# Problèmes d'équilibrage des lignes de production

Pour répondre concrètement aux problèmes qui se posent dans la production d'une façon générale et dans l'usinage en particulier, il est indispensable d'admettre des hypothèses de travail. Ces hypothèses restreignent la part de réalité du problème traité dans le but de réduire sa complexité. Elles sont nécessaires pour apporter des réponses précises à des problèmes difficiles à aborder. Une bonne illustration de cette approche est le problème de l'équilibrage des lignes d'assemblage (ALB) qui a été largement étudié dans la littérature. Il en dérive plusieurs problèmes : le simple, noté SALBP, pour *Simple Assembly Line Balancing Problem*, et les généralisés notés GALBP, pour *Generalized Assembly Line Balancing Problem*. Ces problèmes sont moins complexes que celui que nous traitons, néanmoins nous avons fait le choix de les aborder car ils ont de nombreux points en commun avec les problèmes de configuration de lignes d'usinage que nous étudions dans ce mémoire. Nous les décrivons de façon formelle indiquons leurs hypothèses dans cette section. Nous présentons également quelques méthodes d'optimisation qui ont été employées pour les résoudre.

## 3.1 Le SALBP et ses variantes

Une ligne d'assemblage consiste en une série de stations, chacune effectuant un ensemble d'opérations[1]. Les opérations sont caractérisées par leur temps d'exécution et elles sont le plus souvent reliées par des contraintes de précédence. Les opérations d'une même station sont exécutées de façon séquentielle. Il en résulte que le temps de travail d'une station est égal à la somme des temps d'exécution de ses opérations. Le temps de cycle impose une cadence aux stations car celles-ci ne doivent pas avoir un temps de travail qui lui est supérieur. Le temps mort d'une station est défini comme étant la différence entre le temps de cycle et son temps de travail. L'équilibrage d'une ligne d'assemblage revient à trouver une affectation de l'ensemble des opérations telle que les contraintes de précédence entre les opérations soient respectées et que le temps mort total de la ligne soit minimal.

[1]Le mot *tâche* est également utilisé pour désigner une opération.

La littérature est particulièrement abondante en travaux pour le SALBP, elle l'est davantage pour sa variante SALBP-1. Nous ne rapporterons, dans ce qui suit, que les travaux qui ont un lien méthodologique avec nos travaux ou qui emploient des techniques proches de celles que nous avons étudiées.

Les hypothèses générales du SALBP, qui ont été posées par Baybars [Bay86], sont les suivantes :

1. les temps d'exécution des opérations sont déterministes,
2. les temps d'exécution des opérations sont indépendants de la station sur laquelle elles sont effectuées,
3. toute opération peut être exécutée sur toute station,
4. toutes les opérations doivent être exécutées,
5. un ordre partiel entre les opérations doit être respecté,
6. une opération est exécutée complètement sur une seule station, c'est-à-dire qu'il ne peut y avoir de chevauchement sur une autre station,
7. la ligne est constituée d'une série de stations et le mode de fonctionnement est synchrone,
8. les stations sont visitées dans un ordre donné,
9. un seul type de produit est considéré,
10. selon les variantes du SALBP soit le temps de cycle $T_0$ est donné soit le nombre de stations $m$ est donné.

Il existe plusieurs variantes de SALBP, elles se différencient notamment par la variation de la dernière hypothèse mais également par leur objectif. La sous-section qui suit est consacrée à décrire ces différentes types en précisant à chaque fois leurs particularités.

Le tableau 3.1 [SB06] synthétise les différentes variantes du SALBP. Le problème SALBP-F est un problème de décision (le F désignant faisabilité). Dans ce cas, le problème revient à trouver une solution réalisable qui respecte les données, en l'occurrence le temps de cycle et le nombre de stations. Contrairement au SALBP-F, les variantes SALBP1 et SLABP2 sont des problèmes d'optimisation. Le premier problème a comme objectif de minimiser le nombre de stations en respectant un temps de cycle donné et fixe, alors que le second, à l'inverse, doit minimiser le temps de cycle en respectant un nombre de stations donné et fixe. Le SALBP-E est plus général car il a pour objectif de maximiser l'efficacité $E = \sum_{i=1}^{n} t_i / mT_0$ (tel que $n$ est le nombre d'opérations et $t_i$ est le temps d'exécution de l'opération $i$). Du fait que $\sum_{i=1}^{n} t_i$ est fixe, l'objectif revient à minimiser le produit du temps de cycle et du nombre de stations en même temps soit à minimiser $mT_0$.

Plusieurs états de l'art détaillés sur le SALBP sont proposés dans [BSW73, Bay86, GG89, ES98, RDDB02] et plus récemment [SB06].

### 3.1.1 Complexité

Du fait que la variante du SALBP-F traitant de la faisabilité du SALBP est, elle-même, un problème NP-complet, les problèmes d'optimisation correspondants, c'est-à-dire SALBP-

| | **Temps de cycle** | |
|---|---|---|
| **Nombre de stations** | Donné | À minimiser |
| Donné | SALBP-F | SALBP-2 |
| À minimiser | SALBP-1 | SALBP-E |

TAB. 3.1 – Les variantes du SALBP

1 et SALBP-2, le sont également, car ces derniers peuvent être résolus en parcourant les solutions du SALBP-F [Kar72, WM86, SB06]. La forte combinatoire du problème a conduit à la recherche de bornes inférieures afin de réduire l'espace de recherche. Comme le montrent les travaux faits dans ce sens que nous citons ci-dessous, les bornes inférieures sont essentiellement obtenues en résolvant des relaxations du problème original en un problème plus facile. Dans ce qui suit, nous rapportons les différentes bornes inférieures proposées pour le SALBP-1, c'est-à-dire pour le nombre de stations (minimiser le temps mort pour le SALBP-1 est équivalent à la minimisation du nombre de stations).

### 3.1.2 Bornes inférieures pour le SALBP-1

Différentes bornes ont été proposées pour la fonction objectif du SALBP-1. Elles se distinguent par leur complexité, celles qui ont une complexité linéaire permettent d'obtenir une estimation rapide mais souvent imprécise. Comme nous l'avons déjà évoqué précédemment, plus d'efforts sont déployés dans les calculs, meilleure est la qualité de la borne. Ce principe est confirmé dans le cas du SALBP-1 ainsi les bornes les plus élaborées fournissent une meilleure approximation de l'objectif, contre-partie d'un plus grand temps de calcul.

