Learning to Schedule

Jeroen van Riel

November 2023

Combinatorial Optimization

knapsack (example)

- finite set of items
- item i has non-negative weight w_i and value v_i
- choose items to maximize total value while keeping total weight at most c
- as a MILP

$$\begin{aligned} \max\ & vx \\ \text{s.t. } & wx \leq c \\ & x \geq 0 \\ & x \in \mathbb{Z} \end{aligned} \tag{1}$$

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

$$\max_{a \in \mathcal{A}} \mathbb{E}_{i \sim P}\, m(i, a) \tag{2}$$

- no access to P, so use samples

$$\max_{a \in \mathcal{A}} \sum_{i \in D_{train}} \frac{1}{|D_{train}|} m(i, a) \tag{3}$$

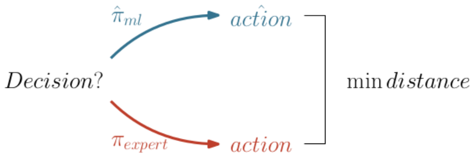
Learning problem

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

$$\max_{\theta \in R^p} \mathbb{E}_{i \sim P} m(i, a(\theta)) \tag{4}$$

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

- fast approximations
  - derived in generic way
- learning from demonstration
  - imitation learning
  - expert such as MILP solver



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

- better algorithms
  - by systematically exploring $\mathcal{A}$
  - by exploiting instance distribution
- learning from experience
  - objective encoded in rewards
  - algorithms (examples)
    - search-based and branch & bound
    - genetic algorithms
    - reinforcement learning

$$\begin{tikzcd} Decision? \arrow[r, "\hat{\pi}_{ml}"] & action \arrow[r, "score"] & reward & \uparrow & \max return \end{tikzcd}$$

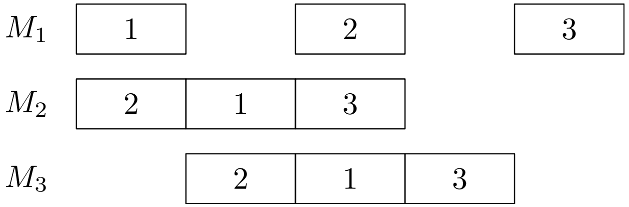
Job shop ($= \mathcal{I}$)

- 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

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

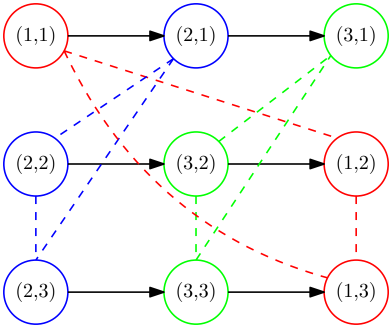
Job shop MILP

- 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

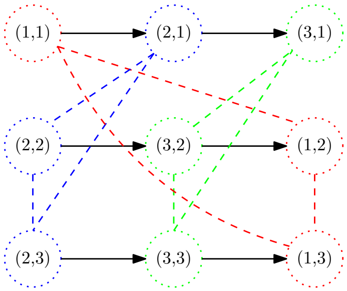
- directed graph $G = (N, \mathcal{C}, \mathcal{D})$
- conjunctive arcs
- disjunctive arcs



Dispatching rule

- widely used in practice
- order jobs and put in schedule one by one
- examples
  - SPT/LPT
  - MWR/LWR

Dispatching rule (example)

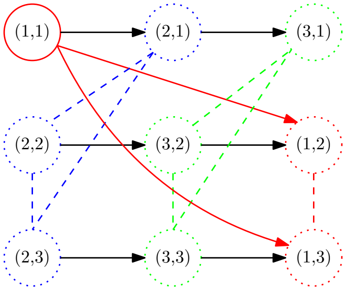


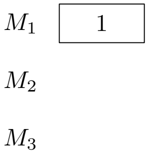
M_1

M_2

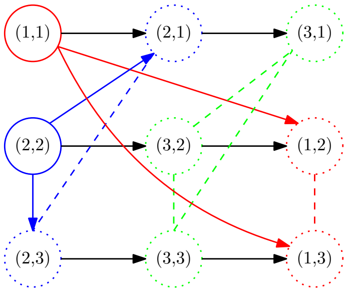
M_3

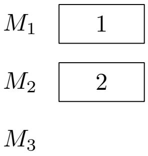
Dispatching rule (example)



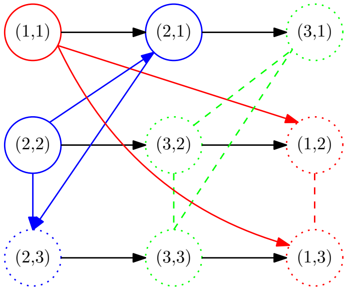


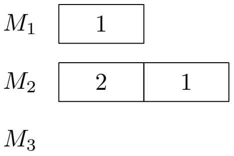
Dispatching rule (example)



