UNE EXTENSION DU RCPSP POUR LA MUTUALISATION DE RESSOURCES ENTRE PLUSIEURS SITES : LE MULTI LOCATION RCPSP

Arnaud Laurent, Laurent Deroussi, Nathalie Grangeon, Sylvie Norre

10ème Conférence Internationale de MOdélisation, Optimisation et SIMulation - MOSIM'14 - 5 au 7 Novembre 2014 - Nancy -France « de l'Économie linéaire à l'économie circulaire »

Une extension du RCPSP pour la mutualisation de ressources entre plusieurs sites : le Multi Location RCPSP

Arnaud LAURENT, Laurent DEROUSSI, Nathalie GRANGEON, Sylvie NORRE

LIMOS UMR CNRS 6158
Complexe scientifique des Cézeaux
63173 AUBIERE Cedex - France
arnaud.laurent@isima.fr, deroussi@moniut.univ-bpclermont.fr
nathalie.grangeon@moniut.univ-bpclermont.fr, sylvie.norre@moniut.univ-bpclermont.fr

RÉSUMÉ : *Cet article propose une extension du Resource Constrained Project Scheduling Problem (RCPSP) dans un environnement multi-sites avec mutualisation de ressources. Cette extension intègre la prise en compte de nouvelles contraintes telles que les temps de déplacement des ressources et le choix des sites d'affectation pour chaque tâche. Une modélisation mathématique du problème sous forme linéaire est donnée. Puis une approche de résolution à l'aide d'une méta-heuristique basée individu est présentée. Des résultats avec IBM ILOG CPLEX sont présentés sur des instances générées du problème et mis en parallèle avec les résultats obtenus avec la méta-heuristique. Les résultats obtenus sont encourageants*

MOTS-CLÉS : *RCPSP, Multi-Sites, Temps de déplacement, Modélisation Mathématique, Méta-Heuristique.*

1 INTRODUCTION

Depuis plusieurs années, une nouvelle vision de la gestion des hôpitaux publics a vu le jour. Cette nouvelle vision repose sur la mutualisation des ressources des hôpitaux publics afin d'optimiser leur utilisation. Une loi française de 2009 (loi « Hôpital, patients, santé et territoires ») rend cette mutualisation possible.

Le terme employé est « Communauté Hospitalière de Territoire » (C.H.T.). Elle désigne un regroupement d'hôpitaux publics qui peuvent mettre en commun leurs ressources humaines (médecins, infirmiers, manipulateurs,...), leurs ressources matérielles (blocs opératoires, radios, scanners,...), ainsi que leurs patients et leurs opérations et examens. Le but de cette structure est d'améliorer l'offre de soin territoriale et de créer une meilleure complémentarité entre les acteurs.

Ce problème a déjà été spécifié et traité par (Gourgand et al. 2014a) et (Gourgand et al. 2014b), dans le cadre de l'imagerie médicale. Leur problème a pour but d'assigner des tâches à des périodes de travail sous certaines contraintes. Cependant leur modèle ne prend pas en compte les temps de déplacement des ressources, ni le fait que les patients puissent avoir plusieurs examens à passer. De plus les ressources humaines sont considérées uniquement dans (Gourgand et al. 2014b). Notre objectif est de prendre en compte toutes les contraintes liées à l'aspect multi-sites. Nous présentons dans ce papier la spécification du problème de C.H.T. ainsi que les différents problèmes de la littérature qui s'en approchent. Nous proposons un modèle mathématique pour la résolution exacte, ainsi qu'une méthode approchée pour résoudre ce type de problème. Nous étudions les résultats avec les deux méthodes et concluons sur leur efficacité.

2 LE PROBLÈME DES COMMUNAUTÉS HOSPITALIÈRES DE TERRITOIRE

Un des problèmes inhérents des C.H.T. consiste à ordonnancer les opérations et examens des patients, à leur affecter les ressources humaines et matérielles nécessaires et définir dans quel hôpital ils vont être pris en charge, en optimisant certains critères.

Pour illustrer notre problème, nous nous concentrerons sur un seul aspect du problème de la C.H.T., celui de l'imagerie médicale.

Les patients sont les entités qui doivent passer les examens. Ils sont donc obligatoirement présents à l'hôpital pendant toute la durée de leurs examens et sont soumis à des contraintes dans le temps et l'espace. En effet un patient ne peut pas passer deux examens en même temps, et met un certain temps pour aller d'un hôpital à un autre si deux de ses examens n'ont pas lieu dans le même hôpital. Un patient est caractérisé par une liste d'examens, dans laquelle il existe un ordre total pour leur exécution tel que

pour tous les couples d'examens a et b il existe une contrainte de précédence.

Chaque examen est défini par une durée et des quantités de types de ressources matérielles et humaines nécessaires à son exécution.

Les ressources humaines sont soumises aux mêmes contraintes dans le temps et dans l'espace que le patient. Une ressource humaine est caractérisée par son type. Les ressources humaines peuvent se déplacer d'hôpital en hôpital pour effectuer différentes tâches.

Les ressources matérielles sont des ressources qui ne peuvent pas quitter leur hôpital de référence. C'est la différence majeure avec les ressources humaines du point de vue du problème. Une ressource matérielle est caractérisée par son type et son hôpital de référence.

Les hôpitaux représentent les sites où vont être effectués les examens. Tous les hôpitaux sont éloignés d'un temps de trajet connu. Cette durée va être le temps à considérer pour un déplacement de ressource ou de patient.

