Approche hybride multi-étape pour la résolution du problème de job-shop flexible robuste avec budget d'incertitude

Carla Juvin, Laurent Houssin, Pierre Lopez

Approche hybride multi-étape pour la résolution du problème de job-shop flexible robuste avec budget d'incertitude

Carla Juvin[1], Laurent Houssin[1,2], Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{carla.juvin,pierre.lopez}@laas.fr
[2] ISAE-SUPAERO, Université de Toulouse, France
laurent.houssin@isae-supaero.fr

Mots-clés : *Problème d'ordonnancement jop-shop flexible, optimisation robuste multi-étape, budget d'incertitude, programmation linéaire en nombres entiers, programmation par contraintes.*

1 Introduction

En ordonnancement, le problème de job-shop fait l'objet d'une vaste littérature. Il s'inscrit dans la catégorie des problèmes d'optimisation NP-difficiles [7]. Pour rappel, dans un problème de job-shop, un travail (ou job) est composé d'une séquence d'opérations devant être exécutées sans interruption sur des machines suivant une gamme propre au job et chaque machine ne peut réaliser qu'une opération à la fois. Le problème de job-shop flexible (FJSSP) est une généralisation du job-shop dans le sens où chaque opération peut être réalisée sur un ensemble de machines et il faut décider quelle machine allouer à l'opération.

Le FJSSP a également été largement étudié, la majorité des travaux traitant de problèmes à paramètres parfaitement déterminés [3, 4]. Toutefois, en pratique, de nombreuses sources d'incertitudes existent, par exemple sur la durée des opérations, les pannes machines et globalement le fonctionnement de l'atelier ou l'état d'avancement de la fabrication des produits.

Pour prendre en compte ces incertitudes, on trouve principalement deux types d'approches : l'optimisation stochastique qui s'appuie sur des distributions de probabilités sur la valeur des paramètres et l'optimisation robuste qui considère des scénarios d'incertitudes et qui cherche à optimiser des solutions dans le pire des cas sur ces scénarios.

Dans ce travail, nous nous intéressons à la conception de méthodes exactes pour résoudre le FJSSP robuste. Nous proposons pour cela une optimisation robuste en deux étapes, la première étape s'occupant de l'affectation des opérations et de leur séquencement sur les machines, la seconde déterminant la date de début des opérations.

Deux modèles robustes, basés sur des formulations de programmation linéaire en nombres entiers mixtes et de programmation par contraintes, sont présentés, de même qu'un algorithme de génération de colonnes et de contraintes. Des résultats expérimentaux permettent de tirer des conclusions sur la performance des méthodes proposées.

2 Position du problème

2.1 Job-shop flexible déterministe

Nous définissons ici le FJSSP avec l'objectif de minimisation de la durée totale (*makespan*). Une instance du FJSSP consiste en un ensemble de jobs $\mathcal{J}$ et un ensemble de machines $\mathcal{M}$. Chaque job est composé d'une séquence de n_i opérations. La $j^{ème}$ opération $O_{i,j} \in \mathcal{O}_i$ d'un job i doit être exécutée sur une machine parmi l'ensemble de machines éligibles $\mathcal{M}_{i,j} \subseteq \mathcal{M}$. Soit

$\mathcal{I}_m$ l'ensemble d'opérations pouvant être réalisées sur la machine m ($m \in \mathcal{M}$). On note $p_{i,j,m}$ la durée de l'opération $O_{i,j}$ exécutée sur la machine $m \in \mathcal{M}_{i,j}$. Chaque machine peut traiter au plus une opération à la fois, sans interruption (cas non préemptif). L'objectif est de trouver une affectation et une séquence d'opérations sur chaque machine de manière à minimiser le makespan $C_{\max}$.

