J Heuristics (2015) 21:687–717
DOI 10.1007/s10732-015-9295-0

Une heuristique relax-and-fix avec fix-and-optimize appliquée aux problèmes de dimensionnement de lots à plusieurs niveaux

Claudio Fabiano Motta Toledo1 · Marcio da Silva Arantes1 ·
Marcelo Yukio Bressan Hossomi1 · Paulo Morelato France2 ·
Kerem Akartunali3

Reçu : 19 août 2014 / Révisé : 6 février 2015 / Accepté : 15 juin 2015 /
Publié en ligne : 2 juillet 2015

Résumé Dans cet article, nous proposons une heuristique simple mais efficace qui combine des idées d'heuristiques de construction et d'amélioration pour résoudre des problèmes de dimensionnement multi-niveaux. Une heuristique relax-and-fix est d'abord utilisée pour construire une solution initiale, et cela est encore amélioré en appliquant une heuristique fix-and-optimize. Nous introduisons également une nouvelle façon de définir les sous-problèmes d'entiers mixtes résolus par les deux heuristiques. L'efficacité de l'approche est évaluée en résolvant deux classes différentes de problèmes de dimensionnement de lots à plusieurs niveaux : le problème de dimensionnement de lots à capacités multiples avec retard et le problème d'ordonnancement de la production de récipients en verre à deux étapes (TGCPSP). Nous présentons des résultats de calcul étendus, y compris quatre ensembles de tests de la bibliothèque Multi-item Lot-Sizing with Backlogging, et des problèmes de test du monde réel définis pour le TGCPSP, où nous nous comparons aux méthodes de pointe de la littérature récente. Les résultats

B Claudio Fabiano Motta Tolède
claudio@icmc.usp.br

Marcio da Silva Arantes
marcio@icmc.usp.br

Marcelo Yukio Bressan Hossomi
marcelo.hossomi@usp.br

Paulo Morelato France
paulo.morelato@fct.unesp.br

Kerem Akartunali
kerem.akartunali@strath.ac.uk

1 Institut de mathématiques et d'informatique, Université de São Paulo, Avenida trabalhador são-carlense, 400, Centro, São Carlos, SP 13566-590, Brésil

2 Département de mathématiques et d'informatique, UNESP, R. Roberto Simonsen, 305, CP 266, Presidente Prudente, SP 19060-080, Brésil

3 Département des sciences de gestion, Université de Strathclyde, 40 George Street, Glasgow G1 1QE, Royaume-Uni

montrent que notre approche heuristique combinée est très efficace et compétitive, surpassant les méthodes de référence pour la plupart des problèmes de test.

Mots-clés Lot-sizing · Heuristique · Relax-and-fix · Fix-and-optimize · Backlogging

Classification des matières mathématiques 90C11

1. Introduction

Les systèmes de fabrication ont été étudiés analytiquement pendant plus d'un siècle pour obtenir de meilleurs rendements et rendements, puisque la fabrication était un élément clé, sinon "l'élément clé", du progrès économique des pays développés. L'année 2013 marque le centenaire de la célèbre formule de « quantité de commande économique », qui fut la première tentative d'optimisation des quantités de production dans des conditions très particulières. Depuis lors, de nombreux chercheurs opérationnels du milieu universitaire et de la pratique ont construit de nombreux modèles plus réalistes et proposé diverses méthodes de solution sophistiquées pour résoudre les problèmes de dimensionnement des lots/planification de la production évidents dans la pratique, où des décisions telles que la quantité à produire ou à stocker sont contraintes par diverses limitations naturelles. telles que les capacités et les temps d'installation.

Nous étudions deux classes de problèmes de dimensionnement de lots à plusieurs niveaux : le problème de dimensionnement de lots à capacités multiples (MLCLSP) avec retard et le problème d'ordonnancement de production de récipients en verre à deux étapes (TGCPSP). Le premier ensemble de problèmes, MLCLSP avec backlog, est particulièrement difficile d'un point de vue informatique, ce qui ressort également d'un certain nombre de nouveaux problèmes de dimensionnement des lots inclus dans MIPLIB (2010). De plus, la question théorique de la description complète de l'enveloppe convexe du problème mono-item avec backlog est restée ouverte pendant des décennies jusqu'à l'étude récente de Küçükyavuz et Pochet (2009), qui indique la sophistication impliquée dans ces problèmes. Enfin, dans un contexte de problème pratique, le backlog n'est jamais interdit car tous les fabricants échoueront tôt ou tard à satisfaire les demandes de leurs clients et leur backlog, et donc le problème du backlog présente un cas plus réaliste que celui sans. Le deuxième ensemble de problèmes, TGCPSP, représente un problème de planification et d'ordonnancement de la production à court terme dans le monde réel avec une première formulation de programmation en nombres entiers mixtes (MIP) proposée dans Almada-Lobo et al.

(2010). Les auteurs de Toledo et al. (2013) ont amélioré la formulation MIP précédente pour le TGCPSP, ont proposé un algorithme génétique hybride pour le résoudre et ont défini des ensembles de problèmes de test complexes. TGCPSP n'autorise pas le backlog comme MLCLSP et prend certaines caractéristiques spécifiques au problème telles que les coûts de perte de production et les temps et coûts de configuration dépendant de la séquence.

La littérature sur le dimensionnement des lots peut être divisée de manière plus appropriée en deux domaines principaux en raison de la nature des méthodes de résolution utilisées : (i) les méthodes exactes, et (ii) les méthodes heuristiques. Bien que même le problème à un seul élément capacitif soit N P-difficile (voir, par exemple, Florian et al. 1980) et que les attentes de solutions optimales diminuent à mesure que les problèmes deviennent plus réalistes, des méthodes exactes peuvent être très utiles pour comprendre les difficultés sous-jacentes à résoudre ces problèmes. problèmes. Ces méthodes incluent des inégalités valides (voir, par exemple, Barany et al. 1984 ; Miller et al. 2003), des reformulations étendues (voir, par exemple, Krarup et Bilde

1997 ; Eppen et Martin 1987 ; Rardin et Wolsey 1993), la relaxation lagrangienne (voir, par exemple, Billington et al. 1986) et la décomposition de Dantzig-Wolfe (voir, par exemple, Degraeve et Jans 2007). La plupart de ces études se concentrent principalement sur des problèmes simplistes (souvent à un seul élément), cependant, bon nombre des méthodes développées pourraient également être étendues à des problèmes réalistes. L'étude récente d' Akartunalı et Miller (2012) fournit plus d'informations sur les complexités apparentes dans les problèmes réalistes de dimensionnement des lots, et une discussion approfondie des techniques de programmation mathématique utilisées dans le domaine peut être trouvée dans Pochet et Wolsey (2006).

Bien que les méthodes exactes soient puissantes car elles fournissent une exposition de structures compliquées et une garantie sur la qualité de la solution, elles présentent un inconvénient important du point de vue informatique : même avec les ordinateurs rapides modernes et les progiciels d'optimisation de pointe, la résolution des problèmes industriels Les problèmes de dimensionnement des lots sont une tâche très compliquée (et souvent impossible). Pour compenser les lacunes informatiques des méthodes exactes et apporter des solutions en temps réel à des problèmes de taille industrielle, les méthodes heuristiques ont été largement utilisées dans ce domaine, depuis des cadres très simples jusqu'à des cadres très sophistiqués, voir, par exemple, Van Vyve et Pochet (2004), Wu et al. (2011), Kébé et al. (2012), Absi et al. (2013), Tolède et al. (2013), Baki et al. (2014). Nous renvoyons également le lecteur intéressé à Ball (2011) pour une revue de littérature récente sur l'heuristique de programmation mathématique générale. Notons enfin qu'un certain nombre de chercheurs ont proposé des frameworks utilisant des heuristiques ou des méta-heuristiques combinées à des techniques de programmation mathématique, l'inconvénient majeur des méthodes heuristiques étant de ne pas garantir la qualité des solutions. Les résultats récents incluent l'algorithme de colonie de fourmis couplé à des solutions MIP réduites pour le MLCLSP avec des heures supplémentaires proposées par Almeder (2010), l'heuristique de recuit simulé basée sur MIP et hybride pour le problème de dimensionnement stochastique proposé par Ramezanian et Saidi-Mehrabad (2013), et enfin l'algorithme génétique multi-population avec résolution de modèle LP pour le MLCLSP avec backlog proposé par Toledo et al. (2013).

La méthode décrite dans cet article combine deux heuristiques basées sur la programmation mathématique. Relax-and-fix (RF), une heuristique de construction qui résout séquentiellement les sous-problèmes MIP relâchés et fixe les variables binaires tout au long du processus pour l'accélérer, a été utilisée par un certain nombre de chercheurs pour des problèmes de dimensionnement par lot : Belvaux et Wolsey (2000) ont inclus une heuristique RF de base dans leur solveur sophistiqué de dimensionnement des lots, tandis que Stadtler (2003) a proposé une RF orientée dans le temps pour MLCLSP avec des résultats impressionnants. Les applications plus récentes de la RF dans la littérature sur le dimensionnement des lots incluent Federgruen et al. (2007) et Akartunalı et Miller (2009), où les premiers augmentent de manière itérative la taille du problème pour des solutions efficaces tandis que les seconds utilisent des inégalités (, S) pour des formulations plus fortes, surpassant les solutions trouvées par l'heuristique de Stadtler Stadtler (2003). Fix-and-optimize (FO), une heuristique d'amélioration basée sur MIP, est d'abord décrite dans Helber et Sahling (2010) pour résoudre le MLCLSP avec des délais et des coûts d'heures supplémentaires. Les auteurs proposent des décompositions orientées produits, ressources et processus pour le problème, qui définissent des sous-ensembles de variables binaires à optimiser. Seeanner et al. (2013) étendent ces idées de décomposition au problème de dimensionnement et d'ordonnancement des lots à plusieurs niveaux, où la recherche de décomposition de voisinage est combinée avec FO.

Nous proposons une méthode de résolution simple et facile à mettre en œuvre qui s'avère également efficace sur le plan informatique. Contrairement aux travaux récents d' Almeder (2010),

Ramezanian et Saidi-Mehrabad (2013), Toledo et al. (2013) combinant des méta-heuristiques complexes avec des heuristiques MIP, notre méthode combine deux heuristiques très simples basées sur MIP : RF avec fix-and-optimize (« RFFO », comme nous le verrons dans la suite de l'article). La simplicité est l'une des principales forces de la méthode proposée, permettant à tout chercheur ou praticien intéressé de l'implémenter facilement si nécessaire. De plus, nous proposons de nouvelles façons de construire des sous-problèmes à partir de l'approche classique de l'horizon temporel glissant, qui sont des composants importants de l'heuristique RF et FO, et étudions leur efficacité dans la pratique par des tests informatiques approfondis, y compris sur certaines instances MIPLIB 2010 (MIPLIB 2010) .

La méthode montre des performances de calcul impressionnantes pour la majorité des problèmes de test difficiles du MCLSP avec backlog, surpassant les méthodes de référence. De plus, alors que de nombreuses études telles que Helber et Sahling (2010) et Seean ner et al. (2013) explorent des structures de problèmes très spécifiques pour leur conception méthodologique, notre cadre RFFO proposé est conçu aussi générique que possible pour éviter de tirer parti d'une structure de problème spécifique et peut donc être étendu à d'autres problèmes si nécessaire, et afin de soutenir cela. argument, nous l'avons également appliqué à TGCPSP, où RFFO a pu trouver des résultats compétitifs par rapport au solveur par défaut IBM Ilog Cplex et à l'algorithme génétique hybride de Toledo et al. (2013).

Pour résumer, la méthode RFFO proposée a deux contributions principales : (i) Il s'agit d'un cadre simple combinant des heuristiques de construction et d'amélioration, qui renvoie également des résultats compétitifs dans des tests de calcul approfondis par rapport aux méthodes de référence de pointe de la récente littérature. (ii) La taille de la fenêtre de l'horizon glissant est orientée non seulement par colonne (c'est-à-dire la période dans le dimensionnement des lots, ce qui est la pratique courante) mais aussi par lignes (c'est-à-dire les familles de produits) ainsi que par une combinaison de colonnes et Lignes. Par conséquent, le procédé permet de faire défiler des fenêtres avec différentes combinaisons de colonnes et de lignes dans la représentation matricielle bidimensionnelle.

L'article est organisé de la manière suivante. Dans la section suivante, nous donnons une brève description mathématique des problèmes étudiés. Insecte. 3, nous définissons en détail notre cadre proposé, y compris une discussion sur de nouvelles façons de construire des sous-problèmes. Ensuite, nous présentons les résultats numériques de tests informatiques approfondis dans la Sect. 4 avec des comparaisons à deux méthodes de référence de la littérature récente, montrant l'efficacité de la méthodologie proposée. Enfin, nous concluons avec quelques orientations futures dans la Sect. 5.

