet).
- $T(I)$ est le temps d'exécution de l'intervention I.
- $cost(I)$ est le coût de l'intervention I.

- A est le budget alloué pour les interventions sous-traitées.
- $P(t, j)$ est égale à 1 si le technicien t travaille le jour j, 0 sinon.
- $C(t, i)$ est le niveau de compétence du technicien t dans le domaine i.
- $R(I, i, n)$ est le nombre de techniciens requis de niveau n dans le domaine i pour traiter l'intervention I.
- $Pred(I)$ est la liste des interventions qui doivent être terminées avant de commencer l'intervention I.

Nous utilisons également les variables listées ci-dessous :
- $s(I)$ est la date de début de l'intervention I.
- $e(t, j)$ est le numéro de l'équipe à laquelle est rattaché le technicien t pour le jour j. L'équipe 0 est une équipe spécifique composée des techniciens ne travaillant pas ce jour.
- $d(I)$ est le jour où l'intervention I est planifiée.

Une intervention qui nécessite, pour le domaine i, au moins un technicien de niveau trois et un technicien de niveau deux aura ses demandes notées : $R(I, i, 1) = 2$, $R(I, i, 2) = 2$, $R(I, i, 3) = 1$, $R(I, i, 4) = 0$.

Les constantes utilisées pour le modèle mathématique sont les suivantes :
- $Pr(k, I)$ est égale à 1 si la priorité de l'intervention I est k et 0 sinon.
- $\mathcal{P}(I_1, I_2)$ est égal à 1 si l'intervention I_1 est un prédécesseur de l'intervention I_2 et 0 sinon.

Nous utilisons également les variables suivantes dans le modèle mathématique :
- $x(I, j, h, \epsilon)$ est égale à 1 si l'équipe ϵ travaille sur l'intervention I le jour j à la date de début h et 0 sinon.
- $y(j, \epsilon, t)$ est égale à 1 si le technicien t est dans l'équipe ϵ le jour j et 0 sinon.
- t_k, $k = 1, 2, 3$ est la date de fin de la dernière intervention de priorité k.
- t_4 est la date de fin de l'ordonnancement.

2.3 Modèle mathématique

Comme nous l'avons mentionné dans la section 2.1, le modèle mathématique suivant ne considère pas les interventions sous-traitées. Ce problème, noté (P'),

peut être modélisé de la manière suivante :

$$\text{Minimiser} \quad 28t_1 + 14t_2 + 4t_3 + t_4$$

$$\text{sujet à} \sum_{j,h,\epsilon} x(I,j,h,\epsilon) = 1 \qquad \forall I \tag{1}$$

$$y(j,0,t) = 1 - P(t,j) \qquad \forall j,t \tag{2}$$

$$\sum_{\epsilon} y(j,\epsilon,t) = 1 \qquad \forall j,t \tag{3}$$

$$x(I,j,h,0) = 0 \qquad \forall I,j,h \tag{4}$$

$$\sum_{h_1=\max(h_2-T(I_1)+1,0)}^{\min(h_2+T(I_2)-1,H_{\max})} x(I_1,j,h_1,\epsilon) + x(I_2,j,h_2,\epsilon) \leq 1 \qquad \forall I_1,I_2,h_2,j,\epsilon \tag{5}$$

$$\sum_{j,h,\epsilon} (jH_{\max} + h)\,(x(I_1,j,h,\epsilon) - x(I_2,j,h,\epsilon)) + T(I_1)x(I_1,j,h,\epsilon) \leq 0$$

$$\forall I_1,I_2 \mid \mathcal{P}(I_1,I_2) = 1 \tag{6}$$

$$x(I,j,h,\epsilon) = 0 \qquad \forall I,j,h,\epsilon \mid h + T(I) > H_{\max} \tag{7}$$

$$\sum_{h} R(I,i,n)x(I,j,h,\epsilon) \leq \sum_{t|C(t,i)\geq n} y(j,\epsilon,t) \qquad \forall I,i,n,\epsilon,j \tag{8}$$

$$\sum_{j,h,\epsilon} (jH_{\max} + h + T(I))\,Pr(k,I)x(I,j,h,\epsilon) \leq t_k \qquad \forall I, k = 1,2,3 \tag{9}$$

$$\sum_{j,h,\epsilon} (jH_{\max} + h + T(I))\,x(I,j,h,\epsilon) \leq t_4 \qquad \forall I \tag{10}$$

La contrainte (1) assure que chaque intervention est traitée par une seule équipe à une journée et à une date fixée. La contrainte (2) garantit que si un technicien t ne travaille pas le jour j, il est dans l'équipe 0. La contrainte (3) spécifie qu'un technicien appartient à une seule équipe chaque jour. La contrainte (4) assure qu'aucune intervention n'est réalisée par l'équipe 0. La contrainte (5) certifie que deux interventions réalisées le même jour par la même équipe sont effectuées à des dates différentes. La contrainte (6) spécifie que tous les prédécesseurs d'une intervention donnée doivent être réalisés avant la date de début de l'intervention. La contrainte (7) assure que chaque jour a une durée limite de $H_{\max}$, nombre maximal de portions de temps par jour. La contrainte (8) spécifie qu'une équipe qui travaille sur l'intervention I satisfait les demandes en nombre de techniciens par niveau de compétence. Finalement, la contrainte (9) spécifie que t_k est la date de fin de la dernière intervention de priorité k, $k = 1,2,3$ et la contrainte (10) spécifie que t_4 est la date de fin de l'ordonnancement.

3 Approche générale

L'algorithme proposé est centré sur une approche GRASP qui est une méthode à mémoire adaptative basée sur la génération de solutions de manière

gloutonne. Dans cette section nous détaillons le rôle ainsi que les principes de chaque partie de l'approche.