#### Méthode constructive

Les bornes suivantes sont les plus intéressantes du point de vue du temps de calcul car elles peuvent être obtenues avec un algorithme de complexité linéaire (c'est-à-dire en $\mathcal{O}(n)$).

Notons l'ensemble des opérations $\mathbf{N} = \{1, \ldots, n\}$, pour toute opération $i \in \mathbf{N}$, son temps d'exécution est donné par $t_i$. La borne donnée par (3.2) est appelée *nombre de stations minimum théorique* [Bay86]. Elle dérive du temps de séjour sur la ligne qui doit être au moins égal à la somme des temps d'exécution de toutes les opérations $i \in \mathbf{N}$, c'est-à-dire :

$$mT_0 \geq \sum_{i=1}^{|\mathbf{N}|} t_i \tag{3.1}$$

avec $m$ désignant le nombre de stations et $T_0$ le temps de cycle.

$$LB_1 = \left\lceil \sum_{i=1}^{|\mathbf{N}|} \frac{t_i}{T_0} \right\rceil \tag{3.2}$$

Tel que $\lceil a \rceil$ est le plus petit entier supérieur ou égal à $a$.

La borne $LB_1$ a une complexité en $\mathcal{O}(n)$, ce qui correspond, en pratique, au parcours des $|\mathbf{N}|$ temps d'exécution des opérations.

Une deuxième borne est proposée par Johnson [Joh88], elle est obtenue en considérant les opérations avec un certain temps d'exécution. Plus exactement, si on considère la proportion de temps correspondant à l'occupation d'une opération dans une station, *i.e.* $p_i = t_i/T_0$, alors il est possible de déduire que les opérations ayant une proportion supérieure à 0,5 ne peuvent pas partager une même station, elles définissent alors une borne inférieure sur le nombre de stations. Celles qui ont une proportion égale à 0,5 se voient attribuer la moitié d'une station.

$$LB_2 = \big|\{i|\frac{1}{2} < p_i \leq 1\}\big| + \Big\lceil \frac{1}{2}.\big|\{i|p_i = \frac{1}{2}\}\big| \Big\rceil \tag{3.3}$$

Il est à noter que la borne $LB_2$ est moins performante que la précédente lorsque le problème comporte de nombreuses opérations ayant des proportions strictement inférieures à 0,5, toutefois dans le cas inverse c'est $LB_2$ qui fournie une meilleure valeur.

Le raisonnement fait précédemment pour les proportions égales à la moitié de la charge d'une station peut être appliqué aux proportions de tiers. Ainsi, les opérations ayant une proportion supérieure à 2/3 déterminent des stations différentes. Les opérations avec un $p_i$ de 1/3 et 2/3 peuvent partager une même station et leur est attribué un coefficient de 1/3 et 2/3 respectivement. Les opérations avec $\frac{1}{3} \leq p_i \leq \frac{2}{3}$ peuvent au moins partager une même station, ils leur est attribué un coefficient de 1/2. Ainsi la borne est égale à :

$$LB_3 = \big|\{i|\frac{2}{3} \leq p_i \leq 1\}\big| + \Big\lceil \frac{2}{3}\big|\{i|p_i = \frac{2}{3}\}\big| + \frac{1}{2}\big|\{i|\frac{1}{3} < p_i < \frac{2}{3}\}\big| + \frac{1}{3}\big|\{i|p_i = \frac{1}{3}\big| \Big\rceil \tag{3.4}$$

**Relaxation à un problème d'ordonnancement à une machine**

Une borne plus élaborée est obtenue en relâchant le SALBP-1 à un problème d'ordonnancement à une seule machine [Joh88]. Les opérations deviennent les travaux (jobs) qui doivent passer sur la machine avec des temps d'exécution égaux à $p_j = t_j/T_0$ pour chaque opération $j$. Toutes les opérations doivent passer sur la machine de façon séquentielle. Après chaque travail $j$, un temps, noté $n_j$ et appelé *tail*, est nécessaire pour achever le reste. L'objectif est de trouver la séquence qui minimise le temps d'exécution total, c'est-à-dire le *makespan*. La solution optimale pour un tel problème est obtenue en exécutant les travaux dans un ordre décroissant de $n_j$. Pour le SALBP-1, pour chaque opération $j$, la valeur de $n_j$ est une borne inférieure sur le nombre de stations nécessaires pour exécuter toutes ses opérations successeurs. Les $n_j$ sont calculés de façon récursive en sens inverse de l'ordre topologique, tel que l'ordre topologique est déduit des rangs des opérations dans le graphe de précédence (il faut ajouter deux nœuds fictifs une source 0 et un puits $n+1$). Ainsi, pour obtenir $n_j$ d'une opération $j$ ayant l'ensemble $F^*$ de successeurs, il est nécessaire de calculer l'ensemble des tails $n_i$ de ces successeurs $i$ démarrant par le dernier nœud $n+1$ qui n'a pas de successeurs. Nous notons $< h_1, \ldots, h_r], \quad r \in F^*$ la liste ordonné des successeurs de $j$.

$n_j = maximum\{p_{h_1} + n_{h_1}, p_{h_1} + p_{h_2} + n_{h_2}, ..., p_{h_1} + ... + p_{h_n} + n_{h_n}\}$

La borne $n_j$, pour l'opération $j$, n'est pas forcément entière. Elle peut également être obtenue au moyen des bornes décrites précédemment, c'est-à-dire $LB_1$, $LB_2$ ou $LB_3$ sans les arrondir.

Ainsi, une borne inférieure sur le nombre de stations de la ligne est donnée par $\lceil n_0 \rceil$ , c'est-à-dire le tail du nœud fictif n'ayant aucun prédécesseur.

Le résultat précédent peut être affiné davantage en calculant, pour chaque opération $j$, une borne inférieure (dite *head* et notée $a_j$) sur le nombre de stations nécessaires à l'execution des opérations prédécesseurs de $j$. Par analogie au calcul des $n_j$ le même raisonnement est fait pour l'obtention des $a_j$. Celles-ci sont calculées de façon récursive en commençant par $a_0$ remontant jusqu'à $a_{n+1}$. De la même façon que pour $\lceil n_0 \rceil$, $\lceil a_{n+1} \rceil$ est une borne sur le nombre de stations.