2 Problèmes de dimensionnement de lots à plusieurs niveaux

Comme nous l'avons vu précédemment, l'approche RFFO développée dans cet article ne dépend pas de la structure du problème afin qu'elle puisse être adaptée à d'autres problèmes MIP. Dans cette section, nous présentons les formulations MIP des deux classes de problèmes de lot-sizing multi-niveaux. Tout d'abord, nous décrivons la formulation MIP de Toledo et al. (2013), basée sur la formulation d' Akartunalı et Miller (2009), pour le MLCLSP avec backlog, puis nous décrivons la formulation MIP pour le TGCPSP telle que présentée dans Toledo et al. (2013).

2.1 MLCLSP avec backlog

Dans cet article, nous considérons MLCLSP avec des familles de produits, c'est-à-dire plusieurs produits sont regroupés en familles en fonction de leurs similitudes. Étant donné que le backlog est naturel pratique dans les environnements de fabrication en raison de limitations de capacité, il est permis produits avec des exigences externes.

Paramètres

J	Nombre total de produits.
J	Nombre total de périodes.
M	Nombre total de machines/ressources.
F	Nombre total de familles.
amj	Temps nécessaire pour produire une unité de produit j sur la machine m.
Bjt	Limite supérieure pour la taille du lot du produit j à la période t.
bc j	Coût de retard du produit j.
cmt	Capacité totale de la machine m à la période t.
Djt	Demande primaire (externe) du produit j à la période t.
h	Coût de détention par unité de produit j en une période.
j p	1 si le produit j appartient à la famille f .
jf	Quantité de produit j nécessaire pour produire une unité de produit k.
rjk	Temps de préparation de la famille de produits f sur machine m.
stm	Ensemble des successeurs immédiats du produit j.
f δ(j) Δ	Ensemble des produits finis.

variables

x jt	Taille du lot du produit j à la période t.
w ft	Variable de configuration de la famille f à la période t.
je	Quantité détenue en stock du produit j pendant la période t.
jt bjt	Quantité de backlog du produit j dans la période t.

$$\text{Min} \sum_{j=1}^{J} \sum_{t=1}^{J} bc\,j \cdot bjt + h\,j \cdot je\,jt \tag{2.1}$$

Sujet à:

$$je\,jt-1 + bjt + x\,jt = je\,jt + bjt-1 + Djt \quad je\,jt-1 + x \qquad j, t|\ j \quad \Delta \tag{2.2}$$

$$jt = je\,jt + rjk \cdot xkt \quad k \quad \delta(j) \qquad j, t|\ j \quad / \Delta \tag{2.3}$$

$$\sum_{j=1}^{J} amj \cdot x\,jt + \sum_{f=1}^{F} stm\,f \cdot w\,ft \le Cmt \qquad m, t \tag{2.4}$$

$$x\,jt \le w\,ft\ x \quad \cdot Bjt \qquad j, f, t|p\ jf = 1\ w\,ft \tag{2.5}$$

$$jt,i\ jt, bjt \ge 0 \qquad \{0, 1\} \tag{2.6}$$

Les coûts d'inventaire et de backlog sont minimisés dans la fonction objectif (2.1).
Nous notons que nous n'incluons pas les coûts d'installation pour des raisons de simplicité du modèle (et

considérez plutôt les temps de configuration), mais le modèle proposé peut être facilement modifié pour les inclure. Les contraintes d'équilibre des flux (2.2) et (2.3) assurent la satisfaction des demandes externes et internes, respectivement, où la demande externe pour la fin les produits peuvent également être satisfaits grâce au backlog. Ici, nous notons que nous utilisons ces contraintes par souci de simplicité ainsi que par cohérence avec les formulations de Akartunali et Miller (2009), Toledo et al. (2013); cependant, les demandes externes aussi car le backlog peut également être inclus dans les niveaux supérieurs de l'échelon. Le grand seau capacités des machines intégrant à la fois des temps de traitement variables et des temps de réglage fixes dans chaque période sont définis par des contraintes (2.4), où nous supposons que chaque produit appartient à une seule famille de produits et il n'y a que des temps d'installation de famille de produits (plutôt que pour chaque produit). La contrainte (2.5) assure qu'un produit j ne peut pas être produit (c'est-à-dire x jt = 0) s'il n'y a pas de configuration pour sa famille de produits (c'est-à-dire w ft = 0). Le la limite supérieure de la taille du lot du produit j à la période t est représentée par le paramètre Bjt , qui peut être défini en utilisant les définitions suivantes de (2.7) et (2.8) (dans un mode à Akartunalı et Miller 2009).

$$Bjt = \min dj(1..T), \quad \frac{Cmt - stm\ f}{amj} \tag{2.7}$$

$$dj(t..T) = \sum_{u=t}^{J} Dju + rjk \cdot dk(t..T)\ k \quad \delta(j) \tag{2.8}$$

Notez que l'éq. (2.7) limite la taille du lot soit par la demande totale sur l'horizon (le premier terme à droite de l'équation) soit par la capacité maximale disponible pour la production (temps de réglage à soustraire de la capacité totale pour identifier le temps de production). Enfin, les domaines variables sont établis par des contraintes (2.6). Nous notez que cette formulation peut être étendue pour incorporer d'autres éléments d'un système de production, comme les heures supplémentaires, afin de le rendre plus réaliste. Cependant, nous le laissons tel quel pour une meilleure compréhension. Enfin, notons qu'en raison du backlog autorisé jusqu'à la dernière période de l'horizon, ce problème est toujours réalisable. Cependant, comme nous ont observé à partir de nos propres expériences de calcul ainsi que de nos discussions avec d'autres chercheurs, c'est une caractéristique qui rend ces problèmes calcul difficile lors de la tentative d'optimisation.

2.2 Problème d'ordonnancement de la production de récipients en verre en deux étapes

Ce problème trouve son origine dans la fabrication de récipients en verre, où un four fond la matière première dans la première étape du processus de production, et les machines de moulage sont utilisé dans la deuxième étape pour finaliser les contenants. Chez un fabricant typique de récipients en verre, la capacité journalière du four peut varier de 100 à 650 tonnes/jour. Nous renvoyons le lecteur intéressé à Toledo et al. (2013) pour plus de détails techniques sur la production processus. En bref, le TGCPSP est un problème de dimensionnement et d'ordonnancement des lots à deux niveaux avec machines parallèles et coûts et temps de configuration dépendant de la séquence. Différent de la problème abordé dans la section précédente, il ne permet pas le backlog.

Paramètres

C	: Capacité de fusion du four dans une période (en tonnes).
$\overline{N}_{ik}$	: Nombre maximum d'empreintes de moule de la machine k pour le produit i.
$\underline{n}_{ik}$	: Nombre minimal d'empreintes de moule de la machine k pour le produit i.
p_{ik}	: Quantité de produit i produite par empreinte de moule de la machine k au cours d'une période (en tonnes).
$salut$	: Coût de détention pour transporter une tonne de produit i dans la période suivante.
c_{ijk}	: Coût de configuration de la machine k du produit i au produit j, i = j.
s_{ijk}	: Capacité nécessaire pour régler la machine k du produit i au produit j, i = j (en tonnes).
d_{it}	: Demande du produit i à la fin de la période t (en tonnes).
ω	: Coût de pénalité par tonne de sous-utilisation du four.

Variables de décision

Y_{itk}	: 1 si le produit i est affecté à la machine k à la période t ; 0 sinon.
Q_t	: 1 si le four est actif en période t ; 0 sinon.
Z_{ijtk}	: 1 s'il y a un changement de configuration du produit i à la période t − 1 vers le produit j en période t sur la machine k ; 0 sinon.
N_{ick}	: Nombre d'empreintes actives du moule sur la machine k dédiée au produit i dans période t.
$I_{je\ l'ai}$	: Inventaire du produit i à la fin de la période t (en tonnes).
$\overline{je\ dt}$	: Capacité à vide du four à la période t (en tonnes).

$$\text{Min} \sum_{je,j,t,k} c_{ijk} \cdot Z_{ijtk} + \omega \cdot \sum_{t} \overline{je\ dt} + \sum_{il} salut \cdot ça \tag{2.9}$$

Sujet à:

$$I_{it} - I_{i,t-1} + d_{it} = \sum_{k} p_{ik} \cdot N_{ik} - \sum_{k,j} s_{jik} \cdot Z_{jik} \qquad (je, t) \tag{2.10}$$

$$\sum_{je,\,k} p_{ik} \cdot N_{itk} + I\ \overline{dt} = \overline{C} \cdot Q_t \qquad (t) \tag{2.11}$$

$$N_{itk} \le n_{ik} \cdot \overline{Y}_{itk} \qquad (i, t, k) \tag{2.12}$$

$$N_{itk} \ge n_{ik} \cdot \underline{Y}_{itk} \qquad (i, t, k) \tag{2.13}$$

$$\sum_{je} Y_{itk} \le 1 \qquad (t, k) \tag{2.14}$$

$$Q_t = \sum_{je} Y_{itk} \qquad (t, k) \tag{2.15}$$

$$Y_{itk} \ge \sum_{je} Y_{i(t+1)k} \qquad (t, k) | t < T \tag{2.16}$$

$$Y_{jtk} + Y_{i(t-1)k} \le Z_{ijtk} + 1 \qquad (i, j, t, k) \tag{2.17}$$

$$\sum_{j \in J} Z_{ijtk} \le Q_t \qquad (t, k) \qquad (2.18)$$

$$p_{ik} \cdot N_{ik} - \sum_{j} s_{jik} \cdot Z_{jitk} \ge 0 \qquad (i, t, k) \qquad (2.19)$$

$$I_{it}, I_{dt}, Q_t \ge 0, N_{itk} \in \mathbb{Z}^+, Y_{itk}, Z_{ijtk} \in \{0, 1\} \qquad (2.20)$$

Le problème vise à minimiser le coût total sur l'horizon de planification qui implique des coûts d'inventaire et d'installation ainsi que des pénalités pour les capacités d'inactivité des fours, comme indiqué dans (2.9). Les demandes de récipients en verre doivent être satisfaites sans retard comme le garantit (2.10), où le « temps d'installation » (c'est-à-dire le nombre de tonnes de produits perdus à partir de la capacité) est également pris en compte. La contrainte (2.11) impose la limite de capacité du four et garantit également que le temps d'inactivité est capturé si le four est utilisé. Le nombre maximum et minimum de sections de moule actives dans une machine donnée sont respectivement imposés par (2.12) et (2.13) , lorsqu'un produit est fabriqué. Chaque machine peut produire au plus un produit dans une période de temps (2.14), et le processus en deux étapes est synchronisé par (2.15), qui activerait le four si un produit est affecté à une machine. Si le four est désactivé à la période t, alors toutes les machines associées seront également inactives dans le reste de l'horizon, comme imposé par (2.16). Les contraintes (2.17) et (2.18) capturent les changements de produit et garantissent qu'ils ne peuvent se produire que lorsque le four est actif. La contrainte (2.19) impose que le « temps de préparation » utilisé ne soit pas supérieur à la quantité produite. Enfin, (2.20) définit les domaines variables.

3 Heuristique proposée : relax-and-fix avec fix-and-optimize

Nous décrivons ici les deux heuristiques et comment elles sont combinées pour résoudre les problèmes de dimensionnement de lots à plusieurs niveaux. Pour les deux heuristiques, considérons une matrice F × T où chaque entrée est une variable binaire w_{ft}. Le RF est une heuristique de construction qui définit une solution initiale en résolvant plusieurs petits problèmes à nombre entier mixte (MIP). Cela se fait en fixant ou en assouplissant la plupart des variables binaires, en n'obligeant que quelques-unes à être entières et en les optimisant. Nous appelons ce petit ensemble de variables entières une fenêtre dans le reste de l'article. Le pseudo-code de l'approche RF proposée dans cet article est résumé dans la Fig. 1.

