Journal international de recherche en production, 2018
Vol. 56, n ° 7, 2684-2706, https://doi.org/10.1080/00207543.2017.1381348

Planification stochastique des commandes clients avec des temps de configuration pour minimiser le temps de cycle attendu

Yaping Zhao, Xiaoyun Xu *, Haidong Li et Yanni Liu

Département de génie industriel et de gestion, Université de Pékin, Pékin, Chine

(*Reçu le 8 janvier 2016; accepté le 10 septembre 2017)*

Le temps de cycle court des commandes clients est crucial pour les entreprises afin de réaliser une personnalisation de masse et une réponse rapide. Cependant, l'environnement compliqué et stochastique, en particulier l'existence de temps de configuration, rend extrêmement difficile l'optimisation de l'ef fi cacité d'un système. Dans cette étude, les commandes stochastiques des clients sont planifiées pour minimiser leur temps de cycle prévu en tenant compte des temps de configuration. Les commandes des clients arrivent dynamiquement et chaque commande nécessite plusieurs types de produits avec des charges de travail aléatoires. Ces charges de travail seront affectées à un ensemble de machines parallèles non liées à traiter. En particulier, pour chaque machine, des temps de configuration sont requis chaque fois qu'il y a un changement de type de produit, et les longueurs dépendent à la fois de la machine et du type de produit. Ce document vise à minimiser le temps de cycle de commande attendu à long terme par des politiques appropriées, notamment l'allocation de la charge de travail et le séquençage des types. Les impacts de la séquence des types de produits et de la variance de la charge de travail sont évalués par une étude théorique et plusieurs propriétés analytiques sont développées. À l'aide de ces propriétés, trois algorithmes d'ordonnancement sont proposés et une borne inférieure est dérivée pour évaluer les algorithmes proposés. Une expérience de calcul est menée pour démontrer l'efficacité de la borne inférieure et des algorithmes dans diverses circonstances, et plusieurs informations de gestion importantes sont également fournies. trois algorithmes d'ordonnancement sont proposés et une borne inférieure est dérivée pour évaluer les algorithmes proposés. Une expérience de calcul est menée pour démontrer l'efficacité de la borne inférieure et des algorithmes dans diverses circonstances, et plusieurs informations de gestion importantes sont également fournies. trois algorithmes d'ordonnancement sont proposés et une borne inférieure est dérivée pour évaluer les algorithmes proposés. Une expérience de calcul est menée pour démontrer l'efficacité de la borne inférieure et des algorithmes dans diverses circonstances, et plusieurs informations de gestion importantes sont également fournies.

Mots clés: ordonnancement stochastique; commande du client; temps d'installation; machine parallèle indépendante; temps de cycle prévu

1. Introduction

Alors que la personnalisation de masse constitue la nouvelle frontière de la concurrence commerciale, une réponse rapide aux demandes changeantes des clients est devenue une condition préalable pour que les fabricants conservent un avantage concurrentiel. Afin d'augmenter la réactivité du processus de traitement des commandes clients, la réduction du temps de cycle est un choix impératif. Les avantages d'une approche systématique de la planification de la réduction du temps de cycle ont été bien perçus dans toutes les industries manufacturières. Comme l'indiquent de nombreux chercheurs (par ex. Tige 1988 ; Hopp et Spearman 2008), un temps de cycle plus court peut grandement faciliter la tâche de planification opérationnelle en production et améliorer l'ef fi cacité de l'exécution des commandes clients.

La présence du temps de configuration a un impact dramatique sur le temps de cycle (KimandBobrowski 1997) Enquête sur la fabrication aux États-Unis entreprises (Kim et Bobrowski 1997) indique que le délai de configuration est l'une des principales causes qui compromettent la livraison à temps des commandes des clients. La programmation en présence de temps de configuration a attiré une grande attention au cours des dernières décennies en raison de son grand potentiel d'économies (Allahverdi, Gupta et Aldowaisan 1999 ; Allahverdi et al. 2008). Une enquête récente menée par Allahverdi et Soroush (2008) examine un grand nombre d'applications industrielles et suggère qu'il y a d'énormes avantages lorsque le temps de configuration est incorporé dans les décisions de programmation.

Cet article étudie un problème de planification de commande client avec des temps de configuration dépendants du type de machine et de produit dans un environnement stochastique. Les commandes des clients arrivent dynamiquement dans une installation de traitement. Plusieurs types de produits sont requis par chaque commande client entrante, et chaque type nécessite une charge de travail aléatoire indépendante. L'usine de transformation comprend un total de M machines pour traiter les charges de travail en parallèle, et les vitesses des machines sont prédéterminées et indépendantes les unes des autres. Les charges de travail peuvent être arbitrairement divisées et affectées aux machines pour être traitées indépendamment. Le temps d'installation est engagé chaque fois qu'une machine passe du traitement d'un type de produit à un autre. La durée de la configuration dépend à la fois des deux types de produits adjacents et de la machine de traitement elle-même. Une commande client ne sera livrée qu'après que toutes ses charges de travail auront été traitées. Ce document vise à minimiser le temps de cycle de commande prévu à long terme, à savoir,

$$\min OBJ(r, \delta) = \lim_{n \to \infty} E[CT_n(r, \delta)], \tag{1}$$

où $CT_n(r, \delta)$ représente le temps de cycle du n e commande client sous politique (r, δ). L'ensemble de variables de décision est la politique (r, δ) où δ et r sont respectivement l'affectation de la charge de travail et la relation adjacente au type de produit.

* Auteur correspondant. Email: xiaoyun.xu@pku.edu.cn

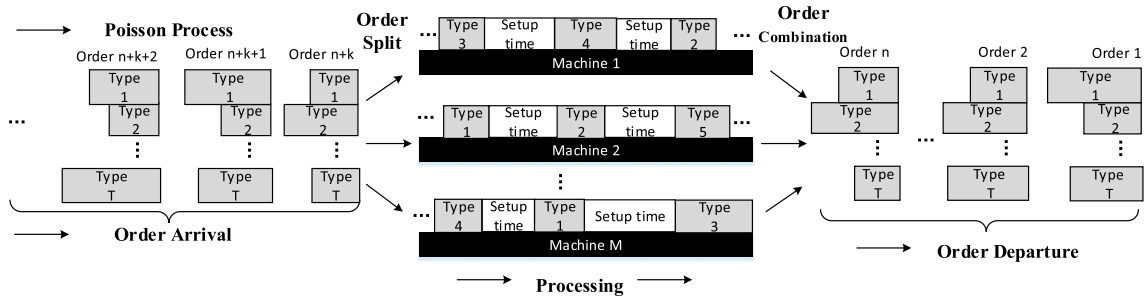

Figure 1. Problème de planification de commande client dynamique avec les temps de configuration: une illustration.

Figure 1 illustre le problème de planification des commandes clients dans un environnement stochastique tel que décrit ci-dessus. Le problème étudié se pose dans diverses industries. Par exemple, dans l'industrie de la production de peinture, une commande contient généralement différents types de peintures et un temps de configuration est nécessaire pour nettoyer la machine chaque fois qu'un changement de couleur est nécessaire dans une unité de production. Le temps de configuration de l'opération de nettoyage dépend à la fois de la couleur à retirer et de la couleur pour laquelle la machine est préparée (Conway, Maxwell et Miller 2012). Un autre exemple est l'opération de tissage dans l'industrie textile. Pour cette opération, il existe une énorme catégorie de tissus à tisser sur les lignes de production, et les temps de configuration du changement dépendent des types de tissus traités en séquence (Jungwattanakit et al. 2008). Le problème concerné par cette étude se retrouve également dans l'industrie du plastique, l'industrie du verre, l'industrie des boissons sans alcool, l'industrie de l'imprimerie, l'industrie chimique et l'industrie de la fabrication du papier (Radhakrishnan et Ventura 2000 ; Og, Salman et Yalçın 2010).

La décision d'affectation de la charge de travail dans le problème étudié partage certaines similitudes avec celle des problèmes d'emballage de bacs. Les problèmes d'emballage des bacs consistent à emballer des articles de différentes tailles dans des bacs de manière à optimiser certaines fonctions objectives données telles que le nombre de bacs (Garey et Johnson 1981 ; Coffman, Garey et Johnson 1984 , 1996 ; Martello, Pisinger et Vigo 2000 ; Lodi, Martello et Monaci 2002 ; Lodi, Martello et Vigo 2002). Par analogie avec les problèmes d'emballage de bacs, la charge de travail dans cette étude peut être considérée comme des «articles» et les machines comme des «bacs». Cependant, le problème étudié a certaines caractéristiques distinctes qui sont différentes des problèmes classiques d'emballage de bacs. Premièrement, notre problème est de nature stochastique, et les charges de travail et les heures d'arrivée des commandes des clients sont des variables aléatoires. La distribution de probabilité (et non les valeurs) est la seule information disponible avant l'arrivée de la commande. Cependant, les problèmes d'emballage des bacs sont déterministes et des informations telles que les dimensions des articles et le volume des bacs sont données à titre de connaissance a priori. Deuxièmement, dans notre problème, nous considérons un flux infini de commandes de clients entrants (également des charges de travail), tandis que le nombre total d'articles dans des problèmes d'emballage de bacs est fini. Troisièmement, en raison de l'existence de temps de configuration dans notre étude, les deux types de produits eux-mêmes et leur séquence sur chaque machine ont un effet sur la valeur de l'objectif. Les temps d'installation, cependant, n'ont pas été pris en compte dans la littérature existante pour les problèmes d'emballage de bacs à notre connaissance. En outre, dans ce problème, il faut non seulement décider de l'affectation des charges de travail aux machines, mais également de la quantité affectée et des séquences de types de produits sur chaque machine. Ces décisions sont couplées à travers un ensemble de contraintes et de fonctions de coût, ce qui complique la recherche de la solution optimale. Au contraire, les problèmes génériques d'emballage de bac se concentrent uniquement sur la décision d'affectation et ne prennent pas en considération les autres décisions simultanément. n'ont pas été pris en compte dans la littérature existante pour les problèmes d'emballage de bacs à notre connaissance. En outre, dans ce problème, il faut non seulement décider de l'affectation des charges de travail aux machines, mais également de la quantité affectée et des séquences de types de produits sur chaque machine. Ces décisions sont couplées à travers un ensemble de contraintes et de fonctions de coût, ce qui complique la recherche de la solution optimale. Au contraire, les problèmes génériques d'emballage de bac se concentrent uniquement sur la décision d'affectation et ne prennent pas en considération les autres décisions simultanément. n'ont pas été pris en compte dans la littérature existante pour les problèmes d'emballage de bacs à notre connaissance. En outre, dans ce problème, il faut non seulement décider d

Pour résoudre le problème considéré, cette étude adopte l'approche du modèle Fork – Join queue (FJ queue) comme méthodologie principale. Un modèle de file d'attente FJ décrit un système stochastique dans lequel les commandes entrantes sont divisées à l'arrivée pour le service par plusieurs machines en parallèle et combinées ensemble après la fin de tous les services (Baccelli, Makowski et Shwartz 1989 ; Kim et Agrawala 1989). L'adoption de l'approche du modèle de file d'attente FJ pour le problème étudié est motivée par les trois considérations suivantes. Premièrement, un modèle de file d'attente FJ peut facilement capturer le caractère aléatoire dans des systèmes stochastiques tels que les arrivées dynamiques et les charges de travail aléatoires. Deuxièmement, à la fois dans un modèle de file d'attente FJ et dans le problème de planification des commandes client étudié, toutes les charges de travail entrantes sont divisées et affectées à un ensemble de machines parallèles, puis combinées ensemble pour partir une fois tous les services terminés. La manière de gérer la charge de travail et la structure de traitement de la machine sont également assez similaires. Enfin, la modélisation du système étudié par le modèle de file d'attente FJ permet d'effectuer une analyse en régime permanent, qui est un élément essentiel de l'optimisation de la planification de la production. Au contraire,

Avec l'aide de l'approche du modèle de file d'attente FJ, les principales contributions de cette étude peuvent être résumées comme suit. Premièrement, il fournit un modèle plus réaliste d'applications pratiques en considérant à la fois les arrivées dynamiques et les charges de travail aléatoires des commandes clients. Comparé aux problèmes d'ordonnancement statiques traditionnels, ce modèle est capable de capturer une composition de commande compliquée et un comportement stochastique du client. Deuxièmement, les temps de configuration dans cette étude sont à la fois machine et produit

en fonction du type et les vitesses des machines sont complètement hétérogènes. Un tel problème a été à peine envisagé dans la littérature existante. Troisièmement, cette étude résout un problème d'optimisation stochastique plutôt que de simplement fournir des évaluations des performances de la fonction objectif. En particulier, trois heuristiques sont proposées pour optimiser le système. Ces heuristiques se sont avérées efficaces contre une variété de problèmes de planification d'ordres caractérisés par des structures de configuration et des distributions de charge de travail différentes.

Le reste de l'article est organisé comme suit. Dans la section 2 , les études existantes pertinentes sont passées en revue. Section 3 décrit le problème en détail à l'aide des notations et hypothèses principales. Plusieurs propriétés analytiques du problème sont développées dans la section 4 . Sur la base de ces propriétés, trois algorithmes heuristiques sont proposés dans la section 5 et une borne inférieure de calcul est développée à des fins d'évaluation. La performance des algorithmes proposés et de la borne inférieure est démontrée dans la section 6 à travers des expériences numériques. Enfin, des conclusions sont tirées dans la section 7 et les futures orientations de recherche sont également décrites.

2. Revue de la littérature

De nombreux travaux ont examiné les problèmes de planification des commandes clients. Cependant, la plupart de la littérature existante se consacre aux cas statiques où les arrivées de commandes et les charges de travail correspondantes sont prédéterminées et connues à l'avance. Le principe général du problème de planification des commandes clients est d'abord introduit par Julien etMagazine (1990) et Leung, Li et Pinedo (2005b) trie trois catégories de problèmes dans un environnement de machine parallèle statique après un examen approfondi: (1) le cas entièrement dédié, (2) le cas entièrement fl exible et (3) le cas arbitraire. En ce qui concerne l'environnement de traitement, le problème abordé dans cette étude appartient à la dernière catégorie, qui est la plus générale des trois concernant la capacité de traitement. Leung, Li et Pinedo (2005a) et Roemer (2006) démontrent qu'il est NP-difficile au sens strict de minimiser le temps moyen d'achèvement, même pour le cas entièrement dédié, sans tenir compte du temps de configuration. Une vaste littérature a été consacrée à la recherche des deux premières catégories (Blocher et Chhajed 1996 ; Ng, Edwin Cheng et Yuan 2003 ; Ahmadi, Bagchi et Roemer 2005 ; Yang et Posner 2005 ; Leung, Li et Pinedo 2005a , 2006 , 2007 ; Wang et Edwin Cheng 2007 ; Lee 2013 ; Su, Chen et Chen 2013). En raison de la complexité considérable, cependant, beaucoup moins a été fait dans le cas arbitraire.