Il est à signaler que $Z = \max_{j=0,\ldots,n+1}\{a_j + p_j + n_j\}$ est aussi une borne inférieure sur le *makespan* du problème d'ordonnancement à une seule machine utilisant les *heads* et les *tails* [Car82], alors :

$$LB_4 = \max\{\lceil n_0 \rceil, \lceil a_{n+1} \rceil, Z\} \tag{3.5}$$

La complexité des bornes $LB_1$,$LB_2$,$LB_3$ est linéaire c'est-à -dire en $\mathcal{O}(n)$. Par contre, le calcul de $LB_4$ a au moins une complexité en $\mathcal{O}(n^2)$, selon la borne utilisée pour le calcul des limites $a_j$ et $n_j$, $j = 1, \ldots, n$.

**Amélioration incrémentale**

Par opposition aux calculs des bornes décrites précédemment, la borne $LB_5$ est basée sur un raisonnement incrémental sur la valeur de $m$ (le nombre de stations). Cette valeur peut être initialisée avec la valeur de $LB_1$, par exemple. L'idée d'amélioration des itérations réside dans la détection d'une ou plusieurs infaisabilités qui seraient engendrées par la valeur courante de $m$, si c'est le cas la borne est incrémentée et le processus est réitéré, autrement le processus s'arrête car il n'est plus possible d'améliorer la borne.

La borne $LB_5$, proposée par [SB87], utilise la notion de station au plus tôt, notée $E_j$, et de station au plus tard notée $L_j$, pour chaque opération $j$ telle que la station au plus tard est calculée en fonction de $m$. Cependant, afin de distinguer les stations au plus tard liées à chaque valeur de $m$, $L_j(m)$ est utilisé pour désigner la station au plus tard de l'opération $j$ à l'itération correspondante à la valeur $m$. Pour obtenir une solution réalisable, il est nécessaire que la condition $E_j \leq L_j(m)$ soit vérifiée pour toutes les opérations, autrement l'opération correspondante ne pourra être effectuée avec le nombre de stations courant. Ainsi :

$$LB_5 = min\{m | L_{j(m)} \geq E_j\}$$

Cette dernière expression peut être affinée en utilisant les limites $a_j$ et $n_j$ introduites pour le calcul de $LB_4$. Ainsi, les $E_j$ et $L_{j(m)}$ sont données par (3.6) et (3.7) :

$$E_j = \lceil a_j + p_j \rceil, \quad \forall j = 1, \ldots, n \tag{3.6}$$

$$L_{j(m)} = m + 1 - \lceil p_j + n_j \rceil, \quad \forall j = 1, \ldots, n \tag{3.7}$$

Dans ce cas, le temps de calcul pour obtenir la borne $LB_5$ dépend du temps de calcul des limites $a_j$ et $n_j$, $j = 1, \ldots, n$.

## 3.2 Méthodes exactes pour le SALBP-1

Deux familles d'approches de résolution exactes ont été proposées pour le SALBP, la première est basée sur la programmation dynamique [HKS63] et la deuxième est basée sur une procédure par séparation et évaluation. Nous présentons brièvement la première approche et développons plus en détails la seconde car celle-ci a montré sa supériorité tant sur le plan du temps de calcul que sur le plan de l'espace mémoire requis. Nous présentons donc trois procédures : Fable [Joh88], Eureka [Hof92] et Salome-1 [SK97]. Ces procédures mettent en pratique un large éventail des techniques de réduction et de calcul de bornes les plus efficaces qui ont été développées pour le SALBP-1.

### 3.2.1 Programmation dynamique

Les approches basées sur la programmation dynamique se déclinent, selon leur orientation, en deux catégories, elles peuvent être orientées stations ou opérations (tâches). Par exemple, la procédure de Jackson [Jac56] est orientée station. De plus, les nœuds correspondent à différents états représentant chacun un chargement complet d'une station. L'affectation de l'ensemble des opérations d'une station est représentée par un arc valué avec le temps mort de la station correspondante. Pour résoudre le SALBP-1, il suffit de résoudre le problème de plus court chemin dans le graphe construit, voir également [GN64, Man67, Kle63].

Concernant les approches orientées opérations où les états correspondent à l'affectation d'une seule opération, nous citons celle proposée par Baker & Scharge [BS78] et Kao & Queyranne [KQ82]. La dernière est une amélioration de celle de Held, Karp et Shareshian [HKS63]. L'inconvénient majeur est le large espace mémoire requis, et ce malgré les efforts d'améliorations apportés pour le réduire.

### 3.2.2 Fable

La procédure *Fable* met en place un schéma de branchement *orienté tâche*, c'est-à-dire qu'un branchement correspond à l'affectation d'une seule opération à la fois. Un procédé itératif est enclenché pour toutes les opérations qui peuvent être assignées au vu de la solution courante. Les sous-problèmes sont construits en ajoutant, à chaque fois, une seule opération parmi celles qui peut être affectées à la station courante. Si le temps restant sur cette station ne permet pas une telle affectation alors la station est chargée au maximum et l'algorithme ouvre (ajoute) une nouvelle station qui sera désignée à son tour *station courante*. Le branchement se fait en profondeur d'abord, *Depth First Search*, les opérations candidates

sont triées selon l'ordre topologique. Dès qu'une solution réalisable est atteinte, un mécanisme de backtracking est enclenché remontant jusqu'à un point où d'autres branchements sont encore possibles en choisissant toujours de telle façon que l'ordre topologique entre opérations soit respecté.

En termes de bornes inférieures, Fable employe les bornes $LB_1$,$LB_2$,$LB_3$ et $LB_4$, décrites dans les sections précédentes. Les trois premières bornes sont utilisées à chaque nœud, alors que $LB_4$ n'est utilisée qu'au nœud racine car son calcul nécessite plus de temps en raison de sa complexité. Des règles de dominance sont également intégrées dans le schéma de branchement et permettent de couper des branches non prometteuses. En outre, une règle d'incrémentation des temps opératoires est employée pour les opérations qui ne peuvent être combinées avec les autres : le principe est d'arrondir leur temps opératoire à la valeur du temps de cycle, ce qui permet d'améliorer la valeur de la borne inférieure.

