# Modélisation d'un *flexible job-shop* complexe avec batches et opérateurs pour la production d'acier avec Hexaly

Léa Petit-Jean Genat

Hexaly, 251 Boulevard Pereire, Paris, France
lpetitjean@hexaly.com

**Mots-clés** : *ordonnancement, solveur, modélisation, batches, opérateurs, industrie.*

## 1 Introduction

La production d'acier exige une gestion complexe des ressources similaire à un problème de *flexible job-shop*. Chaque commande est constituée d'une séquence d'opérations à réaliser sur différentes machines, avec une flexibilité sur l'équipement. Certaines machines fonctionnent par batches et uniquement lorsque des opérateurs sont disponibles.

Hexaly est un solveur d'optimisation mathématique basé sur différentes techniques de recherche opérationnelle, combinant des méthodes exactes, telles que la programmation linéaire, non linéaire et par contraintes, et heuristiques, comme la recherche locale [1].

Les sections suivantes décrivent le problème métier et une approche de modélisation, exploitant la flexibilité d'Hexaly, pour résoudre efficacement cet ordonnancement de production d'acier.

## 2 Problème métier

Les commandes de produits en acier doivent être produites en respectant des dates de disponibilité (*release date*) et d'échéance (*due date*). Les produits étant variés, chaque commande (*job*) requiert une séquence d'opérations spécifiques sur différents types de machines.

Chaque type de machine peut inclure plusieurs machines, et chaque type est rattaché à un groupe. Les groupes de machines sont associés à un ou plusieurs opérateurs, dont les disponibilités définissent l'emploi du temps du groupe. Si un groupe ne dispose pas d'opérateurs dédiés, la gestion de la disponibilité se fait au niveau de chaque machine, incluant des périodes de maintenance planifiées.

La plupart des machines sont limitées à une seule tâche à la fois, mais d'autres, tels que des fours, fonctionnent par batches. Un batch ne peut contenir qu'un seul type de produit, et la somme des poids des produits au sein d'un même batch ne doit pas excéder une limite.

Certaines opérations sont déjà planifiées et il est impossible de changer leur machine et leur ordre de passage. Le temps d'exécution des opérations dépend de la machine utilisée et peut inclure des délais de transport entre les opérations. Certaines machines nécessitent également des temps de configuration (*setup*) supplémentaires lorsqu'un changement de caractéristiques du produit, comme la taille, survient entre deux opérations consécutives.

L'objectif est d'attribuer une machine et une plage d'exécution à chaque opération afin de minimiser les délais de production.

## 3 Modélisation du problème

La modélisation est réalisée en séparant les décisions liées aux opérations séquentielles et de batches. Pour les opérations séquentielles, l'ordre sur chaque machine est décidé grâce à des

variables de listes [1]. L'ordre des opérations déjà planifiées peut donc être fixé facilement :

```
sequence[0...nbMachines] <- list(nbOperations);
for [m in 0...nbMachines][i in 0...fixedSequence[m].count()]
    constraint sequence[m][i] == fixedSequence[m][i];
```

Des variables d'intervalles sont utilisées pour représenter les plages d'exécution des opérations. La disponibilité des opérateurs est représentée par un tableau de booléens, égaux à 1 si l'opérateur est disponible, 0 sinon. Le temps réel d'exécution d'une opération correspond au nombre d'unités de temps où l'opérateur est disponible pendant sa plage d'exécution.

Dans le formalisme de modélisation d'Hexaly, cela se formule aisément à l'aide de l'opérateur somme variadique, qui calcule la somme des valeurs renvoyées par une lambda fonction. Dans ce cas, la lambda fonction retourne les valeurs issues du tableau de disponibilités. La somme est calculée durant la plage d'éxécution définie par la variable d'intervalle, soit pour chaque unité de temps $t$ comprise dans cet intervalle. Le temps d'exécution dépend de la machine utilisée, et l'opérateur *find* permet d'identifier la machine sur laquelle l'opération est réalisée.

```
operation[0...nbOperations] <- interval(0, maxEnd);
machine[i in 0...nbOperations] <- find(sequence, i);
for [g in 0...nbGroups][i in groupOperations[g]]
    constraint sum(operation[i], t => disponibility[g][t]) == processingTime[machine[i]][i];
```

Les opérations des batches sont déterminées à l'aide de variables de sets [2]. Pour restreindre un batch à un seul type de produit, deux opérateurs sont utilisés. D'une part l'opérateur *distinct*, qui extrait l'ensemble non ordonné des valeurs renvoyées par une lambda fonction appliquée à chaque élément d'un tableau ou d'une collection, comme des variables de listes ou de sets. Dans ce contexte, il permet d'identifier tous les types de produits associés aux opérations contenues dans le set représentant le batch. D'autre part, l'opérateur *count* permet de dénombrer les éléments d'un tableau ou d'une collection. Enfin, l'opérateur somme variadique est utilisé pour calculer le poids total des éléments présents dans la variable de set.

```
batchContent [0...nbBatchs] <- set(nbOperations);
for [b in 0...nbBatchs] {
    constraint count(distinct(batchContent[b], i => type[i])) <= 1;
    constraint sum(batchContent[b], i => weight[i]) <= maxWeight;
}
```

## 4 Résultats et conclusion

Le formalisme de modélisation ensembliste et non-linéaire d'Hexaly permet de formuler simplement et de façon compacte les contraintes complexes du problème, comme les batches et les opérateurs. L'optimisation est réalisée pour une usine et une période d'un mois, pour environ 2000 opérations dont 200 opérations de batches. Des solutions de qualité sont trouvées en 5 minutes de calcul.

## Références

[1] F. Gardi, T. Benoist, J. Darlay, B. Estellon, et R. Megel. *Mathematical Programming Solver Based on Local Search*, Wiley, 2014

1. Variable de décision dont la valeur est une permutation d'un sous-ensemble de $\{0, \ldots, n-1\}$
2. Variable de décision dont la valeur est un sous-ensemble de $\{0, \ldots, n-1\}$