

Résolution du RCPSP avec production et consommation de ressources : modèles PLNE basés sur les événements

Oumar Koné, Christian Artigues, Pierre Lopez, Marcel Mongeau

▶ **To cite this version:**

Oumar Koné, Christian Artigues, Pierre Lopez, Marcel Mongeau. Résolution du RCPSP avec production et consommation de ressources : modèles PLNE basés sur les événements. International Conference of Modeling and Simulation (MOSIM'10), May 2010, Hammamet, Tunisie. 10p. hal-00484259

HAL Id: hal-00484259
https://hal.science/hal-00484259

Submitted on 18 May 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

8e Conférence Internationale de MOdélisation et SIMulation - MOSIM'10 - 10 au 12 mai 2010 - Hammamet - Tunisie
« Évaluation et optimisation des systèmes innovants de production de biens et de services »

RÉSOLUTION DU RCPSP AVEC PRODUCTION ET CONSOMMATION DE RESSOURCES : MODÈLES PLNE BASÉS SUR LES ÉVÉNEMENTS

Oumar KONÉ[1,2,3], Christian ARTIGUES[1,2], Pierre LOPEZ[1,2] & Marcel MONGEAU[3,4]

[1] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
[3] Université de Toulouse ; UPS, INSA, UT1, UTM ; Institut de Mathématiques de Toulouse
[4] CNRS ; Institut de Mathématiques de Toulouse UMR 5219 ; F-31062 Toulouse, France

RÉSUMÉ : *Cet article traite de l'extension du problème d'ordonnancement de projet à moyens limités qui prend en compte la consommation et la production de ressources. Nous proposons deux modèles basés sur l'utilisation de variables indicées par des événements. Puis, nous comparons leurs résultats à ceux de quelques modèles classiques de la littérature.*

MOTS-CLÉS : *Ordonnancement de projet, RCPSP, programmation linéaire en nombres entiers, modèles à événements, formulation on/off.*

INTRODUCTION

Le problème d'ordonnancement de projet à moyens limités (RCPSP : *Resource-Constrained Project Scheduling Problem*), est l'un des problèmes d'ordonnancement cumulatif les plus connus, du fait de l'intérêt que lui ont accordé les chercheurs du domaine de la recherche opérationnelle et de ses nombreuses applications industrielles. Dans cet article nous nous intéressons à l'extension du RCPSP qui, au-delà des ressources renouvelables, que nous considérons dans sa version basique, comporte également des ressources pouvant être *produites et consommées* lors de l'exécution des activités. Cette extension est désignée par le terme *RCPSP avec production et consommation de ressources.*

Il existe dans la littérature des travaux se rapportant à ce type de RCPSP avec production et consommation de ressources. Le problème d'ordonnancement de projet avec contraintes d'inventaire de stock et contraintes temporelles généralisées, incluant à la fois des ressources renouvelables et non renouvelables, a été formalisé par Neumann et Schwindt [19] en 2002. Ces auteurs proposent également un branch-and-bound avec un filtrage par une heuristique de recherche en faisceau, pour la résolution de ce problème. Carlier *et al.* [11] traitent ce type de problème (RCPSP avec production et consommation de ressources) incluant des contraintes de "time-lag" généralisées, en proposant un nouvel algorithme dit de liste. Pour plus de détails sur l'état de l'art des travaux traitant de ce type de problème, nous renvoyons le lecteur aux articles suivants [18, 8, 11]. Celui de Laborie [18] définit les bases du concept du RTN (*"Resource Temporal Network"*), offrant une grande puissance de modélisation et sert de support pour d'autres auteurs tels que Bouley *et al.* [8].

Dans notre étude, après une description détaillée du RCPSP et l'extension que nous traitons dans cet article, nous présentons des extensions des modèles de programmation linéaire en nombres entiers pour le RCPSP basique, dont deux nouvelles formulations, que nous étendrons à la résolution du RCPSP avec production et consommation de ressources. Pour finir, nous présentons quelques résultats expérimentaux.

1 DÉFINITION DU PROBLÈME

Formellement, le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) est un problème d'optimisation combinatoire défini par un 6-uplet (V, p, E, R, B, b), où V est un ensemble d'*activités*, p est un vecteur de *durées d'exécution*, E est un ensemble de *relations de précédences*, R est un ensemble de *ressources*, B est un vecteur de *capacités (disponibilités des ressources)*, et b est une matrice de *demandes (consommations de ressource).*

1.1 Description du RCPSP basique

Soit n le nombre d'activités à ordonnancer, et m le nombre de ressources disponibles. Les activités constituant le projet sont identifiées par un ensemble $\{0, \ldots, n+1\}$. L'activité 0 représente par convention le début de l'ordonnancement (début du projet), et l'activité $n+1$ quant à elle représente symétriquement la fin de l'ordonnancement (fin du projet). Toutes deux sont des activités fictives appelées également *activités jalons*. L'ensemble des activités non-fictives est noté $A = \{1, \ldots, n\}$. Les durées d'exécution sont représentées par un vecteur p de $\mathbb{N}^{n+2}$, où la $i^{ième}$ composante, p_i, est la durée d'exécution de l'activité i, avec les valeurs spéciales $p_0 = p_{n+1} = 0$, caractéristiques des deux activités fictives.

Les *relations de précédences* (représentées par exemple sur un graphe *potentiels-tâches* acyclique) sont données par un ensemble E de paires d'indices d'activités telles que $(i, j) \in E$ signifie que l'activité i précède l'activité j.

On parle de *ressource cumulative* si la ressource peut être utilisée au même moment par plusieurs activités lors de leur exécution. Dans le cas contraire, on parle de *ressource disjonctive*. Une ressource est dite *renouvelable* si elle redevient disponible dans sa quantité initiale, une fois libérée par les activités. Dans sa version basique, le RCPSP met en jeu des ressources renouvelables formalisées par l'ensemble $R = \{1, \ldots, m\}$. Les *disponibilités* des ressources renouvelables sont représentées par un vecteur B de $\mathbb{N}^m$ tel que B_k indique la capacité de la ressource renouvelable k. Les *consommations des activités* pour les ressources renouvelables figurent dans la matrice b, de taille $(n+2) \times m$, dont la composante b_{ik} représente la quantité de ressource renouvelable k utilisée par l'activité i sur toute sa durée d'exécution. On pose $b_{0k} = b_{n+1,k} = 0$, pour tout $k \in R$.

