Etude des critères d'optimisation pour la replanification des chaînes d'assemblage aéronautiques

Damien Lovato, Romain Guillaume, Caroline Thierry, Olga Battaïa

ETUDE DES CRITERES D'OPTIMISATION POUR LA REPLANIFICATION DE CHAINES D'ASSEMBLAGE AERONAUTIQUES

D. LOVATO, R. GUILLAUME, C. THIERRY,

IRIT – Université de Toulouse - 31400
TOULOUSE France
damien.lovato@univ-tlse2.fr
romain.guillaume@irit.fr
caroline.thierry@univ-tlse2.fr

O. BATTAIA

KEDGE Business School
BORDEAUX France
olga.battaia@kedgebs.com

RESUME : *Dans cet article, nous considérons le problème de replanification des processus d'assemblage aéronautique en réponse à une interruption causée par un aléa. Le problème d'ordonnancement étudié doit tenir compte des contraintes de précédence et de ressources (par ces aspects il est proche de celui de la formulation classique du Resource Constrained Project Scheduling Problem) mais aussi d'autres contraintes comme la considération des métiers des opérateurs, leurs compétences et la gestion des zones de travail. Notre objectif est d'étudier différents critères d'optimisation qui peuvent être utilisés dans ce contexte afin de fournir une aide aux manageurs en charge de la replanification. Nous présentons un modèle de Programmation Par Contraintes (PPC) pour le problème considéré en implémentant plusieurs fonctions multi-objectifs proposées. Les performances de ces fonctions sont testées sur un ensemble d'instances issues du terrain. Nous discutions ainsi l'impact de chaque critère sur les caractéristiques des solutions trouvées lors de la résolution.*

MOTS-CLES : *RCPSP, gestion d'aléas, réactif, PPC, multi-objectifs.*

1 CONTEXTE DU PROBLEME ETUDIE

De nos jours, l'ordonnancement des systèmes de production est de plus en plus complexe et présente des nombreuses sources d'incertitude. Ces incertitudes sont liées à des aléas divers et variés perturbant les plannings de production établis au préalable. Ces aléas peuvent engendrer des coûts de production supplémentaires voire un arrêt de production et ont ainsi un impact non négligeable sur les performances des systèmes de production. Afin d'anticiper leurs conséquences négatives, l'étude de leur gestion a fait naitre de nombreuses approches du génie industriel et de la recherche opérationnelle.

Le secteur de l'aéronautique est fortement concerné par la gestion des aléas. Assembler un avion est un processus long et couteux. De plus, chaque modèle d'avion possède un processus d'assemblage particulier et les exigences en termes de qualité sont très encadrées par les critères de certification. Ces contraintes doivent être prises en compte lors de la planification des tâches, ce qui consiste à indiquer à quel moment chaque tâche doit être commencée et par quel opérateur. Etant donné que le nombre de tâches à planifier peut être très significatif, entre centaines et milliers de tâches par poste de travail, des algorithmes de planification automatiques sont nécessaires afin d'aider les manageurs dans leur prise de décision. Afin d'obtenir des solutions satisfaisantes par l'application de ces algorithmes, leur conception doit tenir compte de nombreuses contraintes techniques et managériales du terrain.

L'affectation des tâches aux opérateurs doit respecter les compétences acquises par chaque opérateur et son appartenance au même corps de métier que celui requis par la tâche. De plus, les opérateurs travaillant en équipe ont des affinités sociales qui doivent être prises en compte lors de la répartition des tâches afin d'améliorer le bien-être au travail du personnel et indirectement la qualité de travail de celui-ci.

Une façon possible de procéder est d'affecter les membres d'une équipe à la même famille de tâches. Dans le processus d'assemblage, les tâches sont regroupées sous forme de familles qui représentent l'assemblage d'un élément fonctionnel complexe. Il sera préférable d'affecter un opérateur à un nombre limité de familles de tâches, afin de diminuer sa charge cognitive liée au changement du contexte particulier pour chaque famille de tâches. En même temps, il sera aussi préférable dans la mesure du possible que le même opérateur commençant l'assemblage d'un élément complexe finalise le traitement des tâches de la même famille et que les membres de la même équipe travaillent sur les tâches de la même famille.

Pour réaliser une tâche, il est parfois nécessaire de recourir à des équipements particuliers disponibles. La disponibilité de tels outils doit donc constamment être vérifiée. Il en est de même pour la disponibilité de chaque opérateur qui a ses propres horaires pré définies.

Enfin, il est nécessaire de prendre en compte l'existence de zones de travail. Ces zones peuvent être situées au sol tout comme dans certaines parties de l'avion. Chaque tâche s'effectue sur une ou plusieurs zones, et le nombre d'opérateurs présents simultanément dans chaque zone de travail est limité par l'espace disponible pour que les opérateurs puissent effectuer leur travail sans se gêner mutuellement.

Le problème de planification initial présente déjà une complexité accrue à cause de ces nombreuses contraintes et spécificités. Lorsque celui-ci est remis en cause par un aléa (par exemple l'impossibilité de réaliser une tâche à cause de l'absence de l'opérateur, la pièce à assembler ou un problème de qualité constaté), il est urgent de replanifier l'exécution des tâches restantes en tenant compte de cet empêchement afin de ne pas trop retarder la finalisation de l'assemblage tout en cherchant une solution pour l'aléa présent (afin de rendre possible la réalisation de la tâche entravée). A la complexité du problème s'ajoute donc la contrainte temporelle de la prise de décision pour répondre aux aléas survenus. Dans ce papier, nous développons des méthodes de planification qui peuvent être utilisées dans ce contexte.

Le reste du papier est organisé ainsi : la section 2 est dédiée à l'étude de la littérature et notre positionnement dans l'état de l'art. Dans la section 3, nous présentons notre modèle de Programmation Par Contraintes (PPC). Les résultats des expérimentations sur des instances inspirées du terrain sont analysées dans la section 4. Nous terminons sur comparaison des différents critères qui peuvent être utilisés pour résoudre le problème étudié en analysant leur efficacité.