Une première partie (sect. 3.1) est consacrée à l'identification heuristique des interventions sous-traitées. Ces interventions sont supprimées une fois pour toute du problème et l'algorithme GRASP décrit en section 3.2 est alors appliqué au problème P' (cf. sect. 2.3).

Une des phases principales de la métaheuristique GRASP est l'algorithme glouton mis en oeuvre pour générer des solutions réalisables. Comme nous l'expliquons en section 3.2.1, cet algorithme est également utilisé pour mettre à jour dynamiquement les critères de choix des interventions. Ces critères, ou poids, associés à chaque intervention constituent une mémoire. L'initialisation de cette mémoire est décrite en section 3.2.2. La méthode GRASP est également constituée d'une phase de recherche locale qui est décrite en détails dans la section 3.2.3. Nous concluons cette section par un schéma d'ensemble de notre approche (sect. 3.3).

3.1 Choix des interventions à sous-traiter

L'identification des interventions à sous-traiter se fonde sur un critère heuristique, ou poids, attribué à chaque intervention. Ce poids ω_I est établi à partir d'une borne du nombre minimum de techniciens nécessaires à la réalisation de l'intervention I ($mintec(I)$), et de la durée de celle-ci ($T(I)$). Il est égal au produit de ces deux valeurs : $\omega_I = mintec(I) \times T(I)$.

Soit Ω_t l'ensemble des indices des variables représentant les techniciens et $x \in \{0,1\}^{|\Omega_t|}$ un vecteur de variables de décision. Dans un premier temps, la valeur $mintec(I)$ est calculée en résolvant le programme linéaire associé à I suivant :

$$\begin{cases} \text{Minimiser } \sum_{t \in \Omega_t} x_t \text{ sujet à,} \\ \sum_{t/C(t,I) \geq n, t \in \Omega_t} x_t \geq R(I,i,n) \qquad \forall i, n, \\ x_t \in \{0,1\} \qquad t \in \Omega_t \end{cases}$$

Soit Ω_I l'ensemble des indices des interventions et $x \in \{0,1\}^{|\Omega_I|}$ le vecteur de variables de décision tel que $x_I = 1$ si l'intervention I est sous-traitée et 0 sinon. Nous devons trouver, ensuite, un sous-ensemble d'interventions Γ à sous-traiter tel que $\sum_{x_I \in \Gamma} w_I x_I$ soit maximal et le coût total ne dépasse pas le budget total disponible, A. Nous avons donc affaire avec un problème de sac à dos avec contraintes de précédence [7]. Soit PCKP ce problème (pour *Precedence Constraint Knapsack Problem*) et $S(x_I, x_{I'}) = 1$ si l'intervention I' doit débuter après la fin de l'intervention I et 0 sinon. Le PCKP peut être formulé de la manière suivante :

$$PCKP \begin{cases} \text{Maximiser } \sum_{I \in \Omega_I} w_I x_I \text{ sujet à,} \\ \sum_{I \in \Omega_I} cost(I) \cdot x_I \leq A, \\ x_I \leq x_{I'} \qquad \forall I, I' \in \Omega_I \mid S(x_I, x_{I'}) = 1 \\ x_I \in \{0,1\} \qquad I \in \Omega_I \end{cases}$$

Ce problème est résolu par un algorithme glouton qui sélectionne les interventions de ratio $\omega_I/cost(I)$ maximal, pour lesquelles tous les successeurs sont sous-traités, tant que le coût total ne dépasse pas le budget disponible.

Cette choix de gestion des interventions à sous-traitées a été réalisé dans un contexte de challenge et n'est probablement pas optimal. Toutefois, l'expérimentation a montré qu'il fournit de meilleurs résultats que d'autres stratégies testées telles que :

- Fixer le maximum de variables pour réduire le problème. Ce qui correspond à affecter un poids $w_I = 1$ pour toutes les interventions.
- Prendre en compte la priorité des interventions, c'est-à-dire, fixer $w_I = 28$ si I est de priorité 1, $w_I = 14$ si I est de priorité 2 etc.

3.2 Phase GRASP

Lorsque les interventions sous-traitées ont été supprimées du problème original, le sous-problème résiduel composé des interventions restantes est résolu par un algorithme de type GRASP [4].

La terminologie GRASP se réfère à une classe de procédures dans laquelle une heuristique gloutonne randomisée ainsi qu'une technique de recherche locale sont employées. La métaheuristique GRASP a été appliquée à de nombreux problèmes d'optimisation combinatoire comme les problèmes d'ordonnancement [12], les problèmes de routage [1], les problèmes de graphes [9], les problèmes d'affectation [6], etc. Le lecteur peut se référer par exemple à [10] ou [5] pour une bibliographie plus complète.

Une implémentation classique de la méthode GRASP consiste généralement à répéter la procédure suivante jusqu'à satisfaire un critère d'arrêt (nombre maximum d'itérations, temps CPU fixé, niveau de qualité de la solution...) :

- Générer une solution réalisable avec un algorithme glouton randomisé.
- Appliquer un algorithme de recherche locale à la solution précédente.
- Mettre à jour la meilleure solution.

La méthode GRASP a été utilisée pour résoudre un problème dans un contexte similaire [12]. Dans ce papier, les auteurs développent un algorithme glouton, un algorithme de recherche locale et un algorithme GRASP pour résoudre un problème de planification de techniciens. Ils ont montré que la méthode GRASP fournit les meilleurs résultats en dépit d'un accroissement important du temps d'exécution. Ils ont implémenté une version parallèle de leur algorithme pour remédier à ce problème. Cette approche n'était pas applicable dans le contexte de notre challenge.

Dans la section qui suit, nous présentons l'algorithme glouton utilisé pour générer une solution réalisable. Nous expliquons également comment la mémoire est mise à jour durant la recherche.

3.2.1 Utilisation de la méthode GRASP pour construire une solution réalisable

Nous rappelons que l'algorithme glouton, décrit ici, ne considère que les interventions non sous-traitées que nous appelons les *candidats*.

