Modélisation et résolution du TLBP/B-P

Dans ce chapitre, nous proposons d'abord une formulation générique. Puis, nous décrivons le schéma global adopté pour une approche basée sur la programmation par contraintes (PPC) en décrivant les règles de propagation qui sont appliquées ainsi que les règles de dominance. Ensuite, deux bornes inférieures sont fournies, l'une est basée sur une relaxation linéaire et la seconde sur une relaxation à un problème particulier de set partitioning. Nous proposons également un modèle basé sur la programmation linéaire en nombres entiers que nous améliorons par la suite en reformulant certaines contraintes. Enfin, nous suggérons des techniques efficaces pour réduire le nombre de variables binaires et augmenter les performances du modèle linéaire.

5.1 Approche basée sur la PPC

Dans cette partie du travail, nous proposons deux bornes inférieures pour le TLBP/B-P : la première est une relaxation à un problème particulier de set partitioning [BDG+04], quant à la seconde, elle est obtenue grâce à une relaxation linéaire d'un des modèles linéaires en nombres entiers que nous proposons dans la suite.

Nous décrivons ensuite le schéma global de l'approche PPC, celui-ci est basée essentiellement sur une méthode de type séparation et évaluation combinée à une règle de dominance. Nous décrivons le modèle utilisé pour cette approche, ensuite nous indiquons la politique de branchement utilisée pour la séparation et l'évaluation.

5.1.1 Une relaxation au problème de set partitioning

Dans cette section, nous proposons une borne inférieure basée sur une relaxation du TLBP/B-P à un problème de set partitioning [BDG+04]. Pour ce faire, nous calculons d'abord une borne inférieure sur le nombre des stations et ensuite une estimation du coût engendré par les blocs sélectionnés.

Nous proposons d'exploiter la structure spécifique du problème pour en déduire une borne

inférieure sur le nombre de stations. Ensuite, nous proposons une relaxation à un problème de partitionnement particulier afin de déduire une borne sur le second terme de l'objectif, à savoir : le coût des blocs sélectionnés. Cette borne est une adaptation de celle proposée dans les travaux de [DI05] où les auteurs se sont intéressés au problème mixte (de type TLBP/B-M, voir la classification dans la figure 3.1).

Nombre de stations

Pour obtenir une borne inférieure sur le nombre de stations nous exploitons d'abord les contraintes d'exclusion entre les blocs avec les contraintes de précédence entre les opérations. Puis, nous renforçons la valeur de la borne en prenant en compte également les contraintes sur le nombre maximum de blocs par station.

Dans un premier temps, nous construisons un graphe non orienté pour les blocs à partir du graphe de précédence G^{or}. Nous notons ce graphe $G^{br} = (\mathbf{B}, E^{br})$. Plus précisément, le graphe G^{br} est obtenu après une transformation de G^{or}, en déduisant les contraintes de précédence entre les blocs à partir de celles qui existent entre les opérations. En particulier, chaque arc qui lie deux opérations est transformé en une ou plusieurs arêtes qui lient les blocs qui les contiennent. Ainsi, s'il existe un arc entre les sommets i et j dans G^{or}, sa correspondance, dans G^{br}, est telle que, tout bloc qui contient l'opération i est lié par une arête à tout bloc qui contient l'opération j. De plus, nous appliquons la fermeture transitive du graphe G^{br} ainsi obtenu[1]. Ainsi, si $\exists(b, b') \in E^{br}$ et $\exists(b', b'') \in E^{br}$ alors nous ajoutons l'arête (b, b'') dans E^{br}.

Ensuite, nous fusionnons les deux graphes G^{ex} et G^{br} pour obtenir le graphe G^B. Ce graphe contient ainsi l'ensemble des contraintes d'exclusion et toutes les précédences reliant les blocs. Puis, nous construisons le complément de ce graphe que nous notons $\overline{G^B}$, tel que $\overline{G^B}$ contient toutes les arêtes qui existent dans le graphe complet[2] et qui n'existe pas dans le graphe G^B. Nous notons $\overline{G^B_k} = (V_k, E^{V_k}), k = 1, \ldots, l$, toute composante connexe du graphe $\overline{G^B}$. Il est clair que deux blocs qui appartiennent à deux composantes distinctes ne peuvent pas être affectés à la même station. Ainsi, le nombre de composantes connexes de $\overline{G^B}$ fournit une borne inférieure sur le nombre de stations. De plus, le nombre de stations nécessaires pour les blocs de chaque composante connexe ne peut pas être inférieure à la cardinalité de l'ensemble indépendant maximum de la composante [Aig95].

$$\omega(\overline{G}^B_k) \geq \sum_{\sigma \in V_k} (1 + deg(\sigma))^{-1} \tag{5.1}$$

où $deg(\sigma)$ est le degré du nœud σ appartenant à la composante $\overline{G}^B_k$.

L'ensemble des blocs de chaque composante connexe $\overline{G}^B_k$ requière donc au moins $\lceil \omega(\overline{G}^B_k) \rceil$ stations.

[1] En effet, le graphe de précédence ne contient pas les arcs redondants, ils sont considérés de façon implicite.

[2] Le graphe complet comporte les mêmes sommets, soit dans ce cas l'ensemble des blocs, et contient toutes les arêtes possibles, en d'autres termes tout couple de sommets est lié par un arc.

Cette valeur peut être renforcée en prenant en compte les contraintes de capacité sur le nombre de blocs par station :

$$\tilde{m}_k \geq \left\lceil \frac{|V_k|}{n_0} \right\rceil \tag{5.2}$$

De (5.1) et (5.2) la borne pour chaque composante $\overline{G}_k^B$ est comme suit :

$$m^* = max \left\{ \lceil \omega(\overline{G}_k^X) \rceil, \left\lceil \frac{|V_k|}{n_0} \right\rceil \right\} \tag{5.3}$$

En additionnant l'ensemble des bornes nécessaires pour chaque composante du graphe, nous obtenons une borne inférieure sur le nombre de stations pour l'ensemble des opérations :

$$\tilde{m} = \sum_{k=1}^{l} \tilde{m}_k \tag{5.4}$$

Coût des blocs

Pour obtenir une borne inférieure sur le coût des blocs, il faut résoudre à l'optimum le problème de set partitioning. Pour ce faire, il est nécessaire de trouver le meilleur sous-ensemble $\beta \subseteq \mathbf{B}$ pour exécuter les opérations non encore affectées, c'est-à-dire $\mathbf{N}' = \mathbf{N} - \{i \mid i \in b, b \in B\}$, tel que $B \subset \mathbf{B}$ est l'ensemble des blocs qui ont déjà été affectés.

Si nous définissons $\mathcal{N}(\beta)$, comme étant l'ensemble des opérations des blocs β, alors $\mathcal{N}(\beta)$ définit l'ensemble des blocs qui restent à affecter, à savoir : $B_2 = \{b \mid b \subseteq N'\}$.

Le sous-ensemble β est alors une partition des opérations non encore affectées en des blocs appartenant à $\mathbf{B} - B$. C'est un problème de set partitioning qui doit vérifier les conditions suivantes :

$$b \in B_2 \tag{5.5}$$

$$\mathcal{N}(b) \bigcap \mathcal{N}(b') = \varnothing, \qquad \forall b, b' \in \beta \tag{5.6}$$

$$\bigcup_{b \in \beta} \mathcal{N}(b) = \mathbf{N}' \tag{5.7}$$

De plus, il faut que la valeur de l'objectif soit minimale :

$$\sum_{b \in \beta} q_b + (m' - 1 + m^*)C \tag{5.8}$$

Nous utilisons une procédure de séparation et d'évaluation pour résoudre le problème (5.5)-(5.8).

5.1.2 Une relaxation linéaire

Cette borne inférieure est obtenue en relâchant les contraintes d'intégrité des variables binaires utilisées dans le modèle linéaire en nombres entiers (5.17)-(5.25) et (5.36)-(5.43) (pour la seconde formulation le coût m^*C induit par l'ouverture sûre de m^* doit être ajouté).

Toutefois, afin que cette borne soit valide dans le schéma global, c'est-à-dire à un nœud donné de l'arbre, il faut intégrer l'ensemble des contraintes qui ont été fixées auparavant. Plus exactement, il faut considérer l'ensemble des blocs qui ont déjà été affectés aux stations et intégrer les contraintes correspondantes dans la relaxation linéaire.

