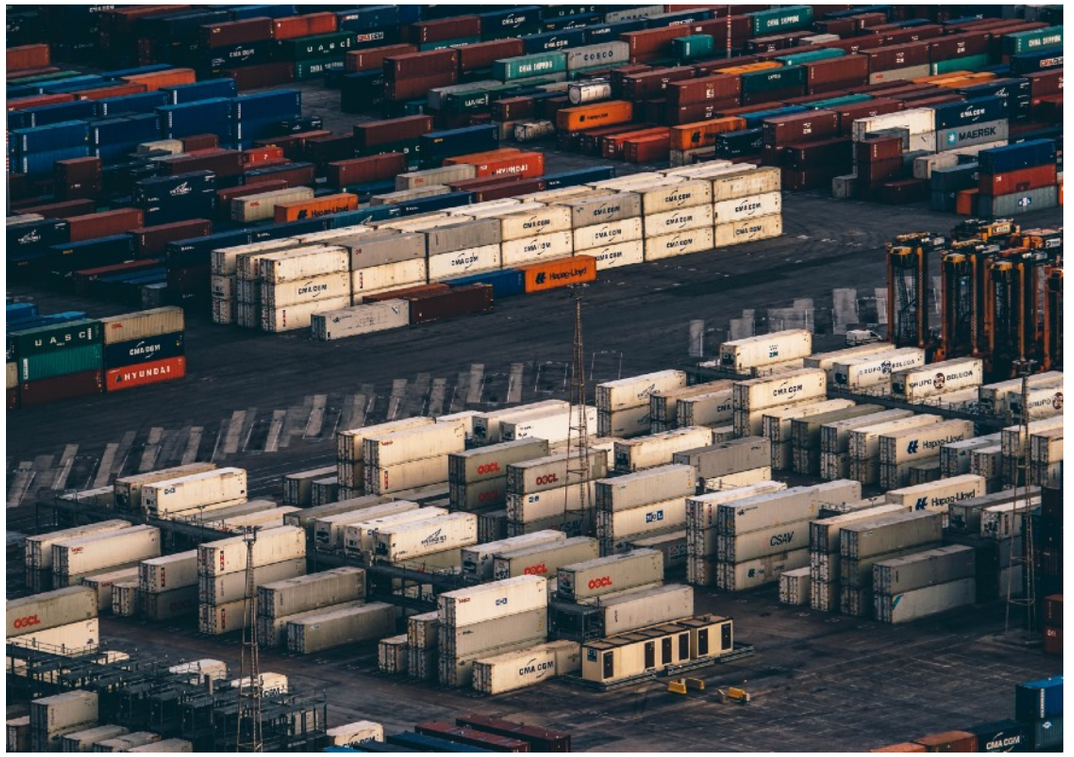
Machine Learning for Combinatorial Optimization

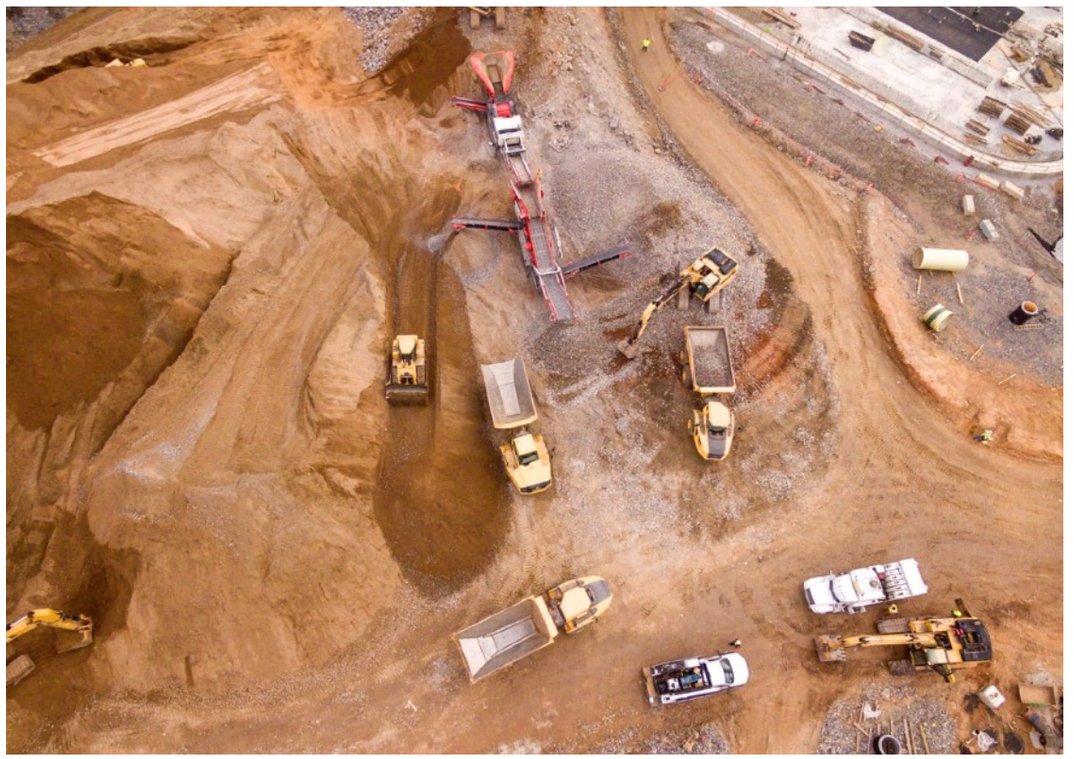
Bengio, Lodi, Prouvost
ElementAi
21st of February 2019