Un *ordonnancement* est un point S de $\mathbb{R}^{n+2}$ tel que sa $i^{ème}$ composante, S_i, représente la date de début de l'activité i. La date de démarrage est $S_0 = 0$. Le *makespan* d'un ordonnancement S est égal à S_{n+1}, la date de début de l'activité de fin de projet. Une solution S est dite *réalisable* si elle est compatible avec les contraintes de précédences

$$S_j - S_i \geq p_i \quad \forall (i, j) \in E, \tag{1}$$

et les contraintes de ressources

$$\sum_{i \in A_t} b_{ik} \leq B_k \quad \forall k \in R, \forall t \in H, \tag{2}$$

où $A_t = \{i \in A \mid S_i \leq t < S_i + p_i\}$ représente l'ensemble des activités non-fictives en cours d'exécution à l'instant t et la non-préemption des activités, $H = \{0, 1, \ldots, T\}$ est *l'horizon d'ordonnancement*, et T la longueur de l'horizon d'ordonnancement (pouvant être considérée comme une borne supérieure pour le makespan.

Le *RCPSP* est le problème visant à trouver un ordonnancement non-préemptif S de makespan minimal S_{n+1} soumis à des contraintes de précédences et des contraintes de ressources.

Le RCPSP est l'un des problèmes d'optimisation combinatoire les plus difficiles à résoudre. La démonstration de sa NP-difficulté au sens fort est donnée par Blazewicz *et al.* [7].

1.2 RCPSP avec consommation et production de ressource

La particularité du RCPSP avec production et consommation de ressources, est qu'en plus d'utiliser les ressources décrites ci-dessus, on a aussi affaire à des ressources particulières, dites non renouvelables. Certains auteurs [19] parlent de ressources cumulatives pour ce type de ressources. Il s'agit de ressources qui sont consommées (ou pas) au début de l'exécution d'une activité, dans une certaine quantité, puis reproduites dans une autre quantité à la fin de l'exécution de cette activité. Plus précisément, une activité i consomme c^-_{ip} unités de la ressource p au début de son exécution et en produit c^+_{ip} unités à la fin de son exécution. On désignera par P l'ensemble de ces ressources et C_p le niveau de stock initial de la ressource $p \in P$. De ces ressources, on peut distinguer les cas de figure suivants :

- Si $c^-_{ip} = c^+_{ip}$, alors p est une ressource renouvelable comme dans le RCPSP basique.

- Dans le cas où l'on a $c^-_{ip} \neq c^+_{ip}$:

  - Si $c^-_{ip} = 0$ et $c^+_{ip} > 0$ (production de ressources). Généralement précédé de la consommation d'une certaine quantité d'une autre ressource, ce cas est rencontré dans les industries de transformation de matières et parfois en production d'énergie (sauf que l'énergie produite de façon continue peut être utilisée avant même l'achèvement de l'activité productrice).

  - Si $c^-_{ip} < c^+_{ip}$, alors on consomme au départ une certaine quantité du produit afin d'en produire davantage.

  - Si $c^-_{ip} > c^+_{ip}$, alors p est une ressource consommable telle qu'un budget, une matière première, etc.

2 FORMULATIONS PLNE À TEMPS DISCRET

Basées sur l'utilisation de variables indexées sur un horizon de temps discrétisé, les formulations à temps discret ont donné lieu à la formulation basique à temps discret (DT) et la formulation désagrégée à temps discret (DDT).

Proposée en 1969 par Pritsker *et al.* [20], la formulation basique à temps discret (DT) est basée sur l'utilisation, pour chaque activité i, de variables d'optimisation binaires x_{it} indexées par le temps et la discrétisation de l'horizon d'ordonnancement. Cette variable vaut 1 si l'activité i démarre son exécution à l'instant t, et 0 sinon.

Le modèle DDT proposée par Christofides en 1987 [12] est très proche de DT. Cependant, ces formulations diffèrent par leur manière de formuler la contrainte de précédence. Typiquement, alors que le modèle DT définit la contrainte pour chaque paire d'activités de l'ensemble des précédences, le modèle DDT propose une contrainte pour chaque paire d'activités de l'ensemble des précédences *et* pour chaque instant de l'horizon d'ordonnancement.

Ces modèles, classés parmi les modèles comprenant un nombre pseudo-polynomial de variables et de contraintes, s'avèrent pénalisant pour la résolution des problèmes à horizon de temps très étendu. Cependant, ils sont connus (principalement DDT) pour fournir d'assez bons résultats et des bornes intéressantes par relaxation continue sur les instances classiques du RCPSP (cf. [4]).

Pour la prise en compte de l'aspect production de ressources dans ces modèles, nous introduisons la variable continue s_{tp} indiquant le niveau de stock de chaque ressource $p \in P$ à chaque instant t. Les contraintes additionnelles sont les suivantes :

$$s_{0p} = C_p - \sum_{i=1}^{n-1} x_{i0} c_{ip}^- \qquad \forall p \in P \tag{3}$$

$$s_{tp} = s_{t-1,p} - \sum_{i=1}^{n-1} x_{it} c_{ip}^- + \sum_{i=1}^{n-1} x_{i,t-p_i} c_{ip}^+ \quad \forall (t,p) \in H \times P, t > 0 \tag{4}$$

$$s_{tp} \geq 0 \qquad \forall (t,p) \in H \times P. \tag{5}$$

La contrainte (4) impose à chaque instant t, que le niveau de stock (s_{tp}) de chaque ressource $p \in P$ soit égal au niveau de stock à l'instant précédent $(t-1)$ de cette ressource, augmenté de la somme des productions ($\sum_{i=1}^{n-1} x_{it} c_{ip}^-$) de cette même ressource par les activités finissant leur exécution à l'instant t, et diminué de la somme des consommations ($\sum_{i=1}^{n-1} x_{i,t-p_i} c_{ip}^+$) des activités débutant leur exécution à ce même instant t.

Quant à la contrainte (3), elle traite du cas particulier de la contrainte (4) à l'instant $t = 0$, c'est-à-dire quand on débute le projet.

En outre, une condition supplémentaire à respecter s'ajoutant au problème, consiste à imposer que les activités débutant leur exécution disposent de suffisamment de ressources pour être exécutées. Pour ce faire, on impose avec la contrainte (5), que la variable s_{tp} reste toujours positive, exprimant ainsi le fait qu'on ne doit à aucun moment être en déficit de ressource.

