7e Conférence Internationale de Modélisation et SIMulation -MOSIM'08- 31 mars, 1er et 2 avril 2008, Paris, France

« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination. ».

RÉSOLUTION D'UN PROBLÈME INDUSTRIEL D'ORDONNANCEMENT DE PROJET AVEC FLÉXIBILITÉ DE RESSOURCES

Marouane ARROUB
IRCCyN, UMR 6597
1 rue de la Noë, BP 92101
44321 Nantes
marouane.arroub@irccyn.ec-nantes.fr

Najib M NAJID
Université de Nantes - IRCCyN
2 avenue Jean Rouxel, BP 539
44475 Carquefou Cedex
najib.najid@univ-nantes.fr

RÉSUMÉ : Cet article a pour objectif d'étudier un problème industriel d'ordonnancement de projet à moyens limités multi-mode. Le problème traité est une extension du problème RCPSP (Resource Constrained Project Scheduling Problem) classique. Ce problème que nous appellerons MRCPSP-MSL-SDAL (Multi Mode Resource Constrained Project Scheduling Problem with Multi Skilled Labor and Schedule-Dependent on Assignement of Labors) est un problème d'ordonnancement de projet à moyens limités multi-mode avec contraintes de compétences et de déplacements. De plus, le problème étudié est soumis à des contraintes spécifiques liées à la culture organisationnelle de l'entreprise. Dans un premier temps, nous caractériserons le problème, puis nous présenterons des heuristiques pour une résolution du problème. Enfin, nous présenterons quelques résultats obtenus sur des instances générées par notre propre générateur d'instances.

MOTS-CLÉS : MRCPSP, poly-compétence, déplacement des opérateurs, heuristiques

1. INTRODUCTION

Le problème d'ordonnancement de projets sous contraintes de ressources (RCPSP) est un problème classique en ordonnancement qui a fait l'objet de nombreux travaux. Ceci est dû au fait qu'il est très présent dans les problématiques industrielles. Un problème RCPSP est formulé par la donnée d'un ensemble d'activités et d'un ensemble de ressources renouvelables. Chaque ressource est disponible en une quantité constante et chaque activité a une durée opératoire bien définie et nécessite pour sa réalisation une quantité constante de chaque ressource. On distingue dans un RCPSP des contraintes temporelles qui sont définies par des relations de précédence et des contraintes de ressources qui exigent qu'à chaque instant et pour chaque ressource la demande totale en ressources ne dépasse pas la quantité de ressources disponibles. La réalisation d'un ordonnancement consiste en l'exécution, par des ressources, d'un ensemble d'activités ou tâches de durées données et liées entre elles par des contraintes de précédence. Dans les problèmes RCPSP, on modélise couramment le projet par un graphe valué, orienté et sans circuit G, le graphe potentiel-tâches, formé du graphe associé à la relation formée par les contraintes de précédence, auquel sont ajoutées deux tâches fictives de durée nulle, représentant respectivement, le début et la fin du projet. De nombreuses extensions du RCPSP ont été développées dans la littérature. Le RCPSP Multi-Mode est l'une de ces extensions. Dans le MRCPSP (Multi-mode Resource Constrained project Scheduling Problem), une activité possède plusieurs modes d'exécution, un mode étant défini par une durée d'exécution et une quantité pour chacune des ressources. Il s'agit alors de choisir un mode pour chaque activité et de trouver une solution respectant les modes choisis.

Les problèmes d'ordonnancement de projet et leurs différentes extensions sont largement traités dans la littérature et ont donné lieu à plusieurs états de l'art (Bibo Y et al. 2001.) (Herroelen et al. 2001). Ils sont connus comme des problèmes d'optimisation NP-difficile au sens fort (Blazewicz et al. 1983). Le regain d'intérêt par les chercheurs à ces problèmes s'est traduit par l'étude de nouvelles classes ou extensions de problèmes et par l'étude de nouvelles méthodes de résolution. Ainsi, la considération des ressources humaines dans les projets et l'intégration de la notion de compétence ont donné naissance à de nouvelles classes de problèmes tels que le problème MRCPSP-MSL (pour MRCPSP with multi skill labors) (Kadrou et al. 2006) ou le problème de gestion de projet avec multi compétence (Bellenguez 2006). La gestion des ressources humaines peut engendrer des contraintes spécifiques telles que les contraintes légales (amplitude maximale d'une journée de travail, repos compensateur…) (Drezet 2005). La gestion des ressources humaines et les impératifs industriels, nous

ont amené à considérer le problème MRCPSP-MSL-SDAL.

Cet article s'articule autour de 5 paragraphes. Dans le deuxième paragraphe, nous présenterons le problème étudié. Nous présenterons des heuristiques pour la résolution du problème dans le troisième paragraphe puis nous présenterons quelques résultats expérimentaux dans le quatrième paragraphe avant de conclure.

