RAIRO-Oper. Res. 43 (2009) 277–296
DOI: 10.1051/ro/2009015

RAIRO Operations Research
www.rairo-ro.org

CONFIGURATION DES LIGNES D'USINAGE À BOÎTIERS MULTIBROCHES : UNE APPROCHE MIXTE*

Olga Guschinskaya[1] et Alexandre Dolgui[1]

Résumé. Ce travail porte sur l'optimisation des lignes d'usinage pour la grande série. Une telle ligne comporte plusieurs postes de travail, chacun étant équipé avec boîtiers multibroches. Un boîtier multibroche exécute plusieurs opérations en parallèle. Lors de la conception en avant-projet, il est nécessaire d'affecter toutes les opérations à des boîtiers et des postes de travail de sorte à minimiser le nombre de postes et de boîtiers utilisés. Pour ce nouveau problème d'équilibrage des lignes de production, nous proposons une approche de résolution par décomposition en utilisant des méthodes exactes et heuristiques. Les résultats des tests numériques effectués sur des instances proches des problèmes réels sont présentés et analysés.

Mots Clés. Lignes d'usinage, boîtiers multibroches, équilibrage des lignes, optimisation, décomposition, méthodes exactes, heuristiques.

Abstract. This paper deals with the optimization of machining lines for mass production. Such a line consists of a sequence of machines equipped with several spindle heads each. A spindle head performs all its operations simultaneously. At the preliminary design stage, the goal is to assign all the operations to spindle heads and machines minimizing the number of machines and spindle heads required. This is a new line balancing problem. An optimization approach is suggested. It is based conjointly on the use of decomposition and exact and heuristic methods. The results of numerical tests made on instances similar to real industrial problems are presented and analyzed.

Keywords. Machining lines, multi-spindle stations, line balancing, optimization, decomposition, methods exacts, heuristics.

Classification Mathématique. 90B80, 90C27, 49M27, 90C59.

Received May 10, 2006. Accepted February 14, 2009.

* *Ce travail a été supporté par le projet INTAS 03-51-5501.*

[1] Centre Génie Industriel et Informatique, École des Mines de Saint-Étienne, 158 Cours Fauriel, 42023 Saint-Étienne Cedex 2, France; guschinskaya@emse.fr; dolgui@emse.fr

Introduction

Dans cet article, nous étudions le problème d'optimisation de la configuration des lignes d'usinage dédiées. Les lignes de ce type forment le coeur de beaucoup de systèmes de fabrication en grande série [18]. Une ligne d'usinage dédiée est conçue pour fabriquer un seul type de pièce. Une telle ligne est constituée de postes de travail reliés par un dispositif de transfert de pièces. Pour fabriquer une pièce, il est nécessaire de réaliser un ensemble d'opérations d'usinage sur chaque poste de travail. La pièce visite les postes dans l'ordre de leur disposition dans la ligne. Toutes les pièces se trouvant sur la ligne sont déplacées vers le poste de travail suivant simultanément. Afin d'assurer ce déplacement synchrone de pièces, le temps d'usinage sur chaque poste doit être inférieur ou égal au temps de cycle, désigné par T_0.

Pour effectuer les opérations d'usinage qui lui sont assignées, chaque poste de travail est muni d'un ou de plusieurs boîtiers avec outils tranchants. Souvent ces boîtiers sont multibroches (comportant plusieurs outils). Chaque boîtier active simultanément tous ses outils pour effectuer les opérations qui lui sont affectées. Un boîtier multibroche permet donc d'effectuer un ensemble d'opérations en parallèle. Dans ce qui suit, nous appelons l'ensemble des opérations exécutées par un boîtier *bloc d'opérations*. Si un poste comporte plusieurs boîtiers, ils sont activés séquentiellement dans un ordre fixé à l'étape de conception de la ligne. Le changement du boîtier actif nécessite un temps auxiliaire, désigné par τ^b. Ce changement n'est pas synchronisé entre les différents postes de travail.

À l'étape de la conception d'une telle ligne, l'ensemble de toutes les opérations, nécessaires pour fabriquer la pièce, est connu. Il convient donc d'affecter ces opérations à des blocs et des postes de travail. Cette affectation détermine la configuration de chaque boîtier et chaque poste de travail et, par conséquent, la configuration de la ligne et son coût. Une fois cette configuration approuvée, elle est figée et ne peut plus être modifiée ultérieurement à l'étape d'exploitation de la ligne.

Un regroupement des opérations en blocs et postes de travail définit l'ordre et le temps d'exécution des opérations ainsi que le nombre de postes et de boîtiers utilisés. Cette répartition doit tenir compte de nombreuses contraintes liées d'une part au processus de fabrication et d'autre part aux caractéristiques des équipements utilisés (ces contraintes et leur modélisation sont présentées dans la Sect. 2). Ceci rend ce problème de configuration des lignes d'usinage difficile. Mais étant donné que le nombre de postes de travail et de boîtiers détermine le coût de la ligne, il est impératif de rechercher une bonne configuration, et si possible une configuration optimale, minimisant ce nombre (et donc le coût de la ligne). Dans cet article, nous présentons un nouvel algorithme pour la résolution de ce problème et nous étudions ses performances.

Le reste de l'article est organisé de la manière suivante. Dans la Section 1, nous positionnons le problème traité par rapport aux travaux existants dans la littérature. Puis, dans la Section 2, nous proposons une formulation mathématique du problème d'optimisation. Dans la Section 3, nous présentons notre algorithme

de résolution. Cet algorithme a été appelé « l'algorithme mixte », car il fait appel à deux méthodes, exacte et heuristique, à tour de rôle. La Section 4 est consacrée à l'analyse des performances de ce nouvel algorithme sur un échantillon de problèmes choisis. Enfin, les conclusions et les perspectives de ce travail font l'objet de la Section 5.

1. État de l'art

Le problème d'affectation des opérations aux postes de travail sous contraintes de précédence et de temps de cycle a été initialement formulé pour des lignes d'assemblage. Le premier modèle mathématique de ce problème a été publié par Salveson en 1955 [21]. Plus tard, en 1986, Baybars [1] a donné, à ce modèle simplifié, le nom *Simple Assembly Line Balancing Problem* (SALBP). Il a été démontré que le SALBP est un problème NP-difficile [24].

Les modèles mathématiques et les méthodes de résolution proposés dans la littérature pour le SALBP ont été exposés dans plusieurs états de l'art : les méthodes exactes ont été analysées dans [1], des comparaisons des performances de méthodes approchées ont été faites dans [3,19,23]. Plus récemment, un large aperçu de méthodes de résolution dédiées au SALBP, d'algorithmes de calcul des bornes inférieures, de méthodes de prétraitement et de règles de dominance, a été donné dans [22].

À cause de ses hypothèses simplificatrices, le SALBP n'exprime pas toute la réalité industrielle [13]. Plusieurs études ont été menées dans le but d'introduire des hypothèses plus générales. N'importe quel problème de type ALBP ayant au moins une hypothèse moins restrictive par rapport au SALBP a été nommé dans la littérature *Generalized Assembly Line Balancing Problem* (GALBP).

Les publications présentant l'état de l'art sur GALBP [2,13,20] montrent l'évolution des hypothèses et des approches de résolution proposées.