5.1.3 Un modèle générique

Le modèle que nous avons utilisé pour la résolution par PPC est orienté stations. Nous désignons par $L = \{Sta_1, \ldots, Sta_k, \ldots, Sta_m\}$ une solution du problème, tel que : $Sta_k \subset \mathbf{B}$ est l'ensemble des blocs affectés à la station k.

L'objectif qui est exprimé par (5.9) minimise le coût de la lignes composé du coût des m stations ouvertes (le premier terme) ainsi que le coût des blocs sélectionnés (le second terme).

$$Minimiser \quad Cm + \sum_{k=1}^{m} \sum_{b \in Sta_k} q_b \tag{5.9}$$

Pour l'exécution de l'ensemble des opérations nous utilisons la contrainte (5.10). Cette égalité permet de vérifier que l'ensemble constitué de l'union des opérations contenues dans les blocs sélectionnés coincide effectivement avec l'ensemble des opérations à effectuer $\mathbf{N}$.

$$\bigcup_{k=1}^{m} \mathcal{N}(Sta_k) = \mathbf{N} \tag{5.10}$$

Les contraintes (5.11) imposent aux blocs sélectionnés d'être mutuellement disjoints. Ainsi, ces contraintes et les contraintes (5.10) assurent conjointement l'unicité des opérations. De façon plus précise, sans les contraintes (5.11), les solutions pourraient contenir les mêmes opérations plusieurs fois car les contraintes (5.10) imposent uniquement la présence de l'ensemble des opérations sans vérifier leurs duplications éventuelles. Dans ce cas, certaines opérations seraient effectuées plusieurs fois, ce qui est à proscrire.

$$\forall b, b' \in \bigcup_{k=1}^{m} Sta_k, \quad \mathcal{N}(b) \cap \mathcal{N}(b') = \varnothing \tag{5.11}$$

Les contraintes (5.12) imposent le respect des précédences. Nous employons dans ces contraintes la notation $\Gamma^-(b)$ afin de représenter l'ensemble des opérations prédécesseurs de

celles appartenant au bloc b. Ainsi, pour tout bloc b affecté à une station k, nous imposons que ses prédécesseurs soient affectés aux stations antérieures strictement à k.

$$\Gamma^{-}(b) \subseteq \bigcup_{h=1}^{k-1} \mathcal{N}(Sta_h), \quad \forall b \in Sta_k, \forall k = 1, \ldots, m \tag{5.12}$$

Les contraintes d'inclusion pour les opérations sont assurées par (5.13). Elles imposent, plus précisément, à l'ensemble des blocs Sta_k composant une station k de contenir soit toutes les opérations d'un ensemble d'inclusion d, soit aucune d'elles.

$$\Big(\bigcup_{b \in Sta_k} \mathcal{N}(b) \bigcap d \Big) \in \{\varnothing, d\}, \quad \forall d \in D^{in}, \forall k = 1, \ldots, m \tag{5.13}$$

Les contraintes d'exclusion sont exprimées dans (5.14). Elles interdisent aux blocs appartenant à la station Sta_k de contenir un sous-ensemble $E \in D^{ex}$ de blocs incompatibles.

$$E \not\subseteq Sta_k, \quad \forall E \in D^{ex}, \forall k = 1, \ldots, m \tag{5.14}$$

Les contraintes (5.15) interdisent l'affectation de plus de n_0 blocs par station en imposant à chaque ensemble Sta_k d'avoir une cardinalité inférieur ou égale à n_0.

$$|Sta_k| \leq n_0 \tag{5.15}$$

Les contraintes (5.16) vérifient le respect du nombre maximum de stations.

$$m \leq m_0 \tag{5.16}$$

Nous avons implémenté ce modèle à l'aide des libraires de ILOG Solver [ILO].

À présent, que nous avons fourni le modèle générique pour l'approche PPC, nous indiquons les algorithmes de filtrage que nous avons employés afin de réduire l'effort de recherche.

5.1.4 Propagation des contraintes

Dans la présente section, nous décrivons les filtrages des domaines des variables Sta_k que nous avons intégrés dans le schéma de résolution global. Il est à signaler que les règles de déduction proposées dans cette section sont appliquées à chaque nœud de l'arbre de séparation et d'évaluation. De plus, nous utilisons une approche orientée stations dont le principe est d'ouvrir les stations au fur et à mesure de l'affectation des blocs. Ainsi, la première station ayant l'indice 1 est ouverte. Une fois que l'ensemble des affectations possibles de Sta_1 a été considéré, la seconde station est ouverte et ainsi de suite. Ce processus est réitéré jusqu'à ce que l'ensemble des opérations soit affecté.

À présent, nous introduisons le domaine de chaque ensemble Sta_k pour décrire la propagation des contraintes que nous utilisons, celui-ci est défini comme suit :

$D_{Sta_k} = \{b \in \mathbf{B} \mid k \in [head_b, tail_b]\}$.

où $head_b$ (respectivement $tail_b$) est l'indice de la station au plus tôt (respectivement au plus tard) pour le bloc b tel que $b \in \mathbf{B}$. Nous avons proposé dans la section 5.4 plusieurs algorithmes pour le calcul de ces limites (par défaut $head_b = 1$ et $tail_b = m_0$).

La propagation des contraintes de précédence est appliquée lorsque les deux conditions suivantes sont vérifiées simultanément :

– le bloc b est assigné à Sta_k ;
– le bloc b a des successeurs qui appartiennent au domaine D_{Sta_k}.

Dans ce cas, il est possible d'éliminer l'ensemble des blocs b' appartenant au domaine de la station k qui comportent des opérations successeurs aux opérations de b. Plus formellement, nous appliquons la règle suivante :

$\forall b \in Sta_k$,
Si
$\exists b' \in D_{Sta_k} \mid i \in b, j \in b', (i,j) \in D^{or}$
alors
$D_{Sta_k} := D_{Sta_k} - \{b'\}$

Concernant la propagation des contraintes d'exclusion, nous éliminons un bloc b du domaine D_{Sta_k} lorsqu'un sous-ensemble de $E' \subset E$ tel que $E' = E - \{b\}$ a été affecté à Sta_k. Le bloc b ne doit pas être affecté à la station k car autrement des blocs incompatibles formant E se verraient attribuer une même station. Plus exactement, nous employons la règle suivante :

$\forall E \in D^{ex}$,
Si
$\exists E' \subset \mathbf{B}, \exists b \in \mathbf{B}$ tel que $E' \subseteq Sta_k \wedge E' = E - \{b\}$
alors
$D_{Sta_k} := D_{Sta_k} - \{b\}$

5.1.5 Politique de branchement

Nous avons opté pour un branchement de type : en profondeur d'abord (*DepthFirst*) afin d'obtenir au plus vite des solutions réalisables ce qui fournit une borne supérieure sur l'objectif. L'arbre construit est un arbre n-aire, c'est-à-dire qu'à partir d'un nœud parent plusieurs nœuds fils peuvent être générés.

Avant d'entamer le processus de branchement, il faut construire le graphe de précédence des blocs à partir du graphe de précédence G^{or}. Un branchement correspond à l'affectation d'un bloc disponible à la station courante. La liste de blocs disponibles (noté B_{dispo} dans l'organigramme 5.1) est construite pour chaque nœud de l'arbre car selon les blocs qui ont déjà été affectés de nouveaux blocs deviennent disponibles. Pour obtenir cette liste nous combinons les contraintes de précédence et la propagation des contraintes décrite dans la section précédente.

L'arbre de séparation et d'évaluation est initialisé en fournissant à la racine une borne inférieure sur l'objectif. Un premier niveau de branchements correspond à un point de choix concernant la sélection d'un bloc parmi ceux de la liste des blocs disponibles à la racine. Pour le premier niveau, un branchement revient à affecter un bloc n'ayant pas de prédécesseurs à la première station. Il y aura autant de nœuds fils que de blocs n'ayant pas de prédécesseurs.

Ensuite, pour chaque nœud, une liste des blocs disponibles est construite et le noeud traité devient le noeud parent tel que les fils correspondent à l'affectation des nouveaux blocs disponibles à la station courante. De plus, il faut considérer l'affectation de l'ensemble vide à la station courante au même titre que l'affectation d'un bloc disponible. Ceci signifie que nous mettons fin au chargement de la station courante pour ouvrir une nouvelle station. En effet, cela a pour objectif d'éviter un chargement maximal systématique pour une station. Dans le cas du TLBP, la meilleure solution ne contient pas forcément des stations chargées au maximum car ce sont les blocs les moins chers par opération qui déterminent la solution optimale[3]. De plus, un chargement maximal fait intervenir systématiquement l'ensemble des blocs disponibles ce qui a pour conséquence d'omettre certains blocs alternatifs qui ne seront disponibles qu'à partir des stations ultérieures.

