Optimization of foundry production processes

Anna Kelemen
Supervisor: Alpár Jüttner

2024 December

1. Introduction

Casting is the primary process carried out in a foundry, where melted metal is poured into a mold, where after cooling and solidifying, it reaches its new form. In order to perform this task, various preparations are needed - the molds of the workpieces have to be prepared and assembled. When casting objects with internal cavities, it is also necessary to prepare a core which can be inserted into the mold. These molds (and cores) then have to be assembled in order to produce the desired shape when metal is poured into them.

The variety of mold and core types, preparation and assembly times, alloy types, deadlines, and the capacities of working shifts make manually scheduling these tasks difficult. This semester, my project focused on modelling the production process for mold and core making, assembling, and casting of various objects using an integer programming (IP) formulation.

2. IP model

Scheduling the tasks and assigning the objects to casting shifts represent a combined problem of bin packing and scheduling. When scheduling the tasks, the first step is deciding what to consider optimal. In the casting process, due to the high costs of metal melting, it is important to minimize the number of casting rounds. To reduce operational inventory and due to space limitations, it is also important to complete each task as close to its deadline as possible without exceeding it.

Suppose that there are two types of shifts: one used only for casting and the other for all remaining tasks. During a casting shift, it is possible to complete up to K rounds of casting, with each round processing at most 4 tonnes of the same type of metal. In an assembly shift, it is possible to finish at most L tasks, while the tasks shouldn't overlap and their total length shouldn't exceed the amount of time available in the shift. There are specified waiting times between the consecutive processes (i.e. mold making and assembly, core making and assembly, assembly and casting) of a workpiece during which no further work can be done on that object.

Considering all these constraints, we can formulate an IP that will provide the optimal scheduling of the tasks, where using only $\epsilon \geq 0$ times more rounds of casting than necessary, we can achieve the least waiting time possible. Let $I = \{1, ..., n\}, J_1 = \{1, ..., C\}, J_2 = \{1, ..., A\}, \kappa_1 = \{1, ..., K\}$, $\kappa_2 = \{1, ..., L\}$, where A and C denote the number of assembly and casting shifts and n is the number of objects needing to be cast. First, we solve the following IP excluding all but (2) and (10)-(16), minimizing the objective $\sum_{j=1}^{C}\sum_{k=1}^{K} v_{jk}$. This way, we know the minimum number of G casting rounds required for producing every product. Then we solve the IP once more, now including all constraints and choosing the value of ϵ.

For the sake of completeness, the whole IP formulation is included in the appendix, and the following paragraphs will include a short explanation of the constraints and variables.

Every object has its deadline, weight, alloy type, mold and core making time, assembly time and the required waiting times between the different tasks given as $D_i, w_i, t_i, m_i, c_i, a_i,\ mw_i, cw_i, aw_i$. We are given the type of each shift — either casting or assembly — as well as their start and end times: CS_j, CE_j for the casting and AS_j, AE_j for the assembly shifts. Using this information, we can determine for each object i the last casting shift j in which the object can be cast such that it still can be completed before its deadline D_i.

z_{ijk}: binary variable, 1 if the casting of the i^{th} object happens in the k^{th} round of the j^{th} casting shift
X^m_{ijk}: binary variable, 1 if making the mold of the i^{th} object is the k^{th} task of the j^{th} assembly shift

X^c_{ijk}: binary variable, 1 if making the core of the i^{th} object is the k^{th} task of the j^{th} assembly shift
X^a_{ijk}: binary variable, 1 if assembling the i^{th} object is the k^{th} task of the j^{th} assembly shift
v_{jk}: binary variable, 1 if there is an object, whose casting happens in the k^{th} round of the j^{th} casting shift
u_{ij}: binary variable, 1 if the casting of the i^{th} object happens in the j^{th} casting shift
mt_{jk}: the code representing the type of metal cast in the k^{th} round of the j^{th} casting shift
S_{jk}: the start time of the k^{th} task of the j^{th} assembly shift
T_{jk}: the duration of the k^{th} task of the j^{th} assembly shift
Y^m_i: the start time of the mold making for the i^{th} object
Y^c_i: the start time of the core making for the i^{th} object
Y^a_i: the start time of the assembly of the i^{th} object

(2) ensures that every object is cast before its deadline and (1) guarantees that no more than ϵ-times the essential number of casting rounds are used. (3)-(9) ensure that for each object, every task is completed and the required waiting times are kept between them. (10) is for keeping the weight limit in each casting round and (11)-(12) assigns the type of metal to be cast in each round. (13)-(14) and (15)-(16) are responsible for setting the values of v_{jk} and u_{ij}.

(17)-(19) allocates the tasks of the j^{th} shift into the time interval assigned to that shift and (20)-(31) ensure that if the task of the mold-making/core-making/assembling of the i^{th} object is assigned to be the k^{th} task of the j^{th} shift, then the start time of that task equals to the start time of the k^{th} task of the j^{th} shift and their duration is the same. We achieve this by choosing a sufficiently large N that ensures that the two variables corresponding to these in an inequality are forced to be equal when required or remain independent when there is no need for them to be equal.