Le problème d'optimisation de la configuration des lignes d'usinage étudié dans cet article est différent des problèmes GALBP connus dans la littérature. Une comparaison détaillée a été faite dans [14]. Nous ne mentionnons ci-dessous que les points les plus importants :

- L'utilisation des boîtiers multibroches activés de façon séquentielle fait que certaines opérations sont exécutées simultanément, d'autres séquentiellement.
- Les temps d'opérations ne sont pas connus avant l'optimisation. Des fonctions spécifiques non linéaires doivent être utilisées pour les calculer pour chaque affectation d'opérations, car le temps d'usinage d'un boîtier (et donc le temps d'exécution de chacune de ses opérations) dépend de l'ensemble des opérations qui lui sont assignées et de leurs paramètres.
- Il est possible d'utiliser des outils tranchants étagés ce qui permet l'exécution simultanée des opérations liées par une contrainte de précédence, d'où la nécessite d'une modélisation adaptée des contraintes de précédence.

- L'affectation des opérations doit se faire à deux niveaux : aux blocs et aux postes ; il est impossible de considérer ces affectations séparément sans prendre le risque de perdre l'optimalité.

Historiquement, la première formulation du problème d'optimisation de la configuration des lignes d'usinage à boîtiers multibroches a été présentée lors de la conférence ETFA par Dolgui *et al.* [9] en 1999. Par analogie avec SALBP, ce problème a été appelé TLBP (Transfer Line Balancing Problem). Dans un premier temps, deux méthodes exactes ont été développées pour sa résolution. La première s'appuie sur la transformation du problème initial en un problème de recherche du plus court chemin sous contraintes dans un graphe spécialement construit [9]. Dans cette approche, la difficulté principale réside dans la construction de ce graphe [10,11]. La deuxième approche utilise la modélisation du problème sous forme d'un programme linéaire en variables mixtes (MIP pour Mixed Integer Programming) en vue de le résoudre à l'aide d'un logiciel d'optimisation, en l'occurrence Cplex d'ILOG [8,12].

L'étude des performances de ces deux méthodes a montré qu'elles sont capables de résoudre à l'optimalité, avec un temps de calcul raisonnable, uniquement les problèmes de petite taille ou de taille moyenne (moins de 80 opérations). Ces résultats ne sont pas inattendus, étant donné que le problème est NP-difficile [10]. Il est nécessaire alors de recourir à des méthodes approchées pour la résolution des instances de taille plus grande.

Quelques heuristiques ont été proposées dans [6–8,12], dont la plus performante à ce jour, selon notre étude présentée dans [15], est l'heuristique FSIC (First Satisfy Inclusion Constraints). Elle a la même structure que celle des heuristiques « orientées postes de travail » basées sur une règle de priorité, développées pour la résolution du SALBP. Cependant, l'heuristique FSIC choisit une opération de la liste de candidats non pas en fonction d'une priorité, mais de manière aléatoire. De plus, les procédures de la construction de la liste de candidats et de l'affectation de l'opération sélectionnée tiennent compte des contraintes d'exclusion et d'inclusion (ignorées par la formulation du SALBP), ainsi que de la nécessité d'affecter les opérations non seulement aux postes de travail, mais aussi de les regrouper en blocs.

Une première tentative de mettre en place une méthode d'amélioration des solutions fournies par l'heuristique FSIC a été faite dans [17]. Dans l'article présent, nous approfondissons cette approche et proposons de nouvelles techniques ainsi qu'une étude des performances de ce nouvel algorithme sur un ensemble d'instances ayant la même structure des données que les problèmes industriels.

2. Formulation du problème d'optimisation

Le problème de configuration des lignes d'usinage à boîtiers multibroches, étudié dans cet article, consiste à répartir les opérations d'un ensemble **N** à des postes de travail en les regroupant en même temps en blocs.

Une solution faisable est caractérisée par :

- le nombre m de postes de travail ;
- l'affectation des opérations de l'ensemble $\mathbf{N}$ aux m postes de travail, l'ensemble des opérations affectées au poste k est noté N_k ;
- le nombre r_k de blocs du poste k ; le nombre total de blocs sur la ligne est $n = \sum_{k=1}^{m} r_k$;
- la répartition des opérations de chaque ensemble N_k entre les r_k blocs : N_{k1}, N_{k2}, ..., N_{kr_k}, $k = 1, 2, ..., m$.

Avec ces notations, une solution faisable peut être représentée par une collection $S = \{\{N_{11}, ..., N_{1r_1}\}, ..., \{N_{m1}, ..., N_{mr_m}\}\}$. L'objectif est de trouver la collection S_{opt} correspondant à la solution optimale, c'est-à-dire de minimiser la fonction objectif (1) qui représente une estimation du coût de la ligne à l'étape de la conception préliminaire. Pour estimer ce coût, nous utilisons les coefficients C_1 et C_2 qui sont les coûts relatifs d'un poste de travail et d'un boîtier, respectivement.

$$\text{Minimiser } C(S) = C_1 m + C_2 \sum_{k=1}^{m} r_k. \tag{1}$$

Une solution faisable doit respecter l'ensemble des contraintes ci-dessous.

Toutes les opérations de l'ensemble $\mathbf{N}$ doivent être affectées :

$$\bigcup_{k=1}^{m} \bigcup_{r=1}^{r_k} N_{kr} = \mathbf{N}. \tag{2}$$

Chaque opération de l'ensemble $\mathbf{N}$ doit être affectée une seule fois :

$$\begin{gathered} N_{k'r'} \cap N_{k''r''} = \emptyset, \\ \forall\, k', k'' = 1, ..., m; r' = 1, ..., r_{k'}; r'' = 1, ..., r_{k''}, \text{ où } (k', r') \neq (k'', r''). \end{gathered} \tag{3}$$

L'équation (4) définit les contraintes de précédence entre les opérations, où $\text{Pred}(j)$ est l'ensemble des prédécesseurs directs de l'opération $j \in \mathbf{N}$. L'opération $i \in \mathbf{N}$ appartient à l'ensemble $\text{Pred}(j)$ si et seulement si l'opération j ne peut pas être affectée à un bloc précédant le bloc où se trouve l'opération i.

$$(k' - 1)n_0 + r' \leq (k'' - 1)n_0 + r'', \forall\, i \in \text{Pred}(j), i \in N_{k'r'}, j \in N_{r''k''}. \tag{4}$$

L'équation (5) traduit les contraintes d'inclusion au niveau des postes de travail imposant que certains sous-ensembles d'opérations soient réalisés sur le même poste de travail. Pour les modéliser, nous utilisons la collection I^p d'ensembles $e \subset \mathbf{N}$ où toutes les opérations de chaque ensemble e doivent être affectées au même poste de travail.

$$(N_k \cap e) \in \{\emptyset, e\}, \text{ pour } \forall\, e \in I^p \text{ et } k = 1, ..., m. \tag{5}$$

Les équations (6) et (7) représentent les contraintes d'exclusion relatives aux blocs (6) et aux postes de travail (7). Ces contraintes interdisent l'exécution de

certains sous-ensembles d'opérations sur le même poste de travail ou par le même boîtier. Pour les modéliser, nous utilisons les collections E^p et E^b composées des ensembles $e \subset \mathbf{N}$ où toutes les opérations de chaque ensemble $e \in E^p$ ne peuvent pas être affectées au même poste de travail et toutes les opérations de chaque ensemble $e \in E^b$ ne peuvent pas être groupées dans le même bloc.

$$e \not\subset N_{kr}, \text{pour } \forall\, e \in E^b \text{ et } k = 1, ..., m; r = 1, ..., r_k \tag{6}$$

$$e \not\subset N_k, \text{pour } \forall\, e \in E^p \text{ et } k = 1, ..., m. \tag{7}$$

Si un ensemble $e \in E^p$ (ou $e \in E^b$) est constitué de plus de deux opérations, alors seule l'affectation de toutes les opérations de cet ensemble au même poste (ou au même bloc) est interdite. Si au moins une seule opération d'un tel ensemble e est affectée à un poste différent (à un bloc différent) des autres, la contrainte d'exclusion correspondante est respectée. Pour interdire l'affectation, par exemple, de n'importe quelle combinaison des opérations i, j, k au même poste de travail, il est nécessaire d'utiliser les contraintes suivantes : $\{i, j\}, \{i, k\}, \{j, k\} \in E^p$ et non $\{i, j, k\} \in E^p$.