Yang (2005) prouve l'exhaustivité NP du cas arbitraire. Le problème déterministe sans configuration est considéré dans Xu et al. (2013) pour minimiser le temps total d'exécution d'un ensemble de commandes client. Une version stochastique de ce problème dans le but de minimiser le temps de cycle d'ordre attendu est étudiée plus en détail dans Xu et al. (2015b , 2016) où sont construits respectivement la méthode d'optimisation de la simulation et des algorithmes heuristiques. Étant donné que les configurations ne sont pas impliquées, la variable de décision est beaucoup plus simple et ne contient que l'allocation de la charge de travail entrante. L'objectif de maximisation du débit est considéré dans

Xu et al. (2015a) et Zhao, Xiaoyun et Li (2017) dans le cadre de deux schémas d'attribution, dont l'un est l'ordonnancement sur des machines parallèles arbitraires. Suite à ces deux études, Zhao et al. (2016) analyse plus en détail les commandes clients prioritaires. Différentes de notre article, ces trois études se concentrent sur le débit et la relation entre les deux schémas différents étudiés. Aucune étude supplémentaire sur le cas arbitraire, à la connaissance des auteurs, n'a été trouvée dans la littérature.

Lorsque le temps de configuration est impliqué, le problème devient beaucoup plus compliqué et très peu de recherches ont été menées directement concernant les commandes des clients, même pour les cas déterministes. Au lieu de cela, la majorité des études se concentrent sur les emplois individuels plutôt que sur les commandes des clients. Par exemple, Xing et Zhang (2000) proposent une heuristique pour minimiser la durée de vie d'un lot de travaux fractionnables sur des machines parallèles identiques avec un temps de configuration indépendant de la séquence. Un problème similaire avec le temps de configuration dépendant de la séquence est considéré dans Yalaoui et Chu (2003) à travers une heuristique, et approfondi dans Tahar et al. (2006). Pour les problèmes de planification de machines parallèles non liés, afin de minimiser le temps moyen d'achèvement pondéré, Weng, John et Ren

(2001) présentent sept algorithmes heuristiques prenant en compte les temps de configuration dépendants de la séquence. Le même problème est considéré dans Banque et Werner (2001) pour minimiser la somme pondérée de la précocité et du retard. L'Etude de Kim et al. (2002) adopte un recuit simulé pour minimiser le retard total de plusieurs travaux, chacun étant composé de quelques éléments identiques, et les auteurs étendent ensuite l'enquête à la planification par lots par quatre heuristiques de recherche (Kim, Na et Chen 2003). Les auteurs de Lin et Hsieh (2014) construire une métaheuristique hybride itérée pour minimiser le retard pondéré total avec des temps de configuration et des temps de préparation dépendant de la séquence et de la machine, tandis qu'un problème similaire est analysé dans Chen (2009), Lin, Chung-Cheng et Ying (2011) et Lee, Jae-Min et Lee (2013) avec une contrainte supplémentaire de contraintes de date d'échéance strictes pour certains travaux. La durée des emplois est minimisée grâce à une heuristique dans Rabadi, Moraga et Al-Salem (2006) avec des temps de configuration dépendant de la machine et de la séquence de travaux. Le même problème est également résolu par l'algorithme de recherche tabou (Helal, Rabadi et Al-Salem 2006) et algorithme d'optimisation des colonies de fourmis (Arnaout, Rabadi et Musa 2010 ; Arnaout, Musa et Rabadi 2014). Contrairement aux cas déterministes où des informations complètes sont disponibles au tout début, les problèmes de planification dynamique avec les temps de configuration sont beaucoup plus difficiles à résoudre même pour les travaux (Cheng, Gupta et Wang 2000 ;

Allahverdi et al. 2008) en raison des incertitudes liées aux informations. Aucune documentation existante n'a été trouvée concernant les problèmes de planification des commandes clients dans les cas dynamiques, et seules quelques études existent concernant les travaux individuels. Logendran et Subur (2004) propose un algorithme de solution heuristique basée sur la recherche tabou avec des offres d'emploi dynamiques et une disponibilité dynamique des machines

considération. Cependant, le temps de configuration est supposé être inclus dans le temps de traitement. Logendran, McDonell et Smucker (2007) étudie plus en détail ce problème avec les configurations dépendantes de la séquence, mais les travaux ne peuvent pas être fractionnés. La planification par lots avec des temps de configuration dépendants de la séquence est analysée dans Arnaout, Rabadi et Mun (2006) où les temps de traitement et les temps de configuration sont stochastiques.

La conclusion de l'examen ci-dessus est que, compte tenu de l'état des recherches actuelles, le problème de planification des commandes client stochastiques sur une machine parallèle sans rapport avec les temps de configuration est nouveau. À notre connaissance, ce problème n'a pas été étudié auparavant par d'autres spécialistes de la littérature.

3. Description du problème

Le problème considéré dans cette étude peut être formellement décrit comme suit: les commandes des clients arrivent dynamiquement dans un centre de traitement, et les délais entre arrivées $\{une_n\}_{n=1}^{\infty}$ suivre un processus de Poisson. Pour une commande client entrante NT différents types de produits, $T = \{1, 2, \ldots, T\}$, sont nécessaires avec des charges de travail aléatoires indépendantes $\{q_t$ $_n\}_{n=1}^{\infty}$ où q_t $_n$ est la charge de travail du produit type t.

Dans l'usine de transformation, M Machines, $M = \{1, 2, \ldots, M\}$, existent pour traiter les charges de travail en parallèle. Les vitesses des machines sont hétérogènes et prédéterminées par la matrice de vitesse $V = [v_{mt}]_{M \times T}$. Les charges de travail de chaque commande client entrante peuvent être fractionnées et affectées à toutes les machines, chaque machine traite indépendamment les charges de travail qui lui sont affectées. Devant chaque machine, une file d'attente séparée avec un tampon infini peut être formée pour attendre les commandes des clients.

La politique (r, δ), qui se compose de l'ensemble des variables de décision, est appliqué à l'arrivée d'une commande client. Ici, la variable de décision $r = [r_{ijm}]_{M \times T \times T}$ désigne la relation d'adjacence du type de produit dans une commande, où r_{ijm} $\in \{0, 1\}$ est égal à 1 si type de produit je et j, $\forall\, i = j, i, j \in T$ sont adjacents sur la machine m, et 0 sinon. Variable de décision $\delta = [\delta_{mt}]_{M \times T}$ représente l'affectation de la charge de travail dans une commande, et $\delta_{mt} \in [0, 1]$ est la portion du type de produit t charge de travail affectée à la machine m dans chaque ordre où $\sum_{m \in M} \delta_{mt} = 1$, $\forall\, t \in T$. La planification nécessite que la stratégie (r, δ) devrait être appliqué à tous les commandes entrantes.

Chaque fois qu'un changement se produit entre deux types de produits adjacents je et j, $\forall\, i = j$ sur machine m, un temps de configuration déterministe s_{ijm} est encouru et la matrice de temps de configuration $S = [s_{ijm}]_{M \times T \times T}$ est prédéterminé. La file d'attente premier entré, premier sorti (FIFO) la discipline est appliquée à la file d'attente devant chaque machine et la préemption est interdite. Par conséquent, les relations suivantes existent entre les temps d'attente dans la file d'attente $\{w_{mn}(r, \delta)\}_{n=0}^{\infty}$ et temps de cycle $\{CT_{mn}(r, \delta)\}_{n=0}^{\infty}$:

$$w_{mn}(r, \delta) = (w_{mn-1}(r, \delta) + p_{mn-1}(r, \delta) + cuillère\ à\ café_{mn-1}(r, \delta) - une_n)^{+}, \quad \forall\, m \in M,\ n \in Z^{+}, \tag{2}$$

$$CT_{mn}(r, \delta) = w_{mn}(r, \delta) + p_{mn}(r, \delta) + cuillère\ à\ café_{n}(r, \delta), \quad \forall\, m \in M, \tag{3}$$

où $p_{mn}(r, \delta)$ indique le temps de traitement total de la charge de travail affectée à la machine m du n e ordre sous politique (r, δ), et $cuillère\ à\ café_n(r, \delta)$ est le temps d'installation total du n e commande sur machine m. Équation (2) tient parce qu'un travail doit attendre s'il arrive avant la fin du premier. Plus précisément, w_{m0} est le temps pour la machine m pour effacer ses charges de travail système initiales.

De plus, étant donné qu'une commande client ne sortira du système que lorsque toutes ses charges de travail seront terminées, le temps de cycle du n e commande client, $\{CT_n(r, \delta)\}_{n=0}^{\infty}$, satisfait les relations suivantes:

$$CT_n(r, \delta) = \max_{1 \le m \le M} \{CT_m^n(r, \delta)\}. \tag{4}$$

L'objectif est de minimiser le temps de cycle de commande attendu à long terme en déterminant la meilleure politique (r, δ), à savoir,

$$\min OBJ(r, \delta) = \lim_{n \to \infty} E[CT_n(r, \delta)] = \lim_{n \to \infty} E\left[\max_{1 \le m \le M} \{CT_m^n(r, \delta)\}\right]. \tag{5}$$

Les cinq hypothèses suivantes (ASP1) - (ASP5) sont adoptées dans cette étude: (ASP1) Temps entre arrivées une_n suivent une distribution exponentielle avec taux λ;

(ASP2) La charge de travail du type de produit t dans chaque ordre, q_t, est indépendant des autres types ou ordres et aléatoire avec attente définitive, c.-à-d. $E[q_t] < \infty$;

(ASP3) Heures inter-arrivées une_n et charges de travail q_t $_n$ sont mutuellement indépendants, $\forall\, NT$;

(ASP4) Sur toute machine, la charge de travail du même type de produit dans une commande sera traitée ensemble; (ASP5) Matrice de vitesse $V = [v_{mt}]_{M \times T}$ et configurer la matrice de temps $S = [s_{ijm}]_{M \times T \times T}$ sont prédéterminés et indépendants de $\{une_n\}_{n=1}^{\infty}$, $\{q_t$ $_n\}_{n=0}^{\infty}$, et w_{m0}'s. Pour les temps de configuration, $s_{ijm} = s_{mji}$, $\forall\, i = j, i, j \in T$.

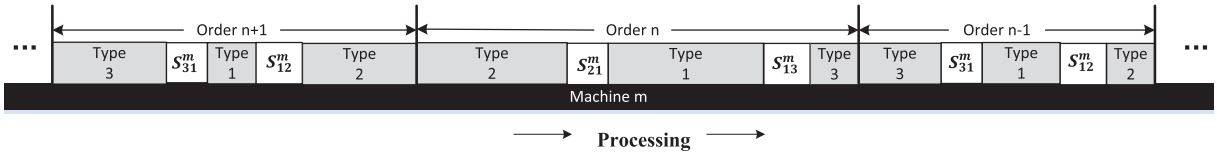

Figure 2. Illustration d'une séquence optimale.

4. Propriétés analytiques

Dans cette section, il est destiné à montrer que le problème décrit dans la section 3 présente de nombreuses propriétés intéressantes. En particulier, l'impact de la séquence de traitement est étudié dans la section 4.1 . Sur la base de ce résultat, la section 4.2 explore davantage l'impact de la variabilité de la charge de travail sur la fonction objectif. Les informations obtenues à partir des propriétés analytiques proposées seront utilisées dans la construction d'algorithmes et l'évaluation des performances dans les sections suivantes.

4.1 *L'impact de la séquence de traitement*

Envisager une politique réalisable (*r, δ)* d'abord. Laisser *seq m (r, δ)* désigne la séquence de traitement de tous les types de produits dans une seule commande affectée à la machine *m* sous politique (*r, δ),* et laisse *seq'*

m (r, δ) être sa séquence antithétique. Par exemple, si pour n'importe quelle machine

m, la séquence de traitement de *seq m (r, δ)* est ' *une, b, c, d, e* ', puis *seq'* *m (r, δ)* représente la séquence ' *e, ré, c, b, a* '. cependant,

puisque les deux *seq m (r, δ)* et *seq'* *m (r, δ)* donner le même (*r, δ),* il convient de noter que les séquences de types de produits entre

les commandes ne sont pas complètement spéci fi ées même lorsque la police (*r, δ)* est déterminé. Par conséquent, la règle dominante suivante sur la séquence de types peut être développée:

Théorème 1 *Le temps de cycle optimal peut être atteint lorsque la séquence de traitement seq m (r, δ),* ∀ *m* ∈ M *et sa séquence antithétique seq'*

m (r, δ), ∀ *m* ∈ M *sont exécutés en alternance pour chaque paire adjacente de commandes client entrantes.*

Preuve. En vertu d'une politique donnée (*r, δ),* s'il n'y a qu'un seul type de produit sur la machine *m,* il est évident que *seq m (r, δ)* et *seq'* *m (r, δ)* sont identiques.

Sinon, sous Hypothèse (ASP4), considérons le cas où le premier et le dernier type de produit attribué sont différents. Étant donné que le dernier type de produit de l'ancienne commande est différent du premier de la commande actuelle, il doit exister un temps de configuration entre deux commandes adjacentes sur la machine. *m* si la séquence *seq m (r, δ)* ou *seq'*

m (r, δ) est effectuée sur des commandes successives.

Cependant, si des séquences *seq m (r, δ)* et *seq'* *m (r, δ)* sont effectués en alternance, ce temps de configuration est éliminé. Par ailleurs, l'Assomption (ASP5) implique que le temps de configuration *s m* *ij = s mj i,* ∀ *i, j* ∈ T, ce qui rend le temps de configuration total dans une commande sous *seq m (r, δ)* et *seq'* *m (r, δ)* égal. Par conséquent, le délai d'exécution de toute commande sur la machine *m* est moins si *seq m (r, δ)* et *seq'* *m (r, δ)* sont appliqués alternativement.

Étant donné que la preuve ci-dessus est vraie pour toute politique (*r, δ),* l'optimal doit avoir la même propriété. Figure 2 est une □

illustration de la séquence optimale décrite dans le théorème 1 .

Étant donné que le problème concerne un flux infini de commandes clients, il est plausible de supposer que la relation entre les commandes adjacentes doit également être prise en considération. Avec l'aide de Theorem 1 , on peut conclure que la décision optimale concernant (*r, δ)* est suf fi sant pour déterminer le temps de cycle optimal. La séquence de traitement spécifiée dans le théorème 1 sera donc adopté par défaut dans l'étude suivante.

4.2 *L'impact de la variabilité de la charge de travail*

Il est bien reconnu que la présence d'une variance de temps de traitement a un impact négatif sur le temps de cycle (Hopp et Spearman 2008). Puisque la matrice de vitesse de la machine est fixe et prédéterminée, la source de variation du temps de traitement vient du caractère aléatoire de la charge de travail. Pour évaluer l'impact de la variation de la charge de travail, considérons un cas spécial où les commandes des clients ont le même modèle d'arrivée et de traitement que celui spécifié dans la section 3 , la seule différence étant que le flux de traitement est déterministe. Pour être précis, chaque élément du flux de traitement { *p m*

n (r, δ)} ∞ *n* = 0 du système étudié est remplacé par le attente correspondante *p m (r, δ),* à savoir,

$$p_m(r,\delta) = \sum_{t \in T} \frac{E[q_{tj} \times \delta_{mt}}{v_{mt}}, \quad \forall m \in M, \tag{6}$$

où $E[q_{tj}]$ représente la charge de travail attendue du type de produit t comme indiqué dans Hypothèse (ASP2), et δ_{mt} est la portion de la charge de travail attribuée du type de produit t usiner m.

