RAIRO-Oper. Res. 43 (2009) 387–407
DOI: 10.1051/ro/2009027

RAIRO Operations Research
www.rairo-ro.org

UN ALGORITHME GRASP POUR LE PROBLÈME DE PLANIFICATION DE TECHNICIENS ET D'INTERVENTIONS POUR LES TÉLÉCOMMUNICATIONS

Sylvain Boussier[1], Hideki Hashimoto[2], Michel Vasquez[1] et Christophe Wilbaut[3]

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.

Mots Clés. Planification, heuristique, mémoire Adaptative.

Reçu le 4 novembre, 2008. Accepté le 17 juin, 2009.

[1] LGI2P, École des Mines d'Alès, Parc Scientifique Georges Besse, 30035 Nîmes Cedex 1, France ; Sylvain.Boussier@ema.fr, Michel.Vasquez@ema.fr

[2] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 606-8501 Kyoto, Japan ; hasimoto@nagoya-u.jp

[3] LAMIH, Université de Valenciennes et du Hainaut-Cambrésis, Le Mont Houy, 59313 Valenciennes Cedex 9, France ; christophe.wilbaut@univ-valenciennes.fr

Article published by EDP Sciences

Abstract. The Technicians and Interventions Scheduling Problem for Telecommunications embeds the scheduling of interventions, the assignment of teams to interventions and the assignment of technicians to teams. Every intervention is characterized, among other attributes, by a priority. The objective of this problem is to schedule interventions such that the interventions with the highest priority are scheduled at the earliest time possible while satisfying a set of constraints like the precedence between some interventions and the minimum number of technicians needed with the required skill levels for the intervention. We present a Greedy Randomized Adaptive Search Procedure (GRASP) for solving this problem. In the proposed implementation, we integrate dynamic update of the insertion criteria to the GRASP framework in order to generate good-quality solutions using information brought by previous ones. We also compute lower bounds and present experimental results that validate the effectiveness of this approach.

Keywords. Technicians and intervention scheduling, GRASP, metaheuristics.

Classification Mathématique. 90C59, 90B35, 90B50.

1. Introduction

Dans cet article, nous nous intéressons à la résolution d'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 5e challenge de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)[1].

Dans la société *France Telecom*, les décideurs doivent déterminer, pour chaque jour, quels techniciens doivent travailler ensemble et quelles interventions ils devront réaliser. Avec l'accroissement du nombre d'interventions dû à l'expansion des nouveaux services associés à Internet, et pour maintenir la compétitivité tout en limitant l'accroissement du nombre de techniciens, les problèmes à traiter deviennent de plus en plus difficiles. L'objectif du TIST est de fournir des ordonnancements efficaces afin d'aider les décideurs dans leur travail.

Le TIST est un problème d'affectation de tâches et de planification comprenant cependant quelques caractéristiques particulières. Premièrement, les tâches (*i.e.* les interventions dans notre cas) peuvent être liées par des contraintes de précédence. Il est ainsi impossible de commencer une intervention avant que son, ou ses prédécesseurs, ne soient complètement achevés. Deuxièmement, les interventions sont de différents types et requièrent des compétences différentes dans des

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

domaines différents. Les techniciens sont spécialisés dans certains domaines et sont caractérisés par un certain niveau de compétence dans chaque domaine. De plus les équipes de techniciens sont formées pour la journée, en raison du nombre limité de véhicules disponibles et du temps que cela prendrait de rapatrier le personnel afin de former de nouvelles équipes. Troisièmement, les interventions ont une certaine priorité selon leur importance. Cette priorité est bien entendue à prendre en compte lors de la résolution du problème. Finalement, il est possible de faire appel à une entreprise extérieure pour sous-traiter une ou plusieurs interventions selon un budget total disponible.