L'équation (8) assure que le nombre de postes de travail ne dépasse pas le nombre maximum autorisé, désigné par m_0.

$$m \leq m_0. \tag{8}$$

L'équation (9) interdit le montage de plus de n_0 boîtiers sur un poste de travail.

$$r_k \leq n_0, \text{ pour } k = 1, ..., m. \tag{9}$$

Chaque opération $i \in \mathbf{N}$ est caractérisée par les deux paramètres suivants : la longueur de la course de l'outil λ_i et l'avance par minute υ_i. Ces paramètres sont analysés pour déterminer les valeurs des paramètres $\lambda(N)$ et $\upsilon(N)$ d'un bloc N. Les paramètres $\lambda(N)$ et $\upsilon(N)$ seront employés par le boîtier multibroche correspondant (celui qui exécute l'ensemble N). Ils sont utilisés pour calculer le temps d'exécution du bloc, désigné par $t^b(N)$.

$$\lambda(N_{kr}) = \max\{\lambda_i | i \in N_{kr}\} \tag{10}$$

$$\upsilon(N_{kr}) = \min\{\upsilon_i | i \in N_{kr}\} \tag{11}$$

$$t^b(N_{kr}) = \frac{\lambda(N_{kr})}{\upsilon(N_{kr})} + \tau^b. \tag{12}$$

Comme les boîtiers multibroches d'un poste de travail sont activés de façon séquentielle, le temps d'usinage sur le poste de travail k, désigné par $t^p(N_k)$, est égal à la somme des temps d'usinage de tous ses blocs.

$$t^p(N_k) = \sum_{r=1}^{r_k} t^b(N_{kr}) + \tau^p. \tag{13}$$

Les coefficients τ^p et τ^b permettent de tenir compte des temps auxiliaires au niveau d'un poste et d'un bloc, respectivement.

Pour respecter la contrainte de temps de cycle, la condition suivante doit être valide :

$$t^p(N_k) \leq T_0, k = 1, ..., m \tag{14}$$

où T_0 est le temps de cycle objectif.

Dans la section suivante, nous présentons l'algorithme mixte pour la résolution du problème introduit.

3. Algorithme mixte

3.1. Approche générale

L'algorithme mixte est constitué des trois étapes suivantes : une résolution heuristique, une décomposition en sous-problèmes et une résolution exacte des ces derniers.

À l'étape de la résolution approchée, nous employons l'heuristique FSIC [7,12]. Moyennant cette heuristique, une solution admissible du problème initial est obtenue. Une telle solution peut être représentée comme une séquence S_{heur} avec m_{heur} éléments, où chaque élément contient l'ensemble des opérations affectées à un poste de travail. Bien évidemment, toutes les contraintes de précédence et de compatibilité sont respectées pour chaque élément de la séquence et entre eux. Ensuite, cette solution S_{heur} est utilisée pour décomposer l'ensemble $\mathbf{N}$ en sous-ensembles indépendants. Ceci est obtenu par un découpage de la séquence S_{heur} en w sous-séquences, $S_{\text{heur}} = \{N^1, ..., N^u, ..., N^w\}$, en respectant les conditions suivantes :

- chaque sous-ensemble N^u contient les opérations affectées à un nombre entier de postes de travail successifs, c'est-à-dire :
  $N^u = \bigcup_{k=k_1}^{k_2} N_k, 1 \leq k_1 \leq k_2 \leq m_{\text{heur}}$;
- toutes les opérations de chaque poste de travail sont dans le même sous-ensemble et dans un seul, c'est-à-dire si $N_k \subset N^u$, alors $N_k \not\subset N^i, i \in \{1, ..., w\} \backslash u$;
- l'union de tous les sous-ensembles N^u donne l'ensemble $\mathbf{N}$, c'est-à-dire :
  $\bigcup_{u=1}^{w} N^u = \mathbf{N}$.

Ensuite, à partir des sous-ensembles N^u, obtenus par le découpage de la séquence S_{heur}, nous formons de nouveaux problèmes, dit sous-problèmes SP_u, qui sont de même type que le problème initial mais de taille plus petite. Nous avons élaboré deux techniques de formation de ces sous-problèmes :

- sans intersection (indépendante) : chaque sous-problème peut être résolu indépendamment des autres (les sous-problèmes n'ont pas d'opérations en commun), la solution du problème initial est alors formée par la concaténation des solutions des sous-problèmes ;

- avec agrégation (agrégée) : les opérations qui étaient dans les sous-problèmes précédents sont également présentes dans le sous-problème courant, mais sous forme de macro-opérations (voir la Sect. 3.5). La solution du problème initial est alors le résultat de la résolution du dernier sous-problème.

Quelle que soit la technique de décomposition utilisée, chaque sous-problème est résolu par une méthode exacte : soit en utilisant le modèle MIP présenté dans [8], soit en appliquant la méthode de plus court chemin sous contraintes de [11].

Après la résolution exacte de tous les sous-problèmes, une itération de l'algorithme est terminée. Une nouvelle itération commence par le retour à l'étape de la résolution approchée, où une nouvelle solution admissible est obtenue, ensuite décomposée, et ainsi de suite. L'Algorithme 1 décrit le schéma global de l'approche proposée.

```
1  Initialiser C_min = ∞, IT_tot = 0, IT_nimp = 0
   répéter
2      Choisir une solution heuristique S_heur avec son coût C_heur
3      si C_heur < C_min alors
           C_min = C_heur, S_min = S_heur
       fin
4      Initialiser k = 1, u = 1
       tant que k < m_heur faire
5          Déterminer N^u et k_u
6          Générer un nouveau sous-problème SP_u à partir de N^u
           k = k + k_u, u = u + 1
7          Résoudre le sous-problème SP_u par une méthode exacte
       fin
8      Former la solution finale S_mix avec son coût C_mix à partir de u
       sous-problèmes résolus
       IT_tot = IT_tot + 1
9      si C_mix < C_min alors
           C_min = C_mix, S_min = S_mix, IT_nimp = 0
       sinon
           Incrémenter le compteur IT_nimp = IT_nimp + 1
       fin
   jusqu'à T_cur > T_res || IT_nimp > IT_nimp^aut || IT_tot > IT_tot^aut || C_min < C_stop
```

Algorithme 1: Schéma général de l'approche proposée.

Quand le temps de résolution alloué s'écoule ($T_{\text{cur}} > T_{\text{res}}$, où T_{cur} est le temps consommé pour la résolution, T_{res} est le temps alloué), nous choisissons la meilleure solution obtenue. Éventuellement, d'autres conditions d'arrêt peuvent être ajoutées, telles que l'obtention d'une solution avec un coût $C_{\min}$ inférieur à un seuil donné

C_{stop} ou le dépassement des valeurs maximales autorisées ($IT_{\text{tot}^{\text{aut}}}$ et $IT_{\text{nimp}^{\text{aut}}}$) pour le nombre d'itérations effectuées IT_{tot} ou pour le nombre d'itérations sans amélioration IT_{nimp}.