2 ETUDE DE LA LITTERATURE

La présence dans le problème d'ordonnancement étudié de ressources renouvelables en quantité limitée et la notion de précédence entre les tâches fait référence à un problème type du RCPSP (Resource Constrained Project Scheduling Problem). Cependant, les nombreuses autres contraintes indiquent que la formulation classique du RCPSP n'est pas suffisante. De plus, comme expliqué précédemment, les incertitudes sont nombreuses dans le contexte de l'assemblage aéronautique, et les aléas peuvent subvenir à tout moment et ralentir voire arrêter la réalisation des tâches pour une durée indéterminée. L'approche à proposer aux managers doit en tenir compte.

Pour analyser l'impact des incertitudes, Hazir et Ulusoy (2019) les ont classifiées en fonction de leurs sources, qui peuvent être aussi bien externes qu'internes. Leur étude montre que les incertitudes peuvent être d'origine organisationnelle (au niveau du management), liées aux ressources utilisées, à la nature du travail ou encore à la logistique en amont. Les conséquences possibles de ces incertitudes entrainent l'impossibilité de démarrer une ou plusieurs tâches comme prévu. Une revue des approches permettant de gérer ces incertitudes a été également présentée dans cette étude mais aussi dans les travaux de Chaari et al. (2014) où des méthodes proactives, réactives et hybrides ont été présentées. Nous pouvons trouver par exemple le « right shift scheduling » qui consiste à décaler l'ensemble des tâches affectées par l'aléa plus tard dans l'ordonnancement. Cela permet de conserver la structure de la solution en conservant l'ordre des affectations mais présente peu d'opportunités d'amélioration de solution.

Dans le contexte que nous étudions, les manageurs ont besoin d'une approche réactive. Celle-ci intervient en réaction à un aléa perturbant le bon déroulement du planning de l'assemblage. Son objectif est par conséquent de déterminer une solution faisable en un court laps de temps afin que le processus d'assemblage puisse reprendre rapidement.

Une approche réactive peut englober plusieurs méthodes. L'une d'entre elles, présentée par Zhu et al. (2005), utilise un modèle de type PLNE (Programme Linéaire en Nombres Entiers) qui prend en compte plusieurs solutions prédéterminées pour s'adapter à certains types d'aléas. Kuster et al. (2008) ont introduit le concept des tâches alternatives avec des états d'activation. Cela leur permet de gérer ainsi une plus grande variété d'aléas avec un seul modèle qui leur offre différents paternes de formulation (alternation de mode d'exécution, de ressource ou de capacité, insertion/suppression d'une tâche, flexibilité de l'ordre d'exécution et sérialisation ou parallélisation des tâches).

Dans ce papier, nous proposons un ensemble de critères pouvant être appliqués dans ce cas et nous étudions leur efficacité. L'analyse des objectifs considérés lors de la planification initiale a été réalisée par Hartmann et Briskorn (2010) mais dans notre cas il s'agit des critères d'optimisation bien spécifiques car nous devons tenir compte de la planification initiale. Ainsi les critères considérés concerneront le makespan final, mais aussi la proximité du planning obtenu après la replanification avec le planning initial. Plus les deux plannings sont proches, moins il y aura perturbations dans l'organisation du système de production et de son systèmes logistique.

Comme précisé dans la section 1, le contexte considéré présente une forte composante humaine et sociale, avec la nécessité de tenir compte des savoir-faire des opérateurs et de leurs affinités sociales. Les travaux de Bhadury et al. (2000) ont mis en avant les bénéfices que peut apporter le regroupement en familles (ou équipes) des opérateurs. L'intégration des compétences des opérateurs au problème de planification a été notamment étudiée par Valls et al. (2009) qui ont développé un algorithme génétique pour résoudre un problème de type MSPSP (Multi-Skilled Project Scheduling Problem).

La prise en compte des zones de travail peut être retrouvée dans les travaux de Gharbi et al. (1999) qui gèrent cet aspect dans le cadre d'un projet de réparation. Dans leur cas, il s'agit d'un projet lié à des opérations réalisées par

des machines, sans intervention manuelle, alors que dans notre cas, nous dévons gérer les personnes effectuant des tâches sur des zones de travail qui représentent une ressource disjonctive.

L'association de tous ces aspects décrits ci-dessus résulte en un problème d'ordonnancement bien spécifique nécessitant une modélisation adaptée. En tenant compte de la contrainte temporelle pour la recherche d'une bonne solution, nous avons choisi le formalisme de la programmation par contraintes pour modéliser ce nouveau problème d'optimisation et le résoudre. Ce modèle est présenté dans la section suivante.

3 MODELE PPC

La PPC est une méthodologie très efficace pour la résolution des problèmes d'ordonnancement même pour des instances de taille importante. Avant de présenter le modèle construit, nous expliquons la notion des « tuples ». Ces objets peuvent être apparentés à des *n*-uplet, avec n un entier représentant le nombre d'éléments de ce *n*-uplet. Ainsi, utilisés pour définir certains éléments (notamment les tâches et les opérateurs), ces *n*-uplet permettent une meilleure gestion des informations. Afin d'accéder aux éléments de ces *n*-uplet, il est possible d'utiliser le point de ponctuation suivi du nom de l'élément à récupérer comme sur l'exemple ci-dessous :

$$uplet.élément$$

3.1 Variables

Les variables de décision de notre modèle seront pour la plupart des intervalles. Les intervalles définissent une période durant laquelle un évènement se déroule. Il est ainsi caractérisé par une date de départ, de fin, une intensité et une durée. Ses dates de début et de fin sont inconnues et seront déterminées durant la résolution du modèle. Sa durée est déterminée à l'avance (dans notre cas, la durée de chaque intervalle correspondra à la durée de la tâche associée). L'intensité est par défaut de 100%, ce qui sera considéré dans notre modélisation.

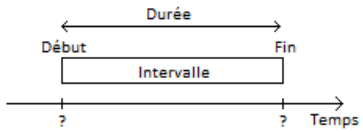

Figure 1 – Variable de type intervalle

Enfin, un intervalle peut être optionnel. Cela veut dire que celui-ci ne sera pas obligatoirement pris en compte dans le calcul de la solution (ni dans les contraintes). Ce sera ainsi au programme de déterminer quels intervalles optionnels seront actifs, et ceux qui seront inactifs. Voici les variables qui seront utilisées :

- $intvl_i$: intervalle sur lequel i est actif (i ∈ I).
- $x_{i,k}$: intervalle optionnel sur lequel l'opérateur k effectue la tâche i. Sa taille est égale à la durée de i et est restreinte par le calendrier de l'opérateur k (i ∈ I, k ∈ H).