Le processus est réitéré tant qu'il y a des blocs disponibles et que le nombre de stations ouvertes est inférieur à m_0 (le nombre maximum de stations autorisé). Plus exactement, lorsqu'il n'est plus possible d'affecter un bloc à la station en cours, celle-ci est fermée et une autre est ouverte à condition que le nombre de stations ouvertes soit inférieur à la limite m_0. Lorsque celui-ci est supérieure à m_0 l'algorithme s'arrête car le problème est infaisable.

5.1.6 Règle de dominance

Nous avons adapté une règle de dominance d'après les principes d'optimalité de [Bel57]. Cette règle a déjà été appliquée dans [DI05] pour la résolution du TLBP/B-M.

Pour pouvoir appliquer cette règle, il faut définir l'état d'une solution partielle. Celui-ci correspond à l'ensemble des opérations contenues dans les blocs qui ont été déjà affectés.

Proposition 1.

Si pour deux solutions partielles S_1 et S_2, les conditions suivantes sont vérifiées :

- les deux états S_1 et S_2 sont équivalents, c'est-à-dire que les solutions partielles correspondantes contiennent les mêmes opérations ;
- toutes les stations de chacune des solutions partielles sont fermées ;
- Le coût de S_1 est strictement inférieur au coût de S_2.

alors S_1 domine S_2 et cette dernière peut, par conséquent, être éliminée de la recherche.

L'idée de cette proposition vient de la réflexion suivante :

[3]Prenons le cas extrême, s'il y a n opérations et s'il existe un bloc pour chaque opération tel que chacun coûte c et qu'ils peuvent tous être sur une même station. Par ailleurs, s'il existe un bloc contenant toutes les opérations ayant un coût q et que $nc > q$ alors c'est le bloc effectuant toutes les opérations qui déterminera la meilleure solution.

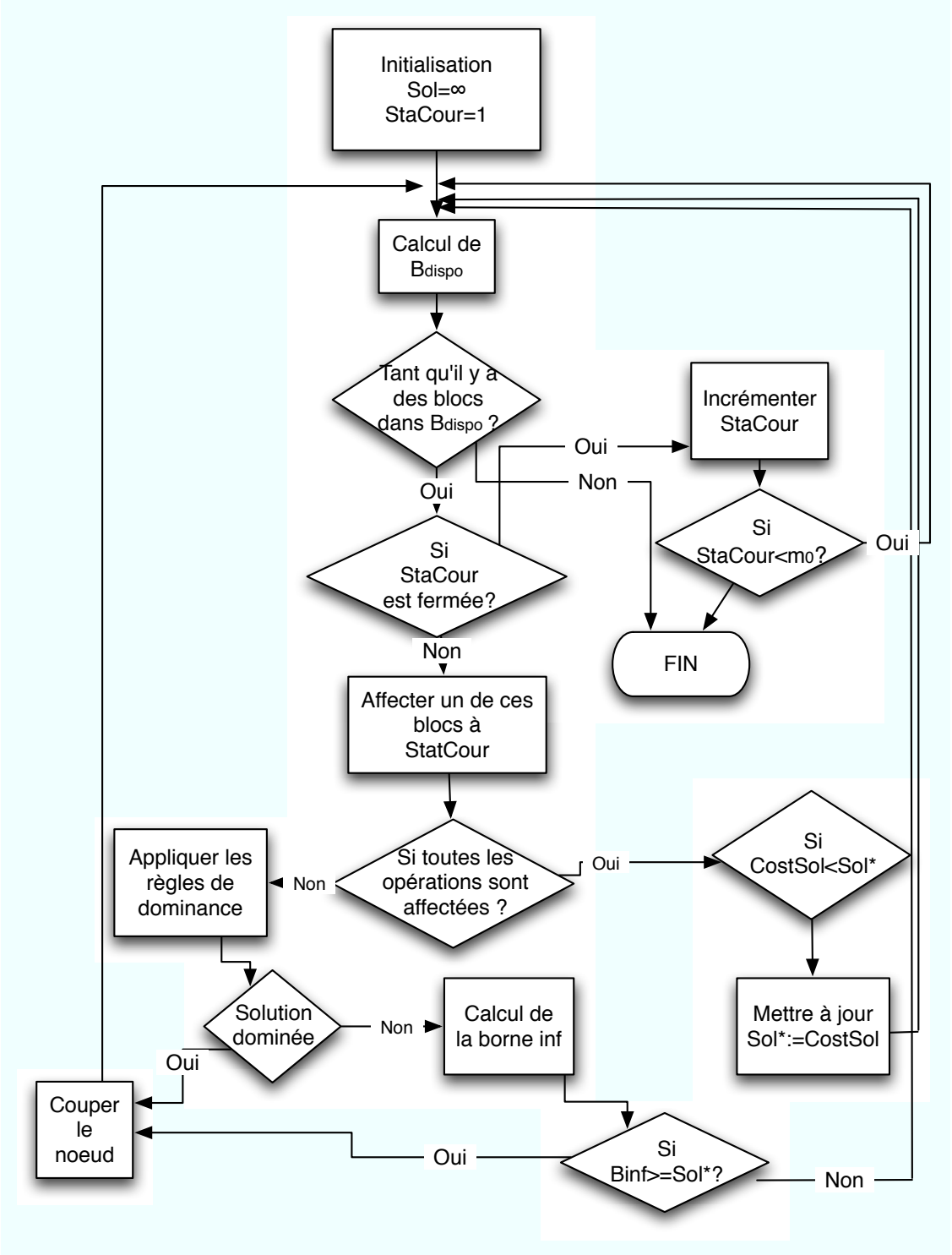

FIG. 5.1 – Schéma de l'approche globale

1. s'il existe deux solutions partielles S_1 et S_2 ayant le même état mais deux coûts différents, c'est-à-dire C_1 et C_2 respectivement, tel que $C_1 < C_2$

2. et si toutes les stations de S_1 et S_2 sont fermées et qu'il existe une sous-séquence optimale de blocs Seq pour compléter une de ces solutions partielles

alors cette sous-séquence est la même pour les deux solutions S_1 et S_2.

De ce fait, les deux solutions réalisables obtenues auront le même rapport de coût que les solutions partielles S_1 et S_2. En d'autres termes, si S_1 domine S_2 alors la solution réalisable composée de S_1 et Seq domine forcément celle qui est composée de S_2 et Seq. Ainsi, toute solution dominée peut être éliminée de la recherche.

Proposition 2.

Lorsque deux solutions partielles S_1 et S_2 ont le même état et qu'elles contiennent les mêmes blocs dans la station courante alors l'une de ces solutions peut être écartée.

Le principe de cette proposition est de détecter les solutions symétriques afin de les éliminer. Par solutions symétriques, nous entendons les solutions considérant les mêmes blocs sélectionnés mais pas la même affectation aux stations.

5.1.7 Approche globale

L'organigramme de l'approche globale est présenté dans la figure 5.1. La procédure commence par calculer l'ensemble des blocs disponibles sur la base des précédences uniquement pour le premier niveau puis en intégrant la propagation des contraintes pour les autres niveaux. Les branchements sont effectués sur la base de cet ensemble de blocs disponibles tel qu'un branchement corresponde à l'affectation de chacun des blocs disponibles à la station courante. Nous appliquons les règles de dominance décrites précédemment avant de calculer la borne inférieure dont le développement est décrit dans la section 5.1.2. L'algorithme s'arrête lorsque tous les nœuds ont été parcourus, la meilleure solution est donnée par Sol^*.

5.2 Un PLNE orienté blocs

Dans cette section, nous proposons une modélisation basée sur la programmation linéaire en nombres entiers. Nous nous sommes d'abord intéressé à une formulation orientée blocs. Nous définissions les variables de décision et présentons le modèle linéaire en nombres entiers ainsi obtenu [DGL$^+$04].