2.2 Job-shop flexible avec durées incertaines

Pour une opération $O_{i,j}$ sur la machine m, on a : $p_{i,j,m} = [\bar{p}_{i,j,m}; \bar{p}_{i,j,m} + \hat{p}_{i,j,m}]$, où $\bar{p}_{i,j,m}$ est la valeur nominale et $\hat{p}_{i,j,m}$ la déviation maximale de la durée par rapport à sa valeur nominale.

Exemple 1 *Soit une instance du FJSSP avec 3 jobs et 2 machines. Le tableau 1 décrit les intervalles des durées des opérations sur chaque machine éligible.*

		$M1$	$M2$
$J1$	$O_{1,1}$	[43 ; 86]	–
	$O_{1,2}$	[87 ; 170]	[95 ; 190]
$J2$	$O_{2,1}$	[63 ; 142]	[53 ; 166]
	$O_{2,2}$	–	[73 ; 131]
$J3$	$O_{3,1}$	[125 ; 239]	[135 ; 224]
	$O_{3,2}$	[43 ; 73]	[61 ; 174]

TAB. 1 – Durées opératoires pour une instance du FJSSP

2.2.1 Budget d'incertitude

Afin d'établir un compromis entre la robustesse d'une solution obtenue et sa qualité, nous définissons un ensemble d'incertitude en exploitant une approche proposée dans [2], basée sur la notion de budget d'incertitude qui permet une restriction sur le nombre de déviations pouvant se produire en même temps.

Soit ξ un scénario et soit Γ le budget d'incertitude, c'est-à-dire le nombre maximum d'opérations dont la durée peut varier dans un même scénario. On a :

$$p_{i,j,m}(\xi) = \bar{p}_{i,j,m} + \xi_{i,j} \cdot \hat{p}_{i,j,m} \tag{1}$$

où $\xi_{i,j}$ est égal à 1 si la durée de l'opération dévie, 0 sinon.

On définit alors l'ensemble d'incertitude $\mathcal{U}^\Gamma$:

$$\mathcal{U}^\Gamma = \{(\xi_{i,j})_{i \in \mathcal{J}, 1 \leq j \leq n_i} \mid \sum_{i \in \mathcal{J}} \sum_{j=1}^{n_i} \xi_{i,j} \leq \Gamma\} \tag{2}$$

2.2.2 Optimisation robuste multi-étapes

Dans l'optimisation robuste multi-étapes introduite dans [1], une partie des variables de décision doit être instanciée avant la révélation de l'incertitude, tandis que les autres variables peuvent être ajustées au scénario. Appliqué à notre problème, l'objectif est de trouver une affectation des opérations et leur séquence sur les machines permettant de définir une date de début pour chaque opération et chaque scénario, et minimisant le makespan dans le pire des cas en considérant le budget d'incertitude.

Exemple 2 *En reprenant l'exemple 1, considérons une solution admissible où les opérations $O_{1,1}$, $O_{3,1}$, $O_{3,2}$ sont affectées à la machine M_1, et les opérations $O_{2,1}$, $O_{2,2}$, $O_{1,2}$ à la machine M_2, suivant ces ordres d'opérations sur chacune des machines. La figure 1 représente cette*

solution lorsque toutes les durées prennent leur valeur nominale ; le makespan est de 221. En considérant un budget d'incertitude $\Gamma = 2$, le pire cas pour cette solution est rencontré lorsque les durées des opérations $O_{2,1}$ et $O_{1,2}$ dévient et prennent leur plus grande valeur. La figure 2 illustre ce cas montrant un makespan de 429.

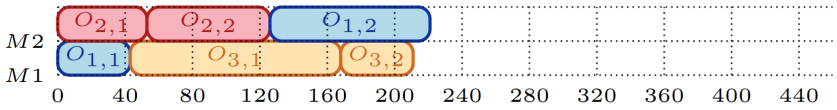

FIG. 1 – Solution admissible pour une durée nominale des opérations

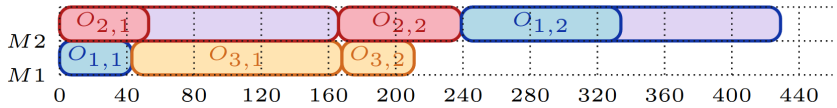

FIG. 2 – Solution dans le pire cas pour un budget d'incertitude de 2

3 Modèle robuste pour le FJSSP

Nous présentons ici deux formulations robustes du FJSSP, l'une basée sur la programmation linéaire en nombres entiers (PLNE), l'autre sur la programmation par contraintes (PPC).

