Effet d'apprentissage et activités composées dans le RCPSP à haute multiplicité : application à la production de satellite

Duc Anh Le[1*] **Stéphanie Roussel**[1] **Christophe Lecoutre**[2]

[1] ONERA-DTIS, Université de Toulouse, France

[2] CRIL, Univ. Artois & CNRS, France

{duc_anh.le,stephanie.roussel}@onera.fr christophe.lecoutre@cril.fr

Résumé

Cet article traite du problème d'ordonnancement de projet contraint par les ressources et à haute multiplicité (High Multiplicity Resource-Constrained Project Scheduling Problem - HM-RCPSP), dans lequel plusieurs projets sont réalisés plusieurs fois tout en partageant des ressources limitées. Nous étendons ce problème en considérant l'effet d'apprentissage, c.-à-d. le fait que la durée de certaines activités diminue lorsqu'elles sont répétées. De plus, nous supposons que certaines activités sont composées, permettant ainsi de raisonner de manière plus abstraite sur les prérequis et les ressources. Le problème résultant est noté HM-RCPSP/L-C. Un modèle de programmation par contraintes est proposé pour ce problème étendu, incluant une technique de cassage de symétrie. L'efficacité de notre approche est évaluée à travers des expérimentations menées sur des données réelles provenant d'une ligne d'assemblage de satellites, ainsi que sur des benchmarks de la littérature que nous avons adaptés.

1 Introduction

Dans le domaine de la production, un challenge classique est d'ordonnancer plusieurs projets, chaque projet étant lui-même répété plusieurs fois dans un environnement avec des ressources partagées limitées. Dans la littérature, il s'agit du problème d'ordonnancement de projet contraint par les ressources et à haute multiplicité (High Multiplicity Resource-Constrained Project Scheduling Problem - HM-RCPSP) [7]. Une application de ce problème est l'ordonnancement des tâches d'assemblage de produits complexes, par exemple des satellites. En effet, plusieurs types de satellites doivent être produits, chaque type avec son propre processus d'assemblage, avec des quantités à produire qui peuvent varier, et tout en partageant les ressources et les installations de l'usine qui sont limitées.

Deux caractéristiques du problème de ligne d'assemblage de satellite sont difficilement représentables dans le cadre du HM-RCPSP. Tout d'abord, en plus de la relation de précédence entre les activités, il existe une relation de composition, qui permet de représenter le fait qu'une activité en englobe d'autres (démarrant et finissant respectivement avec celle qui commence au plus tôt et celle qui termine au plus tard). Une telle activité, appelée ici *composée*, peut par exemple représenter « l'installation d'un mirroir sur le satellite » qui est composé de plusieurs activités *atomiques*, comme « fixer le miroir », « vérifier l'alignement optique du miroir », *etc.* Les activités composées permettent d'exprimer les précédences de manière plus compacte et de représenter une consommation de ressource globale (la ressource est consommée pendant toute la durée de l'activité composée). Par exemple, un banc de test peut être nécessaire pour l'activité « installation de miroir », représentant non seulement que chaque activité la composant peut utiliser ce banc de test mais aussi que ce banc ne peut pas être utilisé par une autre activité tant que l'installation n'est pas terminée.

Deuxièmement, dans le contexte de la production, le temps requis pour réaliser une opération évolue généralement dans le temps. Au fur et à mesure que le système industriel est opéré, il gagne en efficacité globalement : les techniciens et ingénieurs acquièrent de l'expertise et améliorent éventuellement le processus d'assemblage,

*Papier doctorant : Duc Anh Le[1] est auteur principal.

l'interaction avec la chaîne logistique est consolidée, l'utilisation des outils est améliorée, *etc.* Par exemple, la durée d'assemblage du $5^{ème}$ exemplaire d'un produit peut être égale à la moitié de la durée nécessaire pour le premier. Ce phénomène s'appelle l'*effet d'apprentissage* et la variation de la durée en fonction de la répétition d'une même opération est la *courbe d'apprentissage* ([21]). Dans le cas de produits complexes comme un satellite ou un avion, ce phénomène peut avoir un impact significatif sur les plannings moyen et long terme de l'usine. Prendre en compte l'effet d'apprentissage permet d'estimer de manière plus réaliste les dates auxquelles les produits seront disponibles. Il peut être également pertinent d'évaluer en quoi l'investissement dans des ressources additionnelles améliore ou non la capacité de production de l'usine. Cependant, la considération de l'effet d'apprentissage complexifie le HM-RCPSP car il crée une inter-dépendance entre les différents projets : la durée d'une activité du $i^{ème}$ projet dépend du nombre de fois où cette activité a déjà été réalisée avant.

L'objectif de ce papier est de traiter le RCPSP à haute multiplicité avec effet d'apprentissage et activités composées, que l'on note HM-RCPSP/L-C. Plus précisément, les contributions de ce papier sont les suivantes.

- HM-RCPSP/L-C est formellement défini, notamment par l'introduction d'une composante effet d'apprentissage générique ;
- une méthode de cassage de symétrie est proposée et prouvée correcte ;
- un modèle en Programmation Par Contraintes (PPC) est introduit ;
- de nouvelles instances académiques sont proposées ;
- une expérimentation est conduite sur ces instances mais aussi sur des données provenant d'une usine de production de satellites, montrant ainsi le potentiel de l'approche PPC (avec et sans cassage de symétrie).

Le papier est structuré de la manière suivante. Après avoir présenté les travaux connexes en section 2, HM-RCPSP/L-C est introduit formellement dans la section 3. Nous traitons le cassage de symétrie dans la section 4, définissons le modèle PPC dans la section 5, et présentons les résultats des expérimentations dans la section 6. Finalement, nous concluons et discutons des perspectives dans la section 7.

2 Travaux connexes

Le problème RCPSP est un problème classique d'optimisation combinatoire pour lequel de nombreuses approches de résolution ont été proposées ([2]). Comme décrit dans [11], plusieurs extensions ont été proposées. Dans cette section, nous nous concentrons d'abord sur les extensions qui prennent en compte la présence de plusieurs projets ou leur répétition dans un contexte de ressources partagées. Nous présentons ensuite les travaux dans lesquels l'ordonnancement et l'effet d'apprentissage sont traités simultanément.

Multi-projets et répétition en ordonnancement. Dans l'étude [10], les auteurs présentent plusieurs extensions du problème d'ordonnancement multi-projets sous contrainte de ressources (RCMPSP). Ces extensions peuvent prendre en compte les caractéristiques des activités (par exemple, la préemption ou l'incertitude), les relations entre les activités (par exemple, la concurrence ou les décalages temporels), les projets et les caractéristiques des ressources (par exemple, la renouvelabilité ou la disponibilité). En raison de l'effet d'apprentissage, les projets que nous considérons dans ce papier sont interdépendants, ce qui empêche de considérer le modèle HM-RCPSP/L-C comme un cas particulier du RCMPSP.