Les entrées du RF sont l'ensemble des variables binaires (sol.w), le nombre de variables binaires (windowSize) à choisir, les critères de sélection pour choisir les variables (windowT ype), le taux de recouvrement des variables binaires à re- optimisé (overlap) et la limite de temps d'exécution (timeLimit). Initialement, toutes les variables binaires de la solution RF (sol.w) sont relaxées, ce qui signifie qu'elles peuvent prendre n'importe quelle valeur entre 0,0 et 1,0. Une fenêtre est définie comme un ensemble qui comprend une quantité fixe (windowSize) de variables (ligne 2, Fig. 1). Les variables à l'intérieur de la fenêtre sont forcées d'être entières dans l'ensemble wMIP (lignes 4 et 10, Fig. 1), tandis que les autres sont maintenues relaxées dans wL P (lignes 5 et 11, Fig. 1). Le MIP résultant est ensuite résolu (ligne 7, Fig. 1). Ensuite, un nouvel ensemble de variables (fenêtre) est défini, un sous-ensemble de variables entières est fixé (w fix) et un autre ensemble de variables entières et relâchées est optimisé (lignes 8 à 11, Fig. 1).

```
1: function RelaxAndFix(sol.w, windowSize, windowType, overlap, timeLimit)
2:     window ← initWindow( windowSize, windowType, sol.w )
3:     w_fix ← ∅
4:     w_MIP ← window
5:     w_LP ← sol.w − window
6:     while fixed solution not reached and elapsedTime < timeLimit do
7:         Solve( w_fix, w_MIP, w_LP )
8:         window ← moveWindow( overlap, windowSize )
9:         w_fix ← w_fix ∪ (w_MIP − window)
10:        w_MIP ← window
11:        w_LP ← w_LP − window
12:    end while
13:    sol.w ← w_fix
14:    return sol.w
15: end function
```

Fig. 1 Pseudocode de relax-and-fix

Le type de fenêtre (windowT ype) définit la manière dont les variables sont sélectionnées pour composer la fenêtre, ainsi que la manière dont elle est déplacée après chaque itération. Nous proposons trois types de fenêtres différents : rangée par rangée, dans laquelle la fenêtre se déplace le long des rangées ; par colonne, dans lequel la fenêtre se déplace le long des colonnes ; et par valeur, dans laquelle la fenêtre sélectionne les variables avec des valeurs relâchées les plus proches de 0,5. La fenêtre déplace les variables d'étape à chaque itération, avec step = |overlap windowSize|(ligne 8, Fig. 1), où chevauchement [0, 1] est le taux de chevauchement défini par l'utilisateur. La correction se produit pour toutes les variables qui quittent la fenêtre à l'itération suivante (ligne 9, Fig. 1), et le même nombre de variables relâchées est forcé à être entier. L'algorithme continue le traitement de cette manière jusqu'à ce que toutes les variables soient fixées. Nous notons que le processus RF serait bénéfique si le problème considéré, tel que le MLCLSP avec backlog, a toujours une solution réalisable ; il s'agit d'une propriété couramment exploitée par d'autres chercheurs utilisant également la RF [voir, par exemple, Stadtler (2003) et Akartunalı et Miller (2009)].

La figure 2 montre des exemples des trois types de fenêtres ainsi que la façon dont ils procèdent pour windowSize = 5 et step = 2. La figure 2a illustre la fenêtre par ligne, où les variables de wF1,T1 à wF1,T5 sont d'abord incluses dans la fenêtre à optimisé en tant que variables binaires. Après avoir trouvé la solution de ce MIP, les variables wF1,T1 et wF2,T2 sont fixes et laissent les fenêtres définies, tandis que les variables wF1,T6 et wF2,T1 sont forcées d'être des variables binaires. Cette procédure permet de ré-optimiser les variables wF3,T1 , wF4,T1 et wF5,T1 dans cette étape. Une idée similaire est appliquée dans la fenêtre par colonne comme illustré par la Fig. 2b.

Lors de l'utilisation de la fenêtre de valeur, le modèle est d'abord résolu avec toutes les variables relâchées afin que la solution relâchée soit obtenue pour l'évaluation. Ceci est illustré par la première matrice sur le côté gauche de la figure 2c, où les variables wF1,T2 , wF4,T2 , wF3,T3 , wF2,T4 et wF3,T4 sont les plus proches de 0,5. Ainsi, elles sont sélectionnées pour être optimisées en tant que variables binaires dans l'ensemble de fenêtres . Lorsque deux variables ou plus ont la même valeur, celles des premières colonnes sont préférées. Si la variable avec la même valeur se trouve dans la même colonne, les premières lignes seront sélectionnées en premier. Dans la matrice bidimensionnelle F × T définie pour les problèmes de lotissement, cela revient à choisir les produits des périodes antérieures (premières colonnes) et les produits finaux (premières lignes). Ces critères sont également utilisés pour décider comment corriger les variables, une fois le problème MIP résolu. Par exemple, les variables wF1,T2 et wF4,T2 sont choisies fixes (matrice du milieu, Fig. 2c), une fois qu'elles sont dans la première colo

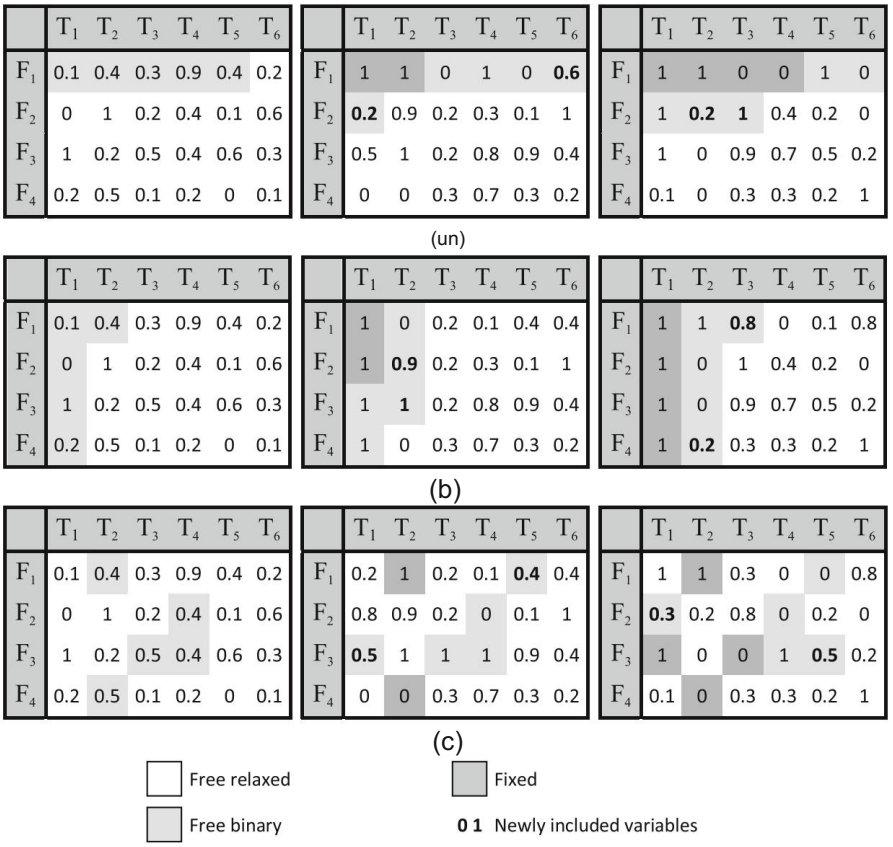


Fig. 2 Relax-and-fix différents types de fenêtres : une rangée ; b en colonne ; c en termes de valeur. Pour chaque fenêtre, trois itérations sont affichées en séquence. Taille de fenêtre = 5, taux de chevauchement = 60 %

parmi les variables de l' ensemble de fenêtres . Dans cette étape, les variables wF3,T1 et wF1,T5 sont maintenant incluses dans la fenêtre, et wF3,T3 , wF2,T4 et wF3,T4 restent à ré-optimiser.

Dans notre cadre, la solution construite par le RF sera utilisée comme solution pour initier le FO. Les étapes exécutées par FO sont très similaires à RF, où plusieurs problèmes MIP doivent être résolus. La figure 3 montre le pseudo-code FO. Une fenêtre glissante, couvrant le nombre de variables windowSize dans la matrice F × T (sol.w), est également définie pour FO et ces variables sont ajustées en binaire pour être optimisées par un solveur. Cependant, toutes les variables en dehors de la fenêtre sont maintenues fixes dans l'heuristique FO. A chaque itération, après avoir résolu le sous-problème MIP, la fenêtre est déplacée vers l'avant avec step = |overlap windowSize| (ligne 9, figure 3).

FO améliore les valeurs binaires en suivant les directions des lignes et des colonnes et, par conséquent, deux types de fenêtres sont définis. Le premier type de fenêtre combine la rangée, qui couvre la matrice le long des rangées (Fig. 4a) et la colonne, qui fait de même le long des colonnes (Fig. 4b) suivant la même idée définie pour RF. Cependant, FO applique les deux types de fenêtres lors de son exécution en ajustant windowT ype aux lignes 2 et 14 (Fig. 3).

```
1: function FixAndOptimize(sol.w, windowSize, overlap, timeLimit)
2:     windowType ← 0
3:     repeat
4:         window ← initWindow( windowSize, windowType )
5:         w_MIP ← window
6:         w_fix ← sol.w − window
7:         while window not reached end do
8:             Solve( w_fix, w_MIP )
9:             window ← moveWindow( overlap, windowSize)
10:            w_MIP ← window
11:            w_fix ← sol.w − window
12:            sol.w ← w_fix ∪ w_MIP
13:        end while
14:        windowType ← 1 − windowType
15:    until elapsedTime < timeLimit and windowType ≠ 0
16: end function
```

Fig. 3 Pseudocode de fix-and-optimize

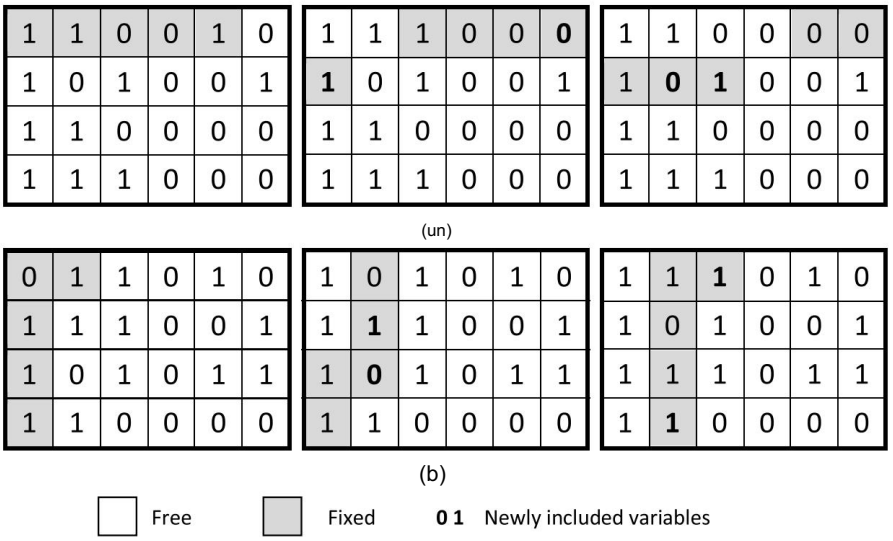


Fig. 4 Corriger et optimiser : a par ligne et b par colonne. Trois itérations de chaque direction sont affichées avec une taille de fenêtre = 5 et un taux de chevauchement = 60 %

Dans ce cas, windowT ype = 0 signifie appliquer par ligne et windowT ype = 1 par colonne.

Le deuxième type est une fenêtre carrée qui couvre la matrice le long des lignes (Fig. 5 a) et des colonnes (Fig. 5b) simultanément, composant un carré à travers la matrice. Notez que ce carré se chevauche des deux côtés avec le même taux de chevauchement, et le pas est arrondi au multiple entier le plus proche du côté du carré. Le carré se déplace le long des lignes et des colonnes pendant l'exécution de FO en fonction de la valeur windowT ype . Dans ce cas, windowT ype = 0 signifie déplacer le carré dans le sens des lignes et windowT ype = 1 le déplace dans le sens des colonnes.

