Chapitre 5

Ordonnancement d'une machine à traitement par fournées

Nous rappelons la définition et la classification des problèmes de fournées ainsi que les principaux résultats de complexité. Nous nous intéressons particulièrement aux problèmes de fournées pour lesquels la capacité de la machine est bornée et les dates de disponibilité des tâches sont identiques. Nous établissons ensuite un état de l'art sur ces problèmes.

Sommaire

L'ordonnancement de fournées consiste à exploiter au mieux des moyens limités, une machine à traitement par fournées, pour organiser un ensemble de tâches. La complexité de ce problème ne réside pas seulement dans l'ordonnancement des fournées, mais aussi dans leur construction. Ce chapitre dresse un panorama de la classification et de la résolution des problèmes de fournées. L'objectif principal est de donner les clés pour la compréhension du chapitre 8 présentant notre approche pour le problème de base. Ce chapitre est organisé de la manière suivante. La section 5.1 définit le problème de base et introduit la classification des problèmes de fournées. Ensuite, les sections 5.2 et 5.3 récapitulent respectivement les principaux résultats de complexité et méthodes de résolution. Finalement, la section 5.4 situe nos axes de recherche par rapport à la littérature.

5.1 Définition des problèmes de fournées

Nous donnons dans cette section une définition des problèmes de fournées et introduisons leur classification avant d'illustrer ces problèmes par quelques applications réelles.

5.1.1 Définition du problème de base

Le problème de base consiste à trouver un ordonnancement de n tâches sur une machine à traitement par fournées minimisant une fonction objectif régulière. Une machine à traitement par fournées peut exécuter simultanément plusieurs tâches. En ordonnancement, une fonction objectif est dite régulière lorsqu'elle est croissante en fonction des dates d'achèvement des tâches. Des tâches exécutées simultanément forment une fournée. Toutes les tâches d'une fournée débutent et s'achèvent aux mêmes instants, car leurs dates de début et de fin sont celles de la fournée à laquelle elles appartiennent. Aucune tâche ne peut être ajoutée ou retirée de la fournée pendant son exécution. Nous supposerons que les tâches et la machine sont disponibles depuis l'instant 0 ou, ce qui est équivalent, que leurs dates de disponibilité sont identiques.
Plus précisément, nous nous intéresserons au modèle *burn-in* ou *paralell-batch* ou encore *p-batch*, dans lequel la durée d'exécution d'une fournée est égale à celle de la plus longue tâche appartenant à cette fournée. Il existe deux variantes du modèle *p-batch* : le modèle non borné (*unbounded model*), dans lequel $b > n$ implique qu'une fournée peut contenir un nombre quelconque de tâches ; le modèle borné (*bounded model*), dans lequel $b < n$ implique qu'une fournée peut contenir un nombre limité de tâches. Le cas particulier $b = 1$ est équivalent à problème d'ordonnancement sur une machine, puisque la machine ne peut exécuter qu'une tâche à chaque instant. De ce fait, le modèle borné donne naissance à des problèmes qui sont au moins aussi difficiles que leurs homologues traditionnels à une machine.
Dans cette thèse, nous considérons une extension du modèle borné dans laquelle les tâches ont des tailles différentes définie formellement de la manière suivante. On considère un ensemble J de n tâches et une machine à traitement par fournées de capacité b. Chaque tâche j est caractérisée par un triplet d'entiers positifs (p_j, d_j, s_j), où p_j est sa durée d'exécution, d_j est sa date échue, et s_j est sa taille. La machine à traitement par fournées peut traiter plusieurs tâches simultanément tant que la somme de leurs tailles ne dépasse pas sa capacité b. Toutes les données sont déterministes et connues a priori. La date de fin C_j d'une tâche est la date de fin de la fournée à laquelle elle appartient. Le critère d'optimalité étudié est la minimisation du retard algébrique maximal : $L_{max} = \max_{1 \leq j \leq n}(C_j - d_j)$. Ce problème est NP-difficile *au sens fort* puisque Brucker *et al.* [124] ont prouvé que le même problème avec des tâches de tailles identiques était NP-difficile *au sens fort*. Le problème étudié dans cette thèse est noté $1|p\text{-}batch; b < n; non\text{-}identical|L_{max}$ en suivant la classification présentée dans la section suivante.
La figure 5.1 illustre un ordonnancement optimal pour un notre problème de fournées où le retard maximal est égal à -90, c'est-à-dire que toutes les tâches sont en avance. La machine à traitement par fournées est représentée par un diagramme dans lequel les axes horizontaux et verticaux correspondent respectivement au temps et à la charge de la ressource. La capacité de la machine à traitement par fournées est $b = 10$.