### 3.2.3 Eureka

L'arbre de recherche est orienté station, c'est-à-dire chaque branchement correspond à un chargement total d'une station. Connue aussi sous le nom de méthodes de borne inférieure, Eureka est un processus répétitif qui tente de trouver une solution n'excédant pas la meilleure valeur théorique du nombre de stations ($LB$), c'est-à-dire la valeur de la borne inférieure. Néanmoins, si aucune solution n'est trouvée alors la valeur de $LB$ est incrémentée et le processus est réitéré. Tous les nœuds qui correspondent à des solutions partielles dans lesquelles le temps mort est supérieur à la valeur $LB \times T_0 - \sum_{i \in A} t_i$, tel que $A$ est l'ensemble des opérations qui ont déjà été affectées, sont élagués (car la solution correspondante contient plus que $LB$ stations). Lorsque la recherche est fructueuse, une solution réalisable est obtenue, elle correspond forcément à une solution optimale car elle contient $LB$ stations et la condition d'arrêt est satisfaite. L'algorithme fonctionne en deux étapes, la première construit les solution avançant de la première station vers la *n-ème* ; quant à la seconde, elle construit en sens inverse les stations à partir de la dernière remontant jusqu'à la première, et ce en utilisant le graphe de précédence inversé. En fait, un temps limité est accordé à la procédure en avant pour trouver une solution et lorsque celle-ci échoue la procédure cherchant dans le sens inverse est enclenchée pour le même intervalle de temps. Si une des procédures trouve une solution, l'algorithme est arrêté car cette dernière est la solution optimale.

### 3.2.4 Comparaison entre Eureka et Fable

La stratégie de branchement orienté tâche de Fable permet, contrairement à Eureka, d'éviter de construire systématiquement des solutions complètes. Toutefois, les bornes sont plus efficaces lorsque les stations sont chargées au maximum, en outre, le temps mort de la station ne peut être estimé qu'à ce niveau et donc ce sont à ces points précisément qu'un élagage potentiel est plus probable. Fable peut être renforcé dans ce sens, en calculant la borne sur le nombre de stations au niveau des nœuds où le chargement des stations est maximal (c'est-à-dire aux points où les stations sont fermées).

Par ailleurs, le calcul des bornes est moins fréquent dans le cas d'Eureka. En même temps, l'utilisation de $LB_1$ qui est propre à Eureka n'est pas le meilleur choix car la valeur de la borne inférieure peut être plus précise en employant, par exemple, la borne $LB_4$ [Sch99].

Ainsi, ces deux méthodes présentent des points forts mais aussi des faiblesses, elles peuvent être améliorées et combinées de façon à exploiter leur avantages. C'est dans cette perspective que s'inscrit la méthode SALOME-1, proposée dans [SK97].

### 3.2.5 SALOME-1

Le schéma de branchement de SALOME-1 est orienté station, comme dans Eureka, car la borne inférieure est plus efficace dans ce cas, et la stratégie de l'exploration de l'arbre est en profondeur d'abord afin d'obtenir des solutions faisables au plus tôt. L'idée de l'amélioration apportée par SALOME-1 est de remplacer le processus itératif d'Eureka qui construit plusieurs arbres par un autre n'en développant qu'un seul. L'effort est centralisé dans la construction de cet arbre en mettant en place une borne inférieure de bonne qualité. Celle-ci sera calculée à chaque nœud au lieu de l'employer juste à la racine comme dans Eureka. Notons que cette méthode a été adaptée avec succès pour un problème de bin-packing [SK97].

Pour des problèmes de plus grande taille, des méthodes approchées semblent plus appropriées pour une résolution efficace. Pour ce faire, une approche basée sur une procédure par séparation et évaluation tronquée est souvent employée dans ce but. Le critère d'arrêt peut être soit un temps limite à ne pas dépasser soit un nombre de nœuds maximum [Hof63, GP78, HM78]. Quelques autres méthodes approchées sont présentées dans la section suivante.

## 3.3 Méthodes approchées pour le SALBP-1

### 3.3.1 Méthodes constructives

Il existe un nombre important de travaux proposant des heuristiques pour résoudre le SALBP-1. Une revue et comparaison de ces heuristiques est rapportée dans [TPG86]. Dans ce qui suit nous avons choisi de présenter RPW pour *Ranked Positional Weight* qui est une heuristique basée sur une simple règle de choix (une seule solution est construite, c'est-à-dire une heuristique mono-passage) puis nous rapportons une description de COMSOAL pour *COmputer Method of Sequencing Operations for Assembly Lines* qui, à l'inverse de la précédente, exploite un choix aléatoire en générant plusieurs solutions et en gardant la meilleure (une heuristique multi-passage).

#### RPW

Cette procédure est introduite dans [HB61]. Il s'agit d'un algorithme glouton basé sur une règle de priorité statique car le choix des opérations à sélectionner est indépendant du

processus de construction des solutions. Cette règle heuristique s'appuie sur un calcul de poids pour chaque opération, ce calcul n'est effectué qu'une seule fois et les valeurs des poids obtenues restent inchangées pendant le déroulement de l'algorithme. Le poids d'une opération est obtenu en additionnant son temps d'exécution avec la somme de tous les temps opératoires de ses successeurs dans le graphe de précédence. Une liste est obtenue en triant l'ensemble des opérations par ordre décroissant de leur poids. Le processus de construction propose d'affecter sur la station courante d'abord l'opération en tête de liste, c'est-à-dire celle qui a le plus grand poids, à condition que ce qui suit soit vérifié :

1. le temps restant sur la station courante est supérieur ou égal au temps nécessaire à l'exécution de l'opération candidate,
2. tous les prédécesseurs de cette opération ont été déjà affectés.

Si l'opération en tête de liste ne respecte pas ce choix alors c'est l'opération suivante de la liste qui est considérée, etc. Toutefois, si aucune opération de la liste ne peut être affectée à la station courante celle-ci est fermée et une nouvelle station est ouverte, qui devient la station courante. Ensuite, la liste des opérations candidates est mise à jour en supprimant celles qui ont été affectées. Le processus est réitéré jusqu'à ce que toutes les opérations soient assignées.

#### COMSOAL

Arcus [Arc66] propose une version stochastique d'une heuristique multi-passage, la procédure génère plusieurs solutions complètes jusqu'à ce qu'un certain nombre soit atteint. À chaque itération, une opération à affecter est choisie aléatoirement à partir de la liste des opérations disponibles, c'est-à-dire celles qui ont leurs prédécesseurs déjà affectés et dont le temps d'exécution est inférieur au temps restant sur la station courante. Le processus est réitéré jusqu'à l'obtention d'une solution complète. À l'évidence, plus le nombre de solutions construites est important, plus élevées sont les chances de trouver l'optimum (ou une solution proche).

