Approche intégrée pour des problèmes à un niveau

Ce chapitre présente une méthode d'optimisation permettant de résoudre de manière intégrée le problème de planification et d'ordonnancement de la production mono-niveau, dans des systèmes de production complexes de type job-shop[1]*. L'approche permet d'obtenir des solutions réalisables proches de l'optimum pour différentes tailles de problèmes, avec des temps de calcul raisonnables.*

3.1 *Introduction*
3.2 *Modélisation*
3.3 *Méthode de résolution*
3.4 *Analyses et améliorations*
3.5 *Résultats expérimentaux*
3.6 *Conclusion*

1. Des parties de ce chapitre ont été publiées dans les conférences **ROADEF2011** [11], **IWLS2011** [84], **ROADEF2012** [87], **ILS2012** [92] et **IESM2013** [88] et dans la revue **International Journal of Production Research** [91].

3.1 Introduction

Comme montré dans le chapitre précédent, peu de travaux dans la littérature ont étudié l'intégration des décisions de planification et d'ordonnancement de la production dans des systèmes complexes, afin de déterminer au niveau tactique des plans de production réalisables au niveau opérationnel, i.e. des plans qui respectent les contraintes de capacité détaillées. Parmi les approches présentées, nous nous intéressons à celle de Wolosewicz *et al.* [233, 234, 235], qui est une version évoluée de plusieurs approches et qui, contrairement aux autres travaux, considère le problème d'ordonnancement sur tout l'horizon de planification, permettant ainsi d'utiliser efficacement la capacité de production.

L'approche de Wolosewicz *et al.*, bien qu'elle offre une bonne performance par rapport au solveur commercial IBM ILOG CPLEX, peut être améliorée en termes de qualité de la solution et de temps de calcul, en modifiant plusieurs parties de l'algorithme, comme nous allons le constater dans ce chapitre. En fait, nous proposons, dans ce qui suit, une nouvelle approche basée sur le principe de combinaison d'une heuristique Lagrangienne et d'une recherche taboue pour résoudre le problème intégré dans des systèmes complexes multi-produits, multi-opérations et multi-ressources avec partage de ressources et avec une politique d'ordonnancement sur tout l'horizon de planification.

Il s'agit d'une approche décomposant le problème en un ensemble de sous-problèmes avec séquence fixée, pour diminuer l'effort de calcul, avec une heuristique qui permet d'obtenir des solutions de bonne qualité. Cette stratégie est basée sur une reformulation du modèle intégré de [48], comme introduit dans [234] pour résoudre le problème à séquence fixée.

Nous commençons ce chapitre par une description du problème auquel nous nous intéressons, ainsi que sa modélisation, dans la Section 3.2. Ensuite, dans la Section 3.3, nous expliquons les principes de la méthode de résolution. Dans la Section 3.4, nous décrivons les différentes améliorations qui ont été apportées à la méthode de Wolosewicz *et al.*, ainsi que les différentes analyses réalisées. Puis, dans la Section 3.5, nous comparons les résultats obtenus avec ceux de l'approche de Wolosewicz *et al.* et ceux du solveur IBM ILOG CPLEX. Enfin, nous concluons dans la Section 3.6 sur la pertinence de cette approche.

3.2 Modélisation

3.2.1 Description du problème

Nous nous intéressons à la résolution du problème intégré dans des systèmes de type job-shop. Nous voulons planifier la production de N produits sur T périodes à capacité finie, en utilisant M machines, tout en ordonnançant O opérations qui composent les étapes de production (gamme) des différents ordres de fabrication (jobs). Le but est de satisfaire les demandes des différents clients dans les délais (la rupture de stock n'est pas autorisée) au moindre coût, incluant la somme des coûts de production, de stockage et de setup (ou lancement de la production). Le traitement d'une opération ne peut pas être arrêté avant que celle-ci ne soit terminée (la préemption n'est pas autorisée), et une machine ne peut traiter qu'une opération à la fois. Des délais d'obtention sont définis pour chaque produit. La demande est dynamique et connue à l'avance. La capacité de production du système est limitée, et la capacité de stockage est considérée infinie. Les durées unitaires opératoires et de setup des opérations sont connues à l'avance. Les coûts et les temps de setup sont indépendants de la séquence. Les notations utilisées pour définir les données considérées dans ce problème sont les suivantes :

- $D_{i,l}$: demande du produit i à la fin de la période l,
- c_i^p : coût unitaire de production du produit i,
- c_i^{inv} : coût unitaire de stockage du produit i,
- c_i^s : coût unitaire de setup du produit i,
- L_i : délai d'obtention du produit i,
- $i(o)$: produit associé à l'opération o,
- $l(o)$: période associée à l'opération o,
- $m(o)$: ressource sur laquelle l'opération o doit être traitée,
- p_o^u : durée opératoire de l'opération o par unité du produit $i(o)$,
- s_o^t : temps de setup de l'opération o par unité du produit $i(o)$,
- $r(o)$: date de disponibilité de l'opération o,
- $d(o)$: date de fin souhaitée de l'opération o,
- c_l : longueur de la période l (capacité disponible),
- $\mathcal{O}$: ensemble des opérations,
- $\mathcal{A}$: ensemble des paires d'opération dans la gamme des produits ($(o, o') \in \mathcal{A}$ signifie que l'opération o précède l'opération o' dans la gamme opératoire),
- $\mathcal{L}$: ensemble des dernières opérations dans les gammes opératoires,
- $\mathcal{F}$: ensemble des premières opérations dans les gammes opératoires,
- $\mathcal{E}$: ensemble des paires d'opérations qui sont traitées sur la même ressource,

– $\mathcal{S}(y)$: séquence des opérations associées à la séquence y ($(o, o') \in S(y)$ signifie que l'opération o précède l'opération o' dans la séquence d'une ressource).

Du point de vue de la planification, il s'agit de déterminer la taille de lot de chaque produit à chaque période, et du point de vue de l'ordonnancement, il s'agit de déterminer les dates de début et de fin de chaque opération et leur ordre de passage sur les machines.

Le problème d'ordonnancement est modélisé à travers un graphe disjonctif, où les nœuds correspondent aux opérations des jobs et les arcs représentent les contraintes entres les opérations. Les arcs reliant les opérations des gammes de produit ont toujours la même orientation (arcs conjonctifs) ; tandis que l'orientation des arcs reliant les opérations traitées sur la même ressource doit être définie (arcs disjonctifs). Le problème d'ordonnancement consiste donc à trouver la meilleure orientation pour tous les arcs disjonctifs, i.e. déterminer un graphe conjonctif sans créer des circuits. La Figure 3.1 présente un exemple de graphe disjonctif, ainsi qu'un graphe conjonctif possible pour ce graphe disjonctif.

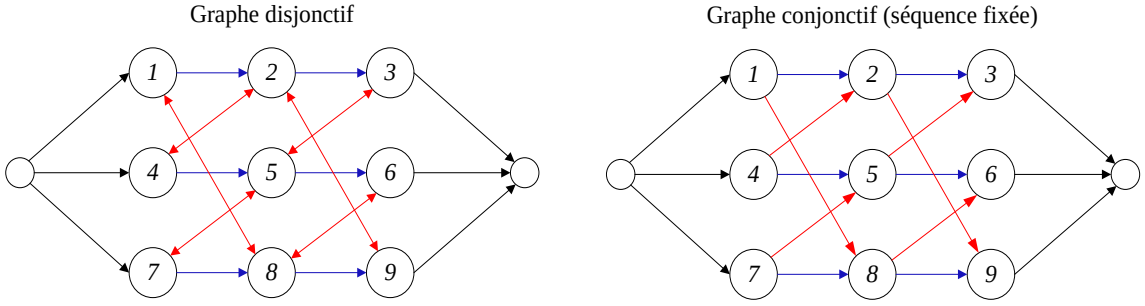

FIGURE 3.1 – Exemple d'un graphe disjonctif et d'un graphe conjonctif associé pour un atelier de type job-shop avec 3 jobs et 3 machines

Dans cet exemple, les opérations 1, 2 et 3 font partie de la gamme de fabrication du job J_1 et doivent être traitées dans l'ordre indiqué par les arcs conjonctifs. De la même façon, les opérations 4, 5 et 6 sont nécessaires pour produire le job J_2, tandis que les opérations 7, 8 et 9 permettent de produire le job J_3. Les opérations 1, 8 et 6 doivent être traitées sur la ressource R_1 ; les opérations 4, 2 et 9 sur la ressource R_2, et les opérations 7, 5 et 3 sur la ressource R_3.

3.2.2 Modèle intégré

L'objectif étant de minimiser la somme des coûts de planification, les **variables de décision** que nous considérons sont celles qui sont habituellement utilisées dans les modèles de dimensionnement de lots, i.e. *quand produire ?* et *combien produire ?*. Ces variables sont modélisées par les notations suivantes :

– $Y_{i,l}$: variable de setup (= 1 si le produit i est fabriqué à la période l, 0 sinon).
– $X_{i,l}$: taille de lot du produit i disponible à la fin de la période l,

Les **variables dépendantes** (variant en fonction de Y et X) du problème intégré sont les niveaux de stock des produits et les dates de début des opérations. Elles sont définies par les notations suivantes :

– $I_{i,l}$: inventaire du produit i à la fin de la période l,
– $t(o)$: date de début de l'opération o,

Les dates de début des opérations servent notamment à modéliser les contraintes de précédence entre les opérations faisant partie de la même gamme de fabrication et entre celles étant traitées sur la même ressource. Ces contraintes permettent de garantir la cohérence du plan de production au niveau ordonnancement.

Les contraintes de précédence entre les opérations d'un même job étant toujours les mêmes, indépendamment de la séquence, elles sont représentées par l'équation (3.1).

$$t(o') \geq t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \ \forall \ (o,o') \in \mathcal{A} \tag{3.1}$$

D'autre part, les contraintes de précédence entre les opérations partageant la même ressource sont modélisées par l'équation (3.2) qui tient compte de la disjonction.

$$\begin{cases} t(o') \geq t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \\ \text{ou} \qquad\qquad\qquad\qquad\qquad\qquad \forall (o,o') \in \mathcal{E} \\ t(o) \geq t(o') + p_{o'}^u X_{i(o'),l(o')} + s_{o'}^t Y_{i(o'),l(o')} \end{cases} \tag{3.2}$$

Si une séquence y est fixée (graphe conjonctif), les contraintes de précédence entre les opérations réalisées sur la même ressource sont définies par l'équation (3.3).

$$t(o') \geq t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \ \forall \ (o,o') \in \mathcal{S}(y) \tag{3.3}$$

Un modèle intégré de dimensionnement de lots et d'ordonnancement doit inclure les contraintes (3.1) et (3.2) pour garantir la recherche de la meilleure solution réalisable possible. L'équation (3.3) permet de garantir une solution réalisable, mais pas forcement optimale.

Nous utilisons dans ce chapitre le modèle MIP proposé par Dauzère-Pérès et Lasserre dans [48], mais sans tenir compte de l'aspect multi-niveaux ni de la rupture de stocks. C'est un modèle incluant les contraintes traditionnelles de dimensionnement de lots, avec en plus des contraintes

de capacité et d'ordonnancement détaillées, et en tenant compte des délais d'obtention des produits. Cette modélisation permet donc de planifier la production au niveau tactique, en incluant des décisions opérationnelles. Le modèle est le suivant :

$$\min \sum_{i=1}^{N} \sum_{l=1}^{T} \left(c_i^p X_{il} + c_i^{inv} I_{il} + c_i^s Y_{il} \right) \tag{3.4}$$

s.c.

$$I_{il} = I_{i,l-1} + X_{il} - D_{il} \qquad \forall\, i, l \tag{3.5}$$

$$t(o') \geq t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \qquad \forall\, (o, o') \in \mathcal{A} \tag{3.1}$$

$$\begin{cases} t(o) \geq t(o') + p_{o'}^u X_{i(o'),l(o')} + s_{o'}^t Y_{i(o'),l(o')} \\ \text{ou} \\ t(o') \geq t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \end{cases} \qquad \forall\, (o, o') \in \mathcal{E} \tag{3.2}$$

$$t(o) + p_o^u X_{i(o),l(o)} + s_o Y_{i(o),l(o)} \leq \sum_{l=1}^{l(o)} c_l \qquad \forall\, o \in \mathcal{L} \tag{3.6}$$

$$t(o) + p_o^u X_{i(o),l(o)} + s_o Y_{i(o),l(o)} \geq \sum_{l=1}^{l(o)-1} c_l \qquad \forall\, o \in \mathcal{L} \tag{3.7}$$

$$t(o) \geq \sum_{l=1}^{l(o)-L_{i(o)}} c_l \qquad \forall\, o \in \mathcal{F} \text{ et } L_{i(o)} > 0 \tag{3.8}$$

$$X_{il} \leq \left(\sum_{k=1}^{T} D_{ik} \right) Y_{il} \qquad \forall\, i, l \tag{3.9}$$

$$X_{il}, I_{il} \geq 0 \qquad \forall\, i, l \tag{3.10}$$

$$Y_{il} \in \{0, 1\} \qquad \forall\, i, l \tag{3.11}$$

$$t(o) \geq 0 \qquad \forall\, o \tag{3.12}$$

La fonction objectif (3.4) minimise le coût total (coûts de production, de stockage et de setup). Les contraintes (3.5) correspondent aux équations d'équilibre de stock. Les contraintes (3.1) permettent de garantir les contraintes de précédence entre les opérations faisant partie de la même gamme opératoire. Les contraintes (3.2) garantissent les contraintes de précédence entre les opérations partageant la même ressource. Les contraintes (3.6), (3.7) et (3.8) correspondent aux contraintes de capacité. Plus précisément, les contraintes (3.6) forcent la dernière opération de gamme à finir au plus tard à la date de fin de sa période associée ; les contraintes (3.7) forcent

la dernière opération de gamme à finir au plus tôt à la date de début de sa période associée et, les contraintes (3.8) forcent la première opération de gamme à démarrer au plus tôt à la date de début minimale respectant le délai d'obtention. Les contraintes (3.9) permettent de lier les variables de décisions (taille de lot et setup). Les contraintes (3.10) indiquent la non-négativité de la taille de lot et de l'inventaire. Les contraintes (3.11) forcent les variables de setup à prendre des valeurs binaires, et les contraintes (3.12) garantissent la non-négativité des dates de début des opérations.

3.2.3 Modèle avec chemins et séquence fixée

Afin de réduire la difficulté associée à la résolution du modèle intégré, nous utilisons une stratégie qui consiste à résoudre un sous-ensemble de problèmes de planification à séquence fixée. Une première version de cette approche a été proposée dans [233, 234, 235], et nous y avons apporté des modification jusqu'à obtenir une version améliorée que nous présentons dans la section 3.3. Cette approche est basée sur la résolution d'un modèle utilisant les chemins du graphe conjonctif associé à une séquence fixée.

La première modification consiste à exprimer les contraintes de capacité, non pas en fonction de la date de début des opérations, mais en fonction de la date de disponibilité, c'est-à-dire la date minimale à laquelle une opération peut démarrer. Ces dates sont définies à travers les contraintes (3.13), (3.14), (3.15) et (3.16). Les contraintes (3.13) permettent d'établir que la première opération de gamme doit démarrer à une date garantissant le respect du délai d'obtention. À travers les contraintes (3.14), la dernière opération de chaque gamme est disponible à partir de la date de début de sa période associée (période de production du produit $i(o)$). Les contraintes (3.15) définissent la date de disponibilité pour les opérations comme étant à la fois la première et la dernière dans la gamme de fabrication. La date de disponibilité n'étant pas considérée pour les opérations intermédiaires, les contraintes (3.16) sont utilisées pour formaliser la définition. La différence par rapport au modèle intégré précédent est que la dernière opération de chaque gamme doit démarrer et finir dans sa période associée. Dans le modèle intégré, cette opération peut démarrer à une période différente.

$$r(o) = \sum_{l=1}^{l(o)-L_{i(o)}} c_l \quad \forall\, o \in \mathcal{F} \text{ et } o \notin \mathcal{L} \tag{3.13}$$

$$r(o) = \sum_{l=1}^{l(o)-1} c_l \quad \forall\, o \in \mathcal{L} \text{ et } o \notin \mathcal{F} \tag{3.14}$$

$$r(o) = \max \left(\sum_{l=1}^{l(o)-L_{i(o)}} c_l, \sum_{l=1}^{l(o)-1} c_l \right) \quad \forall\, o \in \mathcal{F} \cap \mathcal{L} \tag{3.15}$$

$$r(o) = 0 \quad \forall\, o \notin \mathcal{F} \cup \mathcal{L} \tag{3.16}$$

La première opération de chaque job peut démarrer à une période entre 0 et $l(o)$ si $L_{i(o)} = 0$ ou entre $l(o) - L_{i(o)}$ et $l(o)$ si $L_{i(o)} > 0$. La dernière opération de chaque job doit démarrer et terminer dans sa période associée $l(o)$. La date de disponibilité $r(o)$ est connue uniquement pour la première et la dernière opération de chaque job, et une date de fin souhaitée $d(o)$ est imposée uniquement pour la dernière opération de chaque job, comme défini par (3.17). Le respect de la date de fin souhaitée est garanti par (3.25).

$$d(o) = \sum_{l=1}^{l(o)} c_l \;\forall\, o \in \mathcal{L} \tag{3.17}$$

D'autres variables dépendantes qui ne font pas partie de la modélisation mathématique, mais qui sont utilisées par le graphe conjonctif pour le calcul des marges de capacité, sont les suivantes :

- $t_e(o)$: date de début au plus tôt de l'opération o,
- $t_l(o)$: date de début au plus tard de l'opération o,
- $f(o)$: date de fin de l'opération o,
- $f_e(o)$: date de fin au plus tôt de l'opération o,
- $f_l(o)$: date de fin au plus tard de l'opération o.