2. PRÉSENTATION DU PROBLÈME

Dans notre étude, nous considérons les ressources renouvelables suivantes :

- L'ensemble Z des zones de travail : Il est constitué des postes de travail rattachés au projet. Une zone de travail z est une ressource renouvelable et cumulative. Elle est caractérisée par sa capacité maximale d'accueil C_z. Pour gérer les déplacements des opérateurs, les zones « voisines » sont regroupées dans des groupements de zones.
- L'ensemble des opérateurs O : Un opérateur est une ressource renouvelable et disjonctive. Chaque opérateur o_r possède un ensemble de compétences Sk_r et appartient à une équipe. Il sera disponible pendant un ensemble de quarts Sh_r. Par exemple dans une organisation en 2x8 sans roulement, l'équipe du matin sera présente du lundi au vendredi entre 5h00 et 12h00 et l'équipe du soir sera disponible du lundi au jeudi entre 12h00 et 20h30 donc les opérateurs de l'équipe matinale seront présents pendant les quarts impairs et ceux de l'équipe nocturne pendant les quarts pairs. L'ensemble Sh des quarts de travail permet de faire un double découpage temporel en couvrant la réalisation du projet. Les quarts sont de longueurs variables pour tenir compte du calendrier et ils sont caractérisés par leurs dates de début et dates de fins. Dans cette organisation, une équipe sera disponible dans chaque quart et les quarts ne se chevauchent pas. Pour éviter les longues périodes d'inactivité les opérateurs ne sont dédiés ni à une zone ni à un groupement de zones. Ils peuvent ainsi intervenir sur tous les postes de travail du projet. Cependant, les déplacements des opérateurs doivent être limités et optimisés. C'est pour cela que nous introduisons la contrainte de groupe pour gérer les déplacements entre zones du même groupement de zones et entre groupements de zones différents.
- Chaque tâche t_i est caractérisée par la compétence requise sk_i, l'ensemble des zones Z_i, les prédécesseurs P_i, la charge de travail Ch_i et l'ensemble des modes M_i. Un mode m est associé aux quantités $no_{i,m}$ et $cons_{i,m,z}$ qui décrivent respectivement le nombre d'opérateurs maîtrisant sk_i pour exécuter t_i en mode m et le nombre de places consommées dans chacune des zones z de Z_i en mode m. Une Tâche ne peut démarrer que si toutes les ressources nécessaires associées au mode choisi pour son exécution sont disponibles et sont affectées à cette tâche. Dans chaque quart, une fois qu'un mode est choisi pour une tâche, ce dernier sera conservé le long du quart. Les tâches qui ne sont pas achevées à la fin d'un quart seront préemptées et reprises dans le(s) quart(s) suivant(s) tant que la charge restante n'est pas nulle. Dans un quart s, si une tâche t_i est lancée en mode $m \in M_i$, alors ce mode permet de déterminer la durée opératoire $p_{i,m,s}$, l'affectation des opérateurs et de calculer le nombre de places consommées dans chacune des zones de Z_i.

Notre objectif est de minimiser la date de fin de projet ou C_{max} sous les contraintes suivantes :

- La contrainte de précédence entre les tâches : Une tâche ne peut commencer que si tous ces prédécesseurs sont terminés.
- La contrainte de compétence : Un opérateur n'exécute que les tâches pour lesquelles il est habilité. Les opérateurs sont poly compétents (i.e. maîtrisant plusieurs compétences) mais leurs niveaux de (poly) compétences sont différents.
- La contrainte de saturation de zone : Une zone peut accueillir des tâches qui s'exécutent en parallèles dans la limite de sa capacité maximale d'accueil.
- La contrainte de quart : Une tâche est dite terminée si la charge restante est nulle. Les tâches qui sont lancées et qui ne sont pas terminées au changement de quart seront préemptées et reprises dans le(s) quart(s) suivant(s) avec les ressources ad hoc. De plus, un mode choisi dans un quart est conservé le long de ce quart.
- La contrainte modale : Pour les tâches multi-modes ($|M_i| > 1$), l'ensemble des modes dans le quart s sera réduit à l'ensemble $M_{i,s} \subset M_i$ pour éviter que les durées opératoires $p_{i,m,s}$ ne passent sous un certain seuil S_{FT} : $M_{i,s} = \{m \in M_i | p_{i,m,s} \geq S_{FT}\} \cup \min_{m \in M_i} m$. Cette contrainte se justifie par le fait qu'un mode important correspond à une faible durée opératoire et à un engagement important des ressources, ce qui peut poser des problèmes organisationnels sur une courte durée.
- La contrainte de groupe : Pour limiter les déplacements des opérateurs, des pénalités sont introduites. Les pénalités peuvent s'assimiler soit à une distance soit à un temps de transit entre zones du même groupement ou entre groupements différents. Le temps de transit inter groupement est plus important que le temps de transit inter zones. Dans la contrainte de groupe, la distance dépend du mode des tâches, des affectations des

opérateurs à ces tâches et de la localisation du plus lointain des opérateurs affectés avant l'exécution de ces tâches.

A notre connaissance, les seuls travaux qui sont proches de la contrainte de groupe portent sur le MRCPSP-SDST (pour MRCPSP with schedule-dependent setup times) (Mika et al. 2006a et 2006b). Dans le MRCPSP-SDST, le temps de préparation (assimilable à une distance entre tâches) dépend de l'affectation des ressources. Tandis que dans notre problème, c'est plus générale puisque l'opérateur peut travailler sur plusieurs zones et donc se déplacer d'un ensemble de zones vers un autre ensemble de zones et que l'ordonnancement dépend de la flexibilité des ressources, du choix du mode, de l'affectation qui est associé à ce mode et des déplacements des opérateurs.

3. RÉSOLUTION DU PROBLÈME

Nous avons développé deux heuristiques que nous appellerons H1 et H2. Dans la suite, nous détaillerons ces heuristiques.