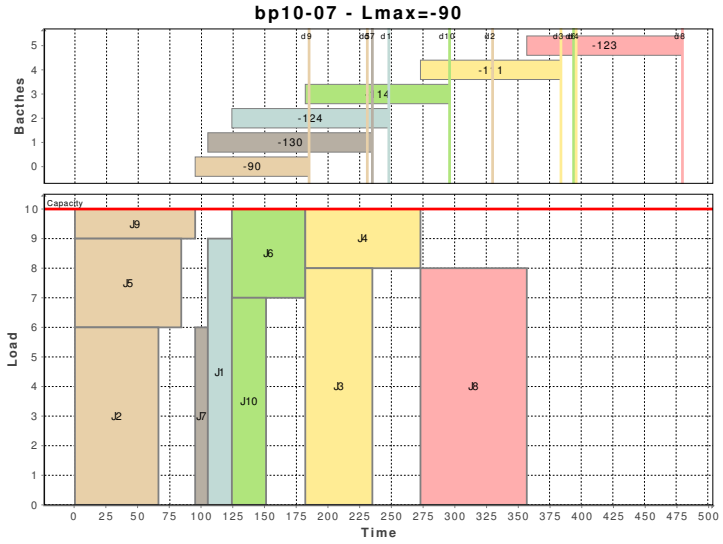

FIGURE 5.1 – Un ordonnancement optimal de retard maximal -90 pour l'instance `bp10-07`.

Une tâche est dessinée comme un rectangle dont la longueur et la hauteur représentent respectivement sa durée et sa taille. Les tâches dont les dates de début sont identiques appartiennent à la même fournée. La solution contient donc six fournées. Le retard algébrique d'une fournée est représenté dans la partie haute de la figure par un rectangle allant de sa date échue à sa date d'achèvement. Dans cet exemple, la première fournée contenant les tâches 2, 5 et 9 détermine la valeur de l'objectif.

5.1.2 Variantes et classification de Lawler

Nous présentons une classification des problèmes de fournées et complétons la notation $\alpha|\beta|\gamma$ présentée en section 4.1 pour ces problèmes.

- α permet de spécifier l'environnement machine.
  - $\alpha = 1$, car nous ne considérons que des problèmes à une seule machine.
- β décrit les caractéristiques des tâches et des diverses autres restrictions : $\beta = \beta_1; \beta_2; \beta_3; \beta_4$. Nous ne discuterons que des problèmes sans contrainte de précédence (*prec*), ni date de disponibilité (r_j).
  - β_1 = *p-batch* ou *parallel-batch* lorsque la durée d'exécution d'une fournée est égal à la plus longue durée d'exécution de ses tâches, sinon *s-batch* ou *serial-batch* lorsque la durée d'exécution d'une fournée est égale à la somme des durées d'exécution de ses tâches.
  - $\beta_2 = b < n$ dans le cas du modèle borné (*p-batch*), o dans le cas du modèle non borné.
  - $\beta_3 = (p_j = p)$ si les tâches ont des durées identiques, $(p_j = 1)$ si les tâches ont des durées unitaires, o sinon. Outre les modèles *p-batch* et *s-batch*, Webster et Baker [125] considère un troisième modèle correspondant à *p-batch*$; p_j = p$, dans lequel la durée d'exécution d'une fournée est constante et indépendante de ses tâches.
  - β_4 = *non-identical* si les tâches ont des tailles différentes dans le modèle borné ($b < n$), o sinon.
- γ est un des critères d'optimalité présenté en section 4.1. Nous nous limiterons aux critères d'optimalité pour lesquels l'homologue à une machine $1||\gamma$ peut être résolu en temps polynomial : f_{max}, C_{max}, $\sum C_j$, $\sum w_j C_j$, L_{max}, $\sum U_j$.

