Application de Resolution search au problème de planification de techniciens et d'interventions pour les télécommunications

Nous présentons, dans ce chapitre, un travail réalisé dans le cadre du $5^{\text{ème}}$ challenge de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)[1]. Le sujet, proposé par *France Télécom*, concernait la planification de techniciens et d'interventions dans le domaine des télécommunications. Nous détaillons, dans une première partie, l'algorithme de recherche de bornes supérieures que nous avons présenté pour ce challenge et nous exposons ensuite une étude réalisée a posteriori sur l'application de Resolution search à la résolution d'un sous–problème du problème général.

L'objectif du problème que l'on note TIST (pour *Technicians and Interventions Scheduling Problem for Telecommunications*) est de concevoir des ordonnancements de techniciens et d'interventions dans le but d'aider les décideurs de *France Télécom* dans la réalisation de leurs plannings. Chaque jour, des équipes de techniciens doivent être formées afin d'effectuer différentes tâches à différents lieux géographiques. L'accroissement du nombre de clients et la diversification des services dus au développement du haut débit comme la voix sur ip ou la télévision par Internet rendent la réalisation des plannings de plus en plus complexe.

Les interventions sont caractérisées par des critères spécifiques comme une priorité de planification et une durée d'exécution, et certaines interventions doivent être réalisées avant d'autres. 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 compétence et chaque technicien a une liste de jours d'indisponibilité. D'autre part, chaque

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

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. Le TIST consiste à 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).

Dans une première partie, nous décrivons l'algorithme de résolution que nous avons présenté lors du challenge ROADEF'07. Cet algorithme, basé sur la métaheuristique GRASP (*Greedy Randomized Adaptive Search Procedure* [**?**]), donne des résultats prometteurs et nous a permis d'être classé $4^{ème}$ sur 35 équipes participantes lors du classement final. Néanmoins, l'un des défauts majeurs de cet algorithme concerne le choix des interventions à sous–traiter. Ce choix est effectué par une heuristique simple consistant à choisir dès le départ un ensemble d'interventions à sous–traiter et à ne plus considérer ces interventions durant la recherche. Un mauvais choix de départ peut donc pénaliser l'efficacité de l'algorithme durant tout le processus de recherche. Nous avons réalisé une étude a posteriori sur ce problème dans le but d'améliorer les résultats précédemment obtenus. Nous présentons, dans la deuxième partie de ce chapitre, les différentes voies qui ont été explorées pour cette étude et montrons en particulier que l'utilisation de Resolution search permet d'obtenir de meilleurs résultats que les autres méthodes testées sur la résolution de ce sous–problème. Une partie du travail décrit ici a fait l'objet de publications [42, 8].

5.1 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.

5.1.1 Description générale

Le problème traite d'interventions devant 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 le nombre de techniciens requis pour chaque niveau et domaine de compétence. L'objectif consiste à construire des équipes de techniciens pour chaque journée et à 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 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 au plus 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 pour former de nouvelles équipes.

Une équipe doit satisfaire les demandes de chaque intervention qui lui est assignée. 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.

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 5.1.2 ne prend pas en compte les interventions sous–traitées. En effet, cette partie du problème est réalisée dans une phase de prétraitement décrite en section 5.2.1.

La méthode GRASP consiste à alterner itérativement une phase de construction et une phase d'amélioration. La phase de construction génère une solution réalisable en insérant des interventions suivant un critère d'insertion, le voisinage de cette solution est ensuite exploré avec un algorithme de recherche locale dans la phase d'amélioration jusqu'à ce qu'un minimum local soit identifié. Un des inconvénients de l'implémentation classique de la méthode GRASP est qu'elle ne prend pas en compte l'information fournie par les solutions précédemment explorées car chaque itération est indépendante de l'autre. Nous proposons une implémentation intégrant l'apprentissage[2] dans le but de diriger la recherche vers des solutions de qualité. En effet, à chaque itération, les critères d'insertion utilisés pour construire une solution réalisable sont mis à jour en prenant en compte les explorations passées.

[2]Une bibliographie sur d'autres versions de GRASP intégrant l'apprentissage peut être trouvée dans l'article de Pitsoulis et Resende [57].

Notre approche est centrée sur trois phases principales : une phase de prétraitement qui sélectionne un sous–ensemble d'interventions à sous–traiter ; une phase d'initialisation qui cherche à identifier de bons critères d'insertion pour la phase de construction ; et une phase de recherche qui utilise GRASP pour trouver la meilleure solution possible.

5.1.2 Notations et modèle mathématique

Dans cette section, nous introduisons un ensemble de notations ainsi que le modèle mathématique du problème.

Les constantes du problème sont les suivantes :

- $H_{\max}$ 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 prédecesseurs de I.
- $Succ(I)$ est la liste des successeurs de I.

Les variables sont les suivantes :

- $s(I)$ 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 3 et un technicien de niveau 2 aura ses demandes notées : $R(I, i, 1) = 2$, $R(I, i, 2) = 2$, $R(I, i, 3) = 1$ et $R(I, i, 4) = 0$.

Pour le modèle mathématique, nous utilisons les constantes suivantes :

- $Pr(k, I)$ égale 1 si la priorité de l'intervention I est k et 0 sinon.
- $\mathcal{P}(I_1, I_2)$ égale 1 si l'intervention I_1 est un prédécesseur de l'intervention I_2 et 0 sinon.

et les variables suivantes :

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

Ce problème, noté (P'), peut être modélisé de la manière suivante :

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

$$\text{sujet à} \quad \sum_{j,h,\epsilon} x(I,j,h,\epsilon) = 1, \qquad \forall I, \tag{5.1}$$

$$y(j,0,t) = 1 - P(t,j), \qquad \forall j,t, \tag{5.2}$$

$$\sum_{\epsilon} y(j,\epsilon,t) = 1, \qquad \forall j,t, \tag{5.3}$$

$$x(I,j,h,0) = 0, \qquad \forall I,j,h, \tag{5.4}$$

$$\sum_{h_1=\max(h_2-T(I_1)+1,0)}^{\min(h_2+T(I_2)-1,H_{\max})} x(I_1,j,h_1,\epsilon) + x(I_2,j,h_2,\epsilon) \leq 1, \quad \forall I_1,I_2,h_2,j,\epsilon, \tag{5.5}$$

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

$$\forall I_1,I_2 \mid \mathcal{P}(I_1,I_2) = 1, \tag{5.6}$$

$$x(I,j,h,\epsilon) = 0 \qquad \forall I,j,h,\epsilon \mid h + T(I) > H_{\max}, \tag{5.7}$$

$$\sum_{h} R(I,i,n)x(I,j,h,\epsilon) \leq \sum_{t|C(t,i)\geq n} y(j,\epsilon,t), \qquad \forall I,i,n,\epsilon,j, \tag{5.8}$$

$$\sum_{j,h,\epsilon} (jH_{\max}+h+T(I))\,Pr(k,I)x(I,j,h,\epsilon) \leq t_k, \quad \forall I, k = 1,2,3, \tag{5.9}$$

$$\sum_{j,h,\epsilon} (jH_{\max}+h+T(I))\,x(I,j,h,\epsilon) \leq t_4, \qquad \forall I. \tag{5.10}$$

La contrainte (5.1) assure que chaque intervention est traitée par une seule équipe à une journée et à une date fixée. La contrainte (5.2) garantit que si un technicien t ne travaille pas le jour j, il est dans l'équipe 0. La contrainte (5.3) spécifie qu'un technicien appartient à une seule équipe chaque jour. La contrainte (5.4) assure qu'aucune intervention n'est réalisée par l'équipe 0. La contrainte (5.5) certifie que deux interventions réalisées le même jour par la même équipe sont effectuées à des dates différentes. La contrainte (5.6) spécifie que tous les prédécesseurs d'une intervention donnée doivent être réalisés avant la date de début de l'intervention. La contrainte (5.7) assure que chaque jour a une durée limite de $H_{\max}$, nombre maximal de portions de temps par jour. La contrainte (5.8) spécifie qu'une équipe qui travaille sur l'intervention I satisfait les demandes en nombre de techniciens par niveau de compétence. Finalement, la contrainte (5.9) spécifie que t_k est la date de fin de la dernière intervention de priorité k, $k = 1,2,3$ et la contrainte (5.10) spécifie que t_4 est la date de fin de l'ordonnancement.