Certains problèmes de la littérature se rapprochent du TIST. Ainsi dans le domaine des télécommunications, Xu et Chiu [15] ont appliqué la méthode GRASP pour résoudre un problème de planification de techniciens dans un contexte similaire. Dans cet article, les auteurs développent un algorithme glouton, un algorithme de recherche locale et un algorithme GRASP, et montrent que la méthode GRASP fournit les meilleurs résultats en dépit d'un accroissement important du temps d'exécution. Une implémentation parallèle de leur algorithme leur permet néanmoins de remédier à ce problème. Tsang et Voudouris [14] ont travaillé sur un problème similaire pour les télécommunications britanniques et ont proposé un algorithme de recherche locale permettant d'obtenir de meilleurs résultats que d'autres algorithmes approchés de la littérature. De nombreuses autres méthodes peuvent être recensées pour résoudre des problèmes d'affectation ou de planification. Par exemple, Tang *et al.* [13] ont proposé un algorithme basé sur la recherche tabou [7] utilisant une mémoire adaptative pour résoudre un problème réel de planification de maintenances d'équipements distants permettant d'obtenir des solutions de bonne qualité en des temps raisonnables.

Le cœur de notre approche est constitué d'un algorithme GRASP [4]. La terminologie GRASP se réfère à une classe de procédures dans laquelle une heuristique gloutonne ainsi qu'une technique de recherche locale sont employées. La métaheuristique GRASP a été appliquée à de nombreux problèmes d'optimisation combinatoire comme des problèmes d'ordonnancement [15], de routage [1], de graphes [10], d'affectation [6], etc. Le lecteur peut se référer par exemple à [11] ou [5] pour une bibliographie plus complète.

Une implémentation classique de la méthode GRASP consiste 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, etc.) : (i) Générer une solution réalisable avec un algorithme glouton ; (ii) Appliquer un algorithme de recherche locale à la solution précédente ; (iii) Mettre à jour la meilleure solution.

Notre algorithme peut être divisé en trois phases : premièrement, une phase de prétraitement qui sélectionne de manière heuristique les interventions à sous-traiter ; deuxièmement, une phase d'initialisation de la mémoire au cours de laquelle un algorithme glouton détermine de manière approchée le « meilleur » critère d'insertion des interventions ; troisièmement, une phase de recherche de solutions par une procédure GRASP.

L'article 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, présentés dans la section 5.

2. Description du problème

Cette section est consacrée à la description du problème et à l'introduction d'un ensemble de notations utilisées tout au long de l'article.

Le but du TIST est de déterminer un ordonnancement d'*interventions* satisfaisant un ensemble de contraintes. Une intervention I de l'ensemble des interventions $\mathcal{I}$ est caractérisée par : son temps d'exécution $T(I)$, son coût si elle est sous-traitée $cost(I)$ et une liste de ses prédécesseurs $Pred(I)$. Nous notons $\mathcal{P}(I_1, I_2) = 1$ si l'intervention I_1 est un prédécesseur de l'intervention I_2 ($\mathcal{P}(I_1, I_2) = 0$ sinon). Les interventions pouvant être très diverses, elles sont également caractérisées par un ensemble de compétences requises, ou demandes. De manière à pouvoir facilement savoir si un *technicien* t parmi l'ensemble $\mathcal{T}$ des techniciens peut répondre aux attentes d'une intervention, les compétences sont regroupées en un certain nombre de *domaines*, et chaque technicien t a un *niveau* n d'un ensemble de niveaux $\mathcal{N}$ dans chaque domaine i de l'ensemble $\mathcal{D}$ des domaines, noté $C(t, i)$. À titre d'exemple un technicien ayant un niveau 0 dans un domaine n'a pas de connaissance dans ce domaine précis, alors qu'un technicien de niveau 4 sera considéré comme un expert. Pour chaque intervention I, $R(I, i, n)$ représente le nombre de techniciens requis de niveau n dans le domaine i pour traiter l'intervention I. Une intervention I qui nécessite, pour le domaine i, au moins un technicien de niveau trois et un technicien de niveau deux a donc ses demandes notées : $R(I, i, 1) = 2$, $R(I, i, 2) = 2$, $R(I, i, 3) = 1$, $R(I, i, 4) = 0$.

Un technicien t est également caractérisé par un ensemble de jours de disponibilités représenté par la notation $P(t, j) = 1$ si le technicien est disponible le jour j ($P(t, j) = 0$ sinon).