5.2.1 Éléments de modélisation

Pour les précédences

Afin de transcrire les contraintes de précédence qui lient certaines opérations en des contraintes qui lient les blocs qui les contiennent, nous introduisons des ensembles $\Gamma^-(b)$, H, $H(b)$, et une valeur h_{tb} comme suit :

Pour chaque bloc $b \in \mathbf{B}$, nous introduisons un ensemble $\Gamma^{-}(b)$ qui contient les opérations qui précèdent chaque opération appartenant au bloc b, ainsi :

$$\Gamma^{-}(b) = \{i \in \mathbf{N} - \{b\} | \exists j \in b, \quad (i,j) \in D^{or}\}$$

L'ensemble H contient tous les blocs ayant au moins une opération prédécesseur, ainsi :

$$H = \{b \in \mathbf{B} \mid \Gamma^{-}(b) \neq \varnothing\}$$

Pour chaque bloc $b \in \mathbf{B}$, nous calculons l'ensemble $H(b)$ qui contient les blocs effectuant au moins une opérations prédécesseur de b, on écrit alors :

$$H(b) = \{t \in \mathbf{B} \mid t \cap \Gamma^{-}(b) \neq \varnothing\}, \quad \forall b \in \mathbf{B}$$

Le nombre d'opérations prédécesseurs pour les opérations du bloc b qui sont contenues dans le bloc t est donné par :

$$h_{tb} = \mid t \cap \Gamma^{-}(b) \mid, \quad \forall b \in H, \forall t \in H(b)$$

Pour les inclusions

Concernant les relations d'inclusion qui lient certaines opérations, nous introduisons w_{bd} et W_d.

Pour chaque bloc $b \in \mathbf{B}$ et pour chaque sous-ensemble d'opérations $d \in D^{in}$, nous introduisons l'indicateur w_{bd} qui fournit le nombre d'opérations appartenant au bloc b qui sont liées par les contraintes d'inclusion données par d.

$$w_{bd} = \mid \mathcal{N}(b) \cap d \mid, \quad \forall b \in \mathbf{B}, \forall d \in D^{in}$$

Pour chaque élément d'inclusion d, nous introduisons un sous-ensemble W_d qui contient tous les blocs effectuant au moins une opération de d, ainsi :

$$W_d = \{b \in \mathbf{B} \mid w_{bd} > 0\}, \quad \forall d \in D^{in}$$

À chaque opération $i \in d$ correspond un sous-ensemble U_i de $\mathbf{B}$ qui contient tous les blocs pouvant effectuer l'opération i. Formellement, on écrit :

$$U_i = \{b \in \mathbf{B} \mid i \in \mathcal{N}(b)\}, \quad \forall i \in d, \forall d \in D^{in}$$

Pré-traitements spécifiques

À présent, nous suggérons quelques pré-traitements qui exploitent la spécificité des données du problème afin de réduire sa taille. Cette analyse préliminaire a pour objectif de réduire le nombre total de contraintes. Ainsi, entre autres nous distinguons les blocs qui feront l'objet d'un choix de ceux qui seront sélectionnés de façon certaine.

Nous dissocions, parmi l'ensemble $\mathbf{B}$ deux types de blocs :

- Les blocs qui ont des opérations en commun, c'est-à-dire ceux dont l'intersection est non vide. Ces blocs représentent des choix possibles pour chacune des opérations en commun. Une solution réalisable du problème ne doit contenir qu'un seul bloc parmi ces derniers.
- Les blocs qui n'ont pas d'opérations communes. Ces derniers sont donc l'unique possibilité pour effectuer les opérations qui leur appartiennent. Ainsi, ils participent forcément à toute solution réalisable. Formellement, on écrit :
  1. F est un ensemble de sous-ensembles de $\mathbf{B}$ contenant les blocs qui ont au moins une opération en commun. $F = \{F_s, s = 1, \ldots, v\}$ tel que :
     - $s \in \{1, \ldots, v\}, \forall b, b' \in F_s$, $b \cap b' \neq \varnothing$;
     - $\forall (b, b') \in \mathbf{B} \times \mathbf{B}$, si $\mathcal{N}(b) \cap \mathcal{N}(b') \neq \varnothing$ alors $\exists s \in \{1, \ldots, v\}$ tel que $b, b' \in F_s$;
  2. F_0 est le sous-ensemble de $\mathbf{B}$ qui contient les blocs qui n'ont aucune intersection avec d'autres blocs. Par conséquent, si pour une opération donnée $i \in \mathbf{N}$ il n'y a que le bloc b qui la contient alors b appartient forcément à F_0.

- Nous introduisons le sous-ensemble B^* de $\mathbf{B}$ qui contient les blocs qui peuvent être affectés à la dernière station de la ligne.
- m^* est une borne inférieure sur le nombre de stations à ouvrir. Celle-ci est au moins égale à la longueur plus un du plus long chemin dans le graphe de précédence G^{or}. Du fait que les opérations qui sont exécutées sur une même station sont effectuées simultanément, il en résulte que la longueur du plus long chemin du graphe de précédence plus un détermine un nombre minimal de stations à établir[4]. Nous fournissons à la fin de ce chapitre plusieurs algorithmes qui permettent de calculer cette borne.

5.2.2 Les variables de décision

La résolution du problème d'optimisation apporte deux informations pour pouvoir décider de la structure d'une ligne : d'une part il fournit le nombre de stations à établir et d'autre part il détermine l'ensemble des unités d'usinage à mettre en place en précisant leur affectation aux stations. Ainsi, nous définissons les variables de décision correspondantes comme suit :

$$x_{bk} = \begin{cases} 1 & \text{si un bloc } b \in \mathbf{B} \text{ est affecté à la station } k \\ 0 & \text{sinon} \end{cases}, \quad \forall k = 1, \ldots, m_0$$

[4]Ce qui revient à initialiser m^* au rang maximum dans le graphe de précédence, sachant que le rang d'une opération est déterminé par le maximum des rangs de ces prédécesseurs plus un. Il est égal à 1 si l'opération n'a pas de prédécesseur. Nous reviendrons ultérieurement sur le calcul des rangs dans un graphe lors de l'introduction des stations au plus tôt.

La variable de décision y représente le nombre de stations établies.

5.2.3 Formulation des contraintes et objectif

L'objectif est de minimiser le coût total de la ligne. Ce coût est composé du coût engendré par la mise en place de toutes les stations (C étant le coût d'une station), ce qui correspond au premier terme de (5.17), et du coût des unités d'usinage, ce qui est égal au second terme.

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=1}^{m_0} q_b x_{bk} \tag{5.17}$$

Les contraintes (5.18) forcent la sélection des blocs qui n'ont pas d'alternative. Cependant, chacun de ces blocs ne peut être affecté qu'une seule fois.

$$\sum_{k=1}^{m_0} x_{bk} = 1, \quad \forall b \in \mathbf{B} \mid b \in F_0 \tag{5.18}$$

Les contraintes données par (5.19) interdisent l'affectation de plus d'un bloc parmi ceux qui effectuent les mêmes opérations.

$$\sum_{b \in F_s} \sum_{k=1}^{m_0} x_{bk} = 1, \quad \forall s \in \{1, \ldots, v\} \tag{5.19}$$

L'équation (5.20) impose que le nombre total des opérations contenues dans les blocs sélectionnés soit égal au nombre d'opérations qui doivent être effectuées, soit $|\mathbf{N}|$.

$$\sum_{b \in \mathbf{B}} \sum_{k=1}^{m_0} |b| x_{bk} = |\mathbf{N}| \tag{5.20}$$

Afin d'assurer le respect des contraintes de précédence, il suffit d'imposer pour chaque bloc b qui a au moins un prédécesseur, que le nombre d'opérations affectées à des stations strictement antérieures[5] soit supérieur ou égale au nombre d'opérations prédécesseurs du bloc b. Ainsi, c'est l'ensemble des contraintes (5.18), (5.19), (5.20) et (5.21) qui imposent conjointement que toutes les opérations prédécesseurs soient affectées avant les opérations successeurs.

$$\sum_{t \in H(b)} \sum_{s=1}^{s<k} h_{tb} x_{ts} \geq |\Gamma^-(b)| x_{bk}, \quad \forall b \in H, \forall k = 2, \ldots, m_0 \tag{5.21}$$

[5]Une station strictement antérieure est une station qui a un indice strictement inférieur à celui de la station courante.