5.2 Méthode de résolution

Dans cette section, nous détaillons les trois phases principales de notre approche : (i) la phase de prétraitement qui consiste à sélectionner un sous–ensemble d'interventions à sous–traiter ; (ii) la phase d'initialisation qui cherche à identifier de bons critères d'insertions pour la construction des solutions ; et (iii) la phase de recherche qui s'appuie sur la métaheuristique GRASP pour trouver la meilleure solution possible. La terminologie GRASP se réfère à une classe de procédures dans laquelle une heuristique gloutonne et 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 [75], les problèmes de routage [1], les problèmes de graphes [59], les problèmes d'affectation [23], les problèmes de planification [75] etc. On peut se référer à [**?**] pour une bibliographie plus complète sur le sujet.

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...) :

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

5.2.1 Heuristique de choix des interventions à sous–traiter

Dans le contexte du challenge, nous devions trouver une méthode efficace dans un temps limité. Pour cela, nous avons opté pour une méthode heuristique qui consiste à choisir un ensemble d'interventions à sous–traiter au début du processus. Une fois cet ensemble déterminé, les interventions sous–traitées ne sont plus prises en compte durant le processus de recherche. Cette heuristique est fondée sur un critère d'insertion lié à un poids spécifique attribué à chaque intervention. Ce poids $\omega(I)$ est établi à partir d'une borne supérieure du nombre minimum de techniciens nécessaires à la réalisation de l'intervention I ($\texttt{mintec}(I)$), et de la durée de celle-ci ($T(I)$). Il est égal au produit de ces deux valeurs : $\omega(I) = \texttt{mintec}(I) \times T(I)$.

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 $\texttt{mintec}(I)$ est calculée en résolvant de manière heuristique le programme linéaire en nombres entiers suivant :

$$\begin{aligned} \text{Minimiser} \quad & \sum_{t \in \Omega_\text{t}} x_t \\ \text{sujet à} \quad & \sum_{t|C(t,I) \geq n, t \in \Omega_\text{t}} x_t \geq R(I,i,n), \quad \forall i,n, \\ & x_t \in \{0,1\}, \quad t \in \Omega_\text{t}. \end{aligned}$$

L'heuristique utilisée pour résoudre ce problème est décrite par l'algorithme 5.1.

Algorithme 5.1 – Calcul de la valeur `mintec`(I) pour une intervention I

```
mintec(I)
{
   ϵ := équipe vide ;
   T := sous-ensemble de techniciens satisfaisant au moins une demande de I ;
   for(chaque technicien t dans T) {
      sk(t,I) := nombre de niveaux de compétence de t satisfaisant
      les demandes de I pour tous les domaines ;
   }
   réarranger T selon l'ordre décroissant des valeurs sk(t,I) ;
   while(ϵ ne satisfait pas les demandes de I) {
      ajouter un nouveau technicien t ∈ T à ϵ ;
      mettre à jour T := T − {t} ;
   }
   for(chaque paire de techniciens {t,t'} dans ϵ)
      for(chaque technicien t'' de T)
         if({t,t'} peut être remplacé par {t''} pour satisfaire les compétences)
            remplacer {t,t'} par {t''} dans ϵ ;
            return nombre de techniciens dans ϵ ;
      }
   }
}
```

Soit $\omega(I)$ 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, ce qui correspond à un problème de sac à dos avec contraintes de précédence [44]. Ce problème, noté PCKP (pour *Precedence Constraint Knapsack Problem*), peut être formulé de la manière suivante :

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

Il 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.

Bien que ce choix de gestion des interventions à sous–traiter ne soit pas optimal, 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.

5.2.2 Phase de construction

Dans une implémentation classique de la métaheuristique GRASP, la phase de construction consiste à déterminer un ensemble d'éléments candidats pouvant être ajoutés à la solution partielle courante tout en maintenant la réalisabilité. La sélection de l'élément à incorporer est déterminée selon un poids : les éléments correspondant aux poids les plus forts sont insérés en priorité. Les poids sont donnés par un algorithme glouton qui évalue l'augmentation de la fonction objective qu'engendre l'insertion des candidats. Dans notre cas, ils sont tout d'abord initialisés à une valeur donnée (liée au coût de la priorité de l'intervention dans la fonction objective) et mis à jour à chaque itération en prenant en compte les solutions précédemment rencontrées. La méthodologie GRASP classique considère une liste restrictive de candidats (RCL pour *restricted candidate list*) composée des α % des candidats ayant les poids les plus élevés, $\alpha \in [0, 100]$. Les candidats sont alors choisis aléatoirement dans la RCL afin de les insérer dans la solution partielle courante. Dans notre approche, la RCL est composée de *toutes* les interventions, ce qui correspond à fixer la variable α à la valeur 100. Les interventions sont insérées selon l'ordre décroissant des poids et sont choisies aléatoirement en cas d'égalité.

Sélection d'un candidat

Initialement, le poids d'un candidat est fixé à la valeur du coefficient de sa priorité dans la fonction objective 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. D'autre part, un candidat ne peut être sélectionné 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 l'intervention 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 insérée 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 nécessaire à la construction d'une nouvelle équipe 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 notons $techs^{\epsilon}(I)$ 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)

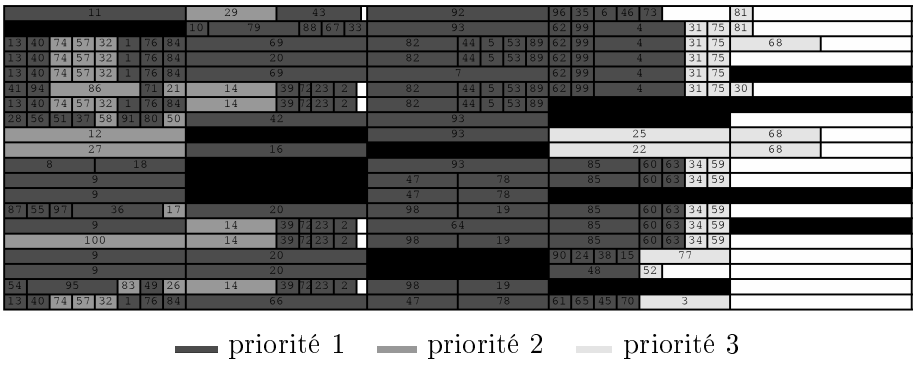
L'ordre des critères (2a) et (2b) dépend de la valeur de $d(I)$ obtenue précédemment. Il dépend également de la date de fin de la dernière intervention de même priorité que I pour les priorités 1, 2, 3 et de la date de fin totale pour la priorité 4 (i.e. t_i, où i désigne la priorité du candidat I). 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 de la date de fin de la dernière intervention de même priorité que I, alors elle n'influe pas sur la valeur de la fonction objective. Dans ce cas, nous cherchons à minimiser 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 ayant la valeur minimale $techs^{\epsilon}(I)$ s'il existe au moins deux équipes avec la même valeur $s^{\epsilon}(I)$.

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

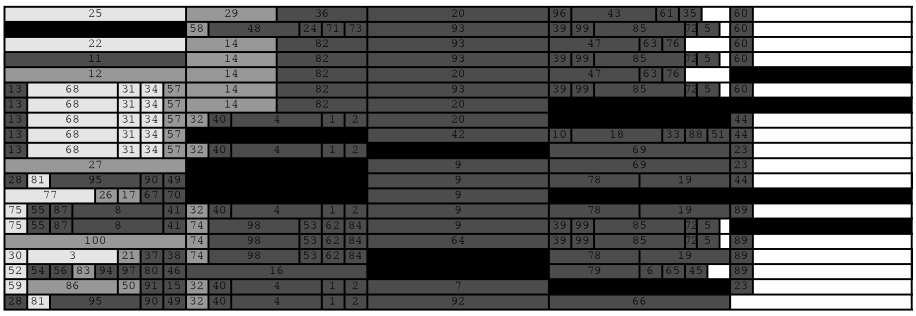
Des expériences préliminaires nous ont montré que le critère de choix des interventions pour l'algorithme glouton n'est pas trivial et, en particulier, qu'il ne correspond pas toujours à l'ordre des coefficients des priorités dans la fonction objective. Notons $w(I)$ le poids de l'intervention I. Supposons que $w(I)$ soit égal au coefficient de la priorité de I dans la fonction objective. Cela revient à choisir les interventions de haute priorité d'abord. La figure 5.1 représente une solution générée par l'algorithme glouton dans ces conditions. Les poids des interventions sont : 28 pour les interventions de priorité 1 ; 14 pour les interventions de priorité 2 ; 4 pour les interventions de priorité 3 ; 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.

Fig. 5.1 – Solution avec un objectif de 17820 pour l'instance 8 de l'ensemble data-setA



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 5.2 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.

FIG. 5.2 – Solution avec un objectif de 17355 pour l'instance 8 de l'ensemble data-setA



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 le cas de figure où il n'y a pas d'intervention de priorité 4.