Dans [6], les auteurs abordent le RCPSP cyclique (un seul projet est répété à l'infini) par le biais d'une approche de programmation par contraintes en temps discret (DT-CP). L'objectif est de trouver un modèle qui intercale plusieurs répétitions du projet afin d'utiliser les ressources de manière optimale. Dans le contexte des projets de construction ([8]), des activités identiques sont répétées un certain nombre donné de fois. Dans [7], les auteurs s'intéressent au HM-RCPSP/max dans lequel plusieurs projets sont répétés et partagent des ressources communes. Les projets peuvent être liés par une relation de précédence généralisée qui spécifie des délais maximums entre les activités. Ils proposent une méthode de cassage de symétrie pour les projets identiques de la même classe.

Effet d'apprentissage et ordonnancement Le phénomène de l'effet d'apprentissage a été introduit dans [19]. L'auteur définit un modèle d'apprentissage log-linéaire que nous utilisons dans nos expériences. Diverses courbes d'apprentissage ont été proposées depuis lors ([21]).

Il existe deux types de modélisation de l'effet d'apprentissage dans le cadre de l'ordonnancement : (i) *modélisation basée sur la position*, dans laquelle l'effet d'apprentissage est fonction du nombre de fois qu'une activité a été exécutée ; et (ii) *modélisation basée sur la somme des durées* dans laquelle l'effet d'apprentissage est fonction de la durée cumulée de l'exécution de chaque activité ([5]). Certains travaux ont étudié les conditions dans lesquelles le problème devient po-

lynomial ([4, 3, 22]). Dans [1], le problème d'ordonnancement avec effet d'apprentissage modélisé avec la position est étudié à l'aide d'une approche d'approximation en deux étapes. Les projets sont identiques et peuvent être exécutés en parallèle tout en respectant la capacité des ressources, et chaque activité nécessite exactement une unité d'un type de ressource spécifique.

L'affectation du personnel a été étudiée dans [20] et [16], respectivement avec une modélisation d'effet d'apprentissage basée sur la position et sur la durée cumulée. L'effet d'apprentissage a également été étudié dans le cas de temps discret/ressource trade-off (Discrete Time/Resource Trade-off Problem - DTRTP), comme présenté dans [18, 17]. Le DTRTP est un sous-problème du RCPSP multi-mode, où la durée de chaque activité dépend du nombre de travailleurs ou de ressources qui lui sont affectés. Enfin, dans [12], les auteurs introduisent quatre formulations DT-CP pour le RCPSP avec un effet d'apprentissage basé sur la position, et fournissent une comparaison empirique de leurs performances en matière d'ordonnancement et de bornes inférieures. Dans leur étude, l'effet d'apprentissage est modélisé par deux valeurs de durée pour chaque activité : une nominale et une réduite, et il n'y a pas de facteur de répétition.

Pour conclure cette section, à notre connaissance, il n'y a pas de travaux dans la littérature traitant simultanément de multi-projets partageant des ressources, avec des activités composées et dont la durée de l'activité dépend de l'effet d'apprentissage.

3 HM-RCPSP/L-C

Entrées du problème. Une instance de HM-RCPSP/L-C est un tuple composé des éléments suivants :

- $\mathcal{C}$ est l'ensemble des classes de projet ;
- pour chaque classe $c \in \mathcal{C}$, $\mathcal{I}_c$ est l'ensemble des projets (ou instances) de c qui doivent être réalisés. Pour chaque projet $p \in \mathcal{I}_c$, due_p représente sa date de livraison ;
- pour chaque classe $c \in \mathcal{C}$, $\mathcal{A}_c$ représente l'ensemble des activités non-préemptives qui doivent être réalisées pour chaque projet de c. Les activités $\mathcal{A}_c$ peuvent être partitionnés en deux ensembles $\mathcal{A}_c^A$ et $\mathcal{A}_c^C$ qui représentent respectivement les activités atomiques et composées. Intuitivement, une activité composée peut être vue comme un groupe d'activités. Une telle activité recouvre temporellement toutes les activités dans le groupe ;
- pour chaque classe $c \in \mathcal{C}$, $\mathcal{H}_c \subseteq \mathcal{A}_c^C \times \mathcal{A}_c$ est la relation de composition de c. Si a est une activité composée et b une activité, alors $(a, b) \in \mathcal{H}_c$ exprime que b est englobée par a (ou une activité enfant de a). Notons qu'il est possible d'avoir (a, b) dans $\mathcal{H}_c$ avec b qui est également une activité composée ;
- pour chaque classe $c \in \mathcal{C}$ et pour chaque activité atomique $a \in \mathcal{A}_c^A$, nous considérons une fonction de durée monotone décroissante, notée $dur_a : \mathbb{N}^* \to \mathbb{N}^*$, qui renvoie la durée de l'activité en fonction du nombre de fois où elle a été complètement exécutée auparavant ;
- pour chaque classe $c \in \mathcal{C}$, $\mathcal{P}_c$ est une relation de précédence entre les activités de $\mathcal{A}_c$. Si $(a, b) \in \mathcal{P}_c$, alors a doit être terminé avant le début de b ;
- $\mathcal{R}$ est l'ensemble des ressources cumulatives renouvelables. Pour chaque $r \in \mathcal{R}$, $capa_r$ est la capacité de r ;
- pour chaque ressource $r \in \mathcal{R}$, pour chaque classe $c \in \mathcal{C}$, pour chaque activité $a \in \mathcal{A}_c$, $cons_{r,a}$ représente la quantité de r consommé par a ;
- $H \in \mathbb{N}^+$ est l'horizon temporel.

Nous notons a^p l'instance de $a \in \mathcal{A}_c$ qui doit être réalisée pour le projet $p \in \mathcal{I}_c$ de la classe $c \in \mathcal{C}$.

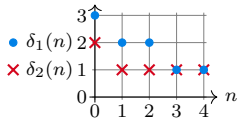
Hypothèses. Une instance de HM-RCPSP/L-C est bien formée si et seulement si les hypothèses suivantes sont satisfaites. Toutes les instances considérées dans ce papier sont bien formées, sauf mention contraire.

