Optimization of foundry production processes 2

Anna Kelemen
Supervisor: Alpár Jüttner

2025 May

1. Introduction

Continuing the work from last semester, our goal remains to optimize the production process of a foundry. Previously, we developed an integer programming (IP) model, that, when solved, produced an optimal task schedule. However, solving that formulation turned out to be too slow.

In this project, instead of concentrating on the IP formulation and trying to achieve an integer solution, we construct a different model and obtain integer results by deriving them from the optimal solution of its linear programming (LP) relaxation. We solve the LP relaxation of the problem using column generation, where the pricing problem is solved by a dynamic programming algorithm. Then we obtain an integer solution and the corresponding task schedule by iteratively rounding the (fractional) values of the optimal solution.

2. IP formulation

Given a list of the orders with each product including their weight, deadline and the duration of each required task (mold making, core making and assembly), along with a list of shifts categorized by their types - assembly shifts, covering all tasks other than casting and casting shifts - and their start and end times, our goal is to assign a feasible schedule to each product minimizing the total cost of all schedules. In the casting process, due to the high costs of metal melting, it is important to minimize the number of casting rounds. Additionally, to reduce operational inventory and due to space limitations, orders should be completed as close to their deadlines as possible without exceeding them, and idle time between consecutive tasks done on the same workpiece should be minimal.

Let P be the set of products and let A and C denote the sets of assembly and casting shifts. For each product $p \in P$, s_{p_i} is considered a feasible schedule if all tasks related to p are assigned to a shift with sufficient time/weight capacity, the tasks are completed in the correct order [2] - mold and core making are done before assembly and the assembly is complete before casting - and are finished before the product's deadline. The cost c_{p_i} of a schedule $s_{p_i} \in S_p$ is defined as the sum of the idle time between consecutive tasks and the time between the completion of the order and the deadline.

We can formulate it as an IP the following way.

$$\min \sum_{p \in P} \sum_{s_{p_i} \in S_p} c_{p_i} x_{s_{p_i}} + \sum_{c \in C} k_c y_c \tag{1}$$

$$x_{s_{p_i}} \in \{0,1\} \qquad \forall p \in P \quad \forall s_{p_i} \in S_p \tag{2}$$

$$y_c \in \{0,1\} \qquad \forall c \in C \tag{3}$$

$$\sum_{s_{p_i} \in S_p} x_{s_{p_i}} = 1 \qquad \forall p \in P \tag{4}$$

$$\sum_{p \in P} \sum_{\substack{s_{p_i} \in S_p \\ a \in s_{p_i}}} t_p x_{s_{p_i}} \leq T_a \qquad \forall a \in A \tag{5}$$

$$\sum_{p \in P} \sum_{\substack{s_{p_i} \in S_p \\ c \in s_{p_i}}} w_p x_{s_{p_i}} - W_c y_c \leq 0 \qquad \forall c \in C \tag{6}$$

The variables in (2) indicate whether a certain assignment is selected for a product, and the variables in (3) represent whether casting takes place during a given casting shift. (4) ensures that exactly one assignment is selected for each product. With w_p denoting the weight of product p and t_p the time required for its assigned task within the given shift, (5) enforces that the time capacities of shifts are not exceeded. (6) ensures that $y_c = 1$ exactly when there is casting during casting shift c. The objective is to minimize the total cost, which consists of the sum of idle times across all selected assignments and the cost of operating the active casting shifts.

To guarantee feasibility, we also add an extra "assignment" option for each product - one which assigns its tasks to auxiliary shifts with infinite weight/time capacity, representing the choice to not complete this order. The LP relaxation including these slack assignments is the following.

$$\min \sum_{p \in P} \sum_{s_{p_i} \in S_p} c_{p_i} x_{s_{p_i}} + \sum_{c \in C} k_c y_c \tag{7}$$

$$0 \leq x_{s_{p_i}} \leq 1 \qquad \forall p \in P, s_{p_i} \in S_p \tag{8}$$

$$0 \leq y_c \leq 1 \qquad \forall c \in C \tag{9}$$

$$\sum_{s_{p_i} \in S_p} x_{s_{p_i}} = 1 \qquad \forall p \in P \tag{10}$$

$$\sum_{p \in P} \sum_{\substack{s_{p_i} \in S_p \\ a \in s_{p_i}}} t_p x_{s_{p_i}} \leq T_a \qquad \forall a \in A \tag{11}$$

$$\sum_{p \in P} \sum_{\substack{s_{p_i} \in S_p \\ c \in s_{p_i}}} w_p x_{s_{p_i}} - W_c y_c \leq 0 \qquad \forall c \in C \tag{12}$$