Mise à jour des poids

Un des inconvénients potentiels de l'architecture GRASP standard est qu'elle ne prend pas en compte les solutions précédemment visitées. Dans l'implémentation proposée, nous utilisons l'information fournie par les solutions précédentes pour diriger la recherche vers des zones potentiellement « prometteuses ». Ceci est réalisé, à chaque itération, en mettant à jour les poids des candidats en considérant les caractéristiques des solutions précédentes. Notons $w_p(I)$, le poids associé à l'intervention I selon la permutation p des poids associés aux priorités. 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. À la fin de l'exécution de l'algorithme glouton, les poids des interventions sont mis à jour à partir de l'information fournie par la solution générée [65]. Cette mise à jour consiste à ajouter la valeur $w_p(I)$ aux dernières interventions de chaque priorité et à tous leurs prédécesseurs de manière à planifier ces interventions plus tôt à l'itération suivante. L'algorithme 5.2 illustre cette procédure.

Algorithme 5.2 – Phase de mise à jour

```
mise_a_jour_poids(p)
{
   for(chaque priorité prio) {
      I := dernière intervention planifiée de priorité prio ;
      w(I) = w(I) + w_p(I) ;
      for(chaque intervention J ∈ Pred(I))
         w(J) = w(J) + w_p(I) ;
   }
}
```

5.2.3 Phase d'initialisation

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 sur les 12 permutations qui donnent les meilleures valeurs.
- Répéter la même procédure 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.

5.2.4 Phase d'amélioration

Dans cette section, nous décrivons un algorithme de recherche locale qui explore le voisinage des solutions rencontrées en vue d'une amélioration.

Après avoir affecté les interventions aux équipes et déterminé l'ordre dans lequel les interventions sont traitées pour chaque équipe, nous vérifions la faisabilité de l'ordonnancement. De plus, les dates de début optimales des interventions sont 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 [16].

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 leur sont assignées.

Une opération d'échange consiste à intervertir l'affectation et l'ordre de deux interventions et une opération d'insertion supprime une intervention et l'insère à une autre position temporelle.

Phase *critical path*

Le but de la phase *critical path* est de réduire les dates de fin de chaque priorité et la date de fin 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 insérée si l'intervention I_k n'est pas insérée 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.

5.2.5 Schéma général de l'approche de résolution

L'algorithme 5.3 résume les trois phases 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 phase d'initialisation consistant à assigner des poids initiaux aux interventions est décrite par la fonction `initialisation_poids`. Cette procédure fournit $conf_1$ et $conf_2$, qui correspondent aux deux configurations initiales des poids utilisées dans la procédure GRASP. La phase GRASP consiste à répéter l'exécution de l'algorithme glouton avec les deux configurations des poids 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 5.3 – Algorithme général

```
résoudre_TIST(PB,MAX_CPU)
{
   meilleure_solution = ∅ ;
   SPB = glouton_sous_traités(PB) ;
   (conf1, conf2) = initialisation_poids(SPB) ;
   while(MAX_CPU n'est pas atteint) {
      solution_1 = construction_gloutonne(SPB, conf1) ;
      solution_2 = construction_gloutonne(SPB, conf2) ;
      Mettre à jour la mémoire ;
      amélioration = mise_a_jour(solution_1,solution_2,meilleure_solution) ;
      if(amélioration = true)
         meilleure_solution = recherche_locale(SPB, meilleure_solution) ;
   }
}
```

5.2.6 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.

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.
- $MSP(2)$ avec uniquement les interventions de priorité 2 et leurs prédécesseurs.
- $MSP(3)$ avec uniquement les interventions de priorité 3 et leurs prédécesseurs.

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

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 makespans 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 respectivement les dates de fin des priorités 1, 2 et 3, et t_4 est la date de fin de l'ordonnancement global. Toute solution réalisable 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.

Borne inférieure par boîtes

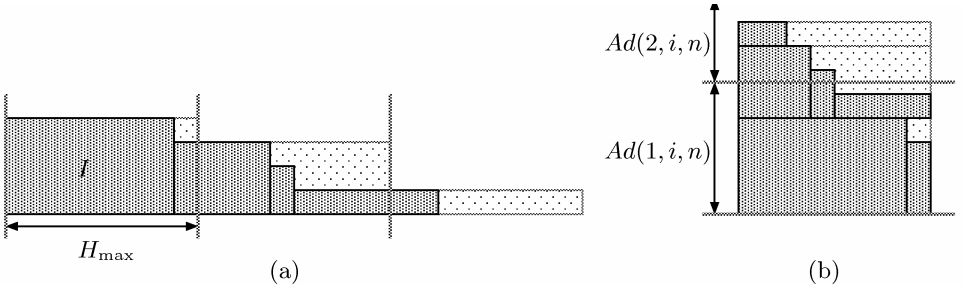
La borne inférieure par boîtes, qui est une borne inférieure sur le nombre de jours de l'ordonnancement, est calculée pour chaque domaine i et niveau de compétence n ; la plus grande valeur est conservée. La même méthode a été proposée par Lodi, Martello et Vigo [47] pour le problème *two-dimensional 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 $T(I)$.

Considérons tous les rectangles insérés bout à bout par hauteur décroissante comme dans la figure 5.3 (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 une largeur égale à $H_{\max}$ et une hauteur égale au $R(I,i,n)$ de la première intervention I qui les constitue. Ils sont ensuite superposés pour en constituer un unique. Cette opération est illustrée par la figure 5.3 (b) où $Ad(j,i,n)$ est le nombre de techniciens qui peuvent travailler le jour j et dont le niveau de compétence dans le domaine i est n.

Fig. 5.3 – Illustration du calcul de la borne inférieure par boîtes



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 5.3 (b) illustre une situation où $\mu^* = 2$.

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 (date de fin de l'ordonnancement) 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) :

$$\text{Minimiser} \quad M \tag{5.11}$$

$$\text{sujet à} \quad \sum_{t|C(t,i)\geq n} x_{I,t} \geq R(I,i,n), \qquad \forall I, \forall i, \forall n \tag{5.12}$$

$$\sum_{I} T(I) x_{I,t} \leq U_\mu(t,M), \qquad \forall t, \tag{5.13}$$

$$0 \leq x_{I,t} \leq 1 \qquad \forall I, \forall t, \tag{5.14}$$

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

Si le problème n'a pas de solutions réalisables ou si 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.

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 d'une séquence d'interventions liées par des contraintes 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.

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 makespans 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.

5.2.7 Résultats préliminaires

Nous présentons dans cette section les résultats obtenus sur l'ensemble des données fournies par *France Télécom* pour ce challenge. Il y a trois ensembles de données disponibles, chaque ensemble contenant 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 (notées Ai) 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 (notées Bi) 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 (notées Xi) 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 le tableau 5.1, 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 : *Pb* : 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 : `BS` : 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 objective. La colonne *Meilleur objectif* a deux valeurs : `BS` : la meilleure valeur de l'objectif trouvée parmi tous les participants au challenge et *gap* : l'écart relatif entre cette meilleure valeur et la valeur que notre méthode a trouvée.

Tab. 5.1 – Résultats officiels obtenus lors du challenge

					GRASP			Meilleur objectif	
Pb	int.	tec.	dom.	lev.	BS	BI	gap.	BS	gap
A1	5	5	3	2	2340	2265	3.2	2340	0
A2	5	5	3	2	4755	4215	11.35	4755	0
A3	20	7	3	2	11880	11310	4.79	11880	0
A4	20	7	4	3	13452	10995	18.26	13452	0
A5	50	10	3	2	28845	26055	9.67	28845	0
A6	50	10	5	4	18870	17775	5.8	18795	0.39
A7	100	20	5	4	30840	27405	11.13	30540	0.97
A8	100	20	5	4	17355	16166	6.85	16920	2.50
A9	100	20	5	4	27692	25618	7.48	27692	0
A10	100	15	5	4	40020	35405	11.53	38296	4.3
						Moyenne	**9.01**		**0.81**
B1	200	20	4	4	43860	38385	12.48	34395	21.58
B2	300	30	5	3	20655	16605	19.6	15870	23.16
B3	400	40	4	4	20565	17460	15.09	16020	22.1
B4	400	30	40	3	26025	19035	26.85	25305	2.76
B5	500	50	7	4	120840	106290	12.04	89700	25.76
B6	500	30	8	3	34215	24450	28.54	27615	19.28
B7	500	100	10	5	35640	28470	20.11	33300	6.56
B8	800	150	10	4	33030	32820	0.63	33030	0
B9	120	60	5	5	29550	26310	10.96	28200	4.56
B10	120	40	5	5	34920	32790	6.09	34680	0.68
						Moyenne	**15.24**		**12.64**
X1	600	60	15	4	181575	140025	22.88	151140	16.76
X2	800	100	6	6	7260	6840	5.78	7260	0
X3	300	50	20	3	52680	49650	5.75	50040	5.01
X4	800	70	15	7	72860	59560	18.25	65400	10.23
X5	600	60	15	4	172500	126465	26.68	147000	14.78
X6	200	20	6	6	9480	6180	34.81	9480	0
X7	300	50	20	3	46680	45000	3.59	33240	28.79
X8	100	30	15	7	29070	20590	29.17	23640	18.67
X9	500	50	15	4	168420	101985	39.44	134760	19.98
X10	500	40	15	4	178560	99705	44.16	137040	23.25
						Moyenne	**23.05**		**13.74**

Un total de 17 équipes participaient 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.

