Modèle mathématique et métaheuristiques pour un problème de mutualisation de ressources en contexte hospitalier

Arnaud Laurent, Laurent Deroussi, Nathalie Grangeon, Sylvie Norre

LIMOS CNRS UMR 6158, Université Blaise Pascal
Campus Universitaire des Cézeaux, 2 rue de la chebarde, TSA 60125,
CS 60026,63173 Aubière Cedex, France
arnaud.laurent@isima.fr,
{laurent.deroussi,nathalie.grangeon,sylvie.norre}@moniut.univ-bpclermont.fr

RÉSUMÉ. Cet article propose une modélisation et une résolution d'un problème de mutualisation de ressources hospitalières dans un contexte multisite. Ce problème est ramené à une extension du resource constrained project scheduling problem : le RCPSP multisite avec mutualisation de ressources entre plusieurs sites qui intègre la prise en compte de nouvelles contraintes telles que les temps de déplacement des ressources et le choix des sites de réalisation des tâches. Une modélisation mathématique du problème est donnée. Trois approches de résolution sont présentées : une recherche locale, un recuit simulé et une recherche locale itérée avec deux critères d'acceptation : un critère d'acceptation de type recuit simulé et un critère de sélection de la meilleure solution. Nous comparons les résultats obtenus avec chaque approche. Les meilleurs résultats sont globalement obtenus avec la recherche locale itérée.

ABSTRACT. This article proposes a modelisation and a resolution of a problem of resource pooling in a hospital context. This problem is an extension of the Resource Constrained Project Scheduling Problem: the Multi-Site RCPSP with resource pooling in a multisite environment. This extension considers new constraints for the RCPSP like transportation times and choice of the site where tasks are executed. A linear program of this problem is given. Three resolution methods are described: local search, simulated annealing and Iterated Local Search with two different acceptance criteria: simulated annealing type acceptance criterion and better acceptance criterion. We compare the results obtained with each method. ILS with simulated annealing type acceptance criterion gives the best results.

MOTS-CLÉS : RCPSP, multisite, ordonnancement, temps de transport, mutualisation de ressources, métaheuristique.

KEYWORDS: RCPSP, multi-Site, scheduling, transportation time, resource pooling, metaheuristic.

DOI:10.3166/JESA.49.677-702

1. Introduction

Le ministère français des Affaires sociales et de la Santé définit la communauté hospitalière de territoire (CHT) comme « une innovation de la loi Hôpital, Patients, Santé, Territoires (HPST). Sa finalité est la recherche de la meilleure utilisation des ressources à disposition des établissements, la complémentarité entre les acteurs ». C'est dans ce cadre que nous nous sommes plus particulièrement intéressés à l'étude de la mutualisation des ressources de l'imagerie médicale entre plusieurs établissements publics de santé. Notre objectif est plus précisément d'ordonnancer les examens des patients, sachant que ceux-ci requièrent à la fois des ressources matérielles, seulement disponibles dans certains établissements, et des ressources humaines qui peuvent se déplacer d'un établissement à l'autre, et donc être mutualisées.

Afin de traiter ce problème, nous proposons de le modéliser à l'aide d'une nouvelle extension du RCPSP (*Resource Constrained Project Scheduling Problem*) que nous appelons le RCPSP multisite. Nous présentons un modèle mathématique sous forme d'un programme linéaire en nombres entiers ainsi qu'une résolution par des métaheuristiques.

Dans la section 2, nous revenons plus en détail sur le contexte de la CHT et notre choix de le modéliser par un RCPSP multisite. La section 3 est consacrée à un état de l'art du RCPSP et des extensions pertinentes à notre étude, ainsi que des problèmes d'ordonnancement qui prennent en compte le transfert de tâches et de ressources. Nous présentons dans la section 4 le programme linéaire du RCPSP multisite. Les métaheuristiques utilisées sont décrites dans la section 5. L'étude expérimentale est présentée dans la section 6 avant de conclure et de donner les perspectives de nos travaux.

2. Problématique

2.1. Description

Un des enjeux de la CHT est de mutualiser ses ressources afin d'améliorer la performance des établissements de santé et d'assurer une meilleure qualité de soins pour les patients. Dans cette étude nous nous plaçons dans le cadre d'une convention passée entre plusieurs établissements pour améliorer le planning des examens d'imagerie médicale (IRM, scanner...). Un ensemble de patients doit subir une liste d'examens médicaux nécessitant chacun différentes ressources matérielles et humaines qui sont toutes mono-compétentes. Par exemple pour qu'un patient passe un scanner, deux manipulateurs et un médecin sont nécessaires.

– Les ressources matérielles sont des appareils fixes, par conséquent associées à un site et ne peuvent en changer. Une ressource matérielle est caractérisée par son type.

– Les ressources humaines sont également caractérisées par leur type. De telles ressources sont partagées entre plusieurs sites distants les uns des autres. Un temps

de déplacement doit être pris en compte lorsqu'une ressource humaine se déplace entre deux sites.

Les patients sont obligatoirement présents sur site pendant toute la durée de leurs examens. Chaque examen est défini par une durée, indépendante des ressources affectées. Un patient ne peut pas passer deux examens en même temps, et met un certain temps pour aller d'un site à un autre si ces deux examens n'ont pas lieu dans le même site. Un patient est caractérisé par une liste ordonnée d'examens.

L'objectif est de déterminer, pour chaque examen de chaque patient, son lieu d'exécution, sa date de début et les ressources qui lui sont affectées tout en minimisant la durée totale de réalisation de tous les examens (makespan).

2.2. Exemple

Prenons une instance de CHT simplifiée et de petite taille, avec 3 patients qui doivent passer différents examens sur deux sites éloignés d'une durée de 4 périodes. Pour les réaliser, nous disposons de 3 manipulateurs, d'un IRM dans le site 1 et d'un scanner dans le site 2. Le tableau 1 précise les examens que doit passer chaque patient ainsi que leur durée et les ressources nécessaires. Notons que le nombre de manipulateurs requis pour un même examen peut différer d'un patient à l'autre en fonction de son état de santé ou de son âge.

Tableau 1. Exemple d'instance

	Examen 1	**Examen 2**
Patient 1	IRM – Durée : 3 2 manipulateurs	Scanner – Durée : 4 2 manipulateurs
Patient 2	Scanner – Durée : 2 1 manipulateur	IRM – Durée : 2 1 manipulateur
Patient 3	IRM – Durée 2 1 manipulateur	

Une solution de cette instance est donnée sur la figure 1. On peut voir que l'un des manipulateurs et que les patients 1 et 2 doivent se déplacer entre les deux sites. Ces déplacements influent sur les dates de début des seconds examens que doivent passer les patients 1 et 2.

3. Les problèmes similaires de la littérature

Lors de précédents travaux effectués dans l'équipe (Gourgand *et al*., 2014), nous nous sommes intéressés à ce même problème de la CHT à un niveau de décision tactique. L'objectif était alors de planifier les examens des patients sur les sites sur deux ou trois mois à la maille jour. Ce problème avait été modélisé par un bin-

packing dans lequel une boîte représentait un couple composé d'une ressource matérielle et d'une période. La capacité d'une boîte correspondait au temps d'ouverture de la ressource sur la période. Les hypothèses simplificatrices principales consistaient à ne pas prendre en compte les temps de transport entre les sites et supposaient que la réalisation d'un examen ne nécessitait qu'une seule ressource. Pour résoudre ce problème, des couplages entre des métaheuristiques (recherche locale itérée et PSO) et des algorithmes de liste (*Best-Fit et First-Fit*) ont été proposés.