Cet intervalle est donc optionnel pour dire qu'il y a plusieurs possibilités pour chaque couple $< tâche; opérateur >$ et que chaque intervalle défini comme « actif » représentera l'affectation réalisée dans la solution.
En plus de ces intervalles, nous utiliserons également une fonction d'état :

- is_usable_a : fonction d'état qui nous permet de contrôler l'utilisation de la zone a (a ∈ A).

Celle-ci prendra la valeur 0 sur les intervalles où la zone a n'est pas utilisée et 1 sur ceux où elle l'est.

3.2 Fonction objectif

La fonction objectif est composée de plusieurs expressions pour les différents critères que nous allons chercher à optimiser :

$$\max_{i \in I, k \in H} endof(x_{i,k}) \quad (1)$$

$$\sum_{i \in I} |endof(intvl_i) - old\_end_i| \quad (2)$$

$$\sum_{i \in I, k \in H} presenceof(x_{i,k}) * old\_x_{i,k} \quad (3)$$

Le critère (1) représente le makespan de la solution. Le critère (2) mesure les décalages des dates de fin de chaque tâche. Le critère (3) mesure le nombre de couples <tâche ; opérateur> qui seront conservés d'une solution à l'autre.

Grâce à ces expressions, on peut donc définir un critère multi-objectif pour évaluer la solution :

$$\min \text{staticLex}((1),(2),(3)) \quad (4)$$

La fonction « staticLex » est utilisée pour de l'optimisation multi objectifs. Grâce à celle-ci il est possible de donner la priorité à certains critères. Dans le cas considéré par la fonction (4), l'ordre d'optimisation est : makespan (1), décalages dates de départ des tâches (2), changements d'affectations (3). Si nous désirons changer les priorités, il suffit d'interchanger les positions (pour mettre (3) à la place de (1) par exemple, faisant ainsi passer le critère de makespan en dernier en termes de priorité). En termes d'objectif, cela signifie que notre fonction va améliorer en premier le critère (1) jusqu'à atteindre l'optimum. Ensuite elle va passer au critère (2) pour finir par le (3). Cela signifie qu'on ne pourra pas avoir la valeur optimale pour le critère (3) si (2) n'est lui aussi pas à l'optimum. Interchanger l'ordre des critères permet donc d'explorer de nouvelles solutions et d'attaquer le problème sous un angle différent.

3.3 Contraintes

Voici les ensembles utilisés dans la définition des contraints :

I	ensemble des tâches ;
E	ensemble des couples de précédence ;
H	ensemble des ressources humaines ;
M	ensemble des ressources matérielles ;

P ensemble des métiers ;
A ensemble des zones ;
D ensemble des tâches déjà démarrées/terminées ;
F ensemble des tâches à réaffecter.

Respect des contraintes de précédence :

$$\forall (i, j) \in E \quad \text{endBeforeStart}(\text{intvl}_i, \text{intvl}_j) \tag{5}$$

Respect de la limite des ressources matérielles :

$$\forall k \in M \quad \sum_{i \in I} \text{pulse}(\text{intvl}_i, i.\text{m\_needed}_k) \leq \text{capacity}_k \tag{6}$$

La fonction $\text{pulse}(\text{interval}, \text{quantity})$ est utilisée pour caractériser l'utilisation d'une ressource (ici k) et représente la contribution d'un intervalle à cette ressource sur le laps de temps où il est défini actif.

Respect de la limite du nombre d'opérateurs pour chaque métier :

$$\forall p \in P, \forall k \in H \text{ si } p = k.\text{métier} \quad \sum_{i \in I} \text{pulse}(x_{i,k}, i.\text{w\_needed}_p) \leq \text{workers}_p \tag{7}$$

Double contrainte pour indiquer que chaque opérateur ne doit réaliser qu'une tâche à la fois :

$$\forall p \in P, \forall k \in H \text{ si } p = k.\text{métier} \quad noOverlap(x_{i,k} : \forall i \in I \text{ si } i.\text{w\_needed}_p > 0) \tag{8}$$

La contrainte $noOverlap(\{b_1, \dots, b_n\})$ stipule qu'un seul intervalle de l'ensemble $\{b_1, \dots, b_n\}$ peut être actif au même moment.

$$\forall i \in I, \forall p \in P \text{ si } i.\text{w\_needed}_p > 0 \quad alternative(intvl_i, x_{i,k} : \forall k \in H \text{ si } k.\text{prof} = p, i.\text{w\_needed}_p) \tag{9}$$

La fonction $alternative(intvl, \{b_1, \dots, b_n\}, n)$ indique que si l'intervalle $intvl$ est actif alors il doit y avoir exactement n intervalles de l'ensemble $\{b_1, \dots, b_n\}$ actifs, et que ceux-ci doivent être synchronisés avec $intvl$.

Aucun opérateur ne doit être affecté à une tâche dont il ne maitrise pas le métier requis :

$$\forall i \in I, \forall k \in H \text{ si } i.\text{w\_needed}_{k.\text{métier}} = 0 \quad !\, presenceOf(x_{i,k}) \tag{10}$$

La fonction $presenceOf(intvl)$ force l'intervalle « $intvl$ » à être présent dans la solution. Cette contrainte force donc tous les intervalles $x_{i,k}$ concernés à ne pas être actifs.