Le tableau 5.1 montre que l'écart entre la solution trouvée par notre algorithme et la borne inférieure est important pour certaines instances de 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 accélérer la phase 2 de notre approche, qui est en cause ici, soit en allégeant la procédure d'échantillonnage décrite en section 5.2.3 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). Nous allons voir dans la section suivante que l'amélioration de la procédure de choix de ces interventions permet d'améliorer les performances de notre algorithme.

5.3 Étude sur le choix des interventions à sous–traiter

Dans cette section, nous présentons une étude réalisée sur le problème du choix des interventions à sous–traiter. Rappelons que le problème consiste à sélectionner un sous–ensemble d'interventions de manière à ne pas dépasser le budget alloué et tel que tous les successeurs d'une intervention sous–traitée soient sous–traités. Une question majeure est l'évaluation de la pertinence de sous–traiter une intervention plutôt qu'une autre, et plus généralement de sous–traiter un sous–ensemble d'interventions plutôt qu'un autre. Nous présentons ici quelques pistes qui ont été étudiées sur ce problème et, en particulier, les expérimentations faites avec Resolution search qui nous ont permis d'améliorer les résultats précédemment obtenus.

5.3.1 Résolution exacte du sac à dos avec contraintes de précédence

Une première variante de l'algorithme initial consiste à remplacer la résolution heuristique du problème de sac à dos avec contraintes de précédence (cf. section 5.2.1) par une résolution exacte en utilisant CPLEX 9.2. Les résultats obtenus par cette version n'ont pas été globalement encourageants. Le tableau 5.2 présente un récapitulatif des résultats obtenus en comparaison avec la version heuristique utilisant l'algorithme glouton pour sélectionner les interventions à sous–traiter.

Dans ce tableau, les colonnes *BI* donnent les bornes inférieures obtenues avec le sous–ensemble d'interventions sous–traitées donné par l'heuristique gloutonne et celui donné par la résolution exacte du PCKP. Les colonnes *BS* donnent les bornes supérieure correspondantes. Les valeurs en gras indiquent les meilleures bornes supérieures trouvées parmi les deux méthodes.

Tab. 5.2 – Résultats obtenus par la résolution exacte du PCKP

Pb	Heuristique gloutonne		Méthode exacte		Pb	Heuristique gloutonne		Méthode exacte	
	BI	BS	BI	BS		BI	BS	BI	BS
B1	38385	**43860**	38835	44670	X1	140025	**181575**	140025	**181575**
B2	16605	20655	15930	**20640**	X2	6480	**7260**	6840	**7260**
B3	17460	**20565**	17535	21060	X3	49650	**52680**	50520	55740
B4	19035	**26025**	18690	26505	X4	59560	72860	59570	**71640**
B5	106290	120840	107670	**120630**	X5	126465	**172500**	126465	**172500**
B6	24450	**34215**	24390	34440	X6	6180	**9480**	7020	10740
B7	28470	**35640**	28080	35910	X7	45000	**46680**	45870	46920
B8	32820	**33030**	32820	**33030**	X8	20590	29070	20410	**27840**
B9	26310	**29550**	26310	**29550**	X9	101985	**168420**	103035	168660
B10	32790	**34920**	34470	36600	X10	99705	178560	100950	**176280**

Pour trois instances, les sous–ensembles d'interventions sous–traitées sont les mêmes (B9, X1 et X5). Nous pouvons voir qu'en moyenne, les résultats finaux obtenus sont moins bons du point de vue de la qualité de la borne supérieure finale. Ces résultats nous conduisent donc à chercher dans une autre direction.

5.3.2 Énumération et évaluation des sous–ensembles maximaux

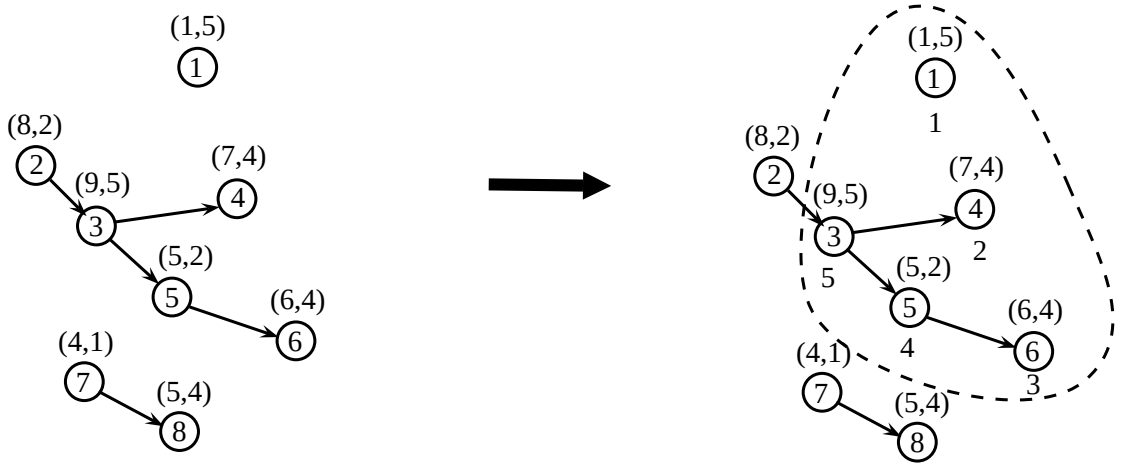
La seconde stratégie envisagée vise à énumérer une série de sous–ensembles maximaux d'interventions (au sens du budget) qui peuvent être sous–traitées, et les évaluer de manière à choisir le « meilleur ». L'évaluation d'un sous–ensemble d'interventions n'est pas triviale. L'approche envisagée consiste à utiliser le calcul de la borne inférieure (section 5.2.6) sur chaque sous–ensemble et à conserver le sous–ensemble générant la plus petite borne inférieure.

Après avoir déterminé les différentes composantes connexes du graphe formé par les interventions de l'instance, une stratégie récursive est mise en place pour parcourir toutes les interventions. Une intervention donnée peut être ajoutée au sous–ensemble courant si son coût n'excède pas le budget restant et si tous ses successeurs sont dans le sous–ensemble courant. L'algorithme commence par considérer les interventions « isolées » (i.e. sans prédécesseur ni successeur), puis il considère chaque composante connexe en commençant par les interventions n'ayant pas de successeurs. L'énumération se base également sur le critère d'insertion utilisé dans l'approche heuristique présentée en section 5.2.1, c'est–à–dire qu'elle va chercher à insérer en priorité les interventions ayant un plus grand ratio $\omega(I)$ défini par

$$\omega(I) = \frac{\texttt{mintec}(I) \times T(I)}{cost(I)}.$$

La figure 5.4 illustre la façon dont l'algorithme génère les sous–ensembles maximaux. On suppose dans cet exemple que le budget alloué est $A = 20$. Le graphe de gauche représente un ensemble d'interventions telles que les valeurs (*val1*,*val2*) sont respectivement le ratio $\omega(I)$ et le coût $cost(I)$, tandis que les flèches représentent les relations de précédence.

FIG. 5.4 – Principe de création des sous–ensembles maximaux par l'énumération



L'algorithme commence tout d'abord par insérer l'intervention 1 car elle n'a ni prédécesseur ni successeur. Il insère ensuite l'intervention 4, car parmi les interventions n'ayant pas de successeurs, elle correspond à celle ayant le plus grand ratio $\omega(I)$. L'intervention qui vient ensuite est l'intervention 6 car tous les successeurs de l'intervention 3 ne sont pas sous–traités et l'intervention 8 a une valeur $\omega(I)$ inférieure. Viennent ensuite les interventions 5 et 3 car l'intervention 2 ne peut être insérée à cause du manque de budget disponible. Lorsqu'un sous–ensemble maximal est identifié, il est évalué en calculant la borne inférieure sur les interventions non sous–traitées. L'ensemble donnant la plus petite borne inférieure constitue l'ensemble des interventions sous–traitées pour l'algorithme GRASP. Le tableau 5.3 expose les résultats obtenus en exécutant l'énumération durant 600 secondes.