3.2. Heuristique utilisée

L'étude effectuée dans [15] a permis de conclure que la meilleure heuristique proposée pour la résolution du problème d'optimisation des lignes d'usinage à boîtiers multibroches est l'heuristique FSIC, publiée dans [7]. Ensuite, quelques extensions permettant d'améliorer ses performances ont été proposées dans [14,16]. Dans ce qui suit, nous décrivons la modification de l'algorithme FSIC que nous utilisons dans cet article pour la recherche de solutions faisables.

3.2.1. *Schéma général de l'heuristique*

L'heuristique FSIC est basée sur l'approche Monte Carlo. Ainsi lors de la construction d'une solution, un choix aléatoire est effectué s'il existe plusieurs décisions possibles. Le recours au facteur « hasard » peut amener à des solutions de bonne qualité ou même optimales, mais également à des solutions de mauvaise qualité, voire inadmissibles (à cause de la contrainte sur le nombre maximum de postes de travail). Pour cette raison, une itération effectuée par l'heuristique FSIC peut être soit « concluante » si elle donne une solution admissible pour le problème, soit « non concluante » si toutes les opérations n'ont pas été affectées et aucune action n'est possible car le nombre de postes de travail dépasse le nombre maximum autorisé. Au total, l'algorithme effectue IT_{tot} itérations, et plus ce nombre est grand plus il y a des chances de trouver une solution de bonne qualité. Pour essayer d'éviter la répétition des solutions générées, le paramètre *seed* du générateur de nombres pseudo-aléatoires est modifié au début de chaque itération. Les conditions d'arrêt de l'algorithme sont les suivantes :

- l'écoulement du temps disponible ($T_{\text{cur}}^{\text{heur}} > T_{\text{res}}^{\text{heur}}$, où $T_{\text{cur}}^{\text{heur}}$ est le temps consommé par l'heuristique, $T_{\text{res}}^{\text{heur}}$ est le budget du temps disponible pour la résolution approchée) ;
- l'obtention d'une solution avec un coût inférieur à un seuil donné $C_{\text{stop}}^{\text{heur}}$;
- le dépassement des valeurs maximales autorisées ($IT_{\text{tot\_aut}}^{\text{heur}}$ et $IT_{\text{nimp\_aut}}^{\text{heur}}$) pour le nombre d'itérations effectuées $IT_{\text{tot}}^{\text{heur}}$ ou pour le nombre d'itérations sans amélioration $IT_{\text{nimp}}^{\text{heur}}$.

Afin de construire une solution, l'heuristique FSIC fait appel aux trois opérateurs suivants : celui de construction d'une liste d'opérations candidates *CL* (cette liste regroupe toutes les opérations qui peuvent être affectées au moment de la décision), celui de sélection d'une opération de la liste *CL* (qui sera effectivement affectée) et, enfin, celui d'affectation de l'opération choisie. Le dernier opérateur entraîne également la modification de la liste des opérations non affectées. Dans ce qui suit, nous décrivons la réalisation de ces trois opérateurs dans la version de l'heuristique FSIC que nous utilisons ici.

3.2.2. *Construction de la liste CL*

Soit m le poste de travail courant et r_m le bloc courant de ce poste. Pour construire la liste *CL*, les opérations non affectées sont analysées. L'opération i est placée dans la liste *CL*, si les conditions suivantes sont respectées :

(1) Tous les prédécesseurs de l'opération i sont déjà affectés, soit si $j \in \mathrm{Pred}(i)$, alors $j \in \bigcup_{k=1}^{m-1} \bigcup_{r=1}^{r_k} N_{kr} \cup \bigcup_{r=1}^{r_m} N_{mr}$;

(2) L'affectation de l'opération i au poste courant m respecterait les contraintes de type E^p, soit $\forall e \in E^p$ tel que $i \in e : e \cap (N_m \cup \{i\}) \neq e$.

3.2.3. *Sélection d'une opération de la liste CL*

À chaque fois, une opération est choisie dans la liste *CL* de manière aléatoire. Pour cela, l'approche Monte-Carlo [4,5] est utilisée.

3.2.4. *Affectation de l'opération choisie*

Les contraintes d'inclusion imposent l'affectation de certains ensembles d'opérations au même poste de travail. Ceci doit être pris en compte lors de l'affectation des opérations, même si les opérations sont traitées une par une. Pour en tenir compte, l'algorithme FSIC (First Satisfy Inclusion Constraints) affecte les opérations liées par des contraintes d'inclusion en priorité, comme nous l'expliquons plus loin dans le texte.

Soit i l'opération choisie dans la liste *CL*. Si $\nexists\, e \in I^p$ tel que $i \in e$, alors l'algorithme essaie d'affecter l'opération i à partir du premier bloc ouvert. L'affectation de l'opération n'est admissible que si toutes les contraintes pour le bloc et le poste de travail correspondants sont respectées. S'il existe au moins une contrainte qui empêche l'affectation de l'opération i à un bloc, alors l'algorithme passe soit au bloc, soit au poste suivant, et ainsi de suite jusqu'à ce que soit l'opération i soit affectée, soit le bloc courant soit atteint. Dans le deuxième cas, si l'affectation de l'opération i même au bloc courant est impossible, alors un nouveau bloc est ouvert pour le poste courant. Si un nouveau bloc ne peut pas être créé à cause de la contrainte sur le temps de cycle ou à cause du nombre de blocs déjà ouverts, et si en même temps il est encore possible de créer un nouveau poste (le nombre de postes est inférieur à m_0), alors un nouveau poste est créé, sinon l'itération en cours est considérée comme non concluante et $C_{\mathrm{cur}}(S) = \infty$.

Si $\exists\, e \in I^p$ tel que $i \in e$, alors l'algorithme crée une liste complémentaire *AL*. Cette liste contient toutes les opérations dont l'affectation au même poste que l'opération i est nécessaire afin de respecter la contrainte e. Par conséquence, l'opération i est placée dans cette liste ainsi que toute opération j telle que $j \in e$, $j \neq i$. Puis, pour chaque opération j ajoutée, les contraintes de précédence sont analysées et tous ses prédécesseurs non affectés sont également ajoutés à la liste *AL*. Ensuite, pour chaque nouvelle opération ajoutée, à son tour, les contraintes d'inclusion et de précédence sont analysées, et ainsi de suite, c'est-à-dire de manière récursive.

La liste *AL* est composée des deux parties suivantes : la première avec les opérations sans contrainte d'inclusion, c'est-à-dire les prédécesseurs non affectés, et la deuxième avec les opérations liées par une contrainte d'inclusion. Les opérations de la première partie sont traitées en priorité.

La procédure d'affectation de toute opération de la liste *AL* est la même que l'affectation d'une opération pour laquelle $\nexists\, e \in I^p$ tel que $i \in e$, à la seule exception près : si l'affectation d'une opération de la liste *AL* au même poste que les autres s'avère impossible, alors l'algorithme annule toutes les affectations d'opérations effectuées à partir de cette liste.