De façon similaire à l'analyse du système stochastique considéré dans cette étude, pour le cas avec un flux de traitement déterministe, dénoter le temps d'attente en file d'attente par $w_{mn+1}(r, \delta)$, le temps de cycle sur chaque machine par $CT_{mn}(r, \delta)$, et le temps de cycle de commande par $CT_n(r, \delta)$. Ensuite, le théorème suivant tient:

Théorème 2 *Pour toute commande client sous une certaine politique (r, δ), son temps de cycle prévu sera réduit si les charges de travail de chaque commande client sont égales à leurs attentes, c'est-à-dire*

$$E[\underline{CT}_n(r, \delta)] \leq E[CT_n(r, \delta)].$$

Preuve. Veuillez vous référer à l'annexe 1 . Théorème 2 est établi pour chaque commande client individuelle et est donc de nature transitoire. Avec une condition □ supplémentaire sur la stabilité du système, le théorème suivant étend ce résultat aux performances en régime permanent.

Théorème 3 *Étant donné la condition de stabilité suivante*

$$E[p_{n(r, \delta) + \text{cuillère à café}(r, \delta)}] < E[a_{n+1}], \forall m \in M,$$

les écarts de charge de travail ont un impact négatif sur la valeur objective, c'est-à-dire

$$\lim_{n \to \infty} E[\overline{CT}_n(r, \delta)] = OBJ(r, \delta) \leq \overline{OBJ}(r, \delta) = \lim_{n \to \infty} E[CT_n(r, \delta)], \tag{7}$$

où OBJ (r, δ) est le temps de cycle de commande prévu à long terme en vertu de la politique (r, δ) avec un flux de traitement déterministe. Preuve. La preuve est très similaire au résultat en régime permanent Baccelli, Makowski et Shwartz (1989) et est donc omis par souci de concision. □

Théorème 3 suggère qu'un temps de cycle plus court peut être obtenu en réduisant la variance de la charge de travail sur chaque machine. Étant donné que les charges de travail des différents types de produits sont indépendantes (Hypothèse (ASP2)), la variance du temps de traitement total sur chaque machine est une combinaison linéaire de toutes les variations de la charge de travail des types de produits correspondants. Par conséquent, afin de réduire la variance, il semble préférable d'éviter autant que possible la répartition de la charge de travail. Cette intuition est utilisée pour guider la conception de la construction de l'algorithme dans la section suivante.

En outre, dans les cas extrêmes où les charges de travail des commandes clients sont déterministes, il peut être démontré que le problème peut être formulé et résolu par le modèle de programmation mixte mixte suivant:

$$(MIP) \min C_{max}$$

$$\text{st } C_{max} \geq \sum_{t \in T} [\delta_{mt} \times E[q_{tj}/v_{mt}] + \sum_{je \in T} \sum_{j \in T} r_{mij} \times s_{mij}/2, \forall m \in M \tag{8}$$

$$\sum_{m \in M} \delta_{mt} = 1, \forall t \in T \tag{9}$$

$$r_{mij} = r_{mji}, \forall m \in M, \forall i, j \in T \tag{dix}$$

$$\delta_{mt} \geq \varepsilon \times b_{mt}, \forall m \in M, \forall t \in T \tag{11}$$

$$\delta_{mt} \leq b_{mt}, \forall m \in M, \forall t \in T \tag{12}$$

$$b_{mi} \geq r_{mij}, \forall m \in M, \forall i, j \in T \tag{13}$$

$$\sum_{t \in T} b_{mt} - 1 = \sum_{je \in T} \sum_{j \in T} r_{mij}/2, \forall m \in M \tag{14}$$

$$r_{mii} = 0, \forall m \in M, \forall je \in T \tag{15}$$

$$r_{mij} \in \{0, 1\}, \forall m \in M, \forall i, j \in T$$

$$b_{mt} \in \{0, 1\}, \forall m \in M, \forall t \in T$$

$$0 \leq \delta_{mt} \leq 1, \forall m \in M, \forall t \in T$$

ε est une constante positive suffisamment petite,

pour lesquelles des explications détaillées des contraintes sont listées ci-dessous:

Contraintes (8): ces inégalités définissent le make-span supérieur ou égal au temps de réalisation sur chaque machine. Étant donné que les types de produits adjacents sont comptés deux fois, le temps de configuration total doit être divisé par 2;

Contraintes (9): toutes les charges de travail de tout type de produit doivent être affectées; Contraintes (dix): type de produit *je* et *j* sont considérés comme adjacents sur la machine *m* peu importe lequel est assigné en premier; Contraintes (11): si type de produit *t* n'est pas affecté sur la machine *m* (c'est à dire $\delta_{mt} = 0$), b_{mt} doit être égal à 0; Contraintes (12): si type de produit *t* est affecté sur la machine *m* (c'est à dire $\delta_{mt} > 0$), b_{mt} doit être égal à 1; Contraintes (13): si type de produit *je* n'est pas affecté sur la machine *m* (c'est à dire $b_{mi} = 0$), il n'est adjacent à aucun autre produit

taper sur la machine *m* (c'est à dire $r_{mij} = 0$). Sinon, elle peut ou non être contiguë à d'autres $r_{mij} = 1$ ou $r_{mij} = 0$);

Contraintes (14): pour toute machine, le nombre de relations adjacentes est égal au nombre de types de produits attribués sur cette machine moins 1;

Contraintes (15): il n'y a pas de temps de configuration lorsqu'il n'y a pas de commutateur de type de produit. L'énoncé formel du résultat ci-dessus est présenté dans le théorème suivant:

Théorème 4 *Lorsque les charges de travail sont déterministes et égales à leurs attentes, une politique optimale peut être obtenue en résolvant le programme mixte en nombres entiers MIP.* *Proof.* Veuillez vous référer à l'annexe 2 . De la construction de *MIP* dans Theorem 4 , il convient de noter que la politique correspondante (r_{MIP}, δ_{MIP}) ne garantit pas automatiquement son optimalité au système concerné car les charges de travail sont aléatoires en fait. Cependant, en raison de la ressemblance étroite entre le système dans cette étude et celui avec des charges de travail déterministes, l'idée d'une construction de calendrier optimale peut être empruntée pour développer des algorithmes de planification pour le problème.

Comme politique (r_{MIP}, δ_{MIP}) minimise la durée du makespan, il est raisonnable de supposer qu'une bonne politique du problème de planification stochastique d'origine doit équilibrer le temps d'occupation (temps de traitement plus temps de configuration) entre les machines via une partition de charge de travail appropriée. Le corollaire suivant sur le temps d'occupation est obtenu:

Corollaire 1 *Lorsque les écarts de charges de travail sont égaux à zéro, pour toute commande client sous la politique optimale (r_{MIP}, δ_{MIP}), la plus grande différence de ses temps d'occupation (temps de traitement plus temps de configuration) entre les machines ne dépasse pas le temps de configuration s_{m*ij}, où $m*$ est la machine avec le plus petit temps de réalisation, i est le dernier type de produit sur la machine $m*$, et j est tout type de produit affecté à la machine avec le temps de réalisation le plus long.* *Preuve.* Par contradiction. Si l'instruction ne tient pas, il est toujours possible de supprimer une partie de la charge de travail sur la machine avec le temps d'exécution le plus long jusqu'à la fin de la machine $m*$. Cela conduira à une C_{max}. Une contradiction. □

Le corollaire ci-dessus du solde du temps d'occupation est employé dans la section 5 pour aider à la construction de l'algorithme.

5. Algorithmes de planification

Compte tenu des propriétés proposées dans les sections 4.1 et 4.2 , cette section est consacrée à la conception et à l'évaluation d'algorithmes d'ordonnancement. Trois algorithmes heuristiques nommés *Alg* 1, *Alg* 2 et *Alg* 3 sont construits, et une borne inférieure est proposée et exprimée sous des formes explicites. Des descriptions détaillées des algorithmes sont présentées dans les sous-sections suivantes.

5.1 *Algorithme Alg* 1

Inspiré par le théorème 4 , *Alg* 1 génère l'affectation initiale en minimisant le makespan (sans temps de configuration en premier). Ensuite, sur la base de l'affectation initiale, une règle gourmande est adoptée pour séquencer les charges de travail affectées sur chaque machine.

Plus précisément, la conception de *Alg* 1 découle de deux considérations. D'une part, la réduction du temps de traitement peut entraîner une réduction du temps de cycle. D'un autre côté, les types de produits avec un temps d'installation adjacent plus court ont tendance à être assemblés de manière à réduire le temps d'installation total. Sur la base de ces considérations, l'affectation initiale est générée en minimisant le makespan (C_{max}).

Les temps d'installation sont délibérément exclus dans la première étape afin de fournir rapidement des solutions initiales. Dans l'étape suivante, les types de produits attribués sur chaque machine sont organisés par une règle gourmande pour réduire le temps de configuration total. C'est-à-dire que pour chaque machine, le type avec le temps de configuration moyen le plus court est mis en première position, puis le type qui prend le temps de configuration le plus court avec le premier placé en position suivante. Cela continue jusqu'à ce que tous les types attribués soient séquencés. Le diagramme de flux en annexe 3 illustre la conception de *Alg* 1 qui peut encore être spécifié par le pseudo-code suivant:

5.2 *Algorithme Alg* 2

Différent de *Alg* 1 où les charges de travail de différents types de produits peuvent être réparties entre les machines, *Alg* 2 nécessite initialement qu'un seul type de produit soit traité par une seule machine, puis équilibre les temps d'achèvement attendus sur toutes les machines. L'impact du temps d'installation est destiné à être minimisé en traitant l'intégralité des charges de travail d'un seul produit

Algorithme *Alg* 1

Contribution: $V = [v_{mj}]_{M \times T}$, $S = [sm_{ij}]_{M \times T \times T}$, $\{E[qt]\}_T$ $t = 1$

Première étape: affectation de la charge de travail

résoudre le programme linéaire suivant pour δ:

$$\min C_{max}$$

$$\text{st } C_{max} \geq \sum_{t \in T} [\delta_{mt} \times E[qt] / v_{mt}], \forall m \in M$$

$$\sum_{m \in M} \delta_{mt} = 1, \forall t \in T$$

$$0 \leq \delta_{mt} \leq 1, \forall m \in M, t \in T;$$

Étape deux: séquence de types

calculer le temps d'installation moyen de chaque type de produit t sur chaque machine m comme $sm_t = (\sum_{j \in T} sm_{tj}) / T$;

pour *chaque machine m* **faire**

trouver t = argmin { $sm_{t} / \delta_{mt} > 0$}; mettre le type de produit t en première position;

tandis que *tous les types de produits ne sont pas placés* **faire**

trouver i = argmin { $sm_{ti} / \delta_{mi} > 0$, type de produit t est le dernier placé}; mettre le type de produit je à la position suivante;

fin fin retour r *Alg* 1 = [$r_{mj}]_{M \times T}$, δ *Alg* 1 = [$\delta_{mt}]_{M \times T}$

tapez ensemble. De plus, selon Corollary 1 , il est raisonnable de rendre approximatifs les temps de traitement prévus pour toutes les machines, c'est-à-dire qu'aucune machine ne devrait être beaucoup plus occupée que les autres.

Plus précisément, lors de la première étape, toutes les machines sont définies pour être inactives et laisser tous les types de produits être «non affectés» au départ, et le temps de traitement attendu de chaque type de produit sur chaque machine est calculé. Ensuite, la paire de type de produit «non affectée» conduisant au temps de cycle attendu actuel le plus court est sélectionnée, et ce type de produit est affecté à la machine couplée. Le processus se poursuit jusqu'à ce que tous les types de produits soient affectés. À la deuxième étape, répertoriez d'abord toutes les machines par ordre croissant de temps d'achèvement prévu. Ensuite, ajustez la charge de travail sur les première et dernière machines s'il est possible de raccourcir le temps d'achèvement prévu de la dernière machine. Autrement dit, allouez une quantité appropriée de charge de travail du dernier type de produit sur la dernière machine à la fin de la première machine pour que ces deux machines finissent en même temps. La même procédure est effectuée pour les deuxième et avant-dernière machines, et se poursuit jusqu'à ce qu'aucune paire de ce type n'existe. Ces opérations peuvent être illustrées par le diagramme de flux en annexe 3 et le pseudo-code suivant fournit des détails sur *Alg* 2:

5.3 *Algorithme Alg* 3

Alg 3 est un algorithme itératif développé à partir de l'idée d'équilibrage de la charge de travail en corollaire 1 . Pour réduire la complexité causée par les temps de configuration, un type de produit factice est introduit et combiné avec les types d'origine pour former un nouveau type de commande client. Cette nouvelle commande est ensuite planifiée en ajustant de manière itérative la séquence de types et l'affectation de la charge de travail.

Essentiellement, la conception de *Alg* 3 entend simplifier le problème en optimisant un seul des δ et r à un moment basé sur les informations de l'autre. Pour être précis, pour un δ, les types de produits attribués sur chaque machine sont séquencés par une règle gourmande, c'est-à-dire que celle qui entraîne le temps de configuration actuel le plus court est placée dans la position vacante suivante. Sur la base de la séquence générée par la règle gourmande, les temps de configuration totaux sur chaque machine peuvent être calculés, et ils sont considérés comme les temps de traitement d'un type de produit factice. Ensuite, la charge de travail de tous les types d'origine est affectée avec le type factice pour minimiser le temps d'exécution maximal via un programme mathématique. Si la solution suggère une règle où un certain type ne doit pas être placé sur une machine spécifique, cette règle sera maintenue pour le reste des itérations. Sur la base de cette solution, les temps de traitement du type factice doivent être ajustés en conséquence, ce qui rend nécessaire de reséquencer les types de produits. Un tel processus itératif se poursuit jusqu'à ce qu'une affectation indique qu'aucun type n'a besoin d'être ajusté davantage. La description ci-dessus peut être illustrée par le diagramme de flux en annexe 3 . Le pseudo-code ci-dessous explique plus en détail les opérations de *Alg* 3 plus en détail.