The objective is to minimize the sum of the time differences between the deadline and the completion time of each task for every object, which is exactly what is needed:

$$\begin{aligned} obj = \sum_{i=1}^{n} \Bigg[\left(D_i - \sum_{j=1}^{C} u_{ij} \cdot CE_j \right) + \left(\sum_{j=1}^{C} u_{ij} \cdot CS_j - aw_i - a_i - Y^a_i \right) + \\ + (Y^a_i - mw_i - m_i - Y^m_i) + (Y^a_i - cw_i - c_i - Y^c_i) \Bigg] \end{aligned}$$

2.1. Implementation

This semester I used the CP-SAT solver from Google's OR-Tools library in Python to solve the presented IP. The initial testing shows that for small values of n ($n \leq 10$), it delivered results relatively quickly (under 1 minute) and when used to determine the minimum required casting rounds only, it even performed well for larger values ($n \leq 30$). However, when tested on the three-month order population data from a foundry, the running time was too slow.

3. Future goals

The aim this semester was to establish a model, and the goal for future projects is to explore the theoretical background in more depth and speed up the IP-solving algorithm. This could be achieved either by using a two-phase approach, separating the scheduling of casting and assembly tasks or by applying cutting plane, relaxation or other heuristic methods for a faster algorithm.

References

[1] P. Beeley. *Foundry Technology.* Butterworth-Heinemann, 2001.

A. IP

$$\min\{obj\}$$

$$\sum_{j=1}^{C}\sum_{k=1}^{K} v_{jk} \le (1+\epsilon)\cdot G \tag{1}$$

$$\sum_{j=1}^{d_i}\sum_{k=1}^{K} z_{ijk} = 1 \qquad \forall i \in I \tag{2}$$

$$Y_i^m + m_i + mw_i \le Y_i^a \qquad \forall i \in I \tag{3}$$

$$Y_i^c + c_i + cw_i \le Y_i^a \qquad \forall i \in I \tag{4}$$

$$Y_i^a + a_i + aw_i \le D_i \qquad \forall i \in I \tag{5}$$

$$\sum_{j=1}^{A}\sum_{k=1}^{L} X_{ijk}^m = 1 \qquad \forall i \in I \tag{6}$$

$$\sum_{j=1}^{A}\sum_{k=1}^{L} X_{ijk}^c = 1 \qquad \forall i \in I \tag{7}$$

$$\sum_{j=1}^{A}\sum_{k=1}^{L} X_{ijk}^a = 1 \qquad \forall i \in I \tag{8}$$

$$\sum_{i=1}^{n} X_{ijk}^m + X_{ijk}^c + X_{ijk}^a \le 1 \qquad \forall j \in J_2, \forall k \in \kappa_2 \tag{9}$$

$$\sum_{i=1}^{n} z_{ijk}\cdot w_i \le 4000 \qquad \forall j \in J_1, \forall k \in \kappa_1 \tag{10}$$

$$mt_{jk} \le t_i + (1 - z_{ijk}) \qquad \forall j \in J_1, k \in \kappa_1, i \in I \tag{11}$$

$$mt_{jk} \ge t_i - (1 - z_{ijk}) \qquad \forall j \in J_1, k \in \kappa_1, i \in I \tag{12}$$

$$v_{jk} \ge z_{ijk} \qquad \forall j \in J_1, k \in \kappa_1, i \in I \tag{13}$$

$$v_{jk} \le \sum_{i=1}^{n} z_{ijk} \qquad \forall j \in J_1, k \in \kappa_1 \tag{14}$$

$$u_{ij} \ge z_{ijk} \qquad \forall i \in I, j \in J_1, k \in \kappa_1 \tag{15}$$

$$\sum_{j=1}^{C} u_{ij} = 1 \qquad \forall i \in I \tag{16}$$

$$S_{jk} \ge AS_j \qquad \forall j \in J_2, k \in \kappa_2 \tag{17}$$

$$S_{jk} + T_{jk} \le AE_j \qquad \forall j \in J_2, k \in \kappa_2 \tag{18}$$

$$S_{jk} + T_{jk} \le S_{j(k+1)} \qquad \forall j \in J_2, k \in \kappa_2 \tag{19}$$

$$Y_i^m - M_{jk} \le N\cdot(1 - X_{ijk}^m) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{20}$$

$$M_{jk} - Y_i^m \le N\cdot(1 - X_{ijk}^m) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{21}$$

$$m_i - T_{jk} \le N\cdot(1 - X_{ijk}^m) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{22}$$

$$T_{jk} - m_i \le N\cdot(1 - X_{ijk}^m) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{23}$$

$$Y_i^c - M_{jk} \le N\cdot(1 - X_{ijk}^c) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{24}$$

$$M_{jk} - Y_i^c \le N\cdot(1 - X_{ijk}^c) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{25}$$

$$c_i - T_{jk} \le N\cdot(1 - X_{ijk}^c) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{26}$$

$$T_{jk} - c_i \le N\cdot(1 - X_{ijk}^c) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{27}$$

$$Y_i^a - M_{jk} \le N\cdot(1 - X_{ijk}^a) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{28}$$

$$M_{jk} - Y_i^a \le N\cdot(1 - X_{ijk}^a) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{29}$$

$$a_i - T_{jk} \le N\cdot(1 - X_{ijk}^a) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{30}$$

$$T_{jk} - a_i \le N\cdot(1 - X_{ijk}^a) \qquad \forall i \in I, j \in J_2, k \in \kappa_2 \tag{31}$$