Un exemple d'une répartition des ressources possible de ce problème est représenté par la figure 1.

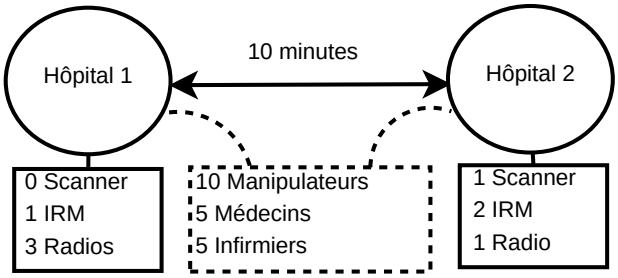

Figure 1: Exemple de répartition des ressources par hôpitaux

L'objectif de ce problème est d'affecter un site et les ressources que les examens nécessitent pour être exécutés et de les ordonnancer sur un horizon temporel. Pour évaluer les solutions d'ordonnancement et d'affectation des examens, on utilisera la date de fin du dernier examen (Makespan).

3 PROBLÈMES SIMILAIRES DANS LA LITTÉRATURE

Des problèmes ont été identifiés dans la littérature pour répondre aux problématiques d'affectation de ressources et d'ordonnancement. Un problème se rapproche plus particulièrement du notre, le "Resource Constrained Project Scheduling Problem" (RCPSP). Nous présentons ce problème ainsi que plusieurs de ses extensions.

3.1 Le problème de gestion de projet à contraintes de ressources

Le problème de gestion de projet à contraintes de ressources est aussi appelé RCPSP pour Resource Constrained Project Scheduling Problem et est noté $PS|prec|C_{max}$ par (Kan A. R., 1976). Ce problème consiste à réaliser N tâches qui nécessitent chacune une ou plusieurs ressources. Il existe K types de ressources différentes. Parmi ces N tâches, il y a deux tâches fictives de durée nulle $j = 1$ et $j = N$, tel que $j = 1$ précède toutes les tâches et $j = N$ succède à toutes les tâches. Les ressources sont dites renouvelables, ce qui signifie que lorsque la tâche à laquelle elles sont affectées est terminée, les ressources sont de nouveau disponibles. On dispose de T périodes et $R_{k,t}$ ressources de type k à la période t pour exécuter toutes les tâches. Les tâches sont non-préemptives. Une fois commencée, chaque tâche ne peut pas être stoppée pendant toute la durée de son exécution p_j (en nombre de périodes). Chaque tâche $j = 1, N$ possède une liste P_j de tâches qui doivent la précéder. Chaque tâche $j = 1, N$ nécessite une affectation de $r_{j,k}$ ressources de type $k = 1, K$.

3.1.1 Le modèle mathématique

Une formalisation mathématique linéaire de ce problème a été proposée par (Oğuz O. et Bala H., 1994). On utilise la variable $X_{j,t}$ pour désigner la date de fin de la tâche j, tel que $X_{j,t} = 1$ si la tâche j se termine pendant la période t, sinon $X_{j,t} = 0$.

$$Minimiser \quad \sum_{t=1}^{T} t * X_{N,t}; \tag{1}$$

$$\sum_{t=1}^{T} X_{j,t} = 1; \quad \forall j = 1, N; \tag{2}$$

$$\sum_{t=1}^{T} t * X_{h,t} + p_j \leq \sum_{t=1}^{T} t * X_{j,t}; \forall j = 1, N; \forall h \in P_j; \tag{3}$$

$$\sum_{j=1}^{N} (r_{j,k} * \sum_{q=t}^{t+p_j-1} (X_{j,q})) \leq R_{k,t}; \forall t = 1, T; \forall k = 1, K; \tag{4}$$

$$X_{j,t} \in \{0, 1\}; \quad \forall j = 1, N; \forall t = 1, T; \tag{5}$$

Le but est de minimiser la durée du projet, appelée makespan (1). Pour minimiser cette fonction on doit respecter les contraintes de non préemption et de réalisation (2); les contraintes de précédence (3); les contraintes de respect des quantités de ressources disponibles (4); les contraintes qui font que les variables sont binaires (5).

3.2 Le problème de gestion de projet à contraintes de ressources avec time lags

Le principe des time lags est de rajouter des délais maximaux et/ou minimaux entre la fin d'une tâche et le début d'une autre.

3.2.1 Time lags minimaux

Les time lags minimaux représentent un délai minimum à respecter entre deux tâches. Ces time lags minimaux peuvent servir à modéliser un temps de transfert d'une tâche à l'autre ou bien un temps d'attente obligatoire. Ils s'appliquent dans le cas où une tâche i doit succéder à une autre tâche j ($j \Leftarrow i$) avec un time lag minimal $lagmin(j,i)$ en respectant un délai minimum de façon à respecter l'équation (6) :

$$\sum_{t=1}^{T} t * X_{i,t} \geq \sum_{t=1}^{T} t * X_{j,t} + p_i + lagmin(j,i) \quad (6)$$

3.2.2 Time lags maximaux

Les time lags maximaux représentent un délai maximum à respecter entre deux tâches. Ces time lags maximaux peuvent servir à modéliser une durée maximum à ne pas dépasser entre deux étapes d'un travail sur un objet. On modélise ce time lag pour une tâche i qui dit succéder à la tâche j ($j \Leftarrow i$) avec un time lag maximal $lagmax(j,i)$ par l'équation (7) :

$$\sum_{t=1}^{T} t * X_{i,t} \leq \sum_{t=1}^{T} t * X_{j,t} + p_i + lagmax(j,i) \quad (7)$$