3.1. Heuristique constructive H1

Pour la résolution de notre problème, nous avons développé une heuristique avec schéma de construction ou SGS (Shedule Generation Schemes). On distingue deux types de construction avec la méthode SGS :

- Schéma de construction en série: Pour laquelle on ordonnance une tâche à une date donnée avant de passer à la tâche suivante
- Schéma de construction en parallèle : Pour laquelle on ordonnance le maximum de tâches possibles à une date donnée avant de passer à la date suivante.

L'approche de cette heuristique nommée H1 consiste en un ordonnancement « forward » avec possibilité de report des tâches à l'aide d'une heuristique SGS parallèle avec une exploration en largeur à chaque instant de décision. La solution est construite par étape successive en ordonnançant à chaque instant de décision une combinaison de tâches. Pour délimiter l'espace de recherche, nous nous restreignons à l'exploration des combinaisons dites non dominées (i.e. une combinaison de {tâche éligible ; mode ; ressources affectées} qui optimise localement l'affectation des ressources). Le choix de la combinaison non dominée à ordonnancer résulte de l'application de règles de priorité. Ces règles de priorité permettent le classement des tâches et des ressources affectées dans la combinaison non dominée ainsi que le choix des modes des tâches. Nous avons testé les règles classiques telles que Random pour un classement aléatoire, SPT pour un classement suivant la durée opératoire la plus courte, LPT pour le classement suivant la durée opératoire la plus longue, MWKR modifiée pour le classement suivant la charge de la tâche augmentée de la charge de tous ces successeurs immédiats,…..
Nous avons également testé d'autres règles de priorité que nous avons développées.

3.1.1. Pseudo code de H1

Tant qu'il reste des tâches non ordonnancées
Mettre à jour (MAJ) les Tâches Terminées s'il y en a.
MAJ les Tâches Sans Prédécesseurs ou celles dont les prédécesseurs sont ordonnancés.
Calculer les dates de début au plus tôt et les marges.
MAJ les Tâches Eligibles et les classer suivant le critère de classement des tâches
Faire un pré-classement des opérateurs éligibles suivant le critère de classement des ressources
Génération des Combinaisons Non Dominées et évaluation du critère de classement des combinaisons
MAJ les Tâches En Cours en ordonnançant la combinaison non dominée qui minimise le critère de classement des combinaisons.
Avancer à l'instant de décision suivant
Fin tant que

3.1.2. Règles de classement des tâches

Plusieurs critères sont développés tels que :

$\forall i \in I_{El}$

$$CT_i^1 = EXP(pm_{i,s} - \frac{Mar_i}{ChR_{i,s}}) \quad (1)$$

$$CT_i^2 = EXP(max(1, Nb_{suc}^2) + pm_{i,s} - \frac{Mar_i}{ChR_{i,s}}) \quad (2)$$

$$CT_i^3 = CT_i^2 \times EXP(\max(1, Nb_{pre}^2)) \quad (3)$$

Avec EXP la fonction exponentielle, I_{El} l'ensemble des tâches éligibles, pm_{is} le pseudo mode de la tâche i dans le quart *s*, i.e. le produit des modes le plus et le moins important divisé par le carré du nombre de ressources capables d'exécuter la tâche, mar_i la marge libre de la tâche i, ChR_{is} la charge restante de la tâche i dans le quart s, Nb_{suc} le nombre de successeur de la tâche i et Nb_{pre} le nombre de préemption de la tâche i. Le pseudo mode d'une tâche est une quantité positive. Un pseudo mode faible signifie que la flexibilité sur les ressources est importante alors qu'un pseudo mode plus important signifie que l'ensemble des modes est réduit ou que la flexibilité sur les ressources est moins importante..

Dans ces différents critères exposés, nous cherchons un compromis entre le pseudo mode, la marge, la charge restante, le nombre de successeurs ou encore le nombre de préemption d'une tâche. Dans le premier critère noté CT_i^1 par exemple, nous cherchons un compromis entre

le pseudo mode, la marge et la charge restante : Nous favorisons les tâches qui ont le pseudo mode le plus important, la marge la moins importante et la charge restante la plus importante. Le critère CT_i^2 se base sur le critère CT_i^1 en favorisant en plus, les tâches qui ont le plus de successeurs pour avoir plus de choix et plus de combinaisons par la suite. Enfin, le critère CT_i^3 se base sur le critère CT_i^2 en favorisant en plus les tâches les plus préemptées.

3.1.3 *Règles de classement des opérateurs*

Après un tri des opérateurs par zones et par groupements, les règles de classement des opérateurs visent à déterminer la criticité de chaque opérateur en tenant compte à la fois de ces compétences acquises et du besoin des tâches à l'instant de la prise de décision et dans un futur proche. Plusieurs critères sont développés tels que :

$$CR_r^1 = \sum_{\substack{i \in El \\ sk_i \in Sk_r}} \frac{CT_i^1}{\varepsilon} + \sum_{\substack{j \in FEl \\ sk_j \in Sk_r}} \frac{CT_j^1}{\sqrt{es_j - t}}, \forall o \in O_{El} \quad (4)$$

Avec « t » l'instant de décision, ε un nombre très petit, es_jla date de début au plus tôt de la tâche t_j, FEl les futures tâches éligibles (i.e. qui seront éligibles au(x) prochain(s) instant(s) de décision) et O_{El} les opérateurs éligibles. Dans le critère CR_r^1, la criticité d'un opérateur dépend du nombre de sollicitations de l'opérateur par les tâches éligibles et anticipe sur les sollicitations des futures tâches éligibles. La pondération des tâches éligibles est plus importante que celles des futures tâches éligibles. De plus la pondération des futures tâches éligibles tient en compte la position relative des tâches (écart entre l'instant de décision et la date de début au plus tôt).