3.1 Programmation linéaire en nombres entiers

On s'inspire du modèle séquentiel de [11] avec les variables suivantes :

$$x_{i,j,m} = \begin{cases} 1 & \text{si l'opération } O_{i,j} \text{ est exécutée sur la machine } m\,; \\ 0 & \text{sinon} \end{cases} \tag{3}$$

$$y_{i,j,i',j'} = \begin{cases} 1 & \text{si l'opération } O_{i,j} \text{ est exécutée avant } O_{i',j'}; \\ 0 & \text{sinon} \end{cases} \tag{4}$$

Dans notre approche multi-étape, ces variables sont celles de premier niveau. Les dates de début $t_{i,j}(\xi) \in \mathbb{R}_+$ de chaque opération $O_{i,j}$ dans le scénario ξ sont, elles, des variables de second niveau.

La formulation PLNE est la suivante (H représente une constante arbitrairement grande) :

$$\min C_{\max} \tag{5}$$

s.t.

$$\sum_{m \in \mathcal{M}_{i,j}} x_{i,j,m} = 1 \quad \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i \tag{6}$$

$$\sum_{m \in \mathcal{M}_{i,j-1}} x_{i,j-1,m} \cdot p_{i,j-1,m}(\xi) + t_{i,j-1}(\xi) \leq t_{i,j}(\xi), \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i, \forall \xi \in \mathcal{U}^\Gamma \tag{7}$$

$$t_{i',j'}(\xi) + p_{i',j',m}(\xi) - (2 - x_{i,j,m} - x_{i',j',m} + y_{i,j,i',j'}) \cdot H \leq t_{i,j}(\xi), \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}_m^2, \forall \xi \in \mathcal{U}^\Gamma \tag{8}$$

$$t_{i,j}(\xi) + p_{i,j,m}(\xi) - (3 - x_{i,j,m} - x_{i',j',m} - y_{i,j,i',j'}) \cdot H \leq t_{i',j'}(\xi), \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}_m^2, \forall \xi \in \mathcal{U}^\Gamma \tag{9}$$

$$t_{i,n_i}(\xi) + \sum_{m \in \mathcal{M}_{i,n_i}} x_{i,n_i,m} \cdot p_{i,n_i,m}(\xi) \leq C_{\max}, \forall i \in \mathcal{J}, \forall \xi \in \mathcal{U}^\Gamma \tag{10}$$

$$x_{i,j,m} \in \{0,1\} \quad \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i, \forall m \in \mathcal{M}_{i,j} \tag{11}$$

$$y_{i,j,i',j'} \in \{0,1\} \quad \forall (i,i') \in \mathcal{J}^2, \forall j \in \mathcal{O}_i, \forall j' \in \mathcal{O}_{i'} \tag{12}$$

L'objectif est de minimiser le makespan $C_{\max}$ (5). Les contraintes d'affectation (6) assurent que chaque opération est associée à exactement une machine. Les contraintes (7), (8) et (9) sont dupliquées pour chaque scénario réalisable. Les contraintes (7) représentent les relations de précédence entre deux opérations consécutives d'un même travail. Les contraintes disjonctives (8) et (9) évitent le chevauchement d'opérations qui s'exécutent sur la même machine. Enfin, les contraintes (10) permettent de calculer le plus grand makespan parmi tous les scénarios réalisables et donc de déterminer la solution optimale dans le pire des cas.