Les Définitions 1 et 2 permettent de lier les dates de début au plus tôt $t_e(o)$ et plus tard $t_l(o)$ et les dates de fin au plus tôt $f_e(o)$ et au plus tard $f_l(o)$ de l'opération o.

- **Définition 1 :** $t_e(o) \leq t(o) \leq t_l(o)\ \forall\, o \in \mathcal{O}$.
- **Définition 2 :** $f_e(o) \leq f(o) \leq f_l(o)\ \forall\, o \in \mathcal{O}$.

En suivant la logique du graphe conjonctif, les dates de début au plus tôt sont déterminées de manière récursive, tout au long de l'horizon de planification, à partir du sommet source O (avec $t_e(\mathrm{O}) = 0$), et ensuite, les dates de début au plus tard sont déterminées d'avant en arrière, à partir du sommet puits $*$. Comme illustré sur la Figure 3.2, chaque sommet du graphe conjonctif représente une opération, chaque sommet ayant plusieurs prédécesseurs et successeurs, que nous généralisons dans le Tableau 3.1.

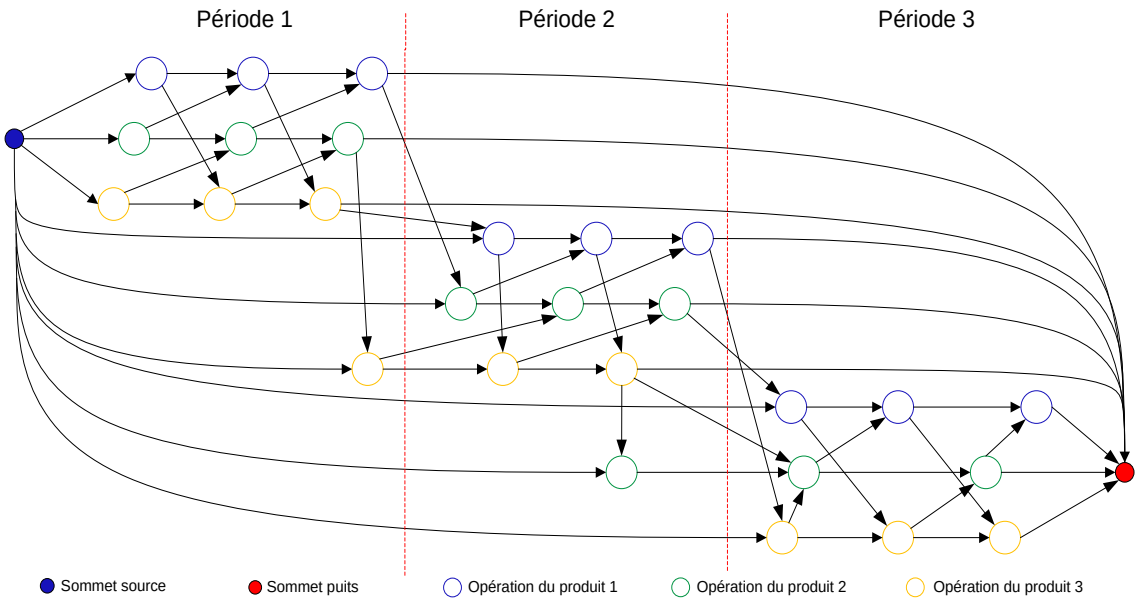

FIGURE 3.2 – Graphe conjonctif pour un atelier de type job-shop avec 3 produits, 3 machines 3 périodes

TABLEAU 3.1 – Prédécesseurs et successeurs dans le graphe conjonctif

Sommet	**Prédécesseurs**	**Successeurs**
Source (O)	Aucun	$\mathcal{F}$
Première opération de gamme ($o \in \mathcal{F}$)	- O - o' avec $(o', o) \in \mathcal{S}(y)$	- o' avec $(o, o') \in \mathcal{A}$ - o'' avec $(o, o'') \in \mathcal{S}(y)$
Opération intermédiaire ($o \notin \mathcal{F} \cup \mathcal{L}$)	- o' avec $(o, o') \in \mathcal{A}$ - o'' avec $(o'', o) \in \mathcal{S}(y)$	- o' avec $(o, o') \in \mathcal{A}$ - o'' avec $(o, o'') \in \mathcal{S}(y)$
Dernière opération de gamme ($o \in \mathcal{L}$)	- o' avec $(o', o) \in \mathcal{A}$ - o'' avec $(o'', o) \in \mathcal{S}(y)$	- o' avec $(o, o') \in \mathcal{S}(y)$ - $*$
Puits ($*$)	$\mathcal{L}$	Aucun

Appelons maintenant $\mathcal{SP}(s)$ l'ensemble de sommets prédécesseurs et $\mathcal{SS}(s)$ l'ensemble de sommets successeurs du sommet $s \in \mathcal{O} \cup \{\mathrm{O}, *\}$. Pour chaque prédécesseur n du sommet s ($n \in \mathcal{SP}(s)$), une date fictive $\lambda(n)$ est calculée avec (3.18).

$$\lambda(n) = \begin{cases} r(s) & \text{si} \quad n = \mathrm{O} \\ t_e(n) + p_n^u X_{i(n),l(n)} + s_n^t Y_{i(n),l(n)} + \sum_{k=l(n)+1}^{T} c_l & \text{si} \quad s = * \\ t_e(n) + p_n^u X_{i(n),l(n)} + s_n^t Y_{i(n),l(n)} & \text{sinon} \end{cases} \tag{3.18}$$

À noter que dans ce cas, $r(s)$ est la longueur de l'arc entre $n = \mathrm{O}$ et s quand $s \in \mathcal{F}$; $p_n^u X_{i(n),l(n)} + s_n^t Y_{i(n),l(n)} + \sum_{k=l(n)+1}^{T} c_l$ est la longueur de l'arc entre n et $s = *$ quand $n \in \mathcal{L}$ et, $p_n^u X_{i(n),l(n)} + s_n^t Y_{i(n),l(n)}$ est la longueur de l'arc entre n et s quand $s \notin \mathcal{F} \wedge n \notin \mathcal{L}$.

Les dates de début au plus tôt sont finalement calculées avec (3.19).

$$t_e(s) = \begin{cases} r(s) \text{ si } s \in \mathcal{L} \wedge \max\limits_{n \in \mathcal{SP}(s)} \lambda(n) < r(s) \\ \max\limits_{n \in \mathcal{SP}(s)} \lambda(n) \text{ sinon} \end{cases} \quad \forall\ s \in \mathcal{O} \cup \{*\} \tag{3.19}$$

De manière similaire, pour chaque successeur n du sommet s ($n \in \mathcal{SS}(s)$), une date fictive $\mu(n)$ est calculée avec (3.20). La date de début au plus tard du sommet puits $t_l(*)$ est fixée comme étant égale à la date de début au plus tôt $t_e(*)$.

$$\mu(n) = \begin{cases} t_l(n) - r(n) & \text{si} \quad s = \mathrm{O} \\ t_l(n) - \left(p_s^u X_{i(s),l(s)} + s_s^t Y_{i(s),l(s)} + \sum_{k=l(s)+1}^{T} c_l \right) & \text{si} \quad n = * \\ t_l(n) - \left(p_s^u X_{i(s),l(s)} + s_s^t Y_{i(s),l(s)} \right) & \text{sinon} \end{cases} \tag{3.20}$$

Les dates de début au plus tard sont finalement déterminées avec (3.21).

$$t_l(s) = \min_{n \in \mathcal{SS}(s)} \mu(n)\ \forall\ s \in \mathcal{O} \cup \{\mathrm{O}\} \tag{3.21}$$

Connaître les dates de début au plus tôt et au plus tard des opérations permet de calculer les marges de capacité et de gérer des éventuelles modifications du plan de production. La date de début au plus tôt du sommet puits (égale à la date de début au plus tard) correspond à la date de fin du dernier job du plan de production. Donc, **la capacité est violée si** $t_e(*) =$

$t_l(*) > \sum_{l=1}^{T} c_l$. Par ailleurs, minimiser le *makespan* correspond à minimiser la valeur de $t_e(*)$. La date de fin d'une opération o quelconque peut être déterminée à travers l'équation (3.22).

$$f(o) = t(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \ \forall \ o \in \mathcal{O} \tag{3.22}$$

Les dates de fin au plus tôt et au plus tard, liées par la Définition 2, peuvent être calculées avec (3.23) et (3.24), respectivement.

$$f_e(o) = t_e(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \ \forall \ o \in \mathcal{O} \tag{3.23}$$

$$f_l(o) = t_l(o) + p_o^u X_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \ \forall \ o \in \mathcal{O} \tag{3.24}$$

Dans le modèle mathématique avec chemins et séquence fixée, un nouveau type de contraintes est introduit pour gérer la cohérence des liens de précédence entre les opérations et pour intégrer la capacité de production. Cette expression permet de modéliser les contraintes de capacité détaillées, en suivant les informations du graphe conjonctif. Pour chaque chemin c du graphe conjonctif, il existe une contrainte de capacité détaillée. La formulation est la suivante :

$$\min \sum_{i=1}^{N} \sum_{l=1}^{T} \left(c_i^p X_{il} + c_i^{inv} I_{il} + c_i^s Y_{il} \right) \tag{3.4}$$

s.c.

$$I_{il} \ = \ I_{i,l-1} + X_{il} - D_{il} \qquad \forall \ i, l \tag{3.5}$$

$$r(o_c^f) + \sum_{o \in c} \left(p_o^u X_{i(o)l(o)} + s_o^t Y_{i(o)l(o)} \right) \leq \sum_{l=1}^{l(o_c^l)} c_l \qquad \forall \ c \in \mathcal{C}(y) \tag{3.25}$$

$$X_{il} \leq \left(\sum_{k=1}^{T} D_{ik} \right) Y_{il} \qquad \forall \ i, l \tag{3.9}$$

$$X_{il}, I_{il} \geq 0 \qquad \forall \ i, l \tag{3.10}$$

$$Y_{il} \in \{0, 1\} \qquad \forall \ i, l \tag{3.11}$$

La fonction objectif (3.4) et les contraintes d'équilibre des stocks (3.5) du modèle à séquence fixée sont similaires à celles du modèle intégré. La différence réside dans les contraintes (3.25), qui représentent les contraintes de capacité détaillées pour une séquence y, où c représente un chemin et $\mathcal{C}(y)$ est l'ensemble des chemins dans le graphe, et o_c^f et o_c^l sont les première et dernière

opérations du chemin c, respectivement. La différence entre les contraintes (3.25) et (3.3) est que les premières considèrent la capacité des périodes, tandis que les deuxièmes respectent uniquement les contraintes de précédence. Finalement, de la même manière que dans le modèle intégré, les contraintes (3.9) relient les variables de production X aux variables de setup Y, les contraintes (3.10) assurent la non-négativité des variables et les contraintes (3.11) gèrent la valeur des variables binaires de setup.

Ce modèle mathématique est une généralisation du CLSP, avec des contraintes de capacité détaillées qui servent à garantir le respect des contraintes de précédence pour une séquence fixée, en utilisant les informations dérivées du graphe conjonctif. Résoudre un problème de dimensionnement de lots avec des contraintes de capacité agrégées, comme le CLSP, est $\mathcal{NP}$-difficile pour les ateliers à une seule machine [31]. Donc, utiliser des contraintes de capacité détaillées pour des problèmes multi-ressources avec partage (de type job-shop), rend le problème encore plus difficile à résoudre. Le nombre de chemins dans le graphe conjonctif peut être très important (sa croissance est en effet exponentielle), et satisfaire toutes les contraintes de capacité peut devenir impossible. Notre stratégie de résolution consiste à appliquer une heuristique Lagrangienne pour résoudre le problème de dimensionnement de lots et d'ordonnancement avec séquence fixée, intégrée dans une recherche taboue qui permet d'améliorer la séquence.

3.3 Méthode de résolution

La plupart des problèmes de dimensionnement de lots avec contraintes de capacité sont $\mathcal{NP}$-difficiles. Si l'on ajoute des contraintes du type (3.1) et (3.2), le problème devient encore plus difficile à résoudre. Par ailleurs, le problème d'ordonnancement dans un atelier de type job-shop multi-périodes est l'un des problèmes les plus difficiles à résoudre ($\mathcal{NP}$-difficile au sens fort). La combinaison des variables et des contraintes des deux problèmes est donc $\mathcal{NP}$-difficile. L'implémentation d'une méthode exacte permettant de trouver des solutions optimales pour des problèmes de taille raisonnable paraît impossible. En outre, dans plusieurs cas, les solveurs commerciaux sont même incapables de trouver des solutions réalisables pour des problèmes de ce type. Face à ce constat, nous proposons une approche basée sur des heuristiques et illustrée dans la Figure 3.3. Premièrement, nous résolvons le problème de dimensionnement de lots avec contraintes de capacité détaillées pour une séquence fixée, et deuxièmement nous modifions le graphe conjonctif pour déterminer une nouvelle séquence. Cette procédure est répétée durant un certain nombre d'itérations.

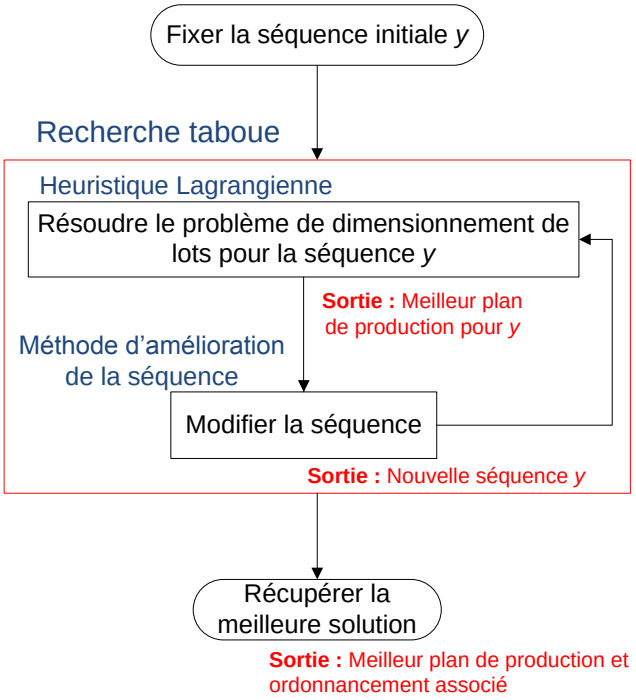

FIGURE 3.3 – Approche intégrée

3.3.1 Heuristique Lagrangienne pour la résolution du problème à séquence fixée

Pour réduire la difficulté de résolution du problème de dimensionnement de lots avec séquence fixée, nous utilisons une heuristique Lagrangienne qui a pour but de : (i) calculer un plan de production optimal pour la séquence fixée avec contraintes de capacité relâchées, et (ii) réparer les contraintes de capacité non respectées afin de construire une solution réalisable.

3.3.1.1 Relaxation Lagrangienne

La stratégie consiste à relâcher les contraintes de capacité, de façon à résoudre un ensemble de problèmes mono-produit à capacité infinie, en utilisant la propriété de Wagner et Whitin [231]. Seuls les coûts de setup et de stockage ont une influence sur le choix des variables de décision.

Pour chaque problème à un produit, nous combinons l'algorithme dynamique de dimensionnement de lots proposé par Wagelmans *et al.* [230], où chaque problème peut être résolu en un temps $O(T \log T)$, avec la technique de la relaxation Lagrangienne, en utilisant la méthode

des sous-gradients ([69] ; [146]). La fonction (3.26) est la fonction objectif du problème dual, où β_c correspond au multiplicateur Lagrangien associé au chemin $c \in \mathcal{C}(y)$. Cette fonction est exprimée en termes des variables de décision X et Y et des multiplicateurs Lagrangiens β.

$$\begin{aligned} F(X,Y,\beta) = \min \sum_{i=1}^{N}\sum_{l=1}^{T} \Bigg\{ \Bigg(c_i^s + \sum_{c\in C(y)} \beta_c \sum_{o\in c} s_o^t \Bigg) Y_{i,l} + \Bigg[c_i^p + \sum_{k=l}^{T} c_i^{inv} \\ + \sum_{c\in C(y)} \beta_c \sum_{o\in c} p_o^u \Bigg] X_{i,l} \Bigg\} - \sum_{i=1}^{N}\sum_{l=1}^{T} D_{i,l} \sum_{k=l}^{T} c_i^{inv} \\ + \sum_{c\in C(y)} \beta_c \left[r(o_c^f) - \sum_{l=1}^{l(o_c^l)} c_l \right] \end{aligned} \tag{3.26}$$

Comme le nombre de chemins $\mathcal{C}(y)$ dans le graphe conjonctif est exponentiel, nous mettons à jour dans la procédure uniquement les multiplicateurs Lagrangiens associés à l'ensemble $\mathcal{CH}(y)$ des chemins les plus violés. Un chemin c appartient à $\mathcal{CH}(y)$ si sa contrainte de capacité associée est la plus violée de tout le graphe à une itération donnée, i.e. si le chemin c est violé et permet de maximiser l'expression $r(o_c^f) + \sum_{o\in c} \left(p_o^u X_{i(o)l(o)} + s_o^t Y_{i(o)l(o)} \right) - \sum_{l=1}^{l(o_c^l)} c_l$.

À chaque itération, le coût du plan de production proposé par l'heuristique Lagrangienne est calculé avec (3.26). Puisque des contraintes sont relâchées, ce coût représente une *borne inférieure* $LB(y)$ du problème primal. À la première itération, comme $\beta_c = 0 \ \forall \ c$, le plan de production proposé correspond à la solution optimale du problème de dimensionnement de lots sans contraintes de capacité, et le coût associé correspond à la *borne inférieure absolue* (ALB). La plus grande borne inférieure $LB^*(y)$, trouvée par l'heuristique, est la *meilleure borne inférieure* pour la séquence y. À n'importe quelle itération de l'heuristique, si nous fixons $\beta_c = 0 \ \forall \ c$, le coût du plan de production est nommé *borne inférieure sans coûts Lagrangiens*, et est noté $LBWLC(y)$.