Sélection d'un candidat

Initialement, le poids (critère de choix) d'un candidat est fixé à la valeur du coefficient de sa priorité dans la fonction objectif et nous fixons arbitrairement un poids d'une valeur de 1 pour les interventions de priorité 4. Ainsi, les candidats de priorité 1 (respectivement 2, 3) ont un poids de 28 (resp. 14 ,4) et les candidats de priorité 4 ont un poids de 1. L'algorithme glouton sélectionne le candidat qui a la plus haute valeur de poids. Lorsque deux candidats ont le même poids, le choix est fait aléatoirement. Par ailleurs un candidat ne peut être ordonnancé si au moins l'un de ses prédécesseurs ne l'est pas.

L'algorithme glouton cherche à insérer un candidat selon les trois critères suivants : (1) Le jour le plus tôt ; (2a) l'équipe qui nécessite le moins de techniciens supplémentaires pour traiter l'intervention ; (2b) la date de début au plus tôt. Le processus est répété jusqu'à ce que tous les candidats soient ordonnancés. Nous décrivons maintenant comment l'algorithme tient compte de ces trois critères.

Recherche du jour au plus tôt

La première étape consiste à calculer la date de départ au plus tôt de I. Elle correspond à un couple de valeurs *jour* et *heure* notées $d(I)$ et $s(I)$. Pour cela, nous cherchons la date de fin au plus tard parmi tous les prédécesseurs de I, $s(I_{max}) + T(I_{max})$ telle que $I_{max} \in Pred(I)$ et I_{max} est séquencée le jour d_{max}. Si $s(I_{max}) + T(I_{max}) + T(I) > H_{\max}$ alors $d(I) = d_{max} + 1$ sinon $d(I) = d_{max}$. Dans tous les cas $s(I) = s(I_{max}) + T(I_{max})$.

Calcul du nombre minimum de techniciens requis pour un candidat

Les critères (2a) et (2b) dépendent des techniciens disponibles et des équipes existantes le jour $d(I)$. Afin de respecter le critère (2a), le nombre de techniciens nécessaires pour la construction d'une nouvelle équipe à laquelle est affectée I doit être connu. L'algorithme vérifie que les niveaux de compétence requis par l'intervention I sont satisfaits par une équipe donnée ϵ. Si c'est le cas, il n'est pas nécessaire d'ajouter un technicien à l'équipe ϵ. Dans le cas contraire, le nombre minimum de techniciens à ajouter à ϵ est déterminé de manière heuristique à partir des techniciens disponibles le jour $d(I)$. Nous noterons $techs^{\epsilon}(I)$ le nombre de techniciens nécessaires pour assigner I à l'équipe ϵ ($\epsilon = 0$ si une nouvelle équipe doit être créée). Il reste à calculer la date de début au plus tôt de I pour ces équipes.

Calcul de la date de début au plus tôt

L'étape suivante de l'algorithme consiste à calculer la date au plus tôt $s^\epsilon(I)$ de planification de l'intervention I avec l'équipe ϵ. Dans le cas d'une nouvelle équipe il s'agit simplement de la valeur $s(I)$. Sinon, pour chacune des équipes retenues, on essaie d'insérer I au plus tôt dans leur planning.

Choix entre (2a) et (2b)

L'ordre des critères (2a) et (2b) dépend de la valeur de $d(I)$ obtenue précédemment. Il dépend également de la date de fin de la dernière intervention de même priorité que I pour les priorités 1, 2, 3 et de la date de fin totale pour la priorité 4 (i.e. t_i, où i désigne la priorité du candidat I). Si $d(I) \times H_{\max} + s(I) + T(I) < t_i$ alors la condition (2a) est considérée avant la condition (2b). Sinon, la condition (2b) a la priorité. En effet, si l'insertion de I ne provoque pas d'augmentation la date de fin de la dernière intervention de la même priorité que I alors elle n'influe pas sur la valeur de la fonction objectif. Dans ce cas nous minimisons le nombre de techniciens à ajouter avant de minimiser la valeur de $s(I)$.

En résumé, pour favoriser la condition (2a), l'algorithme choisit l'équipe ϵ avec la valeur minimale $techs^\epsilon(I)$ correspondante. Si plusieurs équipes ont la même valeur, l'équipe choisie est celle pour laquelle la valeur $s^\epsilon(I)$ est minimale. Pour favoriser la condition (2b), l'algorithme choisit l'équipe ϵ avec la valeur minimale $s^\epsilon(I)$ et celle avec la valeur minimale $techs^\epsilon(I)$ s'il exsite au moins deux équipes avec la même valeur $s^\epsilon(I)$.

Utilisation d'une permutation des poids dans l'algorithme glouton

Nous montrons dans cette section que le critère de choix des interventions pour l'algorithme glouton ne correspond pas toujours à l'ordre des coefficients des priorités dans la fonction objectif.

Notons $w(I)$ le poids de l'intervention I. Supposons que $w(I)$ soit égal au coefficient de la priorité de I dans la fonction objectif. Cela revient à choisir les interventions de haute priorité d'abord. La figure 1 représente une solution générée par l'algorithme glouton dans ces conditions. Les poids des interventions sont donc : 28 pour les interventions de priorité 1, 14 pour les interventions de priorité 2, 4 pour les interventions de priorité 3 et 1 pour celles de priorité 4. Dans cette figure, chaque ligne représente un technicien : le premier technicien est représenté par la ligne du haut et le dernier technicien par la ligne du bas. Chaque rectangle noir correspond à un jour de congés et chaque ligne verticale correspond à la fin d'une journée. La valeur de la fonction objective de cette solution est 17820.