Chaque intervention doit être affectée à une *équipe* de techniciens à une *date* donnée. Nous notons par $e(t, j)$ le numéro de l'équipe à laquelle est rattaché le technicien t le jour j. L'équipe 0 est une équipe spécifique composée des techniciens ne travaillant pas le jour j. Rappelons que chaque technicien ne peut appartenir qu'à une seule équipe chaque jour. La date d'une intervention I est composée du jour de l'intervention, noté $d(I)$, et de l'heure de début, notée $s(I)$. Une journée de travail est constituée de $H_{\max}$ unités de temps ($H_{\max} = 120$ dans le sujet). Une intervention doit être réalisée par une seule équipe à une unique date, et nous considérons que chaque journée de travail est dans l'intervalle de temps $[0, H_{\max}]$, noté IT, sachant 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. Chaque équipe doit bien évidemment satisfaire l'ensemble des compétences requises par les interventions qui lui sont affectées.

Une priorité est également associée à chaque intervention I. Pour chaque priorité k, nous définissons la constante $\Pr(k, I) = 1$ si la priorité de I vaut k, 0 sinon.

Nous notons finalement par A le budget total alloué pour le sous-traitement des interventions.

L'évaluation d'une solution du problème se fait selon la fonction objectif suivante :

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

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. La valeur de t_k correspond ici au nombre total d'unités de temps. Ainsi par exemple si la dernière intervention de priorité 1 se termine le jour 3 à l'heure 60, alors $t_1 = 2 * 120 + 60 = 300$.

3. Approche générale

Dans cette section nous détaillons le rôle ainsi que les principes de chaque partie de notre 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 réduit.

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 I. Ce poids, noté ω_I, est établi à partir d'une borne sur le 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 : $\omega_I = \text{mintec}(I) \times T(I)$.

Soit $\Omega_{\mathcal{T}}$ l'ensemble des indices des variables représentant les techniciens et $x \in \{0,1\}^{|\Omega_{\mathcal{T}}|}$ un vecteur de variables de décision. Dans un premier temps, la valeur $mintec(I)$ est approximée en résolvant de manière heuristique le programme linéaire en nombres entiers associé à I suivant :

$$\begin{cases} \text{Minimiser } \sum_{t\in\Omega_{\mathcal{T}}} x_t \text{ sujet à,} \\ \sum_{t/C(t,i)\geq n, t\in\Omega_{\mathcal{T}}} x_t \geq R(I,i,n) \quad \forall i \in \mathcal{D}, n \in \mathcal{N}, \\ x_t \in \{0,1\} \quad t \in \Omega_{\mathcal{T}}. \end{cases}$$

L'heuristique utilisée pour évaluer la valeur de $mintec(I)$ est décrite dans l'Algorithme 1. Elle consiste à ajouter des techniciens dans une équipe initialement vide selon le plus grand nombre de demandes de l'intervention I satisfaites. Une phase de descente est ensuite appliquée pour essayer de remplacer un couple de techniciens par un seul.

Algorithme 1: mintec(I)

Données: L'intervention I
$\epsilon :=$ équipe vide ;
$T :=$ sous-ensemble de techniciens satisfaisant au moins une demande de I ;
pour *chaque technicien* t *de* T **faire**
  $sk(t, I) :=$ nombre de niveaux de compétences de t satisfaisants les demandes de I ;
Réarranger T selon l'ordre décroissant des valeurs $sk(t, I)$;
tant que ϵ *ne satisfait pas toutes les demandes de* I **faire**
  Ajouter un nouveau technicien t dans ϵ ;
  $T = T - \{t\}$;
pour *chaque paire de techniciens* $\{t,t'\}$ *dans* ϵ **faire**
  pour *chaque technicien* t'' *dans* T **faire**
    si $\{t,t'\}$ *peut être remplacé par* $\{t''\}$ *en satisfaisant les demandes* **alors**
      Remplacer $\{t,t'\}$ par $\{t''\}$ dans ϵ ;
      Retourner le nombre de techniciens dans ϵ ;
Retourner le nombre de techniciens dans ϵ ;