On note que les variables s_{tp} peuvent être substituées par leur valeur en fonction des x_{it}.

3 MODÈLE À TEMPS CONTINU BASÉE SUR LES FLOTS (FCT)

Inspiré par les travaux de Balas *et al.* [5], et sur la base de la formulation d'Alvarez-Valdès et Tamarit [1], Artigues *et al.* [3] proposent le modèle nommé *formulation à temps continu basée sur les flots (FCT)*. Ce modèle utilise des *variables de flots* pour la gestion des ressources, de variables continues usuelles de date de début S_i pour définir la date de début de chaque activité du projet et des variables séquentielles binaires x_{ij} permettant de définir un ordre partiel entre les différentes activités ($x_{ij} = 1$, si l'activité i précède l'activité j). Les variables continues f_{ijk} définissent la quantité de ressource k "envoyée" (redevenue disponible) par l'activité i (à la fin de son exécution) à l'activité j (au début de son exécution). On considère que les ressources sont toutes disponibles et stockées au niveau de l'activité fictive 0 (sommet *source* dans le graphe de précédence associé au projet). Par la suite, les activités démarrant leur exécution utilisent ces ressources pour être exécutées, et une fois terminées, elles transfèrent ces ressources aux activités qui les succèdent. Cette opération se répète jusqu'à ce que les dernières activités à être exécutées, les transmettent à leur tour à l'activité fictive $n+1$ (sommet *puits*).

Pour ce type de modèle, classé parmi les modèles compacts, Applegate et Cook [2] montrent que cette formulation produit de mauvaises relaxations dues à l'utilisation d'une constante "grand_M" dans les contraintes. Par contre, pour la résolution des problèmes ayant de grands horizons, cette formulation peut être préférable. Cependant, l'utilisation des variables séquentielles pourrait être à l'origine d'un trop grand nombre de symétries, affectant de ce fait les performances du modèle pour la résolution de problèmes fortement cumulatifs.

L'adaptation de ce modèle à la résolution du RCPSP avec production et consommation de ressources impose l'ajout d'une variable continue D_{ijp} représen-

tant la quantité de ressource $p \in P$ que l'activité i (à la fin de exécution) envoie à l'activité j (au début de son exécution). Comme l'exprime la contrainte (6) ci-dessous, cette quantité ne peut en aucun cas dépasser le minimum entre la quantité produite par l'activité i et la quantité consommée par l'activité j, si l'activité i précède l'activité j. Ainsi, conformément au principe de ce modèle, les nouvelles ressources non-renouvelables sont également gérées à l'aide des flots, par les contraintes suivantes :

$$D_{ijp} \geq \min(c^+_{ip}, c^-_{jp}) x_{i,j} \qquad \forall (i,j) \in A^2, p \in P \quad (6)$$

$$\sum_{j \in A \cup \{0\}} D_{ijp} = c^-_{jp} \qquad \forall i \in A \cup \{n+1\}, p \in P \quad (7)$$

$$\sum_{i \in A \cup \{n+1\}} D_{ijp} = c^+_{ip} \qquad \forall i \in A \cup \{0\}, p \in P \quad (8)$$

$$c^+_{0p} = C_k \qquad \forall p \in P \quad (9)$$

$$c^+_{(n+1)p} = +\infty \qquad \forall p \in P \quad (10)$$

La contrainte (7) impose que la somme des quantités de ressources $p \in P$ reçues des activités précédentes par une activité i, soit égale à la quantité de ressources qu'elle consomme pendant son exécution. Symétriquement, la contrainte (8) impose que la somme des quantités de ressources envoyées par l'activité i aux autres activités, soit égale à la quantité qu'elle produit. Cependant, à l'aide de la contrainte (9) (respectivement, contrainte (10)), pour chaque ressource $p \in P$, on fixe la production de cette ressource par l'activité 0 (respectivement l'activité $n+1$) à son niveau de stock initial (respectivement, à l'infini).

4 FORMULATIONS PLNE BASÉE SUR LES ÉVÉNEMENTS

À la différence des modèles à temps discret, Zapata *et al.* [21] proposent une formulation de type PLNE basée sur les événements utilisant trois types de variables binaires pour la résolution du problème d'ordonnancement multi-projet multi-mode à ressources limitées (MRCMPSP). Nous présentons ici deux nouveaux modèles utilisant moins de variables binaires.

4.1 Formulation *start/end* basée sur les événements (SEE)

Notre formulation dite *start/end* utilisant également des variables indexées par les événements, se distingue par l'utilisation de deux types de variables binaires (x_{ie} et y_{ie}). Celles-ci déterminent l'exécution d'une activité dans le temps à l'aide d'événements. Plus précisément, x_{ie} indique si oui ou non l'activité i débute son exécution à l'événement e, et y_{ie} indique si oui ou non l'activité i termine son exécution à l'événement e.

Nous avons également besoin d'une variable continue t_e désignant la date de l'événement e et d'une variable continue b_{ek} indiquant le niveau de la ressource k utilisée entre les événements e et $e+1$. L'utilisation des activités fictives 0 et $n+1$ n'étant pas nécessaire dans ce modèle, on ne considère que l'ensemble A des n activités (non-fictives), et donc $n+1$ événements.

$$\min_{r,t,x,y} t_n \quad (11)$$

$$t_0 = 0 \quad (12)$$

$$t_f \geq t_e + p_i x_{ie} - p_i(1 - y_{if}) \qquad \forall (e,f) \in \mathcal{E}^2, f > e, \forall i \in A \quad (13)$$

$$t_{e+1} \geq t_e \qquad \forall e \in \mathcal{E}, e < n \quad (14)$$

$$\sum_{e \in \mathcal{E}} x_{ie} = 1 \qquad \forall i \in A \quad (15)$$

$$\sum_{e \in \mathcal{E}} y_{ie} = 1 \qquad \forall i \in A \quad (16)$$

$$\sum_{e'=e}^{n} y_{ie'} + \sum_{e'=0}^{e-1} x_{je'} \leq 1 \qquad \forall (i,j) \in E, \forall e \in \mathcal{E} \quad (17)$$

$$r_{0k} = \sum_{i \in A} b_{ik} x_{i0} \qquad \forall k \in R \quad (18)$$

$$r_{ek} = r_{(e-1)k} + \sum_{i \in A} b_{ik} x_{ie} - \sum_{i \in A} b_{ik} y_{ie} \qquad \forall e \in \mathcal{E}, e \geq 1, k \in R \quad (19)$$

$$r_{ek} \leq B_k \qquad \forall e \in \mathcal{E}, k \in R \quad (20)$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \quad (21)$$

$$r_{ek} \geq 0 \qquad \forall e \in \mathcal{E}, k \in R. \quad (22)$$

$$x_{ie} \in \{0,1\}, y_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathcal{E} \quad (23)$$

La fonction-objectif est donnée par (11) et l'équation (12) fixe la date de l'événement 0 placé par convention à 0.