3.3 Le problème de gestion de projet à contraintes de ressources avec time lags conditionnels

Une extension du problème de RCPSP avec time lag minimum a été proposée par (Toussaint H., 2004). Le but est de prendre en compte des événements qui pourraient retarder l'exécution d'une tâche. Ces retards peuvent provenir de plusieurs contraintes physiques, telles que des temps de transfert d'un objet d'une tâche vers la prochaine tâche qui le concerne, représentés par des contraintes de précédence ou des temps d'acheminement de ressources sur le lieu d'exécution de la tâche. On peut considérer que les ressources sont toutes stockées dans un dépôt au début du projet, et doivent toutes y retourner à la fin. Contrairement au cas précédent un time lag minimum est donné entre chaque couple de tâche. Le principe repose sur le fait de le prendre en compte que si au moins une des conditions suivantes est vérifiée :

- Les deux tâches se transmettent au moins une ressource
- Il existe une contrainte de précédence entre les deux tâches

L'équation (6) s'applique donc lorsqu'un des deux cas est avéré. Ces deux cas représentent ainsi les temps de transfert des ressources d'un lieu à l'autre ou bien du transfert d'un objet d'une tâche à la prochaine tâche. Ces time lags sont donnés pour un couple de tâches, ce qui nécessite par avance de connaître le lieu où les tâches vont être exécutées dans le cas où ils représentent des temps de transfert.

3.4 Conclusion sur les problèmes de la littérature et proposition d'extension

A notre connaissance aucun modèle déjà traité ou proposé ne permet de résoudre notre problème dans son intégralité. Le RCPSP avec time lags conditionnels est le seul à prendre en compte des temps de déplacement. Cependant ces temps doivent être connus pour chaque couple de tâches. Dans notre problème, cela nécessiterait de connaître le lieu d'exécution des examens, or c'est à nous de le déterminer. Nous avons donc construit à partir des extensions du RCPSP, une nouvelle extension prenant en compte l'aspect multi-sites du problème. Nous l'avons baptisé MLRCPSP pour "Multi Location RCPSP".

Ce problème consiste à ajouter la détermination du site sur lequel chaque tâche est réalisée, au problème de RCPSP avec time lags conditionnels présenté dans la section 3.3. Les time lags ne sont plus donnés pour deux tâches puisqu'ils dépendent des sites $s = 1, S$ où vont être exécutées les tâches et ces sites ne sont pas connus à l'avance. On ajoute donc la notion de site $s = 1, S$, mais aussi la notion de ressource fixe $(k = 1, K; r = 1, R_k;) \wedge (M_{k,r} = 0)$ et mobile $(k = 1, K; r = 1, R_k;) \wedge (M_{k,r} = 1)$. Les ressources mobiles peuvent être affectées aux tâches affectées sur n'importe quel site. Les ressources fixes sont elles restreintes à l'affectation des tâches assignées au même site qu'elles. On appelle son site de référence $loc_{k,r}$ le site où la ressource r de type k est fixée.

Le problème du RCPSP avec time lags conditionnels consiste à trouver le meilleur ordonnancement des tâches et la meilleure affectation des ressources mobiles pour effectuer l'ensemble des tâches. Le but du MLRCPSP est donc d'ajouter l'affectation des sites et des ressources fixes nécessaires à l'exécution des tâches. L'utilisation de ces ressources est limitée aux tâches affectées au site, qui est le site de référence de la ressource r de type k. Chaque site $s = 1, S$ est éloigné d'un trajet en temps $\delta(s, s')$ par rapport à un autre site $s' = 1, S$. On déduit la valeur du temps de déplacement entre deux tâches $i, j = 1, N$ par la distance entre le site qui leur sont assignés. La distance entre un site et lui même est nulle, on considère donc que le temps de déplacement dans ce cas est nul.

4 Modélisation mathématique

4.1 Notations

Les notations présentées sont celles fréquemment utilisées pour la modélisation de problèmes de type RCPSP et ses extensions. Nous présentons aussi les nouvelles notations que notre modèle mathématique utilise.

- **Données**

N Nombre de tâches avec 1 et N les deux tâches fictives de début et fin de projet

p_j Durée de la tâche $j = 1, N$

P_j Ensemble de tâches qui doivent précéder $j = 1, N$

K Nombre de types de ressources

R_k Nombre de ressources de type $k = 1, K$

$r_{j,k}$ Nombre de ressources de type $k = 1, K$ nécessaires pour la tâche $j = 1, N$

T Nombre maximum de périodes

$M_{k,r}$ $= 1$ si la ressource $r = 1, R_k$ de type $k = 1, K$ est mobile, 0 si elle est fixe

S Nombre de sites

$\delta_{s,s'}$ Distance en périodes entre le site $s = 1, S$ et le site $s' = 1, S$

$loc_{k,r}$ Site d'appartenance de la ressource $r = 1, R_k$ de type $k = 1, K$

H Un grand nombre

- **Variables**

$X_{j,t}$ $=1$ si la tâche $j = 1, N$ se termine à la période $t = 1, T$, 0 sinon

$Y_{j,k,r}$ $= 1$ si la ressource $r = 1, R_k$ de type $k = 1, K$ est affectée à la tâche $j = 1, N$, 0 sinon