Soit $\Omega_{\mathcal{I}}$ l'ensemble des indices des interventions et $x \in \{0,1\}^{|\Omega_{\mathcal{I}}|}$ le vecteur de variables de décision tel que $x_I = 1$ si l'intervention I est sous-traitée et 0 sinon. Nous devons trouver 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 sommes donc en face d'un problème de sac à dos avec contraintes de précédence [8]. Soit PCKP ce problème (pour *Precedence Constraint Knapsack Problem*), il peut être formulé de la manière suivante :

$$PCKP \begin{cases} \text{Maximiser } \sum_{I \in \Omega_{\mathcal{I}}} w_I x_I \text{ sujet à,} \\ \sum_{I \in \Omega_{\mathcal{I}}} \text{cost}(I) \cdot x_I \leq A, \\ x_I \leq x_{I'} \quad \forall I, I' \in \Omega_{\mathcal{I}} \mid \mathcal{P}(I, I') = 1 \\ x_I \in \{0,1\} \quad I \in \Omega_{\mathcal{I}}. \end{cases}$$

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

Ce choix de gestion des interventions à sous-traiter 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

Nous commençons cette section par une présentation de l'algorithme glouton utilisé pour construire une solution réalisable.

3.2.1. *Phase constructive*

Dans une implémentation classique d'un algorithme GRASP, la phase constructive détermine un ensemble de *candidats*, ici d'interventions, qui peuvent être ajoutées dans la solution partielle en maintenant la réalisabilité de celle-ci. La sélection de l'élément suivant se fait en fonction d'un poids. Habituellement, les poids sont liés à l'augmentation de la valeur de la fonction objectif lors de l'ajout du candidat. Dans notre cas, ils sont tout d'abord initialisés à une valeur donnée qui correspond à la valeur du coefficient de la priorité du candidat dans la fonction objectif. 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. Les poids sont ensuite mis à jour à chaque itération en fonction de la solution obtenue comme décrit dans la suite.

Sélection d'un candidat

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.

Une fois un candidat sélectionné, l'algorithme glouton commence par calculer la date d'insertion (jour + heure) au plus tôt en fonction de ses prédécesseurs. L'algorithme essaie alors d'insérer l'intervention en favorisant une des deux politiques suivantes : (i) l'équipe qui nécessite le moins de techniciens supplémentaires pour traiter l'intervention ; (ii) l'équipe qui permet de commencer l'intervention au plus tôt. Le processus est répété jusqu'à ce que tous les candidats soient ordonnancés. Nous décrivons dans la suite la mise en application de ces éléments.

Calcul de la date au plus tôt selon les prédécesseurs

La première étape consiste à calculer la date de départ au plus tôt du candidat courant I. Cette date correspond à un couple de valeurs *jour*, $d(I)$, et *heure*, $s(I)$.

Soit I^* l'intervention ayant la date de fin la plus avancée dans la solution courante parmi tous les prédécesseurs de I. Si $s(I^*) + T(I^*) + T(I) > H_{\max}$ alors l'intervention I ne pourra pas être planifiée en $d(I^*)$, et dans ce cas nous avons : $d(I) = d(I^*) + 1$ et $s(I) = 0$. Dans le cas contraire nous avons : $d(I) = d(I^*)$ et

$s(I) = s(I^*) + T(I^*)$.

Une fois cette date ($d(I)$,$s(I)$) déterminée, l'algorithme considère les deux politiques (i) et (ii).

Évaluation du nombre minimum de techniciens supplémentaires requis pour le candidat

Cette première politique consiste à évaluer, pour chaque équipe existante ϵ en $d(I)$, le nombre minimum de techniciens à lui ajouter pour qu'elle puisse répondre aux demandes de I. L'algorithme évalue également la possibilité de créer une nouvelle équipe pour prendre en charge I en fonction des techniciens disponibles. Dans les deux cas, nous appliquons l'heuristique décrite dans l'Algorithme 1 en considérant le premier cas comme un cas particulier avec ϵ non vide au départ, et T ne comprenant que les techniciens disponibles.

Calcul de la date de début au plus tôt dans une équipe

Cette deuxième politique cherche à insérer l'intervention I au plus tôt le jour $d(I)$ dans chaque équipe ϵ existante ce jour, en la positionnant sans créer de chevauchement avec les interventions déjà attribuées à ϵ. Ici encore l'algorithme considère la possibilité de créer une nouvelle équipe qui n'aurait à ce stade que l'intervention I à prendre en charge. Ainsi si I ne peut être insérée entre deux interventions déjà planifiées, elle le sera à la suite dans la journée, à condition d'avoir suffisamment de temps avant la fin de celle-ci.