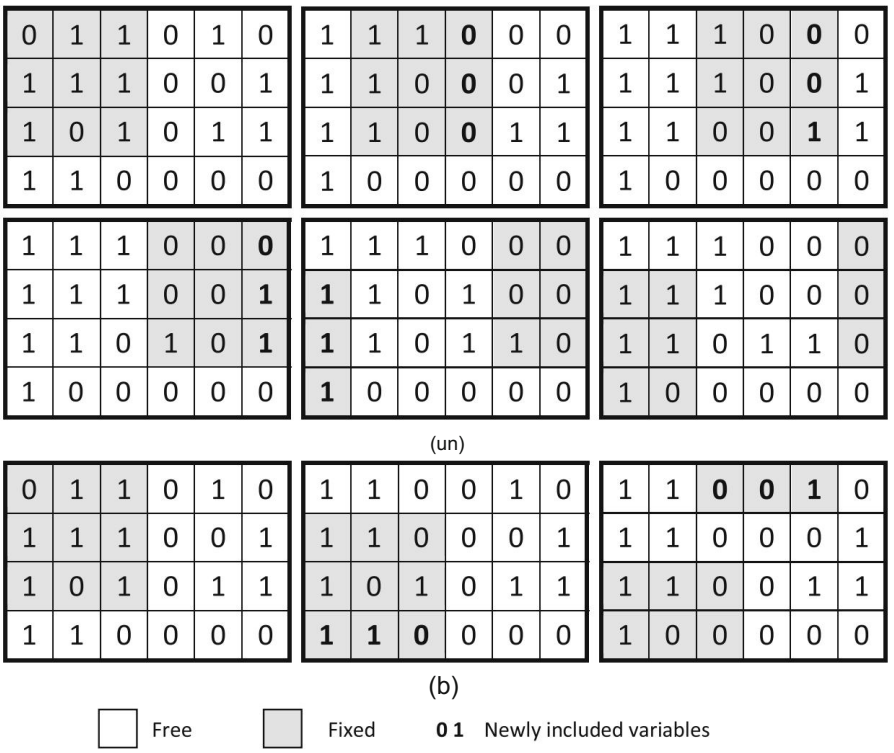


Fig. 5 Fix-and-Optimize avec fenêtre carrée suivant : a lignes et b colonnes. Six itérations de direction de colonne et trois de direction de ligne sont affichées, toutes deux avec une taille de fenêtre = 9 et un taux de chevauchement = 70 %

L'heuristique Relax-and-Fix with Fix-and-Optimize (RFFO) proposée est résumée sur la Fig. 6. Une fois l'exécution RF terminée, FO essaie d'améliorer cette solution initiale jusqu'à ce que la limite de temps soit atteinte. Si l'amélioration obtenue par une solution FO n'a pas satisfait une tolérance tol donnée, la taille de la fenêtre est augmentée de variables inc , un paramètre défini par l'utilisateur. Par la suite, les sous-problèmes MIP deviennent plus importants pour tenter de trouver de meilleures solutions. Lorsque vous utilisez la fenêtre carrée, l'incrément affectera la zone de la fenêtre, rendant sa croissance plus rapide lorsqu'elle est petite et plus lente lorsqu'elle est grande. La zone de la fenêtre sera arrondie à l'entier carré parfait le plus proche afin que la fenêtre carrée puisse être formée, mais la valeur arrondie ne sera pas utilisée dans les calculs d'incrément futurs.

4 Résultats de calcul

Les tests de calcul rapportés ici ont été exécutés sur un PC avec un processeur Intel i7, 2,6 GHz et 8 Go de RAM, et tous les modèles mathématiques ont été implémentés et résolus à l'aide de la bibliothèque appelable IBM ILOG Cplex 12.2. Nous avons d'abord implémenté RFFO pour résoudre le MLCLSP avec backlog, et par conséquent, nous discutons en détail du réglage des paramètres de la méthode ainsi que des résultats étendus obtenus pour le MLCLSP ensuite. Alors,

```
1: function RFFO(rfSize, rfOverlap, foSize, foOverlap, tol, inc, timeLimit)
2:     sol ← RelaxAndFix( rfSize, rfOverlap, timeLimit )
3:     prevCost ← sol.cost
4:     while timeElapsed < timeLimit do
5:         FixAndOptimize( sol, foSize, foOverlap, timeRemaining )
6:         if (sol.cost − prevCost)/prevCost < tol then
7:             foSize ← foSize + inc
8:         end if
9:     end while
10: end function
```

Fig. 6 Pseudocode de RFFO

l'application de RFFO au TGCPSP suivra avec les résultats de calcul obtenus pour les instances basées sur les paramètres obtenus auprès d'un fabricant de récipients en verre.

4.1 Résultats pour MLCLSP avec arriéré

Afin d'évaluer l'efficacité de notre méthode pour résoudre le MLCLSP, nous comparons le cadre RFFO à deux méthodes de pointe issues de la littérature récente : Aheur (Akartunalı et Miller 2009) et LugNP (Wu et al. 2011). Les codes exécutables de ces méthodes ont été gracieusement fournis par les auteurs respectifs afin que nous puissions exécuter toutes les méthodes sur le même ordinateur pour une comparaison équitable.

Nous avons utilisé les quatre ensembles de tests (SET1 à SET4) de Multi-LSB (2014) pour nos expériences de calcul, où pour tous les problèmes, le multi-item et le backlog sont autorisés.
Chacun de ces jeux de test comporte 30 instances avec 6 machines, 78 produits (répartis en 11 familles de produits) et 16 périodes, sauf que les instances SET2 ont 24 périodes. Un produit ne peut être composant que pour un seul produit dans la nomenclature (structure d'assemblage) définie pour ces instances. Le facteur d'utilisation des ressources est de 1,05 pour SET1 et SET2, 2,0 pour SET3 et 1,25 pour SET04. Ce facteur détermine la part de la capacité maximale totale des ressources qui est nécessaire pour répondre à toute la demande, ce qui signifie qu'un facteur supérieur à 1,0 implique que le retard dans la dernière période est nécessaire. Les coûts de backlog sont fixés à deux fois le coût de détention des stocks pour SET1 et SET2, et à 10 fois les coûts de détention des stocks pour SET3 et SET4. Ces caractéristiques rendent SET3 et SET4 plus difficiles à résoudre, comme l'ont également noté Akartunalı et Miller (2012).

Certaines des instances les plus difficiles de ce test sont récemment incluses dans la bibliothèque MIPLIB 2010 (MIPLIB 2010) en tant que « problèmes ouverts ». Nous faisons une remarque pratique que les facteurs d'utilisation très élevés de SET3 et SET4 rendent ces instances de test irréalistes en pratique. Cependant, les défis informatiques qu'ils offrent ainsi que le fait que d'autres chercheurs les ont utilisés rendent ces instances attrayantes, nous donnant une opportunité significative de référence.

Les trois méthodes ont été exécutées pendant 100 s dans SET1, 150 s dans SET2 et 300 s dans SET3 et SET4, pour rester cohérents avec les temps de calcul utilisés par Wu et al. (2011) pour LugNP, où les résultats Aheur et LugNP ont obtenu des écarts de dualité plus petits par rapport à ceux renvoyés par l'algorithme branch-and-cut (B&C) intégré dans Cplex.

Initialement, nous définissons les valeurs des paramètres RFFO de manière empirique, sur la base de tests préliminaires exécutés sur des instances choisies au hasard, où RF et FO s'appliquent en termes de valeur et

ligne/colonne, respectivement. RFFO a été défini avec une taille de fenêtre initiale de 140 variables pour les fenêtres RF et de 5 pour les fenêtres FO. Le taux de chevauchement pour RF a été fixé à 80 % et pour FO à 40 %. La tolérance d'amélioration FO utilisée a été fixée à 5 %, et le fait de ne pas obtenir une telle amélioration augmenterait la taille de la fenêtre de 40 variables. Ensuite, les effets de la modification de ces paramètres initiaux ont été évalués avec des tests de calcul effectués sur SET1 à SET4 de MLCLSP, comme le montre la Fig. 7. Les nouvelles valeurs de paramètres choisies parmi les valeurs initiales sont indiquées par des cercles, et elles nous permettent de personnaliser RFFO pour obtenir de meilleurs résultats pour l'ensemble d'instances de référence de MLCLSP. L'écart moyen pour toutes les instances de chaque ensemble est décrit et ces résultats sont comparés à LugNP et Aheur. Nous calculons l'écart (noté Dev(%)) pour toutes les instances, en utilisant l'Eq. (4.1), où Sol Ref fait référence à la solution « de référence », c'est-à-dire LugNP et Aheur.

$$\text{Dev}(\%) = \frac{\text{Sol}^{\text{RFFO}} - \text{Sol Réf.}}{\text{Sol Réf}} \cdot 100 \tag{4.1}$$

L'effet de la modification des valeurs des paramètres est négligeable pour les instances de SET1 et SET2, et il reste également limité pour les instances de SET3, alors que les instances de SET4 semblent en général assez sensibles aux changements de paramètres. Une grande fenêtre pour RF ne semble pas aussi efficace qu'une fenêtre avec 40 variables, où les résultats pour SET4 semblent fluctuer, comme le montrent les Fig. 7a, b. Le taux de chevauchement RF de 80 % semble être légèrement meilleur que les valeurs basses (Fig. 7c, d), et l'augmentation de la valeur préférable de 1 % pour la tolérance à l'amélioration FO aggrave les résultats par rapport à SET4 (Fig. 7e, f). Une grande taille de fenêtre FO initiale avec 40 variables donne une certaine amélioration pour tous les ensembles (Fig. 7g, h), avec une fluctuation significative pour les instances SET4. Une fois que FO ne parvient pas à améliorer les solutions de 1 %, un incrément de 10 variables semble fonctionner le mieux pour augmenter les sous-modèles (Fig. 7i, j). Enfin, le taux de recouvrement de 50 % produit des écarts moyens légèrement meilleurs que les autres valeurs (Fig. 7k, l). Nous présentons également l'amélioration des écarts après chaque changement de paramètre par rapport aux paramètres initiaux de la Fig. 8, ce qui indique que ces ajustements ont le plus grand impact sur les instances SET4.

Comme indiqué dans tous ces cas, le RFFO fonctionne mieux lorsque RF et FO commencent à résoudre les sous-problèmes MIP avec une taille de fenêtre RF et FO de 40 variables. RF est capable d'obtenir de meilleures solutions pour FO lorsque 80 % de ses variables peuvent être ré-optimisées (taux de chevauchement), tandis que FO fonctionne mieux en ré-optimisant 50 % de ses variables. Un tel comportement semble être lié au fait que FO est une heuristique d'amélioration et RF est une heuristique de construction, donc RF doit revoir plus souvent les décisions passées pour converger vers une solution réalisable. Nous notons également que nous avons expérimenté la sensibilité d'autres paramètres, mais constaté des différences insignifiantes dans de nombreux cas. Par exemple, RFFO atteint des valeurs moyennes inférieures à 0,1 % différentes lorsque la tolérance d'amélioration FO est définie sur 5 et 10 % dans SET4. Une évaluation exhaustive et plus fine de ces paramètres et l'expérimentation d'autres ensembles de tests pourraient potentiellement conduire RFFO à atteindre des performances « optimales ». Cependant, comme indiqué précédemment, notre objectif principal dans cet article est d'évaluer les stratégies concernant les choix de décompositions des sous-modèles MIP apparents dans RF et FO, dont nous discuterons ensuite.

Au total, six configurations de paramètres ont été définies afin de déterminer quelle combinaison de type de fenêtre offre les meilleures performances pour les sous-modèles MIP en RF et FO. Les configurations de paramètres n ° 1 à n ° 3 utilisent respectivement les fenêtres RF par ligne, par colonne et par valeur,

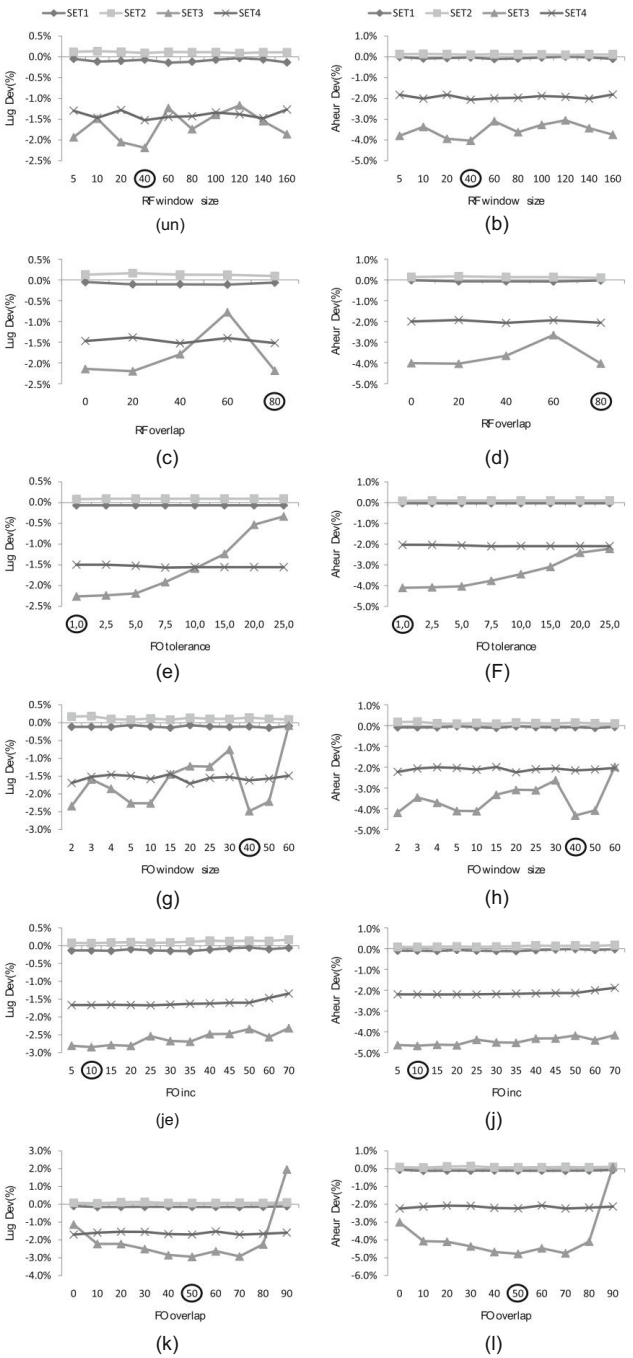
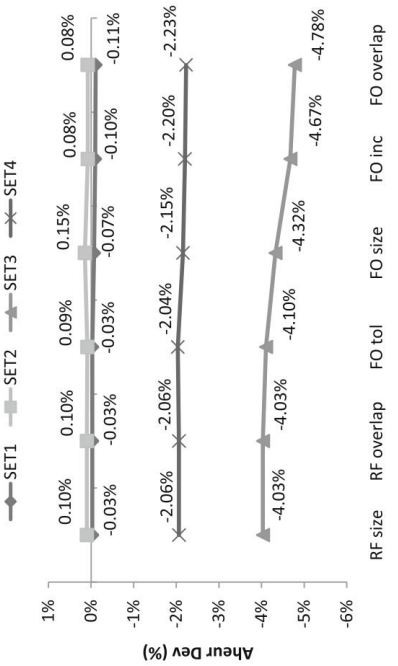