L'inéquation (13) stipule que si l'activité i débute son exécution à l'événement e et la termine à l'événement f, alors le moment qui sépare l'événement f de l'événement e est au moins égal à la durée opératoire de l'activité i.

L'inéquation (14) impose que la date de l'événement e soit antérieure ou égale à la date de l'événement $e+1$.

La contrainte (15) (respectivement (16)) impose qu'une activité ne peut débuter (resp. ne peut se terminer) qu'une et une seule fois, c'est-à-dire n'est associée qu'à un et un seul événement.

L'inéquation (17) représente les contraintes de précédence. Elle impose que :
si $\sum_{e'=0}^{e-1} x_{je'} = 1$ alors $\sum_{e'=e}^{n} y_{ie'} = 0$.
En d'autres termes, pour toute paire d'activités (i,j) appartenant à l'ensemble de précédences E, l'événement de début d'exécution de l'activité j se produira en même temps, ou après, l'événement de fin d'exécution de l'activité i.

Les équations (18) et (19) représentent les contraintes

de conservation des ressources. La contrainte (19) (respectivement (18)) donne la consommation totale r_{ek} (resp. r_{0k}) de la ressource k à l'événement e (resp. à l'événement 0).

Les contraintes (20) représentent les contraintes de ressources. Pour chaque événement e, elles limitent la consommation totale des ressources à la capacité totale disponible de cette ressource, et ce, pour chaque ressource k. Cependant, notons que l'utilisation de variables continues r_{ek} n'est pas indispensable. On peut les remplacer par leur expression, donnée par les équations (18) et (19), dans l'inéquation (20).

Ce modèle, comportant un nombre polynomial de variables et de contraintes, classé parmi les modèles dits *compacts*, se caractérise par la simplicité de modélisation des contraintes de ressources.

Comme les modèles dits *séquentiels* [13, 3] (où une variable binaire x_{ij} décide de l'ordre relatif des activités i et j), le modèle start/end est basé sur la logique qui consiste avant tout à trouver un ordre partiel ou global des activités, puis à appliquer une autre méthode de post-traitement (tel que l'algorithme sériel [15]) pour affiner/déterminer la solution optimale.

Pour la prise en compte de l'aspect production de ressource dans le RCPSP, on introduit de nouvelles variables continues $s_{ep} \geq 0$ indiquant le niveau de stock de chaque ressource $p \in P$ immédiatement après chaque événement e. Ainsi, on définit les contraintes suivantes :

$$s_{0p} = C_p - \sum_{i=1}^{n-1} x_{i0} c_{ip}^- \qquad \forall p \in P \qquad (24)$$

$$s_{ep} = s_{e-1,p} - \sum_{i=1}^{n-1} x_{ie} c_{ip}^- + \sum_{i=1}^{n-1} y_{i,e} c_{ip}^+ \qquad \forall (e,p) \in \mathcal{E} \times P, \quad e > 0 \qquad (25)$$

$$s_{ep} \geq 0 \qquad \forall (e,p) \in \mathcal{E} \times P \qquad (26)$$

Ces contraintes permettent de calculer le niveau de stock de chaque ressource p à chaque événement e, qui est égal à celui à l'événement précédent $e-1$, diminué des quantités consommées à l'événement e par les activités débutant leur exécution, et augmenté de la production de celles finissant en e. Ainsi, l'utilisation des variables binaires x_{ie} (respectivement $y_{i,e}$) indiquant le début (respectivement la fin) des activités, caractéristiques de ce modèle, simplifie l'expression de ces contraintes (24) et (25), dites *contraintes d'état des stocks*. La contrainte (26) permet d'assurer que les activités démarrant leur exécution disposent de suffisamment de ressource pour être exécutées.

4.2 Formulation *on/off* basée sur les événements (OOE)

Cette deuxième formulation basée sur la notion d'événements que nous proposons, utilise un seul type de variables binaires (z_{ie}) en plus des variables continues t_e. La variable binaire z_{ie} permet d'identifier les intervalles pendant lesquels une activité débute ou est en cours d'exécution. Ainsi, z_{ie} vaut 1 si l'activité i débute, ou est en cours d'exécution, immédiatement après l'événement e, sinon elle vaut 0. Quant à la variable t_e, elle détermine toujours la date de l'événement e. Dans ce modèle, on prouve aisément qu'un ordonnancement optimal ne comportera pas plus d'événements de début que d'activités existantes. De ce fait, on pourra restreindre l'ensemble des événements à n éléments : $\mathcal{E} = \{0, 1, \ldots, n-1\}$, sachant que l'utilisation des activités fictives n'est pas indispensable. Le nombre d'événements est ici égal au nombre d'activités.