3. Column generation

We solve this relaxed version of the problem using column generation [3]. Initially we select one schedule for each product (which we always have because of the extra assignments) and add the corresponding columns to the model. Alongside these we include all columns for the casting shifts (y). We obtain an optimal solution for this restricted problem and check whether the dual conditions are also satisfied. If they are, we conclude that the current solution is optimal. Otherwise we select a column corresponding to a violated dual constraint and add it to the model.
We have dual variables for each constraint.

- For (10): $\alpha_p \in \mathbb{R} \quad \forall p \in P$
- for (11) $\beta_a \geq 0 \quad \forall a \in A$
- for (12) $\gamma_c \geq 0 \quad \forall c \in C$

The dual of the LP is the following.

$$\min \sum_{p \in P} \alpha_p + \sum_{a \in A} \beta_a T_a \tag{13}$$

$$c_{p_i} + \alpha_p + \sum_{a \in s_{p_i}} \beta_a t_p + \sum_{c \in s_{p_i}} \gamma_c w_p \geq 0 \quad \forall p \in P, s_{p_i} \in S_p \tag{14}$$

$$k_c - \gamma_c W_c \geq 0 \quad \forall c \in C \tag{15}$$

Since the columns related to the casting shift variables are included in the model since the beginning, (15) is always satisfied, so we need to find feasible assignments that violate inequality (14). We do this using a dynamic programming algorithm. We search for the schedule with the most negative reduced cost for each product, which means solving for each $p \in P$ the optimization problem

$$\min_{s_{p_i} \in S_p} \left\{ c_{p_i} + \alpha_p + \sum_{a \in s_{p_i}} \beta_a t_p + \sum_{c \in s_{p_i}} \gamma_c w_p \right\}. \tag{16}$$

This is equivalent to finding a schedule with the lowest total cost, where the cost consists of shift assignment costs - calculated by multiplying the corresponding dual variables with the task duration for assembly shifts or with the product weight for casting shifts - along with the total idle time.

3.1. Dynamic programming solution for the pricing problem

We can compute the cost of such an optimal schedule by combining the results of two separate subproblems.

Algorithm 1 Minimal cost assignment DP - mold + core + assembly

for $i \leftarrow 1$ to n **do**
  $m[i] \leftarrow \beta_{a_i} \cdot t_{p_{\text{mold}}}$ if feasible, else ∞
  $c[i] \leftarrow \beta_{a_i} \cdot t_{p_{\text{core}}}$ if feasible, else ∞
  $a[i] \leftarrow \beta_{a_i} \cdot t_{p_{\text{assembly}}}$ if feasible, else ∞
end for
Initialize $M[i], C[i], A[i] \leftarrow \infty$ for all i
for $i \leftarrow 2$ to n **do**
  $M[i] \leftarrow \min(m[i-1], M[i-1] - a[i-1]) + a[i] + \text{idle}(i-1, i)$
  $C[i] \leftarrow \min(c[i-1], C[i-1] - a[i-1]) + a[i] + \text{idle}(i-1, i)$
  $A[i] \leftarrow \min\left(A[i-1],\ c[i-1] + M[i-1],\ m[i-1] + C[i-1]\right) - a[i-1] + a[i]$
  if mold and core fit together in a_{i-1} **then**
    $E \leftarrow m[i-1] + c[i-1] + a[i] + \text{idle}_{\text{mold}} + \text{idle}_{\text{core}}$
    $A[i] \leftarrow \min(N[i], E)$
  end if
  $A[i] \leftarrow A[i] + \text{idle}(i-1, i)$
end for

Algorithm 1 dynamically computes the optimal cost of assigning mold making, core making, and assembly tasks with assembly placed in assembly shift i. For each shift i, we consider three possibilities: combining the best core+assembly sequence ending in the previous shift with mold making in $i-1$; combining the best mold+assembly sequence with core making in $i-1$; or reusing the total cost from the previous step. In these three cases we don't have to worry about shift capacities, because either we checked previously in case we use the cost of the $i-1^{th}$ shift, or none of the tasks are assigned to the same shift. Additionally, if both mold and core making can fit in the same shift $i-1$, we consider placing them together and adding the assembly in shift i. The minimum of these cases gives us the optimal cost for shift i.