1. Les projets et les activités sont associés à une unique classe.
2. Pour chaque classe $c \in \mathcal{C}$, nous supposons que $\mathcal{A}_c^A$ contient une activité virtuelle de début α_c et une activité virtuelle de fin ω_c. Ces activités ne consomment pas de ressource, ne sont pas englobées par aucune autre activité et leur durée est nulle. De plus, α_c précède (resp. ω_c suit) toutes les autres activités de $\mathcal{A}_c$.
3. Pour chaque classe $c \in \mathcal{C}$, le graphe induit par la relation de composition $\mathcal{H}_c$ est une forêt $\mathcal{G}_c^H$, *i.e.* chaque nœud a au plus un parent et il est acyclique. Formellement, $\mathcal{G}_c^H$ contient un nœud n_a pour chaque a de $\mathcal{A}_c$ et contient un arc (n_a, n_b) pour chaque paire $(a, b) \in \mathcal{H}_c$.
4. Pour chaque classe $c \in \mathcal{C}$, pour chaque arbre de la forêt $\mathcal{G}_c^H$, et pour chaque chemin de la racine vers une feuille, les capacités des ressources permettent de réaliser simultanément toutes les activités sur le chemin.
5. Pour chaque classe $c \in \mathcal{C}$, le graphe $\mathcal{G}_c^P$ induit par la relation de précédence $\mathcal{P}_c$ est acyclique. Pour construire ce graphe, nous passons par une version atomique de $\mathcal{P}_c$ que nous notons $\mathcal{P}_c^A$ et dans

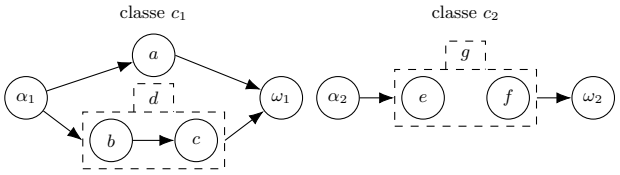
$\mathcal{C}$	$\mathcal{A}_c$	dur_a	r_1	r_2
c_1	a	δ_1	1	0
	b	δ_2	0	1
	c	δ_1	0	1
	d	-	1	0
c_2	e	δ_2	0	1
	f	δ_1	1	0
	g	-	1	0

$\mathcal{C}$	$\mathcal{I}_c$	due_p
c_1	p_1	7
	p_2	8
c_2	q	6

$\mathcal{R}$	$capa_r$
r_1	3
r_2	2



(a) Exemple de classes, projets, activités et ressources



(b) Relation de composition et de précédence

FIGURE 1 – Instance jouet de HM-RCPSP/L-C

lesquelles toutes les précédences de $\mathcal{P}_c$ sont transposées sur les activités atomiques. Nous créons ensuite un nœud par activité atomique et un arc pour chaque précédence.

Exemple 1. La figure 1 illustre un exemple de HM-RCPSP/L-C. Il y a deux classes, c_1 et c_2. Les activités de chaque classe sont décrites dans le tableau de gauche de la figure 1a. Pour chaque projet de la classe c_1, 4 activités (+2 activités fictives α_1 et ω_1) doivent être réalisées. Les activités a, b, c, α_1 et ω_1 sont atomiques ($\mathcal{A}^A_{c_1} = \{\alpha_1, \omega_1, a, b, c\}$). La durée de l'activité a est donnée par la fonction δ_1 décrite dans le coin inférieur droit : si aucune instance de l'activité a n'a encore été achevée (c'est-à-dire $n = 0$), sa durée est 3, si une exécution a été entièrement achevée avant le début de l'activité a, sa durée est 2, et ainsi de suite. L'activité d est composée ($\mathcal{A}^C_{c_1} = \{d\}$) et ses enfants sont b et c, comme le montre le graphique à la gauche de la Figure 1b. Ce graphique représente également la relation de précédence $\mathcal{P}_{c_1}$. Par exemple, (α_1, d) et (b, c) appartiennent tous deux à $\mathcal{P}_{c_1}$. r_1 et r_2 sont les deux ressources et ont respectivement une capacité égale à 3 et 2. Les activités atomiques et composées peuvent consommer ces ressources. Par exemple, $cons_{r_1,d} = 1$ et $cons_{r_2,b} = 1$. Deux projets, p_1 et p_2, de la classe c_1 et un projet, q, de la classe c_2 doivent être réalisés avec des dates d'échéance respectivement égales à 7, 8 et 6.

Ordonnancement. Un ordonnancement σ d'une instance de HM-RCPSP/L-C est défini par l'assignation d'une date de début à chaque activité atomique de chaque projet de chaque classe.

Formellement, la date de début (resp. la date de fin) de a^p dans σ est notée $start^\sigma(a^p)$ (resp. $end^\sigma(a^p)$). Si a^p est atomique, en raison de l'effet d'apprentissage, la durée de a^p dépend du nombre de fois où l'activité a a déjà été réalisée pour d'autres projets de la même classe au début de a^p. La fonction nc^σ appliquée à l'activité a et au pas de temps t, renvoie le nombre de fois où l'activité a a été achevée à t. Formellement, pour chaque classe $c \in \mathcal{C}$, pour chaque projet $p \in \mathcal{I}_c$ et pour chaque atomique activité $a \in \mathcal{A}^A_c$, on définit :

- $end^\sigma(a^p) = start^\sigma(a^p) + dur_a(nc^\sigma(a, start^\sigma(a^p)))$.
- $nc^\sigma(a, t) = \sum_{q \in \mathcal{I}_c \setminus \{p\}} (end^\sigma(a^q) \leq t)$

En pratique, la valeur de ces fonctions peut être déterminée pour un ordonnancement donné σ en triant l'exécution des activités par leur date de début. Les dates de début des deux activités virtuelles α_c et ω_c sont alors définies par :

- $start^\sigma(\alpha^p_c) = \min_{a \in \mathcal{A}^A_c \setminus \{\alpha_c\}} start^\sigma(a^p)$
- $start^\sigma(\omega^p_c) = \max_{a \in \mathcal{A}^A_c \setminus \{\omega_c\}} end^\sigma(a^p)$

Les dates de début et de fin des activités composées peuvent être déduites de celles des activités atomiques : la date de début d'une activité composée a (resp. la date de fin) est égale à la date minimum de début (resp. la date maximum de fin) des activités englobées par a.

Solution. Un ordonnancement σ est une solution pour une instance de HM-RCPSP/L-C si et seulement si les précédences (Equation 1) sont satisfaites et si les capacités des ressources sont respectées tout au long de l'horizon temporel (Equation 2).

$\forall c \in \mathcal{C}, \forall (a, b) \in \mathcal{P}_c, \forall p \in \mathcal{I}_c$

$$end^\sigma(a^p) \leq start^\sigma(b^p) \tag{1}$$

$\forall r \in \mathcal{R}, \forall t \in [\![0, H]\!]$

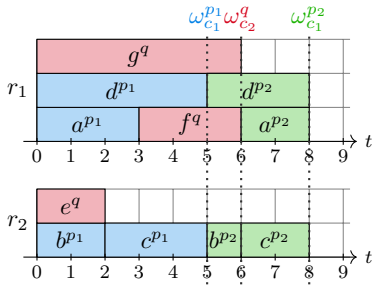
$$\Big(\sum_{\substack{c \in \mathcal{C}, a \in \mathcal{A}_c, p \in \mathcal{I}_c \\ start^\sigma(a^p) \leq t < end^\sigma(a^p)}} cons_{r,a}\Big) \leq capa_r \tag{2}$$

Solution optimale. Une solution σ est optimale si elle minimise les deux critères suivants avec un ordre lexicographique : (1) la somme des retards des projets (Equation 3), et (2) le makespan (Equation 4).