Dans le modèle en série (*s-batch*), l'exécution de tâches avec des caractéristiques différentes entraîne un temps d'attente $s > 0$ sur la machine à traitement par fournées. Ces temps d'attente représentent par exemple un nettoyage de la machine ou un changement d'outils. Le *family scheduling model* est une variante du modèle en série (*s-batch*) assez répandue où les tâches sont partitionnées en familles pour lesquelles il n'y a aucun temps d'attente entre deux tâches d'une famille. Une fournée correspond alors à une séquence maximale de tâches sans temps d'attente. Le lecteur intéressé pourra consulter l'état de l'art de Potts et Kovalyov [126] pour de plus amples informations sur les problèmes de fournées.

5.1.3 Applications

La recherche sur les problèmes de fournées est motivée par l'industrie (chimie, pharmacie, aéronautique et électronique), où l'utilisation de fours, de séchoirs ou encore d'autoclaves [1] est fréquente. Par exemple, Lee *et al.* [127] utilise le modèle *p-batch* en réponse aux problématiques existantes dans la fabrication à grande échelle de circuits intégrés.

Le problème étudié dans cette thèse est issu de l'industrie aéronautique. Les pièces en matériaux composites utilisées dans cette industrie sont constituées de deux catégories de matériaux : une matrice (résine *epoxy*) et un renfort (fibre de carbone). Elles sont fabriquées selon le processus suivant : la matrice est recouverte de fibre de carbone, puis la pièce résultante est consolidée à haute température et à haute pression dans un autoclave (la machine à traitement par fournées). Différentes pièces de tailles différentes (les tâches) peuvent être traitées simultanément dans un même autoclave.

5.2 Résultats de complexité

Nous rappelons les principaux résultats de complexité pour les problèmes de machine à traitement par fournées sans contrainte de précédence, ni date de disponibilité.

1. Un autoclave est un récipient à fermeture hermétique destiné à la cuisson ou à la stérilisation.

5.2.1 Modèle en parallèle

5.2.1.1 Modèle non borné

Nous supposons ici que la machine à traitement par fournées peut exécuter simultanément un nombre quelconque de tâches ($1|p\text{-}batch|\gamma$). Le problème de minimisation du délai total est trivialement résolu en plaçant toutes les tâches dans la même fournée. Le délai total est alors la valeur maximale des durées d'exécution des tâches. Brucker *et al.* [124] donnent une caractérisation d'une classe dominante d'ordonnancement pour les critères réguliers : supposons que les tâches sont triées par durée non décroissante (*shortest processing time (SPT-)rule*), il existe au moins un ordonnancement optimal qui peut être déterminé en précisant uniquement les tâches qui commencent chaque fournée, car il est possible de reconstruire la séquence complète de fournées en suivant la règle SPT. Un algorithme générique en programmation dynamique se base sur cette caractérisation pour la minimisation d'une fonction de coût de la forme $\sum f_j$. Les complexités temporelles et spatiales de cet algorithme sont pseudo-polynomiales et valent respectivement $O(n^2P)$ et $O(nP)$ où P est la somme des durée d'exécution des tâches. Le problème NP-difficile *au sens fort* $1|p\text{-}batch|\sum f_j$ est moins complexe que son homologue à une machine $1|p\text{-}batch|\sum f_j$ qui est NP-difficile *au sens fort*. Cette caractérisation est à la base d'algorithmes polynomiaux pour certaines fonctions objectif. Les auteurs proposent un algorithme de complexité $O(n^2)$ pour minimiser le délai moyen pondéré $\sum w_jC_j$, un autre de complexité $O(n^3)$ pour la minimisation du nombre de jobs en retard $\sum U_j$ et un dernier de complexité $O(n^2)$ pour minimiser le retard algébrique maximal L_{max}. Ce dernier algorithme est utilisé comme une sous-procédure pour construire un algorithme polynomial minimisant le coût maximal d'une fonction de coût régulière f_{max}.