Duo de contraintes pour n'autoriser l'affectation de tâches qu'à des dates où l'opérateur est disponible :

$$\forall i \in I, \forall k \in H \quad forbidStart(x_{i,k}, Calendar_k) \quad forbidEnd(x_{i,k}, Calendar_k) \tag{11}$$

$Calendar_k$ est une fonction qui représente l'activité de l'opérateur k au cours du temps. Elle prend la valeur 100 lorsque l'opérateur est à 100% d'efficacité (et qu'il travaille) et 0 lorsqu'il est en pause et ne travaille pas. La fonction $forbidStart$ empêche ainsi tout intervalle $x_{i,k}$ de démarrer lorsque $Calendar_k$ prend la valeur 0. De la même manière, $forbidEnd$ empêche les intervalles de finir dans ces laps de temps. Si l'intervalle $x_{i,k}$ devait se finir sur un laps de temps où $Calendar_k$ vaut 0, l'intervalle sera prolongé d'autant d'unités de temps d'absence de l'opérateur en charge.

Le prochain duo de contraintes assure qu'aucune tâche ne peut s'effectuer dans une zone exclue :

$$\forall i \in I, \forall a \in i.\text{uses} \quad alwaysEqual(is\_usable_a, intvl_i, 1) \tag{12}$$

La contrainte $alwaysEqual(statef, intvl, n)$ stipule que la fonction d'état $statef$ doit toujours être égale à n sur l'intervalle $intvl$. Dans notre cas, cela signifie que la fonction is_usable_a doit valoir 1 sur l'intervalle $intvl_i$, et ceci pour chaque zone présente dans $i.uses$ qui représente l'ensemble des zones utilisées par i.

$$\forall i \in I, \forall b \in i.\text{excludes} \quad alwaysNoState(is\_usable_b, intvl_i) \tag{13}$$

Le fonctionnement de $alwaysNoState(statef, intvl)$ est similaire à la contrainte (11) à l'exception que cette fonction force $statef$ à 0 sur l'intervalle $intvl$. Cela permet ainsi de forcer chaque fonction d'état is_usable_b liée à la zone b soit égale à 0 lorsqu'une zone est exclue par i.

Chaque zone doit se voir affecter au plus une seule tâche à chaque instant :

$$\forall a \in A \quad \sum_{i \in I \text{ si } a \in i.\text{uses}} \text{pulse}(\text{intvl}_i, 1) \leq 1 \tag{14}$$

Enfin, les deux dernières contraintes ont été mises en place pour la replanification seulement, alors que toutes les contraintes précédentes peuvent être utilisées dans un modèle de planification initiale. La replanification sera lancée comme si elle cherchait à replanifier toutes les tâches, mais celles qui sont déjà réalisées seront forcées à la date à laquelle elles ont été démarrées.

Chaque tâche déjà terminée/commencée au moment de la replanification ne pourra être modifiée :

$$\forall i \in D \quad \text{startOf}(x_{i,i.\text{worker}}) = i.\text{date} \tag{15}$$

Ici, $i.\text{worker}$ représente l'opérateur affecté à la tâche dans l'ordonnancement précédent et $i.\text{date}$ représente l'ancienne date de démarrage.

On force le démarrage des autres tâches après la date de lancement du nouvel ordonnancement (cela permet de contrôler la date à laquelle nous voulons démarrer) :

$$\forall i \in F \quad \text{startOf}(x_{i,i.\text{worker}}) \geq \text{actual\_date} \tag{16}$$

Ainsi, avec l'ensemble de ces contraintes, notre modèle peut être utilisé à la fois pour replanifier que pour générer

la planification initiale (en enlevant les contraintes (15) et (16)). L'objectif de cette étude est de tester ce modèle pour comparer les solutions générées dans différents cas que nous allons définir dans la section suivante.

4 RESULTATS EXPERIMENTAUX

4.1 Instances industrielles

Pour analyser les performances de notre modèle, nous avons utilisé un ensemble d'instances fournies par notre partenaire industriel du secteur aéronautique. Cet ensemble contient dix instances de 50 tâches et dix instances de 100 tâches.

Chaque instance possède entre onze et dix-neuf opérateurs pour un total d'un à quatre métiers. Nous détaillons dans les tableaux 1 et 2 les informations de chaque instance, dont les intitulés des colonnes sont tels quels :

- ID : identifiant de l'instance ;
- Z : nombre de zones ;
- M : nombre de machines ;
- P : ensemble des métiers (codés par chiffres);
- O : nombre d'opérateurs par métier (respectivement à P) ;
- S : nombre de zones avec plus d'un successeur.

ID	Z	M	P	O	S
50_1	106	0	1, 2, 3	11, 3, 4	6
50_2	111	0	1, 2, 3, 4	11, 3, 4, 1	5
50_3	57	1	1, 2	11, 4	3
50_4	62	0	1, 2	11, 4	3
50_5	110	1	1	11	9
50_6	187	0	1, 2	11, 4	7
50_7	217	1	1, 2	11, 4	8
50_8	170	0	1, 2, 3	11, 3, 4	6
50_9	48	0	1, 2	11, 4	5
50_10	57	1	1, 2	11, 4	3

Tableau 1 – Paramètres des instances de 50 tâches

ID	Z	M	P	O	S
100_1	409	3	A, C	11, 4	19
100_2	252	0	A, T, C, U	11, 3, 4, 1	12
100_3	193	0	A, T, C, U	11, 3, 4, 1	14
100_4	207	0	A, T, C, U	11, 3, 4, 1	18
100_5	179	1	A, C	11, 4	9
100_6	89	0	A, C	11, 4	9
100_7	191	1	A, T, C	11, 3, 4	8
100_8	207	0	A, T, C, U	11, 3, 4, 1	23
100_9	181	1	A, C	11, 4	7
100_10	79	1	A, C	11, 4	7

Tableau 2 – Paramètres des instances de 100 tâches

4.2 Protocole de tests

Tout d'abord nous avons résolu les 20 instances décrites avec notre modèle PPC pour obtenir le planning initial. Pour y parvenir, nous avons utilisé notre modèle de PPC en supprimant les contraintes (14) et (15).

Pour simuler un aléa, nous avons procédé de la manière suivante. Deux types d'aléas sont considérés indépendamment :

- Une tâche est décalée ;
- Un opérateur est absent sur un créneau donné de son calendrier habituel.