Fig. 7 Analyse des valeurs des paramètres : (a, b) taille de la fenêtre RF, (c, d) taux de chevauchement RF, (e, f) tolérance FO, (g, h) taille de la fenêtre FO, (i, j) incrément FO, (k, l) taux de chevauchement FO



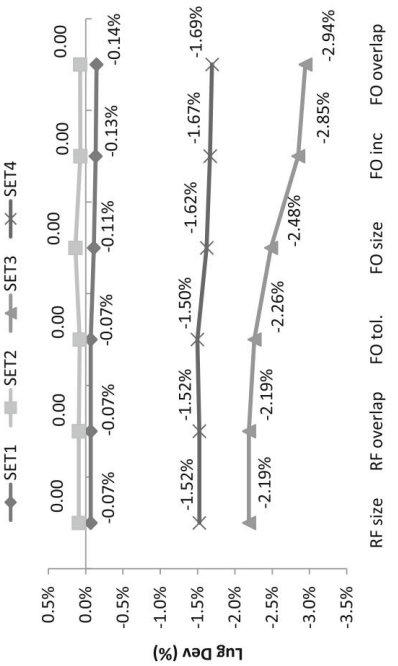


Tableau 1 Écart moyen et nombre de meilleures solutions dans tous les ensembles pour chaque combinaison de type de fenêtre

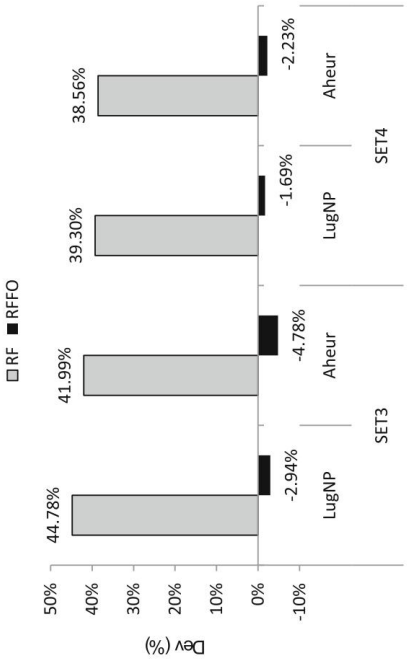
Installation	les fenêtres		De meilleures solutions			
	RF	FO	RFFO	Dessiner	LugNP	Dév. (%)
#1	Ligne	Ligne/colonne	61	38	21	−0,49
#2	Colonne	Ligne/colonne	68	41	11	−0,98
#3	Valeur	Ligne/colonne	68	37	15	−1,18
#4	Ligne	Carré	41	36	43	1.48
#5	Colonne	Carré	57	43	20	−0,55
#6	Valeur	Carré	49	32	39	0,20

combiné avec fenêtre ligne/colonne FO (premier type). Les configurations #4 à #6 utilisent toutes les fenêtres RF combiné maintenant avec la fenêtre carrée FO (deuxième type). Nous avons exécuté les six configurations pour toutes les instances de test, et les résultats ont été comparés à LugNP. Le tableau 1 résume les résultats obtenus par chaque configuration, montrant le nombre de meilleures solutions trouvées par RFFO, LugNp et les tirages. Il est considéré nul lorsque l'écart de solution les valeurs pour certaines instances sont inférieures à 0,01 %. La dernière colonne du tableau 1 indique la moyenne déviation des solutions RFFO par rapport à LugNP pour toutes les instances dans tous les ensembles.

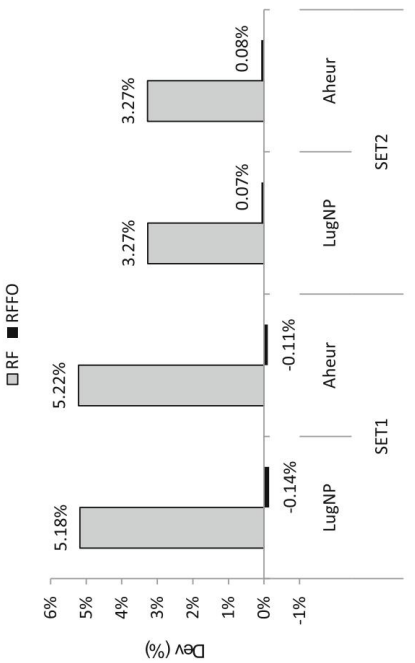
La configuration n ° 3, qui combinait la fenêtre par valeur RF et la ligne / colonne FO fenêtre, a montré la meilleure performance avec un écart moyen de −1,18 % également comme 68 victoires sur LugNP. La combinaison de RF en ligne et en colonne avec FO rangée/colonne sage a également renvoyé une amélioration de LugNP. Cependant, l'OP avec L'approche carrée semble être meilleure uniquement lorsqu'elle est combinée avec RF par colonne. Ainsi, pour le reste des tests de calcul, nous avons utilisé la configuration #3.

Ensuite, nous présentons la Fig. 9, résumant comment l'heuristique FO peut améliorer la solution construite par RF. Il montre l'écart moyen des solutions trouvées par RF et RFFO de LugNP et Aheur pour SET1 à SET4. RFFO a été exécuté avec les valeurs des paramètres et les types de fenêtres discutés précédemment. RF sur ses propres retours sur solutions moyennes avec moins de qualité par rapport aux méthodes de référence. Cependant, le FO améliore significativement ces solutions initiales pour l'ensemble de ces quatre jeux de test, en particulier pour SET3 et SET4.

Enfin, nous discutons des résultats en comparant l'approche RFFO proposée avec Aheur et LugNP, comme résumé dans le tableau 2. En ce qui concerne le pourcentage moyen d'amélioration, les résultats pour SET1 et SET2 ne sont pas nécessairement améliorés par RFFO, où il presque les mêmes performances que les méthodes de référence avec des écarts moyens d'environ 0,0 %. Cela se voit également par le nombre élevé de tirages, mais RFFO a été capable de retourner de meilleures solutions que LugNp et Aheur pour SET1 et SET2. Sur d'autre part, RFFO surpasse les deux méthodes de référence dans SET3 et SET4, réalisant respectivement plus de 4 et 2 % d'amélioration moyenne. C'est assez significatif, puisque ces ensembles incluent les instances les plus difficiles. Considérant la nombre de meilleures valeurs de solution finale, notre cadre proposé a surpassé les approches de référence pour plus de 20 instances sur 30 dans chacun des ensembles SET3 et SET4.



(q)



(un)

Tableau 2 Nombre de meilleures solutions et valeurs d'écart par ensemble

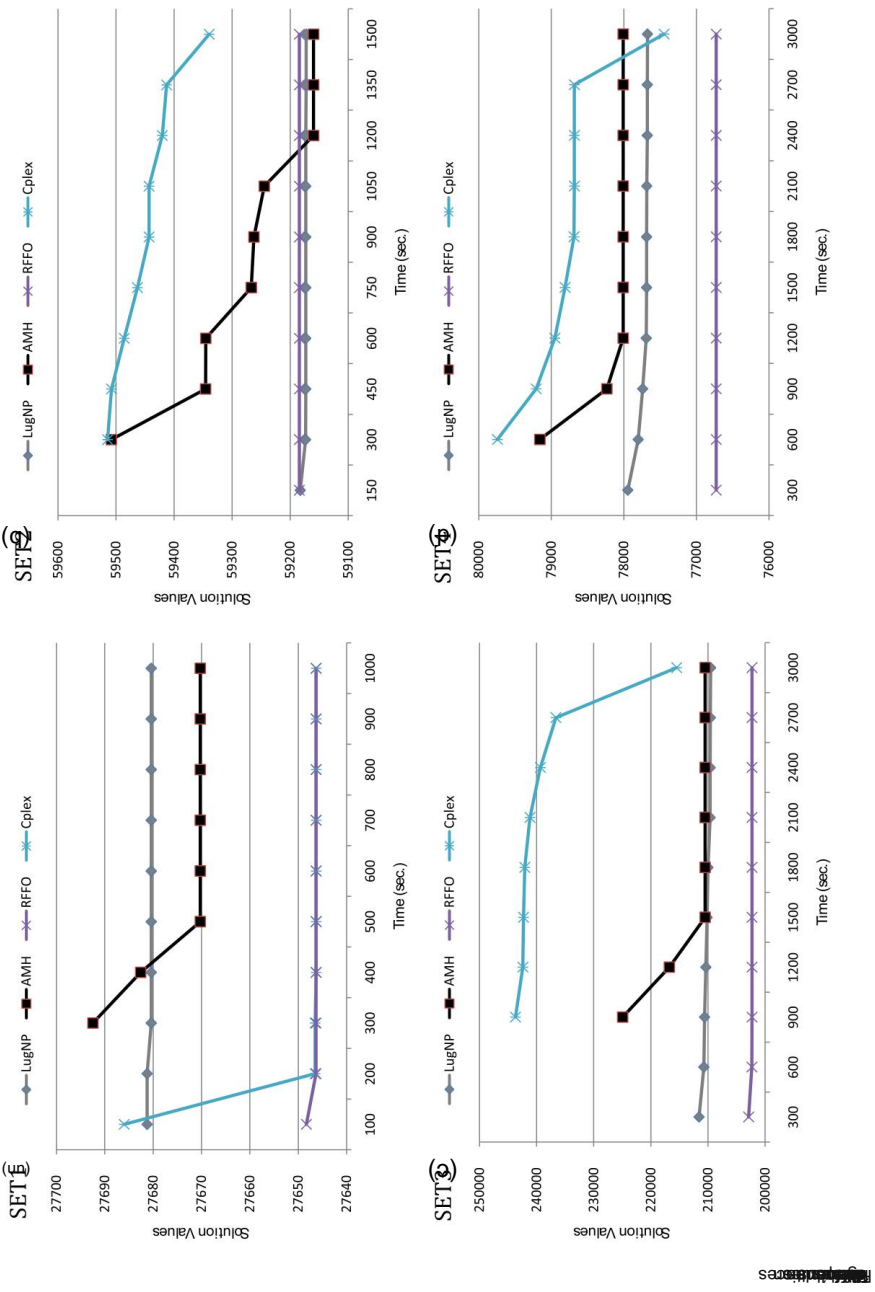
Ensemble	RFFO contre Aheur				RFFO contre LugNP			
	Tirage RFFO		Aheur	Dév. (%)	RFFO	Draw	LugNP	Dev. (%)
ENSEMBLE1	8	21	1	−0,11	dix	19	1	−0,14
ENSEMBLE2	9	15	6	0,08	dix	14	6	0,07
SET3	27	0	3	−4,78	22	0	8	−2,94
SET4	28	2	0	−2,23	26	4	0	−1,69

Ensuite, nous discutons des résultats détaillés pour chaque ensemble de données, où des tableaux avec des résultats détaillés pour toutes les instances sont fournies en annexe. Nous commençons par les tableaux annexes 4 et 5 montrant les résultats pour les instances SET1 et SET2, respectivement. Nous fournissons également la racine bornes inférieures des nœuds avec les inégalités (, S) d' Akartunalı et Miller (2012), représentées par XLP, pour indiquer la complexité de calcul des instances. On peut remarquer que les écarts sont faibles et la plupart sont nuls, avec plusieurs écarts entre 1,00 et 0,00 %, expliquant la faible amélioration moyenne. Par rapport à Aheur et LugNP, SET1 a l'écart le plus positif de 0,24 % et l'écart le plus négatif (meilleur amélioration) de −0,80 % à Aheur et de −1,83 % à LugNP, respectivement. Dans SET2, il convient de noter qu'il y a plus de valeurs négatives que positives, mais la les écarts positifs élevés pour SET2_23, SET2_1 et SET2_7 sont les principales raisons de l'écart moyen positif rapporté dans le tableau 2. Notre expérience de calcul est que les instances de SET1 et SET2 sont assez faciles à résoudre en général, et donc plus difficile à améliorer, probablement parce que les résultats de la littérature sont déjà très bons et proche de l'optimalité. Ceci peut être vérifié par les résultats et les comparaisons effectuées sur Akartunalı et Miller (2009) et Wu et al. (2011) pour soutenir la performance de Aheur et LugNP contre B&C.