### 3.3.2 Méthodes incrémentales

Ces méthodes sont basées sur l'amélioration successives des solutions (celles-ci peuvent être initialement obtenues avec des méthodes constructives). Par exemple, les algorithmes génétiques manipulent une population de solutions grâce à plusieurs opérateurs de type croisement, sélection et mutation. Leur application fait apparaître quelques difficultés telles que le codage des solutions, le lissage de la fonction fitness et la réparation des solutions rendues irréalisables après l'application de croisements. Nous faisons référence aux travaux de [KKK96, Fal93, PCPV03].

Une autre metaheuristique pour le SALBP est proposée dans [LRS06], la procédure est basée sur la recherche tabou avec deux différences majeures par rapport aux travaux antérieurs de [Chi98]. La première réside dans le fait d'employer deux techniques de recherche locales, qui sont complémentaires, pour l'intensification et la diversification. De plus, des

redéfinitions de l'espace de recherche et de l'objectif ont été suggérées afin d'explorer les solutions infaisables ne respectant pas les contraintes de temps de cycle.

Dans ce qui suit, nous rapportons les travaux qui ont traité de problèmes plus larges, tel que l'équilibrage avec le choix d'équipement ou le CALBP.

## 3.4 Généralisations (GALBP)

Les travaux proposés pour le SALBP ont servi de base pour l'étude de plusieurs extensions. Ces généralisations sont englobées sous la notation de *GALBP* pour *Generalized Assembly Line Balancing Problem*. Une de ces généralisation relaxe la deuxième hypothèse de SALBP-1 (voir 3.1) considérant les temps des opérations comme étant indépendant des stations, ce qui implique que l'ensemble de l'équipement est le même. Dans la pratique, c'est rarement le cas, les temps opératoires sont en réalité directement liés au type d'équipement utilisé.

### 3.4.1 Sélection d'équipement

Nous rapportons, ici, les travaux qui ont abordé les problèmes d'équilibrage des lignes d'assemblages en s'intéressant simultanément à la sélection d'équipements. Ces travaux sont présentés par ordre chronologique.

[GL83] considèrent le cas mono-produit et s'intéressent à la conception des lignes d'assemblage avec la sélection d'équipement en considérant un ordre complet entre les opérations.

[PDK83] étudient une version étendue du SALBP considérant plusieurs alternatives de processus pour une ligne d'assemblage manuelle[2]. Du fait que la ligne est manuelle, les opérations peuvent être effectuées indifféremment sur toutes les stations. Chaque alternative est relative à un ensemble d'opérations et demande des équipements spécialisés pour cet ensemble d'opérations ; ces équipements supplémentaires peuvent alors être ajoutés à l'équipement déjà en place dans la station correspondante. Les auteurs posent le problème comme suit : étant donné un coût induit par l'utilisation d'équipements supplémentaires est-il intéressant de les employer pour réduire le temps d'exécution des opérations ? Pour y répondre, une procédure de séparation et évaluation est construite où un SALBP-1 est résolue à chaque nœud. Cette méthode n'est efficace que pour un petit nombre d'équipements possibles.

[GHR88] considèrent le problème de conception des lignes s'assemblage en présence de plusieurs types d'équipements pour le cas multi-produit. Des hypothèses assumant un ordre complet des opérations pour le même produit et imposant une grande similarité entre les produits ont permis de réduire de façon considérable le nombre de stations possibles. Plus précisément, ce nombre est ramené à $N^k$ où $N$ est le nombre total d'opérations et $k$ a une valeur proche de 2. L'algorithme proposé énumère toutes les possibilités d'équipements pour les stations, pour n'en retenir que les meilleurs.

[2]Par ligne d'assemblage manuelle nous désignons une ligne qui fait intervenir uniquement des opérateurs humains.

Dans [BT00], une affectation des opérations avec une sélection d'équipement pour la conception des lignes d'assemblage flexibles est proposée. En présence de plusieurs types d'équipements disponibles, il faut en choisir un seul pour chaque station et y affecter en même temps des opérations. Chaque équipement a un coût spécifique et le temps d'exécution d'une opération dépend du type d'équipements choisi, c'est-à-dire qu'il y a des équipements qui sont plus performants pour effectuer certaines opérations mais qui seront plus chers et/ou moins performantes pour d'autres opérations. Quelque soit l'équipement choisi, les opérations de la même station sont effectuées en séquence. L'objectif est de minimiser le coût de la ligne qui est composé du coût de l'équipement choisi, et ce en respectant le temps de cycle objectif (déduit du taux de production visé). Les auteurs proposent de modéliser le problème à l'aide d'un programme linéaire en variables binaires et de le résoudre avec une procédure de séparation et d'évaluation. Les auteurs proposent une borne inférieure basée sur la relaxation des contraintes de précédence et d'intégrité des variables. Néanmoins l'algorithme n'est capable de résoudre que des problèmes de taille modeste dans un lapse de temps fixé. Pour les problèmes de plus grande taille, ils proposent un algorithme approché obtenu en tronquant la procédure par séparation et évaluation.

Une version étendue du précédent algorithme est présentée dans [BR03]. Le problème traité prend en compte des stations parallèles.

### 3.4.2 Équilibrage orienté coût (CALBP)

Dans [Ame00a], l'auteur présente le problème CALBP, pour *Cost oriented Assembly Line Balancing Problem*, pour lequel il propose un état de l'art dans [Ame00b]. Dans ce problème d'optimisation il ne s'agit plus de minimiser le nombre de stations mais plutôt le coût engendré par la configuration de la ligne. Plus exactement, l'objectif est posé comme la minimisation du coût salarial lié aux niveaux de qualification des ouvriers impliquant des salaires différents. L'idée proposée est de regrouper les opérations par rapport à un niveau de qualification proche afin qu'un opérateur ayant ce niveau les effectue. En d'autres termes, il faut que les opérateurs hautement qualifiés n'interviennent que pour les opérations les plus difficiles à réaliser, le regroupement des opérations de même niveau permet de centraliser leur intervention. Du fait de la nature de l'objectif, la règle appliquée pour le SALBP tendant à minimiser le nombre de stations ne mène plus forcément à la solution optimale. En effet, la minimisation du nombre de stations peut écarter des solutions optimales pour le CALBP. En particulier, certaines solutions peuvent comporter un plus grand nombre de stations mais revenir moins cher que d'autres configurations comportant moins de stations dans lesquelles les qualifications ont été mal affectées. Une méthode exacte et une borne inférieure pour le nombre de stations et pour le coût sont proposées dans [Ame00a] et une méthode heuristique est développée dans [Ame01].

