femto-st SCIENCES & TECHNOLOGIES

Robust Non-Clairvoyant Scheduling with Classification Models

A. Dugois[1], *V. Fagnon*[1], *and G. Lucarelli*[2]

[1]Université Marie et Louis Pasteur, institut FEMTO-ST

[2]Université de Lorraine, LCOMS

March 18, 2026 @ Fréjus

19th Scheduling for large-scale systems workshop

cnrs UNIVERSITÉ MARIE & LOUIS PASTEUR SUP MICRO TECH ENSMM utbm université de technologie Belfort-Montbéliard

Table of Contents

Table of Contents

Motivation

Scheduling problems often suffer from **uncertainty**.

- Typical example: exact processing times are unknown before completion

SPT easily solves $1||\sum C_j$

Motivation

Scheduling problems often suffer from **uncertainty**.

- Typical example: exact processing times are unknown before completion

Non-clairvoyant setting

Our Approach: Classification Models

- Classification models assign *inputs* to **classes**
  - Example: *emails* categorized as **spam** or **not spam**
- Subject to errors (may return the wrong class)
- Classification models can be obtained from machine learning (SVM, random forest, k-NN, etc.), probabilistic data structures (e.g., Bloom filters), or can represent an expert
- Often evaluated through **confusion matrices**

Pred \ True	c_1	c_2	c_3	c_4	c_5
c_1	2	1	3	0	1
c_2	4	5	8	5	3
c_3	3	6	2	2	0
c_4	1	9	5	1	4
c_5	3	3	2	0	4

8 items categorized in class 2 but actually in class 3

Table of Contents

Scheduling Problem

Augment non-clairvoyant $1||\sum C_j$ with a classifier:

- jobs are categorized into K classes
- jobs in class k take time p_k
- the classifier predicts the class of a job
- no additional cost to query the classifier
- represented by a (known) confusion matrix E

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{cccc} \multicolumn{4}{c}{\text{True}} \\ p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 1 & 2 \end{bmatrix} \end{array} \begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}$$

Model Behavior

Pred \ True	p_1	p_2	p_3	p_4	p_5
p_1	12	0	0	0	0
p_2	0	8	0	0	0
p_3	0	0	9	0	0
p_4	0	0	0	5	0
p_5	0	0	0	0	7

Clairvoyant classifier

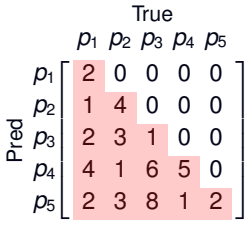
Pred \ True	p_1	p_2	p_3	p_4	p_5
p_1	2	1	1	2	2
p_2	4	4	3	0	3
p_3	1	1	0	1	0
p_4	2	0	1	0	1
p_5	3	2	4	2	1

Intermediary classifier

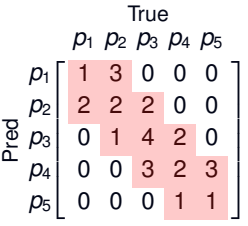
Pred \ True	p_1	p_2	p_3	p_4	p_5
p_1	12	8	9	5	7
p_2	0	0	0	0	0
p_3	0	0	0	0	0
p_4	0	0	0	0	0
p_5	0	0	0	0	0

Non-clairvoyant classifier

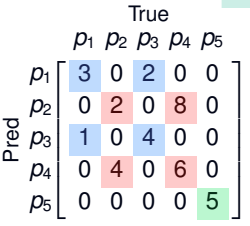
Model Behavior



No underestimates

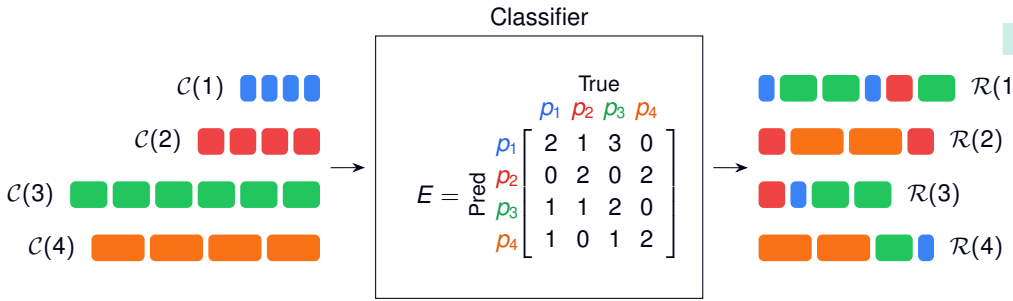


Bounded errors



Pairwise similarities

Classification of Jobs



$\mathcal{C}(k)$: set of jobs actually in class k (= column k in E)