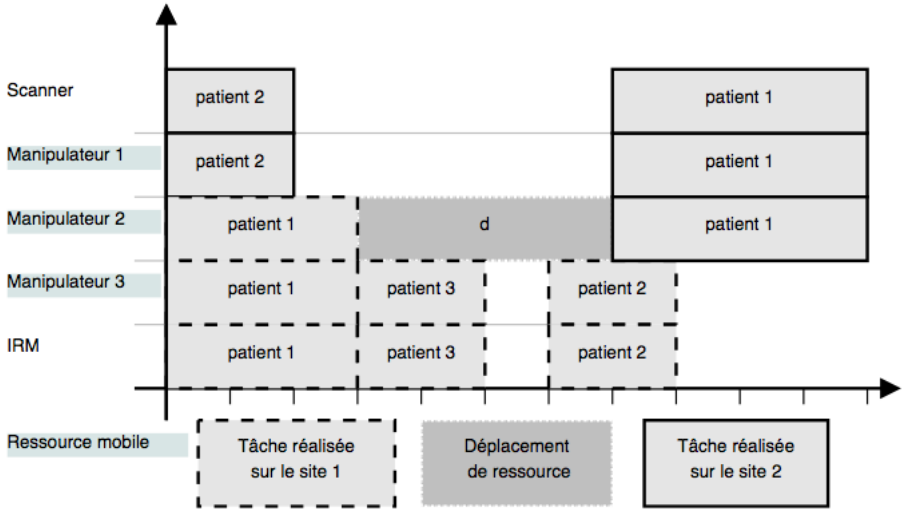

Figure 1. Exemple de solution

Nous poursuivons ici ces travaux en étudiant ce même problème à un niveau de décision opérationnel. Nous sommes amenés à résoudre un problème d'ordonnancement. Nous avons choisi de le modéliser par un RCPSP (*Resource Constrained Project Scheduling Problem*) dans lequel les tâches à ordonnancer représentent les examens, et ce pour plusieurs raisons :

– le RCPSP est un problème d'ordonnancement générique,

– les examens nécessitent une seule ressource matérielle (appareil). Par contre, elles peuvent requérir plusieurs ressources humaines, parfois de même type,

– les patients peuvent être amenés à passer plusieurs examens selon un certain ordre.

Le RCPSP a l'avantage de prendre en compte des graphes de précédence quelconques. Dans le cas de la CHT, les graphes de précédence sont particuliers puisque constitués de plusieurs chaînes (une chaîne représente l'ensemble ordonné des examens que doit subir un patient). Par contre, le RCPSP ne permet pas de prendre en compte la gestion des temps de transfert des patients et/ou des ressources

entre les sites. Nous verrons dans cette partie, au travers de l'étude des extensions du RCPSP, qu'aucune d'elles n'est réellement satisfaisante. Nous introduirons alors le RCPSP multi-site qui fera l'objet de notre étude.

Le RCPSP multisite permettra d'intégrer les aspects mutualisation de ressources dans le contexte de la CHT. Il pourra également servir de cadre d'étude pour d'autres problèmes industriels. Nous avons eu par exemple l'occasion d'étudier par le passé un problème de planification d'essais pneumatiques, nécessitant comme ressources des véhicules, des pilotes et du matériel embarqué, et pouvant se dérouler sur deux circuits, l'un situé en Auvergne, l'autre dans le sud de la France avec des ressources qui pouvaient se déplacer d'un circuit à l'autre.

3.1. Le RCPSP

Le problème classique de gestion de projet à contraintes de ressources, appelé RCPSP pour *Resource Constrained Project Scheduling Problem* est noté PS|prec|Cmax par (Kan, 1976). Ce problème est NP-difficile, ce qui a été démontré par (Blazewicz *et al.*, 1983). L'objectif est de réaliser un ensemble de N tâches en utilisant un ensemble de ressources renouvelables de K types différents. 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 pour exécuter toutes les tâches. Les tâches sont non-préemptives, une fois commencées elles ne peuvent pas être stoppées pendant toute la durée de leur exécution p_j (en nombre de périodes). Chaque tâche possède une liste P_j de tâches qui doivent la précéder. Chaque tâche nécessite une affectation de $r_{j,k}$ ressources de type k. On a une quantité R_k de ressources de type k disponibles. La tâche N est la tâche fictive de fin de projet qui doit succéder à toutes les autres tâches. Une formalisation mathématique de ce problème a été proposée par (Oguz, Bala, 1994).

– Données

J	Ensemble de N tâches
P_j	Ensemble de tâches qui doivent précéder $j \in J$
p_j	Durée de la tâche $j \in J$
R	Ensemble des ressources
K	Nombre de types de ressource
T	Nombre de périodes maximum
R_k	Nombre de ressources de type $k = 1,\ldots,K$ disponibles
$r_{j,k}$	Nombre de ressources de type k nécessaires pour la tâche $j \in J$

– Variables

$X_{j,t} = 1$ si la tâche $j \in J$ se termine à la période $t = 1,\ldots,T$; 0 sinon

$$\min \sum_{t=1}^{T} t\ X_{N,t} \tag{1}$$

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

$$\sum_{t=1}^{T} t\ X_{j,t} \geq \sum_{t=1}^{T} t\ X_{j',t} + p_j, \forall j \in J, \forall j' \in P_j \tag{3}$$

$$\sum_{j=1}^{N} \left(r_{j,k} \sum_{q=t}^{t+p_j-1} X_{j,q} \right) \leq R_k, \forall k \in R \tag{4}$$

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

Le but est de minimiser la durée du projet, appelée makespan (1). Les contraintes du problème sont : 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 de bivalence (5).

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

Le principe des time lags est de rajouter des délais minimaux à respecter entre la fin d'une tâche et le début d'une autre. 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. Il s'applique 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)*. Ainsi l'équation (6) remplace l'équation (3) dans le modèle du RCPSP classique :

$$\sum_{t=1}^{T} t\ X_{j,t} \geq \sum_{t=1}^{T} t\ X_{j',t} + p_j + lagmin(j', j), \forall j \in J, \forall j' \in P_j \tag{6}$$

De nombreux travaux se sont intéressés à la prise en compte de ces time lags minimaux comme par exemple (Chassiakos, Sakellaropoulos, 2005) ; Klein, 2000 ; Klein, Scholl, 2000 ; Kolisch, 1998 ; Klein, Scholl, 1999 ; Lombardi, Milano, 2009) et (Vanhoucke, 2004).

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

Le RCPSP avec time lags conditionnels a été proposé par Toussaint (2010). C'est une extension au problème de RCPSP avec time lags minimaux. Un time lag est donné entre chaque couple de tâches mais son application est conditionnelle. Le principe repose sur le fait de prendre en compte le time lag si au moins une des conditions suivantes est vérifiée :

– les deux tâches partagent au moins une ressource,

– il existe une contrainte de précédence entre les deux tâches.