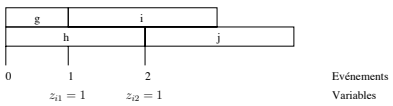


Figure 1: Evénements dans la formulation *on/off*

La figure 1 montre un exemple d'ordonnancement à quatre activités (g, h, i et j) et une seule ressource. On n'a besoin que de trois événements pour sa modélisation, car n'ayant pas d'activité qui démarre à la fin de l'exécution des activités i et j, il n'est pas nécessaire de leur associer des événements de fin. La formulation OOE est donc :

$$\min_{z,t,C_{\max}} C_{\max} \qquad (27)$$

$$C_{\max} \geq t_e + (z_{ie} - z_{i(e-1)}) p_i \qquad \forall e \in \mathcal{E}, \forall i \in A \qquad (28)$$

$$t_0 = 0 \qquad (29)$$

$$t_f \geq t_e + (z_{ie} - z_{i(e-1)} - (z_{if} - z_{i(f-1)}) - 1) p_i \qquad \forall e \in \mathcal{E} \setminus \{0\}, \forall f \in \mathcal{E}, \forall i \in A, f > e \qquad (30)$$

$$t_{e+1} \geq t_e \qquad \forall e \neq n-1 \in \mathcal{E} \qquad (31)$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e(1 - (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathcal{E} \qquad (32)$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n-e)(1 + (z_{ie} - z_{i(e-1)})) \qquad \forall e \neq 0 \in \mathcal{E} \qquad (33)$$

$$\sum_{e \in \mathcal{E}} z_{ie} \geq 1 \qquad \forall i \in A \qquad (34)$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie}) e \qquad \forall e \in \mathcal{E}, \forall (i,j) \in E \qquad (35)$$

$$\sum_{i=0}^{n-1} b_{ik} z_{ie} \leq B_k \qquad \forall e \in \mathcal{E}, \forall k \in R \qquad (36)$$

$$t_e \geq 0 \qquad \forall e \in \mathcal{E} \qquad (37)$$

$$z_{ie} \in \{0,1\} \qquad \forall i \in A, \forall e \in \mathcal{E} \qquad (38)$$

La définition du makespan, $C_{\max}$ est assurée par

la contrainte (28). Elle impose que $C_{\max} \geq t_e + p_i$ si l'activité i est en cours d'exécution juste après l'événement e. En d'autres termes, le makespan doit être supérieur ou égal à la date de fin de toute activité i en cours d'exécution juste après l'événement e.

Tout comme la contrainte (12) du modèle *start/end*, la contrainte (29) fixe la date de l'événement 0 à l'instant 0.

Les contraintes (30) ont deux rôles essentiels. D'une part, elles permettent de lier les variables binaires z_{ie} aux variables continues t_e. D'autre part, elles assurent que si une activité i commence à l'événement e et se termine à l'événement f, alors la date de l'événement f est supérieure à la date de l'événement e, augmentée de la durée opératoire de l'activité i.

Les contraintes (31), tout comme les contraintes (14) imposent que la date de tout événement e soit supérieure ou égale à la date de l'événement précédent, $e-1$.

Les contraintes (32) et (33) représentent les contraintes de non-préemption des activités. Elles assurent la contiguïté des événements durant lesquels une activité reste en cours d'exécution.

Les contraintes (34) imposent que chaque activité soit exécutée au moins une fois pendant le projet.

Les contraintes (35) garantissent les relations de précédence entre les activités.

Les contraintes de ressources (36) imposent que la somme des quantités de chaque ressource consommée par les activités en cours d'exécution à chaque événement e reste inférieure à la capacité disponible de la ressource.

Ce modèle peut être classé parmi les formulations comportant un nombre polynomial de variables et de contraintes. Ainsi, le modèle OOE peut traiter des projets ayant un horizon d'ordonnancement particulièrement étendu. De plus, OOE ne requiert pas l'utilisation d'activités fictives. De façon plus importante, OOE présente une simplicité particulière pour la modélisation des contraintes de ressource renouvelables. Pour finir, OOE est basé sur la même logique de résolution qui consiste d'abord à trouver un ordre partiel ou global des activités, puis à appliquer une méthode de post-traitement pour affiner/déterminer la solution optimale.

La variante OOE_Prec

Cette variante, qui est en fait le modèle OOE avec un "preprocessing", est obtenue en supprimant dans l'ensemble des événements possibles pour chaque activité, les événements pendant lesquels cette activité ne pourra pas être en exécution à cause de ses prédécesseurs (un ou plusieurs de ses prédécesseurs sont en cours d'exécution ou n'ont toujours pas encore été exécutés). Symétriquement, on supprimera aussi pour cette activité, tous les événements pendant lesquels elle ne pourra pas être en cours d'exécution à cause de ses successeurs.

Soit $A(i)$ l'ensemble des prédécesseurs de l'activité i dans le graphe de précédence G, et $D(i)$ l'ensemble de ses successeurs.

Proposition 4.1. *Il existe une solution optimale de OOE telle que pour chaque activité i : $\sum_{e=0}^{|A(i)|} z_{ie} = 0$ et $\sum_{e=n-|D(i)|+1}^{n} z_{ie} = 0$.*

Démonstration **1.** *Lorsqu'on a n événements, il existe toujours une solution optimale telle que ses activités débuteront à des événements distincts. En d'autres termes, pour deux activités distinctes i et j, on aura :*

$$z_{ie} - z_{i,e-1} = 1 \wedge z_{jf} - z_{j,f-1} = 1 \implies e \neq f.$$

Il est important de remarquer que cela n'interdit pas que $t_e = t_f$. De plus, si j est un prédécesseur de i dans G, alors l'événement affecté à j sera strictement antérieur à l'événement affecté à i : $t_e > t_f$. La proposition est une conséquence de ces deux observations.

On obtient OOE_Prec en fixant donc d'emblée les variables suivantes à 0, et on obtient ainsi une formulation correcte pour le RCPSP :

$$z_{ie} = 0, \quad i \in A, e \in \{0, \ldots, |A(i)|\} \cup \{n-|D(i)|+1, \ldots, n\}. \tag{39}$$

Adaptation des modèles OOE et OOE_Prec au RCPSP avec production et consommation de ressources

Pour les modèles OOE et OOE_Prec, la prise en compte de l'aspect production de ressources nécessite non seulement l'ajout de nouvelles contraintes, mais aussi l'introduction de nouvelles variables continues. Nous définissons :

- s_{ep} la variable continue indiquant le niveau de stock de la ressource $p \in P$ à l'événement e ;
- p_{iep} la variable continue indiquant la quantité de ressources $p \in P$ *produite* par l'activité i à l'événement e ;
- u_{iep} la variable continue indiquant la quantité de ressources $p \in P$ consommée (*utilisée*) par l'activité i à l'événement e.