$\mathcal{R}(k)$: set of jobs predicted to be in class k (= row k in E)

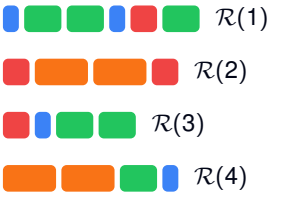
Uncertainty on Permutations

Scheduler Decisions

The only decision we can take at each step is from which set $\mathcal{R}(k)$ we pull the next job to execute.

Example solution

$\mathcal{R}(k)$	1	1	3	2	2	...
job type	●	●	●	●	●	...
p_j	1	3	2	2	4	...
p_j	1	4	6	8	12	...



Scenario σ (adversary)

Table of Contents

Robust Optimization

$\textsc{Obj}(r, \sigma)$ = sum of completion times of the schedule induced by solution r and scenario σ.

Robustness criteria

- Min-Average $\min_r \sum_\sigma \textsc{Obj}(r, \sigma)$
- Min-Max $\min_r \max_\sigma \{\textsc{Obj}(r, \sigma)\}$
- Min-Max Regret $\min_r \max_\sigma \{\textsc{Obj}(r, \sigma) - \textsc{Opt}(\sigma)\}$

Optimal Solution

To what solution do we compare?

- Seems **unfair** to compare with SPT (adversary is too powerful)

Optimal Solution

To what solution do we compare?

- Seems **unfair** to compare with SPT (adversary is too powerful)
- Idea: compare with an optimal that knows the scenario, but **is forced** to follow the imposed ordering

Optimal Solution

To what solution do we compare?

- Seems **unfair** to compare with SPT (adversary is too powerful)
- Idea: compare with an optimal that knows the scenario, but **is forced** to follow the imposed ordering

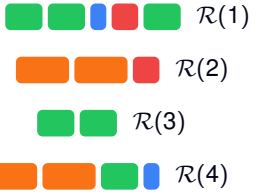
$\rightarrow$ Equivalent to $1|\text{chains}|\sum C_j$

Optimal Algorithm for $1|\textbf{chains}|\sum C_j$

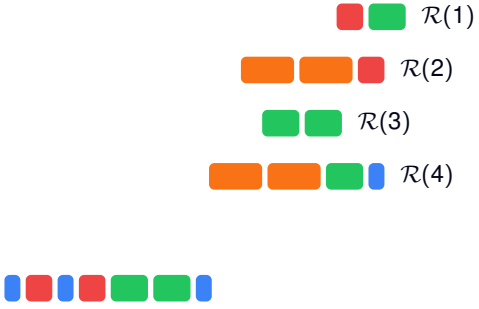
Optimal Algorithm for $1|\textbf{chains}|\sum C_j$

Optimal Algorithm for $1|\textbf{chains}|\sum C_j$

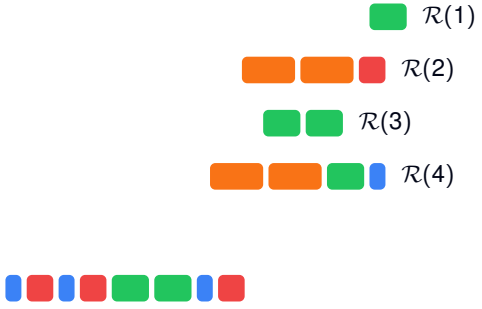
Optimal Algorithm for $1|\text{chains}|\sum C_j$



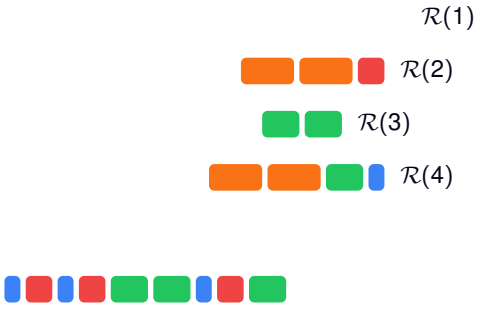
Optimal Algorithm for $1|\textbf{chains}|\sum C_j$