CANADA EXCELLENCE RESEARCH CHAIR
DATA SCIENCE FOR REAL-TIME DECISION-MAKING

Mila

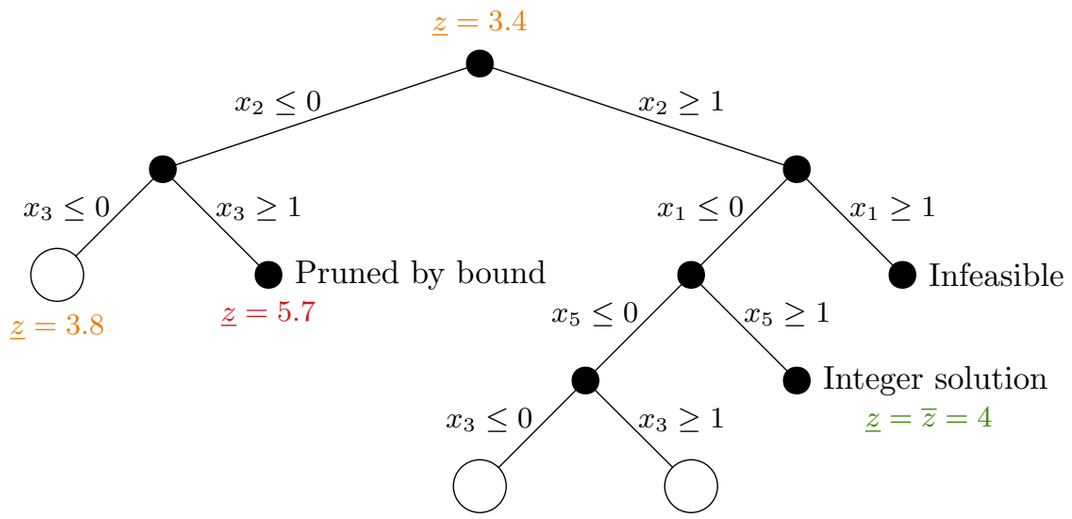
POLYTECHNIQUE MONTRÉAL
WORLD-CLASS ENGINEERING


MAERSK
HYUNDAI
OOCL
CMA CGM
Hapag-Lloyd



Discrete decisions

- MILP is NP-complete
- Many different techniques
  - Branch and bound
  - Cutting planes
  - Decomposition
  - Heuristics
  - Presolving



A Branch and Bound Tree

Too long

- Expert knowledge of how to make decisions
- Too expensive to compute
- Need for fast approximation

Too heuristic

- No idea which strategy will perform better
- Need a well performing policy
- Need to discover policies