5.4 *Évaluation d'algorithme*

Alg 1, *Alg* 2 et *Alg* 3 fournir des solutions réalisables au problème de planification stochastique de la commande client. Pour évaluer la qualité de ces solutions, des comparaisons peuvent être faites avec l'optimum global. Cependant, en raison des contraintes de synchronisation sur

Algorithme *Alg* 2

Contribution: $V = [v_{mt}]_{M \times T}$, $S = [sm$ $_{ij}]_{M \times T \times T}$, $\{E[qt]\}_T$ $t = 1$

Première étape: générer une stratégie initiale sans fractionnement de la charge de travail

ensemble $t_{mt} = E[qt] / v_{mt}$, $\delta_{mt} = rm$ $_{ij} = 0$ et $C_m = 0$, le temps d'achèvement prévu des types de produits actuellement affectés à la machine m,

$\forall m \in M, \forall t, i, j \in T$; étiqueter tous les types de produits comme «non attribués»;

tandis que *tous les types de produits ne sont pas affectés* faire

  trouver la paire type-machine «non affectée» (t, m), cela conduirait au temps de cycle actuel le plus court; ensemble $\delta_{mt} = 1$ et

  $rmit = rm$ $_{ti} = 1$ où type de produit je est celui qui précède le type de produit t sur machine m; étiquette type de produit t comme «attribué»; mettre à jour

  Cm;

fin

Deuxième étape: équilibrer les délais d'achèvement attendus entre les machines en fractionnant

réétiqueter toutes les machines par ordre croissant de Cm; ensemble $C_{max} = 0$;

si $C_1 + s_1$ $_{ij} \geq CM$ où les types de produits i et j sont respectivement les derniers des première et dernière machines puis

  ensemble $C_{max} = CM$; Arrêtez

autre

  pour *chaque paire de machines* (m, $M - m + 1$), *où m augmente de 1 à* M_2 faire —

    si $C_m + sm$ $_{ij} < CM - m + 1$ où les types de produits i et j sont les derniers sur les machines m et $M - m + 1$ respectivement puis

      ensemble $\delta_{mj} = \{(C_{M-m+1} - C_m - s_m$ $_{ij} \cdot v_{M-m+1,j} \cdot v_{mj}) / (v_{M-m+1,j} + v_{mj}) \cdot E[q_{ij}]$ et $\delta_{M-m+1,j} = 1 - \delta_{mj}$, $C_m = C_{M-m+1} = C_m + sm$ $_{ij} + \delta_{mj} E[qj] / v_{mj}$

      et rm $_{ij} = r_{mj}$ $i = 1$;

    autre ensemble $C_{max} = max\{C_{max}, C_{M-m+1}, C_{M_2+1}\}$;

      —

    end end end

return $r_{Alg\,2} = [rm$ $_{ij}]_{M \times T \times T}$, $\delta_{Alg\,2} = [\delta_{mt}]_{M \times T}$

arrivée et départ, le calcul du temps de cycle attendu est difficile même lorsque la politique de programmation est prédéterminée (Baccelli et Makowski 1989). Compte tenu de cette dif fi culté et de l'absence de politique optimale, il est raisonnable de chercher des limites à l'objectif optimal pour que les algorithmes proposés puissent être évalués.

Comme indiqué dans la section 4.2 , Théorème 3 indique que pour toute police individuelle (r, δ), la valeur objective du cas avec des charges de travail déterministes, $OBJ(r, \delta)$ n'est pas plus grand que celui du système étudié, $OBJ(r, \delta)$. Cependant, cette limite n'est utile que lorsque la politique optimale (r_{opt}, δ_{opt}) du système étudié est connu. Comme la politique optimale en général est difficile à obtenir, une telle borne inférieure ne peut pas être utilisée directement. De plus, tout comme la fonction objectif (5), l'expression explicite de $OBJ(r, \delta)$ est assez complexe en raison des contraintes de synchronisation, ce qui limite son applicabilité en analyse numérique.

Depuis le MIP formulé dans le théorème 4 fournit la solution optimale au cas de charge de travail déterministe, sa valeur objective, $OBJ(r_{MIP}, \delta_{MIP})$, doit être inférieure à toute valeur objective (y compris optimale) du système étudié. Par conséquent, cette valeur peut servir de limite inférieure complètement indépendante de la politique (r, δ) du problème d'origine. Le corollaire suivant montre qu'une telle borne inférieure (KG) peut être exprimé explicitement:

Corollaire 2 *Une borne inférieure de la valeur objective, OBJ* (r, δ)*, peut s'exprimer comme suit:*

$$LB = OBJ(r_{MIP}, \delta_{MIP}) = C_{max}(2 - \lambda C_{max}) / 2(1 - \lambda C_{max}), \quad (16)$$

où C_{max} *est le makespan obtenu par MIP, et* λ *est le taux d'arrivée des commandes clients entrantes. Preuve.* Ce résultat est une conséquence facile du théorème 4 . Le système sans variation de la charge de travail peut être modélisé $MARYLAND/1$ file d'attente pour le $T_{je}(r, \delta) = T_j(r, \delta)$ Cas. Pour le $T_{je}(r, \delta) = T_j(r, \delta)$ Dans ce cas, le temps de cycle étant déterminé par le temps d'achèvement de la machine dominante, il devient le premier cas. Équation (16) est simplement l'expression du temps de cycle attendu de $MARYLAND/1$ file d'attente. Le reste de la preuve est trivial et donc omis par souci de concision.

□

Équation (16) peut être utilisé comme critère pour effectuer une évaluation étant donné l'absence de la valeur optimale du système concerné. En outre, en ce qui concerne l'efficacité du calcul, la construction de KG peut être établie de manière pratique pour les problèmes de petite et moyenne ampleur puisque le calcul C_{max} nécessite uniquement la résolution d'un programme entier mixte. Pour les problèmes à grande échelle, une relaxation de la programmation peut être utilisée comme approximation. En raison de sa simplicité, ce

Algorithme *Alg* 3

Contribution: $V = [v_{mt}]_{M \times T}$, $S = [sm_{ij}]_{M \times T \times T}$, $\{E[qt]\}_T$ $t = 1$

initialiser l'affectation de la charge de travail comme $\delta_{mt} = 1/M, \forall m, t;$

Étape 1: Séquence de type pour *chaque machine m* **faire**

trouver $t = \text{argmin}\{sm_{ij}\}$: **mettre le type de produit** *t* **en première position; tandis que** *tous les types de produits ne sont pas placés* **faire**

trouver $i = \text{argmin}\{sm_{ti}\}$ / **type de produit** *t* **est le dernier placé}; mettre le type de produit** *je* **à la position suivante; ensemble** $rm_{it} = 1$;

fin fin pour *chaque machine m* **faire**

calculer le temps de traitement attendu du type $T+1$ sur machine *m* comme $Pm_{T+1} = \sum_{i,j \in T} smij * rm_{ij;}$

fin

Deuxième étape: affectation de la charge de travail

résoudre le programme linéaire suivant pour δ:

$$\min C_{\max}$$

$$\text{st } C_{\max} \geq \sum_{t \in T} [\delta_{mt} \times E[qt] / v_{mt}] + Pm_{T+1}, \forall m \in M$$

$$\sum_{m \in M} \delta_{mt} = 1, \forall t \in T$$

$$0 \leq \delta_{mt} \leq 1, \forall m \in M, t \in T;$$

Troisième étape: amélioration des politiques si *il en sort un* $\delta_{mt} = 0$ *dans* δ *supérieur à 0 dans la dernière affectation* **puis**

définir ces $\delta_{mt} = 0$ **en permanence; retourner à Première étape et mettre à jour** *r* **Compte tenu du** $\delta_{mt} = 0$ **information;**

autre Arrêtez.

retour final $r_{Alg3} = [rm_{ij}]_{M \times T}$, $\delta_{Alg3} = [\delta_{mt}]_{M \times T}$

bound peut être considéré comme une alternative pour évaluer les performances de chaque algorithme de programmation. Ceci est particulièrement intéressant car l'objectif optimal est en général difficile à atteindre.

Outre la qualité de la solution, la complexité de calcul est également cruciale (Juman et Hoque 2015), car il est directement lié à la ef fi cacité des algorithmes. L'analyse et les résultats correspondants sont présentés comme suit:

Lemme 1 *Les complexités informatiques d'Alg* 1, *Alg* 2 *et Alg* 3 *sont* $O((MT)^{3,5} L_{2\,Alg1})$, $O(MT^2)$ *et* $O((MT)^{4.5} L_{2\,Alg3})$, *respectivement, où* L_{Alg1} *et moi* $_{Alg3}$ *sont le nombre de bits dans les entrées correspondantes de Alg* 1 *et Alg* 3.

Preuve. Veuillez vous référer à l'annexe 4 . Lemme 1 indique que les trois algorithmes proposés peuvent être mis en œuvre en temps polynomial. Comme le montrent les résultats de complexité, *Alg* 2 est le plus ef fi cace alors que *Alg* 3 entraîne le coût de calcul le plus lourd. Ces conclusions seront démontrées numériquement par les résultats de l'expérience dans la section 6.2 .

6. Expérience de calcul

6.1 *Conception de l'expérience*

Le but de l'expérience est de tester les performances de la borne inférieure et les algorithmes proposés pour ce nouveau problème d'ordonnancement des commandes client stochastiques. Afin d'atteindre cet objectif, non seulement les instances de problème de petite taille doivent être prises en compte, mais également certaines instances de problème de grande taille. Parce que dans le cas des instances de problèmes numériques de grande taille, certains algorithmes sont inférieurs aux algorithmes existants, bien qu'ils puissent surpasser ceux existants dans le cas des instances de problèmes numériques de petite taille (voir Juman et Hoque 2014). Par conséquent, les problèmes à petite et à grande échelle sont examinés et les tailles sont répertoriées comme suit:

- Problèmes à petite échelle
  - Nombre de machines $M = 3, 4, 5,$

- Nombre de types de produits $T = 6, 8, 10$;

- Problèmes à grande échelle
  - Nombre de machines $M =$ dix, 15, 20,
  - Nombre de types de produits $T = 30, 40, 50$.

Vitesses de la machine v_{mt} sont générés aléatoirement à partir de la distribution normale $N(\mu_v = 5, \sigma_v = 0{,}25)$ et prendre des valeurs absolues au cas où des valeurs négatives seraient générées. Pour évaluer l'impact de la variabilité de la charge de travail et de la différence de temps de configuration sur les performances de l'algorithme, les quatre scénarios suivants sont envisagés:

- Charge de travail

  Pour représenter la différence de demande de produit, la charge de travail moyenne du type de produit t, à savoir $E[q_t]$, $\forall\, t \in T$, est tiré au hasard de la distribution normale $N(\mu_w = 0{,}5, \sigma_w = 0{,}025)$ et prend la valeur absolue. Ce paramètre garantit une variance modérée entre les charges de travail moyennes.

  - **Charge de travail relativement uniforme (RUW):** charge de travail du type de produit t dans chaque ordre est généré aléatoirement à partir de la distribution normale $N(E[q_t], \sigma_t)$, où σ_t est tirée au hasard de la distribution uniforme U (0,03, 0,07).
  - **Charge de travail très variable (HVW):** charge de travail du type de produit t dans chaque ordre est généré aléatoirement à partir de la distribution normale $N(E[q_t], \sqrt{2}\,\sigma_t)$, de sorte que la variance est deux fois plus grande que celle du paramètre RUW.

- Temps d'installation
  - **Temps de configuration relativement uniforme (RUS):** Temps d'installation s_{mij} sont générés aléatoirement à partir de la distribution $N(\mu_s = \mu_w/\mu_v * 0{,}1, \sigma_s = 0{,}05 * \mu_s)$ et prendre des valeurs absolues. La moyenne est établie sur la base de l'étude de Yalaoui et Chu (2003) et l'écart-type fait la différence entre les temps de configuration limités à moins de 35%. Ce paramètre représente le cas où les temps de configuration sont modérément différents.
  - **Temps de configuration très variable (HVS):** Temps d'installation s_{mij} sont générés aléatoirement à partir de la distribution normale $N(\mu_s = \mu_w/\mu_v * 0{,}1, \sigma_s = 0{,}1 * \mu_s)$ et prendre des valeurs absolues pour que la différence entre les temps de configuration puisse atteindre 100%. Ce paramètre représente le cas où les temps de configuration sont très différents.

Notez que le cas de zéro temps de configuration ($s_{mij} = 0$) peut également être couvert par la définition générale du problème dans cette étude. Pour ça cas particulier, *Alg* 1 et *Alg* 3 sont les mêmes et dégénèrent en procédure de la première étape de *Alg* 1. Pour évaluer les algorithmes dans de telles circonstances, l'algorithme génétique basé sur la simulation SimGA Xu et al. (2015b) est également menée, qui s'est avérée plus performante que d'autres méthodes bien connues.

Pour chaque réglage de problème spécifique, les commandes clients arrivent dynamiquement selon un processus de Poisson avec un taux d'arrivée $4 * M / T$. Compte tenu des schémas de génération de vitesses, de charges de travail et de temps d'installation ci-dessus, le taux d'arrivée garantit une intensité de trafic modérée avec une intensité de trafic nominale de 0,6 à 0,8. Une telle con fi guration garantit l'existence d'une solution en régime permanent tout en évitant la banalité du trafic de lumière.

Il y a $|M| \times |T| = 3 \times 3 \times 2 = 18$ combinaisons testées dans chacun des six scénarios ($2 \times 2 + 2$) énumérés ci-dessus. Le comportement à long terme du système est évalué par des simulations informatiques avec des 1000e ordres. L'effet des périodes d'échauffement et de refroidissement est éliminé par la procédure Welch (Welch 1983), c'est-à-dire que les statistiques de 600 commandes (201e à 800e) sont utilisées pour approximer les performances à long terme. En raison de la nature stochastique, 5 répliques indépendantes sont générées aléatoirement sous un paramètre spécifique, et 5 paramètres sont générés pour chaque instance individuelle.

Toutes les simulations sont codées à l'aide de MATLAB R2012a et implémentées sur un ordinateur avec un processeur de 3,6 GHz et une mémoire RAM de 16,0 Go. Pour obtenir la borne inférieure, des tentatives sont faites pour résoudre le *MIP* dans Theorem 4 par CPLEX Linear Optimizer version 12.0. Lorsque les temps de configuration sont pris en compte, étant donné que la résolution d'un programme entier mixte est NP-difficile en général, il est difficile d'obtenir la solution optimale dans un laps de temps raisonnable. Par conséquent, pour les problèmes à grande échelle, la relaxation de programmation linéaire associée de *MIP* est utilisé à la place. Cela permet d'acquérir une solution rapidement mais au prix d'une pire borne inférieure. Lorsque les temps de configuration ne sont pas pris en compte, les séquences de types n'ont en fait aucun effet sur la valeur de l'objectif et

MIP peut être réduit à un programme linéaire avec $\delta = [\delta_{mt}]_{M \times T}$ comme variable de décision.

Les performances des algorithmes proposés sont évaluées selon les deux critères suivants: (1) **Écart de performance**

La solution optimale étant extrêmement difficile à obtenir, les performances des algorithmes seront testées par rapport à la borne inférieure (*KG*) développé dans la section 5.4 . L'écart de performance est dé fi ni comme suit:

$$\frac{OBJ(r_{Alg}, \delta_{Alg}) - KG}{KG} \times 100\%,$$

Tableau 1. Écarts de performances et temps de calcul des algorithmes proposés sous (*RUW & RUS).*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*Alg* 3	min { *Alg* 1, *Alg* 2, *Alg* 3}	*Alg* 1	*Alg* 2	*Alg* 3	*KG*
3	6	7.18	6.27	6,65	6,27 (Alg2)	0,89	0,03	1,31	3.21
	8	4,48	8.12	3,72	3,72 (Alg3)	1.17	0,00	1,40	2,25
	dix	3,84	9.24	3,77	3,77 (Alg3)	1,31	0,00	1,24	3.16
4	6	10h45	7.04	7.71	7,04 (Alg2)	1.17	0,00	1,25	1,97
	8	10,51	6,74	5.13	5,13 (Alg3)	1,23	0,00	2,68	2,61
	dix	8,86	7.72	7.08	7,08 (Alg3)	1,22	0,00	2.12	4.51
5	6	9,50	49,63	9,50	9,50 (Alg3)	1,23	0,00	1,25	2,01
	8	8.44	31,59	6,80	6,80 (Alg3)	1,25	0,00	1,39	3.05
	dix	13,55	10,47	7,66	7,66 (Alg3)	1,34	0,00	3.22	4,81
dix	30	13,84	11,50	10,96	10,96 * (Alg3)	2,42	0,02	6,96	101,62 *
	40	10,72	9.78	10,52	9,78 * (Alg2)	2,92	0,01	22,55	251.14 *
	50	9.33	8,91	7.79	7,79 * (Alg3)	3,69	0,01	10,98	551,79 *
15	30	26,50	18,37	18.80	18,00 * (Alg3)	3,44	0,00	7,58	169,96 *
	40	18.10	26,90	15,66	15,66 * (Alg3)	4.33	0,01	6.29	461,72 *
	50	13,83	17,94	12,39	12,39 * (Alg3)	5.15	0,01	17,86	1069,96 *
20	30	25,75	23.09	23.00	23,00 * (Alg3)	4.19	0,01	14.12	279.04 *
	40	24,39	19.14	20.03	19,14 * (Alg2)	6,41	0,01	38,58	761,37 *
	50	19,26	16,32	16.11	16.11 * (Alg3)	7.41	0,01	19,41	1758,63 *

* *KG* est calculé par la relaxation de programmation linéaire de *MI P.*

où l'exposant ' *Alg* 'représente les algorithmes proposés dans cette étude ou le SimGA Xu et al. (2015b), et le correspondant *KG* sont obtenus en résolvant *MI P* dans Theorem 4 . Pour les problèmes à grande échelle avec les temps de configuration, *KG*

est approchée par la relaxation de programmation linéaire de *MI P.*

(2) **Temps de calcul**

Afin d'évaluer l'efficacité de calcul des algorithmes proposés, leurs temps d'exécution correspondants sont enregistrés en seconde (s) CPU.