3.1.4 *Règle de choix de la combinaison*

A chaque instant de décision t du quart s, nous ordonnançons une des combinaisons non dominées dont le critère C vaut : $\mathrm{Min}_{co \in \Gamma_t} C_{co}$

Pour chaque combinaison non dominée « *co* », le critère C est défini par :

$$C_{co} = \max(A_{co}, B_{co}), \forall co \in \Gamma_t \quad (5)$$

$$A_{co} = \max_{i \in I_{co}}(1; p_{i,m_{i,co},s} - \min_{m \in M_{i,t}} p_{i,m,s}), \forall co \in \Gamma_t \quad (6)$$

$$B_{co} = \max_{j \notin I_{co}}(1; \min_{t' \in T_{co}}(t' + \max_{m \in M_{j,t'|t' \in s'}} p_{j,m,s'} - \min_{m' \in M_{j,t'|t' \in s'}} p_{j,m,s'})), \forall co \in \Gamma_t \quad (7)$$

Avec Γ_t L'ensemble des combinaisons non dominées explorées à l'instant t, I_{co} l'ensemble des tâches de la combinaison non dominée *co*, $m_{i,co}$ le mode choisi pour la tâche « i » dans la combinaison *co*, $M_{i,t}$ l'ensemble des modes admissibles pour la tâche i à l'instant t, T_{co} l'ensemble des dates de fins de la combinaison non dominée « *co* » et des dates de fins décalées du temps de transit inter zones et inter groupes.
Le critère C_{co} tient compte de la contribution des tâches de la combinaison non dominée « co » à travers A_{co} et des tâches qui n'appartiennent pas à la combinaison « co » à travers la contribution de B_{co}. A_{co} évalue l'influence du choix du mode pour les tâches de la combinaison non dominée alors que B_{co} évalue l'influence du report des tâches sur la possible augmentation du C_{max}.

L'exploration des combinaisons non dominées se fait avec une recherche arborescente (cf. figure 1). Chaque niveau de l'arbre est associé à une tâche éligible i. les nœuds du même niveau représentent une tâche i suivant un des modes $m_i \in M_{i,t} \cup \{0\}$ ($m_i = 0$ signifie que la tâche $t_i$est reportée et donc $t_i \notin I_{co}$). L'arbre est exploré en profondeur pour trouver les combinaisons non dominées. Toute affectation des ressources à un nœud donné est propagée sur les niveaux supérieurs pour filtrer l'ensemble des modes et des ressources admissibles. Les contraintes de limitation de ressources (en particulier les affectations des ressources) sont également propagées après chaque transition. Un passage en avant (d'un nœud père vers un nœud fils) est effectué s'il existe au moins une combinaison non dominée dans le sous arbre associé à ce nœud. Dans le cas contraire, le nœud frère (s'il y en a) de mode plus important sera visité puis un passage en avant sera effectué si le sous arbre associé au nœud frère contient au moins une combinaison non dominée. Dans le cas contraire, un retour en arrière (passage d'un nœud fils vers un nœud père) sera effectué.

Le critère d'évaluation est calculé de manière dynamique. L'évaluation de la contribution du nœud dans le calcul du critère d'évaluation permet d'élaguer les branches de l'arbre (ou le sous arbre associé au nœud) contenant des solutions non dominées dont le critère dépasse la valeur minimale calculée du critère.

Pour illustrer nos propos, nous considérons un cas simple de génération de combinaisons non dominées à partir de 3 tâches $\{t_1, t_2, t_3\}$ pour lesquelles les ensembles des modes admissibles sont tous égaux à l'ensemble $\{1, 2\}$ et les ensembles des ressources capables sont tous égaux à l'ensemble $\{o_1, o_2, o_3, o_4\}$. Nous supposons en plus que les capacités des zones requises sont assez grandes. Les nœuds du niveau 1, par exemple, représentent la tâche t_1 dans les modes 0, 1 et 2.