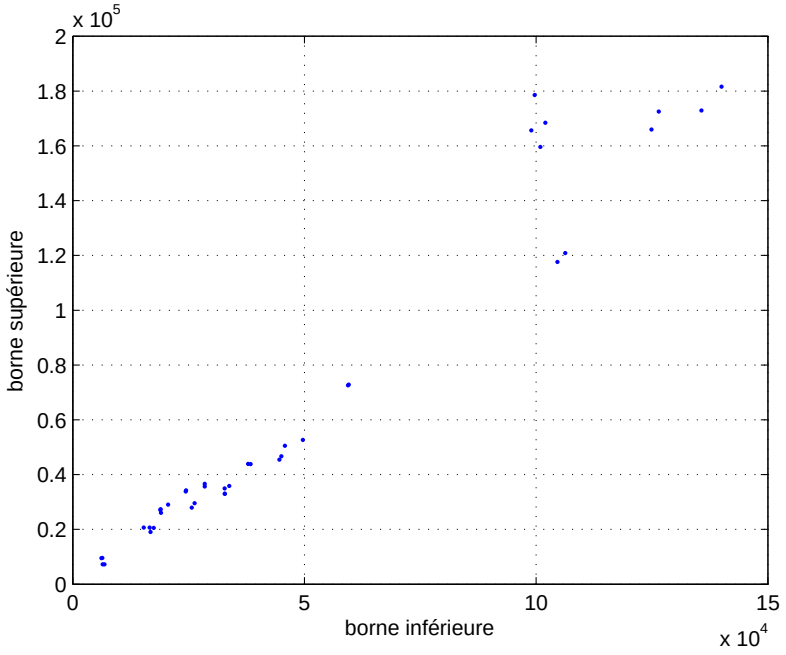
TAB. 5.3 – Résultats obtenus par l'énumération des sous–ensembles maximaux

Pb	Glouton		Enumération			Pb	Glouton		Enumération		
	BI	BS	BI	t(s)	BS		BI	BS	BI	t(s)	BS
B1	38385	**43860**	**37845**	401	43890	X1	140025	181675	**135675**	402	**172920**
B2	16605	20655	**15315**	0	**20640**	X2	6480	7260	6840	0	7260
B3	17460	20565	**16800**	3	**19065**	X3	49650	52680	**45780**	125	**50520**
B4	19035	**26025**	**18975**	1	27330	X4	59560	72860	**59390**	1	**72570**
B5	106290	120840	**104610**	53	**117630**	X5	126465	172500	**124875**	271	**165960**
B6	24450	34215	**24390**	5	**33870**	X6	**6180**	**9480**	6390	452	9600
B7	**28470**	**35640**	28500	0	36660	X7	45000	46680	**44580**	265	**45480**
B8	32820	33030	32820	1	33030	X8	20590	29070	**18890**	5	**27120**
B9	26310	29550	**25695***	40	**27960**	X9	101985	168420	**100920**	16	**159600**
B10	**32790**	**34920**	33765	490	35880	X10	99705	178560	**98970**	342	**165660**

Cette procédure permet d'améliorer 13 valeurs par rapport aux valeurs obtenues avec l'heuristique gloutonne. La valeur encadrée correspond à une amélioration de la meilleure borne supérieure connue. La valeur de borne inférieure avec une étoile signifie que tous les ensembles maximaux ont été énumérés et que cette valeur correspond à l'optimum.

Le calcul de la borne inférieure semble être un bon critère d'évaluation des ensembles d'interventions à sous–traiter car, excepté pour l'instance B4, on observe une corrélation entre la borne inférieure et la borne supérieure. La figure 5.5 expose les différentes valeurs de la borne supérieure en fonction de la borne inférieure dans les résultats obtenus jusqu'à présent.

Fig. 5.5 – Corrélation entre les bornes inférieures et les bornes supérieures trouvées



5.3.3 Enumération des sous–ensembles maximaux par Resolution search

La dernière voie explorée a été l'utilisation de Resolution search pour déterminer l'ensemble d'interventions à sous–traiter. L'expérimentation menée se fonde sur les mêmes principes que la méthode d'énumération présentée précédemment : énumérer des sous–ensembles maximaux et les évaluer en calculant la borne inférieure du sous–problème associé. Notre intuition était que Resolution search permette d'énumérer de manière plus pertinente ces sous–ensembles et ainsi de générer plus rapidement des sous–ensembles donnant une borne inférieure de qualité.

Principe d'exploration

Dans l'implémentation proposée, Resolution search cherche à générer des ensembles maximaux en affectant des valeurs 0 ou 1 à un vecteur u de taille n, n étant le nombre d'interventions.

Si l'intervention I est sous–traitée, $u_I = 1$, si l'intervention I n'est pas sous–traitée, $u_I = 0$ et si le sort de l'intervention I n'est pas décidé, $u_i = *$. Le rôle de la fonction `obstacle` est de trouver un ensemble maximal et d'appeler la fonction `oracle` qui, dans ce cas, consiste à calculer la borne inférieure associée. L'obstacle S représente alors l'ensemble maximal courant et il est ajouté à la famille path–like $\mathscr{F}$ de manière à explorer de nouveaux ensembles dans les itérations futures. De même que pour la méthode d'énumération présentée dans la section précédente, Resolution search explore l'espace des solutions de manière à favoriser l'insertion des interventions isolées et des interventions n'ayant pas de successeurs dans les composantes connexes, tout en prenant en compte un critère d'insertion basé sur un poids attribué à chaque intervention. Comme nous l'avons mentionné dans le chapitre 3, Resolution search prend deux différents critères en considération : (i) un critère d'insertion lors de la phase de descente qui consiste à déterminer quelle intervention sera insérée et (ii) un critère de remise en cause lors du choix du littéral w dans la mise à jour de $\mathscr{F}$ qui consiste à déterminer quelle décision sera remise en cause dans les itérations futures. Nous montrons que l'utilisation de ces deux critères permet d'explorer l'espace des solutions de manière plus adaptée.

Critères d'insertion et de remise en cause

On note $\psi(I)$ le poids d'insertion associé à une intervention I et $\theta(I)$ son poids de remise en cause. Dans les deux cas (insertion ou remise en cause) les interventions sont choisies par ordre décroissant des poids associés. Lors de la création des sous–ensembles, on favorise l'insertion des interventions ayant le moins de successeurs, car la sous–traitance d'une intervention implique la sous–traitance de tous ses successeurs ce qui peut limiter a priori l'insertion d'autres interventions. De plus, on favorise les interventions ayant un ratio $\omega(I)$ important. Le poids d'insertion $\psi(I)$ peut être défini de la manière suivante :

$$\psi(I) = \frac{\omega(I)}{|Succ(I)| + 1}$$

Afin de donner une priorité supplémentaire aux interventions isolées, leur poids a une valeur double :

$$\psi(I) = 2 \cdot \omega(I) \qquad \forall I \text{ tel que } |Pred(I) = 0| \text{ et } |Succ(I) = 0|.$$

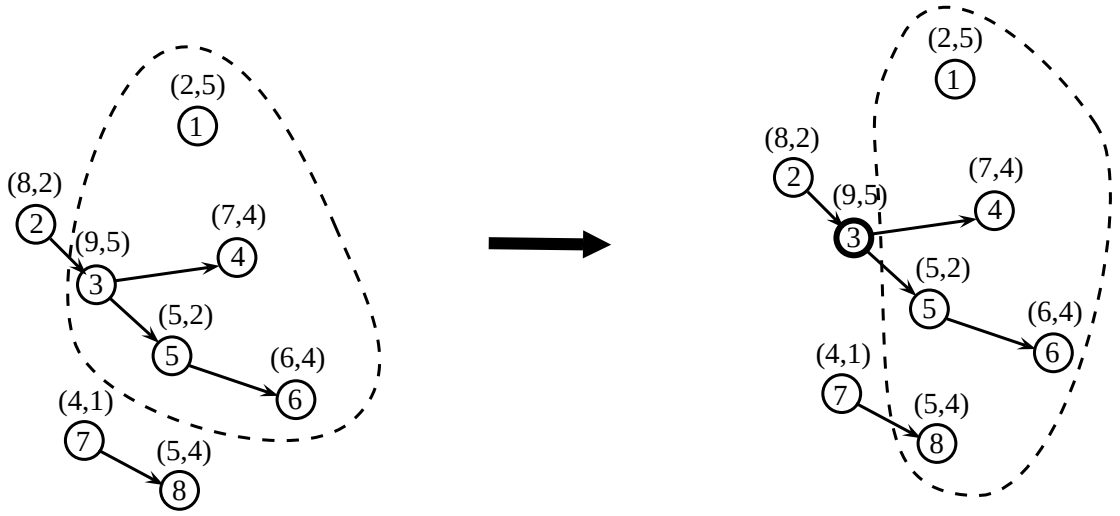
Lorsqu'un ensemble maximal est identifié, il est ajouté à la famille $\mathscr{F}$ sous forme de clause. Cette clause comporte : des instanciations $\bar{x}_I$ signifiant que l'intervention I fait partie de l'ensemble ; des instanciations x_I signifiant que l'intervention I ne doit pas être insérée ; et des variables non instanciées ne faisant pas partie de l'ensemble mais qui pourront être instanciées dans les itérations futures. Durant la mise à jour de $\mathscr{F}$, nous avons vu dans le chapitre 3 qu'il y a deux cas à distinguer : soit $S \not\sqsubseteq \mathscr{F}$ soit $S \sqsubseteq \mathscr{F}$. Dans le premier cas, $R = S$ est simplement ajoutée à $\mathscr{F}$ et dans le deuxième cas, S est simplifiée par résolvantes successives sur certaines clauses de $\mathscr{F}$ pour donner la clause R qui sera insérée dans $\mathscr{F}$.

