Preliminary Results on Robust Non-Clairvoyant Scheduling with Classification Models

Anthony Dugois
(on-going work with Vincent Fagnon and Giorgio Lucarelli)

Université Marie et Louis Pasteur, institut FEMTO-ST, Besançon

SCALE
april 15, 2025

Outline

Model

Some Results

On-Going Work

Outline

Model

Some Results

On-Going Work

Dealing with Uncertainty

- Stochastic models
- Min-max (regret) models
- Learning-augmented models

Classification Model

- Jobs are classified according to K classes
- Jobs of a given class k have similar characteristics (e.g., same processing times)
- The class of a job is unknown
- Access to an oracle based on a classification model, which **can make mistakes**

Confusion Matrix

The oracle is represented as a $K \times K$ *confusion matrix*:

$$E = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

- Row i: jobs that the oracle *believes* to be in class i
- Column j: jobs *actually* in class j
- Entry e_{ij}: number of jobs believed to be in i but actually in j
- We suppose that E is known

A First Scheduling Problem

- $1||\sum C_j$ with K processing times $p(1) \leq \cdots \leq p(K)$
- SPT is optimal when all processing times are known
- In a non-clairvoyant setting, the only solution is to schedule jobs randomly

Question

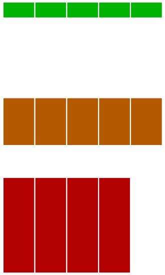
What can we do when we have access to an oracle (and its confusion matrix)?

A First Scheduling Problem

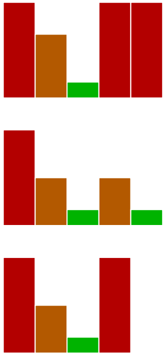
$$E = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

$p(1) = 1, p(2) = 4, p(3) = 6$

Seen by oracle



Reality



A First Scheduling Problem

- The oracle gives us rows $B(1), \ldots, B(K)$, where $B(k)$ contains the jobs that the oracle believes to be in class k
- We know the distribution in each $B(k)$ according to the matrix E, but we cannot actually distinguish jobs
- The feasible strategies all consist in choosing a sequence of rows from which randomly picking a job

Question

How to choose an optimal sequence of rows?

Refined Problem

- Picking a random job in a set can be seen as considering a fixed ordering of the jobs and pulling the job located at the head
- Consider an (unknown) scenario σ, which fixes the ordering of the jobs in each row (let σ_k denote the considered permutation of $B(k)$)
- The goal is to choose a sequence $r = (r(1), r(2), \dots, r(n))$, where $r(t) \in \{1, \dots, K\}$ denotes the row from which we pull a job at step t, that minimizes $\sum C_j$

Outline

Adaptive vs. Non-Adaptive Strategies

- Adaptive: the strategy adapts the sequence r as it learns about the already-executed jobs (i.e., it learns about σ)
- **Non-adaptive**: the strategy must decide the sequence r before executing any job, without knowing σ

Objectives

Let R be the set of feasible sequences, S the set of scenarios, and $\mathcal{C}(r, \sigma)$ the objective of the schedule generated from sequence r when applied on scenario σ.

- Min-Max: $\min_{r \in R} \max_{\sigma \in S} \mathcal{C}(r, \sigma)$
- Min-Average: $\min_{r \in R} \sum_{\sigma \in S} \mathcal{C}(r, \sigma)$
- Min-Max Regret: $\min_{r \in R} \max_{\sigma \in S} (\mathcal{C}(r, \sigma) - \mathcal{C}^*(\sigma))$, where $\mathcal{C}^*(\sigma)$ is the optimal solution for σ

Optimal Sequence for a Fixed σ

- For a fixed scenario σ, the problem is equivalent to $1|\text{chains}|\sum C_j$
- The following algorithm is optimal:
  1. Compute $x(k,j) = \frac{1}{j}\sum_{i=1}^{j} p_{\sigma_k(i)}$ for all $1 \leq k \leq K$ and all $1 \leq j \leq |B(k)|$, where $p_{\sigma_k(i)}$ is the processing time of the i-th job in $B(k)$
  2. Compute $y(k) = \min_j x(k,j)$, $h(k) = \arg\min_j x(k,j)$ for all k
  3. Let $k^* = \arg\min_k y(k)$ and schedule the first $h(k^*)$ jobs of $B(k^*)$
  4. Repeat from step 1

Min-Max: $\min_{r\in R} \max_{\sigma\in S} \mathcal{C}(r,\sigma)$

Lemma

For any fixed sequence r, the worst scenario consists in jobs of each row $B(k)$ being arranged in decreasing order of processing times.

Theorem

Let $\bar{p}(k) = \sum_{j=1}^{K} \frac{e_{kj}}{|B(k)|} p(j)$ be the average processing time of jobs in $B(k)$. Scheduling sets $B(k)$ in increasing order of $\bar{p}(k)$ solves Min-Max.

Min-Average: $\min_{r \in R} \sum_{\sigma \in S} \mathcal{C}(r, \sigma)$

Theorem

Scheduling sets $B(k)$ in increasing order of $\bar{p}(k)$ solves Min-Average.

Open questions

- Generalized problem $\min_{r \in R} \left(\sum_{\sigma \in S} \mathcal{C}(r, \sigma)^p \right)^{1/p}$
- Generalized problem $\min_{r \in R} \sum_{\sigma \in S} w_\sigma \mathcal{C}(r, \sigma)$

Min-Max Regret: $\min_{r \in R} \max_{\sigma \in S} (\mathcal{C}(r, \sigma) - \mathcal{C}^*(\sigma))$

Open questions

- Is the problem NP-hard?
- For a fixed sequence r, can we find the scenario σ maximizing the regret $\mathcal{C}(r, \sigma) - \mathcal{C}^*(\sigma)$?

Outline

Model

Some Results

On-Going Work

On-Going (and Future) Work

- Min-Max Regret
- Adaptive strategies
- More complex scheduling problems, e.g., $P||\sum C_j$, $1|r_j|\sum C_j$, $1||\sum w_j C_j$
- Are there easier oracles than others?