3.2 Programmation par contraintes

La PPC a prouvé son efficacité dans la résolution de nombreux problèmes d'ordonnancement. En revanche, à notre connaissance, elle n'a jamais été utilisée dans les approches robustes multi-étapes. En nous inspirant du modèle présenté dans [9] pour le FJSSP déterministe avec contraintes opérateurs, nous proposons ici un modèle PPC pour le FJSSP robuste. Au préalable, nous introduisons les variables suivantes :

— $task_{i,j,\xi}$: variable d'intervalle entre le début et la fin de l'opération $O_{i,j}$ dans le scénario ξ ;
— $mode_{i,j,m,\xi}$: variable d'intervalle entre le début et la fin de l'opération $O_{i,j}$ sur la machine m dans le scénario ξ (les opérations ayant plusieurs machines éligibles, il s'agit d'une variable optionnelle) ;
— $seqs_{m,\xi}$: variable de séquence des tâches s'effectuant sur la machine m dans le scénario ξ.

Le modèle PPC est le suivant :

$$\min C_{\max} \tag{13}$$

s.t.

$$C_{\max} \geq task_{i,n_i,\xi}.end, \forall i \in \mathcal{J}, \forall \xi \in \mathcal{U}^\Gamma \tag{14}$$

$$EndBeforeStart(task_{i,j,\xi}, task_{i,j+1,\xi}), \forall i \in \mathcal{J}, \forall O_{i,j} \in \mathcal{O}_i \setminus \{O_{i,n_i}\}, \forall \xi \in \mathcal{U}^\Gamma \tag{15}$$

$$Alternative(task_{i,j,\xi}, mode_{i,j,m,\xi} : \forall m \in \mathcal{M}_{i,j}), \forall i \in \mathcal{J}, \forall O_{i,j} \in \mathcal{O}_i, \forall \xi \in \mathcal{U}^\Gamma \tag{16}$$

$$NoOverlap(seqs_{m,\xi}), \forall m \in \mathcal{M}, \forall \xi \in \mathcal{U}^\Gamma \tag{17}$$

$$PresenceOf(mode_{i,j,m,0}) \Rightarrow PresenceOf(mode_{i,j,m,\xi}), \forall i \in \mathcal{J}, \forall O_{i,j} \in \mathcal{O}_i, \forall m \in \mathcal{M}_{i,j}, \forall \xi \in \mathcal{U}^\Gamma \setminus \{0\} \tag{18}$$

$$SameSequence(seqs_{m,0}, seqs_{m,\xi}), \forall m \in \mathcal{M}, \forall \xi \in \mathcal{U}^\Gamma \setminus \{0\} \tag{19}$$

Les contraintes (14) permettent de calculer le makespan, égal à la fin de la dernière opération. Les contraintes (15) représentent les relations de précédence entre deux opérations consécutives d'un même travail. Les contraintes (16) assurent l'affectation d'une opération à une seule machine. Les contraintes (17) garantissent que, dans chaque scénario, chaque machine effectue une seule opération à la fois. Les contraintes (14–17) sont dupliquées pour chaque scénario. Les contraintes (18) et (19) assurent que l'affectation de chaque opération et la séquence sur chaque machine sont les mêmes dans chaque scénario. Pour ces deux dernières contraintes, le premier scénario $\xi = 0$ est utilisé comme référence.

4 Génération de colonnes et de contraintes

4.1 Principe

Nous présentons ici une approche de génération de colonnes et de contraintes pour résoudre le FJSSP robuste. Ce type d'approche, introduit par [13] pour résoudre des problèmes d'optimisation robuste en deux étapes, décompose le problème initial en un problème maître et un sous-problème (*adversarial subproblem*). Sur la base d'un sous-ensemble limité de scénarios,

le problème maître tente de résoudre le problème d'ordonnancement robuste ; s'il n'y parvient pas, le sous-problème identifie les scénarios qui l'en empêchent. Ces scénarios sont inclus dans le problème maître en générant à la volée des variables de recours. Le processus est répété jusqu'à ce qu'une solution admissible soit trouvée pour tous les scénarios [5, 8, 10, 12].