Le tableau 6 en annexe montre les résultats pour SET3, où les résultats indiquent principalement des écarts négatifs atteignant −11,77 % depuis Aheur et −9,18 % depuis LugNP pour instances SET3_21 et SET3_16, respectivement. Une autre remarque importante à faire est que RFFO améliore significativement les solutions Aheur et LugNP (plus de 5 %) pour 15 et 9 instances, respectivement, alors que la pire performance pour RFFO est inférieure à 3,1 % par rapport à ces deux benchmarks (dans le cas de SET3_14, avec 1,14 % contre Aheur et 3,06 % contre LugNP dans SET3_29). Ceci est important puisque SET3 inclut les instances les plus difficiles à résoudre dans ces problèmes.

Le tableau 7 en annexe résume les résultats pour SET4. Les résultats sont en accord avec les résultats de SET3, indiquant des écarts négatifs notables (quoique légèrement moins significative par rapport à SET3). Il n'y a pas d'écart positif considérable avec 2 et 4 résultats considérés nuls, respectivement, contre Aheur et LugNP. Écarts négatifs atteignent −8,53 % contre Aheur et −5,66 % contre LugNP, alors que RFFO s'améliore Solutions Aheur et LugNP supérieures à 3 % pour 8 et 7 instances, respectivement.

Enfin, nous présentons dans la Fig. 10 les performances de calcul de différentes méthodes (y compris Cplex par défaut) avec des temps de calcul étendus, où la valeur moyenne des meilleures solutions trouvées par chaque méthode est donnée. Toutes les méthodes ont été exécutées pendant 10-plier les délais par rapport à nos délais d'origine, c'est-à-dire 1000 s pour les instances SET1



(Fig. 10a), 1500 s pour les instances SET02 (10(b)) et 3000 s pour les instances SET03 et SET04 (Fig. 10c, d), respectivement.

Nous avons omis certaines valeurs initiales (par exemple, les premières valeurs de Cplex et AMH pour SET02), car elles étaient hors de l'échelle des graphiques utilisés et auraient autrement détérioré la visualisation. Comme l'indiquent les graphiques, RFFO trouve rapidement des solutions de haute qualité et n'améliore que légèrement ces solutions pendant les périodes prolongées. De plus, les solutions RFFO sur les temps étendus ne sont surpassées que dans SET02, bien que légèrement, par LugNP et Aheur, où Aheur n'est capable de le faire qu'après 1200 s. RFFO est toujours meilleur en moyenne pour tous les autres ensembles montrant une performance plus stable par rapport aux autres méthodes, réalisant ces solutions très rapidement.

4.2 Résultats pour TGCPSP

L'efficacité de notre méthode est maintenant évaluée en résolvant le TGCPSP, où l'une des principales différences par rapport aux tests de calcul précédents est que la faisabilité des problèmes n'est pas garantie, ce qui était assuré avec le retard à la dernière période en cas de MLCLSP. Nous comparons nos résultats à ceux renvoyés par le solveur Cplex par défaut et l'algorithme génétique hybride (HGA) de Toledo et al. (2013), qui est une méthode spécialement conçue pour le TGCPSP. HGA exécute un algorithme génétique (GA) avec plusieurs populations, où leurs individus sont structurés hiérarchiquement en arbres, et intégrés avec le recuit simulé (SA) et la soi-disant heuristique de cavité (CV). SA est appliqué sur le meilleur individu trouvé par l'AG à chaque génération pour intensifier la recherche sur son voisinage. Le CV détermine le nombre de cavités du moule et, par conséquent, l'efficacité de la machine pendant le processus de production des récipients. HGA a exécuté 10 fois chaque problème de test en 1 h, et le même délai a été consacré par Cplex pour résoudre chaque problème de test en utilisant le modèle décrit dans la section. 2.2. Plus de détails sur l'algorithme et les paramètres utilisés peuvent être trouvés dans Toledo et al. (2013).

Les problèmes de test, basés sur des données fournies par des usines de conteneurs en verre du monde réel, sont composés de 150 problèmes artificiels et de 150 problèmes réels . L'ensemble artificiel est généré de manière aléatoire de manière académique, ne représentant pas nécessairement un scénario du monde réel et implique des instances de taille petite à modérée avec T {7, 14} jours, K {1, 2, 3, 4, 5} machines et N {5, 10, 20} produits par semaine. L'ensemble réel correspond aux scénarios réels qui se produisent dans les usines de récipients en verre, où le processus de production implique T {14, 28, 56} jours avec un nombre de produits d'environ 10 à 90 par semaine, et K {2, 3, 4, 5} machines par four. Pour chaque ensemble de problèmes, le type de solution renvoyé par le solveur Cplex dans un délai de 1 h est utilisé pour classer les problèmes de test comme Optimal (le solveur renvoie une solution optimale), Faisable (le solveur renvoie une solution réalisable sans optimalité garantie) et Inconnu (le solveur ne renvoie pas de solution réalisable). Le tableau 3 indique certaines caractéristiques de ces ensembles de tests ainsi que leurs sous-ensembles.

Nous rappelons que pour le problème MLCLSP discuté dans la section précédente, RFFO a optimisé les variables de configuration w ft en combinant RF avec une fenêtre par valeur et FO avec une fenêtre par ligne/colonne, ce qui signifie que FO recherche d'abord dans les lignes (familles f) puis par colonnes (périodes t) dans la structure de données bidimensionnelle de

Tableau 3 Problème TGCPSP instances

Taper	Statut	# Instances	Moyenne Processeur(s)	Écart (%)
Problèmes de test artificiels				
O0	Optimal	27	1.8	0.0
O1	Optimal	27	419	0.0
F0	Réalisable	24	3600	3.2
F1	Réalisable	24	3600	9.8
F2	Réalisable	25	3600	17.1
tu	Inconnu	23	3600	–
Problèmes de test réels				
O0	Optimal	3	1294	0.0
F0	Réalisable	20	3600	9.1
F1	Réalisable	21	3600	20.6
tu	Inconnu	106	3600	–

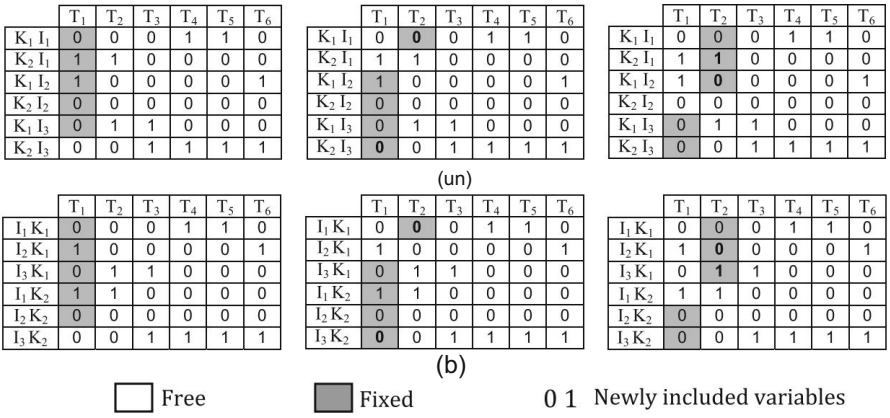


Fig. 11 Corriger et optimiser : a machine-produit-période et b produit-machine-période. Trois itérations de chaque direction est affichée avec une taille de fenêtre = 5 et un taux de chevauchement = 60 %

w pi . Étant donné que les variables de configuration Yitk du TGCPSP sont tridimensionnelles, un autre une élaboration est nécessaire pour le cadre RFFO. Cela ne pose pas de problème dans exécuter RF avec une fenêtre de valeur, mais une stratégie pour exécuter FO doit être adaptée de la fenêtre rangée/colonne précédente. Sur la base de nos tests préliminaires avec diverses options, nous avons conclu d'exécuter FO en suivant d'abord la séquence produit machine-période puis machine-produit-période comme illustré par la Fig. 11.

Sur la figure 11a, la fenêtre comprend des variables sélectionnant les index par les machines Ki en premier suivi pour les postes Ii (produits) et les périodes Ti . Après avoir optimisé de cette façon, le fenêtre dans cette structure de données tridimensionnelle sélectionne les index de variables sur la figure 11b par items suivis des machines et des périodes.

Nous avons exécuté RFFO pour chaque instance de test dans le même délai de 1 h, où nous utilisons les paramètres initiaux présentés dans la section précédente. D'abord

Au total, pour tester la flexibilité de notre approche, nous avons exécuté RFFO sur toutes les instances « Inconnues » identifiées par le Cplex par défaut, qui correspondent respectivement à 15,3 et 70,7 % des problèmes de test artificiels et réels. Dans la même limite d'une heure, RFFO a pu trouver des solutions pour 56,5 et 20,8 % de ces instances inconnues, respectivement, atteignant des taux d'échec de 6,67 et 56 % dans les ensembles globaux de problèmes artificiels et réels, respectivement. Bien que l'amélioration par rapport à Cplex pour les instances artificielles inconnues soit significative, les instances réelles inconnues présentent toujours un défi, notamment en raison de leurs tailles immenses et du nombre élevé de variables binaires (en moyenne 3 329 pour les problèmes réels). De plus, l'implication de variables générales entières complique considérablement ces problèmes et ils n'ont pas été spécifiquement traités dans notre cadre RFFO afin de préserver la structure simple présentée précédemment pour les problèmes binaires mixtes. De plus, les performances peuvent également être affectées par le fait que l'accessibilité d'une solution réalisable est moins simple par rapport à MLCLSP avec backlogging, où la solution simple de production nulle et de backlogging de la demande totale jusqu'à la dernière période est toujours réalisable (mais coûteuse). Nous étudions actuellement ces domaines de manière plus approfondie selon les besoins et prévoyons de relever ces défis dans nos futurs résultats de recherche.

Afin d'évaluer la qualité de la solution que RFFO peut atteindre pour TGCPSP, nous avons ensuite exécuté RFFO pour toutes les instances de test qui ne sont pas "inconnues". En utilisant la même limite de temps de 1 h que Cplex et HGA, nous présentons nos résultats de calcul exécutant RFFO avec les valeurs initiales des paramètres (RFFOd) et avec le meilleur paramétrage pour les instances de référence du MLCLSP (RFFO). Ainsi, l'idée ici est d'évaluer les performances de RFFO en cours d'exécution avec les valeurs initiales obtenues empiriquement ainsi qu'avec les valeurs de paramètres personnalisées pour résoudre les instances MLCLSP.