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 qui doivent s'appliquer entre deux tâches consécutives (qui modélisent par exemple le transport d'un produit) ou des temps d'acheminement de ressources sur le lieu d'exécution d'une tâche. L'équation (6) s'applique donc sous condition qu'un des deux cas précédents soit avéré, ce qui n'a jamais été à notre connaissance modélisé mathématiquement. La prise en compte des time lags nécessite de connaître par avance le lieu où les tâches vont être exécutées dans le cas où ils représentent des temps de transfert.

3.4. Les problèmes d'ordonnancement avec prise en compte de temps de transfert

Plusieurs problèmes d'ordonnancement prennent en compte des temps de transfert de ressources ou d'activités. Les premiers problèmes d'ordonnancement où des temps de transport sont pris en compte par (Maggu, Das, 1980) et (Maggu *et al.*, 1981), sont des flowshop à deux machines. Leur travail consiste à adapter l'algorithme de Johnson (1954) pour résoudre le même problème de flowshop à deux machines mais avec des temps de transport des tâches entre elles. Le second article (Maggu *et al.*, 1981) ajoute la contrainte que certaines tâches doivent se dérouler consécutivement. Yu (1996) étudie plusieurs extensions du problème de (Maggu, Das, 1980). Le problème devient NP-difficile si l'on considère qu'un seul transporteur effectue les transferts de tâches entre les machines (Kise, 1991). D'autres articles traitent des problèmes de flowshop à deux machines sans parler directement de temps de transfert. C'est notamment le cas de (Mitten, 1959) qui considère des time lags entre le début de l'exécution de la tâche sur la première machine et le début de l'exécution de la tâche sur la deuxième machine. Ce problème est repris par (Maggu *et al.*, 1982) qui ajoutent des dates de début des tâches au plus tôt et au plus tard. Lee, Chen (2001) généralisent ces problèmes pour un flowshop à m machines avec un temps de transport entre deux étapes de fabrication. Les tâches sont transportées par n véhicules avec une capacité c. Le temps de transport est déterminé pour chaque couple de machines successives. Les critères minimisés sont le makespan et la somme des dates de fin d'exécution des tâches.

Dans le cas d'un flowshop hybride à k étages avec un choix possible entre plusieurs machines, (Langston, 1987) considère un transport des jobs entre chacun des différents étages. Ce transfert est effectué par un unique transporteur et le temps de ce transport dépend de la localisation des deux machines à relier.

Le problème de placement sur architectures multiprocesseurs avec coûts de communication inter-tâche proposé par (Garey et Johnson, 1979), prend en compte des temps de transfert entre deux tâches dans le cas où il existe une contrainte de précédence entre elles. Le temps de transfert est nul si les deux tâches sont associées au même processeur, sinon ce temps dépend uniquement du couple de tâches. Les auteurs ont prouvé que le problème devient NP-difficile même avec un nombre

infini de processeurs. Plusieurs heuristiques ont été proposées pour résoudre ce problème, notamment des algorithmes de liste comme ELS (*Extended List Schedule*) par (Hwang *et al*, 1989) ou bien MCP (*Modified Critical Path*) par (Wu, Gajski, 1988). Des résolutions par métaheuristiques ont aussi été proposées par (Norre, 1993).

Des extensions du jobshop classique existent avec une prise en compte des temps de transport, c'est notamment le cas de (Bilge, Ulusoy, 1995) qui se sont intéressés à l'ordonnancement simultané des ressources de production et de transport dans les systèmes flexibles de production. Ils ont pour cela proposé une extension du job-shop dans laquelle les pièces sont transportées d'une machine vers une autre au moyen d'un réseau de transport (chariots filoguidés).

En conclusion, nous avons pu voir que de nombreux problèmes d'ordonnancement intègrent la prise en compte de temps de transfert. Il nous apparaît donc d'autant plus intéressant de modéliser et d'étudier le RCPSP multi-site qui pourra servir de cadre pour traiter notre problématique de la CHT.

4. Modélisation mathématique du RCPSP multi-site

L'extension que nous proposons est d'ajouter au problème de type RCPSP un contexte multisite avec le choix du lieu d'exécution pour les tâches. Pour cela nous introduisons la notion de sites et de ressources fixes ou mobiles. On dispose de sites et chaque couple de sites est séparé par un temps de déplacement. Une tâche a obligatoirement besoin d'un site pour être exécutée. Une ressource est soit mobile, soit fixe. La particularité des ressources fixes est qu'elles peuvent être affectées uniquement aux tâches exécutées sur leur site de référence. Les ressources mobiles sont sujettes à des temps de déplacement dans le cas où elles doivent exécuter successivement deux tâches sur deux sites différents. Ce temps de déplacement se traduit par un délai minimum entre la fin de la première tâche et le début de la seconde, égal au temps de déplacement entre les deux sites. Le deuxième cas où ce délai s'applique, est s'il existe une contrainte de précédence entre deux tâches qui ne sont pas réalisées sur le même site.

Contrairement au RCPSP classique il est nécessaire d'identifier individuellement les ressources qui vont effectuer les tâches. Les time lags ne sont plus donnés pour deux tâches puisqu'ils dépendent des sites où vont être exécutées les tâches et ces sites sont à déterminer pour construire une solution au problème. Le problème du RCPSP avec time lags conditionnels consiste à trouver un ordonnancement des tâches et une affectation des ressources pour effectuer l'ensemble des tâches, qui minimisent un critère donné, par exemple le makespan. Pour le RCPSP multi-site, l'objectif est le même mais il faut en plus déterminer l'affectation des tâches aux sites qui a une incidence sur les temps de transport.

4.1. Notations

Les notations présentées sont celles fréquemment utilisées pour la modélisation de problèmes de type RCPSP et ses extensions. Elles sont complétées par de nouvelles notations propres au caractère multisite.

– Données

N Nombre de tâches, N représente la tâche fictive de fin de projet.

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

p_j Durée de la tâche $j = 1,\ldots,N$

K Nombre de types de ressources

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

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

T Nombre maximum de périodes

$M_{k,r} = \begin{cases} 1 \text{ si la ressource } r = 1,\ldots,R_k \text{ de type } k = 1,\ldots,K \text{ est mobile} \\ 0 \text{ si elle est fixe} \end{cases}$

S Nombre de sites

$\delta_{s,s'}$ Temps de déplacement entre le site $s = 1,\ldots,S$ et le site $s' = 1,\ldots,S$

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

H Un grand nombre

– Variables

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

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

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

$\omega_{j,h} = 1$ si un temps de déplacement doit être pris en compte entre la fin de la tâche $j = 1,\ldots,N$ et le début de la tâche $h = 1,\ldots,N$; 0 sinon. C'est le cas lorsque ces deux tâches partagent une ressource ou lorsqu'elles sont reliées par une contrainte de précédence. Toutefois, si les tâches *j* et *h* sont exécutées sur le même site, le temps de déplacement est nul.