Il est possible d'affecter les poids aux interventions de manière différente. Supposons que les interventions de priorité 4 aient un poids de 28, celles de priorité 3 un poids de 14, celles de priorité 1 un poids de 4 et celles de priorité 2 un poids de 1. Cette affectation correspond à utiliser la permutation (4,3,1,2) des poids. La figure 2 donne une solution générée avec l'algorithme glouton en

utilisant ces poids. La valeur de l'objectif de cette solution est 17355. Notons que les poids des interventions ne sont pas utilisés pour évaluer la solution mais uniquement pour guider l'algorithme glouton.

Cet exemple montre que, pour cette instance, il est préférable de fixer un poids fort pour les interventions de priorité 3 et d'utiliser la permutation (4,3,1,2) des poids associés aux priorités. Nous précisons que les permutations (4,3,1,2), (3,2,1,4), (4,3,2,1) et (4,2,1,3) sont équivalentes pour l'algorithme glouton dans ce cas de figure où il n'y a pas d'intervention de priorité 4.

Mise à jour des poids

Notons $w_p(I)$, le poids associé à l'intervention I selon la permutation des poids associés aux priorités (p). Par exemple, supposons que $p = (3, 2, 1, 4)$, alors $w_p(I) = 28$ si la priorité de I est 3, $w_p(I) = 14$ si la priorité de I est 2, etc. A la fin de l'algorithme glouton, les poids des interventions sont mis à jour à partir de l'information fournie par la solution générée [11]. Cette mise à jour consiste à ajouter la valeur $w_p(I)$ aux dernières interventions de chaque priorité et à tous leurs prédécesseurs. L'algorithme glouton planifiera ces interventions plus tôt à l'itération suivante. L'algorithme 1 illustre cette procédure de mise à jour.

Algorithme 1: phase de mise à jour

Données: La liste des interventions. Une permutation p.
pour *chaque priorité prio* **faire**
  $I :=$ dernière intervention planifiée de priorité *prio* ;
  $w(I) = w(I) + w_p(I)$;
  pour *chaque intervention* $J \in Pred(I)$ **faire**
    $w(J) = w(J) + w_p(I)$;

3.2.2 Initialisation de la mémoire

L'identification de la permutation qui conduit au meilleur comportement de l'algorithme glouton est réalisée par la procédure d'échantillonnage suivante :

- Appliquer plusieurs fois l'algorithme glouton pour chacune des 24 permutations des poids associés aux priorités. Trier les permutations selon la meilleure borne supérieure obtenue.
- Répéter la même procédure uniquement sur les 12 permutations qui donnent les meilleures valeurs.
- Répéter la même procédure uniquement sur les 6 permutations qui donnent les meilleures valeurs.

Lorsque cette phase est terminée, nous conservons les 2 *meilleures* permutations. L'algorithme GRASP utilisera ces deux permutations pour générer toutes les autres solutions : les poids des interventions évoluant depuis ces valeurs de départ.

3.2.3 Amélioration des solutions par la recherche locale

Dans cette section nous décrivons un algorithme de recherche locale qui explore des configurations réalisables.

Après avoir affecté les interventions aux équipes et déterminé l'ordre dans lequel les interventions sont traitées pour chaque équipe, nous pouvons vérifier la faisabilité de l'ordonnancement. De plus, les dates de début optimales des interventions peuvent être déterminées facilement dans ce cas, puisque le graphe représentant l'ordre d'exécution des interventions avec les contraintes de précédence est un arbre acyclique direct pondéré [2].

Nous proposons deux algorithmes de recherche locale que nous appelons : phase *critical path* et phase *packing*. Nous utilisons un mouvement d'échange et un mouvement d'insertion, et ne considérons que les mouvements réalisables.

Lors de la recherche locale, nous maintenons le nombre minimal de techniciens affectés pour les interventions planifiées. Ainsi, les techniciens disponibles durant la journée appartiennent soit à une équipe vide pour laquelle aucune intervention n'est assignée, soit aux équipes pour lesquelles le nombre de techniciens est minimal pour les interventions qui leurs sont assignées.

Une opération d'échange consiste à intervertir l'affectation et l'ordre de deux interventions. Dans cette opération, la réaffectation des techniciens n'est pas considérée car ce n'est pas un problème trivial. Cependant, si la solution voisine est acceptée dans le mouvement, les techniciens peuvent être réaffectés afin de préserver le nombre minimum d'équipes. Ceci est fait en déplaçant des techniciens dans des équipes vides.

Une opération d'insertion supprime une intervention et l'insère à une autre position temporelle. Dans cette opération, après avoir retiré l'intervention, il faut éventuellement supprimer des techniciens de l'équipe qui l'avait en charge pour en maintenir le nombre minimal. Symétriquement, il faut éventuellement ajouter des techniciens dans l'équipe qui prendra en charge cette intervention.

Phase *critical path*

Le but de la phase *critical path* est de réduire les dates de fin de chaque priorité et celle de l'ordonnancement global (i.e., t_1, t_2, t_3 et t_4) simultanément.

Un chemin critique pour une priorité est défini comme étant une séquence maximale $(I_1, I_2, \ldots, I_l)$ d'interventions telles que l'intervention I_l donne la date de fin de la priorité considérée, et chaque intervention consécutive I_k et I_{k+1} $(k = 1, \ldots, l-1)$ satisfait

$$d(I_k) = d(I_{k+1}) \text{ et } s(I_k) + T(I_k) = s(I_{k+1})$$

ou

$$d(I_k) + 1 = d(I_{k+1}),\ s(I_{k+1}) = 0 \text{ et } s(I_k) + T(I_k) + T(I_{k+1}) > H_{\max}.$$

L'intervention I_{k+1} ne peut être séquencée si l'intervention I_k n'est pas séquencée plus tôt. Par définition d'un chemin critique, l'intervention I_1 doit être séquencée plus tôt si nous voulons réduire la date de fin de la priorité.

L'algorithme de recherche locale cherche un chemin critique pour chaque priorité et essaie de rompre ce chemin en insérant l'intervention I_1 au plus tôt.