Quelle que soit la politique utilisée par l'algorithme, il est possible que l'insertion du candidat ne soit pas possible en $d(I)$. Si tel est le cas, le processus est répété en incrémentant la valeur de $d(I)$ d'une unité.

Application de (i) ou de (ii)

L'algorithme glouton va choisir une des politiques (i) ou (ii) en fonction des critères suivants. Soit k la priorité de l'intervention I. Si l'insertion de I ne provoque pas d'augmentation de t_k dans la solution courante (*i.e.* si $d(I) \times H_{\max} + s(I) + T(I) < t_k$), alors l'ajout de I n'influe pas directement sur la valeur de la fonction objectif. Dans ce cas l'algorithme favorise la politique (i) en minimisant le nombre de techniciens à ajouter. En cas d'égalité entre plusieurs équipes, l'algorithme favorisera celle qui donne le meilleur résultat selon (ii).

Dans le cas où l'insertion de I entraîne une augmentation de la valeur de t_k, et donc de la fonction objectif, l'algorithme favorisera la politique (ii) avant la politique (i).

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

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

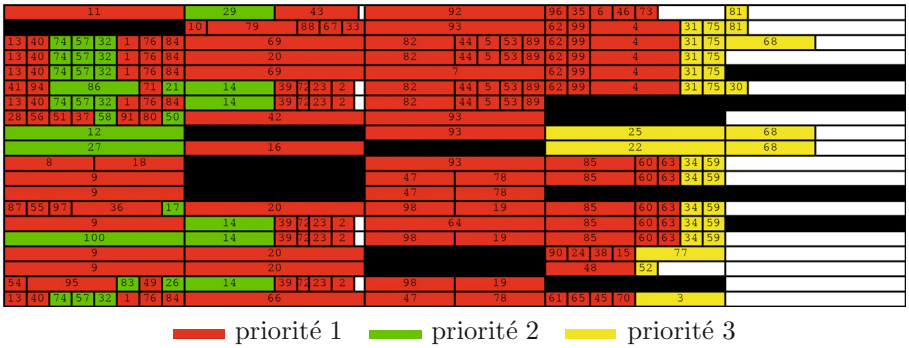

FIGURE 1. Solution avec un objectif de 17820 pour l'instance data 8 de l'ensemble data-setA.

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 pour une des instances du challenge. Les poids des interventions sont alors : 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 cette solution selon la fonction objectif du problème 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 cette solution est alors 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.

Cette constatation nous a guidés vers l'utilisation d'une mémoire adaptative pour mettre à jour les poids des interventions en fonction de la solution générée.

Mise à jour des poids

Notons maintenant $w_p(I)$ le poids associé à l'intervention I selon la permutation des poids p associés aux priorités. Par exemple, supposons que $p = (3, 2, 1, 4)$, alors

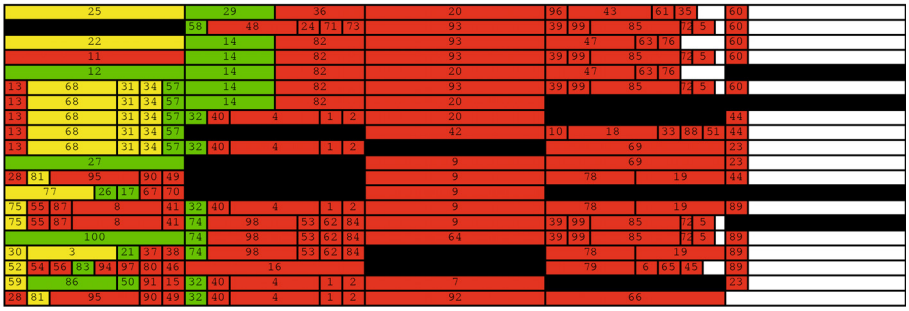

FIGURE 2. Solution avec un objectif de 17355 pour l'instance data 8 de l'ensemble data-setA.