Pour chaque opération i d'un sous-ensemble d'inclusion $d \in D^{in}$ et pour chaque station potentiellement ouvrable k, nous imposons les équations dictées par (5.22). La partie gauche des contraintes détermine le nombre d'opérations qui appartiennent au sous-ensemble d'inclusion d et qui sont affectées à la station k. Quant à la partie droite, elle exprime le positionnement d'une certaine opération i dans la station k imposant, dans le cas de son affectation, que toutes les autres opérations de ce sous-ensemble soient également assignées à cette station :

$$\sum_{b \in W_d} w_{bd} x_{bk} = |d| \sum_{s \in U_i} x_{sk}, \quad \forall i \in d, \forall d \in D^{in}, \forall k = 1, \ldots, m_0 \tag{5.22}$$

Les inéquations données par (5.23) interdisent l'affectation de l'intégralité des blocs d'un sous-ensemble $E \in D^{ex}$ sur la même station k. Ces contraintes proscrivent le positionnement simultané des variables de décision pour l'affectation de tous les éléments de E à même la station k. Toutefois, nous permettons une affectation des sous-ensembles strictement inclus dans E, c'est-à-dire les sous-ensembles qui ont une cardinalité inférieure ou égale à $|E| - 1$, soit :

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \forall k = 1, \ldots, m_0 \tag{5.23}$$

Les limitations sur le nombre de blocs dans les stations sont exprimées comme suit :

$$\sum_{b \in \mathbf{B}} x_{bk} \leq n_0, \quad k = 1, \ldots, m_0 \tag{5.24}$$

La valeur de la variable y (nombre de stations ouvertes) est déterminée par le plus grand indice de station parmi celles qui sont ouvertes.

$$y \geq k x_{bk}, \quad b \in \mathbf{B}^*, k = m^* + 1, \ldots, m_0 \tag{5.25}$$

m^* est la borne inférieure sur le nombre de stations. De ce fait, toute solution engendre sûrement un coût de $m^* \times C$ nous pouvons donc considérer les stations qu'à partir de $m^* + 1$.

Les contraintes ont pour but de considérer le nombre de stations ouvertes dans la fonction objectif. Il est à signaler que la variable y peut être considérée comme variable réelle au vu de l'objectif 5.17 et des contraintes (5.25). Dans ce cas, la formulation devient en variables mixtes.

Dans la section suivante, nous intégrons des limites sur les indices des stations pour chacun des blocs afin de réduire le nombre d'affectations possibles. Nous présentons, dans ce qui suit, la reformulation des contraintes. Quant aux algorithmes calculant ses limites, nous les présentons en fin de chapitre car ils sont valables pour tous les modèles proposés.

5.2.4 Intégration des limites dans le modèle linéaire

Afin de réduire le nombre de contraintes et la taille du modèle précédemment décrit nous réduisons l'intervalle d'indices des stations de chaque blocs $\forall b \in \mathbf{B}$ initialement donné par $[1, m_0]$ à un intervalle $K(r) = [head_r, tail_r]$. Les variables de décisions pour chaque blocs $b \in \mathbf{B}$ sont crées uniquement pour les valeurs comprises dans l'intervalle $K(r)$. De manière formelle nous écrivons :

$$x_{bk} = \begin{cases} 1 & \text{si un bloc } b \in \mathbf{B} \text{ est affecté à la station } k \\ 0 & \text{sinon} \end{cases}, \quad \forall k = head_r, \ldots, tail_r$$

Les modifications dues à la réduction de cet intervalle sont données par :

Pour la fonction objectif, il suffit de prendre, pour chaque bloc b, les indices des stations correspondant à l'intervalle $K(b)$:

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} q_b x_{bk} \tag{5.26}$$

Pour la reformulation des contraintes (5.18), (5.19) et (5.20) il faut utiliser, pour chaque contrainte, uniquement les variables qui correspondent à l'intervalle $K(b)$:

$$\sum_{k \in K(b)} x_{bk} = 1, \quad \forall b \in F_0 \tag{5.27}$$

$$\sum_{b \in F_s} \sum_{k \in K(b)} x_{bk} = 1, \quad \forall s \in \{1, \ldots, v\} \tag{5.28}$$

$$\sum_{b \in \mathbf{B}} \sum_{k \in K(b)} |b| x_{bk} = |\mathbf{N}| \tag{5.29}$$

Les contraintes de précédence sont exprimées pour chaque station k potentiellement ouvrable[6] et pour chaque bloc ayant au moins un prédécesseur. Ainsi, pour chaque station k nous imposons l'affectation de chaque bloc prédécesseur t dans une station antérieure, c'est-à-dire celle avec un indice s strictement inférieur à k. Toutefois, il faut vérifier que cet indice soit compris dans l'intervalle $K(t) = [head_t, tail_t]$, c'est-à-dire $s \leq tail_t$, or $s < k$ donc $s < min(k, tail_t + 1)$.

$$\sum_{t \in H(b)} \sum_{s=head_t}^{s<min(k,tail_t+1)} h_{tb} x_{ts} \geq |\Gamma^-(b)| x_{bk}, \quad \forall b \in H, \forall k \in K(b) \tag{5.30}$$

[6]Une station potentiellement ouvrable pour un bloc est une station dans laquelle ce bloc peut être assigné, c'est-à-dire une station dont l'indice est compris dans l'intervalle $K(b)$.

Concernant les contraintes d'inclusion, pour chaque opération $i \in d \in D^{in}$, on doit considérer l'union des intervalles de tous les blocs contenant i, car aucune possibilité d'affectation d'un de ces blocs ne doit être omise.

$$\sum_{b \in W_d} w_{bd} x_{bk} = |d| \sum_{s \in U_i} x_{sk}, \quad \forall i \in d, \forall d \in D^{in}, \forall k \in \bigcup_{s \in U_i} K(s) \tag{5.31}$$

En ce qui concerne les contraintes d'exclusion, comme elles sont formulées pour interdire l'affectation de certains blocs à une même station, il suffit de les reformuler uniquement pour les stations pour lesquelles ces blocs sont candidats c'est-à-dire pour les intervalles correspondants.

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \forall k \in \bigcap_{s \in E} K(s) \tag{5.32}$$

Pour les contraintes limitant la capacité des stations, il suffit de prendre, pour chacune des stations, les blocs qui peuvent lui être affectés. Nous les formulons comme suit :

$$\sum_{b \in \{b' \in \mathbf{B} | k \in K(b')\}} x_{bk} \leq n_0, \quad k = 1, 2, \ldots, m_0 \tag{5.33}$$

Pour les contraintes relatives au nombre de stations ouvertes y, il suffit de retenir, pour chaque station k, que les contraintes sur les variables d'affectation des blocs qui peuvent lui être assignés.

$$y \geq k x_{bk}, \quad b \in \mathbf{B}^*, k \in K(b), k > m^* \tag{5.34}$$

Du fait que m^* est une borne inférieure sur le nombre de stations le coût m^*C est assurément inclus dans le coût de toute solution nous pouvons ainsi démarrer de $m^* + 1$ pour ces contraintes.

5.2.5 Réduction de l'arbre d'énumération

L'idée de cette sous-section a pour objectif de réduire l'effort de calcul en favorisant certaines structures parmi celles qui ont le même coût. Plus précisément, nous avons introduit un ε, qui est suffisamment petit pour ne pas affecter l'ordre global donné par le coûts des blocs[7], toutefois il modifie légèrement les coûts afin de faire la différence entre deux affectations identiques du point de vue du coût de la solution finale mais dont les indices des stations non vides sont distincts. La fonction objectif, telle qu'elle a été exprimée dans les sections précédentes, octroie le même coût à l'affectation d'un bloc, et ce à n'importe quelle station. À présent, nous introduisons le ε pour favoriser l'affectation des blocs aux stations ayant les plus petits indices.

[7]En fait l'ordre entre les coûts des blocs avant et après l'ajout des ε doit être exactement le même.

Il en résulte que le coût engendré par l'affectation d'un bloc augmente proportionnellement par rapport à l'indice de la station à laquelle il est assigné. De cette façon, on pénalise les affectations aux stations ayant les plus grands indices donc la charge des stations de sera pas forcément équilibré. Ceci permet de pénaliser les solutions contenant des stations vides entre deux stations qui ne le sont pas. Plus exactement, ceci évite d'ouvrir une station ayant l'indice $k+1$ sans avoir ouvert celle avec l'indice k. Notre nouvel objectif s'exprime alors comme suit :

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} (q_b + k.\varepsilon) x_{bk} \tag{5.35}$$

Cette modification permettra, *a priori*, un gain de temps de calcul mais elle favorise le chargement des stations ayant les petits indices. Mais contrairement aux lignes manuelles, où il est important de lisser la charge, pour les lignes complètement automatisées, il n'est ni nécessaire ni pertinent de chercher à équilibrer la charge de travail. Ainsi nous avons choisi cette structure comme nous aurions pu choisir la structure inverse où les blocs sont plutôt calés à droite.