$Z_{j,s}$ $= 1$ si la tâche $j = 1, N$ se déroule sur le site $s = 1, S$, 0 sinon

$\omega_{j,h}$ $= 1$ si un temps de déplacement doit être pris en compte entre la fin de la tâche $j = 1, N$ et le début de la tâche $h = 1, N$, 0 sinon

4.2 Modèle

$$Minimiser \quad \sum_{t=1}^{T} t * X_{N,t} \tag{8}$$

$$\sum_{t=1}^{T} X_{j,t} = 1; \quad j = 1, N; \tag{9}$$

$$\omega_{h,j} = 1; \quad j = 1, N; h \in P_j; \tag{10}$$

$$Y_{j,k,r} + Y_{h,k,r} \leq \omega_{j,h} + \omega_{h,j} + 1; \quad (j, h = 1, N) \wedge (j < h); k = 1, K; r = 1, R_k; \tag{11}$$

$$\sum_{t=1}^{T} t * X_{j,t} \geq \sum_{t=1}^{T} t * X_{h,t} + p_j + (Z_{j,s} + Z_{h,s'} - 1) * \delta(s, s') - H * (1 - \omega_{h,j}); \quad j, h = 2, N-1; s, s' = 1, S; \tag{12}$$

$$\sum_{t=1}^{T} t * X_{j,t} \geq \sum_{t=1}^{T} t * X_{h,t} + p_j; \quad ((j = N) \wedge (h = 2, N-1)) \cup ((h = 1) \wedge (j = 2, N-1)); \tag{13}$$

$$\sum_{r=1}^{R_k} Y_{j,k,r} = r_{j,k}; \quad j = 1, N; k = 1, K; \tag{14}$$

$$Y_{j,k,r} \leq Z_{j,loc_{k,r}}; \quad j = 1, N; (k = 1, K; r = 1, R_k;) \wedge (M_{k,r} = 0); \tag{15}$$

$$\sum_{s=1}^{S} Z_{j,s} = 1; \quad j = 1, N; \tag{16}$$

$$X_{j,t} \in \{0; 1\}; \quad j = 1, N; t = 1, T; \tag{17}$$

$$Y_{j,k,r} \in \{0; 1\}; \quad j = 1, N; k = 1, K; r = 1, R_k; \tag{18}$$

$$Z_{j,s} \in \{0; 1\}; \quad j = 1, N; s = 1, S; \tag{19}$$

$$\omega_{j,h} \in \{0; 1\}; \quad j, h = 1, N; \tag{20}$$

Nous proposons une formulation, basée sur les modèles de la littérature, notamment (Oğuz O. and Bala H., 1994) et (Correia I. and al., 2012). Le but est de minimiser le makespan (8). Les contraintes exprimment : la non préemption et la réalisation des tâches (9); les contraintes de précédence entraînent des déplacements (10); le partage des ressources entraînent des déplacements (11); le calcul des dates de fin de tâche prenant en compte les temps de déplacement (12); le respect des dates de fin de tâche par rapport aux tâches fictives (13); le respect des quantités nécessaires de ressources affectées (14); l'obligation d'exécuter la tâche sur un site compatible avec les ressources fixes affectées (15); l'exécution des tâches sur un seul site (16); les contraintes qui font que les variables sont binaires (17)(18)(19)(20).

Notre modèle est linéaire et ses $(N*(T+(\sum_{k=1}^{K} R_k)+ S + N))$ variables sont toutes binaires. Pour donner un ordre de grandeur, une instance avec 10 tâches à planifier sur 50 périodes, avec 10 ressources toutes fixes, 2 sites, sans contraintes de précédence entre les tâches, est représentée par un modèle à 720 variables et 877 contraintes.

5 MÉTA-HEURISTIQUE

Dans la section précédente, nous avons proposé un modèle mathématique pour résoudre notre problème. Cependant une résolution exacte n'est pas adaptée pour des instances de taille moyenne ou grande. C'est pour cette raison que nous proposons dans cette section une méthode approchée pour la résolution du MLRCPSP.

5.1 La recherche locale

Le principe de la recherche locale est d'explorer le voisinage d'une solution et d'accepter une solution voisine si elle est de meilleure ou d'égale qualité. On réitère un certain nombre de fois l'opération pour obtenir un minimum local. Dans notre algorithme nous utilisons une recherche locale stochastique, qui choisit à chaque itération un seul voisin aléatoirement. L'algorithme de principe de notre recherche locale stochastique est l'algorithme 1.

5.2 La recherche locale itérée

Le principe de la recherche locale itérée (Lourenço H. R. et al., 2001) est d'effectuer une suite de recherches locales, en effectuant une perturbation de la solution entre chacune d'entre elles. L'algorithme de principe de la recherche locale itérée, que nous allons utiliser pour résoudre notre problème, est celui de l'algorithme 2.