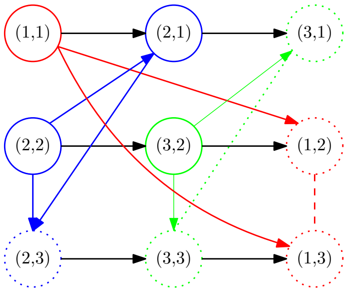


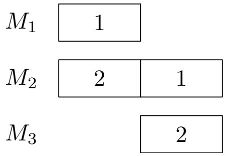
Dispatching rule (example)



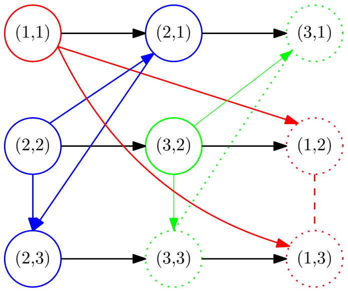


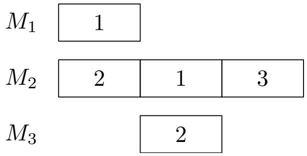
Dispatching rule (example)



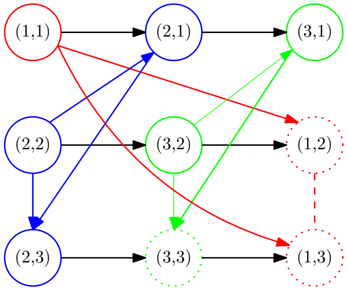


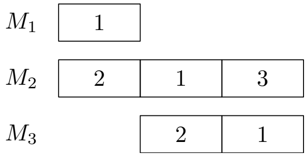
Dispatching rule (example)



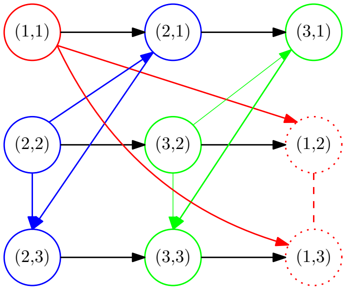


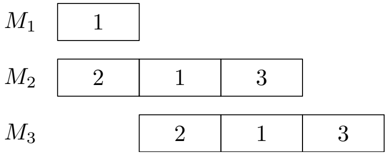
Dispatching rule (example)



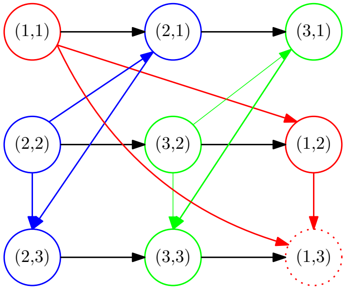


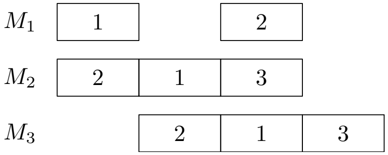
Dispatching rule (example)



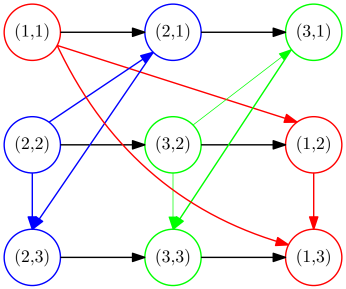


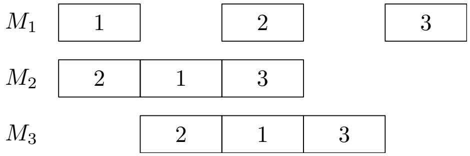
Dispatching rule (example)



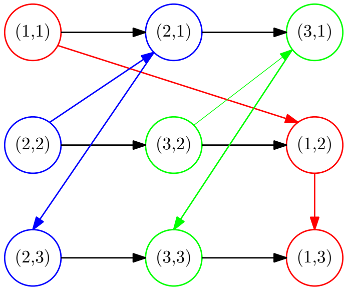


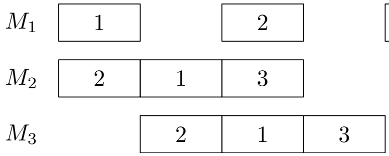
Dispatching rule (example)





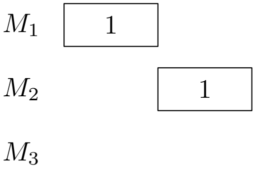
Dispatching rule (example)

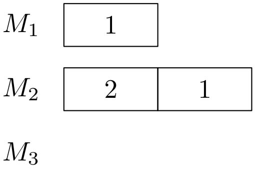




Placement rule

- dispatching rule is not sufficient
- the example applied *last position rule*
- alternatively use *earliest gap rule*





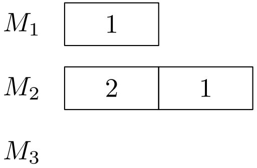
Zhang et al.

"Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning"

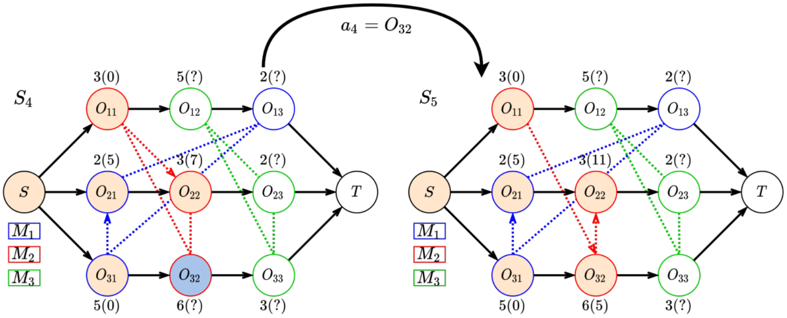
- job shop
- learn dispatching rule
- GNN on disjunctive graph
- remarkable placement rule

Notation

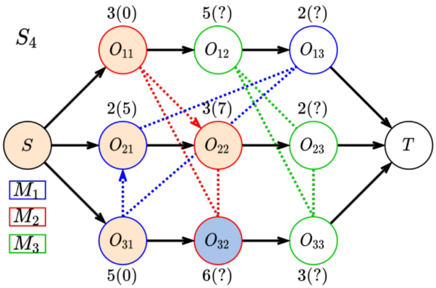
- operation O_{jk} for k'th operation of job j
- example
  - $O_{11} = (1, 1)$
  - $O_{12} = (2, 1)$
  - $O_{21} = (2, 2)$

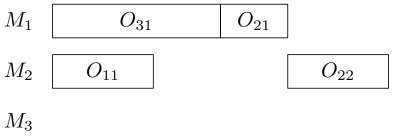


State transition

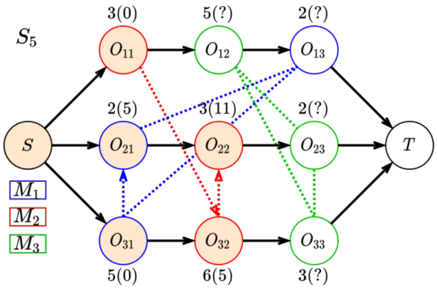


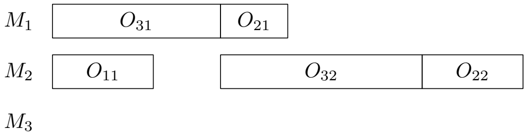
State transition



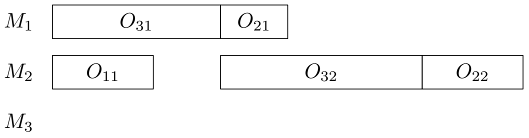


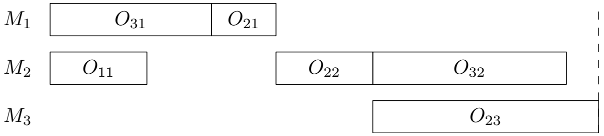
State transition





State transition





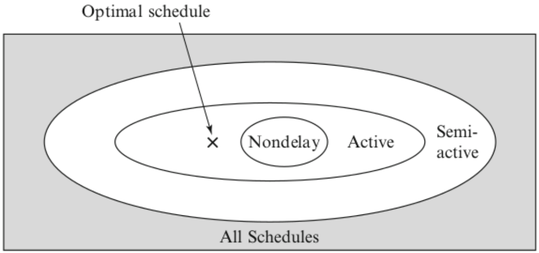
Schedule classes [Pinedo, 2016]

Non-delay Schedule A feasible schedule is called non-delay if no machine is kept idle while an operation is waiting for processing.
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.

Schedule classes [Pinedo, 2016]

Proposition. Scheduling problem $Jm||\gamma$ has optimal active schedule if γ is regular, i.e., non-decreasing function of completion times C_i.



Relation to disjunctive graph

Proposition
Every complete disjunctive graph corresponds to a unique semi-active schedule.

Proposition
For every feasible semi-active schedule, there exists a sequence χ that generates it using the last position rule.

Tassel et al.

"A Reinforcement Learning Environment For Job-Shop Scheduling"

- job shop
- learn something similar to dispatching rule
- actions space subset of $\{J_0, \ldots, J_n, \text{No-Op}\}$

Tassel et al.

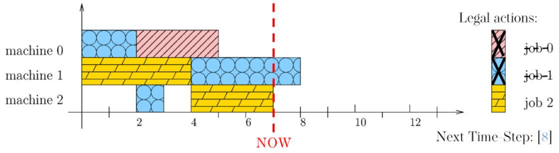


Figure 2: Representation of the environment's internal state comprising the information about the current time-step, the allocation of jobs to machines, the jobs that can be allocated now and the future time-steps.

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}

Next steps

- relate actions in Tassel et al. to disjunctive graph
- argue that traffic scheduling problem has semi-active optimal schedule
- implement *gym* environment for traffic scheduling problem

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.