3.3. Sélection d'une solution de départ

Comme nous l'avons déjà indiqué, les choix aléatoires utilisés dans l'heuristique FSIC font qu'en l'utilisant nous pouvons obtenir une solution de mauvaise qualité à une des itérations. Même si, théoriquement, il est possible d'obtenir une solution finale de bonne qualité à partir d'une mauvaise solution de départ, cela est très peu probable avec les techniques utilisées de décomposition et d'amélioration locale des solutions. En effet, le nombre de réaffectations possibles pour chaque opération est limité par le sous-ensemble auquel elle appartient. Dans ces conditions, l'optimum global ne peut pas être atteint à partir de n'importe quelle solution de départ. L'amélioration de la valeur de la fonction objectif est encore plus limitée à cause de nombreuses contraintes et du fait que beaucoup de solutions voisines sont en fait des permutations de la même solution, voir [14]. Pour la sélection d'une solution de départ qui est faite à l'étape 2 de l'Algorithme 1, et afin de ne garder que des solutions prometteuses, nous avons développé deux techniques suivantes.

La première technique consiste à lancer l'heuristique tant qu'une solution admissible ne soit pas trouvée (le plus souvent, une seule itération suffit). Ensuite, le coût de cette solution est comparé avec le coût de la meilleure solution heuristique connue $C_{\text{heur\_min}}$. Si la différence ne dépasse pas un écart autorisé δ, une telle solution est acceptée pour la décomposition, sinon elle est rejetée.

La seconde technique propose de lancer l'heuristique pendant un certain temps $T_{\text{res}}^{\text{heur}}$, ensuite la **meilleure solution obtenue** est comparée avec $C_{\text{heur\_min}}$ et si la différence ne dépasse pas l'écart δ, une telle solution est acceptée pour l'étape suivante. Cette technique doit normalement fournir des solutions de départ avec un coût plus petit, en contrepartie, elle nécessite plus de temps à cette étape, ce qui diminue le temps restant pour la résolution des sous-problèmes de manière exacte après la décomposition.

3.4. Décomposition d'une solution heuristique

Le problème considéré étant NP-difficile, le temps nécessaire pour la résolution exacte croit exponentiellement en fonction de la taille du problème. Ceci soulève la question suivante : comment découper le problème initial en sous-problèmes pour que la méthode exacte utilisée soit capable de les résoudre tous dans la limite du temps alloué. Étant donné que nous ne disposons pas de critère permettant de

déterminer a priori comment décomposer une solution heuristique afin de s'approcher le plus de l'optimum global, nous employons une décomposition stochastique.

Plus précisément, pour déterminer le nombre de postes k_u dont les opérations constitueront un sous-problème, nous choisissons au hasard un nombre entier dans l'intervalle [1 .. $MaxSt$], où le paramètre $MaxSt$ est utilisé pour limiter la taille des sous-problèmes : il donne le nombre maximum de postes qui peuvent être inclus dans un sous-problème. Ainsi il est possible d'obtenir différents découpages en sous-séquences de la même solution heuristique.

De plus, étant donné que le temps de résolution du sous-problème u dépend de la cardinalité de l'ensemble N^u, nous utilisons le paramètre $MaxOp$ pour contrôler le nombre d'opérations d'un sous-problème. Le processus de formation d'une sous-séquence est arrêté si le nombre d'opérations qu'elle comprend dépasse $MaxOp$. Notons que nous ne pouvons pas toujours garantir que le nombre d'opérations pour aucun sous-problème ne dépasse $MaxOp$, car si le nombre d'opérations affectées à un poste de travail est supérieur à $MaxOp$, nous sommes obligés de les prendre toutes.

Il est nécessaire de contrôler la taille de sous-problèmes pour les raisons suivantes. Plus la taille de sous-problèmes est petite, plus le temps de résolution est raisonnable, mais moins de chances existent d'améliorer la solution de départ et de s'approcher de l'optimum global. Plus la taille de sous-problèmes est importante, plus de temps est consommé à l'étape de résolution des sous-problèmes et moins de solutions de départ peuvent être explorées.

Le choix des valeurs pour les paramètres $MaxOp$ et $MaxSt$ doit être fait à l'étape d'initialisation de l'algorithme en fonction des paramètres suivants : la taille du problème initial, le temps disponible pour la résolution, la technique de formation de sous-problèmes et la méthode exacte utilisée.

Le caractère stochastique de la détermination de la taille de chaque sous-séquence permet d'obtenir différents découpages en utilisant les mêmes valeurs des paramètres $MaxOp$ et $MaxSt$. Ainsi il serait possible d'effectuer plusieurs découpages d'une seule solution de départ et, en conséquence, de résoudre différents sous-problèmes en vue de mieux exploiter le voisinage d'une solution heuristique. Néanmoins, les tests préliminaires ont montré que le même effet était atteint grâce à l'augmentation du nombre de solutions heuristiques de départ. De plus, en tandem avec la technique de sélection des solutions de départ, un seul découpage d'une solution permettait d'obtenir des solutions finales de meilleure qualité. Cela peut être expliqué par le fait que dans ce cas, les mauvaises solutions de départ n'ont été traitées qu'une seule fois. Ainsi le paramètre déterminant le nombre d'essais pour le découpage d'une solution heuristique a été finalement abandonné et l'algorithme effectue un seul découpage de chaque solution heuristique.

3.5. Formation de sous-problèmes

Nous avons mis en œuvre deux techniques de formation de sous-problèmes, à savoir : indépendante et agrégée. Nous détaillons ici leurs principes.

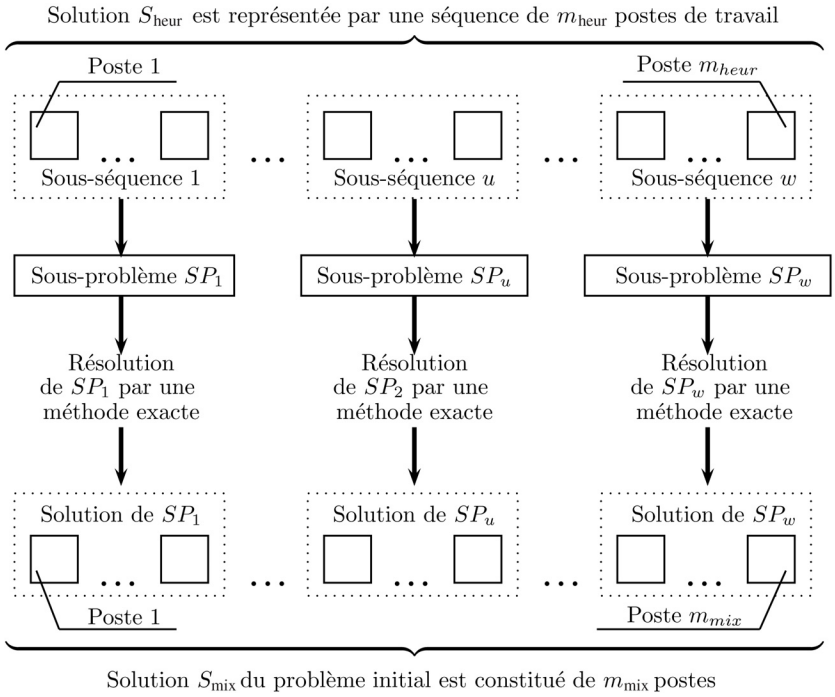

FIGURE 1. Formation des sous-problèmes : technique indépendante.