La perturbation est dans notre cas, le choix aléatoire

Algorithme 1 : Algorithme de principe de notre recherche locale stochastique

Entrées : X_0 : Solution initiale;
Variables : X^* : Meilleure solution trouvée;
X' : Solution courante;
Initialisation : $X' \leftarrow X_0$;
$X^* \leftarrow$X';
1 **Début**
2 **Tant que** *Test d'arrêt est faux* **faire**
3 $X' \leftarrow$ choisir aléatoirement et uniformément d'un voisin dans le système de voisinage de X^*;
Si *H(X')≤ H(X*)* **alors**
4 $X^* \leftarrow$X';
5 **Finsi**
6 **Fintq**
7 **Retourner** X^*
8 **Fin**

Algorithme 2 : Algorithme de principe de la recherche locale itérée

Entrées : X_0 : Solution initiale;
Variables : X^* : Meilleure solution trouvée;
X'' : Solution Voisine;
X' : Solution courante;
Initialisation : $X' \leftarrow X_0$;$X^* \leftarrow X'$
1 **Début**
2 **Tant que** *Test d'arrêt est faux* **faire**
3 $X'' \leftarrow$ Recherche locale sur X";
Si *H(X")≤ H(X*)* **alors**
4 $X^* \leftarrow$X";
5 **Finsi**
6 $X' \leftarrow$ Choix ou non de X" par rapport à X' (critère d'acceptation);
$X'' \leftarrow$ Perturbation de X';
7 **Fintq**
8 **Retourner** X^*
9 **Fin**

d'un voisin dans un système de voisinage. Il nous reste à définir le codage d'une solution, le critère d'acceptation d'une solution, les systèmes de voisinages pour la recherche locale et la perturbation, ainsi que les tests d'arrêt pour la recherche locale et pour la recherche locale itérée.

5.3 Codage d'une solution

Dans un premier temps, nous allons définir le codage d'une solution. Nous proposons de représenter une solution x par la concaténation de deux vecteurs :

$$x = (\sigma, l) \tag{21}$$

Le vecteur σ correspond à une séquence des tâches et le vecteur l correspond à l'affectation des tâches aux sites, tels que :

- $\sigma = \sigma_1, \sigma_2, ..., \sigma_N$, est une permutation des N tâches qui désigne une liste topologique qui respecte les contraintes de précédence. Cette séquence devra être respectée par chaque ressource dans l'ordre d'exécution des tâches qui lui sont assignées. C'est à dire que si une tâche j précède une tâche i dans σ, alors aucune ressource ne pourra exécuter i puis j.
- $l = l_1, l_2, ...l_N$ avec $l_j \in \{1, S\}$ le site où la tâche $j \in \{1, N\}$ va être exécutée. Cette affectation respecte le fait que le nombre de ressources disponibles pour sur ce site, est suffisant pour exécuter la tâche.

Cette structure seule ne représente pas une solution, mais potentiellement plusieurs solutions qui respectent la séquence σ et l'affectation l. Une solution est aussi définie par une date de début pour toutes les tâches ainsi qu'une affectation des ressources aux tâches. Pour déterminer notre solution, nous appliquerons à ce codage un algorithme d'ordre strict, en nous inspirant des travaux de (Carlier J. 1984).

5.4 Algorithme d'ordre strict

5.4.1 Principe

L'algorithme d'ordre strict a pour principe *d'ordonnancer les tâches le plus tôt possible en respectant les contraintes de précédence et les disponibilités des ressources, dans l'ordre donné par une liste topologique* σ (Carlier J. 1984).

Dans notre cas, nous disposons d'une liste topologique σ et d'un vecteur de site l. Le vecteur l va être utilisé pour déterminer les disponibilités des ressources à l'étape on on cherche à fixer la date de fin d_j de la tâche $j \in \{1, N\}$. On détermine l'affectation des ressources et la date de fin de la tâche dans l'ordre σ.

Il y aura alors trois cas pour une ressource $r \in \{1, R_k\}$ de type $k \in \{1, K\}$ pour calculer sa date de disponibilité $dis_{k,r}$ pour réaliser la tâche j :

- Soit cette ressource est mobile, et dans ce cas sa date de disponibilité au plus tôt pour commencer la tâche est égale à :
  - $dis_{k,r} = 0$ si elle n'a encore aucune tâche assignée
  - $dis_{k,r} = d_h + \delta(l_h, l_j)$ si sa dernière tâche assignée est h
- Soit cette ressource est fixe telle que : $l_j \neq loc_{k,r}$. Dans ce cas la ressource ne peut pas être affectée à j ($dis_{k,r} = +\infty$)
- Soit cette ressource est fixe telle que : $l_j = loc_{k,r}$. Dans ce cas sa date de disponibilité pour la tâche $j \in \{1, N\}$ est égale à $dis_{k,r} = d_h$ avec h la dernière tâche assignée à la ressource.

On affecte la tâche j aux $r_{j,k}$ ressources ($Y_{j,k,r} = 1$) de type k qui ont les dates de disponibilité $dis_{k,r}$ les plus petites. La date de fin d'exécution d_j de la tâche j sera égale à l'équation (22).

$$d_j = \max(A, B) + p_j \tag{22}$$
$$A = \max(d_h + \delta(l_h, l_j); \forall h \in P_j) \tag{23}$$
$$B = \max(dis_{k,r}; \forall k, r | Y_{j,k,r} = 1) \tag{24}$$

A représente la plus grande date de fin des tâches qui doivent précéder j. B représente la plus grande date de disponibilité des ressources que l'on a assignée à j. L'algorithme de principe de cette version de l'algorithme d'ordre strict, est décrit dans l'algorithme 3. De plus cet algorithme retourne la valeur de la fonction objective, le makespan.