5.2.1.2 Modèle borné

Concernant le modèle borné ($1|p\text{-}batch; b < n|\gamma$), Brucker *et al.* [124] ont montré qu'il existe un algorithme de complexité $\min\{O(n\log n), O(\frac{n^2}{b}\}$ pour la minimisation du délai total C_{max} et un algorithme de programmation dynamique de complexité pseudo-polynomiale $O(n^{b(b-1)})$ pour la minimisation du délai moyen $\sum C_j$. En outre, Brucker *et al.* montrent que les critères d'optimalité L_{max}, f_{max} et $\sum U_j$ engendrent des problèmes NP-difficiles *au sens fort*. Cependant, lorsque toutes les tâches ont des durées identiques ($p_j = p$), Baptiste [128] montre que les critères d'optimalité L_{max}, $\sum C_j$, $\sum w_jC_j$ et $\sum U_j$ engendrent des problèmes polynomiaux. Pour finir, la complexité engendrée par les critères d'optimalité $\sum C_j$ pour b quelconque et $\sum w_jC_j$ pour b fixé ou quelconque reste une question ouverte.

Pour le modèle borné où les tâches de tailles différentes ($1|p\text{-}batch; b < n; non\text{-}identical|\gamma$), tous les critères d'optimalité engendrent des problèmes NP-difficiles *au sens fort*. Uzsoy [129] a prouvé que les critères C_{max} et $\sum C_j$ engendrent des problèmes NP-difficiles *au sens fort*. Le calcul de la complexité engendrée par les critères d'optimalité L_{max}, f_{max} et $\sum U_j$ est immédiat puisque nous venons de voir que les mêmes problèmes avec des tâches de tailles identiques étaient NP-difficiles *au sens fort*.

5.2.2 Modèle en série

Pour le modèle en série ($1|s\text{-}batch|\gamma$), le problème de minimisation du délai total est trivialement résolu en plaçant toutes les tâches dans la même fournée. Ainsi, le délai total est la somme des durée d'exécution des tâches. Coffman *et al.* [130] ont proposé un algorithme de complexité $O(n\log n)$ pour la minimisation du délai moyen $\sum C_j$. Ng *et al.* [131] ont proposé un algorithme de complexité $O(n^2)$ pour la minimisation du retard algébrique maximal L_{max}. Brucker et Kovalyov [132] ont proposé un algorithme de complexité $O(n^3)$ pour la minimisation du nombre de tâches en retard $\sum U_j$. Par contre, le délai moyen pondéré $\sum w_jC_j$ engendre un problème NP-difficile *au sens fort*. Cependant, Baptiste [128] ont montré que ce problème devient polynomial lorsque les tâches ont des durées identiques ($p_j = p$). À notre connaissance, la complexité engendrée par le critère d'optimalité f_{max} reste une question ouverte.