Phase *packing*

Dans la phase *packing*, l'algorithme cherche à insérer les interventions de manière efficace sans dégrader l'objectif.

Nous considérons une mesure de l'efficacité pour l'équipe ϵ du jour j. Soit $J_\epsilon = \{I_1, I_2, \ldots, I_l\}$ les interventions qui sont affectées à l'équipe ϵ. Soit $N(J_\epsilon, i, n)$ le nombre maximum de techniciens requis pour le niveau n et le domaine i pour réaliser les interventions de J_ϵ (i.e., $N(J_\epsilon, i, n) = \max_{I \in J_\epsilon} R(I, i, n)$). Soit

$$W_{\text{skill}}(J_\epsilon) = \sum_{I \in J_\epsilon} \sum_{i,n} (N(J_\epsilon, i, n) - R(I, i, n))T(I)$$

et

$$W_{\text{time}}(J_\epsilon) = H_{\max} - \sum_{I \in J_\epsilon} T(I),$$

qui représentent respectivement les niveaux de compétence "gaspillés" et le temps "gaspillé" pour l'équipe ϵ et les interventions J_ϵ. Nous estimons l'efficacité de l'affectation des interventions J_ϵ à l'équipe ϵ par la fonction

$$f(J_\epsilon) = W_{\text{skill}}(J_\epsilon) + \alpha W_{\text{time}}(J_\epsilon),$$

où α est fixé à une grande valeur pour prendre en compte en priorité le temps puis les niveaux de compétence.

Dans cette phase, la recherche locale estime la valeur d'une solution en sommant $f(J_\epsilon)$ pour toutes les équipes de tous les jours et elle accepte une solution voisine pour un mouvement si la solution est réalisable et qu'elle n'augmente pas les dates de fin de chaque priorité.

3.3 Schéma général de l'approche de résolution

L'algorithme 2 résume les trois phases que nous avons exposées dans les sections précédentes. L'heuristique de prétraitement qui sélectionne les interventions à sous-traiter est représentée par la fonction *Glouton_Sous_traités*. Cette fonction retourne un sous-problème dans lequel une partie des variables a été fixée (i.e. avec quelques interventions supprimées). La seconde phase qui consiste à déterminer les deux meilleures permutations des poids associés aux priorités est représentée par la fonction *Initialiser_Memoire*. Cette procédure fournit $perm_1$ and $perm_2$, qui correspondent aux deux permutations utilisées dans la procédure GRASP qui est décrite après. Elle consiste à répéter l'exécution de l'algorithme glouton avec les deux permutations sélectionnées, puis à mettre à jour la mémoire et finalement à appliquer la recherche locale lorsque la meilleure solution est améliorée. Le processus s'arrête lorsque le temps CPU alloué est dépassé.

Algorithme 2: Résoudre _TIST(PB,MAX_CPU_Alloué)

Données: Une instance PB de TIST à résoudre ; Le temps CPU alloué.
$Meilleure_Solution$=$\emptyset$;
SPB=$Glouton_Sous_traités(PB)$;
$(perm_1, perm_2)$=$Initialiser_Memoire(SPB)$;
tant que *MAX_ CPU_ Alloué n'est pas atteint* **faire**
  $Solution_1$=$Construction_Gloutonne(SPB, perm_1)$;
  $Solution_2$=$Construction_Gloutonne(SPB, perm_2)$;
  Mettre à jour la mémoire ;
  $Amélioration$=$Mise_A_Jour(Solution_1, Solution_2, Meilleure_Solution)$;

  si $Amélioration = True$ **alors**
    $Meilleure_Solution$=$Recherche_Locale(SPB, Meilleure_Solution)$;
return $Meilleure_Solution$;

4 Calcul d'une borne inférieure

Dans cette section, nous considérons une borne inférieure du problème P' où les interventions sous-traitées ont été supprimées, afin d'évaluer la performance de l'algorithme GRASP.

Pour calculer la borne inférieure de P' nous considérons huit problèmes relaxés avec une restriction sur les interventions choisies. Sur chacun de ces huit problèmes nous calculons une borne inférieure de la date de fin de l'ordonnancement (appelée makespan). La borne inférieure de P' est finalement calculée en utilisant ces valeurs.

Nous considérons les problèmes suivants :

- $MSP(1)$ avec uniquement les interventions de priorité 1 et leurs prédécesseurs pour P'.
- $MSP(2)$ avec uniquement les interventions de priorité 2 et leurs prédécesseurs pour P'.
- $MSP(3)$ avec uniquement les interventions de priorité 3 et leurs prédécesseurs pour P'.
- $MSP(1, 2)$ avec uniquement les interventions de priorité 1 et 2 et leurs prédécesseurs pour P'.
- $MSP(2, 3)$ avec uniquement les interventions de priorité 2 et 3 et leurs prédécesseurs pour P'.
- $MSP(3, 1)$ avec uniquement les interventions de priorité 3 et 1 et leurs prédécesseurs pour P'.
- $MSP(1, 2, 3)$ avec les interventions de priorité 1 et 2 et 3 et leurs prédécesseurs pour P'.
- $MSP(1, 2, 3, 4)$ avec toutes les interventions pour P'.

Si l'on note T_1, T_2, T_3, $T_{1,2}$, $T_{2,3}$, $T_{3,1}$, $T_{1,2,3}$ et $T_{1,2,3,4}$ les bornes inférieures

des makespan respectifs, la résolution du problème suivant fournit une borne inférieure pour P' :

$$\begin{array}{ll} \text{minimiser} & 28t_1 + 14t_2 + 4t_3 + t_4 \\ \text{sujet à} & T_1 \leq t_1,\ T_2 \leq t_2,\ T_3 \leq t_3 \\ & T_{1,2} \leq \max\{t_1, t_2\},\ T_{2,3} \leq \max\{t_2, t_3\},\ T_{3,1} \leq \max\{t_3, t_1\} \\ & T_{1,2,3} \leq \max\{t_1, t_2, t_3\} \\ & T_{1,2,3,4} \leq t_4, \end{array}$$

où t_1, t_2 et t_3 sont les dates de fin des priorités 1, 2 et 3, respectivement, et t_4 est la date de fin de l'ordonnancement global. Toute solution réalisable (i.e., t_1, t_2, t_3 et t_4) doit satisfaire les contraintes précédentes.