Pour simuler le décalage d'une tâche, nous sélectionnons une tâche de façon aléatoire. Ensuite, nous regardons les dates de départ des tâches avant interruption et regardons celle qui aurait dû être la suivante à démarrer en fonction de la date à laquelle nous nous trouvons. Nous décidons ensuite d'imposer sa date de départ comme suit :

$$new\_date \geq old\_date + 1h$$

Ainsi, cela revient à simuler une tâche qui démarre en retard et de voir son impact sur le planning.
Pour l'absence d'un opérateur, nous sélectionnons celui qui est le plus proche de terminer sa tâche pour ensuite forcer son arrêt jusqu'à la fin de la journée. Si c'est une tâche nécessitant plusieurs personnes, nous en sélectionnons une au hasard. Pour simuler cet aléa, nous rajoutons une période de non-travail dans le calendrier de l'opérateur sur le restant de la journée, ce qui permet de simuler un arrêt maladie. Nous considérons également que la durée de son arrêt ne dépasse pas la reste de la journée car l'entreprise aura la possibilité de faire venir quelqu'un pour le remplacer d'ici la journée suivante. Nous considérons que la tâche sur laquelle il travaillait se termine grâce à des équipe d'intervention, mais qu'il est par conséquent nécessaire de replanifier sa charge du travail restante.

Comme indiqué ci-dessus, nous testons différents critères d'optimisation en leur affectant une priorité différente. Le fait d'interchanger ces critères dans la fonction (4) résulte en six fonctions objectif différentes :

- staticLex(makespan, workers, decalages) (i)
- staticLex(makespan, decalages, workers) (ii)
- staticLex(workers, makespan, decalages) (iii)
- staticLex(workers, decalages, makespan) (iv)
- staticLex(decalages, workers, makespan) (v)
- staticLex(decalages, makespan, workers) (vi)

Nous espérons ainsi dégager dans un premier temps les critères qui sont les plus robustes en termes d'ordonnancement mais également la fonction qui offre les meilleurs compromis en termes d'optimalité pour chaque critère par rapport aux deux autres.

Enfin, pour se rapprocher au maximum des conditions de l'ordonnancement réactif, nous imposons au solver une limite de dix minutes pour le temps de calcul. Comme dit auparavant, il est primordial de trouver une solution rapidement, par conséquent étendre les temps de calcul n'aurait aucun sens dans notre cas.

4.3 Discussion des résultats

Les tests ont été effectués avec IBM ILOG CPO 12.10 avec un ordinateur possédant un processeur Intel Core i7-8665U (4 cœurs 1.90GHz) et 16Go de mémoire.

Le premier élément que nous analysons est la capacité des fonctions objectif à trouver les meilleures valeurs pour chaque critère. Nous présentons une synthèse de cette analyse dans les tableaux 2 et 3. Le tableau 2 présente les résultats pour les instances où l'aléa est lié à l'absence d'un opérateur et le tableau 3 présente les résultats pour les instances où l'aléa est lié au décalage d'une tâche.

Dans les tableaux 2 et 3, les colonnes représentent les fonctions objectif utilisées comme introduit ci-dessus et les lignes présentent les résultats obtenus pour les critères suivants : le nombre d'affectations tâche-opérateur inchangées (A), le makespan (M) et le décalage cumulé des fins des tâches (D), « t » représente le temps de résolution. Le préfixe 50 indique que le résultat a été obtenu sur les instances de 50 tâches et le préfixe 100 indique que le résultat a été obtenu pour les instances de 100 tâches. Les valeurs des tableaux 2 et 3 indiquent le nombre de fois que la fonction objectif (colonne) a fourni le meilleur résultat pour le critère indiqué par la ligne. Prenons un exemple dans le tableau 2 qui présente les résultats pour les instances où la source de l'aléa est liée à l'absence d'un opérateur. Nous pouvons voir dans ce tableau que la fonction objectif (vi) qui utilise l'ordre des 3 critères considérés de la manière suivante : staticLex(decalages, makespan, workers) a fourni 3 fois la meilleur valeur pour le critère « le nombre d'affectations tâche-opérateur inchangées » (A) pour les instances de 50 tâches et 4 fois pour le même critère pour les instances de 100 tâches.

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
50_A	3	2	10	10	3	3
50_M	10	10	3	3	8	8
50_D	5	8	3	3	9	10
50_t	0	1	4	5	0	0
100_A	4	3	10	10	4	4
100_M	9	9	3	4	7	7
100_D	4	6	4	4	10	10
100_t	2	1	2	3	1	1

Tableau 2 - Occurrences des meilleures valeurs dans le cas de l'absence d'un opérateur

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
50_A	5	4	10	10	4	4
50_M	10	10	4	4	7	7
50_D	3	6	2	3	9	9
50_t	0	1	0	9	0	0
100_A	5	4	10	10	5	5
100_M	10	10	4	5	9	9
100_D	3	8	3	5	10	10
100_t	0	0	2	8	0	0

Tableau 3 - Occurrences des meilleures valeurs dans le cas d'une tâche reportée

Une première propriété de ces fonctions mise en avant dans les résultats est la corrélation présente entre l'optimisation du makespan et des décalages de la date de fin des tâches. Lorsqu'on regarde les lignes associées à 50_M, 50_D, 100_M et 100_D et qu'on les compare aux lignes 50_A et 100_A, il semblerait que l'amélioration d'un de ces deux critères implique l'amélioration du deuxième (la meilleure valeur est trouvée assez souvent dans ces cas). En revanche, ces deux critères vont impacter négativement les affectations qui vont peiner à atteindre la meilleure solution possible. Cela voudrait donc dire que chercher à avoir le meilleur makespan possible, ou limiter au maximum les décalages, irait à l'encontre de la conservation des plannings pour les opérateurs.
Cependant, chercher à replanifier selon les opérateurs et leurs affectations n'est pas une mauvaise solution pour autant. Lorsqu'on met en lien les lignes 50_A et 100_A avec les lignes 50_t et 100_t, nous pouvons observer que les meilleurs temps sont dans 75% des cas ceux des fonctions ayant pour l'objectif premier les affectations. Dans le cas des reports d'une tâche, cela semble même être la méthode la plus rapide avec 100% des meilleurs temps (dont 80% pour la fonction (iv)). Nous pouvons tirer plusieurs informations de ces résultats :

1. Premièrement, les fonctions qui se basent sur les affectations et les opérateurs en premier critère se résoudront plus rapidement, permettant d'obtenir un planning dans un délai plus court.