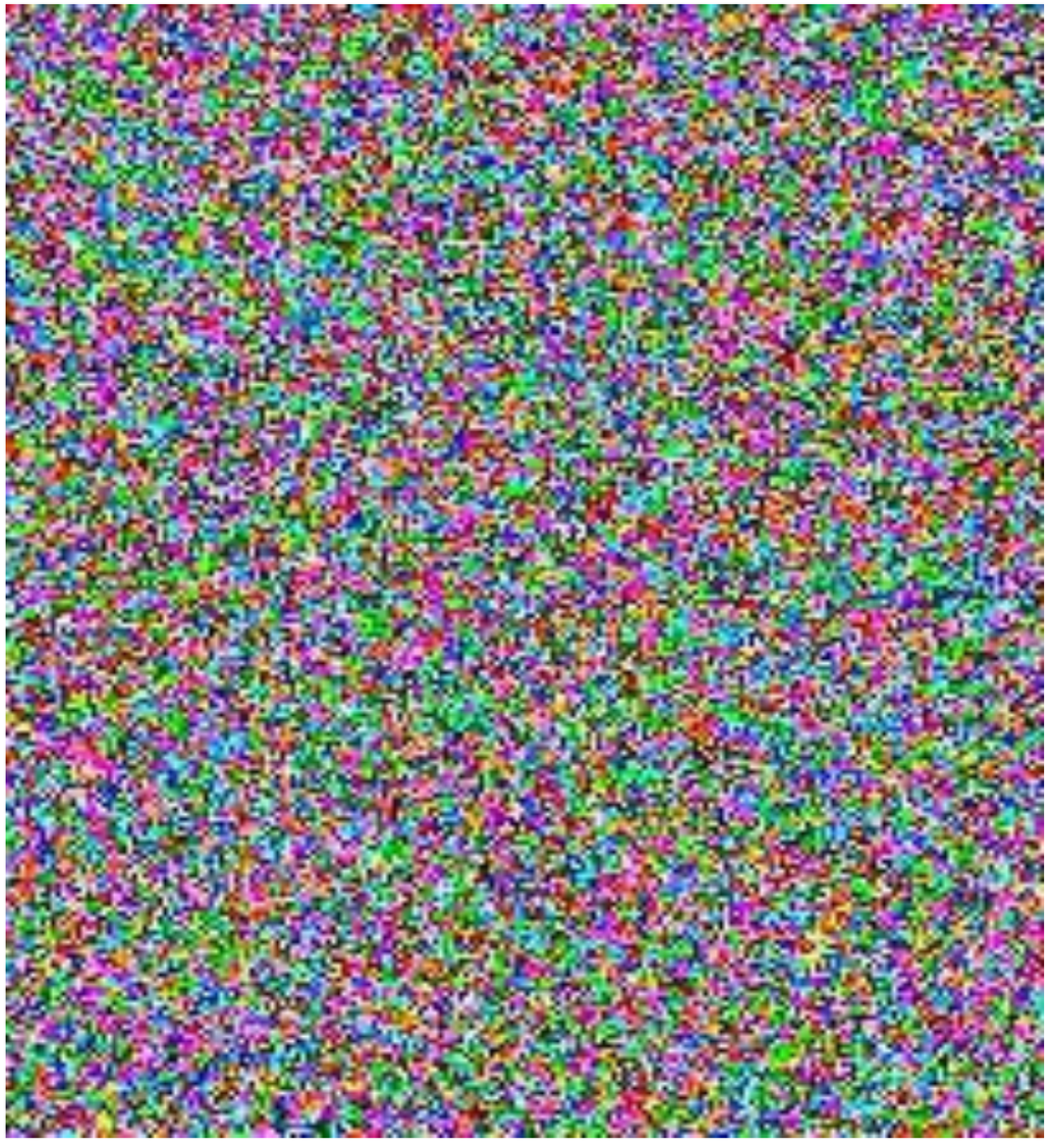
Requirement

- We want to keep the **guarantees** provided by exact OR algorithms (feasibility, optimality)

The structure hypothesis

- We do not care about most instances that coud exist;
- Instead, we look at problem instances as data points from a specific, untractable, probability distribution;
- “Similar” instances show “similar” solving procedures.