4.2 Algorithme hybride

Pour le FJSSP, le problème maître peut être défini comme l'un des modèles étendus présentés dans le paragraphe 3, pour lequel seul un sous-ensemble de scénarios est considéré (cf. contraintes (7–10) pour le modèle PLNE ou (14–17) pour le modèle PPC). Une solution optimale pour ce problème maître détermine une borne inférieure sur le makespan du problème global. Afin de vérifier la faisabilité de cette solution, étant donné les affectations et les séquences sur les machines fixées dans le problème maître, on vérifie si l'un des scénarios possibles conduit à un makespan plus grand que celui trouvé dans le problème maître. Si tel est le cas, les variables et les contraintes qui lui sont associées sont générées et ajoutées au problème maître. Dans notre cas, la recherche d'un tel scénario se ramène à la recherche du scénario qui conduit au plus grand makespan possible.

4.3 Evaluation pire cas

Les contraintes d'affectation et de séquence sur les machines étant réglées, étant donné un budget d'incertitude Γ, l'objectif d'évaluation du pire cas est d'identifier les Γ opérations qui dévient et qui conduisent au plus grand makespan.

4.3.1 PLNE

Soient les variables de décision suivantes :

$$\xi_{i,j} = \begin{cases} 1 & \text{si la durée de l'opération } O_{i,j} \text{ dévie ;} \\ 0 & \text{sinon} \end{cases} \tag{20}$$

$$b_{i,j,i',j'} = \begin{cases} 1 & \text{si l'opération } O_{i,j} \text{ commence à la fin de l'opération } O_{i',j'} \text{ ;} \\ 0 & \text{sinon} \end{cases} \tag{21}$$

$$d_i = \begin{cases} 1 & \text{si la dernière opération } O_{i,n_i} \text{ du job } i \text{ est la dernière opération à être effectuée ;} \\ 0 & \text{sinon} \end{cases} \tag{22}$$

Le modèle PLNE développé pour l'évaluation du scénario le plus défavorable est le suivant ($A_{i,j}$ représente l'ensemble des prédécesseurs immédiats de l'opération $O_{i,j}$) :

$$\max \ C_{\max} \tag{23}$$

s.t.

$$\sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{O}_i} \xi_{i,j} = \Gamma \tag{24}$$

$$t_{i,j} - (t_{i',j'} + \bar{p}_{i',j'} + \xi_{i,'j'} \cdot \hat{p}_{i',j'}) \leq H \cdot (1 - b_{i,j,i',j'}), \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i, \forall (i',j') \in A_{i,j} \tag{25}$$

$$\sum_{(i',j') \in A_{i,j}} b_{i,j,i',j'} \geq 1, \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i \mid A_{i,j} \neq \emptyset \tag{26}$$

$$t_{i,1} = 0, \forall i \in \mathcal{J} \mid A_{i,0} = \emptyset \tag{27}$$

$$C_{\max} - (t_{i,n_i} + \bar{p}_{i,n_i} + \xi_{i,n_i} \cdot \hat{p}_{i,n_i}) \leq H \cdot (1 - d_i), \forall i \in \mathcal{J} \tag{28}$$

$$\xi_{i,j} \in \{0,1\}, \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i \tag{29}$$

$$b_{i,j,i',j'} \in \{0,1\}, \forall i \in \mathcal{J}, \forall j \in \mathcal{O}_i, \forall (i',j') \in A_{i,j} \tag{30}$$

$$d_i \in \{0,1\}, \forall i \in \mathcal{J} \tag{31}$$

L'objectif est de trouver le scénario (les valeurs de $\xi_{i,j}$ pour chaque opération $O_{i,j}$) qui maximise le makespan (23). Les contraintes (24) garantissent qu'exactement Γ opérations ont leur durée qui dévie. Les contraintes (25) et (26) contraignent les opérations à commencer à la plus grande date de fin de leurs prédécesseurs ; à 0 pour les opérations sans prédécesseur (contraintes (27). Les contraintes (28) définissent le makespan (plus grande date de fin des jobs).