Algorithm 2 Minimal cost assignment DP - casting

1: **for** each cast shift c_k **do**
2: **for** each j such that a_j.end $\leq c_k$.start **do**
3: $OPT[k] \leftarrow N[j] + \text{idle}(a_j, c_k) + \text{idle}(c_k, p.\text{deadline}) + \gamma_{c_k} \cdot w_p + \alpha_p$
4: **end for**
5: **end for**

After this, Algorithm 2 computes the best full assignment by combining the results with each feasible casting shift. For each casting shift, we check all assembly shifts that finish before the casting shift starts. We compute the total cost as the precomputed mold+core+assembly cost, plus the sum of the new idle times and the casting cost. The minimum of these values is the total cost of the minimal reduced-cost assignment. If n, m denotes the number of assembly and casting shifts respectively, Algorithm 1 runs in $O(n)$ and Algorithm 2 in $O(nm)$ time.

In each iteration of column generation, we perform this computation for all orders. For those orders where the minimum reduced cost is negative, we add the corresponding minimal-cost assignments as new columns to the model.

4. Rounding

After the initial column generation, we have a (fractional) optimum for the problem. We round the solution to get integer results the following way.

In each round we select the K x variables with the largest (non-negative) value. For each one, we temporarily fix it's value to 1 and we fix all the other x variables present in the model for the same product to 0. To ensure that the casting shift used in the selected assignment is active, we also fix the corresponding y variable to 1. Then we solve the new LP obtained this way. At the end of the round, we permanently fix the x and y for which the LP had the best objective value, meaning which resulted in the least increase in objective value.

We repeat this process until every product has exactly one assignment fixed, and after every rounding step, we perform another column generation phase to update the solution, reflecting the potential changes after the newly fixed schedule and casting shift. Once a schedule is fixed for a product, it takes up a specific amount of time and weight in the corresponding assembly and casting shifts. To ensure that future schedules remain feasible — meaning every task fits within the shift to which it is assigned — we continuously track the remaining capacities of all shifts and during the column generation we only consider assignments that respect these updated limits. Furthermore, after each rounding step, we delete all columns corresponding to assignments that have become infeasible — that is, assignments for which the associated x variable can no longer be fixed to 1 in any future iteration because at least one of their tasks would exceed the capacity

of its assigned shift.

5. Implementation and results

We implemented the algorithm using C++ programming language and the LEMON optimization library [1]. The whole process including the initial column generation and the rounding procedure runs in a few minutes for smaller datasets ($\sim 50 - 100$ products and $\sim$100 shifts) and in about an hour for larger instances ($\sim$1000 products and $\sim$100 shifts - about a month's worth of orders and shifts).

	LP relaxation	**Rounded solution**
Idle time cost	119	247
Casting cost	7206300	8100000
Slack shift cost	0	0

Table 1: Example with 132 products

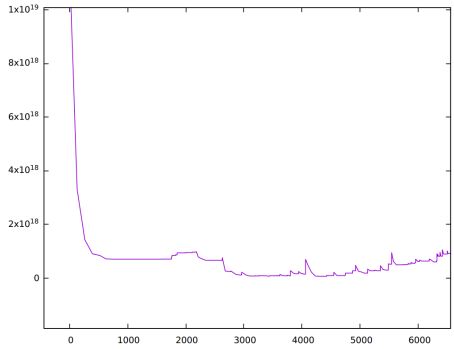

Figure 1: Change in objective value with the number of columns

Table 1 shows a breakdown of the different components of objective value for a run of the algorithm, where each casting shift had cost $9 \cdot 10^5$ and the time values were scaled by a factor of 10^{-5}. Figure 1 presents the change in the objective value with the number of columns throughout both phases of the algorithm.

The most significant difference between the optimum of the LP relaxation and the final rounded objective can be observed when the algorithm assigns slack assignments to some products, meaning these orders are not completed. This is most likely because these products could be produced if fractional time and weight allocation were allowed, but cannot be scheduled as complete assignments. This is supported by our observations that when such slack assignments are present in the final solution, the capacity of the assembly or casting shifts preceding these products deadline is nearly full - ranging from 87% to 100 % in our experiments.

References

[1] LEMON - Library for Efficient Modeling and Optimization in Networks. https://lemon.elte.hu/.

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

[3] Marco Lübbecke. *Column Generation.* 01 2011.