Afin de simplifier l'expression de certaines contraintes, on utilisera temporairement des variables intermédiaires X_{ie} et Y_{ie}. Il ne s'agit en aucun cas de nouvelles variables à ajouter au modèle, mais plutôt de notations permettant de simplifier l'écriture des contraintes. On pourra facilement remplacer directement dans les différentes contraintes ces deux variables intermédiaires par leurs définitions :
$X_{i,e} := z_{ie} - z_{i,e-1}$, ainsi $X_{i,e} \in \{-1, 0, 1\}$;
$Y_{i,e} := z_{ie} - z_{i,e+1}$, de sorte que $Y_{i,e} \in \{-1, 0, 1\}$.
On notera que $X_{i,e} = -Y_{i,e-1}$. De plus $X_{i,e} = 1$ signifie que l'activité i se termine à l'événement e et $Y_{i,e} = 1$ signifie que l'activité i finit son exécution à l'événement e. Lorsque $X_{i,e} = Y_{i,e} = 0$, alors l'activité i ne débute, ni termine son exécution à l'événement e.

Pour le RCPSP avec production et consommation de ressources, les contraintes additionnelles à ajouter aux modèles OOE et OOE_Prec, sont les suivantes :

$$p_{iep} \geq 0 \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{40}$$
$$p_{iep} \geq c^+_{ip} Y_{i,e-1} \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{41}$$
$$p_{iep} \leq c^+_{ip} z_{i,e-1} \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{42}$$
$$p_{iep} \leq c^+_{ip}(1 - z_{i,e}) \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{43}$$
$$u_{iep} \geq 0 \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{44}$$
$$u_{iep} \geq c^-_{ip} X_{ie} \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{45}$$
$$u_{iep} \leq c^-_{ip} z_{ie} \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{46}$$
$$u_{iep} \leq c^-_{ip}(1 - z_{i,e-1}) \quad \forall (e,i,p) \in \mathcal{E} \times A \times P \tag{47}$$
$$s_{ep} = s_{e-1,p} + \sum_{i \in A} p_{iep} - \sum_{i \in A} u_{iep} \quad \forall (e,p) \in \mathcal{E} \times P, e > 0 \tag{48}$$
$$s_{0p} = C_p - \sum_{i \in A} u_{i0p} \quad \forall p \in P \tag{49}$$
$$s_{ep} \geq 0 \quad \forall (e,p) \in \mathcal{E} \times P \tag{50}$$

Ces nombres importants de variables continues et de contraintes supplémentaires sont essentiellement dus au fait que ce modèle ne dispose pas de variable permettant l'identification nette et précise du début (respectivement de la fin) de l'exécution de chaque activité correspondant au point de consommation (resp. de production) de la ressource. Ainsi, pour déterminer la consommation (respectivement la production) des activités débutant (resp. terminant) leur exécution à l'événement e, on utilise à la fois les contraintes (41), (42) et (43) (resp. (45), (46) et (47)).

Explicitement, si une activité i termine son exécution à l'événement e, alors la contrainte (41) a pour vocation d'imposer que la valeur de la variable p_{iep} qui détermine sa production de ressource p à cet événement e, soit au moins égale à sa production de ressource c^+_{ip}, tandis que les deux autres contraintes (42) et (43) imposent que la valeur de cette même variable, à ce même événement e, pour la même activité i, soit au plus égale à cette même production de ressource c^+_{ip}. Cela revient à imposer que la valeur de la variable p_{iep} soit égale à c^+_{ip}, si l'activité i termine son exécution à l'événement e, pour toute ressource $p \in P$. La combinaison de ces contraintes permet également d'annuler la valeur de cette variable à tout autre événement, afin de ne rendre significative que sa valeur à l'événement e considéré plus haut.

De même, dans le cas de la *consommation* de ressource, la contrainte (45) impose que la variable u_{iep} soit au moins égale à c^-_{ip} (la consommation de la ressource p de l'activité i); les contraintes (46) et (47) imposent alors qu'elle ne dépasse pas cette même valeur c^-_{ip}. Ici, on considère que l'activité i débute à l'événement e et se termine à un autre événement quelconque. Ainsi, de façon analogue à la variable p_{iep}, il résulte de la combinaison des contraintes (44), (45), (46), et (47) que la variable u_{iep} sera égale à c^-_{ip} à l'événement e, et nulle pour tous les autres événements.

La contrainte (48) détermine le niveau de stock de la ressource $p \in P$ à l'événement e. Il est égal au niveau de stock à l'événement précédent, moins la somme des quantités produites de la même ressource par les activités en fin d'exécution, moins la somme de celles consommées par les activités en cours, tout ceci à l'événement e. Donc, il est également possible de supprimer les variables s_{ep}, en les remplaçant par leur valeur.

La contrainte (50) stipule qu'une activité ne peut débuter son exécution que si elle dispose de suffisamment de ressource. Pour finir, alors que la contrainte (40) (respectivement la contrainte (44)) interdit que les quantités produites (resp. consommées) soient négatives, la contrainte (50) permet de garantir de façon analogue aux contraintes (5) et (26), qu'on ne sera en aucun cas en situation de déficit de ressource.

5 RÉSULTATS EXPÉRIMENTAUX

Tous les tests ont été effectués sur un PC Dell Xeon 5110 biprocesseur cadencé à 1,6 GHz, avec 4 GB de RAM, tournant sous le système d'exploitation Fedora de Linux. Toutes les formulations ont été codées en C++, dans un environnement ILOG-Concert version 2.6. Nous avons utilisé le solveur commercial ILOG-Cplex version 11.1 pour les résolutions. Le temps limite de résolution pour chaque instance a été fixé à 500 secondes.

5.1 Les instances

Dans notre étude, nous avons utilisé les instances suivantes :

KSD30 : Nous avons retenu les 100 premières des

480 instances (de 30 activités et 4 ressources) générées par Kolisch *et al.* [16].

BL : Les 39 instances (de 20 à 25 activités et 3 ressources) proposées par Baptiste et Le Pape [6]. Connues pour être fortement cumulatives, elles ont été conçues pour répondre aux critiques sur le caractère fortement disjonctif des instances KSD.

PACK : Les 55 instances de 17 à 35 activités, pour 3 ressources, proposées par Carlier et Néron [10] comportant très peu de contraintes de précédences, et connues comme assez difficiles à résoudre [3].