2. Et deuxièmement, le critère basé sur la sauvegarde des affectations tâches-opérateurs semble être le meilleur dans les cas des aléas liés aux tâches décalées. Chercher à optimiser en suivant ce critère semble donc être la meilleure décision.

Pour approfondir ces résultats, nous avons calculé dans les tableaux 4 et 5 la proximité avec la meilleure solution dans le pire des cas pour chaque fonction et chaque instance. Si on prend la fonction (i) pour les instances de 50 tâches du tableau 5, cela veut dire qu'au pire, la valeur trouvée est équivalente à 1,05 fois la meilleure solution, soit à 5% de celle-ci. Ensuite les figures 2-7 présentent respectivement :

- Figure 2 – la répartition des valeurs du critère d'affectation lors de l'absence d'un opérateur pour les instances de 100 tâches,
- Figure 3 – la répartition des valeurs du critère du makespan pour les instances de 50 tâches lors de l'absence d'un opérateur,

- Figure 4 – la répartition des valeurs du critère de déviation dans les instances de 100 tâches lors de l'absence d'un opérateur,
- Figure 5 – la répartition des valeurs du critère de déviation dans les instances de 50 tâches lors de l'absence d'un opérateur,
- Figure 6 – la répartition des temps de calcul pour les instances de 100 tâches lors d'un report de tâche,
- Figure 7 – la répartition des temps de calcul pour les instances de 100 tâches lors de l'absence d'un opérateur.

L'objectif à travers ces figures est maintenant de présenter les cas particuliers qui se sont présentés à nous et ce qu'ils semblent signifier pour la résolution du problème.

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
50_A	0,05	0,06	0,00	0,00	0,04	0,04
50_M	0,00	0,00	0,27	0,27	0,04	0,04
50_D	0,98	0,67	2,14	2,14	0,00	0,00
50_t	820,06	8,88	6,26	0,77	7,06	8,77
100_A	0,11	0,29	0,00	0,00	0,18	0,18
100_M	0,92	0,92	0,92	0,30	0,92	0,92
100_D	245,17	2,53	3207,83	3207,83	0,00	0,00
100_t	38,34	61,15	38,32	81,17	14,63	116,16

Tableau 4 - Proximité avec les meilleures solutions dans le cas de l'absence d'un opérateur

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
50_A	0,12	0,12	0,00	0,00	0,10	0,10
50_M	0,00	0,00	0,31	0,30	0,01	0,01
50_D	10,00	1,04	428,71	21,06	0,00	0,00
50_t	27,25	11,65	4,56	1,32	9,83	9,36
100_A	0,04	0,05	0,00	0,00	0,05	0,05
100_M	0,00	0,00	0,07	0,07	0,01	0,01
100_D	28,55	8,57	80,43	3,16	0,00	0,00
100_t	177,09	139,24	120,51	3,66	105,09	105,12

Tableau 5 - Proximité avec les meilleures solutions dans le cas d'une tâche reportée

Le premier constat que nous pouvons faire en analysant les résultats de ces tableaux concerne la robustesse du critère d'affectation. En comparant les lignes 50_A et 100_A des deux tableaux, nous observons qu'au pire des cas, la valeur de la déviation maximale est à 12% pour les instances à 50 tâches et à 29% pour les instances à 100 tâches.

Or si l'on se penche sur la figure 2, nous pouvons voir que la répartition reste cependant très centrée malgré la présence de ces valeurs extrêmes. Cela semble donc indiquer que ces valeurs supérieures restent peu fréquentes comparées à l'ensemble des instances. La majorité de ces valeurs se situent en dessous de la valeur médiane car les affectations est un critère que nous cherchons à maximiser pour conserver au maximum les plannings de chaque opérateur.

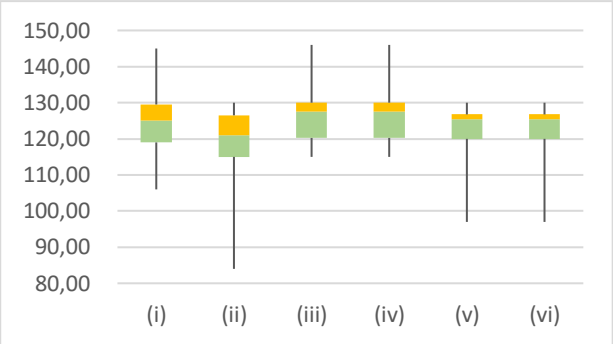

Figure 2 - Répartition des valeurs du critère d'affectation lors de l'absence d'un opérateur pour les instances de 100 tâches.

Nous pouvons donc estimer que le critère des affectations est un critère qui donnera des valeurs dans un espace plus restreint, permettant une robustesse dans l'élaboration des plannings, sans exclure pour autant la possibilité d'aboutir à des valeurs extrêmes dans les cas des plannings très perturbés par la replanification. Cela se vérifie d'ailleurs dans les autres cas où la répartition est similaire aux résultats des fonctions (iii) et (iv) de la figure 2.

Pour le critère du makespan dans les cas des 100 tâches, la répartition est également très centrée autour de la médiane, avec quelques valeurs extrêmes. Or, pour les instances de 50 tâches, les valeurs sont beaucoup plus réparties comme le montre la figure 3. Avec cette plus grande répartition survient une diminution des valeurs extrêmes, et une grande majorité se trouve cette fois-ci au-dessus de la valeur médiane, car nous cherchons ici à minimiser notre critère et donc s'éloigner de ce minimum revient à augmenter la valeur calculée.

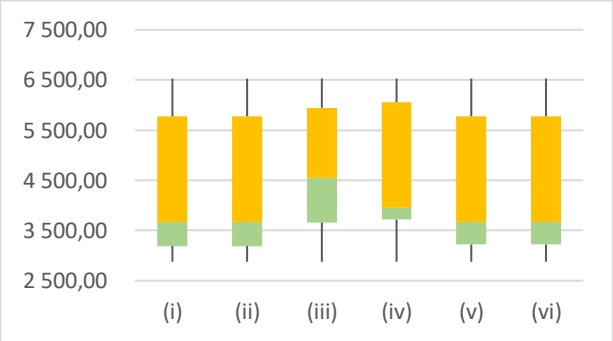

Figure 3 - Répartition des valeurs du critère du makespan lors de l'absence d'un opérateur pour les instances de 50 tâches.