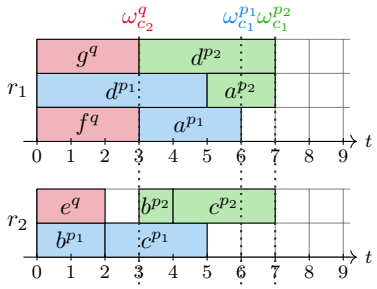
$$\sum_{c \in \mathcal{C}, p \in \mathcal{I}_c} \max\{0, end^\sigma(\omega^p_c) - due_p\} \tag{3}$$

$$\max_{c \in \mathcal{C}, p \in \mathcal{I}_c} end^\sigma(\omega^p_c) \tag{4}$$

Exemple 2. Les figures 2a et 2b illustrent respectivement deux solutions σ_1 et σ_2. Nous avons $start^\sigma(c^{p_1}) = 2$, c'est-à-dire que l'exécution de c dans le projet p_1 commence à l'instant 2 dans la solution



(a) Solution σ_1 de l'exemple 1



(b) Solution σ_2 de l'exemple 1

σ_1. Comme $\delta_1(0) = 3$, c^{p_1} dure 3 unités de temps et $end^{\sigma}(c^{p_1}) = 5$. Toujours dans σ_1, au début de c^{p_2} (pas de temps 6), une exécution de c est terminée donc la durée de c^{p_2} est égale à 2. Notons que dans σ_2, au démarrage de l'activité c^{p_2}, l'exécution de c^{p_1} n'est pas encore terminée, donc la durée de c^{p_2} est égale à 3 . Comme l'activité composée g^q s'étend sur les activités e^q et f^q, dans la solution σ_1, g^q commence au temps 0 et se termine au temps 6. Cependant, dans σ_2, g^q se termine à l'instant 3. Dans les deux cas, la ressource r_1 est consommée pendant toute la durée de g^q. Les deux solutions respectent les dates d'échéance des projets. Par exemple, $end^{\sigma_1}(\omega_{c_1}^{p_1})$ est égal à 5, ce qui est inférieur à 7. Cependant, le makespan de σ_2 est inférieur à la durée de vie de σ_1, donc σ_2 est une meilleure solution.

4 Cassage de symétrie

Dans cette section, nous établissons l'existence de symétries entre des activités identiques issues de différents projets au sein d'une même classe.

Premièrement, en raison de l'effet d'apprentissage, une activité atomique a qui commence dans le projet q plus tard que dans le projet p peut se terminer plus tôt dans q. Nous ignorons ce cas particulier à travers la définition 1 et le lemme 1.

Définition 1 (Début-fin-cohérence) *Une solution σ est* début-fin-cohérente *si pour chaque classe $c \in \mathcal{C}$, pour chaque activité atomique $a \in \mathcal{A}_c^A$, pour chaque paire de projets $p, q \in \mathcal{I}_c$, on a $start^{\sigma}(a^p) < start^{\sigma}(a^q) \iff end^{\sigma}(a^p) < end^{\sigma}(a^q)$*

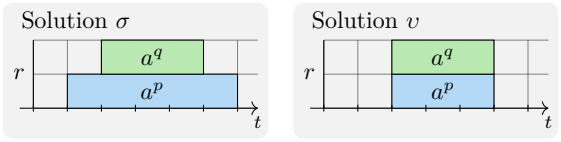


FIGURE 3 – Illustration du lemme 1

Lemme 1 *Si une solution σ n'est pas début-fin-cohérente, alors il existe une solution début-fin-cohérente υ qui est équivalente ou meilleure que σ par rapport aux deux critères d'optimisation.*

Démonstration [Schéma] Soient a une activité atomique et p et q deux projets pour lesquels σ n'est pas début-fin-cohérente, comme illustré sur la figure 3. Nous construisons un ordonnancement υ qui est égal à σ sauf que la date de début de a^p est retardée jusqu'à la date de début de a^q. Les contraintes de précédence et de ressources sont satisfaites dans υ. ■

Dans la suite de ce papier, nous ne considérons que des solutions début-fin-cohérentes.

Nous définissons des mouvements spécifiques sur les ordonnancements, que nous appelons permutations, et montrons qu'ils préservent la satisfaction des contraintes de précédence et de ressources.

Définition 2 ((p,q)-permutation) *Etant donnés une solution σ, une classe $c \in \mathcal{C}$ et p, q deux projets dans $\mathcal{I}_c$, un ordonnancement π est une* (p,q)-permutation *de σ si pour toutes les activités atomiques $a \in \mathcal{A}_c^A \setminus \{\alpha_c, \omega_c\}$, on a :*

- $\forall o \in \mathcal{I}_c \setminus \{p, q\}$, $start^{\pi}(a^o) = start^{\sigma}(a^o)$
- *si* $start^{\sigma}(a^p) > start^{\sigma}(a^q)$, *alors* $start^{\pi}(a^q) = start^{\sigma}(a^p)$ *et* $start^{\pi}(a^p) = start^{\sigma}(a^q)$,
- *sinon* $start^{\pi}(a^p) = start^{\sigma}(a^p)$ *et* $start^{\pi}(a^q) = start^{\sigma}(a^q)$.

L'opération qui consiste à permuter les dates de début d'une activité atomique spécifique, excluant la source et le puits, entre deux projets est appelée un échange.

Intuitivement, une (p,q)-permutation consiste à échanger toutes les activités atomiques de p qui commencent après leurs activités homologues dans q. Étant donné que la date de début des activités composées et de la source et du puits dépend uniquement des dates de début des activités atomiques, un corollaire de la définition 2 est que toutes les activités dans p commencent avant leurs activités homologues dans q. Notons également qu'échanger les dates de début des activités atomiques a^p et a^q permute également leur durée et leurs dates de fin.