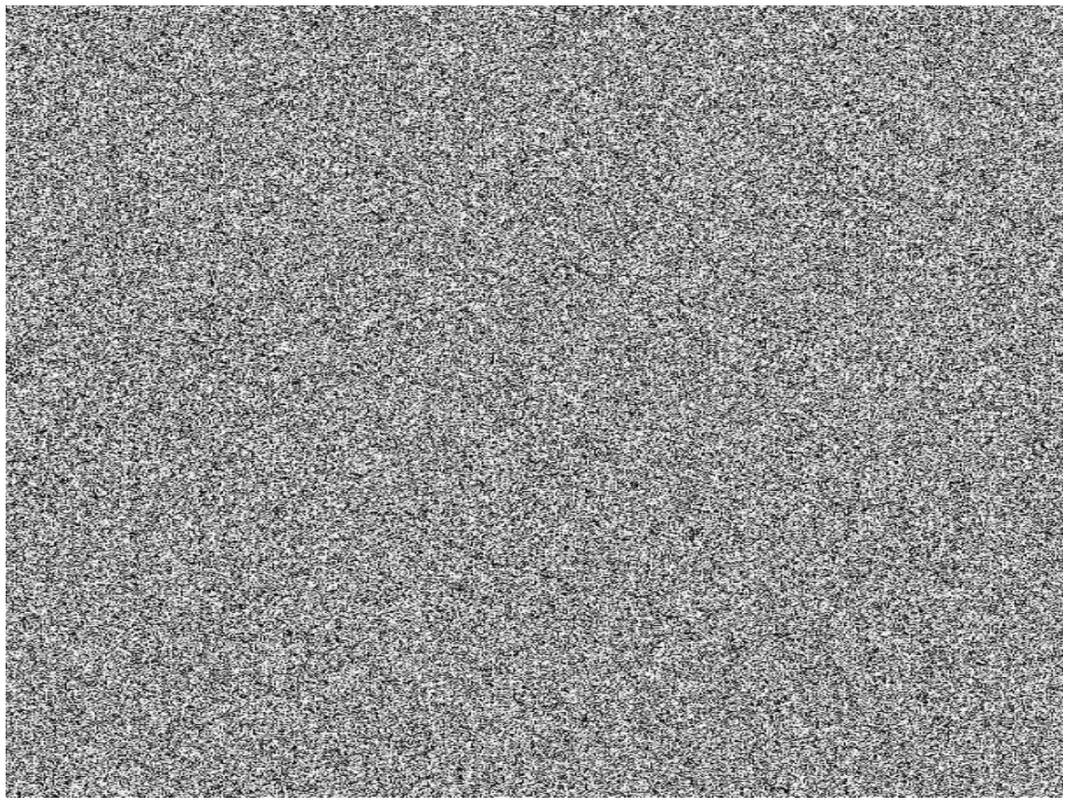
Random Images



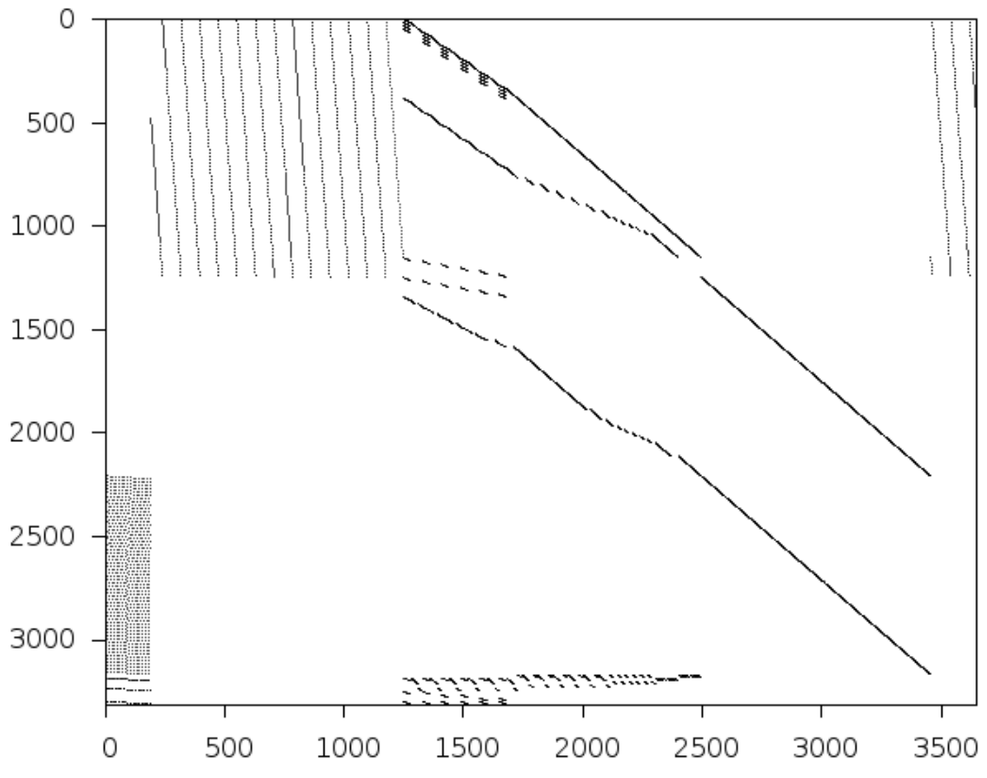
Random iid pixels

Random face (GAN)
thispersondoesnotexist.com

Random Instances



Random iid coefficients