Le nombre total des solutions identiques peut être très important surtout lorsque le nombre de blocs et m_0 croîent. Dans ce cas, le gain en temps de calcul qui peut être obtenu en utilisant cette modification est potentiellement important.

Dans la suite, nous proposons une seconde formulation en nombres entiers dans le but d'améliorer les performances du modèle précédent. L'idée apportée dans cette nouvelle formulation est de raisonner non plus en fonction des blocs d'opérations mais plutôt en fonction des opérations elles-mêmes quand il s'agit de contraintes qui leur sont intrinsèques, telles que les contraintes de précédence et d'inclusion [BDGL05b].

5.3 Un PLNE orienté opérations

Dans cette section, nous présentons les éléments de modélisation que nous avons utilisés pour exprimer les contraintes du point de vue des opérations (à l'exception des contraintes qui sont intrinsèques aux blocs telles que les contraintes d'exclusion ou des contraintes sur le nombre maximum de blocs par station).

5.3.1 Les éléments de modélisation

Pour chaque opération $i \in \mathbf{N}$ nous introduisons $Q(i)$ désignant l'ensemble des unités d'usinage qui peuvent effectuer l'opération i. Plus formellement, on écrit :

$Q(i) = \{b \in \mathbf{B} | i \in \mathcal{N}(b)\}, \forall i \in \mathbf{N}$

Ainsi, une décision réalisable comprend pour chaque opération i un seul bloc de $Q(i)$.

Nous utilisons dans cette seconde formulation les stations au plus tôt et les stations au plus tard des blocs afin de réduire le nombre de variables binaires x_{rk}.

Les intervalles d'indices des stations pour les opérations sont employés pour formuler directement les contraintes les concernant, ils sont notés $KO(j)$ pour toute opération j. Ces intervalles sont déduits à partir des intervalles d'affectation des blocs, soit :

$$KO(j) = \bigcup_{b \in \{b' | j \in \mathcal{N}(b')\}} K(b), \forall j \in \mathbf{N}$$

5.3.2 Les variables de décision

Nous réutilisons les mêmes variables concernant les décisions relatives à l'affectation des blocs aux stations, soit :

$$x_{bk} = \begin{cases} 1, & \text{si le bloc } b \text{ est affecté à la station } k, \\ 0, & \text{sinon} \end{cases} , \forall b \in \mathbf{B}, \forall k = head_r, \ldots, tail_r$$

Toutefois, pour la décision relative à l'ouverture d'une station, nous introduisons les variables binaires suivantes :

$$y_k = \begin{cases} 1, & \text{si la station } k \text{ est créée (ouverte),} \\ 0, & \text{sinon} \end{cases} , \forall k = 1, \ldots, m_0$$

5.3.3 La formulation des contraintes et objectif

L'objectif est exprimé par l'équation (5.36). De la même façon que dans la première formulation, la double somme sur les variables binaires x_{bk} correspond aux coûts engendrés par les unités d'usinage alors que la somme sur les variables y_k représente le coût associé à l'ouverture des stations.

$$Minimiser \sum_{k=m^*+1}^{m_0} C y_k + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} q_b x_{bk} \tag{5.36}$$

Comme nous l'avons déjà évoqué, le coût m^*C est induit dans toute solution réalisable du fait que m^* est une borne inférieure sur le nombre de stations, nous pouvons de ce fait formuler l'objectif en considérant uniquement les stations postérieurs à m^*.

Lors de la précédente formulation nous en avions besoin des différentes contraintes (5.18), (5.19) et (5.20) afin de modéliser l'unicité de l'exécution des opérations. A présent, il suffit d'imposer à chacune des opérations qu'il n'y ait qu'un seul bloc qui la contienne qui soit sélectionné :

$$\sum_{b \in Q(i)} \sum_{k=head_b}^{tail_b} x_{bk} = 1, \quad \forall i \in \mathbf{N} \tag{5.37}$$

L'inéquation (5.38) est construite pour chaque couple d'opérations appartenant à D^{or}. Nous avons, en ce sens, simplifié leur modélisation car dans le modèle précédent une contrainte a été exprimée pour chaque bloc b ayant au moins un prédécesseur. À présent, au lieu de raisonner en fonction de chaque bloc (voir 5.30), nous traduisons la formulation pour tous les couples d'opérations (i, j). En fait, il suffit d'utiliser les ensembles $Q(i)$ et $Q(j)$ et d'imposer l'ordre d'exécution décrit par la contrainte (i, j), soit i avant j. Ce qui revient à forcer les blocs effectuant j à être assignés à une station strictement postérieure[8] à celle qui effectuera l'opération i.

$$\sum_{b \in Q(i)} \sum_{l=head_b}^{min(k-1, tail_b)} x_{bl} \geq \sum_{b \in Q(j)} x_{bk}, \quad \forall (i,j) \in D^{or}, \forall k \in KO(j) \tag{5.38}$$

Pour modéliser les contraintes d'inclusion nous formulons une contrainte pour toute paire d'opérations (i, j) appartenant au même sous-ensemble d'inclusion et ceci pour chaque station possible, comme indiqué dans (5.39). La partie droite de l'équation correspond à l'affectation de l'opération j dans la station k. Pour que ces contraintes soient respectées nous imposons aux variables déterminant l'affectation de l'opération i à être égales à celles de j. Ainsi, si un des deux opérations est affectée à la station k alors la seconde doit l'être également et vice versa[9].

$$\sum_{b \in Q(i)} x_{bk} = \sum_{r \in Q(j)} x_{rk}, \quad \forall (i,j) \subseteq d, \forall d \in D^{in}, \forall k \in KO(j) \tag{5.39}$$

Du fait que par définition les contraintes d'exclusion sont intrinsèques aux blocs, nous réutilisons la formulation introduite dans le modèle précédent (orienté bloc).

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \quad \forall k \in \bigcap_{b' \in E} K(b') \tag{5.40}$$

Chaque station qui peut être ouverte ne doit pas contenir plus de n_0 blocs. Ainsi, après l'introduction des variables relatives à l'ouverture d'une station, ces contraintes sont exprimées au moyen de (5.33).

$$\sum_{b \in \mathbf{B}} x_{bk} \leq n_0, \quad k = 1, \ldots, m_0 \tag{5.41}$$

[8]Car les indices des stations sont numérotées de 1 à m_0.

[9]Comme l'ordre de l'affectation des blocs et par conséquent des opérations n'est pas prise en compte dans la formulation, nous passons outre le fait que l'opération i soit affecté avant j ou que j le soit avant. En fait l'équation (5.39) va forcer les variables correspondantes à l'affectation de l'opération qui n'est pas encore affectée à être égales à celles de l'opération qui l'a déjà été

Dés lors qu'un bloc est affecté à une station, la variable indiquant l'ouverture de cette station est positionnée à un en ajoutant les contraintes suivantes :

$$y_k \geq x_{bk}, \quad \forall k \geq m^* + 1, \forall b \in \{r \mid r \in \mathbf{B}, k \in K(r)\} \tag{5.42}$$

Nous utilisons les contraintes décrites par (5.43) afin de générer des solutions dans lesquelles les stations sont créées dans l'ordre lexicographique, allant de 1 à m_0. Ainsi, dans une solution si la première station est saturée[10] alors la prochaine station à ouvrir aura l'indice suivant, soit égale à 2 et ainsi de suite. En fait, en l'absence de ces contraintes, l'indice de la prochaine station à ouvrir aura une valeur arbitraire comprise entre 1 et m_0 à l'exception des indices des stations qui ont déjà été ouvertes. Ces contraintes jouent le même rôle que la technique utilisant une constante ε dans l'objectif (voir l'expression (5.35)).

$$y_{k-1} - y_k \geq 0, \quad k = m^* + 2, \ldots, m_0 \tag{5.43}$$

5.4 Réduction du nombre de variables

Cette étape consiste à réduire, pour chaque bloc, l'intervalle définissant la plage d'indices des stations dans lesquelles ce bloc peut être affecté. Dans le modèle (5.17)-(5.25) nous avons considéré pour chacun des blocs toutes les affectations possibles (c'est-à-dire toutes les stations potentiellement ouvrables). Ainsi, cet intervalle est égale initialement à $[1, m_0]$. Nous pouvons pourtant affiner cet intervalle en nous basant sur certaines contraintes qui vont interdire certaines affectations. C'est pourquoi nous avons introduit la notion d'intervalle réduit $K(b)$ pour chaque bloc $b \in \mathbf{B}$.