La technique indépendante repose sur la résolution de sous-problèmes de façon séparée. Ceci est possible du fait que les sous-séquences obtenues après le découpage de la solution heuristique ne contiennent pas d'opérations communes et que les contraintes de précédence et d'incompatibilité sont respectées entre les sous-ensembles d'opérations N^1, ..., N^u, ..., N^w. Le sous-problème SP_u concerne alors l'ensemble d'opérations N^u avec les contraintes (2)–(9) modifiées. La modification consiste à remplacer l'ensemble $\mathbf{N}$ par l'ensemble N^u et à supprimer les contraintes contenant des opérations de l'ensemble $\mathbf{N}\backslash N^u$. Nous remplaçons également m_0 pour SP_u par $\lfloor C_u^{\text{heur}}/(C_1 + C_2)\rfloor$, où C_u^{heur} est le coût de la sous-séquence u dans la solution heuristique S_{heur}.

La solution finale du problème initial est obtenue en remplaçant les sous-séquences de S_{heur} par les solutions exactes des sous-problèmes correspondants. Le schéma de la technique indépendante de formation des sous-problèmes est présenté dans la Figure 1.

La technique agrégée, quant à elle, construit une solution finale en augmentant progressivement la taille du sous-problème courant de telle façon que les opérations qui ont donné lieu aux sous-problèmes antérieurs soient également incluses dans le sous-problème courant mais en tant que macro-opérations. Pour cela, après avoir résolu le sous-problème SP_{u-1}, nous remplaçons les blocs obtenus dans la solution S_{u-1} par des macro-opérations qui seront ajoutées dans le sous-problème SP_u. Avec cette stratégie, il est possible d'affecter à des postes de travail

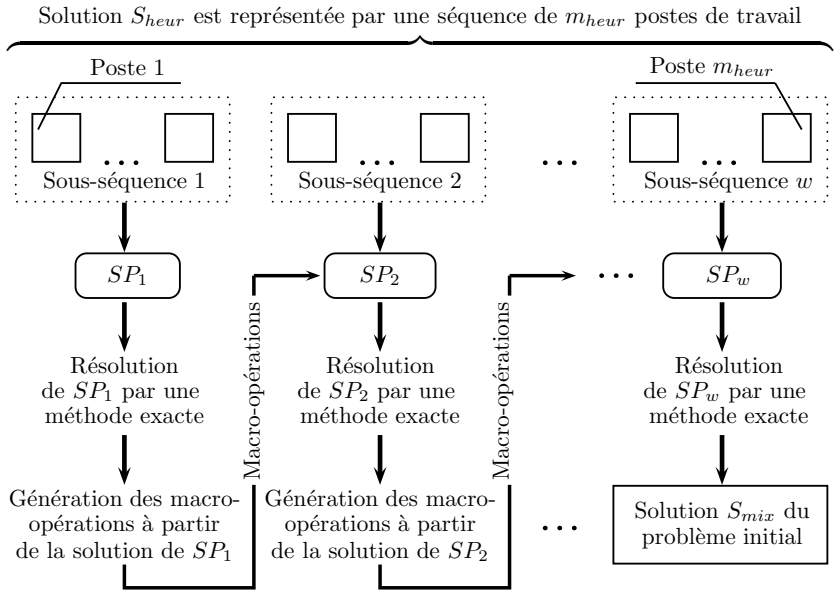

FIGURE 2. Formation des sous-problèmes : technique agrégée.

et à des blocs déjà construits des opérations qui n'étaient pas traitées au moment de la formation de ces blocs et postes, ainsi que d'assigner les blocs existants à d'autres postes de travail que ceux auxquels ils ont été assignés auparavant. La solution du problème initial est le résultat de la résolution du dernier sous-problème. Cette technique est présentée dans la Figure 2.

Nous remplaçons chaque ensemble d'opérations N_{kr} affecté au r-éme bloc du poste k dans la solution S_{u-1} par une macro-opération. Les paramètres d'usinage de cette macro-opération sont calculés de façon suivante : $\lambda_{(\mathrm{MO})} = \lambda(N_{kr})$, $\upsilon_{(\mathrm{MO})} = \upsilon(N_{kr})$. Après avoir créé toutes les macro-opérations, nous ajustons les contraintes du problème initial de la manière suivante : si N_{kr} est un élément de I^p, alors cet élément peut être supprimé, car la contrainte correspondante est déjà respectée. Autrement, s'il existe des opérations appartenant à la fois à l'ensemble N_{kr} et à des ensembles des collections E^b, E^p ou I^p (mais N_{kr} n'est pas un élément entier de I^p), alors nous les remplaçons par les macro-opérations correspondantes.

Les macro-opérations sont ajoutées dans l'ensemble N^u et ensuite le sous-problème SP_u est formé. Notons que nous devons garder les informations concernant la composition des macro-opérations, car pour reconstituer la solution du problème initial après avoir obtenu la solution du dernier sous-problème, il faudra remplacer les macro-opérations dans cette solution par les opérations d'origine.

3.6. Résolution des sous-problèmes

Suite au découpage aléatoire du problème initial, nous pouvons nous retrouver face à deux situations problématiques lors de la résolution d'un sous-problème :

(1) Malgré l'utilisation des paramètres $MaxOp$ et $MaxSt$, limitant la taille des sous-séquences, le sous-problème formé peut s'avérer « trop lourd » pour la méthode exacte utilisée à cause de sa structure particulière. Si nous tentons de le résoudre, nous risquerons de consommer tout le temps alloué rien que pour ce sous-problème et ne pas laisser le temps pour les autres sous-problèmes. Notons que le terme « trop lourd » dépend de la méthode exacte utilisée.
(2) Si la taille d'un sous-problème est « trop petite », alors, la solution heuristique déjà connue a une forte chance d'être optimale, et par conséquent, la résolution de ce sous-problème de manière exacte ne pourra pas l'améliorer. Par expérience, nous qualifions la taille d'un sous-problème comme « trop petite » si le sous-problème contient un seul poste de travail avec un nombre de blocs inférieur à 3, ou deux postes de travail avec un nombre de blocs inférieur à 2 chacun.

Alors, si lors de la résolution d'un sous-problème SP_u, nous apercevons que sa taille est « trop petite » ou le problème est « trop lourd », afin d'éviter les pertes du temps, nous n'appliquons pas la méthode exacte, mais nous copions simplement la sous-séquence u dans la solution S_{mix}.

De plus, nous calculons la borne inférieure du critère d'optimisation pour chaque sous-problème. Si jamais la valeur du critère de la solution heuristique est égale à la borne inférieure, alors il est inutile de résoudre ce sous-problème, car nous avons déjà une solution optimale.

Pour la résolution des sous-problèmes, nous avons utilisé deux méthodes exactes : la première utilisant la théorie des graphes [9,11] et la seconde basée sur la programmation linéaire en variables mixtes [8,12]. Dans le premier cas, nous transformons le problème initial en problème de recherche de plus court chemin dans un graphe spécialement construit. Dans le deuxième, nous utilisons son modèle sous forme d'un programme linéaire que nous résolvons par le biais du logiciel d'optimisation ILOG Cplex.

4. Tests numériques

4.1. Les jeux de données

Dans la littérature de l'optimisation combinatoire, les instances utilisées pour des tests numériques sont soit générées aléatoirement, soit prises à partir des exemples réels des problèmes traités. Dans le premier cas, il est possible d'étudier différents cas possibles, mais il est difficile de juger des performances des algorithmes face à des problèmes réels. Tandis que dans le deuxième cas, une idée sur ses performances est donnée, mais souvent, le nombre de tests effectués étant limité, il est impossible de faire des conclusions générales. Pour étudier les performances de l'approche proposée dans le cadre de résolution de problèmes industriels, nous avons développé un algorithme de génération des instances qui permet de tenir compte de la structure des données dans les systèmes réels. En fait, la fabrication

Tableau 1. Paramètres des séries de tests générées.