Dans la Fig. 12, nous comparons toutes les méthodes pour les cinq sous-ensembles de problèmes de test artificiels impliquant 127 instances, où l'écart (%) est calculé par Gap(%) = (Upper Bound − Lower Bound)/(Upper Bound) en utilisant la meilleure solution obtenue

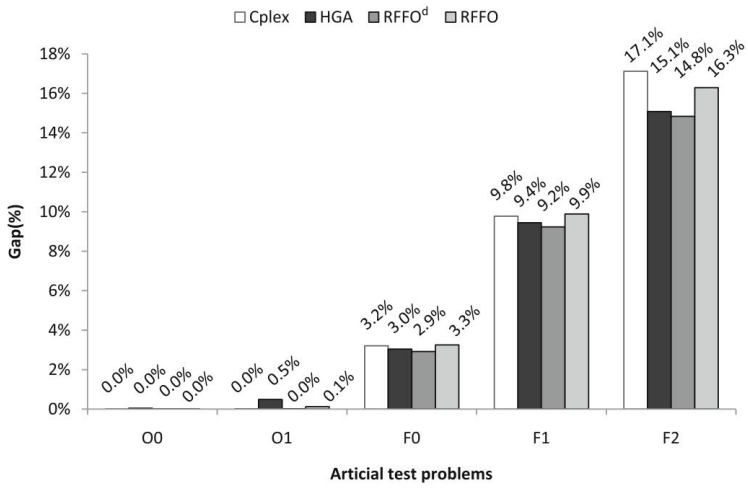


Fig. 12 Comparaisons GAP(%) pour les problèmes de test artificiels

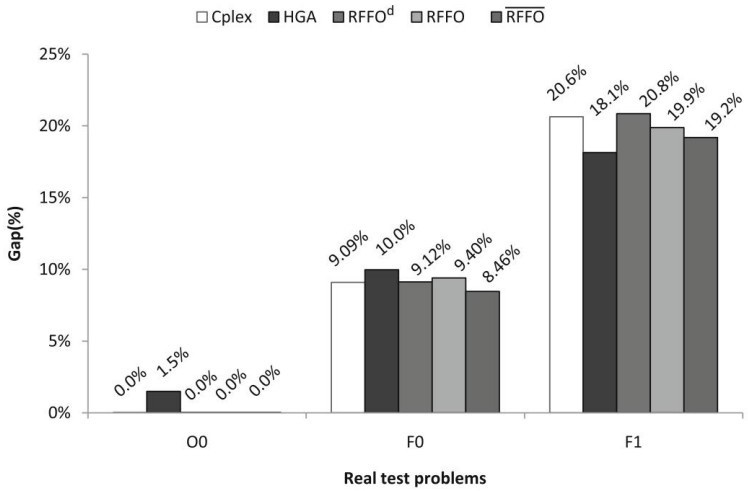


Fig. 13 Comparaisons GAP(%) pour des problèmes de test réels

par chaque méthode pour Upper Bound et la borne inférieure renvoyée par l'algorithme branch & cut de Cplex pour Lower Bound.

Comme l'indique la Fig. 12 , RFFOd a pu trouver des solutions optimales dans l'ensemble d'instances, dont les valeurs optimales ont été renvoyées par Cplex (O0 et O1), alors que HGA et RFFO n'ont pas pu renvoyer de solutions optimales pour tous les problèmes de test appartenant à l'ensemble O1 avec écarts moyens de 0,5 et 0,1 %, respectivement. Pour les trois autres sous-ensembles de problèmes, où seules des solutions réalisables ont été renvoyées par Cplex, toutes les méthodes avaient des performances similaires en ce qui concerne la qualité de la solution, mais RFFOd a réussi à obtenir systématiquement la valeur d'écart moyenne la plus faible. Ceci est prometteur, en particulier si l'on considère que l'HGA est une méthode conçue sur mesure pour ces problèmes.

Dans la Fig. 13, nous présentons la même évaluation pour trois sous-ensembles de problèmes de test réels impliquant 44 instances. Sur la base des résultats obtenus par RFFOd pour les problèmes de test artificiels, nous avons également implémenté une version légèrement modifiée de , nommé RFFOd RFFO, où l'ordre d'optimisation des variables dans le FO est modifié. Dans ce cas, les variables de paramétrage Yitk sont optimisées par FO en suivant d'abord la séquence produit-machine-période puis la séquence produit-machine-période. Semblables aux instances artificielles, les trois versions RFFO ont pu trouver des solutions optimales pour les instances réelles, pour lesquelles Cplex pouvait retourner leurs valeurs optimales, tandis que HGA n'a pas réussi à trouver la solution optimale pour toutes les instances de cet ensemble O0 . Pour les deux ensembles possibles, RFFOd et RFFO renvoient presque la même valeur d'écart moyenne que Cplex, tandis que la version modifiée RFFO surpasse Cplex et HGA pour l'ensemble F0 et surpasse Cplex pour l'ensemble F1. En revanche, HGA génère en moyenne de meilleures solutions que RFFO pour l'ensemble F1, mais nos résultats sont compétitifs, considérant que HGA est spécifiquement conçu pour ces problèmes. Comme l'expérimentation avec RFFO l'a indiqué, nous remarquons que d'autres changements dans les paramètres RFFO peuvent potentiellement améliorer ses performances comme cela a été fait pour l'ensemble de référence de MLCLSP. Cependant, en tant que cadre général, cela fonctionne efficacement. Enfin, notons que RFFO est actuellement

conçu pour optimiser uniquement les variables binaires, mais un cadre RFFO plus sophistiqué pourrait gérer plus efficacement les variables entières générales du TGCPSP, ce que nous prévoyons de traiter dans un avenir proche.

5. Conclusion

Une méthode hybride, RFFO, a été proposée en combinant deux heuristiques bien connues, RF et FO. Une combinaison simple est proposée, où RF est utilisé pour construire une solution initiale qui est encore améliorée par le FO en temps de calcul disponible. Le RFFO est appliqué au MLCLSP avec un backlog et un problème de planification de la production de conteneurs en verre en deux étapes (TGCPSP). En utilisant divers problèmes de test disponibles dans la littérature, la méthode proposée a été comparée aux méthodes de pointe de la littérature : Aheur d' Akartunalı et Miller (2009) et LugNP de Wu et al. (2011) pour MLCLSP, qui sont aussi des heuristiques basées sur la programmation mathématique, et HGA de Toledo et al. (2013) pour TGCPSP, qui est un algorithme génétique.

Dans l'approche proposée, les deux heuristiques utilisent la programmation mathématique pour résoudre des sous-problèmes d'entiers mixtes définis par un certain nombre de variables binaires. Ces variables définissent une fenêtre qui se déplace dans la matrice de solution en utilisant différentes orientations. Aussi, le nombre de variables binaires sous la fenêtre est augmenté si la solution n'est pas suffisamment améliorée en une seule exécution du FO.

Différentes stratégies pour traverser la matrice en optimisant les variables binaires ont été proposées et testées, où la meilleure rapportée combine un RF en valeur avec un FO en ligne/colonne. Ainsi, les résultats rapportés indiquent de meilleures solutions initiales renvoyées par RF lorsque l'optimisation se concentre sur des variables relaxées plus proches de 0,5, suivies par FO travaillant mieux en essayant d'optimiser séparément les variables orientées lignes et colonnes. La meilleure configuration trouvée a permis à la méthode proposée de surpasser de manière significative les approches de référence dans deux des quatre ensembles de tests de MLCLSP. Cependant, il a également été en mesure de retourner des résultats compétitifs dans les deux autres sets. Plus important encore, les résultats indiquent une meilleure performance de RFFO dans les instances de test plus complexes de SET3 et SET4. De même, trois configurations de RFFO ont également été en mesure de renvoyer des résultats compétitifs pour le problème plus sophistiqué de TGCPSP, surpassant régulièrement le Cplex par défaut et obtenant des résultats meilleurs ou comparables avec HGA, qui est une méthode personnalisée rapide et efficace pour ces problèmes. Nous pensons que RFFO est globalement une méthode efficace pour résoudre des problèmes de dimensionnement de lots avec des caractéristiques variables.

Comme travaux futurs, nous prévoyons de mener des tests informatiques approfondis sur différentes combinaisons de valeurs de paramètres. Cela donnerait un meilleur aperçu de la sensibilité de différentes tailles pour les MIP résolus par RF et FO, ainsi que des valeurs de taux de chevauchement différentes. Nous étudions également actuellement la combinaison RF et FO avec d'autres méta-heuristiques. Par exemple, RF pourrait être utilisé pour fournir différentes solutions initiales si un critère aléatoire est incorporé dans la stratégie de valeur. De plus, l'heuristique FO proposée pourrait être appliquée comme recherche locale pour améliorer les meilleures solutions trouvées par d'autres méta-heuristiques. Un autre domaine à étudier est l'amélioration potentielle de la méthode si elle exploitait la structure spécifique du problème. Nous prévoyons d'étudier cela pour différents contextes, par exemple pour les heures supplémentaires.

Enfin, nous notons que la conception proposée dans cet article est générique et indépendante du problème. Pour vérifier sa robustesse, nous envisageons d'étendre cette approche à des Les problèmes MIP qui ont naturellement une structure de prise de décision séquentielle, y compris problèmes avec les variables entières générales. Dans ce cas, il est en particulier de notre intérêt d'étudier les problèmes MIP où l'heuristique RF pourrait échouer à déterminer une valeur initiale. solution. Ainsi, une autre heuristique de construction pourrait être appliquée en tirant parti de la solution partielle apportée par la RF proposée. Nous étudions actuellement certains problèmes d'ordonnancement des équipages avec ce cadre.

Remerciements Nous tenons à remercier le Dr Tao Wu pour nous avoir fourni le code de l'heuristique LugNP utilisée dans les comparaisons. Les travaux menés par les trois premiers auteurs ont été soutenus par la Fundação de Amparo e Pesquisa do Estado de São Paulo (FAPESP) Projets 2010/10133-0, 2011/15534-5 et 2011/15581-3 et Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) Projets 483474/2013-4 et 312967/2014-4.

6 Annexe

Voir les tableaux 4, 5, 6 et 7.

Tableau 4 Comparaison pour les instances SET1 (limite de temps = 100 s)

ENSEMBLE1	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
1	17 888	22 382,5	22 460,7	22 382,1	0	−0,35
2	23 534	27 584,8	27 584,8	27 584,6	0	0
3	21 227	25 187,3	25 187,3	25 246,6	0,24	0,24
4	22 232	26 334,7	26 334,7	26 334,7	0	0
5	21 446	25 145,5	25 145,5	25 145,8	0	0
6	22 974	26 667,4	26 770,8	26 667,5	0	−0,39
7	20 360	24 123,8	24 123,8	24 124,2	0	0
8	25 582	29 640,4	29 640,4	29 639,8	0	0
9	16 321	20 971,2	21 362,7	20 971,0	0	−1,83
dix	17 998	22 645,8	22 647,5	22 562,8	−0,37	−0,37
11	11 080	12 955,6	12 955,6	12 955,3	0	0
12	24 721	26 831,3	26 831,3	26 831,1	0	0
13	20 782	23 127,8	23 127,8	23 128,5	0	0
14	22 264	25 035,8	25 035,8	25 036,0	0	0
15	12 401	14 118,1	14 118,1	14 117,9	0	0
16	15 122	17 540,2	17 400,1	17 400,1	−0,80	0
17	20 468	23 007,5	23 007,5	22 996,2	−0,05	−0,05
18	11 075	12 973,8	12 973,8	12 973,8	0	0
19	13 276	16 502,9	16 502,9	16 349,2	−0,93	−0,93

Tableau 4 suite

ENSEMBLE1	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
20	14 101	17 158,6	17 158,6	17 158,7	0	0
21	10 159	12 421,2	12 421,2	12 421,1	0	0
22	38 040	40 158,3	40 188,7	40 158,4	0	−0,08
23	29 331	30 605,7	30 605,7	30 605,5	0	0
24	28 858	32 190,4	32 145,5	32 007,2	−0,57	−0,43
25	51 371	52 989,2	52 959,9	52 960,3	−0,05	0
26	39 379	41 221,5	41 221,5	41 221,0	0	0
27	40 838	43 319,7	43 319,7	43 289,6	−0,07	−0,07
28	39 846	40 993,5	41 019,8	40 993,5	0	−0,06
29	23 155	25 492,6	25 322,3	25 322,0	−0,67	0
30	68 989	70 863,7	70 863,7	70 863,7	0	0

Tableau 5 Comparaison pour les instances SET2 (limite de temps = 150 s)

ENSEMBLE2	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
1	46 116	52 050,7	52 050,7	52 339,4	0,55	0,55
2	47 780	53 863,4	53 713,4	53 713,0	−0,28	0
3	40 551	46 894,5	47 053,2	46 893,3	0	−0,34
4	36 347	43 009,8	42 977,1	43 063,1	0,12	0,20
5	45 395	51 757,6	51 757,6	51 768,8	0,02	0,02
6	45 902	51 858,1	51 858,1	51 858,4	0	0
7	52 825	58 153,8	58 153,8	58 425,2	0,47	0,47
8	48 033	54 396,2	54 449,6	54 182,9	−0,39	−0,49
9	37 553	43 737,8	43 737,8	43 690,0	−0,11	−0,11
dix	38 751	45 278,8	45 278,8	45 305,8	0,06	0,06
11	65 210	68 488,8	68 646,4	68 487,8	0	−0,23
12	62 792	66 561,9	66 474,5	66 475,4	−0,13	0
13	34 778	39 120,3	39 082,7	38 852,7	−0,68	−0,59
14	62 907	66 373,7	66 383,2	66 325,1	−0,07	−0,09
15	59 079	61 574,1	61 574,1	61 574,0	0	0
16	75 682	79 364,8	79 385,0	79 363,9	0	−0,03
17	36 809	41 298,6	41 282,4	41 192,6	−0,26	−0,22
18	77 873	81 561,8	81 562,9	81 562,5	0	0
19	54 981	58 426,1	58 426,1	58 425,4	0	0
20	119 568	122 827,6	122 827,6	122 829,0	0	0