4.3.2 PPC

Nous introduisons un modèle PPC avec des variables d'intervalle optionnelles $dev_{i,j}$, présentes uniquement si la durée de l'opération correspondante $O_{i,j}$ s'écarte de sa valeur nominale (et prend donc sa valeur maximale). Soient les variables de décision suivantes :

— $task_{i,j}$, variable d'intervalle associée à l'opération $O_{i,j}$ de durée $\bar{p}_{i,j,m}$;
— $dev_{i,j}$, variable d'intervalle optionnelle de durée $\hat{p}_{i,j,m}$, présente si l'opération $O_{i,j}$ s'écarte de sa valeur nominale.

Nous utilisons également la fonction $HeightAtStart$, qui donne la contribution d'une variable d'intervalle à une fonction cumulative à sa date de début, et la fonction cumulative $StepAtStart$, qui définit la contribution d'un intervalle à son début.

$$\max C_{\max} \tag{32}$$

s.t.

$$task_{i,1}.start = 0, \forall i \in \mathcal{J} \mid A_{i,1} = \emptyset \tag{33}$$

$$task_{i,j}.start = \max(\{task_{i',j'}.end \mid O_{i',j'} \in A_{i,j}\}), \forall i \in \mathcal{J}, \forall O_{i,j} \in \mathcal{O}_i \tag{34}$$

$$StartAtEnd(dev_{i,j}, task_{i,j}), \forall i \in \mathcal{J}, \forall O_{i,j} \in \mathcal{O}_i \tag{35}$$

$$\sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{O}_i} HeightAtStart(dev_{i,j}, StepAtStart(dev_{i,j}, 1)) = \Gamma \tag{36}$$

$$C_{\max} = \max(\cup_{\forall i \in \mathcal{J}}(\{task_{i,n_i}.end\} \cup \{dev_{i,n_i}.end\})) \tag{37}$$

L'objectif est de trouver le scénario qui maximise le makespan (32). Les contraintes (34) et (35) garantissent que chaque opération commence au plus tôt, compte tenu de l'affectation et de la séquence déterminées par le problème maître, et des contraintes de précédence. Les contraintes (36) garantissent que, si la variable d'intervalle $dev_{i,j}$ est présente, la durée de l'opération correspondante $O_{i,j}$ prend sa valeur maximale. Les contraintes (36) garantissent que exactement Γ opérations prennent la valeur maximale pour leur durée. Enfin, les contraintes (37) expriment que le makespan la plus grande date de fin des jobs.

5 Résultats expérimentaux

L'implémentation en C++ utilise CPLEX 12.10 pour les modèles PLNE et CPO 12.10 pour les modèles PPC. Le temps CPU et la RAM sont respectivement limités à 1 heure et 16 Go. Nous utilisons trois nœuds de cluster possédant des CPU Intel Xeon E5-2695 v4 à 2,1 GHz.

5.1 Instances et méthodes

Nous nous servons des benchmarks de FJSSP de [6] constitués de 20 instances notées SFJSi et MFJSi ($i = 1..10$), de taille croissante, jusqu'à 12 jobs, 8 machines et 48 opérations. Nous les adaptons au contexte robuste en générant aléatoirement des valeurs de déviation des durées opératoires. Soit $\bar{p}_{max}$ la durée nominale maximale pour toutes les opérations. Pour chaque opération $O_{i,j}$ et chaque machine éligible $m \in \mathcal{M}_{i,j}$, nous générons aléatoirement une valeur pour $\hat{p}_{i,j,m}$ dans un intervalle de 25 % à 100 % de la valeur de $\bar{p}_{max}$. Le processus est répété 3 fois ; nous obtenons donc trois nouvelles instances à partir d'une instance déterministe initiale.