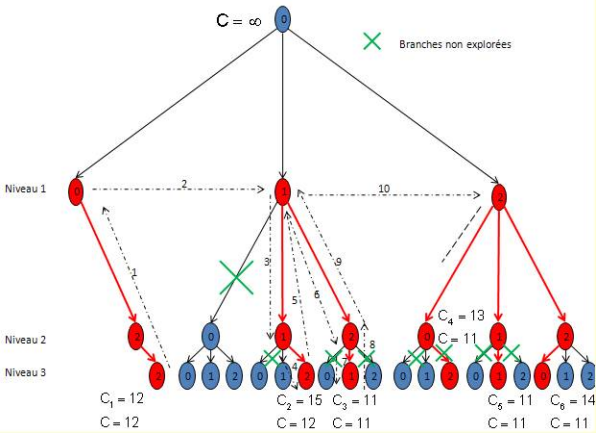

Figure 1: Génération de combinaisons non dominées sur 3 tâches

Initialement, le critère C à minimiser vaut plus l'infini. La première combinaison non dominée est calculée par partir des règles de classement des tâches et des ressources tout en respectant les contraintes du problème (exemple $\{(t_2;\ 2;\ o_1, o_3);\ (t_3;\ 2;\ o_2, o_4)\}$). Son critère C_1 vaut 12. Il permet de mettre à jour la valeur du critère C. Pour générer la combinaison suivante, nous effectuons des retours en arrière jusqu'au niveau 1 (suivant l'arc en pointillé numéro 1) puis nous visitons le nœud frère du même niveau (suivant l'arc 2) puis nous effectuons 2 passages en avant (suivant les arcs 3 et 4). La combinaison non dominée dont le critère C_2 vaut 15 est alors vaudra par exemple $\{(t_1;\ 1;\ o_3);\ (t_2;\ 1;\ o_1);\ (t_3;\ 2;\ o_2, o_4)\}$.
Nous alternons ces procédures pour explorer les autres combinaisons non dominées jusqu'à ce que le critère d'arrêt soit réalisé. Nous ordonnancerons alors une des combinaisons dont la valeur du critère est minimale (par exemple la 5e combinaison non dominée explorée, soit $\{(t_1;\ 2;\ o_{3,}o_4);\ (t_2;\ 1;\ o_1);\ (t_3;\ 1;\ o_2)\}$)

3.2. Heuristique H2

L'idée est de partir d'une solution du problème et d'essayer de l'améliorer en faisant de(s) coupe(s), des mouvements dans un voisinage puis la construction de la partie restante pour obtenir une nouvelle solution. Un mouvement ou « move » (cf. figures 2 et 3)est toute modification dans un voisinage donné du type changement de mode et/ou changement d'affectation des ressources en gardant le même mode et/ou changement de date(s) de début sur un ensemble de tâches d'une solution donnée. En partant d'une solution « sol » du problème, nous effectuons une coupe à un instant de décision t_c, puis nous faisons un mouvement sur les tâches du voisinage V_{t_c}, avant de réordonnancer la solution restante si cela est opportun, en nous appuyant sur le mode de construction de l'heuristique H1. Vu les performances de H1, nous estimons qu'il est préférable de s'appuyer sur H1 pour reconstruire la solution restante et non sur des procédures du type forward et/ou backward. Nous obtenons ainsi une solution « sol' » qui partage avec la solution « sol » les décisions prises en amont du voisinage V_{t_c}. La solution « sol' » améliore « sol », si $C_{max}(sol') \leq C_{max}(sol)$ avec un nombre ou un temps global de déplacements des opérateurs moindre. Cette procédure est répétée itérativement tant que le test d'arrêt est faux.

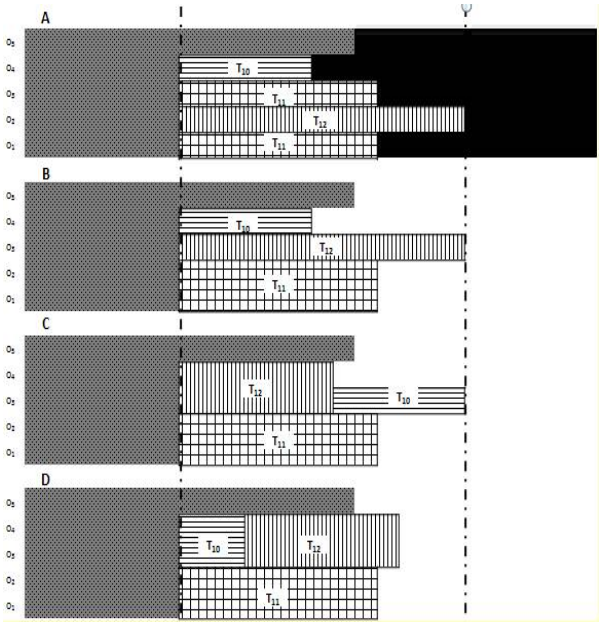

Figure 2: Mouvements sur un voisinage

Les figures 2 illustre quelques mouvements sur le voisinage composé des tâches $\{t_{10}, t_{11}, t_{12}\}$. Dans la solution initiale (cf. figure 2.A) t_{10} est ordonnancée en mode 1 avec l'opérateur o_4, t_{11} en mode 2 avec les opérateurs o_1 et o_3 et t_{12} en mode 1 avec l'opérateur o_2. La figure 2.B illustre un changement d'affectation en conservant les modes des tâches du voisinage. En effet, t_{11} est désormais ordonnancée en mode 2 avec les opérateurs o_1 et o_2 et t_{12} en mode 1 avec l'opérateur o_3. Le passage de la figure 2.B à la figure 2.C se fait par une augmentation du mode de la tâche t_{12} et un changement d'affectation (affectation de l'opérateur o_3 au lieu de l'opérateur o_4 dans la figure 2.B) et de date de début pour la tâche t_{10}. Enfin le passage de la figure 2.C à la figure 2.D s'opère grâce à un mouvement du type changement de date de début pour la tâche t_{12} et une augmentation du mode accompagnée du changement de la date de début pour la tâche t_{10}. Nous constatons au passage que tout mouvement peut être décrit comme une combinaison des mouvements élémentaires donnés dans la définition d'un mouvement.