Le makespan est donc un objectif qui trouvera ses valeurs dans un plus grand espace de solution, mais dont les valeurs extrêmes seront peu éloignées de ces espaces. Par conséquent, de nombreuses solutions fournirons la valeur du makespan relativement proche.

Le cas des déviations est celui où nous avons trouvé les valeurs les plus extrêmes (notamment dans le cas 100_D du tableau 4). Nous avons mis en forme ces données dans la figure 4.

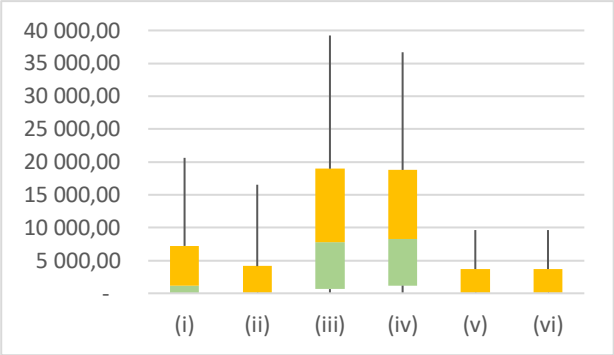

Figure 4 - Répartition des valeurs du critère de déviation pour lors de l'absence d'un opérateur pour les instances de 100 tâches.

Fort est de constater que la répartition est la plus forte pour les fonctions (iii) et (iv), fonctions qui privilégient les affectations. Nous en avions parlé en analysant les premiers tableaux, et cette figure semble le confirmer, mais prendre des décisions en se basant sur la conservation des anciens plannings va fortement impacter les déviations, et même plus fortement que le makespan en lui-même. Cela montre aussi que le makespan et les déviations semblent être caractérisés par une faible présence des valeurs en dessous de la médiane (portion verte) pour les fonctions (i) et (ii). Cela veut dire qu'une grande partie des valeurs de déviation sont optimales dans ces cas-là également. La figure 5 montre que c'est également le cas pour les instances de 50 tâches, et nous affirmons que ces résultats se vérifient aussi dans le cadre d'un aléa de tâche reportée.

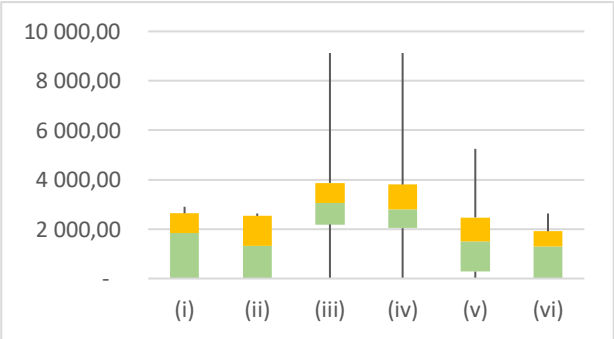

Figure 5 - Répartition des valeurs du critère de déviation dans lors de l'absence d'un opérateur les instances de 50 tâches.

Le constat est moins fort que lorsqu'il y a 100 tâches, mais conserve cependant la même tendance. Cela veut dire que pour obtenir une bonne valeur pour le critère de déviation, il est préférable de se focaliser sur le makespan ou les déviations elles-mêmes. Dans le cas contraire, la solution sera très peu robuste sans être très mauvaise forcément. Etant donné qu'il ne s'agit ici que de décalages, l'échange de deux tâches très éloignées l'une de l'autre va provoquer une très forte dégradation du critère de décalage, alors que les objectifs en termes de makespan et d'affectation pourraient être fortement améliorés grâce à cette opération. Il faut par conséquent rester vigilant quant à l'interprétation de ce critère.

Le dernier critère qu'il nous reste à analyser est celui du temps. Nous allons ici nous pencher sur les instances de 100 tâches car celles-ci auront une amplitude potentiellement plus grande de par leur complexité.

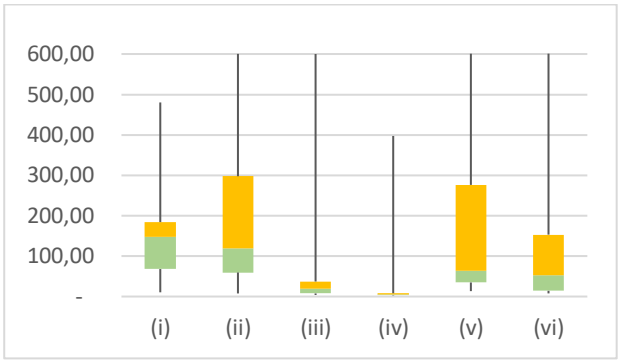

Figure 6 - Répartition des temps de calcul dans les instances de 100 tâches lors d'un report de tâche.

Il faut savoir que les données concernant les temps de calcul sont les plus hétérogènes. Cependant, nous avons pu voir qu'un critère semblait ressortir de toutes ces courbes. Si l'on se penche sur la figure 6, il est possible de voir que malgré les valeurs extrêmes, les fonctions (iii) et (iv) sont celles qui semblent les plus centrées autour d'une même valeur. Cet écrasement autour d'une valeur faible nous permet d'émettre l'hypothèse que ces fonctions sont celles qui vont déterminer une solution le plus rapidement. Cette conjecture semble se vérifier avec les informations des autres instances et dont les courbes ont une allure générale similaire.

Nous pouvons également voir que plusieurs instances ont atteint la date limite de dix minutes qui, nous le rappelons, était un critère d'arrêt lors des calculs. Ces temps extrêmes ne sont cependant pas présents dans les instances de 50 tâches et la répartition est très similaire par rapport aux médianes. La conjecture que nous pouvons faire grâce à ces résultats est que prioriser les affectations amène à les conserver, et donc décaler les tâches ce qui ressemble au comportement du « right shift scheduling », méthode mentionnée dans les travaux de Chaari et al. (2014) dont nous avons parlé dans notre étude de la littérature.