Série	n_{ent}	$\lvert N_{\min}\rvert$	$\lvert N_{\max}\rvert$	$\lvert N_{\text{moy}}\rvert$	T_{res}
1	10	29	47	38	150
2	20	46	92	55	300
3	30	80	127	84	450
4	40	115	158	141	600

d'une pièce consiste à réaliser un ensemble d'entités. Chaque type d'entité est caractérisé par un ensemble d'opérations d'usinage (fraisage, perçage, alésage, etc.) et par des contraintes entre ces opérations. En analysant les processus d'usinage chez nos partenaires industriels, nous avons élaboré une bibliothèque des entités les plus souvent utilisées. Parmi ces entités, nous pouvons citer, par exemple, trou cylindrique, trou conique avec bossage, trou conique ayant un filet anglais, etc., pour plus de détails voir [9].

Lors de la génération des instances, le nombre d'entités à usiner est un paramètre, mais leurs types et leurs emplacements sur la pièce sont choisis au hasard. Les caractéristiques de chaque entité sont dans la bibliothèque élaborée : l'ensemble d'opérations à réaliser, les paramètres d'usinage pour chaque opération, les contraintes, etc.

Nous avons créé 4 séries de tests qui se distinguent par le nombre d'entités utilisées. Chaque série comporte 50 instances distinctes. Donc, au total, 200 instances ont été testées. De manière générale, plus la pièce à fabriquer comporte d'entités, plus la taille de l'ensemble $\mathbf{N}$ est grande [14]. Mais comme nous l'avons déjà dit, lors de la génération de toute instance, le type de chaque entité et la face de la pièce à laquelle cette entité appartient sont choisis au hasard. Ainsi, dans la même série, nous avons des instances avec un nombre d'opérations et de contraintes différents.

Le tableau 1 fournit les paramètres des séries générées. La colonne « Série » comporte le numéro de la série, n_{ent} est le nombre d'entités utilisées lors de la génération des instances, $\lvert N_{\min}\rvert, \lvert N_{\max}\rvert, \lvert N_{\text{moy}}\rvert$ sont les valeurs minimum, maximum et moyenne du nombre d'opérations pour les instances de la série correspondante. La colonne « T_{res} » comporte le temps alloué pour la résolution des instances.

4.2. Les versions de l'approche mixte testées

Notre but était de tester les différentes variantes de l'approche mixte proposée en les comparant avec l'heuristique FSIC utilisée individuellement. De manière générale, une variante de l'approche mixte est définie par le choix des paramètres suivants :

- l'heuristique utilisée pour la construction des solutions de départ ;
- la technique de sélection d'une solution de départ ;

TABLEAU 2. Paramètres de contrôle.

Paramètre	Choix 1	Choix 2
Technique de sélection d'une solution de départ	δ	$\delta\&T$
Technique de formation de sous-problèmes	I	A
Méthode pour la résolution des sous-problèmes	G	MIP

- les valeurs des paramètres $MaxOp$ et $MaxSt$ limitant la taille des sous-problèmes ;
- la technique de formation des sous-problèmes ;
- la méthode exacte utilisée pour la résolution des sous-problèmes.

Dans les expérimentations que nous présentons ici, toutes les versions de l'approche proposée utilisent la même variante de l'heuristique FSIC, celle qui a été la plus performante dans nos études préalables, voir [14]. Elle a été décrite dans la Section 3.2.

Les paramètres $MaxOp$ et $MaxSt$ ont également les mêmes valeurs pour toutes les versions testées, à savoir : $MaxOp = 30$, $MaxSt = 3$. Le choix de ces valeurs a été réalisé à la base des tests faits dans [17].

Nous avons donc étudié l'impact des paramètres présentés dans le Tableau 2, à savoir : la technique de sélection d'une solution de départ (« δ » si seul le paramètre δ est utilisé ou « $\delta\&T$ » si les deux paramètres δ et T_{heur} sont utilisés) ; la technique de formation de sous-problèmes (agrégée « A » ou indépendante « I ») ; et la méthode exacte utilisée (programmation linéaire en variables mixtes « MIP » ou approche par graphe « G »).

Pour présenter un jeu de paramètres, nous utilisons une notation constituée de trois champs sous forme x| y| z. Par exemple, la version δ| I| G fait appel à la technique de sélection d'une solution de départ utilisant le paramètre δ, et à la technique indépendante de formation des sous-problèmes qui sont résolus moyennant l'approche par graphe.

4.3. RÉSULTATS DES TESTS

Les calculs ont été effectués sur un PC Pentium IV, 3 GHz avec 512 Mo de RAM.

Tous les exemples ont été résolus par 5 versions de l'algorithme mixte et par l'heuristique FSIC. Les résultats fournis par ces méthodes ont été comparés afin de trouver la meilleure solution pour chaque problème. Ensuite, les écarts par rapport au meilleur résultat ont été calculés, ils sont représentés dans le Tableau 3, où Δ_{min}, Δ_{moy} et Δ_{max} sont respectivement l'écart minimum, moyen et maximum en pourcentage. *PMS* est le Pourcentage des Meilleures Solutions obtenues par chaque méthode testée.

En comparant les résultats obtenus, nous pouvons constater que les deux versions suivantes : δ| A| G et $\delta\&T$| A| G se démarquent des autres, avec un léger

Tableau 3. Résultats des tests.

Série	Caract.	FSIC	$\delta\vert$ I$\vert$ G	$\delta\vert$ A$\vert$ G	$\delta\&T\vert$ A$\vert$ G	$\delta\vert$ A$\vert$ MIP	$\delta\&T\vert$ A$\vert$ MIP
1	$\Delta_{\min}$	0	0	0	0	0	0
	$\Delta_{\max}$	5,4	3,7	0	0	15	10,9
	Δ_{moy}	0,49	0,35	0	0	4,86	2,63
	PMS %	90	94	100	100	34	56
2	$\Delta_{\min}$	0	0	0	0	0	0
	$\Delta_{\max}$	7,7	4,5	3,92	3,92	8,2	5,88
	Δ_{moy}	3,1	1,8	0,58	0,81	3,74	2,44
	PMS %	18	34	78	70	34	22
3	$\Delta_{\min}$	0	0	0	0	0	0
	$\Delta_{\max}$	7,69	3,5	3,85	1,87	4,22	5,1
	Δ_{moy}	3,87	1,2	0,74	0,13	1,36	1,72
	PMS %	6	42	62	90	34	28
4	$\Delta_{\min}$	0	0	0	0	0	0
	$\Delta_{\max}$	10,08	4,2	3,74	1,56	3,74	3,74
	Δ_{moy}	3,58	1,2	0,82	0,26	1,32	0,84
	PMS %	14	36	50	76	22	46

avantage pour $\delta\&T\vert$ A$\vert$ G. Par ailleurs, elles sont beaucoup plus performantes que l'heuristique FSIC toute seule.

En analysant l'impact de chaque technique sur la qualité des résultats obtenus pour les problèmes testés, nous pouvons tirer les conclusions suivantes :

- les versions comportant la technique agrégée de formation des sous-problèmes montrent de meilleures performances que celles utilisant la technique indépendante ;
- les versions utilisant l'approche par graphe pour la résolution des sous-problèmes ont été capables de fournir des solutions de meilleure qualité par rapport à celles faisant appel à la programmation linéaire en variables mixtes ;
- aucune technique de sélection des solutions heuristiques ne surpasse l'autre, l'utilisation de toute technique proposée peut conduire à des solutions de bonne qualité.