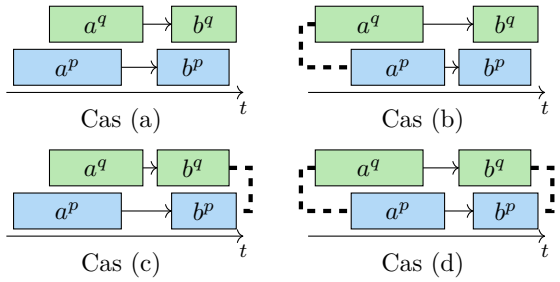
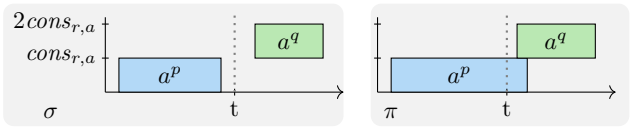
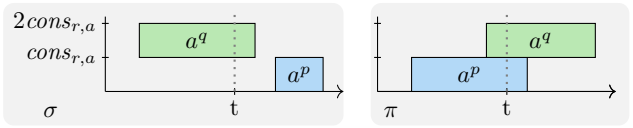


FIGURE 4 – Illustration du lemme 2



(a) Consommation de r à t passant de 0 à $cons_{r,a}$



(b) Consommation de r à t passant de $cons_{r,a}$ à $2cons_{r,a}$

FIGURE 5 – Illustration du lemme 3

Lemme 2 *Étant donnée une solution σ, une classe $c \in \mathcal{C}$ et deux projets $p, q \in \mathcal{I}_c$. La (p,q)-permutation π de σ est précédence-faisable, i.e. elle satisfait l'équation 1.*

Démonstration [Schéma] Comme exprimé dans l'hypothèse 5, nous considérons la précédence sur les activités atomiques. Si π n'est pas précédence-faisable, il existe deux activités atomiques a et b telles que a précède b et telles que $end^\pi(a^p) > start^\pi(b^p)$ ou $end^\pi(a^q) > start^\pi(b^q)$. Comme σ est précédence-faisable, alors $end^\sigma(a^p) \leq start^\sigma(b^p)$ et $end^\sigma(a^q) \leq start^\sigma(b^q)$. Nous considérons ensuite tous les cas en terme d'échange, comme illustré sur la figure 4 où les traits en pointillés correspondent à un échange des activités dans π. Dans tous ces cas, si σ est précédence-faisable alors π l'est aussi. ■

Lemme 3 *Étant donnés une solution σ, une classe $c \in \mathcal{C}$ et deux projets $p, q \in \mathcal{I}_c$, la (p,q)-permutation π de σ est ressource-faisable, i.e. elle satisfait l'équation 2.*

Démonstration [Schéma] La faisabilité des ressources est facile à prouver dans le cas des activités atomiques. En effet, lors d'un échange d'activités, la consommation de ressources ne change pas. Cependant, dans le cas d'activités composées, il est possible que certaines activités enfants soient échangées et d'autres non, ce qui signifie que la durée des activités composées peut changer lors de l'exécution d'une permutation (p,q). Dans ce qui suit, nous considérons tous les cas et montrons que même avec des activités composées, la permutation reste réalisable en termes de ressources.

Nous définissons une fonction $\gamma_{r,a,t}(p,\sigma)$ qui représente combien d'unités de ressource r l'activité a exécutée pour le projet p consomme au pas de temps donné t dans σ, c.-à-d. $\gamma_{r,a,t}(p,\sigma) = cons_{r,a}$ si $start^\sigma(a^p) \leq t < end^\sigma(a^p)$, 0 sinon. La consommation d'une ressource r à chaque pas de temps t dans un ordonnancement σ est donnée par la formule $\sum_{c\in\mathcal{C},p\in\mathcal{I}_c,a\in\mathcal{A}_c} \gamma_{r,a,t}(p,\sigma)$. La preuve se divise en plusieurs étapes.

1. Nous montrons que si π n'est pas ressource-faisable, il existe un pas de temps t et une ressource r telle que $\sum_{c\in\mathcal{C},p\in\mathcal{I}_c,a\in\mathcal{A}_c} \gamma_{r,a,t}(p,\pi) - \gamma_{r,a,t}(p,\sigma) > 0$. Nous considérons cette ressource spécifique r et ce pas de temps t.
2. Nous prouvons que la violation de capacité des ressources ne provient que des projets p et q.
3. Soient c la classe associée à p et q et a une activité de $\mathcal{A}_c$. Il faut montrer que la consommation de r induite par les deux activités a^p et a^q est équivalente dans π et dans σ, i.e. $\gamma_{r,a,t}(p,\pi) - \gamma_{r,a,t}(p,\sigma) + \gamma_{r,a,t}(q,\pi) - \gamma_{r,a,t}(q,\sigma) = 0$. Si a est une activité atomique alors la consommation de r est inchangée. Si a est une activité composée alors nous considérons deux cas illustrés sur la figure 5 : (i) les activités a^p et a^q n'utilisent pas r à l'instant t dans σ, mais au moins l'un d'eux l'utilise dans π, alors nous avons $\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\sigma) = 0$ et $\gamma_{r,a,t}(p,\pi) + \gamma_{r,a,t}(q,\pi) \geq cons_{r,a}$ (figure 5a) (ii) au moment t, exactement une activité (a^p ou a^q) utilise r dans σ et elles l'utilisent toutes les deux dans π, alors $\gamma_{r,a,t}(p,\sigma) + \gamma_{r,a,t}(q,\sigma) = cons_{r,a}$ et $\gamma_{r,a,t}(p,\pi) + \gamma_{r,a,t}(q,\pi) = 2cons_{r,a}$. Nous montrons que ces deux cas aboutissent tous deux à une contradiction.
4. Tous les cas impliquent une contradiction, ce qui signifie que π est ressource-faisible. ■

Lemme 4 *Étant donné un ordonnancement σ, pour chaque classe $c \in \mathcal{C}$, pour chaque paire de projets $p, q \in \mathcal{I}_c$, le makespan de la (p,q)-permutation π de σ est égal au makespan de σ.*

Démonstration $\max_{a\in\mathcal{A}_c}(end^\sigma(a^p), end^\sigma(a^q))$ est la date à laquelle p and q sont terminés dans σ, notée $\epsilon^\sigma_{p,q}$. Dans π, même s'il y a un échange entre les activités, la date à laquelle p et q sont tous deux terminés ne change pas, *i.e.* $\epsilon^\pi_{p,q} = \epsilon^\sigma_{p,q}$. Comme les dates de tous les autres projets dans π restent identiques, les

makespan de π et σ sont égaux. ■

Pour le retard, seule une permutation (p, q) telle que $due_p < due_q$ permet d'obtenir un retard inférieur ou égal. Nous définissons donc un préordre sur les projets cohérent avec les dates de livraison et montrons que cela préserve ou améliore les retards. Notons que nous ne définissons pas un unique préordre $\prec_c$ basé seulement sur la considération des dates de livraison pour une classe c, car nous permettons à l'utilisateur de définir arbitrairement $p \prec_c q$ lorsque $due_p = due_q$, à condition que la relation reste bien formée. Ceci est similaire aux choix arbitraires effectués lors de la publication de la contrainte globale Precedence [15] (par exemple, entre des projets identiques, on peut choisir l'un pour précéder l'autre, comme dans [7]).

Définition 3 *Soit c une classe de $\mathcal{C}$. $\prec_c$ est un préordre strict cohérent avec les dates de livraison pour c si c'est une relation irréflexive, asymétrique et transitive sur l'ensemble $\mathcal{I}_c$ et $\forall p, q \in \mathcal{I}_c$ telle que $due_p < due_q$, alors $p \prec_c q$. $\prec$ est une préordre strict cohérent avec les dates de livraison s'il s'agit de l'union de préordre strict cohérent avec les dates de livraison $\prec_c$ pour chaque classe c de $\mathcal{C}$.*

Dans la suite, pour une classe $c \in \mathcal{C}$, un projet $p \in \mathcal{I}_c$ et un ordonnancement σ, on note τ_p^σ le retard du projet p. τ^σ désigne la somme des retards de tous les projets, tel qu'exprimé dans l'équation 3.