Algorithme 3 : Algorithme d'ordre strict pour le MLRCPSP (H(X))

Entrées : σ : Liste Topologique; l : Affectation des sites
Variables : d : Vecteur des dates de fin des tâches ($d = d_1, d_2, ...d_N$);
dis : Vecteur des dates de disponibilité des ressources ($dis = dis_{1,1}, ..., dis_{R_k,1}, dis_{1,2}...dis_{K,R_K}$);
Initialisation : $dis \leftarrow \{0, ..., 0\}$;$d \leftarrow \{0, ..., 0\}$;

```
Début
    Pour j = σ1σN faire
        Calcul des valeurs du vecteur dis pour j en prenant en compte l et d;
        Calcul de dj;
    Finpour
    Retourner dN
Fin
```

5.5 Systèmes de voisinage

Nous avons besoin d'au moins deux systèmes de voisinage pour exécuter notre recherche locale itérée, un pour la recherche locale et un autre pour la perturbation. Nous ferons en sorte que les deux systèmes de voisinage permettent d'atteindre l'ensemble des solutions possibles avec notre codage des solutions. De ce fait nous proposons deux voisinages, l'un modifiant σ, l'autre modifiant l.

5.5.1 Le voisinage V1

Le premier est un voisinage de type insertion sur σ. Le principe est de déplacer une tâche σ_p de la position $p = 2, N-1$ dans le vecteur σ à la position $p' = 2, N-1$, $p' \neq p$. On n'applique pas le déplacement de la tâche si la nouvelle position viole une contrainte de précédence. Les contraintes de précédence sont violées si :

- $p < p'$ et il existe au moins une tâche à la position p'' entre p et p' telle que : $\sigma_p \Leftarrow \sigma_{p''}$
- $p > p'$ et il existe au moins une tâche à la position p'' entre p et p' telle que : $\sigma_{p''} \Leftarrow \sigma_p$

Ainsi le voisin de $\sigma = \sigma_1, ...\sigma_{p-1}, \sigma_p, \sigma_{p+1}, ..., \sigma_N$ pour le déplacement de la tâche σ_p à la position p' est :

- Si le déplacement viole les contraintes de précédence, alors on choisit un autre p et p'
- Sinon:
  - Si $p < p'$ alors $\sigma' = \sigma_1, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma'_p, \sigma_p, ..., \sigma_N$
  - Sinon : $\sigma' = \sigma_1, ..., \sigma'_p, \sigma_p, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma_N$

Trouver un voisin qui appartient au voisinage V1 correspond à l'algorithme 4.

Algorithme 4 : Algorithme de principe de la recherche d'un voisin dans V1

Entrées : X : Solution initiale;
Variables :
X' : Solution voisine;
Initialisation : $X' \leftarrow X$;
Début
  Tant que $X' = X$ **faire**
    Déplacement d'une tâche dans σ de X';
    Si X' *n'est pas réalisable* **alors**
      $X' \leftarrow X$;
    Finsi
  Fintq
  Retourner X'
Fin

5.5.2 Le voisinage V2

Le deuxième voisinage proposé est une modification du site s en site s' assigné à une tâche $j = 2, N-1$. On applique la modification seulement si la solution reste réalisable. Elle reste réalisable si pour tous les types de ressources nécessaires à j, le nombre de ressources mobiles de ce type, plus le nombre de ressources fixes de ce type sur le site s' est supérieur ou égal au nombre de ressources demandées.

Ainsi le voisin de $l = l_1, ..., l_j, ..., l_N$ avec la modification du site l_j de la tâche j est :

- Si la solution n'est plus réalisable, alors on choisit une autre tâche j
- Sinon : $l = l_1, ..., l'_j, ..., l_N$

Trouver un voisin qui appartient au voisinage V2 correspond à l'algorithme 5.

Algorithme 5 : Algorithme de principe de la recherche d'un voisin dans V2

Entrées : X : Solution initiale;
Variables :
X' : Solution voisine;
Initialisation : $X' \leftarrow X$;
Début
  Tant que $X' = X$ **faire**
    Modification d'un site d'une tâche dans l de X';
    Si X' *n'est pas réalisable* **alors**
      $X' \leftarrow X$;
    Finsi
  Fintq
  Retourner X'
Fin

5.6 Tests d'arrêt et critère d'acceptation

Nous devons définir les tests d'arrêt pour la recherche locale itérée ainsi que pour pour la recherche locale. Pour ces deux algorithmes, nous prenons un même test d'arrêt. Ce test garantit que : *S'il existe pour une solution donnée, une seule solution voisine améliorante, alors la probabilité de la trouver sera de 50%* (Fleury G. 1993). Le principe est de stopper l'algorithme lorsque l'on atteint un certain nombre d'explorations de voisins, sans amélioration stricte de la solution. Ce nombre est appelé "palier" et est calculé selon l'équation (25) :

$$palier = \lfloor ln(2) * |V| \rfloor + 1 \quad (25)$$

La composante $|V|$ représente la taille du voisinage

pour une solution donnée. La taille pour chaque voisinage est la suivante :

- $|V1| \leq (N-1) * (N-2)$
- $|V2| \leq (N-2) * S$

Pour ce qui est du critère d'acceptation de la solution X' après une perturbation et une recherche locale sur la solution X, il est des plus simples :

- Si $H(X') \leq H(X)$ alors X' est retournée
- Sinon on retourne X

6 RÉSULTATS

Maintenant que nous avons présenté nos deux approches de résolution, nous allons étudier leurs performances respectives et comparer leurs résultats sur des instances générées.