Ces instances issues de la littérature n'ont pas toujours fait l'unanimité et ont même été l'objet de critiques. Par exemple, concernant les instances KSD, il apparaît que les instances les plus difficiles à résoudre sont celles qui sont les plus disjonctives, ce qui n'est pas représentatif des instances réelles du RCPSP. Soulignons de plus que la plupart de ces instances ne comportent que des durées opératoires relativement courtes (comprises entre 0 et 10 pour les KSD, par exemple). D'emblée, ceci confère un avantage aux formulations utilisant des variables indicées par le temps, car, dans ce cas, celles-ci ne génèrent pas beaucoup de variables et de contraintes, atténuant ainsi leur handicap lié au fait qu'il s'agit de formulations de type pseudo-polynomial. Alors que dans certaines applications industrielles (pétrochimique ou pharmaceutique, par exemple), on a parfois affaire à des durées longues pour certaines activités, avec de grandes disparités entre les durées. Ces cas se retrouvent également dans le domaine de l'industrie des procédés, et dans certains problèmes d'ordonnancement avec des traitements par lots [14].

De ce fait, nous proposons deux nouveaux types d'instances, qui ne sont en réalité que des versions modifiées de certaines instances existantes : PACK_d et KSD15_d. Nous donnons ci-dessous une description plus précise de la méthode utilisée pour générer ces nouvelles instances.

Pour générer une instance B à partir d'une instance existante A, considérons les paramètres x, y, b et a suivants tel que :

1. On sélectionne les x premières activités non-fictives de l'instance A (on laisse de côté les autres activités ainsi que les contraintes de précédence qui leur sont adjacentes).
2. Pour les activités sélectionnées, on connecte les activités sans prédécesseurs à l'activité 0 et les activités sans successeurs à l'activité $x + 1$.
3. On sélectionne de façon aléatoire y activités parmi les x activités non-fictives et on multiplie leur durée par un coefficient $a + b$, où b est un nombre généré aléatoirement entre 0 et 1 et a un *facteur multiplicatif de la durée* (défini ci-dessous pour chaque type d'instance). Il s'agit d'une valeur arbitraire permettant d'augmenter les durées. Les durées ainsi obtenues sont arrondies au nombre entier le plus proche.

KSD15_d: Ces instances ont été créées à partir des KSD30 selon la procédure évoquée ci-dessus, avec les paramètres suivants : $x = 15$, ce qui correspond à la moitié des activités initiales du KSD30 ; $y = 7$ c'est-à-dire que parmi les 15 activités sélectionnées, seule la durée opératoire de 7 activités d'entre elles sera augmentée, avec la valeur du facteur multiplicatif fixé à $a = 25$. On obtient ainsi des instances de taille plus réduite ($n = 15$), mais avec des durées opératoires plus grandes, allant ainsi de 1 jusqu'à environ 250 unités de temps.

PACK_d : Ce deuxième type d'instances proposées a été obtenu par la modification des instances PACK, avec les paramètres suivants : la totalité des activités initiales $x = n$, avec n le nombre d'activités dans les instances PACK. On modifie la durée opératoire de 10 de ces activités ($y = 10$), en utilisant un facteur multiplicatif $a = 50$. On obtient donc 55 instances de même taille que les PACK originaux, mais avec des durées d'activités atteignant parfois le millier.

Toutes ces nouvelles instances sont disponibles sur demande auprès des auteurs. De plus, pour une description plus exhaustive des ces différentes instances, nous invitons le lecteur à se referer à l'article [17].

5.2 Adaptation des instances au problème étudié

Les instances évoquées ne prenant pas en compte l'aspect consommation et production de ressource du RCPSP, nous les avons modifiées, en générant pour chaque activité de chaque instance, trois nouvelles ressources, cette fois non-renouvelables. Pour chacune de ces nouvelles ressources, la quantité c_{ip}^- consommée (respectivement la quantité c_{ip}^+ produite) au début (respectivement à la fin) de l'exécution est générée aléatoirement entre 0 et 10. À chacune de ces ressources est également associé un stock initial (capacité initiale) C_p. Ces différentes caractéristiques des nouvelles ressources sont générées aléatoirement sous deux contraintes essentielles :

- Aucune quantité consommée ne doit être supérieure au stock initial. Ceci autorise n'importe quelle activité à démarrer le projet.
- Afin de ne pas rendre le problème plus disjonctif qu'il ne l'était avant, la somme des quantités produites doit rester supérieure à la somme des quantités consommées.

5.3 Résultats

Le tableau 1 présente le résultat de nos tests numériques.

Tableau 1: Résultats de la résolution exacte du RCPSP avec consommation et production de ressource

Inst.	Modèles	%Ent.	%Opt	Écart	ΔCPM	Tps
KSD30	DT	84	63	10,21	25,20	52,60
	DDT	82	71	0.13	7.65	83.87
	OOE_Prec	78	1	52.63	76.13	415.70
	FCT	69	20	46.65	70.35	289.39
	SEE	2	0	179.86	179.86	
	OOE	1	0	65.96	65.96	
PACK	OOE	94.55	1.82	20.44	277.83	110.85
	OOE_Prec	92.73	3.64	13.13	258.18	449.26
	DT	90.91	18.18	48.04	365.49	126.63
	DDT	47.27	32.73	1.33	245.66	168.04
	SEE	29.09	0	3.71	134.13	
	FCT	9.09	0	5.90	96.41	
BL	OOE_Prec	100	0	17.61	72.57	
	DDT	94.87	48.72	1.32	47.62	125.55
	DT	87.18	38.46	49.82	119.82	108.60
	OOE	74.36	0	27.64	87.56	
	SEE	41.03	0	31.11	88.51	
	FCT	20.51	0	26.79	72.51	
KSD15_d	FCT	100	93.94	0.12	10.15	18.26
	OOE_Prec	100	80.81	0.05	10.08	30.69
	OOE	100	79.80	0.10	10.14	62.33
	SEE	98.99	75.76	0.32	10.30	49.34
	DT	0	0			
	DDT	0	0			
PACK_d	OOE_Prec	96.36	5.45	1.62	249.77	252.09
	OOE	96.36	5.45	5.80	264.41	320.62
	SEE	29.09	0	4.87	146.09	
	FCT	5.45	1.82	0	44.02	100.01
	DT	0	0			
	DDT	0 %	0 %			

On note %Entiers : le pourcentage de solutions entières trouvées au bout des 500 secondes de résolution ; %Opt : le pourcentage de solutions optimales trouvées au bout des 500 secondes ; Écart : l'écart moyen par rapport à la meilleure solution connue ; ΔCPM : l'écart par rapport à la valeur du chemin critique, et Tps : le temps moyen mis pour trouver une solution optimale.