$w_p(I) = 28$ si la priorité de I est 3, $w_p(I) = 14$ si la priorité de I est 2, etc. Une fois l'algorithme glouton terminé, les poids des interventions sont mis à jour à partir de l'information fournie par la solution générée [12]. Cette mise à jour consiste à ajouter la valeur $w_p(I)$ à la dernière intervention de chaque priorité et à tous ses prédécesseurs. L'algorithme glouton favorisera ainsi ces interventions à l'itération suivante pour les planifier plus tôt. L'Algorithme 2 illustre cette procédure de mise à jour.

Algorithme 2: Phase de mise à jour

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

3.2.2. *Initialisation de la mémoire*

Après la phase de prétraitement visant à supprimer une partie des interventions via le sous-traitement, notre approche consiste à évaluer les 24 permutations possibles des poids associés aux priorités de manière à ne retenir que les deux ayant conduit aux meilleurs résultats. L'algorithme GRASP est ensuite appliqué en alternant la phase constructive utilisant une de ces deux permutations (à tour de rôle et tout en continuant à mettre à jour la mémoire), et la phase d'amélioration décrite dans la section suivante.

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

- Pour chacune des 24 permutations des poids des priorités faire :
  - Initialiser le poids de chaque intervention I au poids associé à la priorité de I selon la permutation courante ;

  - Appliquer plusieurs fois l'algorithme glouton (phase constructive décrite en Sect. 3.2.1) et conserver la meilleure borne supérieure ;
- Trier les permutations selon les valeurs croissantes des bornes supérieures obtenues ;
- 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 ;

3.2.3. *Amélioration des solutions par la recherche locale*

Dans cette section nous décrivons un algorithme de recherche locale qui explore uniquement 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 que nous appelons : phase *critical path* et phase *packing*. Dans les deux cas nous utilisons deux voisinages basés sur un mouvement d'échange et un mouvement d'insertion.

Lorsqu'une solution améliorante est acceptée, nous maintenons le nombre minimal de techniciens affectés pour les interventions planifiées. L'opérateur de mise à jour d'une équipe consiste à essayer de supprimer chaque technicien (l'un après l'autre dans un ordre aléatoire), et de le supprimer uniquement si la solution reste réalisable (*i.e.* si l'équipe répond toujours aux demandes de ses interventions). Les techniciens disponibles durant une journée de travail donnée appartiennent donc 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, les techniciens peuvent être réaffectés afin de préserver le nombre minimum d'équipes. Ceci est fait en déplaçant des techniciens vers l'équipe vide.

Une opération d'insertion supprime une intervention de son équipe « courante », et l'insère à une autre position temporelle. Dans cette opération, après avoir retiré l'intervention, il faut éventuellement mettre à jour les équipes de techniciens concernées par le mouvement.

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 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 comme suit. 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

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

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.

Schéma de la recherche locale

Lors de la phase d'amélioration, l'algorithme alterne entre la phase *critical path*, appliquée en premier lieu, et la phase *packing*. Lors de la phase critical path, le voisinage d'échange est utilisé, puis le voisinage d'insertion. Si une solution améliorante est visitée (*i.e.* s'il est possible de planifier I_1 plus tôt), elle est immédiatement acceptée et la phase packing est appliquée. Si aucune solution améliorante n'est trouvée, la phase d'amélioration s'achève.

Lors de la phase packing, c'est le voisinage par insertion qui est utilisé en premier, et ensuite celui d'échange. Dans les deux cas, si une solution améliorante est visitée, l'algorithme l'accepte et réitère cette phase. La phase d'amélioration est ensuite relancée jusqu'à ce qu'il n'y ait plus d'amélioration.

3.3. Schéma général de l'approche de résolution

L'Algorithme 3 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_traitées*. 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 $perm_1$ et $perm_2$, qui correspondent aux deux permutations utilisées dans la procédure GRASP qui est décrite ensuite. 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 3: Résoudre le TIST

Données: Une instance du TIST, PB ; Le temps CPU alloué, MAX_CPU.
$Meilleure\_Solution = \oslash$;
SPB=$Glouton\_Sous\_traitées(PB)$;
$(perm_1, perm_2) = Initialiser\_Memoire(SPB)$;
tant que MAX_CPU *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**
    $Meilleure\_Solution = Recherche\_Locale(SPB, Meilleure\_Solution)$;
Retourner $Meilleure\_Solution$.