Nous pouvons également voir que pour la Série 1 avec $n_{\text{ent}} = 10$, toutes les versions qui utilisent l'approche par graphe pour la résolution des sous-problèmes ont été capables de trouver des solutions de bonne qualité ($PMS = 94\%$ pour $\delta\vert$ I$\vert$ G, et $PMS = 100\%$ pour $\delta\vert$ A$\vert$ G et $\delta\&T\vert$ A$\vert$ G). À noter également que pour les séries comportant la grande valeur de n_{ent}, l'avantage de la technique agrégée de formation des sous-problèmes sur la technique indépendante est évident.

5. Conclusions

Nous avons étudié le problème d'optimisation de la configuration des lignes d'usinage à boîtiers multibroches. Ce problème consiste à assigner les opérations à des postes de travail et à des boîtiers de sorte à minimiser le coût de la ligne en assurant le taux de productivité requis et en respectant toutes les autres contraintes connues. C'est un nouveau problème d'équilibrage des lignes de production qui a de nombreuses spécificités par rapport aux problèmes étudiés dans la littérature.

À la différence de nos travaux antérieurs où des méthodes heuristiques et exactes ont été étudiées séparément, dans cet article nous avons développé une approche mixte. Cette approche comporte les trois étapes suivantes : la résolution heuristique pour la génération des solutions initiales, la décomposition en sous-problèmes et, enfin, la résolution exacte des ces derniers. La méthode proposée appartient à la famille des méthodes par grands voisinages : la solution obtenue est partiellement relâchée et le problème engendré est ensuite résolu de manière optimale. Ce type de méthodes est de plus en plus employé pour aborder les problèmes réels.

Plusieurs techniques nouvelles ont été élaborées pour réaliser cette approche, notamment deux techniques de sélection des solutions heuristiques à décomposer, deux techniques de formation des sous-problèmes, ainsi que les algorithmes permettant d'intégrer dans cette approche deux méthodes exactes : une basée sur la théorie des graphes et une autre sur la programmation linéaire en variables mixtes.

Les résultats numériques montrent que la décomposition stochastique et la résolution exacte des sous-problèmes permet d'améliorer les solutions fournies par l'heuristique, et par conséquent, l'approche mixte, proposée dans cet article, est relativement efficace pour la résolution des problèmes de grande taille. L'amélioration dépend des caractéristiques du problème ainsi que des paramètres de l'algorithme et du temps de résolution alloué.

Rappelons que l'algorithme peut utiliser des heuristiques et des méthodes exactes quelconques, alors la qualité des résultats obtenus dépend de la qualité des méthodes utilisées. Il serait intéressant d'étudier une possibilité d'arrêter la méthode exacte utilisée dès qu'un seuil d'amélioration est atteint pour un sous-problème (c'est-à-dire sans aller jusqu'à la démonstration de l'optimalité). En perspectives, nous envisageons également le développement d'autres heuristiques pour la recherche de solutions faisables en privilégiant les heuristiques capables d'effectuer un apprentissage lors de la résolution du problème, notamment des méta-heuristiques de type GRASP ou des procédures de Backtracking.

Références

[1] I. Baybars, A survey of exact algorithms for the simple assembly line balancing. *Manage. Sci.* **32** (1986) 909–932.

[2] C. Becker and A. Scholl, A survey on problems and methods in generalized assembly line balancing. *Eur. J. Oper. Res.* **168** (2006) 694–715.

[3] F. Boctor, A multiple-rule heuristic for assembly line balancing. *J. Oper. Res. Soc.* **46** (1995) 62–69.

[4] H. Calos and A. Whitlock, *Monte Carlo Methods, Vol. 1*: *Basics*. John Wiley, New York (1986).

[5] K.M. Decker, The Monte Carlo method: Theory and application. *Comput. Methods Appl. Mech. Engrg.* **89** (1991) 463–483.
[6] A. Dolgui, B. Finel, O. Guschinskaya, N. Guschinsky, G. Levin and F. Vernadat, Balancing large-scale machining lines with multi-spindle heads using decomposition. *Int. J. Prod. Res.* **44** (2006) 4105–4120.
[7] A. Dolgui, B. Finel, N. Guschinsky, G. Levin and F. Vernadat, A heuristic approach for transfer lines balancing. *J. Intell. Manuf.* **16** (2005) 159–171.
[8] A. Dolgui, B. Finel, N. Guschinsky, G. Levin and F. Vernadat, MIP approach to balancing transfer lines with blocks of parallel operations. *IIE Trans.* **38** (2006) 869–882.
[9] A. Dolgui, N. Guschinsky and G. Levin, On problem of optimal design of tranfer lines with parallel and sequential operation, in *Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factor Automation*, Vol. 1, edited by J.M. Fuertes, Barcelona, Spain (1999) 329–334.
[10] A. Dolgui, N. Guschinsky and G. Levin, A special case of transfer lines balancing by graph approach. *Eur. J. Oper. Res.* **168** (2006) 732–746.
[11] A. Dolgui, N. Guschinsky, G. Levin and J.M. Proth, Optimisation of multi-position machines and transfer lines. *Eur. J. Oper. Res.* **185** (2008) 1375–1389.
[12] B. Finel, *Structuration de lignes d'usinage : méthodes exactes et heuristiques*. Thèse de doctorat, Université de Metz (2004).
[13] S. Ghosh and R. Gagnon, A comprehensive literature review and analysis of the design, balancing and scheduling of assembly lines. *Inter. J. Prod. Res.* **27** (1989) 637–670.
[14] O. Guschinskaya, *Outils d'aide à la décision pour la conception en avant-projet des systèmes d'usinage à boîtiers multibroches*. Thèse de doctorat, École Nationale Supérieure des Mines de Saint-Etienne (2007).
[15] O. Guschinskaya and A. Dolgui, A comprehensive comparative analysis of exact and heuristic methods for transfer line balancing problems. *Int. J. Prod. Econ.* (2009) (À paraître).
[16] O. Guschinskaya and A. Dolgui, Heuristic methods for a transfer line balancing problem, in *Proceedings of the 19th International Conference on Production Research*, edited by J.A. Ceroni. Valparaiso, Chile, CD-ROM, 6 pages (2007).
[17] O. Guschinskaya, A. Dolgui, N. Guschinsky and G. Levin, A heuristic multi-start decomposition approach. *Eur. J. Oper. Res.* **189** (2007) 902–913.
[18] K. Hitomi, *Manufacturing Systems Engineering*. Taylor & Francis (1996).
[19] S.G. Ponnambalam, P. Aravindan and G.M. Naidu, A comparative evaluation of assembly line balancing heuristics. *Int. J. Adv. Manuf. Technol.* **15** (1999) 577–586.
[20] B. Rekiek, A. Dolgui, A. Delchambre and A. Bratcu, State of art of assembly lines design optimisation. *Ann. Rev. Control* **26** (2002) 163–174.
[21] M.E. Salveson, The assembly line balancing problem. *J. Ind. Engineering* **6** (1955) 18–25.
[22] A. Scholl and C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *Eur. J. Oper. Res.* **168** (2006) 666–693.
[23] F.B. Talbot, J.H. Paterson and W.V. Gehrlein, A comparative evaluation of heuristic line balancing techniques. *Manage. Sci.* **32** (1986) 430–454.
[24] T.S. Wee and M.J. Magazine, Assembly line balancing as generalized bin packing. *Oper. Res. Lett.* **1** (1986) 56–58.