Lemme 5 *Soit σ une solution, $\prec$ un préordre strict cohérent avec les dates de livraison. Soient c une classe dans $\mathcal{C}$, p, q deux projets dans $\mathcal{I}_c$ tels que $p \prec q$, et π la (p, q)-permutation de σ. Alors $\tau^\pi \leq \tau^\sigma$, c'est-à-dire que le retard de π est inférieur ou égal au retard de σ.*

Démonstration [Schéma] À l'exception des projets p et q, toutes les autres activités du projet restent les mêmes dans σ et π. Ainsi, la différence entre les retards des deux solutions vient de la différence de retard des projets p et q dans σ et dans π. Intuitivement, le retard de p est toujours réduit mais q peut se terminer plus tard dans π que dans σ. Il faut donc montrer que $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$ ou $\tau_q^\pi \leq \tau_q^\sigma$. On note respectivement a et b les activités atomiques de $\mathcal{A}_c^A$ qui ont les dernières dates de fin pour p et q dans σ.

1. Si les deux activités a et b entre les projets p et q ne sont pas échangées dans π, alors b^q conclut toujours q dans π, et donc $\tau_q^\pi = \tau_q^\sigma$.
2. Si a^p et a^q sont échangés, mais b^p et b^q non, alors nous avons deux sous-cas. Si $end^\sigma(a^p) \leq end^\sigma(b^q)$ alors nous montrons que b^q conclut toujours q dans π, donc $\tau_q^\pi = \tau_q^\sigma$. Sinon, a^q conclut q dans π et il faut considérer les cas suivants.
   (a) Si $end^\sigma(a^p) \leq due_q$, alors $\tau_q^\pi = \tau_q^\sigma = 0$.
   (b) Si $end^\sigma(a^p) > due_q$, alors $\tau_q^\pi = end^\sigma(a^p) - due_q$ et $\tau_p^\sigma = end^\sigma(a^p) - due_p$. Quelle que soit la position relative de $end^\sigma(b^q)$, due_p et due_q, on a $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$.
3. Il est impossible d'avoir b^p et b^q échangées mais a^p et a^q non.
4. Si a^p et a^q sont échangées, et b^p et b^q échangées, alors $end^\sigma(a^q) \leq end^\sigma(b^q) < end^\sigma(b^p) \leq end^\sigma(a^p)$ et donc b^p et a^q sont respectivement les dernières activités pour p et q dans π. Nous prouvons alors que $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$ dans les cas suivants dans σ : (a) p et q sont en retard, (b) p en retard et q est en avance et (c) p et q sont en avance.

Dans tous ces cas, $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$, ce qui prouve que $\tau^\pi \leq \tau^\sigma$. ■

Nous définissons maintenant une $\prec$-permutation de σ dans laquelle des (p, q)-permutations sont effectuées pour tous p, q tels que $p \prec q$ et montrons que si σ est une solution alors sa $\prec$-permutation est également une solution meilleure ou équivalente à σ.

Définition 4 ($\prec$-permutation) *Soient σ une solution et $\prec$ un préordre strict cohérent avec les dates de livraison. Un ordonnancement π est une $\prec$-permutation de σ si, pour toute classe $c \in \mathcal{C}$, pour toute paire de projets p, q dans $\mathcal{I}_c$ tels que $p \prec q$, π est une (p, q)-permutation de σ.*

Lemme 6 *Soient σ une solution et $\prec$ un préordre strict cohérent avec les dates de livraison. La $\prec$-permutation π de σ est une solution avec un makespan équivalent à celui de σ et un retard au moins aussi bon que celui de σ.*

Démonstration Soit π une permutation $\prec$ d'un ordonnancement σ. π peut être construit itérativement à partir de σ via plusieurs ordonnancements $\pi_0, \ldots, \pi_n$ où $\pi_0 = \sigma$, $\pi_n = \pi$ et $\forall i \in [\![1, n]\!]$, π_i est une (p, q)-permutation de π_{i-1} avec p et q deux projets tels que $p \prec q$. D'après les lemmes 2, 3, 4 et 5, chacune de ces permutations est une solution avec un makespan équivalent et un retard équivalent ou moindre. ■

Ce résultat montre que pour trouver une solution σ à une instance HM-RCPSP/L-C, on peut considérer le préordre strict cohérent avec les dates de livraison $\prec$ et rechercher des solutions dans lesquelles pour tout $c \in \mathcal{C}$, pour tous p et q dans $\mathcal{I}_c$ tels que $p \prec q$, pour toute activité $a \in \mathcal{A}_c$, $start^\sigma(a^p) \leq start^\sigma(a^q)$.

5 Modèle PPC

Dans cet article, nous utilisons la formulation à base de variables intervalles du langage (OPL) associée au

solveur CP Optimizer [14]. Une variable intervalle permet de représenter une activité avec sa date de début (variable), sa durée (variable), sa présence obligatoire ou facultative dans la solution calculée, ses dates de début au plus tôt et de fin au plus tard. La date de début, la date de fin et la durée d'une variable intervalle sont respectivement accessibles via les fonctions startOf, endOf et lengthOf.

Pour chaque classe $c \in \mathcal{C}$, pour chaque projet $p \in \mathcal{I}_c$, pour chaque activité $a \in \mathcal{A}_c$, nous considérons les variables suivantes :

- $\mathbf{itv}_{a,p} \in [\![0, H]\!]$ est une variable d'intervalle pour l'exécution de l'activité a^p ;
- $\mathbf{n}_{a,p} \in [\![0, |\mathcal{I}_c|]\!]$ est une variable entière représentant le nombre de fois où l'activité a de la classe c a été terminée avant le début de a^p, correspondant à $nc^\sigma(a, start^\sigma(a^p))$ défini dans la Section 3.