Tableau 5 suite

ENSEMBLE2	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
21	22 281	24 013,2	24 014,2	24 013,3	0	0
22	51 279	52 887,1	52 887,1	52 886,8	0	0
23	29 793	32 618,2	32 708,8	33 713,9	3.36	3.07
24	65 891	68 640,6	68 575,1	68 574,8	−0,10	0
25	75 627	78 064,3	78 088,2	78 064,2	0	−0,03
26	60 952	63 275,2	63 285,6	63 273,2	0	−0,02
27	53 016	54 794,1	54 794,1	54 793,9	0	0
28	44 545	46 607,9	46 607,9	46 607,6	0	0
29	93 631	96 278,0	96 157,4	96 152,0	−0,13	0
30	68 324	71 408,0	71 408,0	71 408,7	0	0

Tableau 6 Comparaison pour les instances SET3 (limite de temps = 300 s)

SET3	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
1	65 668	188 294,0	189 400,6	179 554,0	−4,64	−5,20
2	82 342	216 700,4	217 283,4	216 401,0	−0,14	−0,41
3	74 209	216 517,4	207 362,6	198 215,0	−8.45	−4,41
4	78 282	214 175,7	220 062,4	203 208,0	−5,12	−7,66
5	76 607	220 928,0	220 686,4	201 723,0	−8,69	−8,59
6	79 093	213 987,2	210 339,0	203 253,0	−5,02	−3,37
7	72 979	206 793,3	208 245,8	193 804,0	−6,28	−6,93
8	88 610	231 333,9	224 404,5	226 042,0	−2,29	0,73
9	64 180	198 594,1	183 327,9	178 576,0	−10.08	−2,59
dix	66 878	201 771,0	192 069,0	188 790,0	−6,43	−1,71
11	42 946	132 466,6	130 055,9	132 231,0	−0,18	1,67
12	86 047	213 445,5	211 726,2	195 981,0	−8.18	−7,44
13	74 643	199 471,6	197 240,0	195 772,0	−1,85	−0,74
14	85 209	198 005,1	200 193,9	200 257,0	1.14	0,03
15	40 715	135 491,1	125 875,5	127 045,0	−6,23	0,93
16	46 548	144 580,2	149 411,0	135 689,0	−6,15	−9.18
17	71 555	200 971,1	199 875,3	184 830,0	−8.03	−7,53
18	39 533	98 901,8	97 031,1	98 106,0	−0,80	1.11
19	47 495	149 973,9	151 618,8	138 420,0	−7,70	−8,71
20	58 189	170 524,4	163 785,9	163 740,0	−3,98	−0,03

Tableau 6 suite

SET3	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
21	44 182	141 578,2	134 625,9	124 919,0	−11,77	−7,21
22	130 235	256 283,6	245 549,4	246 270,0	−3,91	0,29
23	96 810	229 468,8	215 893,6	209 798,0	−8,57	−2,82
24	105 300	272 965,6	245 491,9	241 071,0	−11,68	−1,80
25	203 044	329 382,0	333 236,6	324 800,0	−1,39	−2,53
26	145 184	286 229,0	289 459,6	280 060,0	−2,16	−3,25
27	145 420	294 614,0	297 025,5	286 754,0	−2,67	−3,46
28	145 227	225 567,2	224 734,0	227 483,0	0,85	1.22
29	79 813	189 879,7	185 569,7	191 242,0	0,72	3.06
30	274 018	415 185,0	407 150,8	399 907,0	−3,68	−1,78

Tableau 7 Comparaison pour les instances SET4 (limite de temps = 300 s)

SET4	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
1	16 353	57 483,0	53 168,4	53 062,3	−7,69	−0,20
2	31 541	80 772,8	77 346,7	73 884,2	−8,53	−4,48
3	24 864	68 176,9	67 097,7	66 030,7	−3,15	−1,59
4	27 786	72 989,4	68 995,1	68 662,6	−5,93	−0,48
5	25 450	67 329,1	66 993,7	66 328,8	−1,49	−0,99
6	30 632	75 042,4	74 601,8	70 698,3	−5,79	−5,23
7	22 650	62 993,3	64 132,5	62 974,3	−0,03	−1,81
8	40 532	81 200,6	84 586,1	80 914,5	−0,35	−4,34
9	13 490	55 901,5	52 041,0	51 457,5	−7,95	−1,12
dix	15 542	55 602,2	57 297,3	55 341,7	−0,47	−3,41
11	12 802	28 415,7	28 323,5	28 207,1	−0,73	−0,41
12	43 341	73 653,4	72 084,8	71 886,9	−2,40	−0,27
13	28 152	52 525,0	55 251,9	52 518,4	−0,01	−4,95
14	56 174	79 086,4	80 501,7	78 903,9	−0,23	−1,98
15	14 628	25 927,5	25 286,3	24 568,9	−5,24	−2,84
16	17 171	35 048,5	35 138,7	34 569,2	−1,37	−1,62
17	29 001	51 396,2	51 671,9	51 266,5	−0,25	−0,78
18	19 184	26 101,5	26 282,3	26 037,4	−0,25	−0,93
19	10 724	31 585,8	33 006,4	31 139,0	−1,41	−5,66
20	18 718	38 796,1	38 781,4	37 179,1	−4,17	−4,13

Tableau 7 suite

SET3	XLP	Valeurs des solutions			Déviation (%)	
		Aheur	LugNP	RFFO	Aheur	LugNP
21	15 812	25 727,0	25 840,8	25 713,0	−0,05	−0,49
22	91 715	120 008,2	119 481,0	118 749,0	−1,05	−0,61
23	55 058	74 180,4	73 297,4	73 296,6	−1,19	0
24	58 919	82 349,4	82 260,2	80 733,2	−1,96	−1,86
25	171 987	196 626,7	196 025,1	196 023,0	−0,31	0
26	110 570	137 224,6	134 856,0	134 854,0	−1,73	0
27	101 114	135 936,6	132 463,3	132 451,0	−2,56	0
28	112 892	126 553,7	126 157,1	125 872,0	−0,54	−0,23
29	51 149	66 131,1	66 217,4	66 131,4	0	−0,13
30	241 678	262 380,7	263 042,1	262 378,0	0	−0,25

Les références

Absi, N., Detienne, B., Dauzère-Pérès, S. : Heuristique pour le problème de lotissement capacité multi-items avec ventes perdues. Calcul. Oper. Rés. 40(1), 264–272 (2013)

Akartunalı, K., Miller, AJ : Une approche heuristique pour les gros problèmes de planification de la production à plusieurs niveaux. EUR. J. Oper. Rés. 193(2), 396–411 (2009)

Akartunalı, K., Miller, AJ : une analyse informatique des limites inférieures pour la planification de la production de grands seaux problèmes. Calcul. Optim. Appl. 53(3), 729–753 (2012)

Almada-Lobo, B., Klabjan, D., Carravilla, MA, Oliveira, JF : lotissage continu de plusieurs machines avec des configurations dépendant de la séquence. Calcul. Optim. Appl. 47(3), 529–552 (2010)

Almeder, C. : Une approche d'optimisation hybride pour les problèmes de dimensionnement de lots capacitifs à plusieurs niveaux. EUR. J. Oper. Rés. 200, 599–606 (2010)

Baki, MF, Chaouch, BA, Abdul-Kader, W. : Une procédure de solution heuristique pour le dimensionnement dynamique des lots problème de refabrication et de récupération des produits. Calcul. Oper. Rés. 43, 225-236 (2014)

Ball, MO : Heuristique basée sur la programmation mathématique. Surv. Oper. Rés. Gérer. Sci. 16(1), 21–38 (2011)

Barany, I., van Roy, TJ, Wolsey, LA : Dimensionnement des lots sans capacité : la coque convexe des solutions. Mathématiques. Programme Étude 22, 32–43 (1984)

Belvaux, G., Wolsey, LA : bc-prod : un système spécialisé de branchement et de coupe pour les problèmes de dimensionnement des lots. Gérer. Sci. 46(5), 724–738 (2000)

Billington, PJ, McClain, JO, Thomas, LJ : Heuristique pour le dimensionnement de lots à plusieurs niveaux avec un goulot d'étranglement. Gérer. Sci. 32, 989-1006 (1986)

Degraeve, Z., Jans, R. : Une nouvelle reformulation de Dantzig-Wolfe et un algorithme de branchement et de prix pour le problème capacité de dimensionnement des lots avec des temps de configuration. Oper. Rés. 55(5), 909–920 (2007)

Eppen, GD, Martin, RK : Résoudre des problèmes de dimensionnement de lots capacitifs à plusieurs éléments à l'aide de la redéfinition de variables. Oper. Rés. 35(6), 832–848 (1987)

Federgruen, A., Meissner, J., Tzur, M. : Heuristique à intervalles progressifs pour le dimensionnement de lots capacitifs à plusieurs éléments problème. Oper. Rés. 55(3), 490–502 (2007)

Florian, M., Lenstra, JK, Rinnooy Kan, HG : Planification déterministe de la production : algorithmes et complexité. Gérer. Sci. 26(7), 669–679 (1980)

Helber, S., Sahling, F. : Une approche de correction et d'optimisation pour le problème de dimensionnement de lots capacitifs à plusieurs niveaux. Int. J. Prod. Écon. 123, 247-256 (2010)

Kébé, S., Sbihi, N., Penz, B. : Une heuristique lagrangienne pour un problème de dimensionnement de lot capacité de stockage à deux échelons. J. Intel. Fab. 23(6), 2477–2483 (2012)

Krarup, J., Bilde, O.: Emplacement de l'usine, recouvrement d'ensembles et tailles de lots économiques: un algorithme O(mn) pour les problèmes structurés. Optimierung bel Graphentheoretischen und Ganzzahligen Probleme, pp. 155–180. Birkhauser (1997)

Küçükyavuz, S., Pochet, Y. : Lot-sizing sans capacité avec backlog : la coque convexe. Mathématiques. Programme. 118(1), 151–175 (2009)

Miller, AJ, Nemhauser, GL, Savelsbergh, MWP : Sur la structure polyédrique d'une production multi-articles modèle de planification avec temps de préparation. Mathématiques. Programme. 94(2–3), 375–405 (2003)

MIPLIB. Une bibliothèque de problèmes d'entiers purs et mixtes (2010). http://miplib.zib.de/. Consulté le 29 déc. 2014

Multi-LSB. Problèmes de dimensionnement de lots multi-articles avec backlog : une bibliothèque d'instances de test (2014). http://dx. doi.org/10.15129/252b7827-b62b-4af4-8869-64b12b1c69a1. Consulté le 29 décembre 2014 Pochet, Y., Wolsey, LA : Production Planning by Mixed Integer Programming. Springer, Berlin (2006)

Ramezanian, R., Saidi-Mehrabad, M. : Recuit simulé hybride et heuristique basée sur le mip pour le problème stochastique de dimensionnement et d'ordonnancement des lots dans un système de production multi-étapes capacitif. Appl Math Modell 37(7), 5134–5147 (2013)

Rardin, RL, Wolsey, LA : inégalités valides et projection de la formulation étendue multi-produits pour les problèmes de flux de réseau à charge fixe sans capacité. EUR. J. Oper. Rés. 71(1), 95–109 (1993)

Seeanner, F., Almada-Lobo, B., Meyr, H. : Combinaison des principes de la recherche de décomposition de voisinage variable et de l'heuristique de correction et d'optimisation pour résoudre des problèmes de dimensionnement et d'ordonnancement à plusieurs niveaux. Calcul. Oper. Rés. 40(1), 303–317 (2013)

Stadtler, H. : Dimensionnement de lots à plusieurs niveaux avec des temps de configuration et de multiples ressources limitées : roulement interne calendriers avec des fenêtres de dimensionnement des lots. Oper. Rés. 51, 487–502 (2003)

Toledo, CFM, da Silva Arantes, M., de Oliveira, RRR, Almada-Lobo, B. : Ordonnancement de la production de récipients en verre via un algorithme évolutif hybride basé sur plusieurs populations. Appl. Calcul doux. 13(3), 1352–1364 (2013)

Toledo, CFM, de Oliveira, RRR, França, PM : Un algorithme génétique multi-population hybride appliqué pour résoudre le problème de dimensionnement des lots à capacités multiples avec arriéré. Calcul. Oper. Rés. 40(4), 910–919 (2013)

Van Vyve, M., Pochet, Y. : une heuristique générale pour les problèmes de planification de la production. INFORME J. Comput. 16(3), 316–327 (2004)

Wu, T., Shi, L., Geunes, J., Akartunalı, K. : Un cadre d'optimisation pour résoudre les problèmes de dimensionnement de lots à plusieurs niveaux capacitifs avec backlog. EUR. J. Oper. Rés. 214(2), 428–441 (2011)