Nous générons au total 60 instances. Nous testons tous les modèles en faisant varier le budget d'incertitude selon quatre ratios : 20 %, 40 %, 60 % et 80 %, soit un total de 240 expériences par méthode.

En plus des deux modèles étendus présentés au paragraphe 3, que nous désignerons par `plne` et `ppc`, nous évaluons quatre combinaisons de l'algorithme de génération de colonnes et de contraintes présentés au paragraphe 4, combinaisons que nous noterons par `gcc_mod1_mod2`, `mod1` et `mod2` pouvant prendre les valeurs `plne` ou `ppc`.

5.2 Résultats

Le tableau 2 présente les performances relatives des méthodes sur les 240 instances testées ; "#Solv." indique le nombre d'instances résolues optimalement. Concernant les méthodes étendues, la formulation PPC est légèrement plus performante que la méthode PLNE. On constate que les méthodes basées sur la génération de colonnes et de contraintes permettent de résoudre considérablement plus d'instances que les méthodes étendues. Plus précisément, celles qui utilisent une formulation PPC pour le problème maître sont celles qui obtiennent les meilleurs résultats. Enfin, nous observons que la résolution du sous-problème par PLNE est marginalement meilleure qu'avec le modèle PPC.

Méthode	#Solv.
`plne`	130
`ppc`	133
`gcc_plne_plne`	167
`gcc_plne_ppc`	166
`gcc_ppc_ppc`	190
`gcc_ppc_plne`	193

TAB. 2 – Performances des méthodes pour les 240 instances générées

Le tableau 3 recense, pour chaque instance originale, le nombre d'instances robustes (parmi les 12) résolues optimalement par chaque méthode et le temps moyen pour y parvenir. Sans surprise, nous observons que plus la taille des instances augmente, plus elles sont difficiles à résoudre, ce qui se traduit par un nombre décroissant d'instances résolues et un temps CPU croissant. Les méthodes étendues parviennent à résoudre toutes les instances jusqu'à SFSJ10 (4 jobs et 5 machines), tandis que `gcc_ppc_plne` résout toutes les instances jusqu'à MFJS3 (6 jobs et 7 machines).

Le tableau 4 montre la performance des méthodes en fonction du budget d'incertitude. Nous remarquons que pour les méthodes étendues, la résolution est plus difficile pour les budgets moyens (40 % et 60 %) ; cela est dû au fait que ces budgets d'incertitude génèrent un plus grand nombre de scénarios et, donc un plus grand nombre de variables de décision. En revanche, pour les méthodes basées sur l'algorithme de génération de colonnes et de contraintes, le nombre d'instances résolues optimalement augmente avec le budget d'incertitude. Cela montre que le nombre de scénarios n'a pas d'impact direct sur la difficulté du problème.

6 Conclusions

Dans cet article, nous proposons une approche robuste multi-étapes pour la résolution du problème d'ordonnancement du type job-shop type flexible où les durées des opérations sont sujets à incertitude. La première étape est consacrée à fixer des décisions sur l'affectation des opérations aux machines et leur séquencement sur ces machines ; la deuxième étape détermine les dates de début des opérations.

Nous introduisons deux types de formulations robustes pour résoudre le problème. Nous utilisons un algorithme de génération de colonnes et de contraintes hybridant programmation linéaire en nombres entiers et programmation par contraintes. Les résultats expérimentaux