## 3.5 Problèmes d'équilibrage des lignes d'usinage

À présent, nous allons présenté les travaux concernant l'équilibrage des lignes d'usinage. La littérature est beaucoup moins abondante dans ce domaine malgré son importance. Pour l'usinage, les objectifs a atteindre ne sont pas forcément les mêmes que pour les lignes d'assemblage. En effet, lorsque pour l'assemblage manuel il est important d'équilibrer la charge de travail entre les postes en raison de la présence d'opérateurs humains, pour les lignes d'usinage la préoccupation n'est plus à lisser la charge car les lignes sont complètement automatisées. Par contre, il est essentiel de réduire le coût d'investissement pour ce type de lignes pour les raisons que nous avons déjà évoquées (voir section 1.4).

Dans la suite, nous passons en revue les travaux proposés dans le cadre de l'usinage en terminant par ceux qui ont été effectués par notre équipe de recherche.

Dans [Sza97], un problème d'optimisation de processus d'usinage est étudié. L'auteur se limite à la formulation du SALBP-1 et applique la méthode de recherche du plus court chemin dans un graphe spécialement construit. L'originalité du travail réside dans le fait que l'auteur considère pour chaque opération plusieurs états représentés par le couple : opération et positionnement de la pièce. L'approche intègre également les régimes d'usinage (vitesse de coupe) dans les variables de décision.

Les travaux de Masood [Mas06] proposent une étude de cas dans le cadre de l'équilibrage d'une ligne de transfert pour la fabrication de blocs de cylindres. L'objectif est de réduire le temps de cycle et d'augmenter le taux d'utilisation de la ligne. Pour ce faire, l'auteur utilise une approche par simulation qui tient compte du changement d'outils traitant des problèmes de 10 outils et 16 opérations.

Les travaux de [YUA02] se placent dans un environnement de RMS. Les auteurs y proposent d'optimiser des lignes d'usinage fabriquant des produits modulaires. Par production modulaire, ils entendent la production de plusieurs instances de modules, qu'ils peuvent combiner par la suite pour obtenir des produits finaux. Cette modularité permet d'atteindre une grande variété de produits finaux lorsque le nombre d'instances par module est suffisamment significatif. Ainsi, une instance de chaque module peut être combiné avec d'autres instances d'autres modules pour composer un produit final. Une ligne est utilisée pour la production de chaque module, il y a donc autant de lignes que de modules. Chaque ligne est initialement configurée pour la fabrication d'une instance donnée et le passage à la production d'une autre instance devient possible suite à une reconfiguration de cette ligne. Le problème posé est alors de trouver la meilleure granularité possible afin d'avoir la diversification voulue à moindre coût (le coût étant engendré par les reconfigurations des lignes). En fait, plus les instances d'un module sont différentes plus la reconfiguration sera coûteuse car les modifications apportées seront plus lourdes. D'un autre côté, si les instances sont choisies de façon à minimiser le coût de reconfiguration, le nombre de ces instances sera élevé implicant une augmentation du nombre de reconfigurations et du coût total.

Il est également intéressant de citer les travaux de [TYHWK03] qui proposent d'appliquer les algorithmes génétiques pour résoudre simultanément le problème de l'équilibrage d'une ligne multi-produit et celui de la sélection des machines et des stocks tampons. L'objectif

considéré est la maximisation de la productivité de la ligne.

## 3.6 Optimisation des lignes de transfert

Nous rapportons dans cette section les travaux qui ont été effectués pour la conception préliminaire des lignes de transfert dans le cadre de la thèse de [Fin04]. Cette thèse s'est intéressée à la structuration de lignes de transfert en termes de définition de blocs d'opérations et de leur affectation aux stations. Ce problème est noté TLBP pour *Transfer Line Balancing Line Problem*. Les opérations d'un bloc sont exécutées en parallèle par la tête multi-broche (unité) correspondante. Le problème d'optimisation posé revient à trouver les regroupements des opérations en blocs et une affectation de ces blocs aux stations.

Plusieurs contraintes ont été introduites :

- certaines opérations ne doivent pas être exécutées sur une même station,
- certaines opérations ne peuvent pas être affectées dans un même bloc,
- certaines opérations doivent être affectées dans une même station,
- certaines opérations doivent être exécutées dans le même bloc.

D'autres contraintes technologiques ont également été prises en compte : elles consistent à limiter le nombre d'opérations par bloc, le nombre de blocs par station et le nombre de stations sur la ligne.

Nous rapportons dans ce qui suit les différences essentielles entre le problème étudié dans [Fin04] et celui que nous traitons dans ce mémoire :

Dans les travaux de [Fin04] tous les regroupements possibles des opérations sont envisagés et représentent des blocs potentiels, tandis que dans notre cas seuls les regroupements des opérations pour lesquels il existe des têtes d'usinage qui peuvent les effectuer sont considérés. Autrement dit, dans le premier cas, les blocs sont construits durant le processus d'optimisation et la solution fournit les regroupements des opérations correspondant aux têtes qu'il faudra acquérir (ou concevoir et fabriquer). Se posent alors les problèmes de la disponibilité (ou faisabilité) de ces têtes d'usinage et de leur coût. En d'autres termes, les solutions fournies ne permettent pas toujours de réaliser les structures de lignes, voire pire, elles peuvent engendrer des coûts supplémentaires inutilement. Pour y remédier, nous avons proposé de construire d'abord l'ensemble des têtes d'usinages faisables qui va constituer la base de la procédure de recherche des configurations. Cet ensemble est une donnée de départ, toutes les structures proposées deviennent réalisables et restent en adéquation avec la disponibilité du marché. Les travaux de [Fin04] peuvent s'inscrire dans une démarche antérieure à l'étude qui nous intéresse dans cette thèse pour nous fournir les unités d'usinage.

Une autre différence importante réside dans la définition de l'objectif à minimiser. Dans [Fin04], l'objectif est de minimiser le nombre de stations et têtes d'usinage, considérant une somme pondérée de ces derniers. Plus exactement, le même coût est attribué à toute tête qui est construite et un coût propre à la création d'une station est intégré. Les pondérations

représentent des poids induisant des coûts relatifs des têtes par rapport aux stations. L'ajustement de ces paramètres permet de définir une proportion de coût d'une tête par rapport à celui d'une station. Du fait que les têtes d'usinage sont construites durant le processus d'optimisation, il devient impossible d'estimer leur coût à cette étape. Dans cette thèse, étant donné l'ensemble des unités d'usinage disponibles nous proposons d'affiner l'optimisation en considérant le coût total comprenant le coût des unités et des stations.