3.2.1. Détermination de l'instant de coupe

Pour déterminer t_c, nous nous basons sur des indicateurs et sur des méthodes de guidage. Les indicateurs portent sur les instants de décision et permettent de suivre l'utilisation des ressources, le nombre de tâches ordonnancées, les nombres de combinaisons non dominées explorées… Par la suite, nous guidons par exemple vers les instants de décisions où l'utilisation des ressources est la plus faible dans l'espoir d'améliorer cette utilisation des ressources ou vers les instants où le nombre de combinaison non dominées explorées est le plus important dans l'espoir d'explorer un front Pareto optimal.

3.2.2. Détermination du voisinage V_{t_c}

En fonction de la méthode de guidage choisie et d'un nombre maximal de tâches ou de fractions de tâches, nous calculons les bornes $lb(V_{t_c})$ et $ub(V_{t_c})$.

Le voisinage sera alors constitué de toutes les tâches ou fractions de tâches qui sont ordonnancées entre ces deux bornes. Par exemple, si le guidage se fait sur les instants de décisions où l'utilisation des ressources est la plus faible, alors les bornes vérifient $lb(V_{t_c}) \leq t_c \leq ub(V_{t_c})$ car le mode de construction de chaque solution est initialement basé sur la maximisation de l'utilisation des ressources. Par conséquent, la chute du taux d'utilisation des ressources est due aux contraintes et aux décisions prises en amont.

3.2.3. Pseudo code de H2

Initialiser les données
Construire une solution initiale sol_0
Initialiser la liste des solutions sauvegardées à explorer avec sol_0
Initialiser BC_{max} et WC_{max}
Tant que le test d'arrêt est faux
  Charger « sol » par une méthode de sélection dans la liste des solutions sauvegardées à explorer
  Sélectionner une méthode de coupe
  Déterminer t_c
  Calculer le voisinage V_{t_c}
  Couper la solution chargée « sol »
  Effectuer des mouvements dans V_{t_c}
  Reconstruire la partie restante de « sol' » si le C_{max}(sol') estimé puis calculé reste inférieur à WC_{max} . Sinon rejeter la solution
  Si la Solution n'est pas rejetée
    MAJ BC_{max} et WC_{max}
    MAJ la liste des solutions sauvegardées à explorer
  Fin si
Fin tant que

Avec BC_{max} et WC_{max} le meilleur C_{max} trouvé et le pire C_{max} autorisé

4. RÉSULTATS EXPÉRIMENTAUX

Compte tenu de la spécificité du problème et des impératifs industriels, nous nous sommes inspirés des travaux sur le générateur PROGEN pour créer notre propre générateur d'instances. Le générateur utilise des indicateurs nouveaux ou existants tels que la complexité du graphe, la largeur maximale du graphe, le mode de fonctionnement (normal, 2x8 ou 3x8), le ratio des tâches en mode exactes, le ratio des tâches mono zones, le nombre maximum des zones requises pour les tâches multi zones, le taux de flexibilité sur les ressources, le nombre d'équipe, les longueurs minimales et maximales des quarts, le ratio des zones par groupements de zones…..

4.1. Calcul d'une borne inférieure

Nous calculons plusieurs bornes inférieures simples, puis nous considérons la borne inférieure:

$$LB = \max_a LB_a \quad (8)$$

La première borne inférieure développée correspond à un taux d'utilisation des ressources humaines à 100 % en relaxant toutes les autres contraintes :

$$LB_1 = \frac{\sum_{i \in I} Ch_i}{E(Eq)} \quad (9)$$

Avec $E(Eq) = \frac{|O|}{Nb_e}$ l'espérance de l'effectif des équipes utilisées dans le projet (i.e. le ratio entre le nombre d'opérateurs et le nombre d'équipes utilisées). Lorsque le nombre de tâches est important et que les contraintes de capacité et de compétences sont moins contraignantes que les autres contraintes, cette borne correspond à la loi empirique dite des grands volumes. Cette borne nous permet également de calculer le taux d'utilisation des ressources à 80 % utilisé par certains industriels : $T_{80\%} = \frac{LB_1}{0.8}$. Ce taux est empirique et correspond à une demande de l'industriel. Dans l'industrie nous trouvons d'autres taux empiriques tels que $T_{70\%}$ qui correspondent à un objectif visé d'utilisation des ressources en respectant les contraintes.

La deuxième borne inférieure correspond à l'utilisation des zones à 100 % de leurs capacités en relaxant toutes les autres contraintes. Si toutes les tâches du projet sont mono zones, cette borne peut être formulée :

$$LB_2 = \max_{z \in Z} \sum_{i \in I | z = z_i} \frac{Ch_i}{\min\left(C_z, \max_{m \in M_i} m\right)} \quad (10)$$

Dans le cas général,

$$LB_2 = \max_{z \in Z} \sum_{i \in I | z \in Z_i} \frac{\alpha_{i,z} \times Ch_i}{\min\left(C_z, \max_{m \in M_i} m\right)} \quad (11)$$

Avec $\alpha_{i,z}$ une pondération du type $\frac{\sum_{m\in M_i} C_{i,m,z}}{\sum_{z\in Z_i, m\in M_i} C_{i,m,z}}$, $\forall i \in I$, $\forall z \in Z_i$

La troisième borne inférieure tient compte des compétences disponibles et des compétences requises. Elle peut être formulée comme suit :

$$LB_3 = \max_{sk\in SK} \sum_{i\in I|sk=sk_i} \frac{Ch_i}{\min\left(\max_{m\in M_i} m ; \max_{e\in E} \sum_{o_r\in Eq_e} \delta_{r,sk}\right)} \quad (12)$$