Nous proposons trois bornes inférieures pour chaque problème : la borne inférieure par *boîtes* qui est calculée combinatoirement ; la borne inférieure par *affectation* qui est obtenue par la résolution d'un programme linéaire qui est un sous-problème de P' ; et enfin la borne *triviale* qui est dérivée de conditions triviales du problème. Nous choisissons la meilleure borne inférieure parmi celles-ci et la renforçons par une procédure d'amélioration.

4.1 Borne inférieure par boîtes

La borne inférieure par boîtes, qui est une borne inférieure sur le nombre de jours de l'ordonnancement, est calculée pour chaque domaine i et niveau de compétence n, et la plus grande valeur est conservée. La même méthode a été proposée par Lodi, Martello and Vigo [8] pour le problème "*two-dimensional level packing problem*".

Pour chaque domaine i et niveau de compétence n, nous considérons un rectangle, associé à chaque intervention I, dont la hauteur est $R(I, i, n)$ et la largeur est $T(I)$ (durée de l'intervention I).

Considérons tous les rectangles arrangés bout à bout par hauteur décroissante comme dans la figure 3 (a).

Ces rectangles sont d'abord découpés en blocs de largeur $H_{\max}$ pour respecter la durée de travail d'une journée. Ces blocs ont donc une largeur égale à $H_{\max}$ et une hauteur égale au $R(I, i, n)$ de la première intervention I qui les constitue.

Nous superposons ensuite ces blocs les uns sur les autres pour n'en constituer plus qu'un. Cette opération est illustrée par la figure 3 (b) où $Ad(j, i, n)$ représente le nombre de techniciens qui peuvent travailler le jour j et dont le niveau de compétence dans le domaine i est n.

Nous calculons enfin $\mu^* = \min\{\mu \in Z \mid H \leq \sum_{j=1}^{\mu} Ad(j, i, n)\}$ qui représente le nombre minimum de jours nécessaires pour pouvoir couvrir la hauteur totale du bloc construit précédemment. μ^* est ainsi une borne inférieure du nombre de jours pour l'ordonnancement si l'on ne considère que le domaine i et le niveau n. La figure 3 (b) illustre une situation où $\mu^* = 2$.

4.2 Borne inférieure par affectation

Pour calculer la borne inférieure par affectation, nous estimons successivement le nombre de jours μ de l'ordonnancement en résolvant un programme linéaire correspondant à μ, et nous répétons le processus jusqu'à ce que l'estimation soit correcte.

Pour un nombre de jours μ donné (nous estimons alors que le makespan M est dans $[\mu H_{\max}, (\mu+1)H_{\max}]$), le temps de travail disponible $U_\mu(t, M)$ jusqu'au temps M pour chaque technicien t peut être calculé facilement (notons que $U_\mu(t, M)$ est représenté par $M - lH_{\max}$ pour $l \leq \mu$ ou 0).

Nous considérons alors le programme linéaire suivant ALP(μ) :

$$\text{minimiser} \quad M \tag{11}$$

$$\text{sujet à} \quad \sum_{t|C(t,i)\geq n} x_{I,t} \geq R(I,i,n), \qquad \forall I, \forall i, \forall n \tag{12}$$

$$\sum_I T(I) x_{I,t} \leq U_\mu(t, M) \qquad \forall t \tag{13}$$

$$0 \leq x_{I,t} \leq 1 \qquad \forall I, \forall t, \tag{14}$$

où $x_{I,t}$ représente l'affectation de l'intervention I au technicien t. L'ensemble des solutions réalisables de ALP(μ) est inclu dans celui de ALP($\mu+1$) et donc, la valeur optimale de ce problème devient plus petite lorsque μ augmente.

Si le problème n'a pas de solutions réalisables ou que la valeur optimale M^* de ALP(μ) est supérieure à $(\mu+1)H_{\max}$, nous en déduisons que la borne inférieure est plus grande. Si la valeur optimale M^* de ALP(μ) est inférieure à $(\mu+1)H_{\max}$, nous en déduisons que la borne inférieure est plus petite. Le processus est répété jusqu'à ce que l'estimation réussisse et la dernière valeur M^* s'avère être la borne inférieure par affectation.

4.3 Borne inférieure triviale

La borne inférieure triviale est dérivée des conditions nécessaires suivantes : (1) Le temps maximal d'exécution d'une intervention parmi toutes les interventions est une borne inférieure (2) La somme des temps d'exécution pour une séquence d'interventions où chaque intervention est liée par une contrainte de précédence est une borne inférieure. La valeur maximale de toutes ces séquences peut être calculée en temps linéaire et donne une borne inférieure du problème.

4.4 Renforcement de la borne

Sachant que le makespan est une combinaison de $T(I)$ et $H_{\max}$, il est possible de renforcer la borne inférieure courante. Pour cela, nous calculons tous les makespan possibles par un algorithme de programmation dynamique pour le jour associé à la borne inférieure, et nous prenons la valeur la plus petite supérieure ou égale à la borne inférieure donnée. Cette valeur renforce la borne inférieure trouvée.