6.2 *Résultats informatiques*

6.2.1 *Résultats en tenant compte du temps de configuration*

les tables 1 - 4 présenter les résultats expérimentaux détaillés des trois algorithmes proposés dans deux scénarios de charge de travail (RUW et HVW) et deux scénarios de temps de configuration (RUS et HVS). Les deux premières colonnes présentent l'échelle du problème sous chaque paramètre. Les colonnes 3 à 5 indiquent l'écart de performances entre les algorithmes et la borne inférieure. L'écart de performance minimal et l'algorithme correspondant sont répertoriés dans la colonne 'min { *Alg* 1, *Alg* 2, *Alg* 3} ». Le temps CPU en secondes est également indiqué dans les quatre dernières colonnes pour les algorithmes et la borne inférieure.

- **Analyse des performances de la borne inférieure**

La qualité de la borne inférieure proposée est cruciale pour l'évaluation des performances. Étant donné l'absence de la valeur objective optimale, l'étanchéité de la *KG* peut être évalué via la colonne 'min { *Alg* 1, *Alg* 2, *Alg* 3} 'qui peut être considéré comme une limite supérieure de la différence en pourcentage entre *KG* et la valeur optimale globale. Il convient de préciser que, pour calculer plus efficacement la borne inférieure proposée, *MI P* pour les problèmes à grande échelle est résolu par une relaxation linéaire du programme entier mixte dans l'étude informatique. Par conséquent, les performances de *KG* est, à condition que l'optimum puisse être localisé, au moins aussi bon que celui de la relaxation linéaire.

le *KG* fonctionne bien en général en ce qui concerne l'écart de performance. Comme le suggèrent les quatre tableaux, la majorité des lacunes fournies (63 cas sur 72) sont inférieures à 20%. Lorsque les charges de travail et les temps d'installation sont relativement uniformes, l'écart de performances

Tableau 2. Écarts de performances et temps de calcul des algorithmes proposés sous (*RUW & HV S).*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*Alg* 3	min { *Alg* 1, *Alg* 2, *Alg* 3}	*Alg* 1	*Alg* 2	*Alg* 3	*KG*
3	6	9.32	4,82	4,82	4,82 (Alg3)	1,21	0,00	2.12	1,67
	8	5.66	9,61	5.11	5,11 (Alg3)	1,21	0,00	1,32	2,36
	dix	7.37	12,64	7,58	7,37 (Alg1)	1,18	0,00	1,46	3,31
4	6	12,21	7,57	8,98	7,57 (Alg2)	1,23	0,00	1,42	1,92
	8	11.19	9,82	9,58	9,58 (Alg3)	1,18	0,00	2.16	2,85
	dix	8,86	7.72	7.08	7,08 (Alg3)	1,28	0,00	1,49	4.16
5	6	9,72	38.02	9,58	9,58 (Alg3)	1,36	0,00	1,47	2.19
	8	11,88	35,72	10,43	10,43 (Alg3)	1,34	0,00	1,58	3,53
	dix	13,89	12,67	6.34	6,34 (Alg3)	1.2	0,00	2,35	4.62
dix	30	16,82	14,24	13.20	13,20 * (Alg3)	2.26	0,00	6,94	97,51 *
	40	11,33	10,86	9,70	9,70 * (Alg3)	3.02	0,01	9.38	245,95 *
	50	11,25	10,63	9.36	9,36 * (Alg3)	3,61	0,01	8.57	552,47 *
15	30	19,46	16,85	17.16	16,85 * (Alg2)	3.26	0,00	15,57	168,85 *
	40	20.01	27,41	15,70	15,70 * (Alg3)	4.36	0,01	6,42	470.33 *
	50	15.03	21h15	13,72	13,72 * (Alg3)	5.82	0,01	13,52	1065,92 *
20	30	28,23	23,56	23,71	23,56 * (Alg2)	4.44	0,00	14.18	273,40 *
	40	24,93	21,60	21,88	21,60 * (Alg3)	6.38	0,01	25,25	761,47 *
	50	19.10	18,74	18,67	18,67 * (Alg3)	7,93	0,01	32,24	1791,37 *

* *KG* est calculé par la relaxation de programmation linéaire de *MI P.*

peut être aussi étroit que 3,72%. Le plus grand est toujours limité en dessous de 29,06%, ce qui est obtenu en résolvant la relaxation de *MI P* plutôt que *MI P* lui-même pour un problème à grande échelle sous *HVW & HV S* scénario. La moyenne de l'écart de performance minimal est d'environ 12,93%, ce qui est raisonnablement robuste compte tenu de la nature stochastique du problème.

L'échelle du problème n'a qu'un impact mineur sur *KG.* Comme le montrent les tableaux 1 - 4 , l'écart de performance est assez étroit pour les problèmes à petite échelle, et se dégrade légèrement lorsque l'échelle du problème devient grande. Cependant, cette dégradation n'est pas signi fi cative. La plus grande différence se situe toujours à moins de 14% *HVW & HV S* scénario. Il convient de noter que la définition du problème à grande échelle ici est résolue par la relaxation de programmation linéaire associée de *MI P,* c'est-à-dire que le véritable écart de performance pourrait être beaucoup plus petit. Par conséquent, le résultat indique que le *KG* est assez robuste par rapport à la capacité de production. Cette caractéristique inspire plus de confiance pour remplacer la valeur optimale par la borne inférieure proposée.

Les résultats montrent également que le nombre de machines *M* et le nombre de types de produits *T* ont une in fl uence différente sur l'ef fi *KG.* Comme indiqué dans les tableaux 1 - 4 , l'écart de performance tend à se réduire à mesure que *T* augmente lorsque *M* est fixe, tandis que pour chaque *T,* il grandit légèrement plus *M* devient plus grand. Cela est attendu car plus de types permettent d'affaiblir l'impact des temps de configuration en manipulant les types attribués sur chaque machine. En outre, pour les problèmes à plus grande échelle, une plus grande *T* est susceptible d'entraîner davantage de types de produits non divisés. Dans *MI P,* cela implique qu'il y a plus de zéros dans la matrice adjacente ce qui peut réduire les impacts de la relaxation linéaire et ainsi améliorer l'étanchéité de la borne inférieure. L'augmentation de *M,* cependant, a un impact négatif. Cette caractéristique favorise une plus grande confiance en substituant la valeur optimale à la borne inférieure lorsque le problème cible implique de nombreux types de produits. Dans de telles circonstances, il est raisonnable de s'attendre à ce que l'écart entre *KG* et l'optimum deviendra assez petit.

De plus, la variabilité de la charge de travail peut affecter l'efficacité de *KG.* En comparant les résultats entre les scénarios *RUW & RUS* et *HVW & RUS (* les tables 1 et 3) et scénarios *RUW & HV S* et *HVW & HV S (* les tables 2 et 4), on peut conclure que l'écart entre *KG* et l'optimum est faible lorsque les charges de travail sont relativement uniformes. Cette observation est cohérente avec le théorème 4 ce qui suggère que *MI P* est souhaitable lorsque toutes les commandes entrantes ont des charges de travail identiques.

Quant au temps de calcul de *KG,* résultats dansTables 1 - 4 manifeste que le *KG* peut être résolu très ef fi cacement pour des problèmes à petite échelle *MI P* dans Theorem 4 . Pour un problème à grande échelle, la relaxation de programmation linéaire de *MI P* est un choix viable, et les plus grandes instances testées, à savoir (*M, T) = (* 20, 50), peut encore être résolu en une demi-heure environ. Par conséquent, il peut

Tableau 3. Écarts de performances et temps de calcul des algorithmes proposés sous (*HVW et RUS).*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*Alg* 3	min { *Alg* 1, *Alg* 2, *Alg* 3}	*Alg* 1	*Alg* 2	*Alg* 3	*KG*
3	6	8.88	8.47	7.02	7.02 (Alg3)	1,48	0,03	1,61	3,47
	8	8.44	13,72	8.07	8,07 (Alg3)	1,73	0,00	1,82	2,85
	dix	7.04	13,99	7.00	7,00 (Alg3)	1,86	0,00	1,85	4,03
4	6	15.13	12,49	13,68	12,49 (Alg2)	1,48	0,00	1,92	2,46
	8	15h40	12,55	14,70	12,55 (Alg2)	1,39	0,00	2,57	3,36
	dix	9,85	9,93	8.34	8,34 (Alg3)	1,75	0,00	1,82	5.37
5	6	9,83	40,77	9,84	9,83 (Alg1)	1,58	0,00	1,83	2,74
	8	12,47	27,72	11,38	11,38 (Alg3)	1,88	0,00	2.17	4,69
	dix	15,47	14.13	14,82	14.13 (Alg2)	1,84	0,00	5,51	6,58
dix	30	16,33	16,37	15,33	15,33 * (Alg3)	2,63	0,02	5.36	105.06 *
	40	15.14	14,62	13,97	13,97 * (Alg3)	3,31	0,01	6,83	243,76 *
	50	12,51	12,62	12,54	12,51 * (Alg1)	3,48	0,01	10,84	562,75 *
15	30	29.00	21,99	18,79	18,79 * (Alg3)	3,47	0,00	12,68	173,78 *
	40	20,66	28.06	18,72	18,72 * (Alg3)	4.51	0,01	6,74	476,75 *
	50	20,52	23h30	16,86	16,86 * (Alg3)	5.66	0,01	13,47	1109.89 *
20	30	35,39	30,35	28,95	28,95 * (Alg3)	4,54	0,00	10.15	278,73 *
	40	27,50	24,91	23,78	23,78 * (Alg3)	6.35	0,01	14.17	763,78 *
	50	22,54	21,87	20,46	20,46 * (Alg3)	7,67	0,01	25,58	1818.42 *

* *KG* est calculé par la relaxation de programmation linéaire de *MI P.*

conclure que le *KG* est efficace lorsque l'échelle du problème est modérée, et sa relaxation ne supporte que des frais de calcul raisonnables, même pour des problèmes à grande échelle. Ceci est très souhaitable dans la pratique, en particulier lorsque les ressources de calcul sont limitées.

- **Analyse des performances de l'algorithme**

Algorithme *Alg* 1 a l'intention de générer d'abord l'affectation initiale en résolvant un problème simplifié, puis atténue l'impact des temps de configuration par une règle gourmande. Il est comparable au meilleur en termes d'écart de performances lorsque le nombre moyen de types de produits sur chaque machine est important (*T / M).* Par ailleurs, pour un *M,* l'écart de performance de *Alg* 1 tend à être réduit en fonction du nombre de types de produits (*T)* augmente. Dans les pratiques de fabrication modernes où la prolifération des produits est inévitable, cette robustesse contre la flambée du type de produit est une propriété souhaitable pour l'algorithme de programmation. La performance de *Alg* 1 devrait s'améliorer lorsque la variance des temps d'installation est modérée. Cela explique le fait que *Alg* 1 donne une approximation des temps d'achèvement prévus en faisant la moyenne des temps d'installation par conception. Une telle approximation est plus précise lorsque la différence de temps d'installation est modérée. Concernant le temps de calcul, *Alg* 1 est très ef fi cace. L'exécution de l'algorithme ne nécessite que quelques secondes, même pour la plus grande instance de test.

Algorithme *Alg* 2 a l'intention d'affecter la charge de travail sans fractionner au début, puis d'ajuster la stratégie en équilibrant les délais d'exécution. Il est le plus ef fi cace en termes de temps de calcul (presque négligeable), et a également les meilleures performances dans certaines circonstances. L'écart de performance peut être aussi petit que 4,82% et sa largeur devient plus étroite à mesure que la variance du temps de configuration diminue. Par conséquent, dans les pratiques de planification où les temps de configuration varient modérément, *Alg* 2 est capable de fournir des solutions de haute qualité. Les données indiquent également que les écarts de performance tendent à être plus *T* grandit pour tout donné *M.* La raison réside dans le fait que l'augmentation *T* peut entraîner la séparation de plusieurs types de produits. Cela coïncide avec la conception de *Alg* 2 au premier étage. De toute évidence, il est reconnu que les performances *Alg* 2 se détériore légèrement lorsque l'échelle du problème augmente. Ceci n'est cependant pas surprenant car le problème d'optimisation devient beaucoup plus complexe lorsque davantage de facteurs sont impliqués.

contrairement à *Alg* 1, *Alg* 3 détermine l'attribution de la charge de travail et la séquence de types de manière itérative. Comme le montrent les quatre tableaux, son écart de performance domine ceux des *Alg* 1 et *Alg* 2 plus de 80% des cas (57/72). L'écart tend également à diminuer avec l'augmentation du nombre moyen de types de produits sur chaque machine (*T / M).* Semblable à *Alg* 1, *Alg* 3 s'améliore lorsque la variance des temps d'installation est modérée. Même lorsque les délais d'installation varient considérablement, l'écart est toujours limité à 30,77% pour

Tableau 4. Écarts de performances et temps de calcul des algorithmes proposés sous (*HVW & HV S).*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*Alg* 3	min { *Alg* 1, *Alg* 2, *Alg* 3}	*Alg* 1	*Alg* 2	*Alg* 3	*KG*
3	6	11,78	10.17	8,98	8,98 (Alg3)	1,58	0,00	2,31	1,72
	8	7.71	17,33	6,54	6,54 (Alg3)	1,51	0,00	1,78	2,62
	dix	6,60	9.06	6.19	6,19 (Alg3)	1,57	0,00	1,73	3,94
4	6	16.14	14,64	11,83	11,83 (Alg3)	1,68	0,00	2,31	2,63
	8	14,71	12,27	10,58	10,58 (Alg3)	1,58	0,00	2,51	3,48
	dix	13,33	11,89	11,43	11,43 (Alg3)	1,78	0,00	1,91	5.37
5	6	14,58	47,90	14,53	14,53 (Alg3)	1,67	0,00	1,77	2,81
	8	20,72	34.17	13,74	13,74 (Alg3)	1,58	0,00	1,69	4,06
	dix	17,77	16,82	12,21	12,21 (Alg3)	1,68	0,00	2,96	6.21
dix	30	19,92	17,88	18,21	17,88 * (Alg2)	3,31	0,01	3.24	227,68 *
	40	15.16	15.16	13,94	13,94 * (Alg3)	3,34	0,01	4.52	245,76 *
	50	14,89	14,66	12.07	12.07 * (Alg3)	3,45	0,01	8.59	553,64 *
15	30	26,89	21,51	18,99	18,99 * (Alg3)	3,62	0,00	10,59	173,72 *
	40	19.03	29,91	18,72	18,72 * (Alg3)	4.38	0,01	9,84	476,27 *
	50	18.09	24,88	18.09	18.09 * (Alg3)	5,57	0,01	13,35	1131,76 *
20	30	34.10	29.06	30,77	29.06 * (Alg2)	4.72	0,00	14,63	285,78 *
	40	29,23	23,33	22,36	22,36 * (Alg3)	6,58	0,01	41,73	778,36 *
	50	24,55	24.04	22,95	22,95 * (Alg3)	7.28	0,01	37,59	1842,72 *

* *KG* est calculé par la relaxation de programmation linéaire de *MI P.*

HVW & HV S scénario (tableau 4), ce qui est raisonnable compte tenu du fait que les charges de travail sont très variables et que *KG* est résolu grâce à un programme détendu. Par conséquent, la règle gourmande de séquencer les types de produits s'avère efficace pour réduire le temps de cycle, et il est favorable de placer les types un par un en fonction de leurs temps de configuration. Bien que nécessitant le plus de temps de calcul par rapport aux autres, *Alg* 3 est capable de fournir des solutions en une minute même pour la plus grande instance de problème testée.