Les contraintes et critères sont encodés par :
minimize

$$\sum_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c\\ a\in\mathcal{A}_c}} \max(0, \mathsf{endOf}(\mathbf{itv}_{a,p}) - due_p) \quad (5)$$

$$\max_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c\\ a\in\mathcal{A}_c}} \mathsf{endOf}(\mathbf{itv}_{a,p}) \quad (6)$$

such that

$$\forall r \in \mathcal{R}, \sum_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c,\\ a\in\mathcal{A}_c}} \mathsf{pulse}(\mathbf{itv}_{a,p}, cons_{r,a}) \leq capa_r \quad (7)$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall (a,b) \in \mathcal{P}_c, \mathsf{endBeforeStart}(\mathbf{itv}_{a,p}, \mathbf{itv}_{b,p}) \quad (8)$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^C \quad \mathsf{span}(\mathbf{itv}_{a,p}, \{\mathbf{itv}_{b,p} | (a,b) \in \mathcal{H}_c\}) \quad (9)$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^A \quad \mathbf{n}_{a,p} = \sum_{q\in\mathcal{I}_c} (\mathsf{endOf}(\mathbf{itv}_{a,q}) \leq \mathsf{startOf}(\mathbf{itv}_{a,p})) \quad (10)$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^A \quad \mathsf{lengthOf}(\mathbf{itv}_{a,p}) = dur_a(\mathbf{n}_{a,p}) \quad (11)$$

Le critère lexicographique est formalisé dans les équations (5) et (6). Les contraintes cumulatives associées aux capacités des ressources sont exprimées via la fonction OPL pulse dans l'équation (7). Les contraintes (8) garantissent que les relations de précédence sont satisfaites (fonction OPL endBeforeStart). Les contraintes (9) imposent que les activités composées englobent leurs enfants. Les contraintes (10) codent le calcul de $nc^\sigma(a,)$ au moment où l'activité atomique a^p commence. La durée des activités atomiques est fixée via contraintes (11).

En plus des contraintes d'origine mentionnées ci-dessus, les contraintes de cassage de symétrie sont représentées par :

$$\forall c \in \mathcal{C}, \forall p, q \in \mathcal{I}_c \ s.t.\ p \prec_c q, \forall a \in \mathcal{A}_c^A, \mathsf{startBeforeStart}(\mathbf{itv}_{a,p}, \mathbf{itv}_{a,q}) \quad (12)$$

6 Expérimentations

6.1 Courbes d'apprentissage

Dans ce papier, nous choisissons une courbe d'apprentissage log-linéaire avec une durée finale constante [19, 21, 9]. Formellement, pour chaque classe $c \in \mathcal{C}$ et pour chaque activité atomique $a \in \mathcal{A}_c^A$, on considère : (a) la durée de la première exécution, notée dur_a^0 ; (b) la durée finale, notée dur_a^∞ ; (c) l'effet d'apprentissage $l_a \in]0, 1]$, qui impacte la pente de la courbe de durée.

Puis, pour tout $n \geq 0$, on a

$$dur_a(n) = dur_a^\infty + \left\lceil (dur_a^0 - dur_a^\infty) \cdot (n+1)^{\log_2 l_a} \right\rceil$$

Nous considérons que $dur_a^\infty = \frac{1}{2} dur_a^0$.

6.2 Environnement de test et instances

Nous avons testé notre approche sur deux types de jeux de données. Nous avons d'abord créé un jeu de données de 50 instances dans lequel chaque classe est une instance RCPSP du PSPLIB ([13]). Les caractéristiques des instances sont sélectionnées aléatoirement dans des plages spécifiques en fonction de la taille de l'instance, comme décrit dans le tableau 1. Les dates de livraison dans chaque classe sont calculées itérativement : due_{p_1} est égal à la date d'échéance maximale d de l'instance PSPLIB d'origine, puis $due_{p_{i+1}} = due_{p_i} + k_i \cdot d$, avec $k_i \in [0.3, 0.8]$. L'horizon temporel est égal à $h \cdot max_d$, avec h un coefficient égal à 40 ou à 70 (petites ou grandes instances) et max_d la plus grande date de livraison des instances PSPLIB impliquées. La capacité de chaque ressource est la somme des capacités de cette ressource de chaque instance PSPLIB impliquée [1]. Cet ensemble de données n'inclut aucune activité composée.

Le jeu de données *satellite* contient 24 instances. L'instance d'origine a été fournie par un fabricant de satellites. Comme décrit dans le tableau 1, elle couvre un horizon temporel de plus d'un an avec des activités durant de quelques heures à quelques jours. Dans notre recherche, nous décidons d'appliquer la loi log-linéaire dont nous faisons varier le taux d'apprentissage. Nous avons créé des instances plus grandes en augmentant le nombre de classes, de projets et l'horizon à partir

1. Chaque instance de PSPLIB dispose de 4 ressources.

Instances	Générée		Satellite	
	petite	grande	original	étendu
$\|\mathcal{C}\|$	$[\![2,4]\!]$	$[\![5,7]\!]$	3	6
$\|\mathcal{A}_c\|$	{30,60}	{60,90,120}	≤ 30	
$\|\mathcal{A}_c^C\|$	0		≤ 3	
$\|\mathcal{I}_c\|$	$[\![5,10]\!]$		≤ 5	≤ 14
$\|\mathcal{R}\|$	4		40	
$capa_r$	$\sum_{i_{LIB}} capa_r^{i_{LIB}}$		≤ 16	
l_a	$[0.45, 0.95]$		$[0.05, 0.95]$	0.85

TABLE 1 – Caractéristiques des instances par type de donnée : nombre de classes, activités atomiques, activités composées, projets, ressources, capacité des ressources et taux d'apprentissage.

Caractéristiques de l'instance				Retard			Makespan		
J.d. données	$\|\mathcal{C}\|$	$\sum_{c,p}\|\mathcal{A}_c\|$	l_a	NLM	LM	LSBM	NLM	LM	LSBM
petite	2	300	0.9	383	**372**	387	**231**	232	235
petite	2	840	0.75	1163	**800**	843	569	**498**	**498**
petite	4	1200	0.85	149	65	**52**	298	**271**	273
petite	4	1560	0.45	209	**0**	**0**	368	**257**	**257**
grande	5	3000	0.45	1225	456	**443**	532	**398**	**398**
grande	5	4380	0.85	6365	**5029**	5814	910	**865**	870
grande	7	4410	0.8	805	**479**	626	467	**425**	433
grande	7	5760	0.85	1769	1260	**1191**	586	**557**	566
original	3	301	0.9	**0***	**0**	**0***	1780*	**1694**	**1694***
original	3	301	0.95	**0***	**0**	**0***	1780*	**1738**	**1738***
étendu	6	637	0.85	-	2972	**1313**	-	**3752**	3759
étendu	6	913	0.85	3909	1407	**1267**	5726	5099	**5092**
étendu	6	1217	0.85	10831	6100	**4181**	7436	**6478**	6483
étendu	6	1526	0.85	22522	10547	**8780**	9255	8112	**8099**
étendu	6	1832	0.85	58045	**19162**	19715	11282	**9579**	10223

TABLE 2 – Retard et de makespan pour des instances représentatives. Les meilleurs résultats sont en gras. Un astérisque (*) indique une preuve d'optimalité.

des données d'origine tout en conservant les capacités des ressources. Les dates de livraison des projets supplémentaires ont été assignées aléatoirement.

Les tests ont été lancés avec IBM CP Optimizer 22.1.1, sur un processeur Intel(R) Xeon(R) E5-2660 v3 2,60-3,30 GHz avec 62 Go de RAM. Le temps maximum de calcul est de 15 minutes pour chaque exécution.