L'idée que nous proposons à présent est d'étudier la consistance aux bornes de ces intervalles $K(b) = [head_b, tail_b]$ tel que $head_b \geq 1$ et que $tail_b \leq m_0$. L'objectif est de réduire au maximum pour chaque bloc b l'intervalle des indices de stations possible à un intervalle. Nous proposons d'appliquer un algorithme qui exploite la structure des contraintes pour déduire des impossibilités d'affectations des blocs dans certaines stations. De ce fait, nous détectons les indices des stations qui mèneront à des solutions non réalisables, ce qui nous permet de réduire les intervalles et d'éliminer ensuite les variables binaires correspondantes à ces affectations[11].

Pour ce faire, nous proposons trois algorithmes, le premier exploite uniquement les contraintes de précédence tandis que le second intègre en plus les contraintes d'inclusion pour affiner les limites des intervalles $K(b)$. Le dernier algorithme est plus complexe dans le sens où il fait intervenir l'ensemble des contraintes.

[10] C'est-à-dire qu'il devient impossible de lui affecter un bloc supplémentaire sans la violation d'une des contraintes.

[11] Ces solutions engendrées par l'affectation des blocs aux stations qui violent certaines contraintes sont écartées et permettent ainsi de réduire l'espace de recherche

5.4.1 Algorithme 1

Nous introduisons le terme de «station au plus tôt», pour désigner la limite $head_b$ du bloc b et «station au plus tard», pour la limite $tail_b$. Les valeurs contenues dans cet intervalle représentent les indices des stations que nous allons considérer pour la création de variables de décision correspondantes. Par souci de clarté, nous employons le terme limite pour tout $head_b$ ou $tail_b$, au lieu du terme borne afin de ne pas créer d'ambiguïté due à l'association du mot borne à la borne inférieure sur la fonction objectif ou sur le nombre de stations.

Le principe de cet algorithme est d'exploiter les contraintes de précédence qui existent entre les opérations pour en déduire des limites pour les blocs qui les contiennent. Par exemple, s'il y a une opération qui doit précéder une opération appartenant au bloc b, alors le bloc successeur ne pourra pas être affecté à la première station, son intervalle peut donc commencer à partir de l'indice 2 indiquant la deuxième station.

Cet algorithme a une complexité polynomiale ce qui le rend très intéressant d'un point de vue pratique.

Rappelons qu'afin d'obtenir les rangs des opérations il faut appliquer ce qui suit :

$$rank(j) = \begin{cases} \max\{rank(i) \mid i \in \mathbf{N}, (i,j) \in G^{or}\} + 1, \text{ si } j \text{ a des prédécesseurs} \\ 1 \text{ sinon.} \end{cases}$$

Le rang d'un bloc est déterminé par le maximum des rangs des opérations qui le composent. La station au plus tôt d'un bloc b correspond à son rang dans le graphe de précédence.

Algorithme 1
Begin

Calculer les rangs des opérations

For all $b \in \mathbf{B}$
$head_b := \mathbf{max}\ \{rank(j), j \in \mathcal{N}(b)\}$
End For

End

Pour obtenir les stations au plus tard, il faut d'abord inverser le graphe de précédence G^{or}, on obtient un graphe G^{or}_{inv}. Ensuite, il faut calculer les rangs des opérations dans ce graphe en appliquant la règle suivante :

$$\overline{rank}(j) = \begin{cases} \min\{\overline{rank}(i) \mid i \in \mathbf{N}, (i,j) \in G^{or}_{inv}\} - 1; \text{ si } j \text{ a un prédécesseur} \\ m_0, \text{ sinon} \end{cases}$$

Ensuite, pour obtenir les $tail_b$ pour chaque bloc b, il suffit de remplacer dans l'Algorithme 1 l'instruction calculant les *head* par la suivante :

$tail_b := \mathbf{min}\{\overline{rank}(j), \forall j \in \mathcal{N}(b)\}$

Un exemple

Nous reprenons l'exemple que nous avons introduit dans la section 4.2 pour appliquer l'Algorithme 1. Nous rapportons les valeurs des rangs dans le tableau 5.1 et celles des stations au plus tôt et au plus tard dans le tableau 5.2.

op	$rank(j)$	$\overline{rank}(j)$	op	$rank(j)$	$\overline{rank}(j)$
1	1	$m_0 - 1$	8	3	$m_0 - 1$
2	1	$m_0 - 1$	9	1	$m_0 - 2$
3	1	$m_0 - 3$	10	2	$m_0 - 1$
4	2	$m_0 - 2$	11	4	m_0
5	3	$m_0 - 1$	12	1	$m_0 - 2$
6	1	$m_0 - 3$	13	2	$m_0 - 1$
7	2	$m_0 - 2$	14	4	m_0

TAB. 5.1 – Les rangs des opérations

bloc	opérations	$head_b$	$tail_b$	bloc	opérations	$head_b$	$tail_b$
b_1	1,3,6	1	$m_0 - 3$	b_{12}	1,2,4,7	1	$m_0 - 3$
b_2	2,4,7	2	$m_0 - 2$	b_{13}	3,6	1	$m_0 - 3$
b_3	1,2,3,6	1	$m_0 - 3$	b_{14}	1,6	1	$m_0 - 3$
b_4	4,7	2	$m_0 - 2$	b_{15}	1	1	$m_0 - 1$
b_5	9,12	1	$m_0 - 2$	b_{16}	2	1	$m_0 - 1$
b_6	10,13	2	$m_0 - 1$	b_{17}	1,2	1	$m_0 - 1$
b_7	5,8	3	$m_0 - 1$	b_{18}	3	1	$m_0 - 3$
b_8	11	4	m_0	b_{19}	6	1	$m_0 - 3$
b_9	14	4	m_0	b_{20}	5	3	$m_0 - 1$
b_{10}	2,3,6	1	$m_0 - 3$	b_{21}	8	3	$m_0 - 1$
b_{11}	1,4,7	2	$m_0 - 2$				

TAB. 5.2 – Les indices de station au plus tôt et au plus tard

5.4.2 Algorithme 2

Nous introduisons un second algorithme pour le calcul des stations au plus tôt et au plus tard. Dans cette version, nous exploitons non seulement les contraintes de précédence mais également les contraintes d'inclusion. De façon similaire à l'Algorithme 1, cet algorithme est d'abord utilisé pour l'obtention des stations au plus tôt $head_b$ puis il est adapté, après quelques substitutions, pour le calcul des stations au plus tard $tail_b$.

La première étape de l'algorithme consiste à initialiser, pour chaque opération j de $\mathbf{N}$ sa limite $ub(j)$ avec la valeur correspondante à son rang dans le graphe de précédence.

La variable n_{imp} est également initialisée, elle sert à détecter la présence éventuelle d'un cycle auquel cas l'algorithme est terminé en indiquant l'absence de solutions réalisables.

Algorithme 2

Begin

Step1. **For all** $(j \in \mathbf{N})$
$ub(j) := rank(j)\,;$
$n_{imp} := 0\,;$

Step2. Set $imp_{ub} := 0\,;$

Step3. **For all** $(d \in D^{in})$
$um := max\{ub(j) \mid j \in d\}\,;$
For all$(j \in d)$
$ub(j) := um\,;$

Step4. **For all** $(j \in \mathbf{N})$
$u_j := max\{ub(i) + 1 \mid, \forall i \in \mathbf{N}, (i,j) \in D^{or}\}\,;$
If$(u_j > ub(j))$ **Then**
$ub(j) := u_j\,;$
$imp_{ub} := imp_{ub} + 1\,;$
EndIf

Step5. Set $n_{imp} := n_{imp} + 1\,;$
If$(n_{imp} > |\mathbf{N}|)$ **Then**
stop (il n'y a pas de solutions réalisables)
EndIf
If$(imp_{ub} > 0)$ **Then**
goto **Step2**;
EndIf

Step6. **For all** $(b \in \mathbf{B}\ )$
Set $head_b := \mathbf{max}\{ub(j) \mid j \in \mathcal{N}(b)\}\,;$

Step7. Set $m^* := \mathbf{max}\{ub(j) \mid j \in \mathbf{N}\}\,;$

End

Quant à la variable imp_{ub} elle compte le nombre d'améliorations effectuées dans une itération de l'algorithme.