3.3.1.2 Heuristique de lissage

Comme la relaxation Lagrangienne ne conduit en général pas à des solutions réalisables, une heuristique de lissage de la production est incorporée, avec pour but de valider les contraintes de capacité violées. Cette procédure, qui modifie les tailles de lots X_{il} et les variables de setup Y_{il}, tout au long de l'horizon de planification, suit les étapes suivantes :

1. Sélectionner le job critique (produit critique i et période source l_s),
2. Sélectionner la période destination l_d et la quantité Q_{i,l_s,l_d} du produit i à transférer,
3. Faire le déplacement.

Le job critique est l'ordre de fabrication (i, l_s), dont le produit critique i à la période source l_s contient la plus grande durée opératoire liée à des opérations sans marge, i.e. sans possibilité d'augmenter les tailles de lots car les dates de début au plus tôt et au plus tard de l'opération sont égales ($t_e(o) = t_l(o)$).

Les mouvements peuvent être réalisés d'avant en arrière ($l_s > l_d$) ou d'arrière en avant ($l_s < l_d$), le choix étant basé sur le moindre coût unitaire parmi tous les mouvements du voisinage, i.e. les différentes périodes destination possibles. Une période est considérée comme une possible période destination si elle ne contient pas d'opérations sans marge. Le coût unitaire C_u est calculé à travers l'équation suivante, où C_{avant} et C_{apres} correspondent aux coûts du plan de production avant et après le transfert, respectivement.

$$C_u = \frac{C_{apres} - C_{avant}}{Q_{i,l_s,l_d}} \tag{3.27}$$

La quantité de transfert Q_{i,l_s,l_d} est déterminée de la manière suivante :

– Si $l_s < l_d$:

1. Nous calculons la quantité maximale possible à déplacer Q_1, afin de respecter les contraintes d'équilibre de stocks

$$Q_1 = \min_{l_s+1 \leq l < l_d} (I_{i,l_s-1} + X_{i,l_s} - D_{i,l_s}, I_{il}) \tag{3.28}$$

2. Nous calculons la quantité de transfert Q_{i,l_s,l_d}, respectant la condition précédente, la quantité de production disponible X_{il} et la marge de capacité $marge$, cette dernière étant définie par (3.30).

$$Q_{i,l_s,l_d} = \min (Q_1, X_{i,l_s}, marge) \tag{3.29}$$

$$marge = \min_{o \in (i,l_d)} m_o \tag{3.30}$$

Afin de pouvoir déterminer la marge de capacité, une marge cible m_o est calculée pour chaque opération o du produit critique i à la période destination l_d avec (3.31) si la production du produit i est déjà prévue à la période l_d ($Y_{il} = 1$), ou avec (3.32) sinon.

$$m_o = \frac{t_l(o) - t_e(o)}{sommeDuree_o + sommeSetup_o} \tag{3.31}$$

$$m_o = \frac{t_l(o) - t_e(o) - sommeSetup_o}{sommeDuree_o + sommeSetup_o} \tag{3.32}$$

La somme des durées unitaires opératoires $sommeDuree_o$ et la somme des temps de setup $sommeSetup_o$ sont calculées avec (3.33) et (3.34), respectivement, où $o^f(i, l_d)$ est la première opération dans la gamme de fabrication du job (i, l_d).

$$sommeDuree_{o(i,l_d)} = \sum_{a=o^f(i,l_d)}^{o(i,l_d)} p_a^u \quad \forall\ o \in (i, l_d) \tag{3.33}$$

$$sommeSetup_{o(i,l_d)} = \sum_{a=o^f(i,l_d)}^{o(i,l_d)} s_a^t \quad \forall\ o \in (i, l_d) \tag{3.34}$$

– Si $l_d < l_s$, les mouvements ne risquent pas de violer les contraintes d'équilibre de stocks, et la quantité de transfert Q_{i,l_s,l_d} est calculée comme suit :

$$Q_{i,l_s,l_d} = \min\left(X_{i,l_s}, marge\right) \tag{3.35}$$

Les trois étapes de l'heuristique de lissage sont répétées jusqu'à ce que le plan de production soit réalisable, ou jusqu'à ce qu'il ne soit plus possible de faire de mouvements, comme dans [235]. Par ailleurs, quand un plan de production réalisable est obtenu, une nouvelle sous-procédure, visant à améliorer la qualité de la solution, est démarrée. Toutes les quantités de production de tous les produits qui peuvent être transférées avec une diminution du coût sont évaluées. Les meilleurs mouvements sont systématiquement effectués, et la procédure s'arrête quand il n'est plus possible de diminuer le coût du plan, comme illustré dans la Figure 3.4.

Dans notre approche, contrairement aux heuristiques de lissage trouvées dans la littérature où les changements sur les tailles de lots dans une période n'ont pas d'impact sur la capacité des autres périodes, les changements sur une période ont une influence sur la capacité de toutes les périodes. Ceci vient du fait que nous résolvons le problème d'ordonnancement sur tout l'horizon de planification, et non période par période, comme c'est le cas dans la littérature. La capacité consommée et les marges à chaque période varient dynamiquement en fonction des mouvements de l'heuristique de lissage, et il par conséquent difficile d'estimer à l'avance les mouvements nécessaires pour respecter les contraintes de capacité. L'avantage est que la

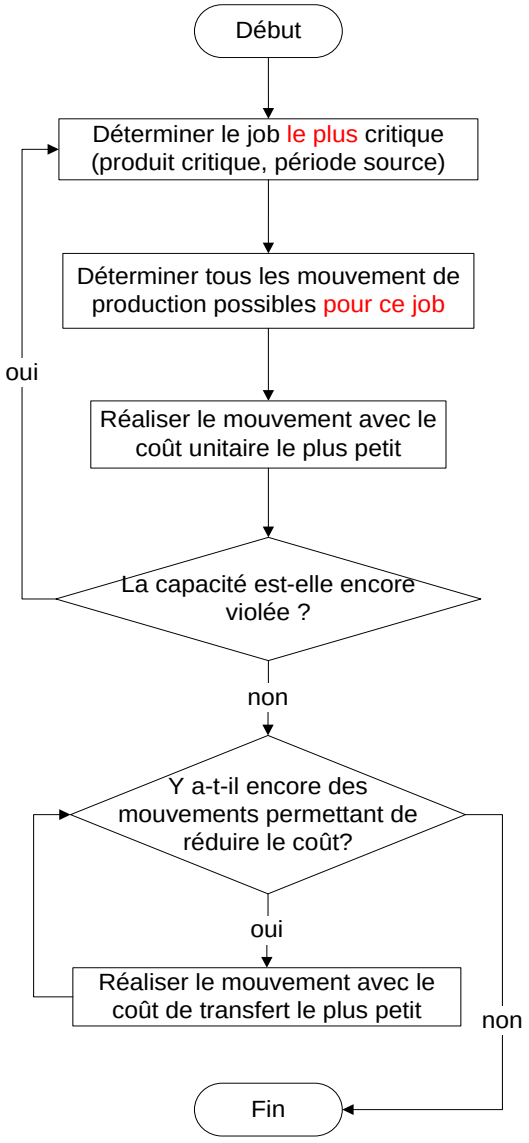

FIGURE 3.4 – Heuristique de lissage de la production

capacité est considérée de manière exacte, permettant une distribution plus efficace des tâches de production.

Le coût du plan final, obtenu à partir de l'heuristique de lissage, correspond à une borne supérieure (UB) du problème intégré de dimensionnement de lots et d'ordonnancement.

3.3.2 Méthode d'amélioration de la séquence

L'heuristique Lagrangienne permet d'obtenir le meilleur plan de production réalisable pour une séquence fixée. Néanmoins, en tenant compte du fait que seulement une instance du graphe disjonctif est considérée, la solution correspond à un optimum local. Pour obtenir ou au moins se rapprocher d'un optimum global, et dans certains cas arriver à trouver une solution réalisable, il est indispensable de modifier la séquence des opérations. C'est pourquoi nous appliquons une méthode d'amélioration de la séquence à l'aide d'une métaheuristique de type recherche taboue [82].

Pour générer une nouvelle séquence, nous changeons l'orientation d'un arc liant deux opérations partageant la même ressource, en prenant comme base le graphe conjonctif courant. Le problème est que le nombre de séquences possibles est exponentiel, et évaluer tous les changements possibles est très coûteux. C'est pourquoi il est très important d'avoir une méthode efficace, permettant de guider la recherche et de changer uniquement l'orientation des arcs les plus prometteurs. Nous appliquons une recherche taboue en utilisant l'information provenant de l'heuristique Lagrangienne. La procédure est résumée sur la Figure 3.5.

3.3.2.1 Principe

Comme il n'est pas possible d'explorer toutes les séquences, nous devons d'abord définir l'espace de recherche de l'heuristique. Nous décidons de restreindre la recherche aux arcs qui ont été violés durant la relaxation Lagrangienne associée à la séquence y, et nous dénotons cet ensemble par $\mathcal{VA}(y)$. Un arc $(o, o') \in \mathcal{S}(y)$ est considéré comme violé s'il est sur un chemin violé, i.e. (o, o') est inclus dans $\mathcal{VA}(y)$ si $(o, o') \in c \in \mathcal{CH}(y)$, où $\mathcal{CH}(y)$ correspond à l'ensemble des chemins les plus violés durant la relaxation Lagrangienne.

Changer l'orientation des arcs appartenant à $\mathcal{VA}(y)$ peut aider à améliorer l'utilisation de la capacité, en diminuant le *makespan*. Cependant, il est difficile d'estimer à l'avance les arcs qui peuvent contribuer à améliorer la qualité de la solution. Par ailleurs, tous les arcs violés ne contribuent pas à la violation de la capacité, et leur impact dépend de la configuration du graphe conjonctif courant. Donc, pour s'adapter à l'évolution du graphe, il est très important de pouvoir changer l'orientation des arcs qui avaient été inversés auparavant. Néanmoins, il n'est pas approprié de les inverser immédiatement après un « mauvais » changement, parce que ceci empêcherait le graphe d'évoluer vers de nouvelles solutions, et la solution finale pourrait converger rapidement vers un optimum local. Il faut ainsi laisser le graphe conjonctif évoluer durant quelques itérations de la recherche taboue, en acceptant des changements d'arcs qui n'améliorent pas forcement la borne supérieure. Cette stratégie peut conduire à l'obtention

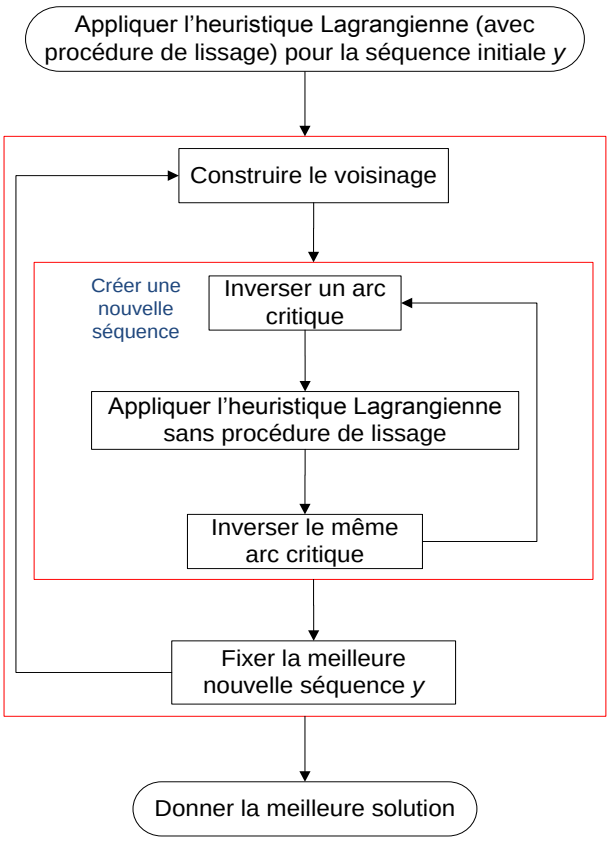

FIGURE 3.5 – Méthode d'amélioration de la séquence

d'une bonne solution, qui ne serait pas atteignable autrement. Dans la recherche taboue, la profondeur de l'exploration de l'espace de recherche est gérée par la taille de la liste taboue. Si un arc $(o, o') \in \mathcal{VA}(y)$ est changé, il est inclus dans la liste taboue, noté $\mathcal{TL}$, et est enlevé de $\mathcal{VA}(y)$. Ainsi, selon la profondeur d'exploration désirée, la taille maximale de la liste taboue ($size\_max$) est fixée. Une fois la liste taboue complètement remplie ($|\mathcal{TL}| > size\_max$), l'arc occupant la première position de la liste est enlevé. Définir la taille de la liste taboue est un problème de compromis entre qualité de la solution (profondeur d'exploration) et temps de calcul. Dans l'approche proposée par Wolosewicz *et al.* [233, 234, 235], la taille de la liste taboue était infinie, ce qui ne permettait pas une exploration appropriée de l'espace de recherche. Dans notre approche, la liste taboue est finie. Nous avons réalisé différents tests afin de déterminer la taille la plus appropriée.

3.3.2.2 Voisinage

Comme $|\mathcal{VA}(y)|$ peut être très grand, évaluer tous les arcs violés devient rapidement très couteux en temps de calcul. Pour augmenter la vitesse de la procédure, nous choisissons de construire un voisinage qui est l'ensemble des arcs critiques $\mathcal{CA}(y)$, avec $\mathcal{CA}(y) \subset \mathcal{VA}(y)$, et nous évaluons non pas l'ensemble des arcs violés, mais l'ensemble des arcs critiques. L'idée est de mesurer l'impact de changer l'orientation de chacun des arcs critiques, pour finalement inverser l'arc qui satisfait le mieux le critère choisi.

Pour sélectionner les arcs critiques, nous définissons tout d'abord un poids $w_{(o,o')}\ \forall\ (o,o') \in \mathcal{VA}(y)$, et nous sélectionnons ensuite, tous les arcs $(o,o') \in \mathcal{VA}(y)$, pour lesquels $w_{(o,o')} \geq \alpha w_{max}$, avec $\alpha \in \mathbb{R}$, $0 \leq \alpha \leq 1$ étant un facteur de réduction et $w_{max} = \max_{(o,o')\in\mathcal{VA}(y)} w_{(o,o')}$ étant le poids maximal parmi tous les arcs violés. Ainsi, α est fixé selon la taille d'exploration désirée et les limitations en temps de calcul. Tous les arcs respectant cette condition sont des arcs critiques et font donc partie de $\mathcal{CA}(y)$.

De nombreuses expérimentations ont été réalisées pour déterminer la valeur appropriée de α. Quand α tend vers 0, une très bonne séquence peut systématiquement être choisie, mais le temps de calcul augmente beaucoup, et seulement quelques arcs peuvent être changés en un temps raisonnable. À l'inverse, quand α tend vers 1, des très mauvaises séquences peuvent parfois être choisies, mais comme le voisinage n'est pas très grand, l'exploration peut être rapidement complétée. Cela permet d'appliquer la recherche taboue pendant beaucoup plus d'itérations, ce qui conduit généralement à l'obtention d'une meilleure solution.

De manière similaire, deux méthodes d'affectation du poids des arcs ont été étudiés. Dans la première méthode, $w_{(o,o')} = \sum_{c\in\mathcal{CH}(y);(o,o')\in c} \beta_c\ \forall\ (o,o') \in \mathcal{VA}(y)$, c'est-à-dire que nous considérons des arcs avec une grande somme de multiplicateurs Lagrangiens. L'idée est que ces arcs sont potentiellement les plus violés, ou au moins ils appartiennent aux chemins les plus violés. Donc, changer l'orientation d'un de ces arcs est probablement judicieux si on veut réduire la violation de la capacité. La deuxième méthode analysée consiste à calculer le poids d'un arc comme le nombre de chemins violés qui contiennent l'arc. De cette façon, changer l'orientation de l'arc avec le plus grand poids a un impact sur un grand nombre de chemins, ce qui peut contribuer à réduire la violation du graphe de façon importante, en réalisant un seul mouvement. Les deux méthodes ont été comparées à travers plusieurs expérimentations, et même si aucune des deux méthodes n'arrive à dominer l'autre sur toutes les instances étudiées, nous avons décidé d'appliquer la deuxième méthode, car elle conduit à des meilleurs résultats dans la plupart des instances. Plus de détails sont données dans la Section 3.4.

3.3.2.3 Sélection de la nouvelle séquence

Une fois que le voisinage est créé, nous devons choisir un critère pour sélectionner l'arc qui va être finalement inversé. Les trois critères que nous avons analysés sont basés sur : $LB^*(y)$, $LBWLC(y)$ et $UB^*(y)$. L'arc générant la meilleure valeur pour le critère étudié est celui qui est changé.

L'avantage d'utiliser $UB^*(y)$ comme critère de décision est qu'elle permet d'obtenir des meilleures solutions à court terme. Néanmoins, l'inconvénient est que l'heuristique de lissage consomme beaucoup de temps, surtout quand le nombre de périodes est important. Le nombre de séquences changées risque ainsi d'être petit, ne permettant pas d'atteindre une solution de bonne qualité.

D'autre part, utiliser $LB^*(y)$ ou $LBWLC(y)$ permet d'éviter de mettre en œuvre l'heuristique de lissage, et ainsi d'évaluer plus de changements de séquences dans l'exploration globale de l'approche intégrée. Les solutions sont parfois de mauvaise qualité dans l'exploration, mais, dans les mêmes temps de calcul, les solutions obtenues sont meilleures que celles obtenues avec le critère basé sur la borne supérieure.