5 Résultats expérimentaux

Notre algorithme a été testé sur l'ensemble des données fournies par *France Telecom* pour le 5eme challenge de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADEF). Il y a trois ensembles de données disponibles, chaque ensemble contient 10 instances avec un nombre différent d'interventions, de techniciens, de domaines de compétence et de niveaux de compétence. Le premier ensemble appelé data-setA ne considère pas le problème des interventions sous-traitées. Il contient des instances ayant de 5 à 100 interventions, de 5 à 20 techniciens, de 3 à 5 domaines de compétence et de 2 à 4 niveaux de compétence. L'ensemble data-setB contient des instances plus difficiles à résoudre qui prennent en compte le problème des interventions sous-traitées. Cet ensemble contient des instances ayant de 120 à 800 interventions, de 30 à 150 techniciens, de 4 à 40 domaines de compétence et de 3 à 5 niveaux de compétence. Finalement, l'ensemble data-setX est l'ensemble d'instances à partir desquelles le classement des équipes participant au challenge a été réalisé. Il contient des instances ayant de 100 à 800 interventions, de 20 à 100 techniciens, de 6 à 20 domaines de compétence et de 3 à 7 niveaux de compétence.

					GRASP			Meilleur objectif	
inst.	int.	tec.	dom.	lev.	valeur	BI	gap.	valeur	gap
1-setA	5	5	3	2	2340	2265	3.2	2340	0
2-setA	5	5	3	2	4755	4215	11.35	4755	0
3-setA	20	7	3	2	11880	11310	4.79	11880	0
4-setA	20	7	4	3	13452	10995	18.26	13452	0
5-setA	50	10	3	2	28845	26055	9.67	28845	0
6-setA	50	10	5	4	18870	17775	5.8	18795	0.39
7-setA	100	20	5	4	30840	27405	11.13	30540	0.97
8-setA	100	20	5	4	17355	16166	6.85	16920	2.50
9-setA	100	20	5	4	27692	25618	7.48	27692	0
10-setA	100	15	5	4	40020	35405	11.53	38296	4.3
						Moyenne	**9.01**		**0.81**
1-setB	200	20	4	4	43860	38385	12.48	34395	21.58
2-setB	300	30	5	3	20655	16605	19.6	15870	23.16
3-setB	400	40	4	4	20565	17460	15.09	16020	22.1
4-setB	400	30	40	3	26025	19035	26.85	25305	2.76
5-setB	500	50	7	4	120840	106290	12.04	89700	25.76
6-setB	500	30	8	3	34215	24450	28.54	27615	19.28
7-setB	500	100	10	5	35640	28470	20.11	33300	6.56
8-setB	800	150	10	4	33030	32820	0.63	33030	0
9-setB	120	60	5	5	29550	26310	10.96	28200	4.56
10-setB	120	40	5	5	34920	32790	6.09	34680	0.68
						Moyenne	**15.24**		**12.64**
1-setX	600	60	15	4	181575	140025	22.88	151140	16.76
2-setX	800	100	6	6	7260	6840	5.78	7260	0
3-setX	300	50	20	3	52680	49650	5.75	50040	5.01
4-setX	800	70	15	7	72860	59560	18.25	65400	10.23
5-setX	600	60	15	4	172500	126465	26.68	147000	14.78
6-setX	200	20	6	6	9480	6180	34.81	9480	0
7-setX	300	50	20	3	46680	45000	3.59	33240	28.79
8-setX	100	30	15	7	29070	20590	29.17	23640	18.67
9-setX	500	50	15	4	168420	101985	39.44	134760	19.98
10-setX	500	40	15	4	178560	99705	44.16	137040	23.25
						Moyenne	**23.05**		**13.74**

TABLE 1 – Résultats obtenus sur les instances fournies par France Telecom

Nous exposons, dans la table 1, les résultats officiels qui ont été publiés sur le site Web du challenge. L'ordinateur utilisé est un AMD avec un processeur de 1,8 GHz et 1 GB de DDR-RAM. Le temps d'exécution est limité à 1200 secondes. La description des données par colonne est la suivante : *inst.* : Nom de l'instance. *int.* : Nombre d'interventions. *tec.* : Nombre de techniciens. *dom.* : Nombre de domaines de compétence. *lev.* : Nombre de niveaux de compétence. La colonne GRASP a trois valeurs : *valeur* : La meilleure valeur de l'objectif trouvée par notre approche, *BI* : La valeur de la borne inférieure une fois les interventions sous-traitées choisies et *gap* : l'écart relatif entre la borne inférieure et la valeur de la fonction objectif. La colonne Meilleur objectif a deux valeurs : *valeur* : La meilleure valeur de l'objectif trouvée parmi tous les participants au challenge *gap* : L'écart relatif entre cette meilleure valeur de l'objectif et la valeur que notre méthode a trouvée. *Moyenne* : La moyenne des écarts.

Il y avait un total de 17 équipes participant à la phase finale du challenge. Notre algorithme a été placé en première position dans la catégorie *Junior* et en quatrième position dans le classement général.

La table 1 montre que l'écart entre la solution trouvée par notre algorithme et la borne inférieure est important pour certaines instances de l'ensemble datasetX. Les expérimentations réalisées sur ces instances, après le challenge, ont montré que notre algorithme n'atteint jamais la phase d'amélioration par la recherche locale. Nous devons accélérer la phase 2 de notre approche, qui est en cause ici, soit en allégeant la procédure d'échantillonnage décrite en section 3.2.2 soit en la remplaçant par une méthode déterministe du calcul de la permutation *optimale*.

Un autre point mis en évidence par ces expérimentations est que le choix des interventions à sous-traiter est sous-optimal et pénalise notre algorithme GRASP. En effet, certaines valeurs de borne inférieure (calculées sur le problème résiduel ne contenant pas les interventions sous-traitées) sont supérieures à la meilleure solution trouvée parmi tous les participants (majorant du problème global). Quelques expérimentations complémentaires confirment cependant l'efficacité de l'algorithme GRASP. Par exemple, pour l'instance 9-setB, en sélectionnant un sous-ensemble d'interventions à sous-traiter qui donne une borne inférieure plus petite (25695 au lieu de 26310) nous obtenons un majorant de 27960 et améliorons la meilleure valeur connue (28200) sur cette instance.