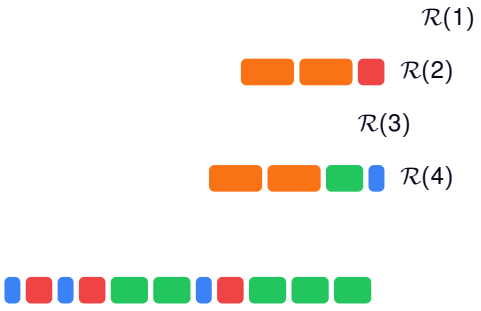
Optimal Algorithm for $1|\textbf{chains}|\sum C_j$



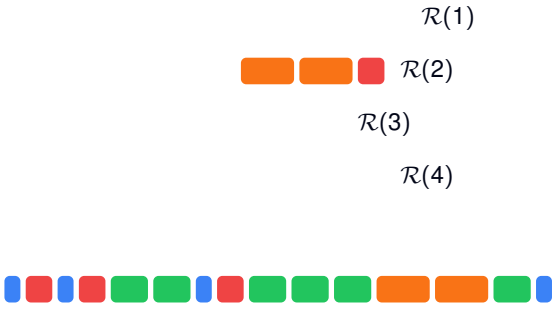
Optimal Algorithm for $1|\textbf{chains}|\sum C_j$



Optimal Algorithm for $1|\textbf{chains}|\sum C_j$



Optimal Algorithm for $1|\textbf{chains}|\sum C_j$



Optimal Algorithm for $1|\textbf{chains}|\sum C_j$

$\mathcal{R}(1)$

$\mathcal{R}(2)$

$\mathcal{R}(3)$

$\mathcal{R}(4)$

A Simple Non-Adaptive Algorithm

MinWeight

- Compute the mean processing time $\bar{p}(k)$ of each $\mathcal{R}(k)$
- Schedule the complete $\mathcal{R}(k)$ in increasing order of $\bar{p}(k)$

A Simple Non-Adaptive Algorithm

MinWeight

- Compute the mean processing time $\bar{p}(k)$ of each $\mathcal{R}(k)$
- Schedule the complete $\mathcal{R}(k)$ in increasing order of $\bar{p}(k)$

Theorem

MinWeight *is optimal for:*

1. *Min-Average* $\min_r \sum_\sigma \textsc{Obj}(r, \sigma)$
2. *Min-Max* $\min_r \max_\sigma \{\textsc{Obj}(r, \sigma)\}$
3. *Min-Max Regret (with 2 classes)* $\min_r \max_\sigma \{\textsc{Obj}(r, \sigma) - \textsc{Opt}(\sigma)\}$

Table of Contents

Adaptive Min-Average

Now the scheduler is allowed to take decisions in reaction to the past.

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{cc} \text{True} & \\ p_1 & p_2 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \end{array} \begin{bmatrix} 1 & 3 \\ 2 & 8 \end{bmatrix} \end{array}$$

$\mathcal{R}(1) \rightarrow$  $\rightarrow \mathcal{R}(2) \ldots$

Adaptive Min-Average

Now the scheduler is allowed to take decisions in reaction to the past.

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{cc} \text{True} & \\ p_1 & p_2 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \end{array} \begin{bmatrix} 1 & 3 \\ 2 & 8 \end{bmatrix} \end{array}$$

$\mathcal{R}(1) \rightarrow$  $\rightarrow \mathcal{R}(2) \ldots$

$\mathcal{R}(1) \rightarrow$  $\rightarrow \mathcal{R}(1) \ldots$

MinWeight for Adaptive Min-Average

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{ccc} \text{True} & & \\ p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} p_1 \\ p_2 \\ p_3 \end{array}$$

2 scenarios: either the small task appears first in $\mathcal{R}(2)$, or it is the big one.