Avec SK l'ensemble des compétences requises par le projet, E l'ensemble des indices des équipes utilisées, Eq_e l'équipe indice « e » et

$$\delta_{r,sk} = \begin{cases} 1, si\ o_r\ maîtrise\ la\ compétence\ sk \\ 0, sinon \end{cases}$$

4.2. Les tests

Les tests sont effectués sur des séries de 100 projets. Pour chaque série le nombre de tâche, d'opérateurs, de zones et de groupements sont fixés et leurs caractéristiques sont choisies dans des intervalles.
Les temps sont exprimés en dixième d'heure ou dh (1dh = 6min.). Pour chaque projet de chaque série, la charge de chaque tâche est comprise entre 40 et 80 dh, les modes entre 1 et 3 et la flexibilité sur les ressources entre 1 et 3 (i.e. le ratio entre le nombre de ressources capables dans chaque équipe et le mode minimum de la tâche est compris entre 1 et 3). La capacité maximale de chaque zone, est comprise entre 1 et 3. Le temps de transit inter zone est fixé à 2 dh et le temps de transit inter groupe est fixé à 3 dh. Le temps d'exécution est le minimum entre la durée d'exécution maximale fixée à 1dh et la durée au terme de laquelle le nombre d'itérations sans améliorations atteint les 100 itérations. Quant, à la longueur du quart, elle est fixée par défaut à 70 dh. Le fonctionnement choisi est le 2x8 sans roulements (i.e. avec 2 équipes : une équipe pour le quart du matin et une autre équipe pour le quart du soir). La différence maximale des effectifs des équipes est fixée à 2.

Pour chaque série, nous évaluons le C_{max} moyen (resp. minimum et maximum), le nombre moyen de déplacement par opérateur DM (resp. minimum et maximum), le taux d'utilisation moyen (resp. minimum et maximum) des ressources humaines TUR, la déviation moyenne Dev_LB (resp. minimum et maximum) du C_{max} par rapport à la borne inférieure *LB*, la déviation moyenne Dev_T (resp. minimum et maximum) par rapport à $T_{80\%}$ et enfin le gain moyen (resp. minimum et maximum) sur le C_{max} de H2 par rapport à H1.

Dans le premier tableau, nous comparons les performances de H1 et H2 sur les séries *I60, I120*, *I240* et *I360* qui sont très contraintes.

|I|	60		120		240		360	
|R|	6		8		8		8	
|Z|	6		6		6		6	
|G|	2		2		2		2	
	H1	H2	H1	H2	H1	H2	H1	H2
Cmax	796	769	1373	1349	2568	2560	3847	3831
min(Cmax)	637	630	1190	1178	2360	2355	3601	3636
max(Cmax)	954	935	1535	1521	2795	2740	4051	4040
DM	2	3	6	6	16	16	24	24
min(DM)	1	2	5	5	14	14	22	22
max(DM)	4	4	7	7	18	18	27	27
TUR	56%	57%	64%	65%	69%	69%	69%	69%
min(TUR)	48%	50%	59%	60%	66%	67%	65%	66%
max(TUR)	65%	66%	69%	70%	73%	75%	78%	72%
Dev_LB	35%	31%	18%	16%	39%	39%	39%	38%
Dev_Lbmin	13%	11%	4%	4%	26%	24%	20%	30%
Dev_Lbmax	52%	48%	48%	27%	45%	46%	45%	44%
Dev_T	43%	38%	55%	53%	16%	15%	16%	15%
min(Dev_T)	22%	21%	44%	41%	9%	6%	2%	10%
max(Dev_T)	65%	97%	67%	65%	20%	20%	22%	20%
Gain	4%		2%		1%		0,40%	
min(Gain)	-9%		-1%		-2%		-9%	
max(Gain)	16%		10%		3%		4%	

Tableau 1.

Nous constatons d'une part que globalement H2 améliore les performances H1 même si nous n'utilisons pas la meilleure solution de H1 pour initialiser H2 (H2 améliore H1 en absolue si nous utilisons la meilleure solution de H1 pour l'initialiser). Sur certaines instances, H1 donne un meilleur résultat que H2. Ceci peut s'expliquer en partie par le temps de résolution fixé de la même manière pour les deux méthodes. En effet, H2 a besoin de plus de temps pour visiter plus de voisinages et effectuer plus de mouvements. Le temps de résolution explique également la diminution du gain moyen de H2 par rapport à H1 lorsque la taille des instances augmente.

Nous constatons également la limite des bornes inférieures calculées lorsque les instances sont très contraintes. Nous remarquons également le faible taux d'utilisation des ressources dans ce cas de figure.

Dans le tableau 2, nous étudions les performances de H2 sur les séries *I60-c*, *I120-c* et *I240-c* qui sont moins contraintes pour suivre le taux d'utilisation des ressources. Pour rendre un problème moins contraint, il suffit de diminuer la complexité du graphe et/ou d'augmenter le nombre de tâches de niveau 1

(décomposition par niveau d'un graphe) et/ou d'augmenter les capacités des zones et/ou d'augmenter le taux de poly compétence des opérateurs (i.e. le ratio entre le nombre de compétences acquises par l'opérateur et le nombre de compétences requises par le projet) pour avoir plus de flexibilité sur les ressources.

|I|	60-c	120-c	240-c
|R|	8	8	8
|Z|	6	6	6
|G|	2	2	2
	H2	H2	H2
Cmax	492	990	2016
min(Cmax)	437	859	1811
max(Cmax)	586	1139	2273
DM	4	8	15
min(DM)	3	6	13
max(DM)	5	9	18
TUR	91%	89%	87%
min(TUR)	80%	77%	78%
max(TUR)	97%	97%	95%

Tableau 2.