6 Conclusion

Dans cet article, nous avons présenté un problème de planification de techniciens et d'interventions pour les télécommunications et en avons fournit une formulation mathématique. Nous avons proposé une heuristique constituée de trois phases : (1) la fixation de certaines variables par la résolution heuristique d'un problème de sac à dos avec contraintes de précédence pour le choix des interventions à sous-traiter, (2) l'initialisation de la mémoire (poids attribués aux interventions) réalisée par la recherche de la meilleure permutation des poids associés aux priorités des interventions et (3) la procédure GRASP qui cherche

à améliorer les solutions initiales tout en mettant à jour les poids attribués aux interventions.

Les expérimentations ont clairement montré que l'heuristique de choix des interventions à sous-traiter est un facteur déterminant pour la qualité des résultats produits par notre approche. En effet, d'une part un calcul de borne inférieure sur les sous problèmes résiduels révèle, en général, un faible écart entre bornes inférieures et majorants. D'autre part, comme nous l'avons observé sur l'instance 9-setB, une autre stratégie de choix d'interventions à sous-traiter améliore significativement la performance de l'algorithme GRASP.

La méthode GRASP est donc prometteuse pour résoudre ce problème, en particulier si nous améliorons la première phase de notre approche globale ainsi que si nous accélérons la phase d'identification des permutations des poids utilisées par l'heuristique gloutonne. C'est sur ces points que nous avons l'intention de conduire nos futurs travaux pour ce problème.

Références

[1] J. B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operational Research Society*, 49 :700–708, 1998.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, 2 edition, 2001.

[3] P.-F. Dutot and A. Laugier. Technicians and interventions scheduling for telecommunications(ROADEF challenge subject). Technical report, France Telecom R&D, 2005.

[4] T. A. Feo and M. G. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8 :67–71, 1989.

[5] P. Festa and M. G. C. Resende. GRASP : An annotated bibliography. In C. C. Ribeiro and P. Hansen, editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[6] C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11 :198–204, 1999.

[7] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[8] A. Lodi, S. Martello, and D. Vigo. Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization*, 8 :363–379, 2004.

[9] M. G. C. Resende and C. C. Ribeiro. A GRASP for graph planarization. *Networks*, 29 :173–189, 1997.

[10] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.

[11] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive memory programming : A unified view of metaheuristics. *European Journal of Operational Research*, 135 :1–16, 2001.

[12] J. Xu and S. Y. Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7 :495–509, 2001.

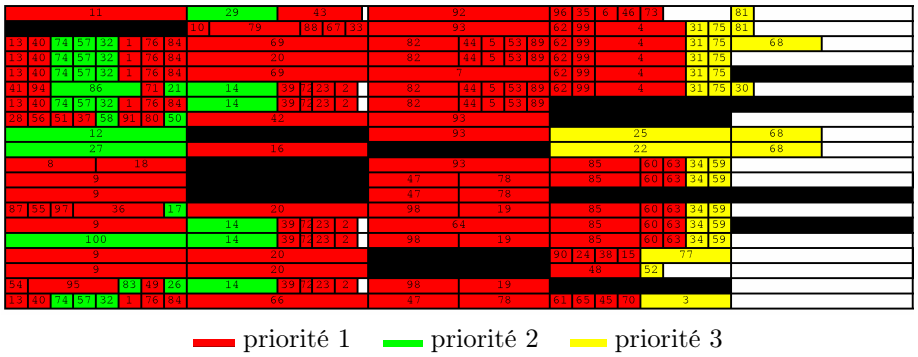

FIGURE 1 – Solution avec un objectif de 17820 pour l'instance data8 de l'ensemble data-setA

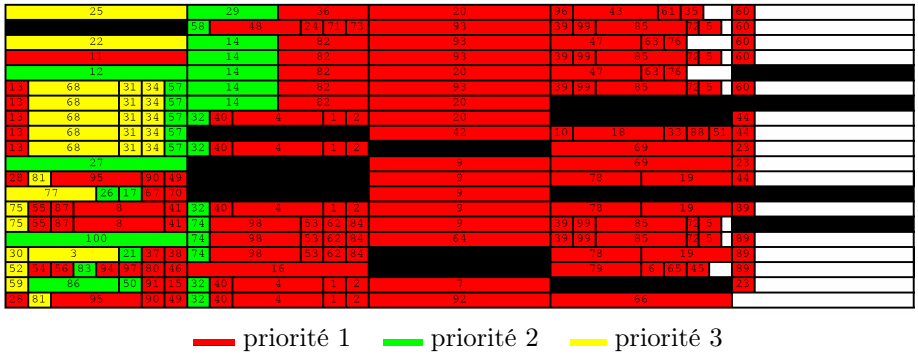

FIGURE 2 – Solution avec un objectif de 17355 pour l'instance data8 de l'ensemble data-setA

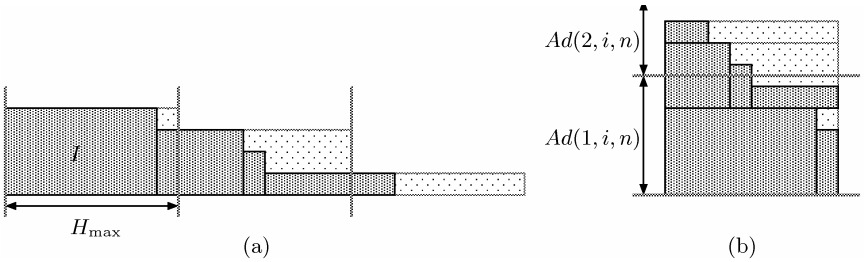

FIGURE 3 – Illustration du calcul de la borne inférieure par boîtes