4. Calcul d'une borne inférieure

Dans cette section, nous proposons une borne inférieure du sous-problème dans lequel les interventions sous-traitées ont déjà été choisies. Cette borne nous permet d'évaluer la qualité des solutions obtenues sur ce sous-problème. Nous commençons par donner une formulation mathématique de ce problème réduit, noté (P'), avant de décrire une méthode de calcul d'une borne inférieure.

4.1. Modèle mathématique

Dans ce qui suit, $\mathcal{J}$ désigne l'ensemble des jours de la planification, $\mathcal{E}$ l'ensemble des équipes existantes (sur l'ensemble de la planification).

Nous définissons les variables suivantes dans le modèle mathématique :

- $x(I, j, h, \epsilon)$ égale 1 si l'équipe ϵ travaille sur l'intervention I le jour j à l'heure de début h, et 0 sinon.
- $y(j, \epsilon, t)$ égale 1 si le technicien t est dans l'équipe ϵ le jour j, et 0 sinon.

Le problème réduit (P') peut alors être modélisé de la manière suivante :

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

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

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

$$\sum_{\epsilon\in\mathcal{E}} y(j,\epsilon,t) = 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{3}$$

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

$$\sum_{h_1=\max(h_2-T(I_1)+1,0)}^{\min(h_2+T(I_2)-1,H_{\max})} x(I_1,j,h_1,\epsilon) + x(I_2,j,h_2,\epsilon) \le 1 \qquad \forall I_1 \in \mathcal{I}, I_2 \in \mathcal{I}, h_2 \in IT, j \in J, \epsilon \in \mathcal{E} \tag{5}$$

$$\sum_{j,h,\epsilon} \Big((jH_{\max} + h)\big(x(I_1,j,h,\epsilon) - x(I_2,j,h,\epsilon)\big) + T(I_1)x(I_1,j,h,\epsilon)\Big) \le 0 \qquad \forall I_1, I_2 \mid \mathcal{P}(I_1,I_2) = 1 \tag{6}$$

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

$$\sum_{h\in IT} R(I,d,n)x(I,j,h,\epsilon) \le \sum_{t\in\mathcal{T}\mid C(t,d)\ge n} y(j,\epsilon,t) \qquad \forall I \in \mathcal{I}, d \in \mathcal{D}, n \in \mathcal{N}, \epsilon \in \mathcal{E}, j \in \mathcal{J} \tag{8}$$

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

$$\sum_{j\in\mathcal{J}, h\in IT, \epsilon\in\mathcal{E}} (jH_{\max} + h + T(I))\, x(I,j,h,\epsilon) \le t_4 \qquad \forall I \in \mathcal{I} \tag{10}$$

$$x(I,j,h,\epsilon), y(j,\epsilon,t) \in \{0,1\} \qquad \forall I \in \mathcal{I}, j \in \mathcal{J}, h \in IT, \epsilon \in \mathcal{E}, t \in \mathcal{T}.$$

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) vérifie 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 $H_{\max}$. 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.

4.2. Borne inférieure

Pour calculer une 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 nous notons 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' :

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

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 de manière combinatoire ; 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.

4.2.1. *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 [9] 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 3a.

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 pour n'en constituer plus qu'un. Cette opération est illustrée par la figure 3b 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 3b illustre une situation où $\mu^* = 2$.

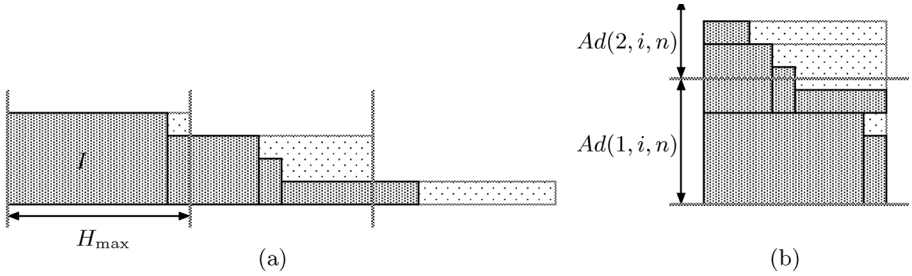

FIGURE 3. Illustration du calcul de la borne inférieure par boîtes.