Nous constatons que le taux d'utilisation des ressources a considérablement augmenté lorsque les contraintes sont faibles.

Dans le tableau 3, nous évaluons l'influence de la longueur des quarts sur les performances de H2 en faisant varier la longueur des quarts de chaque projet entre 40 et 80 dh selon 4 scénarios sur une série *I120-1* avec un fonctionnement en 2x8. Ces 4 scénarios correspondent à 4 calendriers industriels différents pour chaque projet de la série. La différence maximale des effectifs des équipes est fixée à 0.

Nous constatons que si les profils des équipes (en termes de compétences) sont assez proches et que les effectifs des équipes sont identiques, alors l'organisation (qui englobe le choix du fonctionnement et le calendrier industriel) et les performances de H2 sont peu dépendantes. Pour chaque projet, la déviation moyenne du C_{max} pour les 4 scénarios est de l'ordre de 2.4 %. Le but de tableau est de mettre en exergue la dépendance entre le dimensionnement des équipes et leurs profils, l'organisation et les performances. Cette dépendance peut engendrer de nouvelles contraintes : Par exemple, certaines tâches ne peuvent se lancer que dans les quarts pairs (ou bien dans les quarts impairs) dans une organisation en 2x8 à cause des profils des équipes. Dans ce cas de figure, les compétences dans l'équipe 2 permettent de répondre au besoin minimum en compétences de ces tâches alors que les compétences disponibles dans l'équipe 1 ne le permettent pas. D'où l'impossibilité de lancer ces tâches pendant les quarts impairs (i. e. en présence de l'équipe 1)

|I|	I120-1			
|R|	8			
|Z|	6			
|G|	2			
Scénarios	1	2	3	4
	H2	H2	H2	H2
Cmax	1278	1273	1274	1274
min(Cmax)	1124	1112	1125	1126
max(Cmax)	1440	1402	1434	1439
DM	8	8	8	8
min(DM)	7	7	7	6
max(DM)	9	9	9	10
TUR	69%	70%	70%	70%
min(TUR)	64%	65%	64%	65%
max(TUR)	73%	74%	74%	73%
Dev_LB	38%	37%	37%	37%
Dev_Lbmin	24%	24%	21%	21%
Dev_Lbmax	49%	37%	50%	37%
Dev_T	15%	15%	15%	15%
min(Dev_T)	9%	8%	8%	9%
max(Dev_T)	24%	23%	24%	24%
Dev_Cmax	2,40%			
min(Dev_Cmax)	0,40%			
max(Dev_Cmax)	5,30%			

Tableau 3.

5. CONCLUSION

Dans cet article, nous avons présenté un problème original d'ordonnancement de projet en multi mode sous les contraintes de déplacements et de compétences et en tenant compte de l'organisation industrielle autour des projets. Nous avons proposé deux heuristiques pour résoudre ce problème. Ces heuristiques sont testées et comparées avec les méthodes appliquées dans l'entreprise sur des projets allant de 80 à 400 tâches en considérant jusqu'à 28 zones, 8 groupements de zones et 26 opérateurs. Les résultats sont très satisfaisants mais ils ne peuvent pas faire l'objet de communication à cause des clauses de confidentialité. Nous avons également lancé des compagnes de test sur des projets générés par notre propre générateur d'instances et suivant différents scénarios qui s'inspirent des jeux de données industriels (cf. les tablaux 1 et 2). Nous envisageons de développer des bornes inférieures plus élaborées et de tester d'autres heuristiques hybrides.

6. RÉFÉRENCES

Demeulemeester E. and al., 1997, The discrete time/resource trade-off problem in project networks: A branch-and-bound approach, Research

report 9717, Department of Applied Economics, K.U. Leuven.

Blazewicz J. and al., 1983, Scheduling projects to resource constraints: Classification and complexity, Discrete Applied Mathematics.

Herroelen W. and al., 2001, Resource-constrained project scheduling: Notation, classification, models and methods, EJOR.

Bibo Y. and al., 2001, Ressource-Constrainted Project Scheduling: Past Work and New Directions, Research report, Departement of industrial and systems engineering, University of Florida.

Bellenguez O., 2006, Méthode de résolution pour un problème de gestion de projet multi-compétence, thèse de doctorat, université de François Rabelais

Kadrou Y. and Najid N.M., 2006, A new heuristic to solve RCPSP with multiple execution modes and Multi-Skilled Labor, IEEE, Computational Engineering in Systems Applications.

Drezet L., 2005, Résolution d'un problème de gestion de projets sous contraintes de ressources humaines : de l'approche prédictive à l'approche réactive, thèse de doctorat, université de Tours

Reyck B. D. and Herroelen, W., 1999, The multi-mode resource-constrained project scheduling problem with generalized precedence relations, EJOR.

Mika M. and al., 2006a, Tabu search for multi-mode resource constrained project scheduling with schedule-dependent setup times, EJOR.

Mika M. and al., 2006b, Modelling Setup Times in Project Scheduling, International Series In Operations Research & Management Science, Springer, Volume 92, chapter 6.