4.2. Modèle mathématique

Notre problème est modélisé sous la forme du programme linéaire en nombres entiers :

$$\min \sum_{t=1}^{T} t \; X_{N,t} \tag{7}$$

$$\sum_{t=1}^{T} X_{j,t} = 1;\ j = 1,...,N \tag{8}$$

$$\omega_{j,h} = 1;\ h = 1,...,N;\ j \in P_h; \tag{9}$$

$$Y_{j,k,r} + Y_{h,k,r} \le \omega_{j,h} + \omega_{h,j} + 1\ ,\ (j,h = 1,...,N) \wedge (j < h);\ k = 1,...,K;\ r = 1,...,R_k \tag{10}$$

$$\sum_{t=1}^{T} t\ X_{j,t} \ge \sum_{t=1}^{T} t\ X_{h,t} + p_j + \left(Z_{j,s} + Z_{h,s'} - 1\right)\delta(s,s') - H(1-\omega_{h,j});$$
$$j,h = 2,...,N-1;\ s,s' = 1,...,S \tag{11}$$

$$\sum_{t=1}^{T} t\ X_{j,t} \ge \sum_{t=1}^{T} t\ X_{h,t} + p_j\ ,$$
$$\left\{(h = 2,...,N-1) \wedge \left(j = N\right)\right\} \cup \left\{\left(h = 1\right) \wedge \left(j = 2,...,N-1\right)\right\} \tag{12}$$

$$\sum_{r=1}^{R_k} Y_{j,k,r} = r_{j,k};\ j = 1,...,N;\ k = 1,...,K \tag{13}$$

$$Y_{j,k,r} \le Z_{j,loc_{k,r}};\ j = 1,...,N;\ \left(k = 1,...,K;r = 1,...,R_k\right) \wedge \left(M_{k,r} = 0\right) \tag{14}$$

$$\sum_{s=1}^{S} Z_{j,s} = 1;\ j = 1,...,N; \tag{15}$$

$$X_{j,t} \in \{0,1\};\ j = 1,...,N;\ t = 1,...,T \tag{16}$$

$$Y_{j,k,r} \in \{0,1\};\ j = 1,...,N;\ k = 1,...,K;\ r = 1,...,R_k \tag{17}$$

$$Z_{j,s} \in \{0,1\};\ j = 1,...,N;\ s = 1,...,S \tag{18}$$

$$\omega_{j,h} \in \{0,1\};\ j,h = 1,...,N \tag{19}$$

Le modèle proposé s'appuie et étend les modèles de la littérature de (Oguz, Bala, 1994) et (Correia *et al.*, 2012). Le but est de minimiser le makespan (7). La non préemption et la réalisation des tâches sont assurées par la contrainte (8).

Les contraintes (9), (10) et (11) permettent de modéliser qu'un temps de déplacement est à prendre en compte dans les deux cas suivants :

– Deux tâches sont reliées par une contrainte de précédence (contrainte (9)).

– Une même ressource est affectée à deux tâches. Dans ce cas, les deux tâches ne peuvent pas avoir lieu en même temps (contrainte (10)).

Dans les deux cas, les temps de déplacement sont nuls si les deux tâches sont réalisées sur le même site. La contrainte (11) exprime le calcul des dates de fin des tâches en intégrant les temps de déplacement qui sont fonction des sites sur lesquels les tâches sont affectées. Notons que cette modélisation peut s'adapter très

facilement au cas des time lags conditionnels dans lequel la notion de sites n'apparait pas. Il convient alors de remplacer le terme $\left(Z_{j,s} + Z_{h,s'} - 1\right)\delta\left(s, s'\right)$ par la valeur du time lag entre les tâches *j* et *h*.

Le respect des dates de fin de tâche par rapport aux tâches fictives est assuré par la contrainte (12). La contrainte (13) assure que les quantités nécessaires de ressources sont affectées. Chaque tâche doit être exécutée sur le site d'appartenance des ressources fixes affectées (si des ressources fixes sont affectées) (14). Chaque tâche est exécutée sur un seul site (15). Les contraintes de bivalence sur les variables sont les contraintes (16) (17) (18) et (19).

Dans cette modélisation, la prise en compte de l'affectation des tâches aux sites impose l'identification des ressources. Les contraintes de respect des quantités de ressources disponibles (données par les contraintes (4) dans le modèle de base du RCPSP) ne figurent donc plus en tant que telles. Elles sont assurées par les contraintes (10) et (11) dans le modèle du RCPSP multi-site qui indiquent qu'à un instant donné une ressource effectue au plus une action (réalisation d'une tâche ou d'un déplacement).

4.3. Exemple

Afin d'illustrer notre problème, nous considérons une instance du RCPSP multi-site comportant 2 sites, 4 types de ressources et 7 tâches. Les caractéristiques des ressources sont données dans le tableau 2. Le temps de transport d'un site à l'autre est de deux périodes. Les 7 tâches sont reliées par des contraintes de précédence données sur la figure 2. Les tâches 1 et 9 sont les deux tâches fictives de début et fin de projet. Chaque tâche est représentée par un cercle avec au-dessus sa durée et en-dessous les ressources nécessaires à son exécution sous la forme type (quantité).

Tableau 2. Liste des resources disponibles

	Type	Statut	Site d'appartenance
R1,1	1	Fixe	1
R1,2	1	Mobile	-
R2	2	Fixe	2
R3	3	Mobile	-
R4	4	Mobile	-

Une solution est donnée sur la figure 3. On peut voir que sur cette solution les deux ressources mobiles R3 et R1.2 se déplacent. R3 exécute les tâches 2 et 4 sur le site 1 puis se déplace sur le site 2 pour effectuer la tâche 5. La seconde ressource R1 exécute les tâches 2 et 4 sur le site 1 avant de se déplacer sur le site 2 pour réaliser la tâche 6. Par contre, R4, mobile reste sur le site 2. Ainsi des temps de déplacement

s'appliquent pour chacun de ces déplacements de ressources. Des temps de déplacement, s'appliquent aussi entre les tâches 2 et 3, ainsi qu'entre les tâches 4 et 5 puisqu'elles ne sont pas effectuées sur le même site et sont reliées par des contraintes de précédence. On obtient donc une solution avec un makespan de 12 périodes.