Suppose $p_2 < \frac{1}{2}(p_1 + p_3)$, i.e., MinWeight schedules $\mathcal{R}(1)$ then $\mathcal{R}(2)$.

$$\begin{aligned} \mathbb{E}[\text{Obj}(\text{MinWeight})] &= \frac{1}{2}(3p_2 + 2p_1 + p_3) + \frac{1}{2}(3p_2 + 2p_3 + p_1) \\ &= \frac{3}{2}p_1 + 3p_2 + \frac{3}{2}p_3 \end{aligned}$$

MINWEIGHT for Adaptive Min-Average

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{ccc} \text{True} & & \\ p_1 & p_2 & p_3 \end{array} \\ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Optimal solution is to begin with $\mathcal{R}(2)$:

- Scenario 1 (small task first): $\mathcal{R}(2) \to \mathcal{R}(1) \to \mathcal{R}(2)$
- Scenario 2 (big task first): $\mathcal{R}(2) \to \mathcal{R}(2) \to \mathcal{R}(1)$

$$\begin{aligned} \mathbb{E}^* &= \frac{1}{2}(3p_1 + 2p_2 + p_3) + \frac{1}{2}(3p_3 + 2p_1 + p_2) \\ &= \frac{5}{2}p_1 + \frac{3}{2}p_2 + 2p_3 \end{aligned}$$

MinWeight for Adaptive Min-Average

If we take $p_1 = 1, p_2 = 3, p_3 = 6$:

- $\mathbb{E}[\text{Obj}(\text{MinWeight})] = \frac{3}{2}p_1 + 3p_2 + \frac{3}{2}p_3 = 19.5$
- $\mathbb{E}^* = \frac{5}{2}p_1 + \frac{3}{2}p_2 + 2p_3 = 19$

Optimal DP

$$\textsc{Solve}(E) = \min_k \left\{ \sum_{i=1}^{K} \mathbb{P}[\text{col } i \mid \text{row } k] \left(s(E) \cdot p_i + \textsc{Solve}\left(E^-_{(ki)} \right) \right) \right\}$$

- $\mathbb{P}[\text{col } i \mid \text{row } k] = E_{ki} / \sum_{j=1}^{K} E_{kj}$ is the proba to select a task from column i knowing we select from row k
- $s(E)$ is the sum of all elements of E
- $E^-_{(ki)}$ is the matrix E where we subtract 1 to element E_{ki}

Optimal DP

$$\textsc{Solve}(E) = \min_k \left\{ \sum_{i=1}^{K} \mathbb{P}[\text{col } i \mid \text{row } k] \left(s(E) \cdot p_i + \textsc{Solve}\left(E^-_{(ki)} \right) \right) \right\}$$

Complexity

DP table has $O\left(n^{K^2}\right)$ entries in the worst case.

Optimal DP

$$\textsc{Solve}(E) = \min_k \left\{ \sum_{i=1}^{K} \mathbb{P}[\text{col } i \mid \text{row } k] \left(s(E) \cdot p_i + \textsc{Solve}\left(E^-_{(ki)} \right) \right) \right\}$$

Complexity

DP table has $O\left(n^{K^2}\right)$ entries in the worst case.

$\rightarrow$ Complex solution to a simple problem :)

Refined Algorithm

Lower band matrices of order b

For each row k, the first $\max(k-b,0)$ and the last $K-k$ elements are 0.

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{ccccccc} \multicolumn{7}{c}{\text{True}} \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 & 1 & 5 \end{bmatrix} \end{array} \begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array}$$

Refined Algorithm

Lower band matrices of order b

For each row k, the first $\max(k-b,0)$ and the last $K-k$ elements are 0.

$$E = \begin{array}{c} \\ \text{Pred} \end{array} \begin{array}{c} \begin{array}{ccccccc} \multicolumn{7}{c}{\text{True}} \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 & 1 & 5 \end{bmatrix} \end{array} \begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array}$$

Complexity

At most K DP with $O\left(n^{b^2}\right)$ entries each.

Table of Contents

Conclusion

Summary

- Classification models to handle uncertainty
- A very simple algorithm solves non-adaptive robust problems
- Adaptive problem not so easy: optimal DP with exponential complexity

Work in progress

- Min-Max Regret for K classes
- Adaptive versions of Min-Max and Min-Max Regret
- Heuristics/approximations for adaptive Min-Average
- Generalize the model to parallel machines