On distingue deux cas de figure pour la remise en cause d'une instanciation : soit I appartient à l'ensemble et $\bar{x}_I \in R$, soit I n'appartient pas à l'ensemble et $x_I \in R$. Dans les deux cas on choisit un littéral w appartenant à R qui sera fixé à sa valeur opposée dans le prochain vecteur $u(\mathscr{F})$. On préfère choisir un littéral w à valeur 1 dans R ($\bar{x}_I \in R$) pour qu'il soit fixé à 0 dans $u(\mathscr{F})$ de manière à supprimer une intervention de l'ensemble plutôt qu'en ajouter une nouvelle (car l'ensemble est déjà maximal). On préfère également un littéral correspondant à une intervention de faible poids $\psi(I)$ et ayant le moins de prédécesseurs car si l'intervention est retirée de l'ensemble, tous ses prédécesseurs doivent l'être également. Dans le cas où toutes les interventions sont à valeur 0 dans R ($\bar{x}_I \notin R$ pour tout I), le critère de remise en cause est alors directement lié au critère d'insertion, car dans ce cas, choisir w correspond à insérer l'intervention w dans $u(\mathscr{F})$. Si l'on prend en compte les différents critères que l'on vient d'énumérer, le poids de remise en cause $\theta(I)$ lié à une intervention I peut être défini de la manière suivante :

$$\theta(I) = \begin{cases} 1/(\psi(I) + |Pred(I)| + 1) & \text{si } \bar{x}_I \in R \\ -1/\psi(I) & \text{si } x_I \in R \end{cases}$$

Nous allons illustrer la différence de comportement entre l'énumération et Resolution search pour la génération des ensembles maximaux. Comme dans l'exemple précédent, on considère que le budget alloué pour la sous–traitance est $A = 20$. Le schéma de gauche de la figure 5.6 illustre le choix fait par l'énumération pour la création d'un sous–ensemble maximal tel que décrit précédemment.

Fig. 5.6 – Remise en cause du choix des interventions dans le cas de l'énumération



L'intervention 1 étant isolée, elle est choisie en priorité, ensuite les interventions 4 et 6 ayant la plus grande valeur $\omega(I)$ sont sélectionnées et leur prédécesseurs sont insérés jusqu'à ce qu'il ne soit plus possible de rajouter d'interventions. Une fois l'ensemble maximal

$\{1, 3, 4, 5, 6\}$ identifié, la borne inférieure est calculée puis un autre ensemble est généré en supprimant ou en insérant des interventions. Le nouvel ensemble généré par l'énumération est représenté par le schéma de droite. Bien que l'intervention 3 ait un poids $\omega(I)$ important, elle est supprimée de l'ensemble maximal car elle correspond à la dernière intervention insérée précédemment. L'intervention 8, qui semble a priori moins intéressante au regard du critère $\omega(I)$, est insérée pour donner l'ensemble $\{1, 4, 5, 6, 8\}$. On constate ici que le choix de supprimer l'intervention 3 n'est pas très judicieux car, d'une part, cette intervention possède un critère $\omega(I)$ important et, d'autre part, sa suppression empêche l'insertion de son prédécesseur, l'intervention 8, représentant elle aussi un poids $\omega(I)$ important.

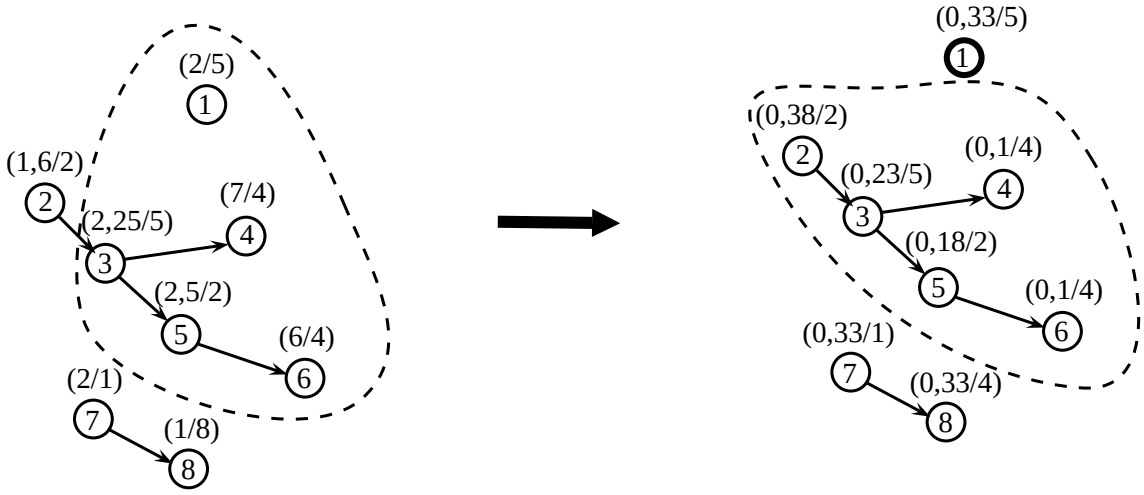
Dans le cas d'une énumération par Resolution search, le calcul des poids $\psi(I)$ et $\theta(I)$ nous donne le tableau 5.4 suivant :

TAB. 5.4 – Poids d'insertion et de remise en cause associés aux interventions

I	1	2	3	4	5	6	7	8
$\psi(I)$	2	1,6	2,25	7	2,5	6	2	1
$\theta(I)$	0,33	0,38	0,23	0,1	0,18	0,1	0,33	0,33

La figure 5.7 illustre le comportement de Resolution search pour la génération des ensembles maximaux.

FIG. 5.7 – Remise en cause du choix des interventions dans le cas de Resolution search



Les valeurs (*val1*,*val2*) correspondent respectivement à $\psi(I)$ et *cost*(I) dans le schéma de gauche et à $\theta(I)$ et *cost*(I) dans le schéma de droite. Le premier ensemble maximal (schéma de gauche) correspond au même ensemble que celui généré par l'énumération.

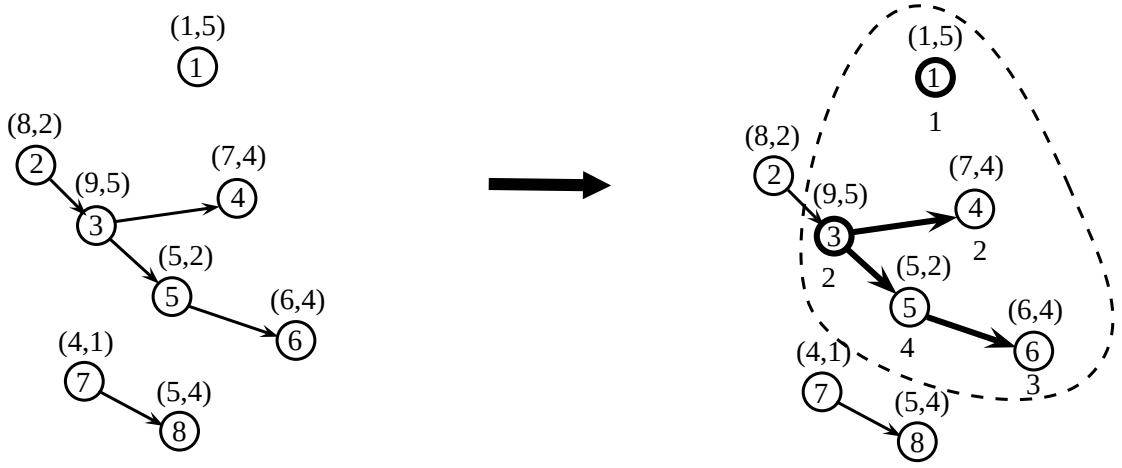
Le deuxième ensemble généré (schéma de droite) est différent : le critère $\theta(I)$ maximal étant celui de l'intervention 1 ($\theta(1) = 0,33$ alors que $\theta(3) = 0,23$), c'est cette intervention qui est supprimée de l'ensemble maximal. L'intervention 2 de valeur $\psi(2) = 1,6$ est alors ajoutée à l'ensemble ce qui nous donne l'ensemble $\{2,3,4,5,6\}$ qui globalement semble meilleur (selon le critère ω) que l'ensemble $\{1,4,5,6,8\}$ généré par l'énumération.