5.2.3 Conclusion

En conclusion, le tableau 5.1 récapitule les résultats de complexité pour les problèmes de fournées sans contrainte de précédence ni date de disponibilité en se restreignant aux critères d'optimalité pour

lesquels les homologues à une machine sont polynomiaux ($b = 1$ voir Lawler *et al.* [88]). Pour le modèle non borné, tous les critères d'optimalité engendrent des problèmes polynomiaux. Pour le modèle borné, certains critères engendrent des problèmes polynomiaux lorsque les tailles sont identiques, mais tous deviennent NP-difficiles *au sens fort* lorsque les tâches ont des tailles différentes. Pour le modèle en série, le délai moyen pondéré engendre un problème NP-difficile *au sens fort*, alors que les autres critères engendrent des problèmes polynomiaux. Par ailleurs, Brucker *et al.* [124] a montré que la présence de dates échues ou d'échéances engendre des problèmes NP-difficiles *au sens fort* pour le modèle borné même lorsque $b = 2$.

Fonction Objectif	*parallel-batch*				*serial-batch*
	Non borné	Borné			
	$b > n$	$b = 1$	$b > 1$		
			identical	*non-identical*	
f_{max}	polynomial	$O(n^2)$	NP-difficile	NP-difficile	open
C_{max}	$O(n)$	$O(n)$	$\min\{O(n \log n), O(\frac{n^2}{b}\}$	NP-difficile	$O(n)$
L_{max}	$O(n^2)$	$O(n \log n)$	NP-difficile	NP-difficile	$O(n^2)$
$\sum C_j$	$O(n \log n)$	$O(n \log n)$	$O(n^{b(b-1)})$	NP-difficile	$O(n \log n)$
$\sum w_j C_j$	$O(n \log n)$	$O(n \log n)$	open	NP-difficile	NP-difficile
$\sum U_j$	$O(n^3)$	$O(n \log n)$	NP-difficile	NP-difficile	$O(n^3)$

TABLE 5.1 – Complexité des problèmes de fournées sans contrainte de précédence ni date de disponibilité.

5.3 État de l'art

Les problèmes de fournées n'ont été abordés que récemment dans la littérature. Les premiers résultats ont surtout concerné la complexité de ces problèmes et des algorithmes d'approximation, mais un certain nombre d'approches ont été proposées depuis.

5.3.1 Jeu d'instances

À notre connaissance, il n'existe aucun jeu d'instances standard pour les problèmes de fournées. Dans cette thèse, nous utilisons le jeu d'instances pour $1|p\text{-}batch; b < n; non\text{-}identical|L_{max}$ proposé par Daste *et al.* [133] comprenant entre 10 et 100 tâches. Dans ce jeu d'instances, les durées d'exécution sont uniformément distribuées dans l'intervalle entier $[1, 99]$ ce qui correspond à des applications dans la fabrication de semi-conducteurs. Les tailles des tâches sont uniformément distribuées dans l'intervalle entier $[1, 10]$ et la machine à traitement par fournées a une capacité b égale à 10 (inspiré par les travaux de Ghazvini et Dupont [134]). Pour une instance donnée, on détermine d'abord les durées d'exécution et les tailles de toutes les tâches avant de générer les dates échues des tâches en se basant sur une formule inspirée par les travaux de Malve et Uzsoy [135] : $d_j = U[0, \alpha] \times \tilde{C}_{max} + U[1, \beta] \times p_j$ où $\tilde{C}_{max} = (\sum_{j=1}^{n} s_j \times \sum_{j=1}^{n} p_j) \div (b \times n)$ est une approximation (borne inférieure) du délai total nécessaire à l'exécution de toutes les tâches sur la machine à traitement par fournées. Pour un nombre de tâches $n \in \{10, 20, 50, 75, 100\}$, 40 instances ont été générées avec les paramètres $\alpha = 0.1$ et $\beta = 3$.