6.2.2 *Résultats sans considération de temps de configuration*

les tables 5 et 6 Énumérer les résultats expérimentaux détaillés des trois algorithmes proposés et de SimGA dans deux scénarios de charge de travail (RUW et HVW). Les deux premières colonnes présentent l'échelle du problème sous chaque paramètre. Les colonnes 3-5 indiquent l'écart de performances entre les algorithmes et la borne inférieure. Il convient de noter que depuis *Alg* 1 et *Alg* 3 sont les mêmes dans les circonstances indiquées dans la section 6.1 , ils sont considérés ensemble et étiquetés comme « *Alg* 1'. L'écart de performance minimal et l'algorithme correspondant sont répertoriés dans la colonne 'min { *Alg* 1, *Alg* 2, *SimGA}* '. Le temps CPU en secondes est également indiqué dans les quatre dernières colonnes pour les algorithmes et la borne inférieure.

- **Analyse des performances de la borne inférieure**

L'analyse de la borne inférieure proposée peut être effectuée de manière similaire à celle du cas de temps de configuration, et les conclusions de ces deux cas ont beaucoup en commun, comme cela sera montré. De même, les colonnes 'min { *Alg* 1, *Alg* 2, *SimGA}* »sont utilisés comme indicateur pour évaluer l'étanchéité du *KG.* Notez que sans tenir compte des temps de configuration, le *MI P* dans Theorem 4 dégénère en un programme linéaire où $\delta = [\delta_m]_{M \times T}$ est la seule variable de décision.

Comme indiqué dans les tableaux 5 et 6 , les *KG* est comparable à celle des temps de configuration en général concernant l'écart de performances. Pour les problèmes à grande échelle, les écarts de performances se réduisent encore, ce qui indique que *KG* tend à se rapprocher de l'optimum. Le plus petit est de 3,83%, et même le plus grand est toujours limité en dessous de 22,66%, ce qui est beaucoup plus petit que 29,06% du temps de configuration.

Pour être plus précis, comme les résultats de la section 6.2.1 , les *KG* n'est que légèrement affecté par l'échelle du problème, et l'effet est moins significatif que celui du cas de temps d'installation car les plus grandes différences sont respectivement de 5,74 et 13,97%. En outre,

Tableau 5. Écarts de performances et temps de calcul des algorithmes proposés et de SimGA sous *RUW.*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*SimGA*	min { *Alg* 1, *Alg* 2, *SimGA}*	*Alg* 1	*Alg* 2	*SimGA*	*KG*
3	6	6,43	6,67	9.28	6,43 (Alg1)	0,25	0,03	31.30	0,25
	8	4.27	15.02	12.00	4,27 (Alg1)	0,37	0,00	25,57	0,37
	dix	3,83	12,85	7.01	3,83 (Alg1)	0,44	0,00	25.02	0,44
4	6	7,97	8.87	13,57	7,97 (Alg1)	0,39	0,00	44,88	0,39
	8	7,59	8.09	11,39	7,59 (Alg1)	0,49	0,00	52,48	0,49
	dix	4,86	5.15	12,36	4,86 (Alg1)	0,78	0,00	44,55	0,78
5	6	8.81	23,51	25,42	8,81 (Alg1)	0,46	0,00	58.02	0,46
	8	8.48	15.16	11.14	8,48 (Alg1)	0,52	0,00	54,09	0,52
	dix	8.87	10,48	11,52	8,87 (Alg1)	0,83	0,00	53,97	0,83
dix	30	8,92	10,56	24,46	8,92 (Alg1)	2.21	0,01	181,65	2.21
	40	7.61	9.01	27,99	7,61 (Alg1)	2,34	0,01	189,26	2,34
	50	7.23	8.52	19,53	7,23 (Alg1)	2,42	0,01	196,95	2,42
15	30	12,97	14,71	34,51	12,97 (Alg1)	2,33	0,00	512,56	2,33
	40	10.20	21,49	23,87	10.20 (Alg1)	2,86	0,01	578,08	2,86
	50	10.19	17,86	29,66	10,19 (Alg1)	3,68	0,01	603.04	3,68
20	30	14,22	17,32	32.19	14,22 (Alg1)	2,85	0,01	1047,34	2,85
	40	12,96	14,91	28.04	12,96 (SimGA)	3,94	0,01	1182,34	3,94
	50	10,84	13,86	26,93	10,84 (Alg1)	6.09	0,01	1368,83	6.09

changements dans *M* et *T* conduire à la même tendance dans l'écart de performances que celle du cas de temps de configuration. Cela est probablement dû au fait que l'augmentation *T* a tendance à répartir les charges de travail de manière plus uniforme entre les machines. Un tel équilibre de la charge de travail contribue à réduire le temps d'attente potentiel encouru pendant le processus de combinaison de commandes et améliore ainsi l'étanchéité de la limite inférieure. L'augmentation de *M,* cependant, a un impact négatif. Conformément au cas de configuration, la variabilité de la charge de travail a également un effet négatif sur l'ef fi *KG* comme les résultats du tableau 5 sont plus petits que ceux du tableau 6 . Cependant, le calcul de la *KG* est beaucoup plus rapide et ne nécessite que des frais de calcul négligeables. Même pour le *(M, T), = (* 20, 50) par exemple, le temps de calcul est toujours inférieur à sept secondes. Cela est particulièrement souhaitable pour les applications où les ressources de calcul sont limitées.

- **Analyse des performances de l'algorithme**

Comme indiqué dans la section 6.1 , *Alg* 1 et *Alg* 3 sont identiques et équivalents à un programme linéaire lorsque les temps de configuration sont omis. les tables 5 et 6 révéler que *Alg* 1 (*Alg* 3) est supérieur à *Alg* 2 et *SimGA* dans la majorité des cas problématiques (2/36) concernant l'écart de performance. Par rapport aux résultats des tableaux 1 - 4 , on peut conclure que la proposition de *Alg* 1 (*Alg* 3) et *Alg* 2 ont tendance à mieux s'exécuter sans temps de configuration, bien que les différences ne soient pas très importantes pour les instances à problème à petite échelle. Pour les instances de problème à grande échelle, les différences se creusent et la plus grande différence peut être aussi importante que 14,01% (14,22% contre 28,23%). Toutefois, cela est probablement dû à la méthode de calcul de la borne inférieure au lieu de la détérioration de *Alg* 1 (*Alg* 3) et *Alg* 2 puisque 28,23% est obtenu par la relaxation de programmation linéaire associée de *MI P.* En ce qui concerne le temps de calcul, *Alg* 1 (*Alg* 3) est très ef fi cace et ne nécessite que quelques secondes, tandis que *Alg* 2 est le plus efficace et prend peu de temps de calcul.

Contrairement aux algorithmes proposés, SimGA est un algorithme génétique basé sur la simulation. Comme on peut le constater dans les tableaux 5 et 6 , SimGA présente le plus grand écart de performances dans l'ensemble, bien que comparable à *Alg* 2 pour les cas de problèmes à petite échelle. Elle se détériore de façon signi fi cative avec l'augmentation de l'échelle du problème. L'écart de performance le plus important est de 45,88%, ce qui est plus du double de celui de *Alg* 1 (*Alg* 3). Pire encore, le temps de calcul de SimGA est le plus parmi les algorithmes considérés et devient beaucoup plus lourd à mesure que l'échelle du problème augmente. Cela suggère que les algorithmes proposés sont une grande amélioration par rapport à l'algorithme existant, et capables de fournir des solutions de haute qualité ef fi cacement.

Tableau 6. Écarts de performances et temps de calcul des algorithmes proposés et de SimGA sous *HVW.*

Réglage du problème		Écart de performance (%)				Temps CPU			
M	*T*	*Alg* 1	*Alg* 2	*SimGA*	min { *Alg* 1, *Alg* 2, *SimGA*}	*Alg* 1	*Alg* 2	*SimGA*	*KG*
3	6	9.04	9.09	20.01	9,04 (Alg1)	0,26	0,00	29,33	0,26
	8	8.63	13,64	11,78	8,63 (Alg1)	0,38	0,00	24,92	0,38
	dix	7.22	10,63	11,39	7,22 (Alg1)	0,47	0,00	23h30	0,47
4	6	12.14	13.10	12,27	12.14 (Alg1)	0,44	0,00	40,77	0,44
	8	9.05	10.15	7.09	7.09 (SimGA)	0,51	0,00	45,20	0,51
	dix	6,98	7.34	16.00	6,98 (Alg1)	0,65	0,00	46.13	0,65
5	6	15h15	24.00	14,91	14,91 (SimGA)	0,50	0,00	55,74	0,50
	8	11,86	18,48	33,80	11,86 (Alg1)	0,62	0,00	54,58	0,62
	dix	11,31	13,35	18h45	11,31 (Alg1)	0,81	0,00	52,79	0,81
dix	30	12.12	13,43	35,27	12.12 (Alg1)	2.13	0,00	176,78	2.13
	40	11,81	13.05	26.05	11,81 (Alg1)	2,57	0,01	185.01	2,57
	50	9,68	10,86	17,43	9,68 (Alg1)	2,89	0,01	210,68	2,89
15	30	17,34	18,66	18,92	17,34 (Alg1)	2.27	0,00	482,87	2.27
	40	14.03	25,24	27,52	14.03 (Alg1)	2,55	0,01	538,50	2,55
	50	13,65	20,64	28,69	13,65 (Alg1)	3,31	0,01	716,32	3,31
20	30	22,66	27.09	45,88	22,66 (Alg1)	2,95	0,00	859.23	2,95
	40	19.00	20,93	31,72	19h00 (Alg1)	4.16	0,01	1104,37	4.16
	50	14,78	18.00	37.01	14,78 (Alg1)	6.38	0,01	1241,76	6.38

7. Conclusion

Un problème de planification des commandes client astochastique est considéré dans cet article avec des machines parallèles non liées et des temps de configuration dépendants de la machine et du type de produit. Plusieurs propriétés analytiques du problème sont développées. Inspiré de ces propriétés, trois algorithmes de programmation, à savoir *Alg* 1, *Alg* 2 et *Alg* 3, sont proposés avec l'analyse de la complexité de calcul correspondante. Pour évaluer la qualité des algorithmes proposés, une borne inférieure facilement calculable est construite à des fins de comparaison. Des études numériques montrent que lorsque le temps d'installation est considéré, *Alg* 3 obtient les meilleurs résultats dans la plupart des circonstances en termes d'écart de *Alg* 2 nécessite le moins d'effort de calcul. Les algorithmes proposés s'améliorent à la fois en termes d'écart de performances et de temps de calcul sans tenir compte du temps de configuration, et surpassent même un algorithme existant SimGA. Différent du modèle statique d'ordonnancement des commandes clients dans la littérature existante, l'homologue stochastique discuté dans cette étude fournit une réflexion compliquée mais plus réaliste sur le problème d'ordonnancement des commandes dans les industries manufacturières.

Cette étude fournit les informations de gestion suivantes:

- Il est hautement souhaitable d'atteindre l'équilibre entre le temps d'occupation (temps de traitement + temps de configuration) entre les machines pour réduire le temps de cycle de commande. Comme une commande client ne peut partir que lorsque ses charges de travail sur toutes les machines sont terminées, aucune machine ne doit être principalement occupée ou inactive.
- Pour atténuer l'impact du temps de configuration sur le temps de cycle de commande, il est essentiel d'organiser la séquence non seulement des types de produits au sein d'une commande client, mais également des types de produits contigus appartenant aux commandes adjacentes.
- La limite inférieure du temps de cycle proposée est facile à calculer et peut être utilisée comme référence pour l'évaluation des performances. Si le temps de cycle d'une ligne de production est proche de cette limite inférieure, cela suggère que cette ligne de production est assez ef fi cace au niveau du travail en cours (WIP).
- L'incertitude de la charge de travail peut gonfler la congestion du système et l'amélioration de la prévision de la charge de travail peut réduire considérablement le temps de cycle de commande. La réduction de la fluctuation de la demande est cruciale pour améliorer les performances globales de la chaîne de production.

Plusieurs directions peuvent être envisagées pour de futures recherches. Une direction possible serait de résoudre un problème à objectifs multiples en introduisant des mesures liées aux coûts d'installation dans la présente étude. De plus, il est également intéressant de détendre l'électricité statique

les affectations de charge de travail et incluent celles dynamiques qui peuvent changer avec les états du système. D'autres approches d'optimisation telles que les méthodes de programmation dynamique peuvent également être favorables pour fournir des solutions de haute qualité.

Remerciements

Les auteurs remercient les critiques anonymes pour leurs précieuses suggestions et commentaires.

Déclaration de divulgation

Aucun conflit d'intérêt potentiel n'a été signalé par les auteurs.

Le financement

Ce travail a été soutenu par la National Science Foundation of China (NSFC) [numéro de subvention 11471023].

Références

Ahmadi, Reza, Uttarayan Bagchi et Thomas A. Roemer. 2005. «Planification coordonnée des commandes clients pour une réponse rapide». *Naval Research Logistics (NRL)* 52 (6): 493-512.

Allahverdi, Ali, Jatinder ND Gupta et Tariq Aldowaisan. 1999. «AReview of Scheduling Research Implolving Setup Considerations.» *Oméga* 27 (2): 219-239.

Allahverdi, Ali, CT Ng, TC Edwin Cheng et Mikhail Y. Kovalyov. 2008. «A Survey of Scheduling Problems with Setup Times or Frais." *Revue européenne de recherche opérationnelle* 187 (3): 985-1032.