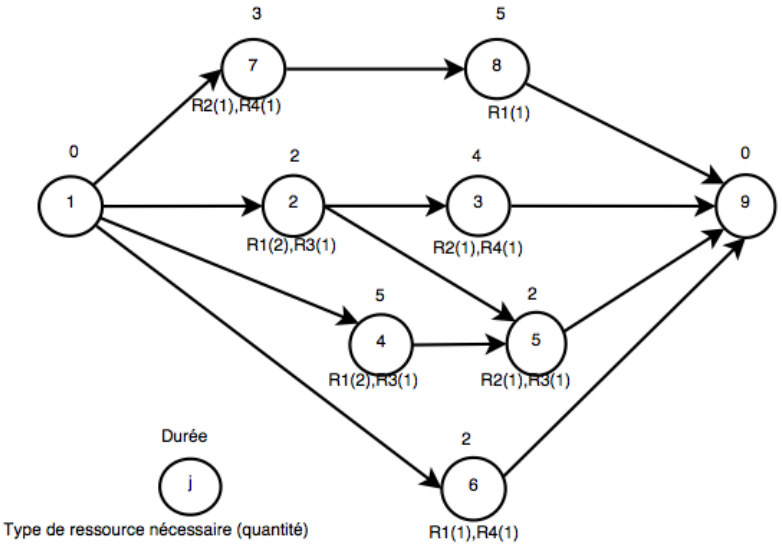

Figure 2. Graphe de précédence du problème

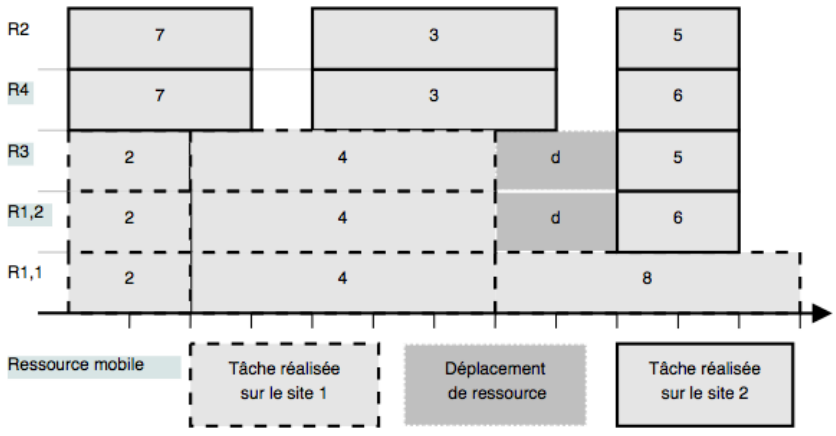

Figure 3. Solution au problème

5. Proposition de méthodes approchées

Ce problème faisant partie des problèmes NP-difficiles au sens fort, des méthodes de résolution approchées semblent être les plus adaptées pour des instances de taille moyenne et grande. Nous proposons d'utiliser des métaheuristiques basées individu. Nous présentons donc dans un premier temps le codage utilisé pour représenter une solution, puis nous proposons un système de voisinage composé de deux mouvements. Enfin, nous détaillons les métaheuristiques utilisées.

5.1. Codage d'une solution

Nous proposons de représenter une solution X par la concaténation de deux vecteurs :

$$X = (\sigma, l) \tag{20}$$

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

– $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ est une permutation des *N* tâches qui désigne une liste topologique respectant les contraintes de précédence. σ_1 et σ_N sont les tâches fictives de début et de fin de projet. Chaque ressource exécutera les tâches dans l'ordre de cette séquence, c'est-à-dire que si une tâche *j* précède une tâche *i* dans σ, alors aucune ressource ne pourra exécuter *i* avant *j*.

– $l = l_1, l_2, \ldots, l_N$ avec $l_j = 1, \ldots, S$ le site où la tâche $j = 1, \ldots, N$ va être exécutée. Cette affectation respecte le nombre de ressources disponibles sur le site : une tâche ne peut pas être affectée à un site qui ne pourra pas l'exécuter.

Un ordonnancement est également défini par une date de début pour toutes les tâches ainsi qu'une affectation des ressources aux tâches. Ce codage seul ne représente pas un ordonnancement, mais potentiellement plusieurs ordonnancements qui respectent la séquence σ et l'affectation *l*. Pour déterminer un ordonnancement, nous appliquerons à ce codage un algorithme d'ordre strict, en nous inspirant des travaux de Carlier (1984).

5.2. Algorithme d'ordre strict

Un 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, 1984).

Dans notre cas, nous disposons d'une liste topologique σ et d'un vecteur de sites *l*. A chaque itération de l'algorithme, le vecteur *l* va permettre de déterminer les temps de déplacement des ressources entre les sites et ainsi de calculer la date de fin

d_j de la tâche $j = 1,\ldots,N$. L'algorithme détermine l'affectation des ressources aux tâches et la date de fin des tâches dans l'ordre σ.

Pour une ressource $r = 1,\ldots,R_k$ de type $k = 1,\ldots,K$, nous distinguons trois cas pour calculer sa date de disponibilité $av_{k,r}$ pour réaliser la tâche *j*:

– Soit cette ressource est mobile et sa date de disponibilité au plus tôt pour commencer la tâche est égale à :

- $av_{k,r} = 0$ si elle n'a encore aucune tâche assignée
- $av_{k,r} = d_h + \delta\left(l_h, l_j\right)$ si sa dernière tâche assignée est *h*

– Soit cette ressource est fixe telle que $l_j \neq loc_{k,r}$ et la ressource ne peut pas être affectée à *j* : $av_{k,r} = +\infty$

– Soit cette ressource est fixe telle que : $l_j = loc_{k,r}$ et sa date de disponibilité au plus tôt pour commencer la tâche $j = 1,\ldots,N$ est égale à $av_{k,r} = d_h$ 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 $av_{k,r}$ les plus petites. La date de fin d'exécution d_j de la tâche *j* est donnée par l'équation (21).

$$d_j = \max\left(A, B\right) + p_j \tag{21}$$

$$A = \max\left(d_h + \delta\left(l_h, l_j\right), h \in P_j\right) \tag{22}$$

$$B = \max\left(av_{k,r},\ \forall k, r / Y_{j,k,r} = 1\right) \tag{23}$$

A représente la plus grande date de fin (temps de déplacement inclus) des tâches qui doivent précéder *j*. *B* représente la plus grande date de disponibilité des ressources que l'on a assignées à *j*. L'algorithme 1 permet de définir les dates de fin d'exécution de chaque tâche et donc de calculer différents critères d'évaluation basés sur les dates, dont le makespan.

5.3. Système de voisinage

5.3.1. Le mouvement V1

Le premier mouvement (algorithme 2) concerne σ. C'est un mouvement de type insertion. Le principe est de déplacer une tâche σ_p de la position $p = 2,\ldots,N-1$ dans le vecteur σ à la position $p' = 2,\ldots,N-1$ $\left(p' \neq p\right)$. 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 σ_p précède $\sigma_{p''}$

– $p > p'$ et il existe au moins une tâche à la position p'' entre p et p' telle que $\sigma_{p''}$ précède σ_p

Algorithme 1. Algorithme d'ordre strict pour le RCPSP Multisite (H(X))

Entrées : $X = (\sigma, l)$
Variables : d : Vecteur des dates de fin des tâches;
av : Matrice des dates de disponibilité des ressources;
Initialisation : $d := \{0, ..., 0\}$;
1 **Debut**
2 **Pour** $j = 1$ *à* N **faire**
3 Calculer les valeurs de av pour σ_j en prenant en compte l et d;
Affecter les ressources à σ_j;
Calculer la date de fin d_{σ_j} (21);
4 **Finpour**
5 **Retourner** d_N
6 **Fin**

Algorithme 2. Algorithme de principe du mouvement V1

Entrées : σ : séquence initiale;
Variables : σ' : séquence voisine;
1 **Debut**
2 **Répéter**
3 Choisir aléatoirement et uniformément deux positions $p \in [2, N-1]$ et $p' \in [2, N-1]/p' \neq p$
4 **Jusqu'à** *le déplacement de* σ_p *à la position* p' *ne viole pas les contraintes de précédence*;
5 **Si** $p < p'$ **alors**
6 $\sigma' = \sigma_1, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma_{p'}, \sigma_p, ..., \sigma_N$
7 **sinon**
8 $\sigma' = \sigma_1, ..., \sigma_{p'}, \sigma_p, ...\sigma_{p-1}, \sigma_{p+1}, ..., \sigma_N$
9 **Finsi**
10 **Retourner** σ'
11 **Fin**

5.3.2. Le mouvement V2

Le deuxième mouvement (algorithme 3) consiste à modifier le site s assigné à une tâche σ_p $j = 1, \ldots, N-1$ en un site s'. Le mouvement est appliqué seulement si la solution reste réalisable c'est à dire si, pour chaque type r de ressource nécessaire

à *j*, le nombre de ressources mobiles de type *r*, plus le nombre de ressources fixes de type *r* sur le site s' est supérieur ou égal au nombre de ressources demandées.