3.4 Analyses et améliorations

Nous avons mené plusieurs expérimentations pour identifier des parties à améliorer dans l'approche de Wolosewicz *et al.* Une fois les points faibles de la procédure identifiés, nous avons testé plusieurs modifications visant à améliorer les résultats, en termes de qualité de la solution et temps de calcul. Quelques idées ont été retenues dans notre approche ; tandis que les autres ont permis de mieux maîtriser le comportement de la procédure, de connaître les limitations de l'approche et d'avoir une vision sur des possibles améliorations à expérimenter.

Dans ce qui suit, nous détaillons les idées testées sur le problème de planification à séquence fixée et sur la partie amélioration de la séquence.

3.4.1 Résolution du problème à séquence fixée

L'heuristique Lagrangienne permettant de résoudre le problème à séquence fixée comporte trois étapes fondamentales : la résolution du problème mono-produit avec contraintes de capacité relâchées, la construction d'une solution réalisable et la mise à jour des multiplicateurs. Nous avons identifié des points faibles au niveau des deux premières étapes.

En ce qui concerne la résolution du problème mono-produit avec contraintes de capacité relâchées, nous n'avons pas trouvé de faiblesses par rapport à l'application de l'algorithme de programmation dynamique de dimensionnement de lots, mais par rapport au réglage des paramètres de la relaxation Lagrangienne. Ces paramètres ayant une influence directe sur l'évolution des multiplicateurs Lagrangiens et sur la rapidité de l'exploration, leur réglage initial a une forte incidence sur les bornes inférieures trouvées par la procédure.

Au niveau de l'heuristique de lissage, nous avons constaté que, dans certains cas, l'écart entre la solution optimale pour une séquence donnée et la borne supérieure pouvait être important, surtout pour des instances avec coûts de setup élevés par rapport aux coûts de stockage. Nous avons donc étudié plus en détail ces instances. En effet, lorsque les coûts de setup sont importants, la différence entre un plan de production avec k ordres de fabrication et un autre avec $k+2$ ordres peut être très significative en termes de coûts. Nous avons analysé des alternatives pour améliorer la qualité des bornes supérieures, que nous décrivons dans la sous-section 3.4.1.2.

Une partie très importante de la procédure, où nous avons détecté une grande possibilité d'amélioration, est le lien entre l'algorithme de dimensionnement de lots avec contraintes relâchées et l'heuristique de lissage (autrement dit, entre LB et UB).

3.4.1.1 Paramètres de la relaxation Lagrangienne

Les paramètres de la relaxation Lagrangienne sont :

- Le nombre maximum d'itérations de l'heuristique,
- Le nombre maximum d'itérations consécutives sans amélioration,
- Le pas de déplacement initial,
- Le pas de déplacement minimum,
- Le facteur de réduction du pas.

Nous avons testé différents jeux de paramètres sur plusieurs instances, mais les résultats ne permettent pas de trouver les paramètres qui donnent la meilleure solution pour n'importe quel problème. Néanmoins, nous pouvons conclure que le pas de déplacement initial et le pas déplacement minimum peuvent être fixés à la même valeur pour tous les problèmes sans trop perdre sur la qualité de la solution. D'autre part, concernant le nombre maximum d'itérations de la relaxation Lagrangienne et le nombre maximum d'itérations sans amélioration, on doit garder des valeurs qui génèrent globalement une solution de bonne qualité, car il n'y a pas une combinaison parfaite pour tous les problèmes.

Le paramètre le plus important est le nombre total d'itérations de l'heuristique Lagrangienne. Un grand nombre d'itérations permet d'obtenir une très bonne borne inférieure pour le problème de dimensionnement de lots à séquence fixée. En outre, l'heuristique de lissage de la production est appliquée plusieurs fois, donc plus de bornes supérieures sont générées et la probabilité d'obtenir un bon plan de production réalisable augmente. Cependant, plus le nombre d'itérations de l'heuristique Lagrangienne est important, plus le temps de calcul passé pour résoudre le problème de dimensionnement de lots à séquence fixée augmente. Étant donné que le temps total d'exécution est limité, moins de séquences sont explorées, c'est-à-dire que le nombre de solutions aux problèmes de dimensionnement de lots à séquence fixée est réduit.

Les résultats relatifs à la comparaison de plusieurs nombres maximums d'itérations peuvent être observés sur la Figure 3.6. L'écart sur la qualité EQ, calculé avec (3.36), permet de comparer les solutions obtenues avec les différentes valeurs du paramètre. Plus EQ est petit, plus le plan déterminé se rapproche de la meilleure solution parmi les autres alternatives. Dans cette expression, UB_i est la valeur de la meilleure borne supérieure obtenue en utilisant un nombre maximum d'itérations de i.

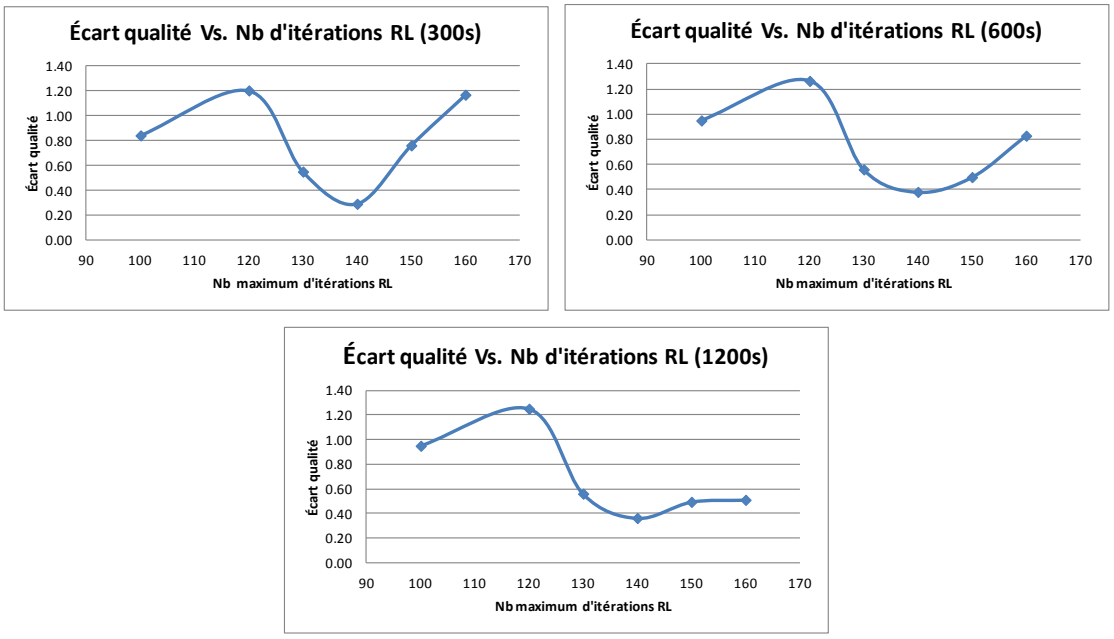

FIGURE 3.6 – Influence du nombre maximum d'itérations de la relaxation Lagrangienne sur le résultat final

$$EQ = \left(\frac{UB_i}{\min_i \{UB_i\}} - 1 \right) \times 100 \tag{3.36}$$

Selon le temps de calcul, l'influence du nombre maximum d'itérations peut être plus ou moins importante. Ainsi, pour un temps de calcul relativement petit, comme 300 secondes, un grand nombre d'itérations (supérieur à 140) peut conduire vers des mauvaises solutions. La raison est que plus de temps que nécessaire est utilisé pour l'heuristique Lagrangienne, laissant moins de temps pour la procédure d'amélioration de la séquence qui s'avère plus importante. D'autre part, avec un temps de calcul 4 fois plus important (1200 secondes), un grand nombre d'itérations a une influence très peu significative sur les résultats, car il y a plus de temps dédié à l'amélioration de la séquence. Globalement, le meilleur nombre maximum d'itérations est égal à 140.

Il est important de remarquer qu'un grand nombre maximum d'itérations peut être un bon choix quand la séquence initiale est proche d'une séquence optimale, i.e. quand peu de changements d'arcs sont nécessaires pour déterminer la solution optimale, ou quand la taille du problème est petite. D'autre part, si le nombre d'itérations est petit, la borne inférieure peut ne pas converger et le nombre de fois où l'heuristique de lissage est appliquée peut ne pas suffire à obtenir une bonne borne supérieure. De plus, des bornes inférieures et supérieures de mauvaise qualité peuvent conduire à des changements d'arcs inappropriés dans la procédure d'amélioration de la séquence. Le problème est de trouver un compromis entre intensification (nombre d'itérations de l'heuristique Lagrangienne) et diversification (nombre d'itérations de la méthode d'amélioration de la séquence), avec l'objectif de trouver le meilleur plan de production réalisable possible en un temps de calcul limité.

3.4.1.2 Heuristique de lissage de la production

L'heuristique de lissage utilisée dans l'approche de Wolosewicz *et al.* a une bonne performance. Elle est appropriée en termes de temps de calcul, car elle évalue les mouvements de production possibles associés à un seul job (le plus critique). Cependant, il n'y a pas de garantie que la borne supérieure obtenue soit optimale, et parfois il y a des écarts significatifs par rapport aux solutions optimales. Pour améliorer l'heuristique de lissage, nous avons analysé les alternatives suivantes :

– Ajouter une **étape d'amélioration**, une fois qu'une solution réalisable est déterminée. Dans [235], l'heuristique originale s'arrête lorsqu'un plan de production réalisable est

déterminé. La nouvelle alternative consiste à chercher de nouveaux mouvements de production réduisant la valeur de la borne supérieure une fois que la solution est réalisable.

– Appliquer une **heuristique de lissage gloutonne**. Dans [235], un seul job critique est considéré à chaque itération, et toutes les périodes destination possibles et ses quantités de transfert associées sont donc évaluées. La nouvelle idée consiste à considérer tous les jobs critiques à chaque itération et à évaluer tous les mouvements possibles pour chacun d'entre eux. À chaque itération, le mouvement avec le coût unitaire de transfert le plus petit est réalisé. Une version simplifiée de l'algorithme est représentée sur la Figure 3.7.

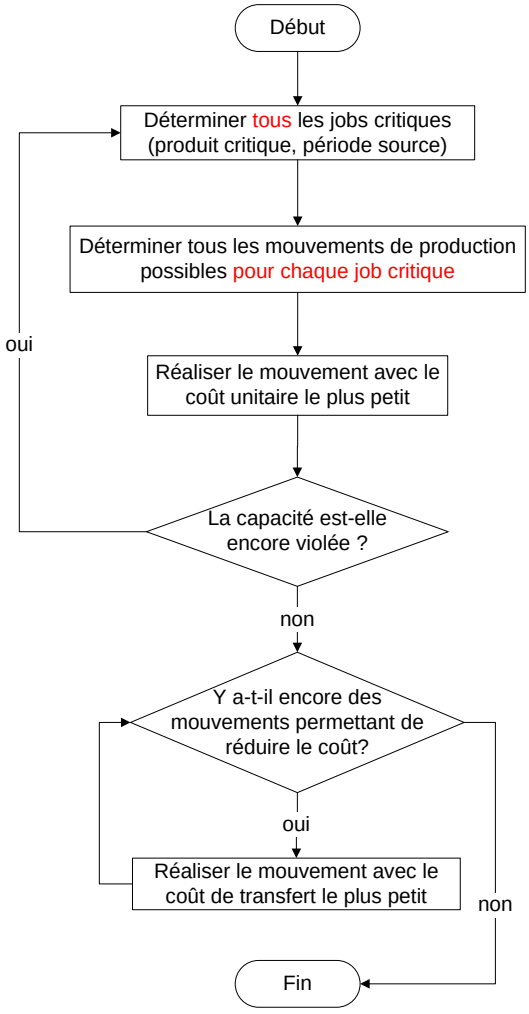

FIGURE 3.7 – Heuristique de lissage gloutonne

Ces deux alternatives supposent une augmentation du temps de calcul de l'heuristique Lagrangienne, réduisant le temps passé dans l'étape de modification de la séquence. Lorsque l'on

résout le problème de dimensionnement de lots à séquence fixée, la première alternative permet d'obtenir au minimum la même qualité de borne supérieure que l'heuristique originale, mais il n'y a pas de dominance lorsque l'on résout le problème intégré. En fait, l'heuristique originale permet de changer plus de séquences, et dans certains cas la solution finale peut être meilleure que celle avec la procédure améliorée. D'autre part, l'heuristique gloutonne peut fournir de très bonnes solutions pour quelques instances, mais elle peut aussi générer des solutions de mauvaise qualité, surtout parce que le nombre de séquences explorées est largement réduit. Sur 153 instances testées, variant différents paramètres pour différentes tailles des problèmes, la procédure originale trouve des meilleures solutions que la procédure gloutonne dans 70.97% des instances. De même, elle permet de changer plus de séquences dans 57.81% des instances.

3.4.1.3 Lien entre la relaxation Lagrangienne et la procédure de lissage

Dans l'approche de Wolosewicz *et al.*, la relaxation Lagrangienne est appliquée deux fois pour chaque itération de la recherche taboue. La première a pour objectif d'obtenir une diminution de la meilleure borne inférieure $LB(y)$, et la deuxième est intégrée à l'heuristique de lissage de la production, pour calculer une borne supérieure. La deuxième exécution est déclenchée seulement si $LB(y) < LB^*(y)$, c'est-à-dire, si la borne inférieure est améliorée. Cette procédure est illustrée sur la Figure 3.8.

D'après plusieurs expérimentations réalisées, nous avons constaté que le fait d'obtenir une bonne borne supérieure n'est pas lié à la valeur de la borne inférieure. Autrement dit, une diminution de LB n'entraîne pas une diminution de UB. Donc, dans la procédure de Wolosewicz *et al.*, un ensemble important de bonnes solutions est négligé. Cette stratégie occasionne également des pertes importantes en temps de calcul, car la deuxième relaxation Lagrangienne donne exactement les mêmes résultats que la première. En effet, la séquence est la même et les multiplicateurs Lagrangiens sont réinitialisés à zéro.

Pour cette raison, dans notre approche, une fois qu'une séquence est fixée et pour déterminer un plan de production réalisable, la relaxation Lagrangienne et la procédure de lissage sont toujours combinées. Rappelons que dans la procédure d'amélioration de la séquence, la relaxation Lagrangienne est appliquée sans tenir compte de la procédure de lissage, le but n'étant pas de déterminer une solution réalisable mais de modifier la séquence.

Même si elle est importante, l'heuristique de lissage consomme beaucoup de temps de calcul, et il est par conséquent préférable de ne pas l'appliquer à chaque itération. Le temps de calcul est plus grand quand la capacité est faible, car beaucoup de chemins dans le graphe conjonctif sont violés, et il est nécessaire de modifier les dates de plusieurs nœuds, en réalisant de nom-

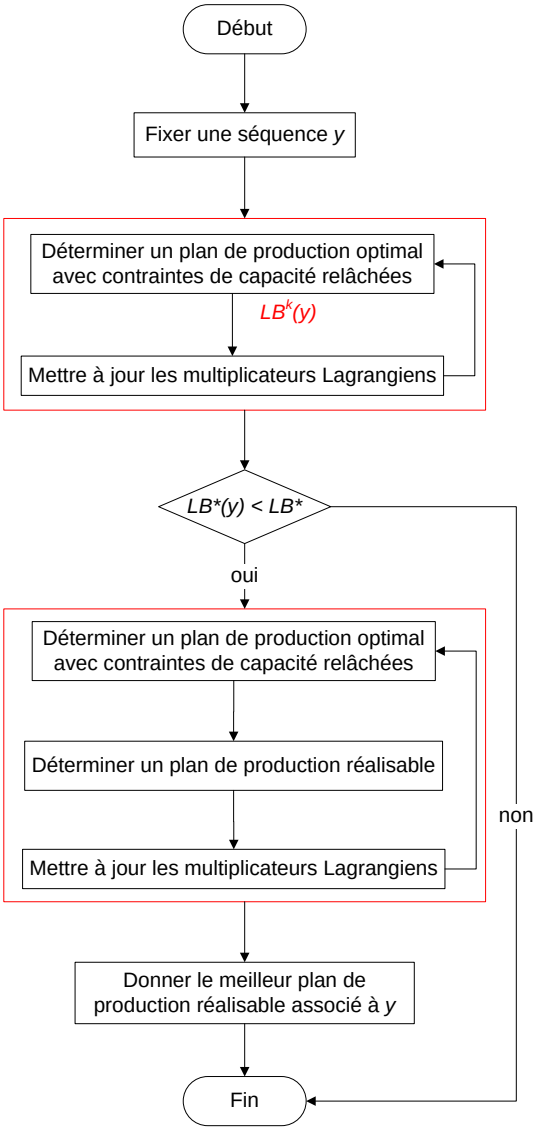

FIGURE 3.8 – Heuristique Lagrangienne de Wolosewicz *et al.* [235]

breux transferts de production pour obtenir une solution réalisable. Appliquer plusieurs fois l'heuristique de lissage augmente le temps de calcul de l'heuristique Lagrangienne et réduit le temps disponible pour explorer plus de séquences dans la recherche taboue. Une fois de plus, il s'agit de trouver un compromis entre intensification et diversification. L'objectif est de pouvoir décider quand appliquer l'heuristique de lissage, afin d'obtenir la meilleure borne supérieure possible avec le plus petit temps de calcul possible.

Quand la procédure de lissage est systématiquement appliquée, la meilleure borne supérieure pour chaque séquence est garantie, mais le temps de calcul est élevé. Dans plusieurs cas, le plan

de production proposé par l'algorithme de programmation dynamique de dimensionnement de lots est identique à celui obtenu à une itération précédente, et la borne supérieure associée est par conséquent aussi identique. Éviter de faire plusieurs fois le même calcul permet de réduire le temps de calcul de l'heuristique Lagrangienne.