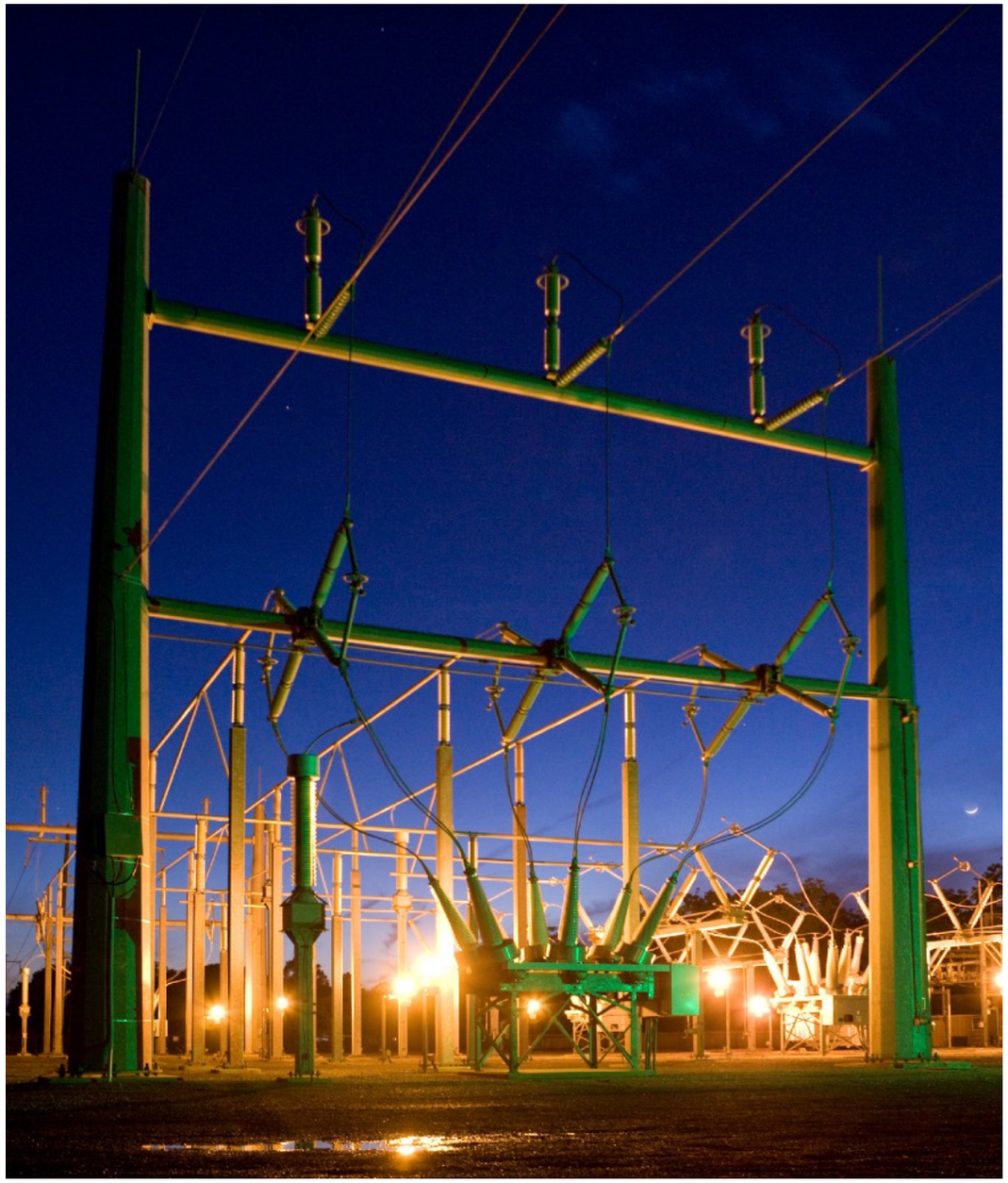
a1c1s1 from MipLib 2017

Business Applications

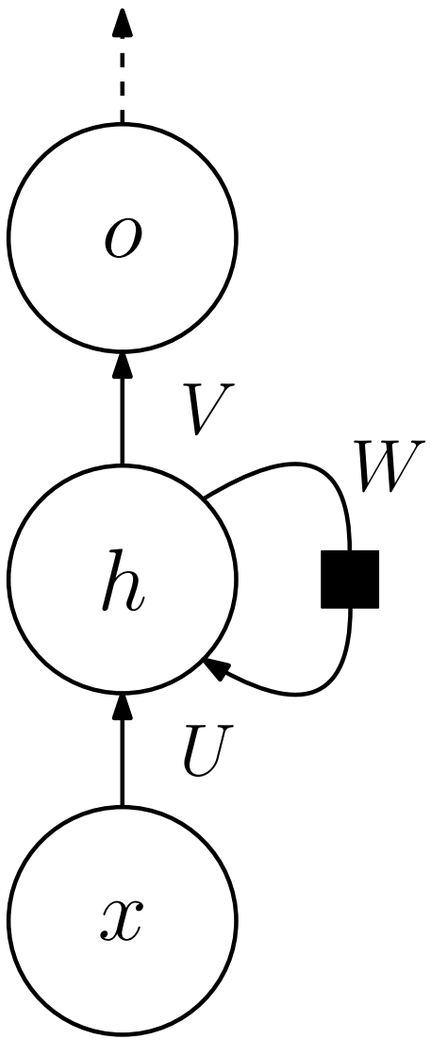
- Many businesses care about solving **similar** problems **repeatedly**

- Solvers do not make any use of this aspect