Algorithme 3. Algorithme de principe du mouvement V2

Entrées : l : vecteur initial;
Variables : l' : vecteur voisin;
1 **Debut**
2 **Répéter**
3 Choisir aléatoirement et uniformément une tâche j ;
  Choisir aléatoirement et uniformément un site $l'_j \neq l_j$ parmi les site qui peuvent traiter la tâche;
4 **Jusqu'à** *j peut être exécutée sur le site l'_j*;
5 $l' = l_1, ..., l'_j, ..., l_N$
  Retourner l'
6 **Fin**

5.3.3. Système de voisinage

Nous proposons d'utiliser un système de voisinage qui consiste à appliquer V1 ou V2 selon une probabilité. Ce système de voisinage garantit l'accessibilité et la réversibilité.

5.4. La recherche locale

Le principe de la recherche locale (algorithme 4) 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 (en fonction du test d'arrêt) 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 jusqu'à ce qu'on estime avoir obtenu un minimum local.

5.5. Le recuit simulé

Le principe du recuit simulé (algorithme 5) permet d'éviter le problème majeur d'une recherche locale, le fait de converger dans un minimum local. Le recuit simulé permet d'accepter des transitions dégradantes avec une certaine probabilité. La probabilité $p(X',i)$ d'accepter une solution X' voisine d'une solution courante X, va dépendre d'une valeur décroissante appelée température T_i avec *i* l'itération courante. La probabilité d'accepter une solution X' telle que $H(X)-H(X')<0$ à l'itération *i*, est :

Algorithme 4. Algorithme de principe de la recherche locale stochastique

Entrées : X_0 : Solution initiale aléatoire et réalisable;
Variables : X : Solution courante;
X' : Solution candidate;
Initialisation : $X := X_0$;
1 **Debut**
2 **Tant que** *Test d'arrêt est faux* **faire**
3 X' := choisir aléatoirement et uniformément un voisin dans le système de voisinage de X;
Si $H(X') \leq H(X)$ **alors**
4 X :=X';
5 **Finsi**
6 **Fintq**
7 **Retourner** X
8 **Fin**

$$p(X', i) = \exp\left(\frac{H(X) - H(X')}{T_i}\right) \tag{24}$$

Nous présentons dans cette partie un recuit simulé inhomogène dans lequel la température diminue à chaque itération suivant une progression géométrique. Ainsi à chaque itération *i*, la température sera mise à jour de la façon suivante :

$$T_i = \alpha T_{i-1}, \forall i = 1, \ldots, iterMax \tag{25}$$

On fixe la température de départ selon l'algorithme de (Kirkpatrick, 1984). Pour que notre algorithme converge efficacement vers une solution optimale, on se fixe T_0 et T_a les températures de début et de fin souhaitées, ainsi qu'un nombre d'itérations *iterMax*. On en déduit α selon l'équation suivante :

$$\alpha = \sqrt[iterMax]{\frac{T_a}{T_0}} \tag{26}$$

5.6. La recherche locale itérée

Le principe de la recherche locale itérée (Lourenço *et al.*, 2003) 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 6.

Algorithme 5. Algorithme de principe du recuit simulé

Entrées : X_0 : Solution initiale aléatoire et réalisable;
T : Température;
Variables : X^* : Meilleure solution trouvée;
X : Solution courante;
X' : Solution voisine;
Initialisation : $X := X_0$;$X^* := X_0$
1 **Debut**
2 **Tant que** *Test d'arrêt est faux* **faire**
3 $X' :=$ choisir aléatoirement et uniformément un voisin dans le système de voisinage de X;
 Si $exp(\frac{H(X)-H(X')}{T}) > random[0,1)$ **alors**
4 $X := X'$
 Si $H(X) < H(X^*)$ **alors**
5 $X^* := X$
6 **Finsi**
7 **Finsi**
8 $T := \alpha * T$;
9 **Fintq**
10 **Retourner** X^*
11 **Fin**

Algorithme 6. Algorithme de principe de la recherche locale itérée

Entrées : X_0 : Solution initiale aléatoire et réalisable;
Variables : X^* : Meilleure solution trouvée;
X' : Solution voisine;
Initialisation : $X^* :=$ Recherche locale sur X_0;
1 **Debut**
2 **Tant que** *Test d'arrêt est faux* **faire**
3 $X' :=$ Perturbation de X^* ;
 $X' :=$ Recherche locale sur X';
 $X^* :=$ Critère d'acceptation de X' par rapport à X^* en prenant en compte l'historique ;
4 **Fintq**
5 **Retourner** X^*
6 **Fin**

Pour parcourir notre espace de recherche nous devons déterminer les critères d'acceptation entre la solution courante après recherche locale et la solution voisine après recherche locale. Nous en utiliserons deux différents (Correia *et al.*, 2012).

– celui de la recherche locale (ILS|LS) qui consiste à remplacer la solution courante *X* par la solution candidate *X*' si sa qualité est équivalente ou meilleure :

Si $H(X') \leq H(X^*)$ alors retourner X'

Sinon retourner X^*

– celui du recuit simulé inhomogène (ILS|SA) (Metropolis *et al.*, 1953) qui consiste à accepter toutes les solutions de qualité équivalente et meilleure ainsi que des solutions moins bonnes selon une probabilité décroissante dans le temps. Ce critère est celui provenant du recuit simulé inhomogène (algorithme 5) :

Si $\exp\left(\frac{H(X^*) - H(X')}{T_i}\right) > random[0,1)$ alors retourner X'

Sinon retourner X^*

6. Expérimentations

6.1. Instances

Dans cette étude préliminaire, nous n'avons pas encore intégré de données réelles (dont nous ne disposons pas). Les instances que nous avons générées correspondent au problème de RCPSP multisite. Nos premières expérimentations ont porté sur des instances de taille raisonnable (de 10 à 30), ceci afin de pouvoir obtenir des solutions optimales avec le modèle mathématique, comparer nos résultats avec les métaheuristiques et valider leur efficacité. Pour toutes les instances, les règles de génération suivantes ont été adoptées :

– La durée des tâches est comprise entre 1 et 10 périodes.

– Les sites sont éloignés d'une durée comprise 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 de ce type.

– Pour chaque couple de tâches, la probabilité est de 5% qu'elles soient liées par une relation de précédence. Ainsi pour une instance de taille 30, la probabilité qu'une tâche possède au moins une relation de précédence, avec une tâche non fictive, est supérieure à 77%.

– 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).

Les instances sont répertoriées selon différentes classes dépendant du nombre de tâches, du nombre de ressources et du nombre de sites. Les valeurs que prennent ces nombres sont les suivantes :

– nombre de ressources ($\sum_{k=1}^{K} R_k$): 10 ou 20 (quel que soit leur type)

– nombre de sites (S) : 2 ou 3

– nombre de tâches (N) : 10, 15, 20, 25 ou 30

Pour chaque combinaison de paramètres nous avons généré aléatoirement huit instances ce qui fait un total de 160 instances.