Dans l'approche de Wolosewicz *et al.*, l'heuristique de lissage est appliquée toutes les 5 itérations, ce qui permet de réduire de manière significative le temps de calcul sur une séquence, mais de bonnes solutions peuvent être manquées par l'algorithme. Pour évaluer les pertes en qualité de solution, nous avons comparé cette alternative avec une stratégie d'application systématique (à chaque itération) de l'heuristique de lissage. Nous avons effectivement constaté que, pour chaque séquence fixée, il y a des écarts significatifs entre les bornes supérieures des deux stratégies. En effet, les solutions déterminées avec un lissage de la production toutes les 5 itérations sont d'une qualité très inférieure pour le problème à séquence fixée. Cependant, étant donné qu'une application systématique de l'heuristique de lissage réduit le temps pour réaliser les itérations de la recherche taboue, la borne supérieure finale (solution du problème intégré), obtenue avec la stratégie de lissage toutes les 5 itérations, est meilleure pour la plupart des instances testées.

Nous avons étudié une deuxième alternative, qui consiste à appliquer l'heuristique de lissage seulement quand $LBWLC^k(y) < UB^*(y)$ à l'itération k. Nous utilisons $LBWLC$ parce que sa valeur est généralement inférieure à UB, i.e. construire une solution réalisable entraîne une augmentation du coût du plan de production. Ceci est dû au fait que $LBWLC$ est déterminé avec des contraintes de capacité relâchées, et donc il y a moins de limitations sur la minimisation du coût. Cette stratégie permet de réduire le temps passé sur l'heuristique Lagrangienne, par rapport au lissage à chaque itération, et la meilleure borne supérieure est obtenue dans tous les cas. De plus, du temps additionnel peut être utilisé pour réaliser plus d'itérations de la recherche taboue, ce qui conduit vers de meilleurs résultats. Néanmoins, la stratégie du lissage toutes les 5 itérations donne des meilleurs résultats, car plus de séquences sont explorées.

Pour augmenter la diversification, nous avons conçu une nouvelle stratégie qui consiste à appliquer l'heuristique de lissage uniquement quand $LBWLC(y) < UB^*(y)$ et $LBWLC(y) \notin \mathcal{B}$, où $\mathcal{B}$ est l'ensemble des 5 dernières valeurs de $LBWLC(y)$. Ce choix est basé sur l'évolution de $LBWLC$. En fait, le plan de production proposé par l'algorithme de programmation dynamique de dimensionnement de lots est souvent le même à différentes itérations. Dans notre méthode de résolution, il est normal d'obtenir un plan qui a été précédemment calculé, dans une fenêtre habituellement (mais pas toujours) inférieure à 5 itérations. Les résultats obtenus avec cette stratégie ont globalement amélioré les solutions des autres alternatives, sauf pour le cas avec construction systématique d'une solution réalisable toutes les 5 itérations. La qualité moyenne

des solutions est, en fait, quasiment la même. Cependant, la nouvelle alternative, que nous appelons *5-LBWLC*, utilise un critère plus cohérent et permet de résoudre plus efficacement le problème à séquence fixée. La nouvelle heuristique Lagrangienne est illustrée sur la Figure 3.9.

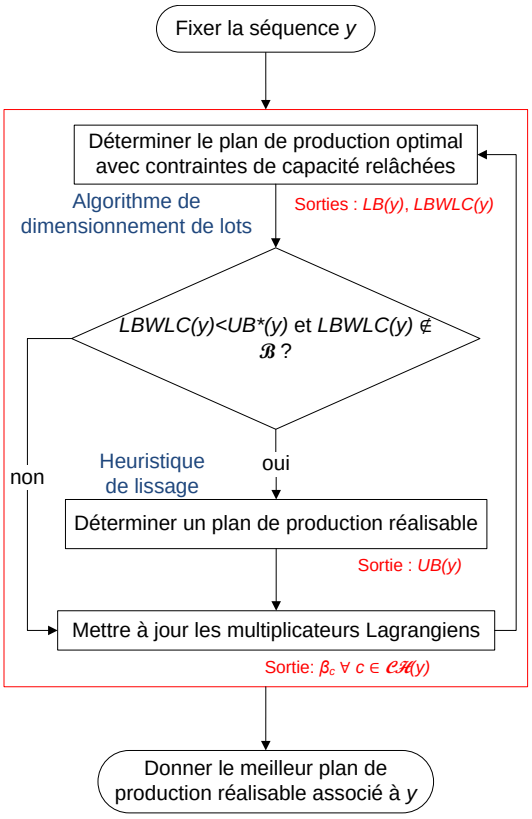

FIGURE 3.9 – Heuristique Lagrangienne modifiée

3.4.2 Amélioration de la séquence

Dans notre approche, la méthode d'amélioration de la séquence est composée principalement de trois étapes : la construction du voisinage, l'exploration du voisinage et la sélection de l'arc à changer. Ces trois parties de l'algorithme diffèrent de la procédure dans l'approche de Wolosewicz *et al.*, schématisée dans la Figure 3.10. Premièrement, dans l'approche de Wolosewicz *et al.*, le voisinage est composé par tous les arcs violés ; tandis que nous définissons un ensemble d'arc critiques pour réduire le nombre de solutions candidates. Deuxièmement, dans l'approche de Wolosewicz *et al.*, il n'y a pas d'exploration de solutions potentielles. Le choix de l'arc à changer est basé sur la plus grande somme de multiplicateurs Lagrangiens (une somme était calculée pour chaque arc). Dans notre approche, illustrée sur la Figure 3.5, nous évaluons

chaque candidat potentiel à travers la relaxation Lagrangienne (sans construction d'une solution réalisable), et nous sélectionnons l'arc avec la plus petite valeur de $LB^*(y)$. Un autre facteur différenciateur est la taille de la liste taboue. Dans l'approche de Wolosewicz *et al.*, la taille de la liste taboue est infinie. Nous avons fixé une taille finie, à partir de différentes expérimentations.

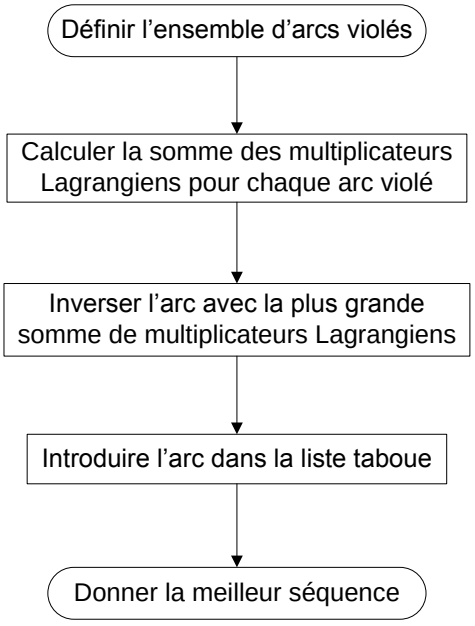

FIGURE 3.10 – Méthode d'amélioration de la séquence dans l'approche de Wolosewicz *et al.*

3.4.2.1 Construction du voisinage

La construction du voisinage des arcs potentiels à inverser comprend deux étapes : l'affectation d'un poids à chaque arc violé et la limitation de l'espace de recherche. Nous avons comparé plusieurs alternatives que nous discutons dans les paragraphes qui suivent.

Affectation du poids aux arcs

Nous avons étudié plusieurs méthodes d'affectation de multiplicateurs Lagrangiens aux arcs faisant partie du voisinage cible. En fait, chaque chemin qui a été violé au moins une fois ($c \in \mathcal{CH}(y)$) a un multiplicateur Lagrangien β_c associé qui vaut zéro quand sa contrainte de capacité est satisfaite, ou qui est supérieur à zéro quand cette dernière est violée. Chacun de ces multiplicateurs est affecté à chaque arc $(o, o') \in c \in \mathcal{CH}(y)$ selon la méthode d'affectation choisie.

– Méthode β_{max}

Cette méthode, qui est utilisée dans l'approche de Wolosewicz *et al.*, consiste à ajouter à chaque arc violé la valeur du multiplicateur Lagrangien de chaque chemin qui le contient. Ainsi, pour chaque arc violé ($(o, o') \in \mathcal{VA}(y)$), un poids est calculé avec (3.37).

$$w(o, o')_y = \sum_{c \supset (o,o') \in \mathcal{CH}(y)} \beta_c \quad \forall \quad (o, o') \in \mathcal{CH}(y) \tag{3.37}$$

– Méthode $\beta + 1$

Cette méthode consiste à affecter le multiplicateur Lagrangien du premier chemin contenant l'arc, et ensuite à ajouter le nombre de fois que l'arc violé apparait dans le parcours d'autres chemins. Le poids est calculé avec (3.38).

$$w(o, o')_y = \beta_{c^1} + \sum_{c \supset (o,o') \in \mathcal{CH}(y); c \neq c^1} 1 \quad \forall \quad (o, o') \in \mathcal{CH}(y) \tag{3.38}$$

De cette façon, les arcs qui appartiennent au plus grand nombre de chemins sont privilégiés. D'autres variantes ont été analysées, mais sans impact significatif sur les résultats. Aucune méthode ne domine complètement les autres, mais $\beta+1$ conduit globalement vers des meilleurs résultats. C'est pourquoi, cette méthode est utilisée dans notre approche.

En comparant les deux méthodes sur différentes instances d'un problème de grande taille (job-shop avec 20 jobs, 5 opérations par job et 5 machines), l'écart absolu moyen entre les bornes supérieures obtenues avec les deux méthodes est égal à 12.73 et l'écart relatif moyen est égal à 0.02% en faveur de $\beta + 1$. Le nombre moyen de séquences changées est également plus favorable pour $\beta + 1$, ce chiffre augmentant de 7.76%, par rapport à β_{max}. Ce qui est vraiment important, c'est que la combinaison de cette méthode d'affectation de poids avec une nouvelle structure de voisinage conduit à des améliorations très significatives par rapport à l'approche de Wolosewicz *et al.*

Une autre méthode d'affectation, basée sur l'arc du graphe conjonctif avec la plus longue durée parmi les arcs violés, a aussi été testée. Cependant, les résultats ne sont pas compétitifs. L'impact sur l'utilisation de la capacité est important, mais pas toujours dans un bon sens. La plupart du temps, la qualité globale de la borne supérieure est significativement détériorée par rapport aux deux autres méthodes.

Analyse des structures de voisinage

Dans l'approche de Wolosewicz *et al.*, un ensemble d'arc violés $\mathcal{VC}(y)$, contenant tous les arcs (o, o') qui appartiennent à l'ensemble de chemins les plus violés $\mathcal{CH}(y)$ durant la relaxation Lagrangienne, a été défini. Étant donné qu'un arc peut faire partie du parcours de plusieurs chemins, une somme de multiplicateurs Lagrangiens est calculée pour chaque arc $(o, o') \in \mathcal{VC}(y)$, ce qui permet de déterminer les arcs qui potentiellement pourraient avoir un impact significatif sur la violation de la capacité.

L'orientation de l'arc avec la somme de multiplicateurs Lagrangiens la plus grande est inversée et l'arc est introduit dans la liste taboue $\mathcal{TL}$. Puis, la nouvelle séquence est fixée, le nouveau problème de planification est résolu, et la procédure continue de façon itérative. Un inconvénient de cette procédure est l'absence d'un voisinage permettant d'explorer plusieurs alternatives (solutions candidates). L'avantage est la possibilité de fixer plus de nouvelles séquences, et donc de résoudre plus de problèmes de planification à séquence fixée. Cependant, la méthode converge rapidement vers un optimum local, sans considérer des branches importantes du graphe disjonctif. De plus, même si modifier l'orientation d'un grand nombre d'arcs réduit potentiellement le degré de violation de la capacité, les arcs choisis ne sont pas toujours les meilleurs. Pour cette raison, il est plus intéressant d'avoir un voisinage de mouvements candidats.

Nous avons étudié les types de voisinage suivants :

- **Voisinage complet :** tous les arcs appartenant à tous les chemins de $\mathcal{CH}(y)$ sont considérés. Donc, plus la capacité est faible, plus la taille du voisinage est importante, et par conséquent, l'exploration nécessite plus de temps de calcul. Néanmoins, à chaque itération de la recherche taboue, une meilleure séquence (ou au moins de la même qualité que la séquence précédente) est fixée.
- **Voisinage réduit :** Uniquement les arcs dont le poids est supérieur ou égal à un pourcentage de la valeur du plus grand poids sont considérés. Nous avons testés des facteurs de réduction de 25%, 50%, 75%, 95%, 97% et 100%. L'intérêt de cette structure de voisinage est de pouvoir déterminer l'impact du changement de différents arcs, avec un temps de calcul raisonnable.

Nous avons réalisé des expérimentations pour les instances les plus difficiles (avec coûts de setup importants par rapport aux coûts de stockage). Avec un voisinage complet, une bonne solution convergeant localement, peut être déterminée en un temps de calcul élevé. D'autre part, avec un voisinage réduit, des très bonne solutions sont obtenues en quelques secondes, et plus de branches du graphe disjonctif sont explorées, évitant la convergence locale prématurée. Pour comparer la performance des différents types de voisinage, nous avons défini un ratio

TABLEAU 3.2 – Comparaison de résultats avec différents voisinages

Instance	**0%**β_{max}		95%β_{max}		97%β_{max}		100%β_{max}		100%$\beta+1$	
	G_{UB} (%)	R_{QT_i}	G_{UB} (%)	R_{QT_i}	G_{UB} (%)	R_{QT_i}	G_{UB} (%)	R_{QT_i}	G_{UB} (%)	R_{QT_i}
Coût de setup 1	1,61	1,74	1,40	1,96	1,40	1,94	1,27	1,85	0,00	2,00
Coût de setup 2	1,08	1,77	0,33	1,98	0,15	1,95	0,00	1,99	0,06	1,99
Coût de setup 3	2,77	1,86	0,00	1,98	1,92	1,95	0,27	1,99	0,19	1,99
Coût de setup 4	2,80	1,87	0,00	1,95	0,00	1,96	2,53	1,94	0,35	1,96
Coût de setup 5	3,16	1,79	0,15	1,97	2,97	1,91	0,00	1,97	0,38	1,98
Coût de setup 6	6,21	1,67	2,79	1,90	0,00	1,99	2,87	1,91	2,93	1,92
Coût de setup 7	0,17	1,81	3,19	1,93	3,30	1,92	3,08	1,92	0,00	2,00
Moyenne	**2,54**	**1,79**	**1,12**	**1,95**	**1,39**	**1,94**	**1,43**	**1,94**	**0,56**	**1,98**

UB/temps de calcul. Le Tableau 3.2 présente une comparaison des résultats obtenus avec différents voisinages lorsque l'on résout des instances difficiles (avec grands coûts de setup). Le voisinage complet est appelé « 0% ». L'écart G_{UB} sur UB est calculé par rapport à la meilleure borne supérieure trouvée, et le ratio UB/temps (R_{QT_i}) est déterminé avec (3.39). Le temps d'exécution est fixé à 5 minutes.

$$R_{QT_i} = \frac{1 - \frac{UB_i - \min_i\{UB_i\}}{\min_i\{UB_i\}}}{t_i} \times 10 \tag{3.39}$$

Un grand écart sur UB représente une mauvaise solution, et une grande valeur de R_{QT_i} indique que la structure de voisinage proposée conduit à une bonne solution en un temps de calcul petit. La conclusion des expérimentations est que le voisinage avec facteur de réduction de 100% et avec $\beta + 1$ comme méthode d'affectation du poids donne des meilleures solutions que les autres alternatives.

Les résultats montrent qu'il est préférable d'utiliser un voisinage réduit qu'un voisinage complet. Le voisinage à 100% de la valeur du poids maximum s'avère être le meilleur, en prenant comme critère le compromis entre qualité de la borne supérieure et temps de calcul. Un constat important est que la méthode d'affectation du poids $\beta+1$ donne des résultats significativement meilleurs à ceux de β_{max} lorsque l'on adopte une structure de voisinage réduite et basée sur un critère de sélection plus cohérent, comme LB, $LBWLC$ ou UB.

3.4.2.2 Taille de la liste tabou

Nous avons réalisé plusieurs expérimentations afin de déterminer la taille de la liste taboue la plus convenable. Les résultats sont illustrés sur la Figure 3.11, où « Ratio Qualité moyen » correspond à R_{QT_i} et est calculé avec (3.39). Les tests correspondent à 151 instances obtenues en faisant varier la taille du problème d'ordonnancement, le nombre de périodes et certains paramètres, comme dans la Section 3.5. À part le dernier groupe d'instances, correspondant

à un atelier de type job-shop 20x5 avec 20 périodes, où toutes les tailles aboutissent à une performance identique, les résultats sont très variables. En moyenne, les tailles avec 10 et 15 éléments conduisent à de meilleures solutions.

