# Learning to Schedule

Jeroen van Riel

November 2023

# Learning problem

- set of problem instances $\mathcal{I}$
- distribution $P$ over instances
- set of algorithms $\mathcal{A}$
- measure of optimality $m : \mathcal{I} \times \mathcal{A} \to \mathbb{R}$

# Learning problem

- general learning objective

$$\min_{a \in \mathcal{A}} \mathbb{E}_{i \sim P}\, m(i, a) \tag{1}$$

- no access to $\mathcal{I}$ or $P$, so use samples

$$\min_{a \in \mathcal{A}} \sum_{i \in D_{train}} \frac{1}{|D_{train}|} m(i, a) \tag{2}$$

# Learning problem

- examples of algorithm spaces $\mathcal{A}$
  - all possible C++ programs
  - finite set of knapsack heuristics
  - algorithm parameterized by neural network with weights $\theta \in \mathbb{R}^p$

$$\min_{\theta \in R^p} \mathbb{E}_{i \sim P} m(i, a(\theta)) \tag{3}$$

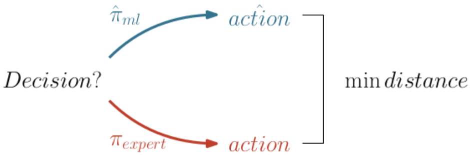
# Motivations for ML in CO [Bengio et al., 2020]

- fast approximations
  - derived in generic way
- better algorithms
  - by systematically exploring $\mathcal{A}$
  - by exploiting instance distribution

- both problems stated in MDP framework
  - environment is **internal state of algorithm**

# Main learning settings

- learning from demonstration $\rightarrow$ fast approximations
  - imitation learning
  - expert such as MILP solver



# Main learning settings

- learning from experience $\rightarrow$ better algorithms
  - objective encoded in rewards
  - algorithms (examples)
    - search-based and branch & bound
    - genetic algorithms
    - reinforcement learning

$$Decision? \xrightarrow{\hat{\pi}_{ml}} \hat{action} \xrightarrow{score} reward \quad \Big\uparrow \quad \max return$$

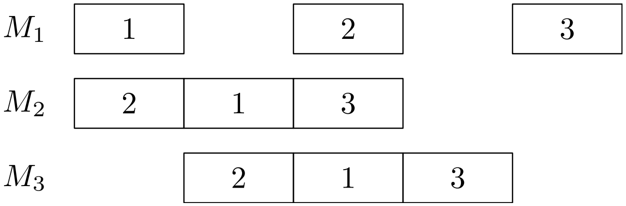
# Job shop $(= \mathcal{I})$

- $m$ machines
- $n$ jobs
- fixed machine order for each job

(closely related to the traffic scheduling variant)

# Job shop $(= \mathcal{I})$

- example schedule



# Job shop ($= \mathcal{I}$)

- job $j$ on machine $i$ is operation $(i, j)$
- operations $N$
- order of operations for particular job $j$ is fixed

$$(i, j) \to (k, j) \in \mathcal{C}$$

- order among jobs $j$ and $l$ is optimization decision

$$(i, j) \to (k, l) \quad \text{or} \quad (i, l) \to (k, j)$$

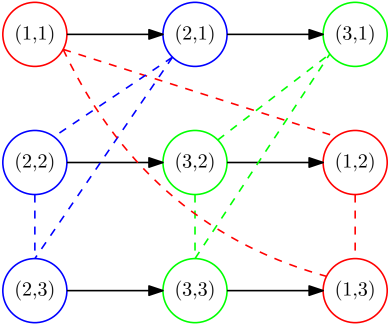
# Job shop MILP

- makespan objective
- mixed-integer linear program

$$
\begin{aligned}
&\text{minimize } C_{\max} \\
&y_{ij} + p_{ij} \le y_{kj} && \text{for all } (i,j) \to (k,j) \in \mathcal{C} \\
&y_{il} + p_{il} \le y_{ij} \text{ or } y_{ij} + p_{ij} \le y_{il} && \text{for all } (i,l) \text{ and } (i,j), i = 1, \ldots, m \\
&y_{ij} + p_{ij} \le C_{\max} && \text{for all } (i,j) \in N \\
&y_{ij} \ge 0 && \text{for all } (i,j) \in N
\end{aligned}
$$

# Disjunctive graph

- directed graph $G = (N, \mathcal{C}, \mathcal{D})$
- conjunctive arcs
- disjunctive arcs