Instance	plne		ppc		gcc_plne_plne		gcc_plne_ppc		gcc_ppc_ppc		gcc_ppc_plne	
	#Solv.	t(s)	#Solv.	t(s)	#Solv.	t(s)	#Solv.	t(s)	#Solv.	t(s)	#Solv.	t(s)
SFJS1	12	0.04	12	0.03	12	0.06	12	0.98	12	1.13	12	0.05
SFJS2	12	0.03	12	0.03	12	0.03	12	0.25	12	0.38	12	0.04
SFJS3	12	0.4	12	0.08	12	0.5	12	3.03	12	3.17	12	0.11
SFJS4	12	0.25	12	0.05	12	0.23	12	1.65	12	1.77	12	0.08
SFJS5	12	0.83	12	0.12	12	1.03	12	2.93	12	3.8	12	0.2
SFJS6	12	4.55	12	0.44	12	1.49	12	5.62	12	4.89	12	0.25
SFJS7	12	1.3	12	0.25	12	0.31	12	1.46	12	1.73	12	0.11
SFJS8	12	7.32	12	0.62	12	1.79	12	5.68	12	6.22	12	0.31
SFJS9	12	11.5	12	2.05	12	1.42	12	2.47	12	3.19	12	0.16
SFJS10	**12**	128	**12**	3.1	12	2.01	12	8.59	12	11.5	12	0.2
MFJS1	6	1115	6	1445	12	257	12	297	12	73.1	12	19.9
MFJS2	4	1246	6	858	12	161	12	199	12	44.6	12	9.55
MFJS3	0	-	1	1301	11	549	10	414	11	174	**12**	99.5
MFJS4	0	-	0	-	5	849	5	1715	11	539	11	450
MFJS5	0	-	0	-	5	694	6	1089	10	544	10	328
MFJS6	0	-	0	-	2	1643	1	2522	7	255	9	739
MFJS7	0	-	0	-	0	-	0	-	4	1006	4	815
MFJS8	0	-	0	-	0	-	0	-	3	1144	3	880
MFJS9	0	-	0	-	0	-	0	-	0	-	0	-
MFJS10	0	-	0	-	0	-	0	-	0	-	0	-

TAB. 3 – Comparaison des méthodes pour chaque groupe de 12 instances dérivées d'une instance initiale

Budget d'incertitude	plne	ppc	gcc_plne_plne	gcc_plne_ppc	gcc_ppc_ppc	gcc_ppc_plne
20 %	35	37	38	38	42	42
40 %	30	31	39	39	45	48
60 %	30	30	42	43	49	49
80 %	35	35	48	46	54	54

TAB. 4 – Comparaison des méthodes pour chaque budget d'incertitude, sur 60 instances

montrent que la meilleure approche est la méthode de génération de colonnes et de contraintes utilisant la programmation par contraintes dans le problème maître et un modèle de programmation linéaire en nombres entiers pour le sous-problème.

Références

[1] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2) :351–376, 2004.

[2] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1) :35–53, 2004.

[3] Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3) :157–183, 1993.

[4] Stéphane Dauzère-Pérès and Jan Paulli. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70 :281–306, 1997.

[5] José Luis Ruiz Duarte, Neng Fan, and Tongdan Jin. Multi-process production scheduling with variable renewable integration and demand response. *European Journal of Operational Research*, 281(1) :186–200, 2020.

[6] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3) :331–342, 2007.

[7] Michael R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2) :117–129, 1976.

[8] Idir Hamaz, Laurent Houssin, and Sonia Cafieri. The cyclic job shop problem with uncertain processing times. In *16th International Conference on Project Management and Scheduling (PMS 2018)*, pages 119–122, Rome, Italy, April 2018.

[9] Dominik Kress and David Müller. Mathematical models for a flexible job shop scheduling problem with machine operator constraints. *IFAC-PapersOnLine*, 52(13) :94–99, 2019.

[10] Mario Levorato, Rosa Figueiredo, and Yuri Frota. Exact solutions for the two-machine robust flow shop with budgeted uncertainty. *European Journal of Operational Research*, 300(1) :46–57, 2022.

[11] Liji Shen, Stéphane Dauzère-Pérès, and Janis S. Neufeld. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265(2) :503–516, 2018.

[12] Marco Silva, Michael Poss, and Nelson Maculan. Solution algorithms for minimizing the total tardiness with budgeted processing time uncertainty. *European Journal of Operational Research*, 283(1) :70–82, 2020.

[13] Bo Zeng and Long Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5) :457–461, 2013.