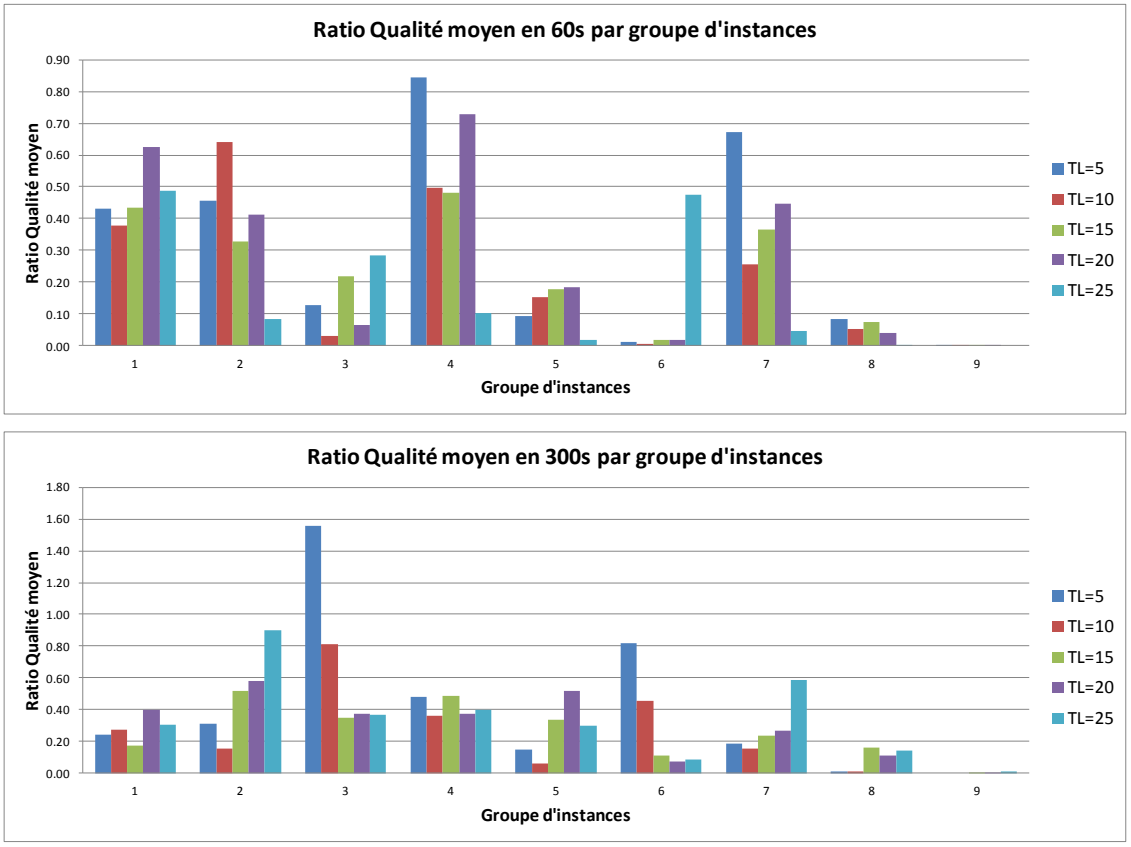

FIGURE 3.11 – Résultats expérimentaux avec plusieurs tailles de liste taboue

3.4.2.3 Sélection de l'arc à changer

Dans l'approche de Wolosewicz *et al.*, la sélection de l'arc à inverser est basée uniquement sur la somme de multiplicateurs Lagrangiens ; une stratégie qui favorise les temps de calcul, mais qui peut conduire à de très mauvaises solutions. C'est pourquoi nous avons analysé d'autres critères de sélection. Comme commenté dans la Section 3.3.2.3, nous avons comparé des stratégies basées sur les valeurs de $LB(y)$, $LBWLC(y)$ et UB. Globalement, la performance de $LBLWLC$ comme critère de sélection n'a pas été satisfaisante. Nous avons par conséquent focalisé l'analyse sur LB et UB.

Comme discuté préalablement, UB guide vers de très bonnes solutions à court terme, mais le temps de calcul de l'heuristique de lissage est très important. Donc, le nombre de séquence changées est limité, ce qui fait que la solution finale du problème intégré n'est pas suffisamment bonne. Sur 151 instances, la méthode guidée par LB a conduit à de meilleures solutions pour 61 instances (39.87%) ; tandis que UB a permis d'obtenir de meilleurs résultats pour 47 instances (30.72%). L'avantage de LB est aussi par rapport au nombre de séquences changées, qui est en moyenne supérieur de 120.78% en le comparant avec UB.

3.4.2.4 Intensification vs. Diversification

Comme dans beaucoup d'heuristiques, il s'agit d'établir un compromis entre intensification et diversification. L'intensification est utilisée pour améliorer la solution dans un espace de recherche réduit, i.e. on se rapproche d'un optimum local ; tandis que la diversification cherche à explorer de nombreuses branches de l'espace de recherche afin de pouvoir potentiellement se rapprocher de l'optimum global. La bonne méthode consiste à équilibrer intensification et diversification. Néanmoins, il n'est pas toujours simple de déterminer le meilleur compromis possible.

Dans notre approche intégrée, nous pouvons identifier différents facteurs de diversification et intensification, comme indiqué dans le Tableau 3.3.

Algorithme/Méthode	**Diversification**	**Intensification**
Ordonnancement	Faire plusieurs modifications du graphe disjonctif en même temps	Changer un seul arc par itération
Voisinage	Sous-ensemble des arcs violés	Ensemble complet des arcs violés
Heuristique de lissage	Sans amélioration du coût	Avec amélioration
Critère de sélection du voisin	Borne inférieure	Borne supérieure

TABLEAU 3.3 – Compromis entre diversification et intensification

Dans notre cas, la diversification fait référence à la fixation d'un grand nombre de séquences différentes, et l'intensification correspond à l'exploration détaillée de chaque séquence. Comme le temps de calcul est limité, une grande intensification réduit la diversification, i.e. le nombre de séquence fixées diminue. Néanmoins, une petite intensification (pour augmenter la diversification), peut ne pas être suffisante pour trouver une solution adéquate à chaque problème à séquence fixée. Ainsi par exemple, si nous explorons à chaque itération de la recherche taboue l'ensemble complet d'arcs violés, nous pourrons garantir la meilleure solution pour chaque problème à séquence fixée résolu. Cependant, le temps alloué au changement de séquences sera

fortement réduit, et la solution du problème intégré ne sera surement pas intéressante. De la même manière, une heuristique de lissage avec amélioration de UB une fois qu'une solution réalisable est trouvée, donnera au moins un plan de production aussi bon que celui trouvé par l'heuristique sans amélioration. Néanmoins, il est possible que le temps passé à faire des modifications supplémentaires du plan consomme beaucoup de temps (notamment pour des problèmes de grande taille avec un grand nombre de périodes), et que moins de séquences soient changées, ce qui peut conduire à une mauvaise solution du problème intégré.

De manière générale, les expérimentations ont montré qu'il est plus intéressant de diversifier la recherche que de l'intensifier. Cependant, les périodes d'intensification doivent être correctement guidées, afin que la diversification soit efficace. Rien ne sert de résoudre beaucoup de problèmes à séquence fixée si les séquences choisies ne sont pas bonnes ou ne permettent pas de faire évoluer le graphe dans le bon sens.

Des expérimentations pour un atelier de type job-shop avec 20 produits, 5 opérations par produits et 5 machines, avec 5, 10 et 20 périodes, ont montré par exemple, qu'en moyenne et sur une limite de temps de calcul de 60 secondes, l'heuristique de lissage réduisait de 623% le nombre de séquences changées, lorsque l'on utilise la condition 5-$LBWLC$. C'est une raison de plus pour éviter d'appliquer l'heuristique de lissage à chaque itération de l'heuristique Lagrangienne, et pour privilégier LB au lieu de UB comme critère de sélection de l'arc à changer. Néanmoins, dans ce dernier cas, il faut s'assurer que la borne inférieure a été correctement calculée, c'est-à-dire que le nombre d'itérations de la relaxation Lagrangienne et les autres paramètres assurent une convergence correcte de LB.

Une opportunité d'amélioration de notre approche actuelle est sur la génération de séquences. Le changement de la séquence par la simple inversion d'un arc étant une procédure intensive, il peut être plus avantageux de réaliser plusieurs changements d'arcs en même temps, ou que la nouvelle séquence ne soit pas directement dérivée de la séquence précédente.

3.5 Résultats expérimentaux

Dans cette section, nous comparons notre approche avec l'approche de Wolosewicz *et al.* [233, 234, 235] et avec le solveur standard IBM ILOG CPLEX Optimization Studio 12.3. L'objectif est de mesurer la performance de ces trois alternatives, en analysant la relation entre qualité de la solution et temps de calcul. Les deux versions de l'approche intégrée ont été implémentées sur Microsoft Visual C++ 2010 Express, et toutes les expérimentations ont été réalisées avec un ordinateur présentant les caractéristiques suivantes : système d'exploitation de 64 bits (Microsoft Windows 7), processeur Intel core i7 4 Quad @2.0GHz et 8GB de mémoire RAM.

Les problèmes étudiés correspondent aux configurations de type job-shop décrites dans le Tableau 3.4. Nous considérons des horizons de planification de 5, 10 et 20 périodes, pour représenter des problèmes de petite, moyenne et grande taille. Les données ont été générées aléatoirement, et 153 instances ont été créées en faisant varier : la capacité, les coûts de setup et les demandes. Le Tableau 3.5 présente les paramètres généraux (par défaut) utilisés pour générer les instances. Les paramètres à faire varier ont été choisis pour leur importance dans le problème. Ainsi, comme discuté dans les Chapitres 1 et 2, la capacité est un facteur très important en planification de la chaîne logistique. La plupart du temps, la capacité n'est pas gérée correctement dans les systèmes ERP, et par conséquent les plans de production tactiques ne sont pas réalisables au niveau opérationnel. C'est pourquoi, nous considérons des instances avec ratios de capacité petits. Un autre facteur important est le rapport entre les coûts de stockage et les coûts de setup. Des coûts de stockage élevés entraînent le lancement de la production plus souvent pour éviter des grands niveaux d'inventaires ; tandis que des coûts de setup élevés génèrent l'effet inverse. Nous étudions des instances avec des coûts de setup élevés par rapport aux coûts de stockage. Dans ce cas, trouver une solution réalisable est déjà difficile, car des grandes tailles de lot (pour éviter des setups) sont restreintes par une petite capacité. Finalement, faire varier les demandes permet de considérer différents niveaux de capacité, et par conséquent différentes décisions de setup faisant varier les niveaux de stocks.

TABLEAU 3.4 – Problèmes étudiés

Job-Shop	**Description**
6x6	6 produits, 6 opérations par produit, 6 ressources
10x10	10 produits, 10 opérations par produit, 10 ressources
20x5	20 produits, 5 opérations par produit, 5 ressources

TABLEAU 3.5 – Paramètres généraux pour la génération d'instances

Job-Shop	T	cap	c_i^s	$D_{i,l}$	L_i	c_i^p	c_i^{inv}
6x6	5	0,46	30	$[3, 8]$	3	4	1
	10	0,46	30	$[3, 8]$	3	4	1
	20	0,46	30	$[2, 5]$	3	4	1
10x10	5	0,44	30	$[3, 8]$	3	4	1
	10	0,50	30	$[3, 8]$	3	4	1
	20	0,58	20	$[2, 5]$	3	4	1
20x5	5	0,44	30	$[3, 8]$	3	4	1
	10	0,48	30	$[3, 8]$	3	4	1
	20	0,54	25	$[5, 10]$	3	4	1

Le paramètre *cap* correspond à un ratio de capacité, qui est utilisé pour modifier une capacité de base, permettant de fabriquer tous les produits à la période où ils sont demandés, en utilisant une seule ressource. Ce ratio permet donc de mesurer l'impact, en termes de difficulté, sur le calcul d'une borne supérieure quand on augmente ou diminue la capacité de chaque période. Ainsi, la capacité c_l du système à la période l est proportionnelle à $0 < cap < 1$, et est calculée avec (3.40).

$$c_l = cap \sum_{o \in l} \left(p_o^u X_{i(o),l(o)} D_{i(o),l(o)} + s_o^t Y_{i(o),l(o)} \right) \ \forall\ l \tag{3.40}$$

Dit autrement, c_l est le temps total requis pour satisfaire la demande totale de la période l, en utilisant uniquement une ressource, multiplié par un facteur de réduction.

Les coûts sont exprimés en unités monétaires, les délais d'obtention sont donnés en périodes et les demandes varient entre les différentes périodes, selon l'intervalle considéré (par exemple, en considérant les paramètres généraux, dans le job-shop 6x6 avec $T = 5$, les demandes varient entre 3 et 5 unités). Pour chaque groupe d'instances, nous fixons les paramètres généraux (Tableau 3.5) et nous faisons varier, soit les coûts de setup, soit les demandes, soit le ratio de capacité par période.

Dans ce qui suit, $UB1$, $UB2$ et CPL correspondent aux solutions déterminées avec notre approche, l'approche de Wolosewicz *et al.* et le solveur IBM ILOG CPLEX, respectivement. G_1 est l'écart entre $UB1$ et $UB2$, G_2 est l'écart entre $UB1$ et CPL et G_3 est l'écart entre $UB2$ et CPL. Les écarts sont calculés avec les équations (3.41), (3.42) et (3.43). PV est la valeur du paramètre qui varie et LBA est la valeur de la borne inférieure absolue (donnée à titre indicatif). Les coûts de setup sont particulièrement élevés par rapport à la valeur par défaut, afin d'analyser les instances les plus difficiles, c'est-à-dire les cas où le ratio entre les coûts de setup et les coûts de stockage est élevé. Les expérimentations ont été réalisées avec des limites de temps de calcul de 60 secondes et 300 secondes.

$$G_1 = \frac{UB2 - UB1}{UB1} \times 100 \tag{3.41}$$

$$G_2 = \frac{UB1 - CPL}{CPL} \times 100 \tag{3.42}$$

$$G_3 = \frac{UB2 - CPL}{CPL} \times 100 \tag{3.43}$$

3.5.1 Expérimentations avec un atelier de type job-shop 6x6

Les solutions obtenues avec les deux approches et le solveur standard sont présentées dans le Tableau 3.6. Les instances sont classées par nombre de périodes et par paramètre qui varie. Les résultats illustrés correspondent aux bornes supérieures et aux écarts entre les solutions des procédures, pour des limites de temps de calcul de 60 secondes et 300 secondes. Nous comparons, dans ce qui suit, la performance de notre approche avec celle du solveur standard et avec celle de l'approche de Wolosewicz *et al.*. Les écarts entre les solutions de l'approche de Wolosewicz *et al.* et celles du solveur sont présentés à titre indicatif.

3.5.1.1 Comparaison entre notre approche et le solveur IBM ILOG CPLEX

Avec 5 périodes de planification et en 60 secondes de temps de calcul, IBM ILOG CPLEX est capable de résoudre de façon optimale 7 des 17 instances. D'autre part, notre approche trouve la solution optimale pour une instance (*Défaut*), et les écarts correspondants aux autres résultats sont globalement favorables au solveur standard, mais ils ne sont pas significatifs (l'écart moyen est égal à 0.38%). De plus, sur 4 instances, notre approche donne de meilleures solutions. Avec un temps de calcul élargi à 300 secondes, le solveur arrive à trouver la solution optimale de 13 instances. Nos solutions restent encore proches, avec un écart maximum de 5.39% (instance *Demande 1*).

En augmentant le nombre de périodes de 5 à 10, nous observons qu'il devient plus difficile pour le solveur de résoudre les différentes instances, avec une limite de temps de 60 secondes. En effet, il n'est pas capable de trouver des solutions réalisables pour 9 instances (53%), et hormis l'instance *Capacité 2*, toutes les solutions obtenues avec notre approche sont meilleures que celles fournies par IBM ILOG CPLEX. Pour l'instance *Demande 4*, notre approche trouve la solution optimale. Avec un temps de calcul de 300 secondes, le solveur arrive à trouver une solution réalisable pour toutes les instances avec 6 solutions optimales. Dans ce cas, des écarts importants (entre 6.93% et 14.79%) avec notre approche, en faveur du solveur, sont constatés pour les instances avec grands coûts de setup. Cepedant, pour l'instance *Coût de setup 7*, un grand écart (6.71%) est en faveur de notre approche. Pour les autres instances, les écarts sont petits (inférieurs à 1%), avec 3 solutions en faveur de notre approche.