Les critères ψ ou θ n'étant pas optimaux, il se peut que dans certains cas, l'énumération donne de meilleurs sous–ensembles que Resolution search. L'exemple proposé illustre cependant bien le fait que Resolution search permet d'effectuer une recherche plus fine qui ne cantonne pas la remise en cause des choix à l'ordre inverse d'instanciation et peut se baser sur d'autres critères heuristiques.

Phase de remontée implicite

Nous montrons qu'il est possible, dans la résolution de ce problème, d'implémenter la phase de remontée implicite décrite dans le chapitre 4. Reprenons l'exemple précédent en le modifiant légèrement : supposons que le poids $\psi(3)$ soit égal à 8 au lieu de $2,25$. La première intervention insérée est donc l'intervention 3. On peut déduire alors par implication que les interventions 4,5 et 6 doivent également être insérées. Comme nous l'avons démontré dans la proposition 4.1 du chapitre 4, puisque $\bar{x}_3 \Rightarrow \bar{x}_4\bar{x}_5\bar{x}_6$ alors $S = \bar{x}_3$. L'intervention qui suit est l'intervention 1 de poids $\psi(1) = 2$ et l'ensemble {1,3,4,5,6} résultant est maximal. L'ensemble maximal généré est alors représenté par la clause $S = \bar{x}_1\bar{x}_3$ et la remise en cause ne porte que sur x_1 ou x_3. Il est en effet inutile d'essayer de supprimer les interventions 4, 5 ou 6 tant que l'intervention 3 est dans l'ensemble. L'application de la phase de remontée implicite permet ici d'identifier plus rapidement quels choix méritent d'être remis en cause ou non. La figure 5.8 illustre le fonctionnement de la phase de remontée implicite dans la construction d'un ensemble maximal.

Fig. 5.8 – Illustration de la phase de remontée implicite



Résultats expérimentaux

Les résultats expérimentaux obtenus en exécutant Resolution search durant 600 secondes sont exposés dans le tableau 5.5. Ces résultats montrent que Resolution search se comporte mieux de manière générale que l'énumération de type backtracking pour trouver un sous–ensemble générant une borne inférieure de qualité. Dans ce tableau, `BI` correspond à la meilleure borne inférieure trouvée, t(s) correspond au temps cpu en secondes mis pour trouver la borne et `BS` correspond à la valeur de l'objectif trouvé par l'algorithme GRASP après 1200 secondes. La colonne *meilleures valeurs* correspond aux meilleures valeurs obtenues jusqu'à présent avec l'heuristique gloutonne ou l'énumération et *Resolution search* correspond aux valeurs trouvées par Resolution search.

TAB. 5.5 – Résultats obtenus par Resolution search

Pb	meilleures valeurs		Resolution search			Pb	meilleures valeurs		Resolution search		
	BI	BS	BI	t(s)	BS		BI	BS	BI	t(s)	BS
B1	37845	43860	**36525**	172	**42180**	X1	**135675**	**172920**	**135675**	401	**172920**
B2	15315	20640	**15225**	0	**20340**	X2	**6480**	**7260**	**6840**	0	**7260**
B3	**16800**	**19065**	16965	0	19410	X3	45780	50520	**45360**	308	**50280**
B4	18975	**26025**	**18915**	1	26265	X4	59390	72570	**59380**	64	**71560**
B5	**104610**	**117630**	106230	113	120030	X5	**124875**	**165960**	**124875**	42	**165960**
B6	**24390**	33870	**24390**	0	**33390**	X6	**6180**	9480	**6180**	36	**9390**
B7	28470	35640	**27930**	286	**35280**	X7	44580	45480	**43320**	460	**43680**
B8	**32820**	**33030**	**32820**	0	35760	X8	18890	27120	**18830**	413	**25680**
B9	**25695***	**27960**	**25695***	5	**27960**	X9	**100920**	**159600**	**100920**	16	164400
B10	**32790**	**34920**	**32790**	0	35040	X10	98970	165660	**98010**	525	**160920**

Sur les 20 instances testées, Resolution search améliore strictement 9 bornes inférieures par rapport à l'heuristique gloutonne et l'énumération et trouve au total 18 meilleures bornes inférieures. Les valeurs encadrées correspondent à une amélioration des meilleures valeurs connues et la valeur avec une étoile correspond à l'optimum. On remarque que dans le cas d'une égalité avec l'énumération, Resolution search est toujours plus rapide pour trouver le meilleur ensemble maximal. Dans certains cas, malgré une borne inférieure plus basse ou égale, les valeurs de la fonction objective sont moins bonnes. Ceci est dû aux raisons suivantes : (i) l'évaluation d'un sous–ensemble par le calcul de la borne inférieure n'est pas optimal, il se peut donc qu'un ensemble donnant une borne inférieure plus basse donne une valeur optimale de moins bonne qualité (cf. instances B4 et B5) et (ii) dans certains cas, deux sous–ensembles différents donnent des bornes de même valeur et donc peuvent amener à une valeur de l'objectif différente (cf. instances X9, B6 et B10).

Utilisation d'une relaxation du problème

Une autre méthode expérimentée consiste à utiliser une relaxation du problème pour identifier les poids d'insertion $\psi(I)$. Considérons les variables de décision $x_{I,t} = 1$ si l'intervention I est affectée au technicien t, et $y_I = 1$ si l'intervention I est sous–traitée. On propose de formuler la relaxation $\tilde{P}$ du problème P de la manière suivante :

$$
\begin{aligned}
(\tilde{P}) \text{ Minimiser} \quad & \rho \\
\text{sujet à} \quad & \sum \{x_{I,t} \mid C(t,i) \geq n\} \geq (1 - y_I) R(I,i,n), \quad \forall I, \forall i, \forall n, && (5.15) \\
& \sum_I T(I) x_{I,t} \leq \rho \cdot H_{max}, \quad \forall t, && (5.16) \\
& x_{I,t} + y_I \leq 1, \quad \forall I, \forall t, && (5.17) \\
& y_I \leq y_J, \quad \forall I \in Pred(J), && (5.18) \\
& \sum_I cost(I) y_I \leq A, && (5.19) \\
& x_{I,t} \geq 0, \quad \forall I, \forall t, && (5.20) \\
& y_I \geq 0, \quad \forall I. && (5.21)
\end{aligned}
$$

Les contraintes (5.15) vérifient le respect des compétences requises par l'intervention ; les contraintes (5.16) assurent que chaque technicien ne travaille pas plus que le temps global de l'ordonnancement ; les contraintes (5.17) imposent qu'une intervention est : soit sous–traitée, soit traitée par au moins un technicien ; les contraintes (5.18) maintiennent les relations de précédence entre les interventions sous–traitées et la contrainte (5.19) veille au respect de la contrainte de budget pour sous–traiter les interventions.

Le problème $\tilde{P}$ est une relaxation du problème P original car les journées et les équipes de techniciens ne sont plus prises en compte. La résolution de $\tilde{P}$ nous donne un ordonnancement qui minimise le nombre de jours en prenant simplement en compte l'affectation des techniciens aux interventions tout en veillant au respect des contraintes de précédence et au respect de la contrainte de budget. Nous proposons d'utiliser $\tilde{P}$ pour l'initialisation du critère d'insertion $\psi(I)$. Pour cela, nous considérons de manière empirique qu'il est préférable de sous–traiter une intervention I de valeur fractionnaire $y_I > 0$ à l'optimum de $\tilde{P}$ plutôt qu'une intervention à valeur $y_I = 0$. Notre expérimentation a consisté à résoudre $\tilde{P}$ et à initialiser les poids d'insertion $\psi(I)$ de la manière suivante :

$$
\psi(I) = \begin{cases} \mathbf{M} + \omega(I)/(|Succ(I)| + 1) & \text{si } y_I > 0 \\ \omega(I)/(|Succ(I)| + 1) & \text{sinon} \end{cases}
$$

avec $\mathbf{M}$ la valeur maximale des ratios $\omega(I)/(|Succ(I)|+1)$ parmi toutes les interventions I. Ainsi, les interventions de valeur $y_I > 0$ sont insérées en priorité et seront classées entre elles selon l'ordre décroissant des valeurs $\psi(I)$ et les autres interventions sont classées en dernier suivant l'ordre décroissant des valeurs $\psi(I)$. Les résultats obtenus en exécutant cette version de l'algorithme pendant 600 secondes sont exposés dans le tableau 5.6. Comme dans les tableaux de résultats précédemment exposées, les valeurs `BS` correspondent à la meilleure valeur de la fonction objective trouvée en exécutant l'algorithme GRASP pendant 1200 secondes.