Pour finir, voici dans la figure 7 un cas extrême que nous avons pu rencontrer.

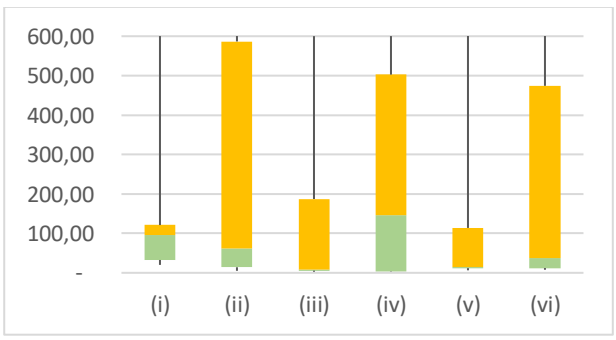

Figure 7- Répartition des temps de calcul dans les instances de 100 tâches lors de l'absence d'un opérateur.

Nous constatons que pour chaque critère, la résolution d'au moins une instance a atteint le critère d'arrêt de dix minutes. Nous n'avons cependant pas encore d'explications suite à ces résultats et des travaux futurs pourrons potentiellement porter sur l'étude des paramètres qui influencent la complexité de chaque instance.

CONCLUSION

Dans cet article, nous avons étudié la problématique de gestion d'aléas dans le contexte d'une chaine d'assemblage aéronautique. En particulier, nous nous sommes intéressés au développement d'une méthode réactive permettant de replanifier un ordonnancement de production devenu irréalisable après un aléa de type de décalage d'une tâche suite au retard de l'approvisionnement ou l'absence d'un opérateur sur un créneau de travail. gérer l'incertitude dans les projets d'ordonnancement. Dans cette situation, l'objectif est donc de générer un nouveau planning le plus rapidement possible afin de ne pas retarder la finalisation de l'assemblage : c'est la replanification réactive, car elle se trouve en aval de l'aléa.

Nous avons donc eu pour objectif de tester différents critères de replanification. Le premier est la minimisation du makespan. Le second vise à conserver au maximum le planning des opérateurs : on parle de maximisation des affectations. Le troisième concerne les dates de fin des tâches : on cherche à faire en sorte que celles-ci soient identiques au maximum à celles dans l'ordonnancement avant perturbation. Ces trois critères furent ainsi associés dans une fonction multi-objectifs, et leur priorité a été modifiée afin de voir l'impact qu'ont ces variations sur la replanification.

Nous avons mis en place un modèle de PPC afin de tester ces différentes fonctions. Pour réaliser ces tests, nous avons utilisé des instances fournies par les entreprises partenaires de notre projet. En utilisant notre modèle PPC réduit, nous avons généré des plannings initiaux, puis un aléa. Cet aléa pouvait prendre deux formes : une tâche étant infaisable à la date à laquelle elle était prévue auparavant par exemple à cause d'un retard d'approvisionnement, et doit donc être retardée ; ou un opérateur ne peut assurer son planning jusqu'à la fin de la journée en ne pouvant plus travailler. Ces deux aléas seront appliqués à chaque instance générée, mais de manière séparée. Par conséquent, deux instances de replanification ont été générées pour chaque instance. Au total, nous avons testé donc quarante instances sur six fonctions multi-objectifs différentes.

Ces tests nous ont donc permis de montrer l'impact qu'ont les différentes priorités dans les fonctions multi-objectifs sur les critères énoncés auparavant. Nous avons vu que le critère d'affectations, s'il est prioritaire, nécessite un temps de calcul très court, mais semble s'apparenter à la méthode du right shift scheduling. D'un autre côté, les critères de makespan et de décalages semblent être liés entre eux et aller à l'encontre des affectations (un bon makespan impliquera de dissocier plus d'opérateurs de leurs tâches d'origine). Nous avons également vu que le critère d'affectations est relativement robuste, ce qui semble indiquer qu'il n'est pas le critère le plus prioritaire. Le mettre en deuxième position pour avoir le makespan ou les décalages en première semble former un excellent compromis entre la date de fin de l'assemblage, la conservation des plannings des opérateurs et le temps de calcul.

L'objectif est désormais de faire des tests sur des instances complètes afin de comparer nos résultats avec nos partenaires. Etudier l'impact qu'ont les différentes données des instances sur la complexité du problème peut également être piste pour mieux comprendre les observations faites dans la section 4. Enfin, une fois ces objectifs atteints, nous pourrons envisager l'implémentation du modèle de PPC dans un outil qui peut être utilisé par des industriels.

REMERCIEMENTS

Ce travail de recherche a été financé par le projet ANR PER4MANCE. Nous remercions les entreprises partenaires du projet pour leur aide et collaboration.

REFERENCES

Bhadury J., Mighty E.J. et Damar H., 2012. Maximizing workforce diversity in project teams: a network flow approach. *Omega*, vol. 28, p. 143-153.

Chaari T., Chaabane S., Aissani N. et Trentesaux D., 2014. Scheduling under uncertainty: Survey and research directions. *2014 International Conference on Advanced Logistics and Transport,* Hammamet, Tunisie.

Gharbi A., Pellerin R. et Villeneuve L., 1999. A new constraint-based approach for overhaul project scheduling with workspace constraints. *International journal of industrial engineering*.

Hartmann S. et Briskorn D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, vol. 207, n°1, p. 1-14.

Hazir Ö. et Ulusoy G., 2020. A classification and review of approaches and methods for modeling uncertainty in projects. *International Journal of Production Economics*, vol. 223.

Kuster J., Jannach D. et Friedrich G., 2008. Extending the RCPSP for modeling and solving disruption management problems. *Applied Intelligence*, vol. 31, n°3, p. 234-253.

Valls V., Pérez Á. et Quintanilla S., 2009. Skilled workforce scheduling in Service Centres. *European Journal of Operational Research*, vol. 193, p. 791-804.

Zhu G., Bard J. et Yu G., 2005. Disruption management for resource-constrained project scheduling. *Journal of the Operational Research Society*, vol. 56, p. 365-381