La difficulté de résolution augmente encore lorsque le nombre de périodes est égal à 20. Comme pour le cas avec 10 périodes, quand le temps de calcul est limité à 60 secondes, IBM ILOG CPLEX n'est pas capable de trouver des solutions réalisables pour 9 instances (majoritairement celles avec coûts de setup élevés). De plus, les écarts entre les solutions du solveur et celles de notre approche, sont largement favorables à notre méthode, avec des valeurs entre

TABLEAU 3.6 – Résultats pour le job-shop 6x6

T	**Instance**	PV	LBA	Résultats après 60 secondes						Résultats après 300 secondes					
				$UB1$	$UB2$	CPL	G_1	G_2	G_3	$UB1$	$UB2$	CPL	G_1	G_2	G_3
5	Défaut	Connu	1190	1201*	1201*	1201*	0.00	0.00	0.00	1201*	1201*	1201*	0.00	0.00	0.00
	Capacité 1	0.42	1190	1233	1271	1261	3.08	−2.22	0.79	1233	1271	1207	3.08	2.15	5.30
	Capacité 2	0.45	1190	1217	1225	1224	0.66	−0.57	0.08	1207	1225	1204	1.49	0.25	1.74
	Capacité 3	0.48	1190	1201	1221	1198*	1.67	0.25	1.92	1200	1221	1198*	1.75	0.17	1.92
	Capacité 4	0.50	1190	1198	1215	1195*	1.42	0.25	1.67	1198	1215	1195*	1.42	0.25	1.67
	Coût de setup 1	80	1529	1807	1880	1804	4.04	0.17	4.21	1807	1880	1801*	4.04	0.33	4.39
	Coût de setup 2	100	1649	2047	2168	2044	5.91	0.15	6.07	2047	2168	2041*	5.91	0.29	6.22
	Coût de setup 3	120	1769	2287	2309	2281	0.96	0.26	1.23	2287	2309	2281*	0.96	0.26	1.23
	Coût de setup 4	140	1889	2527	2678	2527	5.98	0.00	5.98	2527	2678	2521*	5.98	0.24	6.23
	Coût de setup 5	180	2129	3032	3343	3003	10.26	0.97	11.32	3004	3343	3001*	11.28	0.10	11.40
	Coût de setup 6	200	2249	3257	3447	3258	5.83	−0.03	5.80	3247	3447	3258	6.16	−0.34	5.80
	Coût de setup 7	220	2369	3487	3526	3484	1.12	0.09	1.21	3487	3526	3484	1.12	0.09	1.21
	Demande 1	$[1, 5]$	696	803	809	761*	0.75	5.52	6.31	802	809	761*	0.87	5.39	6.31
	Demande 2	$[1, 10]$	906	972	1015	956*	4.42	1.67	6.17	972	1015	956*	4.42	1.67	6.17
	Demande 3	$[4, 8]$	1202	1258	1346	1220*	7.00	3.11	10.33	1258	1346	1220*	7.00	3.11	10.33
	Demande 4	$[10, 15]$	2233	2247	2255	2324	0.36	−3.31	−2.97	2247	2255	2242*	0.36	0.22	0.58
	Demande 5	$[5, 15]$	1760	1787	1798	1785*	0.62	0.11	0.73	1787	1798	1785*	0.62	0.11	0.73
10	Défaut	Connu	2347	2410	2443	-	1.37	$-\infty$	$-\infty$	2373	2443	2378	2.95	−0.21	2.73
	Capacité 1	0.47	2347	2420	2482	-	2.56	$-\infty$	$-\infty$	2388	2482	2366*	3.94	0.93	4.90
	Capacité 2	0.48	2347	2400	2422	2375	0.92	1.05	1.98	2373	2422	2364*	2.06	0.38	2.45
	Capacité 3	0.53	2347	2368	2470	2692	4.31	−12.04	−8.25	2363	2470	2359*	4.53	0.17	4.71
	Capacité 4	0.54	2347	2366	2370	2427	0.17	−2.51	−2.35	2363	2370	2364	0.30	−0.04	0.25
	Coût de setup 1	80	3038	3579	3825	-	6.87	$-\infty$	$-\infty$	3563	3825	3332	7.35	6.93	14.80
	Coût de setup 2	100	3278	4075	4231	-	3.83	$-\infty$	$-\infty$	4001	4231	3620	5.75	10.52	16.88
	Coût de setup 3	120	3518	4694	4542	4786	−3.24	−1.92	−5.10	4375	4542	3996	3.82	9.48	13.66
	Coût de setup 4	140	3730	5009	5089	-	1.60	$-\infty$	$-\infty$	4829	5072	4488	5.03	7.60	13.01
	Coût de setup 5	180	4005	6098	5962	-	−2.23	$-\infty$	$-\infty$	5673	5962	4942	5.09	14.79	20.64
	Coût de setup 6	200	4125	6416	6461	-	0.70	$-\infty$	$-\infty$	6127	6417	5420	4.73	13.04	18.39
	Coût de setup 7	220	4245	6101	6349	-	4.06	$-\infty$	$-\infty$	6101	6349	6540	4.06	−6.71	−2.92
	Demande 1	$[1, 5]$	1426	1472	1526	-	3.67	$-\infty$	$-\infty$	1472	1526	1466*	3.67	0.41	4.09
	Demande 2	$[1, 10]$	2331	2352	2376	2398	1.02	−1.92	−0.92	2352	2376	2342*	1.02	0.43	1.45
	Demande 3	$[4, 8]$	2403	2435	2439	3222	0.16	−24.43	−24.30	2429	2439	2426	0.41	0.12	0.54
	Demande 4	$[10, 15]$	4227	4227*	4269	4257	0.99	−0.70	0.28	4227*	4269	4227*	0.99	0.00	0.99
	Demande 5	$[6, 15]$	3591	3608	3622	3619	0.39	−0.30	0.08	3608	3622	3610	0.39	−0.06	0.33
20	Défaut	Connu	3136	3357	3430	-	2.17	$-\infty$	$-\infty$	3302	3430	-	3.88	$-\infty$	$-\infty$
	Capacité 1	0.50	3136	3333	3351	5183	0.54	−35.69	−35.35	3299	3351	3269	1.58	0.92	2.51
	Capacité 2	0.54	3136	3247	3305	5299	1.79	−38.72	−37.63	3203	3305	3334	3.18	−3.93	0.87
	Capacité 3	0.59	3136	3204	3196	5203	−0.25	−38.42	−38.57	3196	3196	4984	0.00	−35.87	−35.87
	Capacité 4	0.61	3136	3224	3199	5248	−0.78	−38.57	−39.04	3193	3199	3276	0.19	−2.53	−2.35
	Coût de setup 1	40	3385	3539	3590	4246	1.44	−16.65	−15.45	3521	3590	3645	1.96	−3.40	−1.51
	Coût de setup 2	50	3624	3832	3851	7075	0.50	−45.84	−45.57	3810	3851	3864	1.08	−1.40	−0.34
	Coût de setup 3	65	3924	4316	4413	-	2.25	$-\infty$	$-\infty$	4306	4384	5702	1.81	−24.48	−23.11
	Coût de setup 4	80	4213	4759	4818	-	1.24	$-\infty$	$-\infty$	4697	4818	4765	2.58	−1.43	1.11
	Coût de setup 5	90	4393	4988	4912	-	−1.52	$-\infty$	$-\infty$	4988	4912	4954	−1.52	0.69	−0.85
	Coût de setup 6	100	4564	5319	5361	-	0.79	$-\infty$	$-\infty$	5284	5217	5243	−1.27	0.78	−0.50
	Coût de setup 7	120	4875	5812	5929	-	2.01	$-\infty$	$-\infty$	5812	5929	5622	2.01	3.38	5.46
	Demande 1	$[1, 5]$	2734	2833	2780	-	−1.87	$-\infty$	$-\infty$	2818	2780	2800	−1.35	0.64	−0.71
	Demande 2	$[1, 10]$	4048	4097	4096	-	−0.02	$-\infty$	$-\infty$	4097	4096	4110	−0.02	−0.32	−0.34
	Demande 3	$[4, 8]$	4946	4982	4965	7110	−0.34	−29.93	−30.17	4982	4965	5012	−0.34	−0.60	−0.94
	Demande 4	$[6, 15]$	8601	-	-	-	-	-	-	-	-	-	-	-	-
	Demande 5	$[5, 15]$	7231	7254	7265	9015	0.15	−19.53	−19.41	7241	7265	7249	0.33	−0.11	0.22

- Pas de solutions réalisable.

* Solution optimale.

$-\infty$ Pas de solution réalisable avec IBM ILOG CPLEX (écart indéterminé).

TABLEAU 3.7 – Écarts relatifs pour le job-shop 6x6 avec 60 secondes de temps de calcul

T	$G_1(\%)$			$G_2(\%)$			$G_3(\%)$		
	Moy	**Min**	**Max**	**Moy**	**Min**	**Max**	**Moy**	**Min**	**Max**
5	3.18	0.00	10.26	0.38	-3.31	11.32	3.58	-2.97	11.32
10	1.60	-3.24	6.87	-5.35	-24.43	1.98	-4.82	-24.30	1.98
20	0.51	-1.87	2.25	-32.92	-45.84	-15.45	-32.65	-45.57	-15.45

16.65% et 45.84%. Avec un temps de calcul de 300 secondes, le solveur arrive à trouver des solutions réalisables pour presque toutes les instances (à l'exception des instances *Défaut* et *Demande 5*), et les écarts diminuent considérablement. Néanmoins, la plupart des instances sont mieux résolues avec notre approche.

3.5.1.2 Comparaison entre l'approche de Wolosewicz *et al.* et notre approche

Avec un temps de calcul limité à 60 secondes, pour les instances avec $T = 5$, notre approche améliore toutes les solutions de l'approche de Wolosewicz *et al.*, sauf pour la première instance, où les deux méthodes trouvent la solution optimale. Avec $T = 10$, uniquement deux instances (11.76%) sont mieux résolues avec l'approche de Wolosewicz *et al.*. Les 15 restantes (88.24%) sont dominées par notre approche. Finalement, avec $T = 20$, 6 instances (35.29%) sont dominées par l'approche de Wolosewicz *et al.*, 10 (58.82%) par notre approche et une instance n'est pas résolue. Les écarts relatifs moyens, minimum et maximum sont présentés dans le Tableau 3.7. Au total, l'approche de Wolosewicz *et al.* est plus favorable sur 8 instances (15.69%) ; tandis que notre approche donne des meilleures solutions sur 41 instances (80.39%).

Si nous incrémentons le temps de calcul à 300 secondes, pour les instances avec $T = 5$, notre approche trouve à nouveau des meilleures solutions que l'approche de Wolosewicz *et al.* pour toutes les instances, sauf pour la première, où une solution optimale est trouvée avec les 2 procédures. Avec $T = 10$, notre approche domine sur toutes les instances, et avec $T = 20$, l'approche de Wolosewicz *et al.* gagne sur 6 instances (35.29%), la notre sur 10 (58.82%), il y a une parité sur une instance et une instance n'est résolue avec aucune des deux approches, ni par IBM ILOG CPLEX. Les écarts relatifs des expérimentations à 300 secondes sont présentés dans le Tableau 3.8. Au total, l'approche de Wolosewicz *et al.* donne des meilleurs résultats sur 6 instances (11.76%) ; tandis que notre approche domine sur 43 instances (84.31%).

3.5.2 Expérimentations avec un atelier de type job-shop 10x10

Dans la sous-section précédente, nous avons étudié des instances associées à des problèmes de taille relativement petite (job-shop 6x6). Dans ce qui suit, nous nous intéressons à des

TABLEAU 3.8 – Écarts relatifs pour le job-shop 6x6 avec 300 secondes de temps de calcul

T	$G_1(\%)$			$G_2(\%)$			$G_3(\%)$		
	Moy	**Min**	**Max**	**Moy**	**Min**	**Max**	**Moy**	**Min**	**Max**
5	3.32	0.00	11.28	0.84	-0.34	5.39	4.19	0.00	11.40
10	3.30	0.30	7.35	3.40	-6.71	14.79	6.88	-2.92	20.64
20	0.88	-1.52	3.88	-4.51	-35.87	3.38	-3.87	-35.87	5.46

problèmes de moyenne taille, et nous utilisons une configuration 10x10, comme expliqué par le Tableau 3.4. Les résultats des expérimentations sont affichés dans le Tableau 3.9. À nouveau, 51 instances sont étudiées ; elles sont classées par nombre de périodes de planification (5, 10 et 20) et par type de paramètre varié (capacité, coût de setup et demande).

3.5.2.1 Comparaison entre notre approche et le solveur IBM ILOG CPLEX

Résoudre le job-shop 10x10 avec 5, 10 et 20 périodes pose plus de problèmes pour IBM ILOG CPLEX. En effet, le solveur n'est pas capable de trouver des solutions réalisables, ni en 60 secondes ni en 300 secondes. Des expérimentations additionnelles avec un temps de calcul de 600 secondes ont montré le même comportement. Une explication est la grande sollicitation en termes de mémoire. À l'inverse, l'approche intégrée arrive à résoudre toutes les instances, exceptée l'instance *Demande 2* avec $T = 10$.

Ces résultats mettent en évidence la complexité de résolution du problème avec des méthodes exactes, même en utilisant un solveur commercial et une grande capacité de calcul. Notre approche est donc appropriée pour trouver des solutions réalisables pour des problèmes de moyenne taille, en un temps de calcul raisonnable.

3.5.2.2 Comparaison entre l'approche de Wolosewicz *et al.* et notre approche

Avec un temps de calcul limité à 60 secondes, notre approche domine toutes les instances à 5 périodes. Avec 10 périodes, l'approche de Wolosewicz *et al.* est meilleur pour 5 instances (29.41%), la notre pour 11 (64.71%), et une instance n'est pas résolue. Avec 20 périodes, 6 instances (35.29%) sont mieux résolues avec l'approche de Wolosewicz *et al.*, 10 avec la notre (58.82%), et il y a une parité sur une instance. Au total, l'approche de Wolosewicz *et al.* est plus performante sur 11 instances (21.57%) et notre approche domine sur 38 instances (74.51%).

En augmentant le temps de calcul à 300 secondes, toutes les instances avec $T = 5$ sont à nouveau mieux résolues avec notre approche. Avec $T = 10$, 3 instances (17.65%) sont mieux résolues avec l'approche de Wolosewicz *et al.*, 13 avec la notre (76.47%), et une instance reste sans solution. Finalement, avec $T = 20$, l'approche de Wolosewicz *et al.* domine sur 5 instances

TABLEAU 3.9 – Résultats pour le job-shop 10x10

T	**Instance**	PV	LBA	Résultats après 60 secondes						Résultats après 300 secondes					
				$UB1$	$UB2$	CPL	G_1	G_2	G_3	$UB1$	$UB2$	CPL	G_1	G_2	G_3
5	Défaut	Connu	1961	1980	1999	-	0.96	$-\infty$	$-\infty$	1980	1999	-	0.96	$-\infty$	$-\infty$
	Capacité 1	0.42	1961	1986	1999	-	0.65	$-\infty$	$-\infty$	1984	1999	-	0.76	$-\infty$	$-\infty$
	Capacité 2	0.50	1961	1968	1986	-	0.91	$-\infty$	$-\infty$	1968	1986	-	0.91	$-\infty$	$-\infty$
	Capacité 3	0.46	1961	1977	1985	-	0.40	$-\infty$	$-\infty$	1974	1985	-	0.56	$-\infty$	$-\infty$
	Capacité 4	0.48	1961	1976	2002	-	1.32	$-\infty$	$-\infty$	1974	2002	-	1.42	$-\infty$	$-\infty$
	Coût de setup 1	80	2525	2924	3021	-	3.32	$-\infty$	$-\infty$	2913	3021	-	3.71	$-\infty$	$-\infty$
	Coût de setup 2	100	2725	3341	3621	-	8.38	$-\infty$	$-\infty$	3341	3621	-	8.38	$-\infty$	$-\infty$
	Coût de setup 3	120	2925	3642	3823	-	4.97	$-\infty$	$-\infty$	3637	3823	-	5.11	$-\infty$	$-\infty$
	Coût de setup 4	140	3125	4090	4228	-	3.37	$-\infty$	$-\infty$	4090	4228	-	3.37	$-\infty$	$-\infty$
	Coût de setup 5	180	3525	4857	5032	-	3.60	$-\infty$	$-\infty$	4843	5032	-	3.90	$-\infty$	$-\infty$
	Coût de setup 6	200	3725	5245	5467	-	4.23	$-\infty$	$-\infty$	5105	5467	-	7.09	$-\infty$	$-\infty$
	Coût de setup 7	220	3925	5614	5838	-	3.99	$-\infty$	$-\infty$	5614	5838	-	3.99	$-\infty$	$-\infty$
	Demande 1	$[1,5]$	1194	1310	1313	-	0.23	$-\infty$	$-\infty$	1308	1313	-	0.38	$-\infty$	$-\infty$
	Demande 2	$[1,5]$	1623	1654	1656	-	0.12	$-\infty$	$-\infty$	1654	1656	-	0.12	$-\infty$	$-\infty$
	Demande 3	$[2,4]$	1216	1320	1355	-	2.65	$-\infty$	$-\infty$	1320	1355	-	2.65	$-\infty$	$-\infty$
	Demande 4	$[1,2]$	750	942	975	-	3.50	$-\infty$	$-\infty$	918	975	-	6.21	$-\infty$	$-\infty$
	Demande 5	$[5,15]$	3154	3171	3179	-	0.25	$-\infty$	$-\infty$	3169	3179	-	0.32	$-\infty$	$-\infty$
10	Défaut	Connu	3867	3909	3939	-	0.77	$-\infty$	$-\infty$	3898	3939	-	1.05	$-\infty$	$-\infty$
	Capacité 1	0.42	3867	3998	3954	-	-1.10	$-\infty$	$-\infty$	3951	3954	-	0.08	$-\infty$	$-\infty$
	Capacité 2	0.44	3867	3944	3939	-	-0.13	$-\infty$	$-\infty$	3937	3939	-	0.05	$-\infty$	$-\infty$
	Capacité 3	0.46	3867	3929	3937	-	-0.23	$-\infty$	$-\infty$	3920	3928	-	0.20	$-\infty$	$-\infty$
	Capacité 4	0.48	3867	3911	3924	-	0.33	$-\infty$	$-\infty$	3911	3924	-	0.33	$-\infty$	$-\infty$
	Coût de setup 1	80	5035	5518	5597	-	1.43	$-\infty$	$-\infty$	5452	5397	-	-1.01	$-\infty$	$-\infty$
	Coût de setup 2	110	5635	6490	6525	-	0.54	$-\infty$	$-\infty$	6376	6454	-	1.22	$-\infty$	$-\infty$
	Coût de setup 3	130	6033	7047	7199	-	2.16	$-\infty$	$-\infty$	6995	7108	-	1.62	$-\infty$	$-\infty$
	Coût de setup 4	150	6353	7733	7628	-	-1.36	$-\infty$	$-\infty$	7661	7628	-	-0.43	$-\infty$	$-\infty$
	Coût de setup 5	180	6685	8547	9092	-	6.38	$-\infty$	$-\infty$	8396	9092	-	8.29	$-\infty$	$-\infty$
	Coût de setup 6	200	6885	9196	9395	-	2.16	$-\infty$	$-\infty$	8999	9395	-	4.40	$-\infty$	$-\infty$
	Coût de setup 7	220	7085	9759	10002	-	2.49	$-\infty$	$-\infty$	9590	9831	-	2.51	$-\infty$	$-\infty$
	Demande 1	$[1,5]$	2215	2397	2420	-	0.96	$-\infty$	$-\infty$	2367	2420	-	2.24	$-\infty$	$-\infty$
	Demande 2	$[1,6]$	1982	-	-	-	-	-	-	-	-	-	-	-	-
	Demande 3	$[1,5]$	1482	1653	1693	-	2.42	$-\infty$	$-\infty$	1640	1646	-	0.37	$-\infty$	$-\infty$
	Demande 4	$[1,3]$	1285	1322	1309	-	-0.98	$-\infty$	$-\infty$	1316	1309	-	-0.53	$-\infty$	$-\infty$
	Demande 5	$[5,15]$	5764	5780	5789	-	0.16	$-\infty$	$-\infty$	5780	5789	-	0.16	$-\infty$	$-\infty$
20	Défaut	Connu	4769	4788	4788	-	0.00	$-\infty$	$-\infty$	4782	4788	-	0.13	$-\infty$	$-\infty$
	Capacité 1	0.45	4769	4856	4845	-	-0.23	$-\infty$	$-\infty$	4848	4823	-	-0.52	$-\infty$	$-\infty$
	Capacité 2	0.49	4769	4824	4843	-	0.39	$-\infty$	$-\infty$	4819	4843	-	0.50	$-\infty$	$-\infty$
	Capacité 3	0.50	4769	4819	4816	-	-0.06	$-\infty$	$-\infty$	4813	4816	-	0.06	$-\infty$	$-\infty$
	Capacité 4	0.52	4769	4818	4799	-	-0.39	$-\infty$	$-\infty$	4816	4799	-	-0.35	$-\infty$	$-\infty$
	Coût de setup 1	40	5701	5893	5902	-	0.15	$-\infty$	$-\infty$	5881	5876	-	-0.09	$-\infty$	$-\infty$
	Coût de setup 2	55	6293	6621	6644	-	0.35	$-\infty$	$-\infty$	6611	6523	-	-1.33	$-\infty$	$-\infty$
	Coût de setup 3	80	7102	7690	7641	-	-0.64	$-\infty$	$-\infty$	7690	7641	-	-0.64	$-\infty$	$-\infty$
	Coût de setup 4	90	7402	8188	8227	-	0.48	$-\infty$	$-\infty$	8160	8227	-	0.82	$-\infty$	$-\infty$
	Coût de setup 5	100	7693	8715	8693	-	-0.25	$-\infty$	$-\infty$	8618	8692	-	0.86	$-\infty$	$-\infty$
	Coût de setup 6	120	8213	9558	9740	-	1.90	$-\infty$	$-\infty$	9558	9740	-	1.90	$-\infty$	$-\infty$
	Coût de setup 7	125	8333	9875	10004	-	1.31	$-\infty$	$-\infty$	9793	9800	-	0.07	$-\infty$	$-\infty$
	Demande 1	$[1,2]$	2379	2419	2416	-	-0.12	$-\infty$	$-\infty$	2413	2416	-	0.12	$-\infty$	$-\infty$
	Demande 2	$[0,9]$	5489	5562	5577	-	0.27	$-\infty$	$-\infty$	5557	5577	-	0.36	$-\infty$	$-\infty$
	Demande 3	$[1,5]$	3535	3616	3650	-	0.94	$-\infty$	$-\infty$	3601	3629	-	0.78	$-\infty$	$-\infty$
	Demande 4	$[0,9]$	5762	5825	5897	-	0.77	$-\infty$	$-\infty$	5809	5887	-	1.34	$-\infty$	$-\infty$
	Demande 5	$[0,9]$	7047	7163	7187	-	0.34	$-\infty$	$-\infty$	7123	7187	-	0.90	$-\infty$	$-\infty$