6.2. Résolution exacte

Pour chaque instance, nous avons tenté de résoudre le modèle mathématique du problème à l'aide du logiciel IBM ILOG CPLEX Optimization Studio version 12.4. Nous avons borné le temps de résolution à 30 minutes. Si le solveur ne retourne pas la solution optimale dans le temps imparti, l'écart relatif (ER) entre la borne inférieure (BI) et la borne supérieure (BS) est calculé :

$$ER = \frac{BS - BI}{BI}$$

Le tableau 3 présente, pour les différents nombres de tâches considérés, le pourcentage d'instances pour lesquelles une solution optimale a été trouvée et la moyenne des écarts relatifs ($ERAv$) obtenus pour l'ensemble des instances de la classe. Nous constatons que très peu d'instances sont résolues à l'optimal au-delà de 15 tâches.

Tableau 3. Temps de calcul et écart entre les bornes pour CPLEX

Taille d'instance	% de solutions optimales	*ERAv*
10 tâches	100	0
15 tâches	87.5	2.2
20 tâches	9.4	20.1
25 tâches	12.5	34.91
30 tâches	0	54.09

6.3. Résolution approchée

6.3.1. Paramétrage

Plusieurs paramètres sont à déterminer pour exécuter nos différentes approches. Nous utiliserons une recherche locale (LS), un recuit simulé (SA) et deux recherches locales itérées (ILS) pour résoudre notre problème. Le but est de comparer équitablement ces différentes approches, ainsi nous exécuterons les diverses

approches avec un nombre équivalent d'appels à la fonction évaluation. Le premier paramètre à déterminer est donc le nombre d'appels à la fonction coût total. Étant donné que nos instances sont de petite taille, nous avons choisi de les résoudre avec 100 000 évaluations de solutions.

Pour la probabilité d'appliquer le mouvement *V1* ou *V2* à chaque itération, nous avons effectué de nombreux tests. Les résultats sont reportés dans le tableau 4. $\overline{H(X)}$ correspond à la moyenne des critères obtenus en réalisant une réplication sur toutes les instances. *PV1* correspond à la probabilité d'appliquer le mouvement V1 dans le système de voisinage. Les résultats du tableau 4 montrent que la probabilité de 0.5 pour chaque mouvement est le meilleur choix.

Tableau 4. Expérimentation sur la probabilité d'application des mouvements

PV1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\overline{H(X)}$	56.84	56.19	55.91	55.63	**55.62**	55.69	55.92	56.22	56.75

Pour la résolution avec la recherche locale itérée, nous avons fixé le nombre d'itérations maximal pour chaque recherche locale, après expérimentations, à 1000. Pour les deux recherches locales itérées, il nous reste à définir les critères d'acceptation des solutions obtenues par les systèmes de voisinage entre chaque recherche locale et la perturbation des solutions. Pour la perturbation nous appliquons le même voisinage, et nous acceptons la solution obtenue quelle que soit sa qualité. Nous utiliserons deux critères d'acceptation différents, celui de la recherche locale (ILS|LS : on prend le meilleur) et celui du recuit simulé (ILS|SA : on prend le meilleur selon une probabilité décroissante dans le temps). Pour le recuit simulé, nous fixerons une température initiale telle que la plupart des transitions seront acceptées et la température finale sera fixée de façon empirique. Dix réplications sont effectuées sur chaque instance.

6.3.2. Comparaison des méthodes approchées

Les résultats obtenus par les différentes méthodes proposées sont reportés dans le tableau 5. Pour une méthode de résolution donnée, nous calculons pour chaque instance *i* et pour chaque réplication *j*, l'écart relatif $RG_{i,j}$ entre la solution trouvée $X_{i,j}$ et la meilleure solution connue BKS_i (*BKS = Best Known Solution*). Cette meilleure solution est soit la solution optimale retournée par CPLEX, soit la borne supérieure calculée par CPLEX, soit la solution obtenue par une réplication d'une des métaheuristiques.

$$RG_{i,j} = \frac{H(X_{i,j}) - H(BKS_i)}{H(BKS_i)}$$

Tableau 5. Résultats obtenus pour 100K itérations par réplications

		Méta-Heuristique	LS		ILS|BW		ILS|SA		SA		CPLEX
N	S	Ressources	$RGAv$	$SDAv$	$RGAv$	$SDAv$	$RGAv$	$SDAv$	$RGAv$	$SDAv$	$RGAv$
10	2	10	4.17	6.36	0.84	0.18	0.57	0.00	0.57	0.00	**0.00**
	3	10	11.49	7.09	2.96	0.67	2.63	0.00	2.80	0.33	**0.00**
	2	20	3.91	3.16	0.19	0.40	**0.00**	0.00	0.19	0.29	**0.00**
	3	20	4.45	2.68	2.38	0.00	2.38	0.00	2.38	0.00	**0.00**
15	2	10	4.02	2.69	1.11	0.48	0.80	0.40	1.18	0.42	**0.20**
	3	10	14.74	7.51	4.71	2.49	3.21	1.57	4.05	2.03	**0.39**
	2	20	8.07	3.67	1.02	0.55	0.91	0.48	0.99	0.74	**0.37**
	3	20	9.74	4.22	3.74	0.96	3.72	0.86	4.59	0.93	**0.00**
20	2	10	1.84	1.31	0.83	0.60	0.81	0.47	0.67	0.29	**0.18**
	3	10	11.67	5.85	3.68	1.48	3.86	1.32	4.76	1.80	**3.40**
	2	20	6.23	4.59	**2.09**	1.08	2.10	0.92	2.37	1.18	9.25
	3	20	11.27	6.79	3.53	2.61	**2.91**	1.16	3.81	2.09	16.47
25	2	10	4.12	2.81	**1.21**	1.38	1.32	0.68	1.94	1.00	15.58
	3	10	10.58	3.99	4.24	2.19	**3.47**	1.68	4.84	.36	21.83
	2	20	4.58	2.61	1.91	1.30	**1.37**	0.62	2.28	1.35	11.67
	3	20	8.97	4.15	**3.16**	2.20	3.30	1.48	4.25	2.17	28.60
30	2	10	3.65	3.58	1.11	1.41	**0.95**	0.94	1.41	1.11	36.65
	3	10	4.16	1.95	**1.48**	1.06	1.78	1.07	2.36	1.04	40.94
	2	20	6.84	3.12	3.08	2.10	**2.55**	1.83	3.19	2.07	27.70
	3	20	14.42	7.15	5.62	3.61	**4.26**	2.09	6.11	3.26	117.45
		Moyenne	7.45	4.26	2.44	1.34	**2.16**	0.89	2.74	1.22	16.53

Nous calculons ensuite la moyenne RG_i et l'écart type SD_i des écarts relatifs sur l'ensemble des réplications pour une instance donnée.

$$RG_i = \frac{\sum_{j=1}^{10} RG_{i,j}}{10}$$

$$SD_i = \sqrt{\frac{\sum_{j=1}^{10} RG_{i,j}^2}{10} - \left(\frac{\sum_{j=1}^{10} RG_{i,j}}{10} \right)^2}$$

Enfin nous calculons la moyenne de ces deux quantités (*RGAv* et *SDAv*) sur l'ensemble des instances pour une classe d'instances donnée.