6.3 Résultats

Nous avons comparé trois variantes de modèle : No Learning Model (NLM), Learning Model (LM) et Learning Symmetry Breaking Model (LSBM). Dans NLM, il n'y a pas d'effet d'apprentissage, c'est-à-dire que la durée d'une activité a est toujours égale à dur_a^0, et il n'y a pas de cassage de symétrie. LM et LSBM correspondent respectivement au modèle sans et avec contraintes de cassage de symétrie, et avec présence d'effet d'apprentissage. Dans le tableau 2, nous présentons le retard et le makespan obtenus pour des instances représentatives de chaque ensemble de données.

Pour les petites instances générées, LM et LSBM satisfont toutes les dates de livraison pour 19 instances sur 25, alors que NLM seulement pour 12. Comme la durée des activités ne diminue pas pour NLM, les échéances sont, comme prévu, plus difficiles à respecter. Pour les grandes instances, un retard nul est atteint pour la moitié des instances par LM et LSBM. Lorsque le retard n'est pas nul, il n'y a pas de gagnant clair entre LM et LSBM. Concernant le makespan, comme prévu, NLM a généralement une valeur plus élevée. LSBM et LM renvoient des résultats très proches, avec un petit avantage pour LM.

Pour les instances satellites d'origine, toutes les configurations trouvent un retard nul. Les valeurs de makespan optimales sont trouvées par NLM et LSBM. Concernant les cinq instances satellite étendues, qui sont toutes présentées dans le Tableau 2, les valeurs de retard sont beaucoup plus élevées, et LSBM surpasse clairement LM pour toutes les instances sauf une. Les résultats sont beaucoup plus proches en ce qui concerne le makespan, ce qui pourrait s'expliquer par le fait que LM a du mal à ordonner les projets de manière à minimiser les retards.

Enfin, il semble que les contraintes de cassage de symétrie n'améliorent pas systématiquement les résultats. Cependant, ils peuvent faciliter la preuve d'optimalité, notamment pour les instances industrielles.

7 Conclusion

Dans cet article, nous avons i) défini formellement le RCPSP à haute multiplicité avec activités composées et effet d'apprentissage, ii) proposé un modèle de programmation par contraintes pour ce problème, et iii) prouvé l'existence de projets symétriques au sein de chaque classe. En outre, nous avons testé et comparé les performances du modèle PPC lors de l'intégration de contraintes de cassage de symétrie et/ou de courbes d'apprentissage, en utilisant un nouvel ensemble de données spécifiquement généré ainsi qu'un ensemble de données industrielles pour la production satellites.

Pour les recherches futures, il serait utile d'explorer les cas où l'effet d'apprentissage est partagé entre des activités similaires dans différentes classes. De plus, nous pourrions calculer des bornes inférieures et supérieures en relâchant le problème ou en utilisant une approche de type métaheuristique.

Références

[1] Jean-Pierre Amor : Scheduling programs with repetitive projects using composite learning curve approximations. *Project Management Journal*, 33(3):16–29, 2002.

[2] Christian Artigues, Sophie Demassey et Emmanuel Neron : *Resource-Constrained Project Scheduling : Models, Algorithms, Extensions and Applications*. ISTE/Wiley, 2008.

[3] Aleksander Bachman et Adam Janiak : Scheduling jobs with position-dependent processing times. *Journal of the Operational Research Society*, 55(3):257–264, 2004.

[4] Dirk Biskup : Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173–178, 1999.

[5] Dirk Biskup : A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188(2):315–329, 2008.

[6] Alessio Bonfietti, Michele Lombardi, Luca Benini et Michela Milano : A constraint based approach to cyclic rcpsp. *In Principles and Practice of Constraint Programming–CP 2011 : 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings 17*, pages 130–144. Springer, 2011.

[7] Steven J Edwards, Davaatseren Baatar, Kate Smith-Miles et Andreas T Ernst : Symmetry breaking of identical projects in the high-multiplicity rcpsp/max. *Journal of the Operational Research Society*, 72(8):1822–1843, 2021.

[8] Juan Diego García-Nieves, José Luis Ponz-Tienda, Angélica Ospina-Alvarado et Mateo Bonilla-Palacios : Multipurpose linear programming optimization model for repetitive activities scheduling in construction projects. *Automation in Construction*, 105:102799, 2019.

[9] Eric H Grosse, Christoph H Glock et Sebastian Müller : Production economics and the learning curve : A meta-analysis. *International Journal of Production Economics*, 170:401–412, 2015.

[10] Mariam Gómez Sánchez, Eduardo Lalla-Ruiz, Alejandro Fernández Gil, Carlos Castro et Stefan Voss : Resource-constrained multi-project scheduling problem : A survey. *European Journal of Operational Research*, 309(3):958–976, 2023.

[11] Sönke Hartmann et Dirk Briskorn : An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.*, 297(1):1–14, 2022.

[12] Alessandro Hill, Jordan Ticktin et Thomas WM Vossen : A computational study of constraint programming approaches for resource-constrained project scheduling with autonomous learning effects. *In Integration of Constraint Programming, Artificial Intelligence, and Operations Research : 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5–8, 2021, Proceedings 18*, pages 26–44. Springer, 2021.

[13] Rainer Kolisch et Arno Sprecher : Psplib - a project scheduling problem library : Or software - orsep operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216, 1997.

[14] Philippe Laborie, Jérôme Rogerie, Paul Shaw et Petr Vilím : Ibm ilog cp optimizer for scheduling : 20+ years of scheduling with constraints at ibm/ilog. *Constraints*, 23:210–250, 2018.

[15] Y.C. Law et J. Lee : Global constraints for integer and set value precedence. *In Proceedings of CP'04*, pages 362–376, 2004.

[16] Shujin Qin, Shixin Liu, Hanbin Kuang *et al.* : Piecewise linear model for multiskilled workforce scheduling problems considering learning effect and project quality. *Mathematical problems in Engineering*, 2016, 2016.

[17] Vincent Van Peteghem et Mario Vanhoucke : A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, 2010.

[18] Vincent Van Peteghem et Mario Vanhoucke : Influence of learning in resource-constrained project scheduling. *Computers & Industrial Engineering*, 87:569–579, 2015.

[19] Theodore P Wright : Factors affecting the cost of airplanes. *Journal of the aeronautical sciences*, 3(4):122–128, 1936.

[20] Muh-Cherng Wu et Shih-Hsiung Sun : A project scheduling and staff assignment model considering learning effect. *The International Journal of Advanced Manufacturing Technology*, 28:1190–1195, 2006.

[21] Louis E Yelle : The learning curve : Historical review and comprehensive survey. *Decision sciences*, 10(2):302–328, 1979.

[22] Yunqiang Yin, Dehua Xu, Kaibiao Sun et Hongxing Li : Some scheduling problems with general position-dependent and time-dependent learning effects. *Information Sciences*, 179(14):2416–2425, 2009.