- Pas de solution réalisable.
* Solution optimale.
$-\infty$ Pas de solution réalisable avec IBM ILOG CPLEX (écart indéterminé).

TABLEAU 3.10 – Écarts relatifs pour le job-shop 10x10

T	$G_1(\%)$ **après 60 secondes**			$G_1(\%)$ **après 300 secondes**		
	Moy	**Min**	**Max**	**Moy**	**Min**	**Max**
5	2,52	0.12	8.38	2.93	0.12	8.38
10	1.00	-1.36	6.38	1.28	-1.01	8.29
20	0.31	-0.64	1.90	0.29	-1.33	1.90

(29.41%) et la notre sur les 12 instances restantes (70.59%). Les écarts relatifs sont présentés dans le Tableau 3.10. Au total, l'approche de Wolosewicz *et al.* est plus efficace sur 8 instances (15.69%) ; tandis que la notre domine sur 42 instances (82.35%).

3.5.3 Expérimentations avec un atelier de type job-shop 20x5

Nous nous intéressons finalement à la résolution de problèmes de grande taille, correspondant à un atelier de type job-shop avec 20 produits, 5 opérations par produit et 5 ressources (configuration 20x5). Des instances considérant des variations sur les paramètres et les horizons de planification précédemment étudiés, sont également analysées dans ce cas. Les résultats sont présentés dans le Tableau 3.11.

3.5.3.1 Comparaison entre notre approche et le solveur IBM ILOG CPLEX

De manière similaire à ce que nous avons constaté pour le job-shop 10x10, résoudre le job-shop 20x5 suppose un effort de calcul très important. IBM ILOG CPLEX est en effet incapable de déterminer des solutions réalisables pour toutes les instances. Les résultats sont corroborés avec une limite de temps de calcul de 600 secondes. Avec notre approche, des plans de production réalisables sont trouvés pour toutes les instances. Une fois de plus, les limitations des solveurs commerciaux et la pertinence de notre approche sont mises en évidence. L'intégration des décisions de planification et d'ordonnancement de la production est très importante en management de la chaîne logistique, mais le problème issu d'une modélisation intégrée est très complexe à résoudre et nécessite la conception de méthodes dédiées, comme l'approche que nous proposons.

3.5.3.2 Comparaison entre l'approche de Wolosewicz *et al.* et notre approche

Avec les deux limites de temps de calcul (60 secondes et 300 secondes) notre approche domine celle de Wolosewicz *et al.* sur toutes les instances, avec toutes les longueurs d'horizon de

TABLEAU 3.11 – Résultats pour le job-shop 20x5

T	**Instance**	PV	LBA	Résultats après 60 secondes						Résultats après 300 secondes					
				$UB1$	$UB2$	CPL	G_1	G_2	G_3	$UB1$	$UB2$	CPL	G_1	G_2	G_3
5	Défaut	Connu	3921	4023	4035	-	0.30	$-\infty$	$-\infty$	4023	4035	-	0.30	$-\infty$	$-\infty$
	Capacité 1	0.42	3921	4046	4052	-	0.15	$-\infty$	$-\infty$	4046	4052	-	0.15	$-\infty$	$-\infty$
	Capacité 2	0.50	3921	3990	4000	-	0.25	$-\infty$	$-\infty$	3990	4000	-	0.25	$-\infty$	$-\infty$
	Capacité 3	0.46	3921	4022	4038	-	0.40	$-\infty$	$-\infty$	4022	4038	-	0.40	$-\infty$	$-\infty$
	Capacité 4	0.48	3921	3999	4005	-	0.15	$-\infty$	$-\infty$	3999	4005	-	0.15	$-\infty$	$-\infty$
	Coût de setup 1	80	5055	6059	6173	-	1.88	$-\infty$	$-\infty$	6035	6127	-	1.52	$-\infty$	$-\infty$
	Coût de setup 2	100	5455	6929	7006	-	1.11	$-\infty$	$-\infty$	6839	7006	-	2.44	$-\infty$	$-\infty$
	Coût de setup 3	120	5855	7572	7788	-	2.85	$-\infty$	$-\infty$	7572	7788	-	2.85	$-\infty$	$-\infty$
	Coût de setup 4	140	6255	8497	8586	-	1.05	$-\infty$	$-\infty$	8449	8586	-	1.62	$-\infty$	$-\infty$
	Coût de setup 5	180	7055	10074	10529	-	4.52	$-\infty$	$-\infty$	9902	10529	-	6.33	$-\infty$	$-\infty$
	Coût de setup 6	200	7455	11039	11057	-	0.16	$-\infty$	$-\infty$	10871	11057	-	1.71	$-\infty$	$-\infty$
	Coût de setup 7	220	7855	11687	11872	-	1.58	$-\infty$	$-\infty$	11463	11868	-	3.53	$-\infty$	$-\infty$
	Demande 1	$[1, 5]$	2413	2727	2833	-	3.89	$-\infty$	$-\infty$	2704	2833	-	4.77	$-\infty$	$-\infty$
	Demande 2	$[1, 10]$	4056	4165	4187	-	0.53	$-\infty$	$-\infty$	4164	4171	-	0.17	$-\infty$	$-\infty$
	Demande 3	$[4, 8]$	4057	4210	4334	-	2.95	$-\infty$	$-\infty$	4205	4334	-	3.07	$-\infty$	$-\infty$
	Demande 4	$[10, 15]$	7153	7212	7222	-	0.14	$-\infty$	$-\infty$	7212	7222	-	0.14	$-\infty$	$-\infty$
	Demande 5	$[5, 15]$	6230	6311	6353	-	0.67	$-\infty$	$-\infty$	6311	6353	-	0.67	$-\infty$	$-\infty$
10	Défaut	Connu	7614	7732	7761	-	0.38	$-\infty$	$-\infty$	7730	7761	-	0.40	$-\infty$	$-\infty$
	Capacité 1	0.42	7614	7814	7846	-	0.41	$-\infty$	$-\infty$	7774	7846	-	0.93	$-\infty$	$-\infty$
	Capacité 2	0.44	7614	7773	7873	-	1.29	$-\infty$	$-\infty$	7773	7873	-	1.29	$-\infty$	$-\infty$
	Capacité 3	0.46	7614	7771	7781	-	0.13	$-\infty$	$-\infty$	7744	7781	-	0.48	$-\infty$	$-\infty$
	Capacité 4	0.50	7614	7726	7764	-	0.49	$-\infty$	$-\infty$	7719	7764	-	0.58	$-\infty$	$-\infty$
	Coût de setup 1	80	9968	10939	11280	-	3.12	$-\infty$	$-\infty$	10852	11280	-	3.94	$-\infty$	$-\infty$
	Coût de setup 2	100	10768	12159	12606	-	3.68	$-\infty$	$-\infty$	12097	12606	-	4.21	$-\infty$	$-\infty$
	Coût de setup 3	120	11568	13492	13960	-	3.47	$-\infty$	$-\infty$	13402	13960	-	4.16	$-\infty$	$-\infty$
	Coût de setup 4	150	12566	15352	16017	-	4.33	$-\infty$	$-\infty$	15173	16017	-	5.56	$-\infty$	$-\infty$
	Coût de setup 5	180	13233	16851	17759	-	5.39	$-\infty$	$-\infty$	16779	17759	-	5.84	$-\infty$	$-\infty$
	Coût de setup 6	200	13633	18409	19388	-	5.32	$-\infty$	$-\infty$	18068	19388	-	7.31	$-\infty$	$-\infty$
	Coût de setup 7	220	14033	19449	20572	-	5.77	$-\infty$	$-\infty$	19253	20572	-	6.85	$-\infty$	$-\infty$
	Demande 1	$[1, 5]$	4692	4894	4919	-	0.51	$-\infty$	$-\infty$	4886	4919	-	0.68	$-\infty$	$-\infty$
	Demande 2	$[1, 10]$	7393	7522	7569	-	0.62	$-\infty$	$-\infty$	7517	7569	-	0.69	$-\infty$	$-\infty$
	Demande 3	$[4, 8]$	7926	8062	8094	-	0.79	$-\infty$	$-\infty$	8062	8126	-	0.79	$-\infty$	$-\infty$
	Demande 4	$[10, 15]$	14363	14418	14440	-	0.15	$-\infty$	$-\infty$	14415	14440	-	0.17	$-\infty$	$-\infty$
	Demande 5	$[5, 15]$	11951	11981	11985	-	0.03	$-\infty$	$-\infty$	11981	11985	-	0.03	$-\infty$	$-\infty$
20	Défaut	Connu	18272	18323	18335	-	0.07	$-\infty$	$-\infty$	18317	18335	-	0.03	$-\infty$	$-\infty$
	Capacité 1	0.45	18272	18352	18362	-	0.05	$-\infty$	$-\infty$	18352	18362	-	0.05	$-\infty$	$-\infty$
	Capacité 2	0.48	18272	18339	18372	-	0.18	$-\infty$	$-\infty$	18339	18372	-	0.18	$-\infty$	$-\infty$
	Capacité 3	0.50	18272	18340	18372	-	0.17	$-\infty$	$-\infty$	18340	18372	-	0.17	$-\infty$	$-\infty$
	Capacité 4	0.52	18272	18330	18346	-	0.09	$-\infty$	$-\infty$	18330	18346	-	0.09	$-\infty$	$-\infty$
	Coût de setup 1	5	14100	14100*	14100*	-	0.00	$-\infty$	$-\infty$	14100*	14100*	-	0.00	$-\infty$	$-\infty$
	Coût de setup 2	20	17463	17480	17495	-	0.09	$-\infty$	$-\infty$	17479	17495	-	0.09	$-\infty$	$-\infty$
	Coût de setup 3	50	21443	21677	21706	-	0.13	$-\infty$	$-\infty$	21677	21703	-	0.12	$-\infty$	$-\infty$
	Coût de setup 4	100	26001	27107	27116	-	0.03	$-\infty$	$-\infty$	27096	27116	-	0.07	$-\infty$	$-\infty$
	Coût de setup 5	150	29504	31942	31970	-	0.09	$-\infty$	$-\infty$	31942	31970	-	0.09	$-\infty$	$-\infty$
	Coût de setup 6	200	32497	36865	37218	-	0.96	$-\infty$	$-\infty$	36539	37218	-	1.86	$-\infty$	$-\infty$
	Coût de setup 7	250	35314	40692	41086	-	0.97	$-\infty$	$-\infty$	40273	41086	-	2.02	$-\infty$	$-\infty$
	Demande 1	$[5, 7]$	15217	15263	15267	-	0.03	$-\infty$	$-\infty$	14263	15267	-	0.03	$-\infty$	$-\infty$
	Demande 2	$[1, 2]$	5517	5753	5774	-	0.37	$-\infty$	$-\infty$	5737	5765	-	0.49	$-\infty$	$-\infty$
	Demande 3	$[1, 3]$	6527	6715	6770	-	0.82	$-\infty$	$-\infty$	6715	6770	-	0.82	$-\infty$	$-\infty$
	Demande 4	$[3, 6]$	12131	12194	12196	-	0.02	$-\infty$	$-\infty$	12192	12196	-	0.03	$-\infty$	$-\infty$
	Demande 5	$[4, 8]$	15552	15576	15596	-	0.13	$-\infty$	$-\infty$	15576	15596	-	0.13	$-\infty$	$-\infty$

- Pas de solution réalisable.
* Solution optimale.
$-\infty$ Pas de solution réalisable avec IBM ILOG CPLEX (écart indéterminé).

TABLEAU 3.12 – Écarts relatifs pour le job-shop 20x5

T	$G_1(\%)$ **après 60 secondes**			$G_1(\%)$ **après 300 secondes**		
	Moy	**Min**	**Max**	**Moy**	**Min**	**Max**
5	1.33	0.14	4.52	1.77	0.14	6.33
10	2.11	0.03	5.77	2.58	0.03	7.31
20	0.25	0.00	0.97	0.37	0.00	2.02

planification (5, 10 et 20 périodes). Notre approche est ainsi fortement dominante avec une solution de meilleure qualité sur 50 instances (98.04%) pour 60 secondes et 300 secondes de temps de calcul. Les écarts relatifs sont présentés dans le Tableau 3.12. Ces résultats mettent donc en évidence la pertinence de toutes les améliorations apportées sur la méthode de Wolosewicz *et al.*, que nous avons décrites dans ce chapitre.

3.6 Conclusion

Nous avons présenté une nouvelle approche intégrée pour résoudre le problème intégré de dimensionnement de lots et d'ordonnancement en garantissant des solutions réalisables au niveau opérationnel, contrairement aux approches traditionnelles où les décisions sont prises de manière hiérarchique. La plupart des approches existantes peuvent résoudre des problèmes mono-ressource, car les contraintes de capacité sont agrégées. De plus, à cause des limitations sur l'utilisation de la capacité, les solutions sont souvent sous-optimales.

Dans notre approche, des contraintes de capacité détaillées sont considérées pour générer des solutions réalisables dans des systèmes de production complexes. De plus, l'ordonnancement est effectué sur tout l'horizon de planification, et non pas période par période, ce qui permet d'utiliser la capacité totale d'une manière efficiente. Le problème de dimensionnement de lots est résolu pour une séquence fixée avec une heuristique Lagrangienne, et une procédure d'amélioration de la séquence est appliquée pour faire évoluer l'ordonnancement. Ces deux étapes sont connectées de manière itérative à travers une recherche taboue, qui permet d'explorer plusieurs branches de l'espace des solutions.

Cette nouvelle approche intégrée pour la résolution de problèmes avec production mono-niveau, est le fruit de plusieurs améliorations que nous avons apportées sur le travail précurseur de Wolosewicz *et al.* [233, 234, 235]. Des analyses d'autres idées étudiées qui n'ont pas permis d'obtenir de meilleurs résultats ont été également présentées. Les modifications effectuées sur la méthode de résolution ont permis d'obtenir de meilleurs résultats pour la grande majorité des instances testées, et globalement pour tous les groupes d'instances étudiées avec toutes les tailles d'horizon de planification. Par rapport au solveur commercial IBM ILOG CPLEX, l'écart

sur les solutions est encore plus favorable avec notre approche. Il est important de remarquer que le solveur n'est pas capable d'obtenir des solutions réalisables pour les systèmes étudiés avec des horizons de planification comportant 10 et 20 périodes. D'où l'importance de proposer des méthodes de résolution dédiées et la pertinence, dans ce sens, de notre approche intégrée.

Cette approche représente une avancée importante en matière d'intégration de décisions dans la gestion de la chaîne logistique. Elle permet en effet d'intégrer des flux verticaux associés à l'étape de production entre les niveaux tactique et opérationnel. Néanmoins, d'autres activités importantes sont encore à considérer, afin de pouvoir déterminer des solutions qui optimisent globalement la chaîne logistique. C'est pourquoi nous proposons, dans le chapitre suivant, une extension de l'approche au cas de systèmes multi-niveaux. Des flux de l'activité d'approvisionnement sont intégrés (flux horizontaux), notamment pour lier les besoins de production entre les composants et composés de la nomenclature, qui peuvent être fabriqués sur des sites de production différents.