Allahverdi, Ali et HM Soroush. 2008. «L'importance de réduire les temps et les coûts d'installation». *Journal Européen des Opérations Recherche* 187 (3): 978–984.

Arnaout, Jean-Paul, Rami Musa et Ghaith Rabadi. 2014. «Un algorithme d'optimisation des colonies de fourmis en deux étapes pour minimiser le Makespan sur les machines parallèles non liées - Partie II: améliorations et expérimentations. » *Journal of Intelligent Manufacturing* 25 (1): 43-53. Arnaout, Jean-Paul, GhaithRabadi et JiHyonMun. 2006. «ADynamicHeuristic for theStochasticUnrelatedParallelMachineScheduling Problème." *Journal international de recherche opérationnelle* 3 (2): 136-143.

Arnaout, Jean-Paul, Ghaith Rabadi et Rami Musa. 2010. «Un algorithme d'optimisation des colonies de fourmis en deux étapes pour minimiser le Makespan sur les machines parallèles non liées avec des temps de configuration dépendants de la séquence. » *Journal of Intelligent Manufacturing* 21 (6): 693–701. Baccelli, François et Armand M. Makowski. 1989. «Modèles de file d'attente pour les systèmes avec des contraintes de synchronisation». *Actes du l'IEEE* 77 (1): 138-161.

Baccelli, François, Armand M. Makowski et Adam Shwartz. 1989. «The Fork – Join Queue and Related Systems with Synchronization Contraintes: ordre stochastique et limites calculables. » *Progrès de la probabilité appliquée* 21 (3): 629–660. Bank, Jan et FrankWerner. 2001. «HeuristicAlgorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates et pénalités de précocité linéaire et de retard. » *Modélisation mathématique et informatique* 33 (4): 363–383. Blocher, James D. et Dilip Chhajed. 1996. «Le problème de délai de commande client sur les machines parallèles». *Naval Research Logistics (NRL)* 43 (5): 629–654.

Chen, Jeng-Fung. 2009. «Planification sur des machines parallèles non liées avec des heures de configuration et une date d'échéance dépendant de la séquence et de la machine Contraintes." *Le Journal international des technologies de fabrication avancées* 44 (11): 1204-1212.

Cheng, TC Edwin, Jatinder ND Edwin et Guoqing Wang. 2000. «A Review of Flowshop Scheduling Research with Setup Times». *Gestion de la production et des opérations* 9 (3): 262-282.

Coffman Jr, EG Co, MR Garey et DS Johnson. 1996. «Algorithmes d'approximation pour l'emballage des bacs: une enquête». Dans *Approximation Algorithmes pour les problèmes NP-Hard,* édité par Dorit S. Hochbaum, 46–93. Boston, MA: PWS Publishing Co. Coffman Jr, Edward G., Michael R. Garey et David S. Johnson. 1984. «Algorithmes d'approximation pour le casier-bin - une mise à jour Sondage." Dans *Conception d'algorithmes pour la conception de systèmes informatiques,* édité par G. Ausiello, M. Lucertini et P. Serafini, 49–106. New York: Springer.

Conway, Richard W., William L. Maxwell et Louis W. Miller. 2012. *Théorie de l'ordonnancement*. NewYork: Courier Corporation. Garey, Michael R. et David S. Johnson. 1981. «ApproximationAlgorithms for Bin Packing Problems: ASurvey». Dans *Analyse et conception des algorithmes en optimisation combinatoire,* édité par G. Ausiello et M. Lucertini, 147–172. New York: Springer. Helal, Magdy, Ghaith Rabadi et AmerAl-Salem. 2006. «ATabu SearchAlgorithm toMinimize the Makespan for the Unrelated Parallel Problème de planification des machines avec les temps de configuration. " *Journal international de recherche opérationnelle* 3 (3): 182-192. Hopp, Wallace J. et Mark L. Spearman. 2008. *Physique d'usine*. Vol. 2. New York: McGraw-Hill / Irwin New York. Julien, FM et MJ Magazine. 1990. «Planification des commandes des clients - Une approche alternative de planification de la production». *Journal of Gestion de la fabrication et des opérations* 3 (3): 177-199.

Juman, ZAMS et MA Hoque. 2014. «Technique de solution AHeuristic pour atteindre les limites de coût total minimal de transport d'un Produit homogène avec des demandes et des fournitures variées. » *Revue européenne de recherche opérationnelle* 239 (1): 146-156.

Juman, ZAMS et MA Hoque. 2015. «Une heuristique ef fi cace pour obtenir une meilleure solution faisable initiale au transport Problème." *Informatique douce appliquée* 34: 813–826.

Jungwattanakit, Jitti, Manop Reodecha, Paveena Chaovalitwongse et FrankWerner. 2008. «Algorithmes pour les problèmes de FlowShop flexibles avec des machines parallèles non liées, des temps de configuration et des critères doubles. » *The International Journal of AdvancedManufacturing Technology* 37 (3-4): 354-370.

Karmarkar, Narendra. 1984. «Un nouvel algorithme à temps polynomial pour la programmation linéaire». Dans *Actes du seizième ACM annuel Symposium sur la théorie de l'informatique,* 302–311. Washington, DC: ACM. Kim, Cheeha et Ashok K. Agrawala. 1989. «Analyse de la file d'attente Fork-Join». *Transactions IEEE sur les ordinateurs* 38 (2): 250–255. Kim, SC et PM Bobrowski. 1997. «Planification des travaux avec des heures d'installation et une dépendance de séquence incertaines». *Oméga* 25 (4): 437–447.

Kim, Dong-Won, Kyong-Hee Kim, Wooseung Jang et F. Frank Chen. 2002. «Planification parallèle des machines sans rapport avec les heures de configuration Utilisation du recuit simulé. " *Robotique et fabrication intégrée par ordinateur* 18 (3): 223-231.

Kim, Dong-Won, Dong-Gil Na et F. Frank Chen. 2003. «Planification parallèle des machines sans rapport avec les temps de configuration et une pondération totale Objectif de retard. " *Robotique et fabrication intégrée par ordinateur* 19 (1): 173–181. Lee, Ik Sun. 2013. «Minimizing Total Tardiness for the Order Scheduling Problem». *Journal international d'économie de la production* 144 (1): 128–134.

Lee, Jae-Ho, Yu Jae-Min et Dong-Ho Lee. 2013. «ATabu SearchAlgorithm for Unrelated Parallel Machine Scheduling with Sequence- et configurations dépendantes de la machine: minimisation de la lenteur totale. " *Le Journal international des technologies de fabrication avancées* 69 (9-12): 2081-2089.

Leung, JosephY. T., Haibing Li et Michael Pinedo. 2005a. «Planification des commandes dans un environnement avec des ressources dédiées en parallèle.» *Journal of Scheduling* 8 (5): 355–386.

Leung, JosephY. T., Haibing Li et Michael Pinedo. 2005b. "Modèles de planification des commandes: un aperçu." Dans *Planification multidisciplinaire: Théorie et applications,* édité par G. Kendall, E. Burke, S. Petrovic et M. Gendreau, 37–53. Nottingham: Springer. Leung, Joseph YT, Haibing Li et Michael Pinedo. 2006. «Planification des commandes pour plusieurs types de produits avec échéance liée Objectifs." *Revue européenne de recherche opérationnelle* 168 (2): 370–389.

Leung, Joseph YT, Haibing Li et Michael Pinedo. 2007. «Planification des commandes pour plusieurs types de produits afin de minimiser le total pondéré Le temps d'achèvement." *Mathématiques appliquées discrètes* 155 (8): 945–970.

Lin, Shih-Wei, Lu Chung-Cheng et Kuo-Ching Ying. 2011. «Minimisation du retard total sur les machines parallèles indépendantes avec Heures de configuration dépendant de la séquence et de la machine sous contraintes de date d'échéance. » *Le Journal international des technologies de fabrication avancées* 53 (1–4): 353–361.

Lin, Yang-Kuei et Feng-Yu Hsieh. 2014. «Planification parallèle des machines sans rapport avec les temps de configuration et les temps de disponibilité». *International Journal of Production Research* 52 (4): 1200-1214.

Lodi, Andrea, Silvano Martello et Michele Monaci. 2002. «Problèmes d'emballage bidimensionnels: une enquête». *Journal européen de Recherche opérationnelle* 141 (2): 241–252.

Lodi, Andrea, SilvanoMartello et DanieleVigo. 2002. «Progrès récents concernant les problèmes d'emballage de bacs bidimensionnels». *Discrète appliquée Mathématiques* 123 (1): 379–396.

Logendran, Rasaratnam, Brent McDonell et Byran Smucker. 2007. «Planification de machines parallèles non liées avec une séquence dépendante Configurations. " *Ordinateurs et recherche opérationnelle* 34 (11): 3420–3438.

Logendran, Rasaratnam et Fenny Subur. 2004. «Planification parallèle des machines sans rapport avec le fractionnement des travaux». *Transactions IIE* 36 (4): 359–372.

Martello, Silvano, David Pisinger et Daniele Vigo. 2000. «Le problème de l'emballage en trois dimensions.» *Recherche opérationnelle* 48 (2): 256-267.

Ng, CT, TC Edwin Cheng et JJ Yuan. 2003. «Planification simultanée d'une boutique ouverte pour minimiser le nombre pondéré de tâches tardives». *Journal of Scheduling* 6 (4): 405–412.

Og, Ceyda, F. Sibel Salman et Zehra BilgintürkYalçın. 2010. «OrderAcceptance and Scheduling Decisions inMake-to-Order Systems». *Journal international d'économie de la production* 125 (1): 200-211.

Rabadi, Ghaith, Reinaldo J. Moraga et AmerAl-Salem. 2006. «Heuristique pour le problème de planification de machine parallèle sans rapport avec Heures de configuration. " *Journal of Intelligent Manufacturing* 17 (1): 85–97.

Radhakrishnan, Sanjay et JoseA.Ventura. 2000. «SimulatedAnnealing for ParallelMachineSchedulingwithEarliness-tardinessPenalties et les temps de configuration en fonction de la séquence. " *Journal international de recherche en production* 38 (10): 2233-2252. Roemer, Thomas A. 2006. «ANote sur la complexité du problème des boutiques ouvertes simultanées.» *Journal of Scheduling* 9 (4): 389–396. Stalk, George. 1988. «Time - The Next Source of Competitive Advantage». *revue de Harvard business* 66 (4): 41-51. Su, Ling-Huey, Ping-Shun Chen et Szu-Yin Chen. 2013. «Planification sur des machines parallèles pour minimiser le retard maximum pour le Problème de commande client. " *Journal international de science des systèmes* 44 (5): 926–936.

Tahar, Djamel Nait, Farouk Yalaoui, Chengbin Chu et Lionel Amodeo. 2006. «Une approche de programmation linéaire pour un parallèle identique Planification de la machine avec fractionnement des tâches et temps de configuration dépendant de la séquence. » *Journal international d'économie de la production* 99 (1): 63–73.

Wang, Guoqing et TC Edwin Cheng. 2007. «Planification des commandes des clients pour minimiser le temps de réalisation total pondéré». *Oméga* 35 (5): 623–626.

Welch, Peter D. 1983. «L'analyse statistique des résultats de simulation». *Manuel de modélisation des performances informatiques* 22: 268–328.

Weng, Michael X., Lu John et Haiying Ren. 2001. «Planification parallèle des machines sans lien avec la configuration et un total

Objectif de temps d'achèvement pondéré. » *Journal international d'économie de la production* 70 (3): 215-226. Xing, Wenxun et Jiawei Zhang. 2000. «Planification parallèle des machines avec fractionnement des travaux». *Mathématiques appliquées discrètes* 103 (1):

259-269.

Xu, Xiaoyun, Ying Ma, Zihuan Zhou et Yaping Zhao. 2013. «Planification des commandes clients sur des machines parallèles non liées pour minimiser

Temps total d'achèvement. " *Transactions de l'IEEE sur la science et l'ingénierie de l'automatisation* 12 (1): 244–257. Xu, Xiaoyun, Yaping Zhao, Haidong Li et ManxiWu. 2015. «Planification stochastique des commandes clients pour maximiser le débit». Dans *2015*

Conférence internationale de l'IEEE sur la science et l'ingénierie de l'automatisation (CASE), 665–670. Göteborg: IEEE. Xu, Xiaoyun, Yaping Zhao, Haidong Li, Zihuan Zhou et Yanni Liu. 2015b. «Planification stochastique des commandes clients par simulation -

basé sur l'algorithme génétique. " Dans *Actes de la Conférence de simulation d'hiver 2015,* 2317–2328. Huntington Beach, Californie: IEEE Press.

Xu, Xiaoyun, Yaping Zhao, Manxi Wu, Zihuan Zhao, Ying Ma et Yanni Liu. 2016. «Planification stochastique des commandes clients pour minimiser

Temps de cycle de commande prévu à long terme. " *Annales de la recherche opérationnelle* 1–24.

Yalaoui, Farouk et Chengbin Chu. 2003. «Une approche heuristique efficace pour la planification de machines parallèles avec

Temps de configuration dépendant de la séquence. " *Transactions IIE* 35 (2): 183-190.

Yang, Jaehwan. 2005. «La complexité des problèmes de planification des commandes clients sur les machines parallèles». *Ordinateurs et recherche opérationnelle*

32 (7): 1921–1939.

Yang, Jaehwan et Marc E. Posner. 2005. «Planification de machines parallèles pour le problème de commande client». *Journal of Scheduling* 8 (1):

49–74.

Zhao, Yaping, Xiaoyun Xu et Haidong Li. 2017. «Maximisation des débits des commandes clients stochastiques sous deux productions

Schémas. " *Transactions de l'IEEE sur la science et l'ingénierie de l'automatisation* 14 (2): 745–757.

Zhao, Yaping, XiaoyunXu, HaidongLi etYanni Liu. 2016. «Ordonnancement des commandes client priorisé pour maximiser le débit». *européen*

Journal of Operational Research 255 (2): 345–356.

Annexe 1. Preuve du théorème 2

De manière similaire à l'analyse du problème d'ordonnancement dynamique des commandes clients, pour le problème du flux de traitement déterministe, les relations correspondantes peuvent être établies comme suit:

$$\underline{w}_{mn+1}(r,\delta) = [w_{mn}(r,\underline{\delta}) + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) - a_{n+1}]^{+}, \forall m \in M, \tag{A1}$$

$$\underline{CT}_{mn}(r,\delta) = w_{mn}(r,\underline{\delta}) + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta), \forall m \in M, \tag{A2}$$

$$\underline{CT}_{n}(r,\delta) = \max_{1 \le m \le M} \{\overline{CT}_{mn}(r,\delta)\}. \tag{A3}$$

L'approche d'induction mathématique est adoptée dans la preuve suivante. Tout d'abord, générez un σ - algèbre S par les variables aléatoires w_{m0} et $\{une\}_{\infty n=1}$. Pour la charge de travail initiale w_{m0} sur la machine m, puisque w_{m0} sont S- mesurable, l'inégalité $w_{m0}(r,\delta) \le E[w_{m0}(r,\delta)/S], \forall m \in M$ est valable.