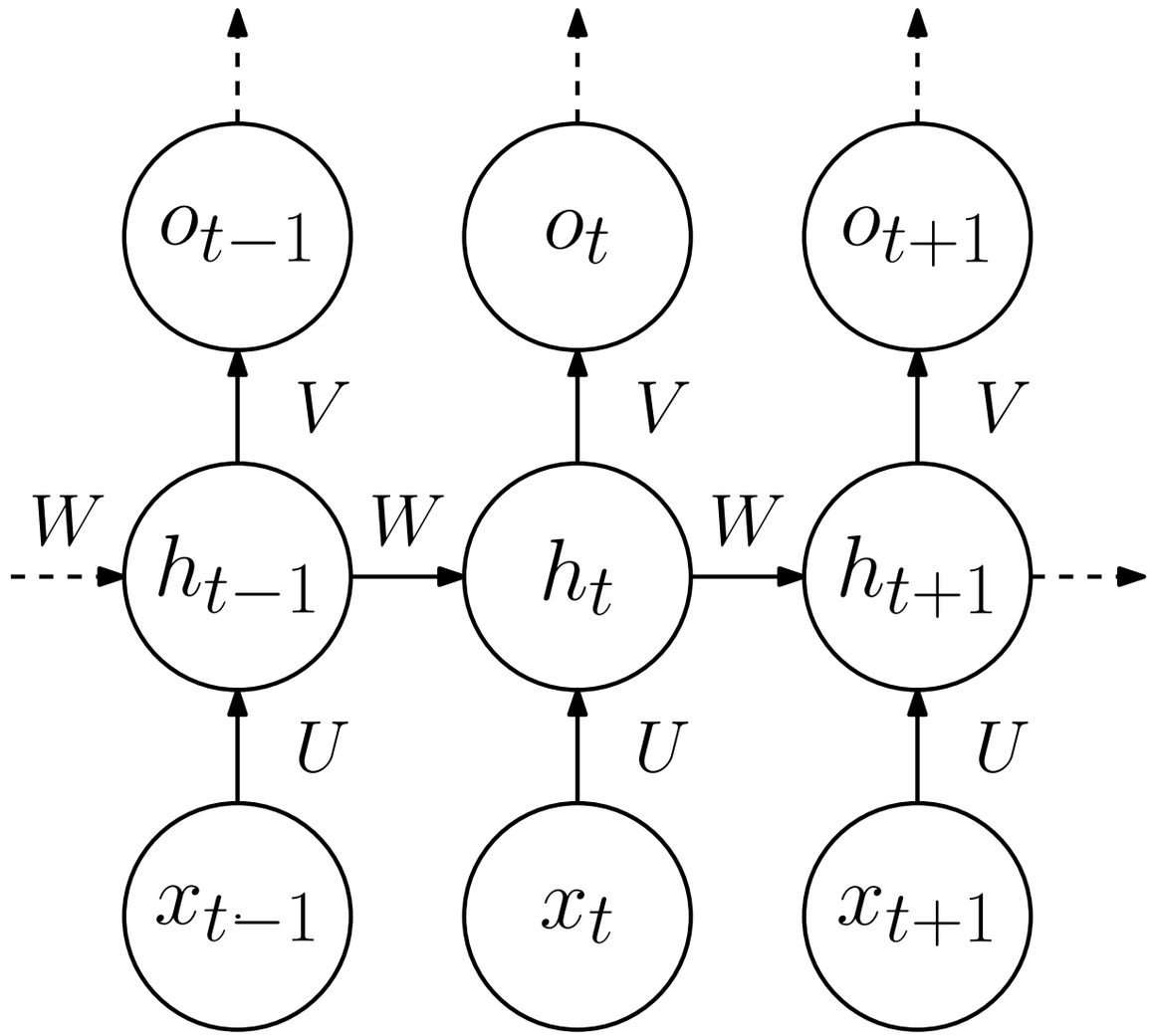
- Power systems and market
  [*Xavier et al. 2019*]
  - Schedule 3.8 kWh ($400 billion) market annually in the US
  - Solved multiple times a day
  - 12x speed up combining ML and MILP