La troisième étape (step3) consiste à ramener le rang de toutes les opérations, pour chaque sous-ensemble d'inclusion d au plus grand rang dans d. En effet, par définition, les opérations de chaque ensemble d doivent être affectées à la même station. L'étape 4 consiste à obtenir les limites des stations pour les opérations en se basant sur les contraintes de précédence. L'étape 5 consiste à détecter une amélioration des limites auquel cas il faut retourner à l'étape 2. Lorsqu'il y a plus de $|\mathbf{N}|$ améliorations le graphe de précédence contient un cycle et le problème correspondant est insolvable. L'étape 6 permet de déduire les limites pour les blocs à partir de celles pour les opérations. L'étape 7, permet d'obtenir une borne inférieure sur le nombre de stations. Elle est obtenue simplement en calculant le maximum des limites de l'ensemble des opérations (ou blocs).

L'amélioration par rapport au précédent algorithme d'imposer aux blocs ayant des opé-

rations liées par des contraintes d'inclusions d'avoir les mêmes limites. En d'autres termes, si un bloc b contient une opération i et un bloc b' contient une autre opération j tel que (i, j) appartiennent à D^{in} alors b et b' doivent avoir le même interval, c'est-à-dire que $K(b) = K(b')$. La complexité reste polynomiale, car au pire des cas, le graphe de précédence est parcourue $|\mathbf{N}|$.

5.4.3 Algorithme 3

Cet algorithme prend en compte toutes les contraintes. Pour l'expliquer nous utilisons la notion de niveau g (voir Algorithme 3). *A priori*, la répartition des opérations en niveaux correspond aux rangs du graphe de précédence mais la prise en compte des autres contraintes peut amener à obtenir plusieurs niveaux par rang. Le principe est d'identifier pour chaque niveau un sous-ensemble maximal de blocs (B^{max}) qui vérifie l'ensemble des contraintes, à savoir : les contraintes de précédence, inclusion, exclusion et n_0 (le nombre maximum de blocs par station). Le sous-ensemble maximal de blocs se voit alors attribuer g comme station au plus tôt. L'algorithme s'arrête lorsque g dépasse le nombre maximum de stations m_0. Aussi, si après la boucle *Repeat while* il subsiste des opérations dont les blocs n'ont pas pu être sélectionnés alors le problème est infaisable.

Comme dans les deux cas précédents, l'algorithme 3 est d'abord utilisé pour l'obtention des stations au plus tôt $head_b$ pour tout bloc b. Il est ensuite adapté pour l'obtention des stations au plus tard $tail_b$ en effectuant les substitutions suivantes :

1. $g = m_0$ au lieu de $g = 1$, à l'étape de l'initialisation
2. $g \geq 1$ au lieu de $g \leq m_0$, dans *Repeat while*
3. $g = g - 1$ au lieu de $g = g + 1$
4. $\Gamma^-(b)$ est remplacé par l'ensemble $i \in \mathbf{N} \backslash \mathcal{N}(b)$ tel que $(j, i) \in D^{or}$ pour tout $j \in \mathcal{N}(b)$
5. $B_1 := B_1 \cup \{b \in \mathbf{B} - B_1 | head_b > tail_b\}$ à mettre après le *End Repeat* ;

L'inconvénient majeur de cet algorithme est incontestablement sa complexité. En effet, la recherche d'un sous-ensemble maximal $B^{max} \subseteq \mathbf{B}_1$ nécessite un parcours total de toutes les combinaisons possibles de sous-ensembles, c'est-à-dire : $\sum_{k=1}^{|\mathbf{B}_1|} C^k_{|\mathbf{B}_1|}$, tel que : $C^k_n = \frac{n!}{k!(n-k)!}$. Cette complexité ne permet pas d'employer l'algorithme de façon efficace. Il est évident qu'il sera moins performant sur des instances de grande taille, c'est pourquoi nous avons opté pour l'implémentation des deux premiers algorithmes uniquement.

Algorithme 3
Begin
$g := 1$; $B_1 := \mathbf{B}$; $B_2 := \varnothing$; $D_0 := D^{in}$;
For all $b \in \mathbf{B}$
$M_1(b) := \Gamma^-(b)$; $M_2(b) := \varnothing$;
Repeat while $(B_1 \neq \varnothing \wedge g \leq m_0)$
$B^{max} = \varnothing$;
For all $(B_{temp} \subseteq B_1)$
$\forall b \in B_{temp}$
IF$\Big(\big(|B_{temp}| > |B^{max}|\big) \bigwedge \big(M_1(b) = \varnothing\big) \bigwedge$
$\big((\forall d \in D_0, w_{bd} = 0) \vee (\exists B' \subseteq B^+, b \in B', d \subseteq \bigcup_{\mathcal{N}(b) \in B'} b, |B'| \leq n_0 \wedge B' \notin D^{ex})\big)$
$\bigwedge \Big((M_2(b) = \varnothing) \vee (\exists B' \subseteq B_2, M_2(b) \subseteq \bigcup_{\mathcal{N}(b) \in B'} b, |B'| \leq n_0, B' \notin D^{ex})\Big)\Big)$
Then $B^{max} := B_{temp}$
End For
If $(B^{max} = \varnothing)$
Then break;
For all $b \in B^{max}$,
$head_b = g$;
End For
$B_1 := B_1 - B^{max}$; $B_2 := B^{max}$; $g := g + 1$;
$D_0 := D_0 - \{d \in D_0 \mid d \subseteq \bigcup_{b \in B^{max}} \mathcal{N}(b)\}$;
For all $b \in B_1$
$M_2(b) := M_1(b) \cap (\bigcup_{b \in B^{max}} \mathcal{N}(b))$;
$M_1(b) := M_1(b) - M_2(b)$;
End For
End Repeat

If$(B_1 \neq \varnothing)$ **Then**
$\mathbf{B} := \mathbf{B} - B_1$;
If$(\bigcup_{b \in \mathbf{B}} \mathcal{N}(b) \neq \mathbf{N})$
Then return **False**
Else
For all $E \in D^{ex}$
If$(E \cap B_1 \neq \varnothing)$
$D^{ex} := D^{ex} - E$;
End If
End If
End If
Return True
End

5.5 Conclusion

Dans ce chapitre, nous avons proposé plusieurs modèles en affinant la formulation des contraintes afin d'exploiter au mieux la structure du problème. Pour la résolution des modèles proposés, nous avons employé les outils d'ILOG. En particulier, nous utilisons Cplex pour la programmation linéaire en nombres entiers et Solver pour la programmation par contraintes. Malgré les performances de ces outils, une bonne modélisation reste un facteur prépondérant quant à l'efficacité de résolution.

Nous avons apporté plusieurs modélisations du problème TLBP/B-P. Nous avons suggéré trois modèles, le premier est générique quant aux deux autres ils sont basés sur la programmation linéaire en nombres entiers.

Dans le cadre de la démarche basée sur la programmation par contraintes, nous avons intégré une borne inférieure basée sur la relaxation linéaire et suggéré également une autre borne obtenue par relaxation en un problème de set partitioning particulier. L'approche s'appuie également sur l'emploi de règles de dominance pour réduire l'espace des solutions énumérées et sur la propagation des contraintes d'exclusion et de précédence.

Pour la programmation linéaire en nombres entiers, nous avons proposé deux formulations : la première est orientée blocs, la seconde est orientée opérations. Le second modèle intègre également des coupes afin de réduire l'espace de recherche en favorisant certaines structures de solutions. Ces coupes ont été introduites pour remplacer la modification introduisant une constante ε dans l'objectif du premier modèle (voir sous-section 5.2.5).

De plus, nous avons fourni plusieurs algorithmes afin de réduire le nombre de variables diminuant ainsi la taille du modèle. Nous soulignons le caractère générique des algorithmes proposés car ils s'appliquent à l'ensemble des modèles présentés.

Dans le chapitre suivant, nous rapportons les résultats des tests expérimentaux. Ainsi, nous évaluons les performances des formulations proposées. De plus, nous y montrons l'apport des améliorations proposées. Enfin, nous retiendrons l'approche la plus performante pour étudier ensuite l'impact de certaines caractéristiques, *a priori*, influentes sur les performances des modèles.