5.3.2 Méthodes de résolution

La première étude semble être la publication de l'article d'Ikura et Gimple [136] en 1986, qui présente un algorithme exact en $O(n)$ pour le problème $1|p\text{-}batch; b < n; r_j; p_j = p|C_{max}$, c'est à dire un problème où les durées d'exécution et les tailles des tâches sont identiques, les tâches ont des dates de disponibilité, et le critère d'optimalité est le délai total de l'ordonnancement. Depuis, différentes approches ont été proposées pour la minimisation de critères d'optimalité dépendant des dates échues dans un problème où les tâches ont une taille identique : des heuristiques [137–139] ; un algorithme génétique [138] ; des

méthodes exactes [125, 140, 141]. Plusieurs articles traitent aussi de problèmes avec des tâches de tailles différentes, mais les critères d'optimalité étudiés ne dépendent que des dates d'achèvement des tâches (C_{max}, $\sum C_j$, $\sum w_j C_j$) : des heuristiques [129, 142] ; un algorithme génétique [143] ; une méthode par recuit simulé [144] ; des méthodes exactes [129, 134, 142, 145–149].
Malgré le succès de la programmation par contraintes sur des problèmes d'ordonnancement variés, il existe, à notre connaissance, une seule approche par contraintes pour ce type de problème : les algorithmes de filtrage de Vilím [150] pour un problème *s-batch* avec des familles de tâches et des temps d'attente dépendant de la séquence. Ces algorithmes étendent les algorithmes de filtrage de la contrainte de partage de ressource disjonctive (voir section 3.3.1), qui raisonne principalement sur les dates d'achèvement des tâches et le délai total de l'ordonnancement.
À notre connaissance, seuls deux articles [133, 151] traitent de la résolution d'un problème avec des tâches de tailles différentes et un critère d'optimalité dépendant des dates échues. Dans ces travaux, Daste *et al.* proposent une formulation en programmation mathématique basée sur la construction et le séquencement des fournées [133], puis une approche par *branch-and-price* [151]. Une approche par branch-and-price intègre un branch-and-bound et des méthodes de génération de colonne pour résoudre des problèmes en nombres entiers de grande taille [152]. L'approche par branch-and-price est basée sur un problème maître où chaque colonne représente une fournée. Une solution du problème maître est une séquence réalisable de fournées. À chaque itération, on résout un sous-problème pour déterminer une colonne (fournée) qui améliore la solution du problème maître. Ce sous-problème est résolu par un algorithme glouton, puis si nécessaire, par une méthode exacte d'énumération.
Pour une synthèse plus poussée sur les problèmes de fournées, les lecteurs intéressés pourront se reporter à l'article de Potts et Kovalyov [126].

5.4 Orientation de nos travaux

Les problèmes avec des tâches de tailles différentes et un critère d'optimalité dépendant des dates échues n'a jamais été abordé en programmation par contraintes, notre premier axe de recherche concerne donc la modélisation de notre problème. Nous nous efforcerons de proposer un modèle concis et élégant en nous appuyant sur la richesse du langage déclaratif de modélisation et la flexibilité des algorithmes de recherche.
Une particularité du problème de fournées étudié dans cette thèse est que son homologue à une machine, noté $1||L_{max}$, peut être résolu en temps polynomial [153] : un ordonnancement optimal est obtenu en appliquant la règle de Jackson, aussi appelée règle EDD (*earliest due date (EDD-)rule*) qui séquence les tâches en ordre non décroissant de date échue [154]. Nous avons vu dans le chapitre 2 que les contraintes globales sont souvent utilisées pour modéliser un sous-problème de manière concise et efficace. Elles raisonnent généralement sur la faisabilité du sous-problème, mais dans le cadre d'un problème d'optimisation, les réductions de domaine peuvent également être réalisées sur la base des coûts. On supprime pendant la propagation des combinaisons de valeurs qui ne peuvent pas être présentes dans une solution améliorant la meilleure solution (ou borne supérieure) découverte jusqu'à présent. Focacci *et al.* [155] ont proposé d'embarquer ces raisonnements dans une contrainte globale qui représente une relaxation d'un sous-problème ou d'elle-même. Ils ont montré l'intérêt d'utiliser cette information pour le filtrage, mais aussi pour guider la recherche sur plusieurs problèmes d'optimisation combinatoire. Notre second axe de recherche consiste à exploiter l'homologue à une machine de notre problème de fournées pour améliorer la résolution de notre modèle.