6.1 Instances

Nos instances sont des instances que nous avons générées en suivant certaines règles.

6.1.1 Règles communes à toutes les instances

- La durée des tâches est entre 1 et 10 périodes.
- Les sites sont éloignés d'une durée entre 1 et 10 périodes.
- On dispose de 4 types de ressources.
- Une tâche nécessite pour chaque type de ressources, un nombre de ressources inférieur au nombre maximum de ressources disponibles.
- La probabilité que deux tâches possèdent une relation de précédence entre elles est de 5%.
- La probabilité qu'une ressource soit fixe est de 50%.
- L'horizon temporel est égal à cinq fois le nombre de tâches (plus si nécessaire).

6.1.2 Instances générées

Pour générer les instances, on fait varier trois paramètres : le nombre de tâches, le nombre de ressources et le nombre de sites. Les valeurs que prennent ces nombres sont les suivantes :

- Le nombre de ressources : 10 ou 20
- Le nombre de sites : 2 ou 3
- Le nombre de tâches : 5, 10, 15, 20, 25 ou 30

Pour chaque combinaison de paramètres on génère aléatoirement huit instances. Soit 192 instances au total.

6.2 Expérimentations

Nos expérimentations ont toutes été réalisées sur un processeur Intel(R) Xeon(R) CPU E7-8870 @ 2.40GHz. Le modèle mathématique a été résolu à l'aide du logiciel IBM ILOG CPLEX Optimization Studio version 12.4. Les méta-heuristiques ont été implémentée en JAVA. Pour les expérimentations des méta-heuristiques nous avons lancé cinq réplications pour chaque instance. Nous n'avons gardé que le meilleur résultat des cinq expérimentations.

Nous comparons les diverses approches de résolution en temps et par rapport à la qualité de la solution trouvée. Pour mesurer la qualité nous comparons la valeur du makespan. Pour cela nous indiquons la somme sur toutes les instances, des valeurs de la meilleure solution trouvée sur les cinq réplications pour la méta-heuristique. Puis dans un second temps nous donnons l'écart relatif par rapport à la borne supérieure donnée par CPLEX. Nous donnons aussi le pourcentage de fois où la méthode a donnée le meilleur résultat.

6.2.1 Choix des voisinages pour la méta-heuristique

Nous avons dans un premier temps cherché à savoir quelle combinaison des voisinages dans la méta-heuristique donne les meilleurs résultats. Nous avons d'abord recueilli les résultats avec $V1$ le voisinage pour la recherche locale et $V2$ pour la perturbation de la solution. Puis en utilisant $V2$ pour la recherche locale et $V1$ pour la perturbation. Les résultats obtenus sont reportés dans le tableau 1.

Expérimentations	Exp. 1	Exp. 2
Recherche locale	V1	V2
Perturbation	V2	V1
Instances avec 5 tâches	100	100
Instances avec 10 tâches	97	78
Instances avec 15 tâches	75	59
Instances avec 20 tâches	78	47
Instances avec 25 tâches	81	38
Instances avec 30 tâches	75	44
Toutes les instances	84	61

Tableau 1: Pourcentage d'instances où la méta-heuristique obtient la meilleure solution en comparant l'utilisation des voisinages

On constate que l'utilisation du voisinage V2 pour la perturbation et V1 pour la recherche locale donne de

meilleurs résultats en termes de qualité. Par la suite nous utiliserons donc uniquement cette configuration pour les voisinages.

6.2.2 Temps de calculs des méthodes

Dans cette partie nous comparons les temps de calcul de la méthode exacte et de la méthode approchée en fonction des paramètres des instances. La méthode exacte ne s'arrête qu'une fois la valeur optimale trouvée. Étant donné que ce temps peut être extrêmement long pour certaines instances, nous avons fixé le temps maximum d'exécution à trente minutes. Pour une meilleure analyse des résultats nous avons regroupé les instances par caractéristiques identiques. Les résultats sont présentés dans le tableau 2.

On remarque qu'à partir de 15 tâches certaines instances ne sont plus résolues et que nous avons donc juste une borne supérieure et une borne inférieure pour la résolution à l'aide de CPLEX. Le temps n'est pas affiché pour les instances de taille supérieure à 10 dans le tableau 2 car il n'est pas significatif, l'algorithme ne s'étant pas terminé pour toutes les instances. On constate aussi qu'à partir de 20 tâches, peu d'instances sont résolues en moins d'une demi-heure. On remarque que le paramètre qui influe le plus en temps et en qualité de résolution par CPLEX est le nombre de tâches. La méta-heuristique est beaucoup plus rapide puisqu'elle donne une solution pour, par exemple, les 32 instances de 30 tâches en 351 secondes, soit moins de 11 secondes en moyenne par instance. La méta-heuristique est cependant plus impactée par l'augmentation du nombre de sites que la résolution exacte. Cela s'explique du fait qu'un des tests d'arrêt est calculé en fonction du nombre de sites de l'instance (V2).

6.2.3 Qualité des solutions trouvées

Dans cette partie nous comparons la qualité des solutions retournées par la méta-heuristique par rapport à la solution optimale ou la borne supérieure retournée par CPLEX. Les valeurs présentées sont les écarts relatifs entre la borne supérieure retournée par CPLEX et les meilleures solutions retournées par la méta-heuristique. On reporte aussi le meilleur et le pire écart entre ces deux valeurs par rapport à CPLEX (/CPLEX). La comparaison est reportée dans le tableau 3.