Pour les instances KSD30, les formulations à temps discret présentent les meilleurs résultats et DDT affiche la meilleure performance. Cette dernière trouve des solutions entières pour 84% des instances, avec 63% de solutions optimales. Le modèle OOE_Prec (3^e performance) trouve des solutions entières pour 78% des cas, même s'il ne prouve l'optimalité que pour très peu d'entre elles.

Concernant les instances PACK, on constate que les modèles OOE et OOE_Prec trouvent le plus grand nombre de solutions entières, battant même les formulations à temps discret DT (deuxième meilleure performance) et DDT (troisième meilleure performance). Cependant, si l'on réduit le critère de performance au seul nombre de solutions optimales (%Opt), DDT suivie de DT fournissent toujours les meilleurs scores.

Les résultats sur les instances BL montrent que même si les modèles à temps discret (DDT suivi de DDT) trouvent le plus grand nombre de solutions optimales, le modèle OOE_Prec affiche la meilleure performance en termes de solutions entières trouvées.

Sur les instances KSD15_d, la formulation FCT est la meilleure, suivie de près par OOE_Prec et OOE. L'ajout de nouvelles ressources sur des instances comportant déjà de très longues durées opératoires rend les modèles à temps discret (DT et DDT) très exigeants en mémoire et donc incapables des traiter ces instances dans les conditions de tests évoquées plus haut (dans notre estimation, il faudrait sans doute une machine comportant au moins 7 Go de RAM pour être à même de les résoudre). Il en est de même pour les instances PACK_d. Pour ces dernières, le modèle OOE_Prec, suivi des modèles OOE, puis SEE et FCT conduisent aux meilleures performances.

De façon globale, on constate que le modèle OOE et sa variante OOE_Prec présentent quasiment la meilleure performance sur trois types d'instances (PACK, BL et KSD15_d), la deuxième meilleure performance sur un type d'instance (KSD15_d) et la troisième sur un autre type (KSD30). Ceci nous amène à conclure que globalement, les nouvelles formulations on/off seraient les mieux adaptées pour le RCPSP avec consommation et production de ressources.

6 CONCLUSION

Dans cet article, nous avons étudié la variante du RCPSP avec production et consommation de ressources lors de l'exécution des activités et nous avons proposé deux modèles PLNE basés sur la notion d'événements (*start/end* et *on/off*).

Nos résultats numériques sur diverses instances montrent qu'en plus d'être la plus indiquée que les modèles classiques sur les instances à longues durées, le modèle *on/off* résout le plus grand nombre d'instances, tous types d'instances confondus.

References

[1] R. Alvarez-Valdès and J.M. Tamarit, "The project scheduling polyhedron: dimension, facets and lifting theorems", *European Journal of Operational Research*, 67(2): 204–220, 1993.

[2] D. Applegate and W. Cook, "A computational study of job-shop scheduling", *ORSA Journal on Computing*, 3(2): 149–156, 1991.

[3] C. Artigues, P. Michelon, and S. Reusser, "Insertion techniques for static and dynamic resource-constrained project scheduling", *European Journal of Operational Research*, 149(2): 249–267, 2003.

[4] C. Artigues, O. Koné , P. Lopez, M. Mongeau, E. Néron, and D. Rivreau, "Computational experiments", in C. Artigues, S. Demassey, and E. Néron, (Eds.), *Resource-constrained*

project scheduling: Models, algorithms, extensions and applications, ISTE/Wiley, pages 98–102, 2008.

[5] E. Balas, "Project scheduling with resource constraints", in E.M.L. Beale, (Ed.), *Applications of Mathematical Programming Techniques*, pages 187–200, American Elsevier, 1970.

[6] P. Baptiste and C. Le Pape, "Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems", *Constraints*, 5(1-2): 119–139, 2000.

[7] J. Blazewicz, J. Lenstra, and A.H.G. Rinnooy Kan, "Scheduling subject to resource constraints: Classification and complexity", *Discrete Applied Mathematics*, 5(1): 11–24, 1983.

[8] H. Bouly, J. Carlier, A. Moukrim, and M. Russo, "Solving RCPSP with resources production possibility by tasks", In : *MHOSI'2005, 24–26 Avril*, 2005.

[9] P. Brucker, "Scheduling and constraint propagation", *Discrete Applied Mathematics*, 123(1–3): 227–256, 2002,

[10] J. Carlier and E. Néron, "On linear lower bounds for resource constrained project scheduling problem", *European Journal of Operational Research*, 149: 314–324 ,2003.

[11] J. Carlier, A. Moukrim, and H. Xu, "The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm", *Discrete Applied Mathematics*, 2009.

[12] N. Christofides, R. Alvarez-Valdès, and J.M. Tamarit , "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research*, 29(3): 262–273, 1987.

[13] S. Demassey, C. Artigues, and P. Michelon, "Constraint propagation based cutting planes: an application to the resource-constrained project scheduling problem", *INFORMS Journal on Computing*, 17(1): 52–65, 2005.

[14] J. M. Pinto and I. E. Grossmann, "A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints", *Industrial & Engineering Chemistry Research*, 34(9): 3037–3051, 1995.

[15] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research*, 90(2): 320–333, 1996.

[16] R. Kolisch and A. Sprecher, "PSPLIB - A project scheduling library", *European Journal of Operational Research*, 96(1): 205–216, 1997.

[17] O. Koné, C. Artigues, P. Lopez, and M. Mongeau, "Event-based MILP models for resource- constrained project scheduling problems", *Computers & Operations Research*, 2010. doi:10.1016/j.cor.2009.12.011.

[18] P. Laborie, "Algorithms for propagating resource constraints in a planning and scheduling: Existing approaches and new results", *Artificial Intelligence*, 143: 151–188, 2003.

[19] K. Neumann and C. Schwindt, "Project scheduling with inventory constraints", *Mathematical Methods of Operations Research*, 56:513–533, 2002.

[20] A. Pritsker, L. Watters, and P. Wolfe , "Multiproject scheduling with limited resources: A zero-one programming approach", *Management Science*, 16: 93–108, 1969.

[21] J. C. Zapata, B. M. Hodge, and G. V. Reklaitis, "The multimode resource constrained multiproject scheduling problem: Alternative formulations", *AIChE Journal*, 54(8): 2101–2119, 2008.