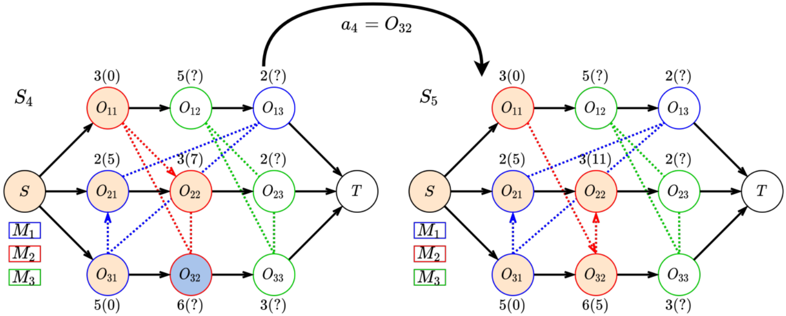
# Dispatching rule

- widely used in practice
- examples
  - SPT/LPT
  - MWR/LWR

# Zhang et al.

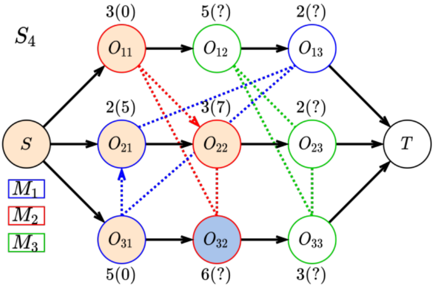
- job shop
- learn dispatching rule
- GNN on disjunctive graph
    - removing-arc or adding-arc strategy

# Zhang et al.

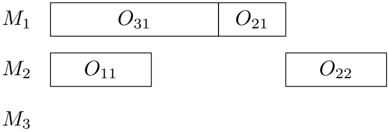


$a_4 = O_{32}$
$S_4$
$S_5$
3(0)
5(?)
2(?)
2(5)
3(7)
3(11)
5(0)
6(?)
6(5)
3(?)
$O_{11}$ $O_{12}$ $O_{13}$ $O_{21}$ $O_{22}$ $O_{23}$ $O_{31}$ $O_{32}$ $O_{33}$
$S$ $T$
$M_1$ $M_2$ $M_3$

# Zhang et al.

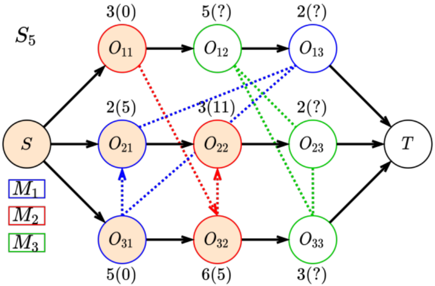


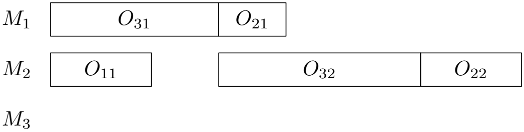
$S_4$
3(0)
5(?)
2(?)
$O_{11}$
$O_{12}$
$O_{13}$
2(5)
3(7)
2(?)
$S$
$O_{21}$
$O_{22}$
$O_{23}$
$T$
$M_1$
$M_2$
$M_3$
$O_{31}$
$O_{32}$
$O_{33}$
5(0)
6(?)
3(?)

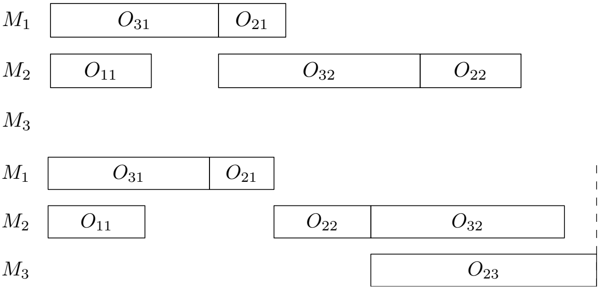
$M_1$
$O_{31}$
$O_{21}$
$M_2$
$O_{11}$
$O_{22}$
$M_3$

# Zhang et al.


$S_5$
3(0)
5(?)
2(?)
$O_{11}$
$O_{12}$
$O_{13}$
2(5)
3(11)
2(?)
$S$
$O_{21}$
$O_{22}$
$O_{23}$
$T$
$M_1$
$M_2$
$M_3$
$O_{31}$
$O_{32}$
$O_{33}$
5(0)
6(5)
3(?)


$M_1$
$O_{31}$
$O_{21}$
$M_2$
$O_{11}$
$O_{32}$
$O_{22}$
$M_3$

# Zhang et al.



# Schdule classes

**Active Schedule**. A feasible non-preemptive schedule is called active if it is not possible to construct another schedule, through changes in the order of processing on the machines, with at least one operation finishing earlier and no operation finishing later.

**Semi-Active Schedule**. A feasible non-preemptive schedule is called semi-active if no operation can be completed earlier without changing the order of processing on any one of the machines.

taken from [Pinedo, 2016]

# Tassel et al.

# Traffic scheduling problem

- total completion time $\sum C_j$
- release dates $r_j$
- chains $j_1 \rightarrow j_2 \rightarrow \cdots \rightarrow j_k$
- setup times (switch-over) $s_{ij}$

# References

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon, March 2020.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26578-0 978-3-319-26580-3. doi: 10.1007/978-3-319-26580-3.