Nous constatons que, dès que le nombre de tâches est supérieur à 15, les méta-heuristiques trouvent en moyenne de bien meilleurs résultats que la borne supérieure donnée par CPLEX. De plus, plus les instances prennent en compte un grand nombre de ressources et de sites, plus notre algorithme est efficace par rapport à CPLEX. Cependant sur les instances de 10 et 15 tâches, des solutions pour certaines instances ne sont pas d'aussi bonne qualité que la borne supérieure donnée par CPLEX.

7 CONCLUSION ET PERSPECTIVES

La communauté hospitalière de territoire a fait émerger de nouvelles problématiques dans le domaine de la recherche opérationnelle. Nous avons constaté qu'aucun travail ne traite d'une problématique similaire, cependant certains problèmes de la littérature s'en approchent. En se basant sur le problème du RCPSP nous avons étudié les différentes extensions qui prennent en compte certains aspects de notre problématique et à partir de cela nous avons pu proposer une extension du RCPSP que nous avons appelée Multi Location RCPSP (MLRCPSP).

Nous avons donné un modèle mathématique linéaire, avec uniquement des variables booléennes, pour sa résolution à l'aide de méthodes exactes. Nous avons aussi présenté une méthode de résolution approchée, avec l'utilisation d'une méta-heuristique appelée recherche locale itérée. Nous avons généré des instances de ce problème pour pouvoir tester les deux approches. Les résultats en temps de calcul mais aussi en qualité des solutions ont été présentés et comparés pour les approches exacte et approchée. Les résultats de la méthode approchée sont meilleurs à partir d'une certaine taille d'instances, cependant la qualité des solutions peut encore être améliorée.

Ce problème étant nouveau il existe plusieurs voies pour l'amélioration de la résolution de ce problème. D'autres méta-heuristiques pourraient être utilisées pour la résolution. Le codage proposé dans cet article peut aussi être modifié, nous ne codons que la liste topologique des examens ainsi que l'affectation des tâches au site. Nous déduisons l'affectation des ressources par une heuristique, cependant on peut imaginer un codage qui déterminerait directement l'affectation des ressources. A l'opposé on peut aussi imaginer un codage fixant uniquement une liste topologique des tâches, où l'affectation des sites et des ressources serait déterminée par une heuristique.

REFERENCES

Correia I. and Lourenço L. L. and Saldanha-da-Gama F., 2012. Project scheduling with flexible resources: formulation and inequalities. *OR Spectrum*, vol. 34, p. 635-663.

Fleury, G., 1993. *Méthodes stochastiques et déterministes pour les problèmes NP-difficiles.* Thèse de Doctorat, Université de Clermont-Ferrand 2, Clermont-Ferrand, France.

Gourgand M. and Grangeon N. and Klement N., 2014a. Activities Planning and Resource Assignment on Multi-place Hospital System: Exact and

Taille d'instances	CPLEX temps total (s)	% d'exécution terminées	% écart moyen entre borne inf. et sup.	Méthode approchée temps total (s)
5 tâches	0.89	100	0	0.38
10 tâches	56.33	100	0	3.29
15 tâches	-	87.5	2.2	18.56
20 tâches	-	9.4	20.1	60.88
25 tâches	-	12.5	34.91	147.73
30 tâches	-	0	54.09	350.50
10 ressources	-	54.17	15.58	246.44
20 ressources	-	48.96	21.80	334.88
2 sites	-	52.08	15.87	175.23
3 sites	-	51.04	21.50	406.09
Toutes	-	51.56	18.69	581.32

Tableau 2: Temps de calcul pour les différentes méthodes en fonction des instances

Taille d'instances	% de valeurs optimales retournées par CPLEX	% d'instances où CPLEX obtient la meilleure solution	% d'instances où la méta-heuristique obtient la meilleure solution
5 tâches	100	100	100
10 tâches	100	100	78
15 tâches	87.5	97	44
20 tâches	9.4	63	75
25 tâches	12.5	25	88
30 tâches	0	6	97
10 ressources	54.17	70	79
20 ressources	48.96	60	81
2 sites	52.08	68	84
3 sites	51.04	62	76
Toutes	51.56	65	80

Tableau 3: Pourcentage d'instances où les méthodes obtiennent la meilleure solution

Approach Methods Adapted from the Bin Packing Problem. *HEALTHINF*, Angers, France, p. 117-124.

Gourgand M. and Grangeon N. and Klement N., 2014b. Activities planning and resources assignment on distinct places: A mathematical model. *RAIRO - Operations Research*, vol. 48 à paraître.

Kan, A. R., 1976. *Machine scheduling problem : Classification , complexity and computations*, Martinus Nijhoff.

Lourenço, H. R. and Martin, O. C. and Stutzle, T., 2001. *Iterated local search*. arXiv preprint math/0102188.

Oğuz, O. and Bala, H., 1994. A comparative study of computational procedures for the resource constrained project scheduling problem. *European Journal of Operational Research*, vol. 3, p. 406-416.

Toussaint , H., 2010. *Algorithmique rapide pour les problèmes de tournées et d'ordonnancement*. Thèse de Doctorat, Université Blaise Pascal - Clermont-Ferrand II, France.

Carlier, J., 1984. *Problèmes d'ordonnancement à contraintes de ressources : algorithmes et complexité*. Thèse de Doctorat, Université Paris VI, France.