4.2.2. *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 jusqu'au temps M pour chaque technicien t peut être calculé facilement, il correspond à la date M moins le nombre $l(t)$ de jours de congés du technicien t avec $l \in \mathbb{N}$ et $0 \leq l \leq \mu$.

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

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

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

$$\sum_{I\in\mathcal{I}} T(I)x_{I,t} \leq M - l(t)H_{\max} \qquad \forall t \in \mathcal{T} \tag{13}$$

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

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 solution réalisable 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 consiste tout d'abord à résoudre ALP(μ) avec $\mu = 0$ puis à incrémenter successivement μ jusqu'à ce que M^* soit inférieure à $(\mu+1)H_{\max}$. La dernière valeur M^* correspond alors à la borne inférieure par affectation.

4.2.3. *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.

4.2.4. *Renforcement de la borne*

Le makespan est nécessairement une combinaison de $H_{\max}$ et de $T(I)$. La borne inférieure $\underline{l}$ obtenue par la méthode décrite précédemment ne répond pas nécessairement à cette condition. Elle peut alors être renforcée à l'aide d'un algorithme de programmation dynamique. Celui-ci calcule toutes les dates susceptibles de correspondre au makespan le jour de $\underline{l}$. Pour cela il combine toutes les valeurs $T(I)$ des interventions I appartenant à $\mathcal{I}$ et mémorise les dates de fin possibles entre 0 et $H_{\max}$. La borne inférieure finale correspond alors à la plus petite valeur mémorisée supérieure ou égale à $\underline{l}$ pour le jour correspondant à $\underline{l}$.

5. Résultats expérimentaux

Notre algorithme a été évalué sur l'ensemble des données fournies par *France Telecom* pour le 5^{e} 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.

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 s pour tous les participants. 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

TABLEAU 1. Résultats obtenus sur les instances fournies par France Telecom.

					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**

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 et celle obtenue par notre méthode. *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, en particulier pour l'ensemble data-setX. 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 donc 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 la phase GRASP de notre approche. 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 cette phase 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. Nous avons proposé une heuristique constituée de trois phases : (1) la fixation de certaines variables par la résolution approchée 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 les résultats obtenus sur les instances ne permettant pas de sous-traiter d'interventions montrent un faible écart entre nos solutions et les meilleures obtenues lors du challenge. 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 notre algorithme.

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.

Remerciements. Nous tenons à remercier les deux arbitres anonymes qui, par la qualité de leur travail, ont permis d'améliorer grandement la version initiale de cet article.

Références

[1] J.B. Atkinson, A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *J. Oper. Res. Soc.* **49** (1998) 700–708.

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

[3] P.-F. Dutot and A. Laugier, *Technicians and interventions scheduling for telecommunications* (ROADEF challenge subject). Technical report, France Telecom R&D (2006).

[4] T.A. Feo and M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8** (1989) 67–71.

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

[6] C. Fleurent and F. Glover, Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS J. Comput.* **11** (1999) 198–204.

[7] F. Glover, Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** (1986) 533–549.

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

[9] A. Lodi, S. Martello and D. Vigo, Models and bounds for two-dimensional level packing problems. *J. Comb. Optim.* **8** (2004) 363–379.

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

[11] M.G.C. Resende and C.C. Ribeiro, Greedy randomized adaptive search procedures, in *Handbook of Metaheuristics*, F. Glover and G.A. Kochenberger (Eds.), Kluwer Academic Publishers (2003) 219–249.

[12] É.D. Taillard, L.M. Gambardella, M. Gendreau and J.-Y. Potvin, Adaptive memory programming: A unified view of metaheuristics. *Eur. J. Oper. Res.* **135** (2001) 1–16.

[13] H. Tang, E. Miller-Hooks and R. Tomastik, *Transp. Res. Part E. Logist. Transp. Rev.* **43** (2007) 591–609.

[14] E. Tsang and C. Voudouris, Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Oper. Res. Lett.* **20** (1997) 119–127.

[15] J. Xu and S.Y. Chiu, Effective heuristic procedures for a field technician scheduling problem. *J. Heurist.* **7** (2001) 495–509.