Supposons que pour l'ordre $n \ge 0$, l'inégalité suivante est vérifiée:

$$\underline{w}_{mn}(r,\delta) \le E[w_{m\ n}(r,\delta)/S], \forall m \in M. \tag{A4}$$

L'application de l'inégalité de Jensen entraîne

$$E[w_{m\ n+1}(r,\delta)/S] \ge [E[(w_{m\ n}(r,\delta) + \text{après-midi}_{n}(r,\delta) + \text{cuillère à café}_{m}(r,\delta) - a_{n+1})/S]]^{+} = [E[w_{m\ n}(r,\delta)/S] + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) - a_{n+1}]^{+}. \tag{A5}$$

La dernière égalité est valide car la variable aléatoire a_{n+1} est S- mesurables, et les hypothèses (ASP3) et (ASP4) indiquent que $E[pm_{n}(r,\delta)/S] = E[pm_{n}(r,\delta)] = pm(r,\delta)$ et $E[\text{cuillère à café } m_{n}(r,\delta)/S] = E[\text{cuillère à café } m_{n}(r,\delta)] = \text{cuillère à café } m(r,\delta)$.

Étant donné l'hypothèse d'induction de l'inégalité (A4), on peut conclure de l'inégalité (A5) cette

$$E[w_{m\ n+1}(r,\delta)/S] \ge [E[w_{m\ n}(r,\delta)/S] + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) - a_{n+1}]^{+} \ge [w_{mn}(r,\underline{\delta}) + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) - a_{n+1}]^{+} = w_{mn+1}(r,\delta),$$

ce qui implique que l'inégalité (A4) est vrai pour $n+1$. Le fait que (A4) est valable pour $n=0$, on peut conclure que l'inégalité (A4) est valable pour tout n.

Sur la base des relations en (3), (A2) et (A4) ainsi que les hypothèses (ASP3) et (ASP4), il peut être démontré que les éléments suivants la relation est valable:

$$\underline{CT}_{mn}(r,\delta) = w_{mn}(r,\underline{\delta}) + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) \le E[w_{m\ n}(r,\delta)/S] + pm(r,\delta) + \text{cuillère à café } m \quad n(r,\delta) = E[(w_{m\ n}(r,\delta) + \text{après-midi}_{n}(r,\delta) + \text{cuillère à café}_{m}(r,\delta))/S] = E[CT_{m\ n}(r,\delta)/S], \forall m \in M. \tag{A6}$$

Par conséquent, on peut conclure que $CT_n(r, \delta) \le E[CT_n(r, \delta) \mid S]$ basé sur l'inégalité (A6) et l'inégalité de Jensen, qui est démontrée comme suit:

$$E[CT_n(r, \delta) \mid S] = E[\max_{1 \le m \le M} \{CT_m^n(r, \delta) \mid S\}] \ge \max_{1 \le m \le M} \{E[CT_m^n(r, \delta) \mid S]\}$$

$$\ge \max_{1 \le m \le M} \{\overline{CT_{mn}}(r, \delta)\} = \overline{CT_n(r, \delta)}.$$

Avec les résultats ci-dessus, il est clair que

$$E[\underline{CT_n}(r, \delta)] \le E[E[CT_n(r, \delta) \mid S]] = E[CT_n(r, \delta)]. \quad \text{(A7)}$$

Annexe 2. Preuve du théorème 4

Étant donné que les charges de travail sont identiques dans chaque ordre et dans la stratégie (r, δ) est appliqué à toutes les commandes, les charges de travail d'une commande affectée à une machine donnée restent constantes. Laisser $T_m(r, \delta)$ désigne la somme des temps de traitement et de configuration des charges de travail allouées à la machine m sous politique (r, δ) dans chaque ordre. Ensuite, toute politique réalisable appartient à un et un seul des deux cas suivants:

Cas n ° 1: une politique dont le résultat est $T_i(r, \delta) = T_j(r, \delta), \forall\, i, j \in M$.

Il peut être observé à partir de l'équation $T_i(r, \delta) = T_j(r, \delta)$ que l'heure de début et l'heure de fin de chaque commande entrante sont les mêmes pour toutes les machines. Par conséquent, le cas de la charge de travail déterministe est égal à *MARYLAND*/1 file d'attente. Étant donné que makespan (c'est-à-dire la somme maximale du temps de traitement et du temps de configuration) est minimisé par *MIP*, il est clair que *MIP* minimise également le temps de cycle. Par conséquent, $OBJ(r_{MIP}, \delta_{MIP}) \le OBJ(r, \delta)$.

Cas n ° 2: stratégie dans laquelle au moins une telle paire de machines (p, q) existe que $T_p(r, \delta) = T_q(r, \delta)$.

Si l'écart de temps de réalisation entre une paire de machines (p, q) est plus grand que sp_{ij} ou sq_{ij} où type de produit je et type de produit j sont les derniers sur la machine p et q respectivement, il doit exister au moins une machine ayant un temps d'achèvement plus long que la durée de production générée par *MIP* ce qui conduit au même temps d'achèvement de la machine p et machine q. Si l'écart est inférieur à sp_{ij} et sq_{ij}, puis le affectation générée par *MIP* sera optimal pour toute commande individuelle. Cependant, étant donné que la politique sera appliquée à toutes les commandes, tous les types de produits sont disponibles en même temps, et la séquence de types de produits et son inverse sont exécutés en alternance afin qu'il n'y ait pas de temps de configuration entre les commandes pour une machine, la dernière machine terminée. dominera toujours. C'est-à-dire que le temps de cycle est décidé par celui dont le temps de réalisation le plus long est égal à la durée de génération générée par *MIP*. Puisque *MIP* minimise le makespan, il est trivial de montrer *MIP* minimise le temps de cycle. Par conséquent, $OBJ(r_{MIP}, \delta_{MIP}) \le OBJ(r, \delta)$.

Sur la base des preuves des deux cas ci-dessus, on peut conclure que la politique (r_{MIP}, δ_{MIP}) généré par *MIP* est optimal dans le cas de la charge de travail déterministe.

Annexe 3. Organigramme de *Alg* 1, *Alg* 2 et *Alg* 3

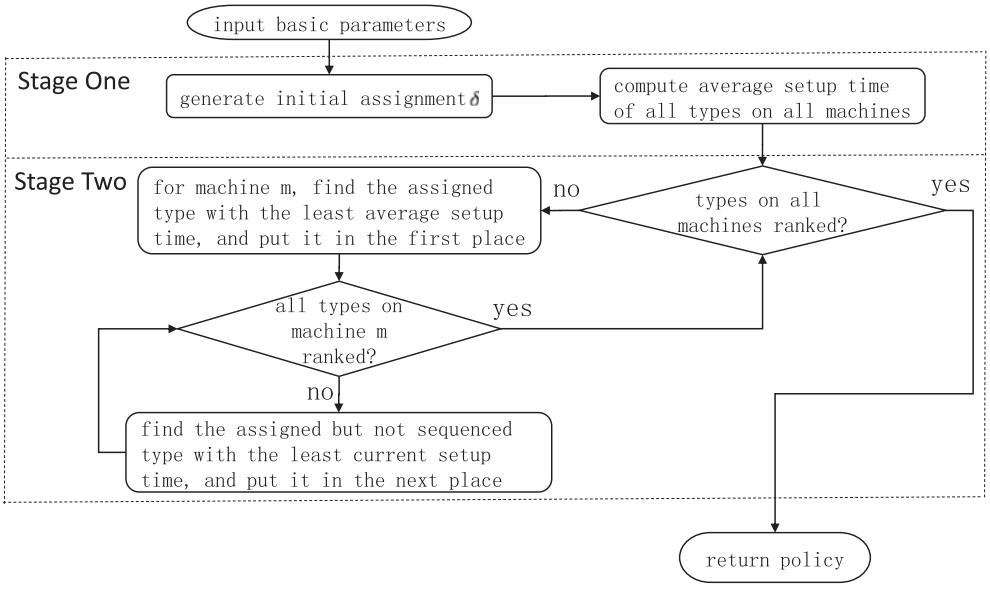

Figure C1. Organigramme de *Alg* 1.

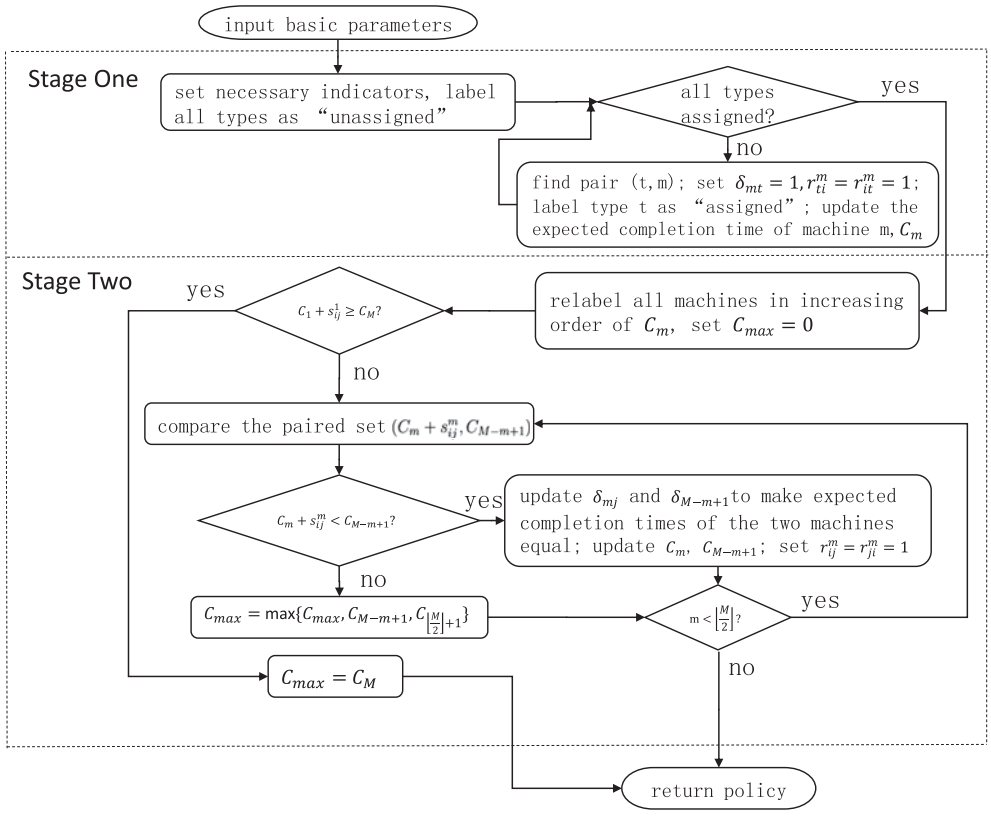

Figure C2. Organigramme de *Alg* 2.

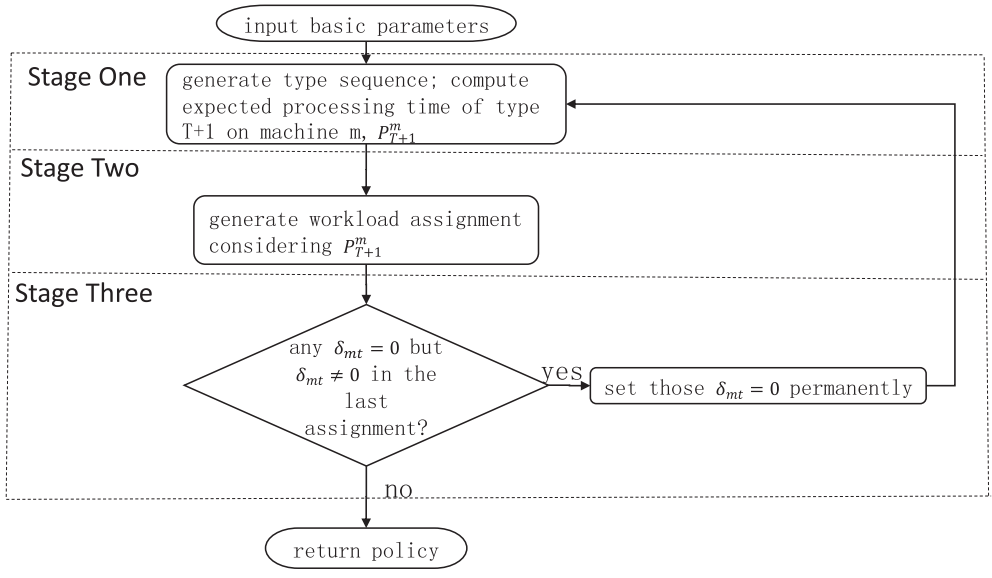

Figure C3. Organigramme de *Alg* 3.

Annexe 4. Preuve de lemme 1

Dans *Alg* 1, la résolution du programme linéaire nécessite $O((MT)^{3,5} L_2 Alg 1)$ opérations élémentaires où $L Alg 1$ est le nombre de bits dans l'entrée de *Alg* 1 (Karmarkar 1984). Le calcul du temps d'installation moyen nécessite $T \times M = O(TM)$ opérations d'affectation. Ensuite, la disposition des types de produits sur chaque machine nécessite au plus $(T^2 + T - 2) / 2 = O(T^2)$ comparaisons et $MT^2 = O(MT^2)$ les opérations d'affectation, et par conséquent un total de $M(O(T^2) + O(MT^2)) = O(M^2 T^2)$ des opérations sont nécessaires pour toutes les machines. Par conséquent, la complexité de calcul de *Alg* 1 est délimité par $O((MT)^{3,5} L_2 Alg 1) + O(TM) + O(M^2 T^2) = O((MT)^{3,5} L_2 Alg 1)$. Dans *Alg* 2, $2 MT + MT^2 + M = O(MT^2)$ des opérations d'affectation sont nécessaires dans un premier temps. Ensuite, un total de $TM(T + 1) / 2 - 1$ comparaisons et au plus 3 *T* des opérations d'affectation sont nécessaires pour affecter tous les types, dont la complexité de calcul est $O(MT^2)$. À la deuxième étape, la détermination des rangs des machines nécessite *M* Journal *M* opérations, et le processus suivant nécessite au plus $5 M / 2 + 1$ opérations. Par conséquent, la complexité de calcul de *Alg* 2 est délimité par $O(MT^2) + O(MT^2) + O(M \text{ Journal } M) + O(M) = O(MT^2)$. Dans *Alg* 3, initialement $O(MT)$ des opérations d'affectation sont nécessaires. La première étape nécessite $M(T^2 + T - 2) / 2 = O(MT^2)$ comparaisons et $MT^2 + M = O(MT^2)$ opérations d'affectation, et donc un total de $O(MT^2) + O(MT^2) = O(MT^2)$ opérations. La deuxième étape nécessite $O((MT)^{3,5} L_2 Alg 3)$ opérations où $L Alg 3$ est le nombre de bits dans l'entrée de *Alg* 3 (Karmarkar 1984). Si la troisième étape est menée, les première et deuxième étapes se répéteront au plus $T(M - 1)$ fois, et à chaque fois au plus 2 *MT* des comparaisons et six opérations d'affectation sont nécessaires pour la troisième étape. Par conséquent, la complexité de calcul de *Alg* 3 est délimité par $T(M - 1)(O(MT^2) + O((MT)^{3,5} L_2 Alg 3) + O(MT)) = O((MT)^{4.5} L_2 Alg 3)$.