Dans ces travaux, c'est le mode séquentiel pour l'activation des têtes d'usinage qui est considéré. Alors que dans cette thèse, nous abordons tous les cas possibles, à savoir : le cas parallèle où les têtes sont exécutés de façon simultanée et le cas mixte qui est une généralisation du mode séquentiel et parallèle.

Pour le problème traité dans [Fin04] de nombreuses approches ont été proposées. Entre autres, une approche basée sur la programmation en variables mixtes [DFG+06b]. Les auteurs ont utilisé Xpress MP et ensuite ILOG Cplex, pour implémenter et résoudre les modèles PLVM proposés. Dans [DFG+05] deux heuristiques ont été suggérées en se basant sur la méthode COMSOAL. Dans [DFG+06a] une autre heuristique est suggérée par les auteurs qui décomposent le problème en plusieurs sous-problèmes où chaque sous-problème est résolu de manière exacte.

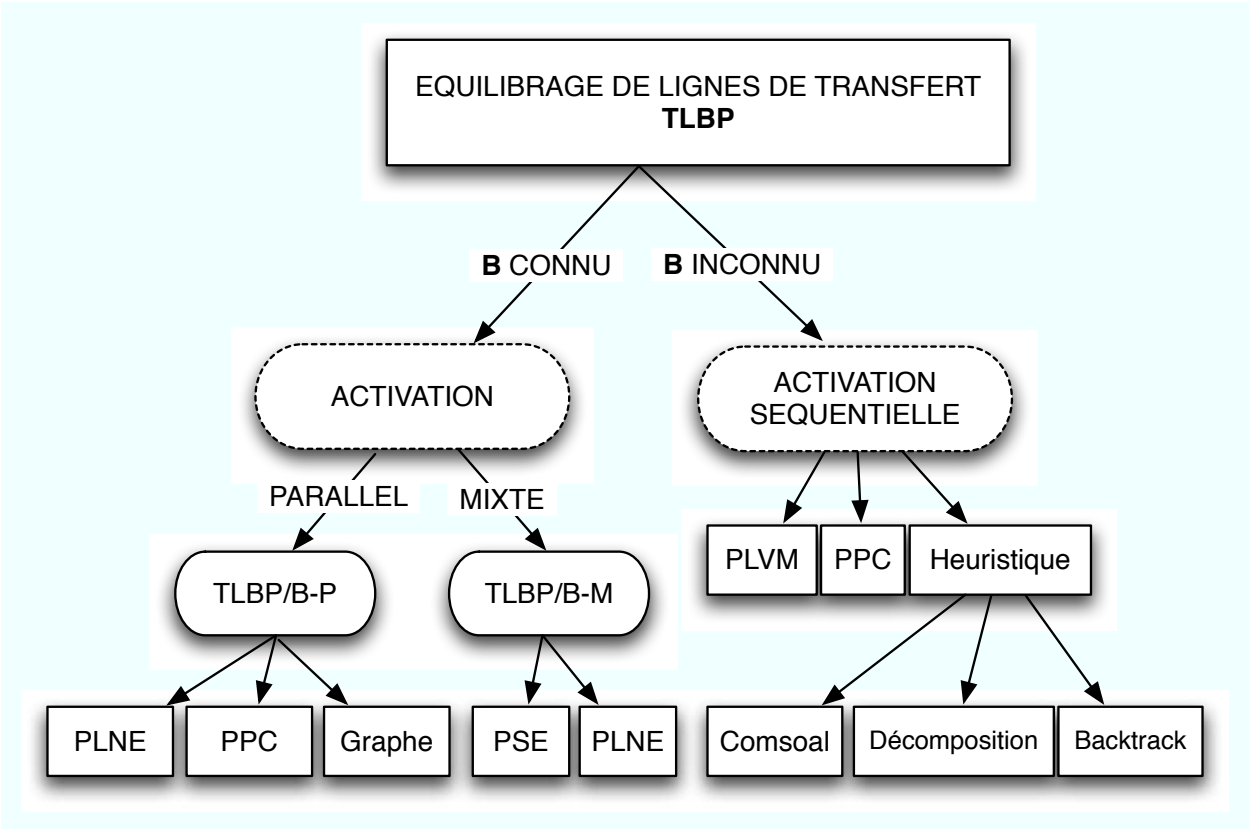


FIG. 3.1 – Classification des problèmes de type TLBP

Le schéma de la figure 3.1 récapitule et classifie l'ensemble des travaux pour les différentes

variantes de TLBP en distinguant ceux qui ont été proposés dans le cadre de cette thèse par un double cadrage.

Dans la suite, nous présentons des travaux dans notre équipe en attirant l'attention sur le fait que ces travaux ont été réalisées en parallèle aux travaux de cette thèse. L'objectif de cette démarche était de proposer un large panel de méthodes pour la résolution du problème traité afin de comparer et d'évaluer les points forts ainsi que les faiblesses de chacune d'entre elles. L'existence d'autres approches pour le problème traité permettent également de valider les résultats obtenus dans cette thèse.

### 3.6.1 Une approche graphe pour le TLBP/B-P

Dans [DGL06], les auteurs proposent une approche basée sur la programmation dynamique (graphe) pour résoudre le TLBP/B-P. Nous avons consacré la majeure partie de cette thèse à l'étude du même problème, aussi nous reviendrons plus en détail sur sa description (voir section 4.1).

Le TLBP/B-P se situe dans la catégorie des problèmes pour lesquels l'ensemble des têtes (unités) d'usinage sont connues à l'avance (voir classification 3.1). Rappelons, qu'il s'agit de sélectionner un sous-ensemble à partir de l'ensemble de départ de manière à exécuter les opérations nécessaires à la fabrication du produit en respectant toutes les contraintes. Plusieurs types de contraintes sont prises en compte, certaines sont propres aux opérations telles que les précédences et les inclusions, d'autres sont relatives aux unités d'usinage telles que les incompatibilités et les contraintes sur le nombre maximum d'unités par station. De plus, un nombre maximal de stations à ne pas dépasser sur la ligne est également imposé. L'objectif est de minimiser le coût total de la ligne qui est composé du coût des unités et du coût des stations.

L'algorithme suggéré transforme le TLBP/B-P en un problème de recherche du plus court chemin sous contraintes dans un graphe orienté. Les sommets de ce graphe correspondent à des solutions partielles, chacune d'elles caractérise un état représentant l'ensemble des opérations déjà affectées.