Les résultats avec les métaheuristiques montrent que les meilleurs résultats sont obtenus avec la recherche locale itérée. En effet, même si la résolution à l'aide de CPLEX est plus efficace sur les petites instances, cette méthode montre vite ses limites quand le nombre de tâches augmente. Le meilleur test d'acceptation pour la recherche locale itérée est celui de type recuit simulé. Les pires résultats sont obtenus par la recherche locale, ce qui signifie que l'espace de recherche est composée de nombreux minimas locaux dont la recherche locale ne peut s'extraire. Les autres métaheuristiques ont des solutions assez proches de la meilleure. Les temps d'exécution de chaque méthode varient entre 3 et 12 secondes selon la taille des instances.

7. Conclusion

Dans cet article, nous avons présenté un problème de mutualisation de ressources dans le domaine hospitalier. Nous avons fait l'analogie entre ce problème et le problème bien connu du RCPSP. Pour résoudre ce problème, nous avons proposé une extension du problème classique de RCPSP en lui ajoutant un contexte multi-site. Nous avons proposé un modèle linéaire en nombres entiers à variables binaires, pour le résoudre. Nous avons aussi proposé une résolution approchée à l'aide de métaheuristiques. Nous avons comparé les résultats obtenus avec plusieurs métaheuristiques et en avons conclu que la recherche locale itérée avec le critère d'acceptation du recuit simulé donne les meilleurs résultats.

Étant donné que le RCPSP est un problème d'ordonnancement généralisé, notre travail peut aussi s'adapter à tous problèmes d'ordonnancement avec temps de transport dépendant du site où les tâches sont effectuées (problème de placement avec temps de transfert, flow shop avec temps de transport, flow shop hybride avec temps de transport entre les étages, etc.). De plus, notre méthode de résolution approchée est facilement adaptable, dans le but de considérer de nouvelles contraintes. Ces nouvelles contraintes (incompatibilités ressources/ressources,

disponibilité des ressources, etc.) peuvent être prises en compte dans l'affectation et l'ordonnancement des tâches, au niveau de l'algorithme d'ordre strict.

Le travail présenté est une première étude sur le RCPSP multisite. Le travail à réaliser est important. La première tâche consiste à enrichir la bibliothèque d'instances avec des instances de taille plus grande, en se basant notamment sur la bibliothèque d'instances du RCPSP. La deuxième piste de réflexion est d'améliorer les métaheuristiques actuelles en incorporant par exemple dans le voisinage des techniques de guidage. Enfin, pour revenir à notre problématique de la CHT, il est important d'intégrer la prise en compte de contraintes terrain telles que la limitation du déplacement des ressources.

Les résultats présentés dans ce papier avec une version standard des métaheuristiques permettront à la fois de juger la pertinence des améliorations envisagées, et d'estimer l'influence de la flexibilité des ressources mobiles sur les solutions pressenties.

Bibliographie

Bilge Ü., Ulusoy G. (1995). A time window approach to simultaneous scheduling of machines and material handling system in an fms. *Operations Research*, vol. 43, nº 6, p. 1058-1070.

Blazewicz J., Lenstra J., Kan A. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, vol. 5, nº 1, p. 11-24.

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

Chassiakos A., Sakellaropoulos S. (2005). Time-cost optimization of construction projects with generalized activity constraints. *Journal of Construction Engineering and Management*, vol. 131, nº 10, p. 1115-1124.

Correia I., Lourenço L. L., Gama F. Saldanha-da. (2012). Project scheduling with flexible resources: formulation and inequalities. *OR spectrum*, vol. 34, nº 3, p. 635-663.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of Np-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gourgand M., Grangeon N., Klement N. (2014). Activities planning and resource assignment on multi-place hospital system: Exact and approach methods adapted from the bin packing problem. In *7th international conference on health informatics*, p. 117-124. Angers, France.

Hwang J., Chow Y., Anger F., Lee C. (1989). Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, vol. 18, nº 2, p. 244-257.

Johnson S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, vol. 1, nº 1, p. 61-68.

Kan A. R. (2012). Machine scheduling problems: classification, complexity and computations. *Springer Science & Business Media*.

Kirkpatrick S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, vol. 34, nº 5-6, p. 975-986.

Kise H. (1991). On an automated two-machine flowshop scheduling problem with infinite buffer. *J. Oper. Res. Soc. Japan*, vol. 34, p. 354-361.

Klein R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research*, vol. 38, nº 16, p. 3937-3952.

Klein R., Scholl A. (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, vol. 112, nº 2, p. 322-346.

Klein R., Scholl A. (2000). Progress: Optimally solving the generalized resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, vol. 52, nº 3, p. 467-488.

Kolisch R. (1998). Integrated scheduling, assembly area- and part-assignment for large scale, make-to-order assemblies. *International Journal of Production Economics*, vol. 64, p. 127-141.

Langston M. A. (1987). Interstage transportation planning in the deterministic flow-shop environment. *Operations Research*, vol. 35, nº 4, p. 556-564.

Lee C.-Y., Chen Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, vol. 4, nº 1, p. 3-24.

Lombardi M., Milano M. (2009). A precedence constraint posting approach for the rcpsp with time lags and variable durations. In I. Gent (Ed.), *Principles and practice of constraint programming - cp 2009*, vol. 5732, p. 569-583. Springer Berlin Heidelberg.

Lourenço H. R., Martin O. C., Stutzle T. (2003). *Handbook of metaheuristics*. In F. Glover, G. Kochenberger (Eds.), p. 321-353. Kluwers Academic Publishers.

Maggu P., Das G. (1980). On 2× n sequencing problem with transportation times of jobs. *Pure and Applied Mathematika Sciences*, vol. 12, nº 1, p. 219-227.

Maggu P. L., Das G., Kumar R. (1981). On equivalent-job for job-block in 2× n sequencing problem with transportation-times. *Journal of the Operations Research Society of Japan*, vol. 24, nº 2, p. 136-146.

Maggu P. L., Smghal M. L., Mohammad N., Yadav S. K. (1982, sep). On n-job, 2-machine flow-shop scheduling problem with arbitrary time lags and transportation times of jobs. *Journal of the Operations Research Society of Japan*, vol. 25, nº 3, p. 219-227.

Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, vol. 21, nº 6, p. 1087-1092.

Mitten L. (1959). Sequencing n jobs on two machines with arbitrary time lags. *Management science*, vol. 5, nº 3, p. 293-298.

Norre S. (1993). *Problèmes de placement de tâches : méthodes stochastique et évaluation des performances*. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand.

Ogŭz O., Bala H. (1994). A comparative study of computational procedures for the re- source constrained project scheduling problem. *European Journal of Operational Research*, vol. 72, nº 2, 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.

Vanhoucke M. (2004, octobre). *Work continuity constraints in project scheduling*. Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium nº 04/265. Ghent University, Faculty of Economics and Business Administration.

Wu M.-Y., Gajski D. (1988). A programming aid for hypercube architectures. *The Journal of Supercomputing*, vol. 2, nº 3, p. 349-372.

Yu W. (1996). *The two-machine flow shop problem with delays and the one-machine total tardiness problem*. Eindhoven, Technische Universiteit Eindhoven p. 127-136.

Article reçu le 25 juin 2015

Accepté le 25 novembre 2015