TAB. 5.6 – Résultats obtenus par Resolution search avec prise en compte de $\tilde{P}$

Pb	Meilleures valeurs		Resolution search+$\tilde{P}$			Pb	Meilleures valeurs		Resolution search+$\tilde{P}$		
	BI	BS	BI	t(s)	BS		BI	BS	BI	t(s)	BS
B1	**36525**	**42180**	38565	120	44430	X1	**135675**	**172920**	**135675**	93	**172920**
B2	**15225**	**20340**	19915	0	**20340**	X2	6480	7260	**6420**	0	**6840**
B3	**16800**	**19065**	17715	0	27015	X3	45360	50280	**44250**	0	**50160**
B4	**18915**	**26025**	**18915**	1	26265	X4	59380	**71560**	**58740**	416	71800
B5	104610	117630	**93360**	0	**104760**	X5	124875	165960	**124875**	282	**165960**
B6	**24390**	**33390**	24870	0	34920	X6	**6180**	**9390**	6825	46	9465
B7	**27930**	**35280**	28470	311	36600	X7	**43320**	**43680**	44340	360	45240
B8	**32820**	33030	**32820**	0	**33000**	X8	**18830**	**25680**	19210	100	27300
B9	**25695**	**27960**	**25695**	2	**27960**	X9	100920	**159600**	**100545**	23	161640
B10	**32790**	**34920**	**32790**	0	35040	X10	98010	**160920**	**92355**	190	172800

Dans l'ensemble, les résultats ne sont pas meilleurs que ceux donnés par la première implémentation de Resolution search, cependant la prise en compte de la solution optimale de $\tilde{P}$ nous donne certaines améliorations importantes. Tout d'abord, on constate une nette diminution de la borne inférieure trouvée pour l'instance B5 qui passe de 104610 à 93360. De plus, on obtient des améliorations des meilleures solutions connues pour les instances B8 et X2. On remarque encore une fois que la corrélation entre la valeur de la borne inférieure et celle de la borne supérieure n'est pas toujours vérifiée. On observe, par exemple, une amélioration de la borne supérieure pour l'instance B8 alors que la valeur de la borne inférieure est la même que pour la version précédente de Resolution Search (tableau 5.5).

Preuves d'optimalité

Nous avons mené une dernière expérimentation consistant à exécuter l'algorithme de recherche des ensembles maximaux par Resolution search durant 10 heures. Les objectifs étaient de savoir jusqu'à quel niveau cette préprocédure pouvait améliorer les résultats obtenus par la méthode GRASP et s'il était possible d'énumérer tous les ensembles maximaux pour d'autres instances que B9. Nous avons exécuté les deux versions étudiées précédemment : la version classique et la version avec prise en compte de la solution optimale de $\tilde{P}$. Les résultats sont exposés dans le tableau 5.7. La colonne *Meilleures valeurs* correspond aux meilleures valeurs obtenues jusqu'à présent, la colonne *Resolution search* correspond aux valeurs obtenus par Resolution search et la colonne *Resolution search* + $\tilde{P}$ aux valeurs obtenues par Resolution search avec prise en compte de la solution optimale de $\tilde{P}$ (les deux variantes ayant été exécutées pendant 10 heures). La colonne *challenge* correspond aux meilleures valeurs obtenues parmi tous les participants au challenge. Les colonnes `BI` et `BS` correspondent respectivement aux bornes inférieures et supérieures obtenues et la colonne *t(s)* est le temps d'exécution en secondes nécessaire à l'obtention de la borne inférieure.

Les valeurs en gras indiquent une amélioration de la borne supérieure par rapport aux meilleures valeurs obtenues jusqu'à présent et les valeurs encadrées correspondent aux améliorations par rapport aux meilleurs résultats obtenus parmi les participants au challenge. Les valeurs de bornes inférieures ayant une étoile correspondent aux valeurs optimales.

Tab. 5.7 – Résultats obtenus par Resolution search après 10h de temps de calculs

	Meilleures valeurs		Resolution search			Resolution search + $\tilde{P}$			challenge
	BI	BS	BI	t (s)	BS	BI	t (s)	BS	BS
B1	36525	42180	36240	1066	**41730**	37455	5441	42630	34395
B2	15225	20340	15015	605	18915	15135	21458	**18705**	15870
B3	16800	**19065**	16770	5403	20565	17715	0	21375	16020
B4	18915	26025	18495	4847	26325	18195	4631	**25965**	25305
B5	93360	**104760**	104310	13654	115260	93360	0	**104760**	89700
B6	24390	**33390**	24390	0	**33390**	24855	2304	34545	27615
B7	27930	**35280**	27930	286	35760	28050	10899	35940	33300
B8	32820	**33000**	32820	0	33030	32820	0	**33000**	33030
B9	25695	**27960**	25695*	4	**27960**	25695*	2	**27690**	28200
B10	32790	**34920**	32790	0	35040	32790	0	35040	34680
X1	135675	**172920**	135675*	510	**172920**	135675*	93	**172920**	151140
X2	6480	**6840**	6420	3435	7050	6270	864	6990	7260
X3	45360	50280	43830	1404	**50160**	43980	1404	**50160**	50040
X4	59380	**71560**	59060	12764	71640	58740	416	71800	65400
X5	124875	**165960**	124875*	282	**165960**	124875*	282	**165960**	147000
X6	6180	**9390**	6180	36	**9390**	6825	46	9465	9480
X7	43320	43680	42990	26364	43560	41310	31091	**41880**	33240
X8	18830	**25680**	18190*	20571	25740	18190*	20205	25800	23640
X9	100920	159600	99015*	1279	**153660**	99015*	1633	167880	134760
X10	98010	**160920**	96750	27011	175440	91500	33110	168060	137040

On constate une nette amélioration des valeurs de borne inférieure (11 valeurs sur les 20 instances ont été améliorées). Cependant les bornes supérieures ne suivent pas toujours cette tendance car seules les instances B1, B2, B4, X3, X7 et X9 sont strictement améliorées. On constate de nouveau que l'évaluation des sous–ensembles maximaux par le calcul de la borne inférieure n'est pas toujours fiable : pour les instances B3, X2, X4, X8 et X10 une borne inférieure plus petite donne une borne supérieure plus grande.

5.4 Conclusion

Dans ce chapitre, nous avons présenté un problème de planification de techniciens et d'interventions pour les télécommunications. Nous avons proposé une heuristique de résolution constituée de trois phases : (i) 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, (ii) 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 (iii) la procédure GRASP qui cherche à améliorer les solutions initiales tout en mettant à jour les poids attribués aux interventions.

Les expérimentations ont montré que le choix de l'ensemble d'interventions à sous–traiter peut directement influencer la qualité des résultats produits par notre approche. Nous avons donc proposé une étude portant sur différentes manières d'identifier ce sous–ensemble et avons en particulier montré que Resolution search offre d'intéressantes possibilités d'exploration. Contrairement à un algorithme d'énumération classique (de type backtracking), Resolution search permet d'énumérer les ensembles maximaux suivant deux critères : un critère d'insertion et un critère de remise en cause. Nous avons proposé différentes manières d'implémenter ces critères en fonction du nombre minimum de techniciens requis et du nombre de prédécesseurs ou de successeurs des interventions. Les résultats expérimentaux montrent clairement l'intérêt de la prise en compte de ces deux critères pour la résolution de ce sous–problème. De manière plus générale, cela montre également les possibilités intéressantes qu'offre Resolution search pour la résolution de problèmes d'optimisation combinatoire. Une perspective intéressante serait d'étudier de manière plus approfondie l'affectation des critères d'insertion et de remise en cause pour une énumération plus pertinente de l'espace des solutions. Il serait également intéressant de trouver d'autres manières d'évaluer la qualité des sous–ensembles maximaux car l'évaluation de ces sous–ensembles par le calcul de la borne inférieure a montré, dans certains cas, qu'il était sous–optimal.