La progression dans la construction du graphe se fait en ajoutant un arc sortant d'un sommet existant vers un nouveau sommet s'il existe un chargement possible d'une nouvelle station par rapport à l'état du sommet existant. En d'autres termes, pour chaque nœud développé, une nouvelle station est ouverte, celle-ci est chargée complètement puis elle est fermée. Ce chargement doit respecter l'ensemble des contraintes du problème à l'exception du nombre maximum de stations. L'arc ainsi ajouté est valué avec le coût correspondant à cette affectation comprenant le coût d'une station additionnelle ainsi que le coût des unités qui y ont été affectées. Ce processus est réitéré jusqu'à ce que toutes les possibilités de chargements, pour chacune des stations, aient été explorées.

Le graphe $G$ ainsi construit contient un ensemble de chemins $X$, chacun d'eux représentant une solution complète allant du sommet de départ au sommet final.

Le problème initial est ramené au problème de recherche du chemin le plus court dans le

graphe $G$ avec au plus $m_0$ arcs où $m_0$ est le nombre maximum de stations sur la ligne. Le problème est formellement décrit comme suit :

$$Minimiser \sum_{k=1}^{m(x)} C(d_k(x)) \tag{3.8}$$

$$x \in \mathbf{X} \tag{3.9}$$

$$m(x) \leq m_0 \tag{3.10}$$

où $d_k(x)$ est l'arc $k$ du chemin $x$, $C(d_k(x))$ est le coût associé à l'arc $d_k(x)$ et $m(x)$ est le nombre de stations pour $x$. La construction du graphe a été améliorée en intégrant plusieurs règles de dominance.

### 3.6.2 Un algorithme de type PSE pour le TLBP/B-M

Cette approche a été proposée dans [DI05], elle est désignée par PSE dans le schéma des travaux pour le TLBP (voir figure 3.1). Le problème étudié est le TLBP/B-M, rappelons que dans cette variante l'ensemble des unités d'usinage est également supposé connu au préalable. Quant à l'activation des unités d'une station, elle est mixte, pour plus de détail se référer à la section 1.5.2.

L'algorithme global est une PSE qui s'appuie sur une politique de branchements orienté blocs. Un branchement correspond ainsi à l'affectation d'une unité à la dernière station ouverte. Le choix du prochain nœud à brancher est de type *Best Lower Bound* de façon à développer d'abord le nœud ayant la borne inférieure minimale.

La borne inférieure sur l'objectif est intégrée à chaque nœud de l'arbre. Pour son obtention, il faut détenir d'une part une borne inférieure sur le nombre de stations et d'autre part une borne sur le coût des unités d'usinage qui restent à affecter.

Le calcul de la borne inférieure sur le nombre de stations s'appuie sur l'exploitation des contraintes d'exclusion, celles des précédences, le nombre maximum d'unités par station et temps de cycle maximum. L'idée est de déduire à partir de ces contraintes des ensembles de blocs qui ne peuvent partager une même station. Plus précisément, les graphes de précédence et d'exclusion sont agrégés dans un graphe commun après une transformation du graphe de précédence. Le complément du graphe est ainsi obtenu, les composantes sont identifiées et une borne sur le nombre de stations pour chacune des composantes est estimée. La borne sur le nombre total de stations est la somme des bornes de chacune des stations.

Pour la borne sur le coût des unités, un problème de set partitioning est défini pour déterminer l'ensemble optimal des unités d'usinage à sélectionner parmi celles qui restent. Plus exactement, le problème de set partitioning se pose comme la détermination d'un sous-ensemble d'unités à partir de celles non encore affectées pour couvrir l'ensemble des opérations résiduelles (les opérations absentes de la solution partielle correspondante). Le sous-ensemble d'unités ayant le coût minimum est obtenu en résolvant un problème de set partitioning en appliquant une méthode PSE. Le coût optimal du set partitioning est une borne inférieure pour le coût des unités du problème initial le TLBP/B-M.

## 3.7 Conclusion

Dans ce chapitre, nous avons repris les travaux relatifs à l'équilibrage des lignes d'assemblage, en particulier ceux concernant le SALBP, en raison des points de similitudes qu'ils présentent avec les problèmes traités dans ce mémoire. En particulier, il s'agit, dans les deux cas, de trouver la meilleure affectation des opérations aux postes de travail. Il reste néanmoins des différences importantes qui font que les méthodes proposées pour l'équilibrage des lignes d'assemblage ne sont plus valides pour le problème de configuration des lignes d'usinage qui nous intéresse. Nous rapportons dans les points suivants, les divergences essentielles avec le SALBP :

1. les regroupements d'opérations en blocs ne sont pas considérés dans le SALBP,
2. il n'y a pas de choix possible pour effectuer une opération donnée,
3. l'ordre d'exécution des opérations est toujours séquentiel au sein d'une station,
4. l'objectif du SALBP est de minimiser le nombre de stations, car dans l'assemblage il reflète le coût de la ligne en raison de la présence d'opérateurs humains (salaires), tandis que pour les lignes d'usinage il faut tenir compte, en plus, du coût de l'équipement.

Concernant les travaux de [BT00] pour le choix d'équipement, au delà des différences (1) et (3), citées pour le SALBP, il subsiste le fait de ne considérer qu'un seul type d'équipement par station.

Nous avons ensuite repris les problèmes proches qui ont été traités dans le cadre de l'équilibrage des lignes d'usinage en mettant l'accent sur les travaux effectués au sein de notre équipe. Nous avons suggéré une classification pour positionner l'ensemble des problèmes étudiés. Puis, nous avons abordé les travaux de [Fin04] en montrant les différences essentielles qui font que les modèles proposés ne sont plus applicable pour le TLBP/B-P étudié dans ce mémoire. En particulier, nous soulignons le fait que dans le TLBP il faut déterminer les blocs d'opérations pendant le processus d'optimisation. En effet, il s'agit de trouver les blocs d'opérations pour lesquels il faut construire des unités d'usinage par la suite ce qui situe le TLBP à une phase antérieure à celle qui nous intéresse lors de la résolution du TLBP/B-P

Enfin, nous avons décrit les travaux qui concernent les mêmes problèmes que ceux qui nous intéressent dans la présente thèse, à savoir : le TLBP/B-P et TLBP/B-M. Ces méthodes nous ont aidé à valider respectivement nos modèles et en évaluer les performances.