Deep Learning Reminder

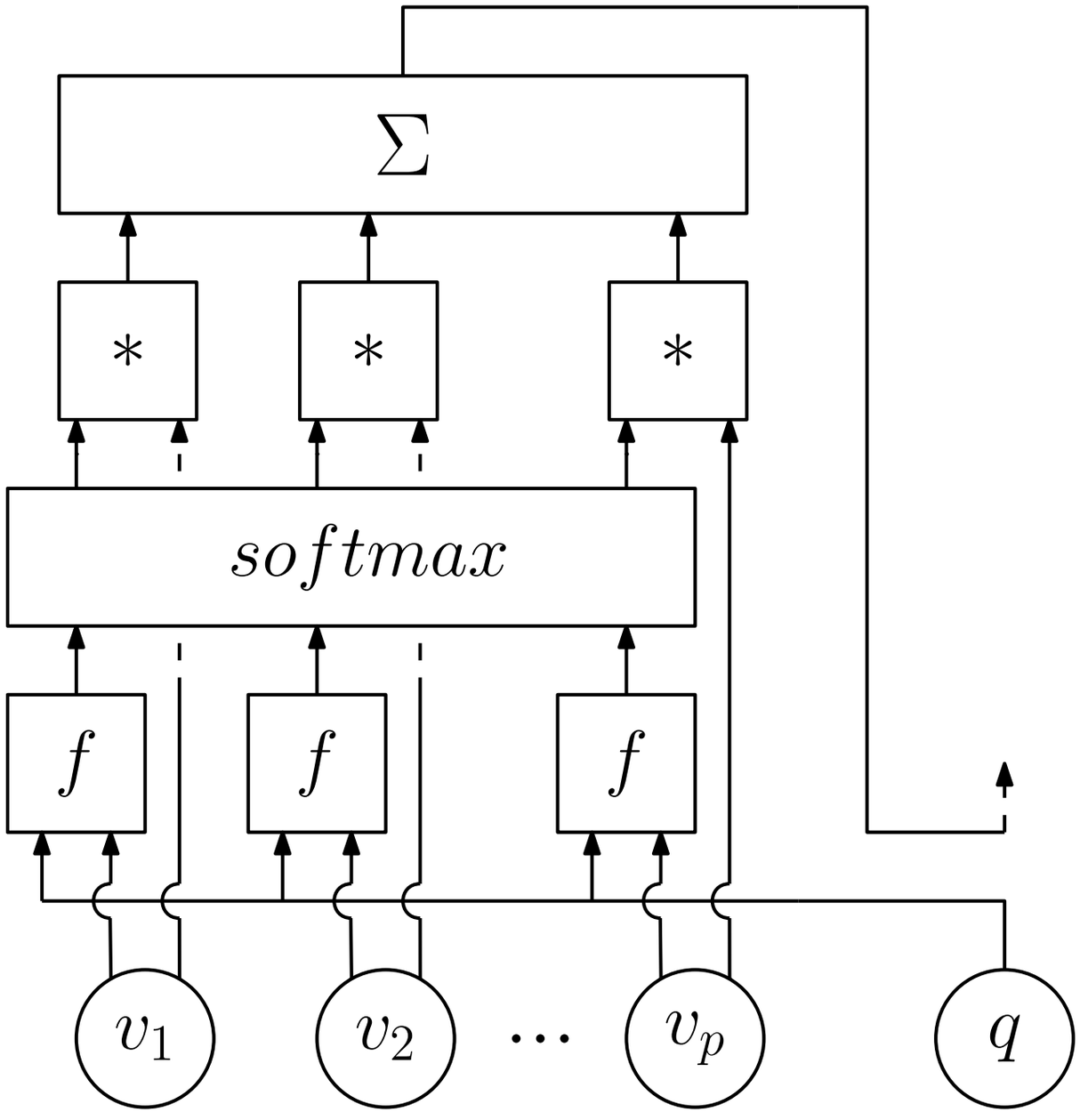


RNN folded



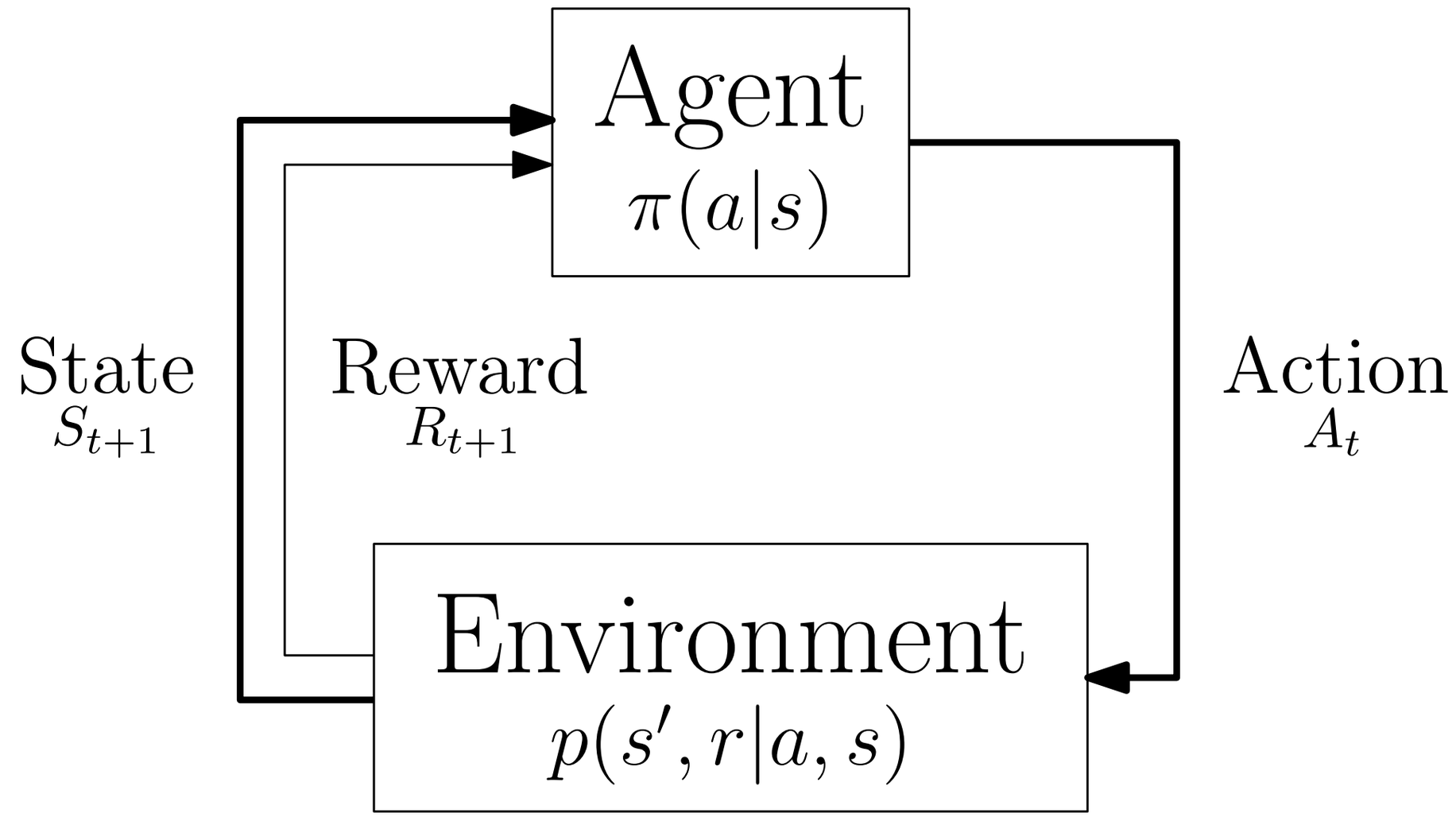
RNN unfolded

Deep Learning Reminder



Attention mechanism

Reinforcement Learning Reminder



Markov Decision Process for Reinforcement Learning

Learning Methods

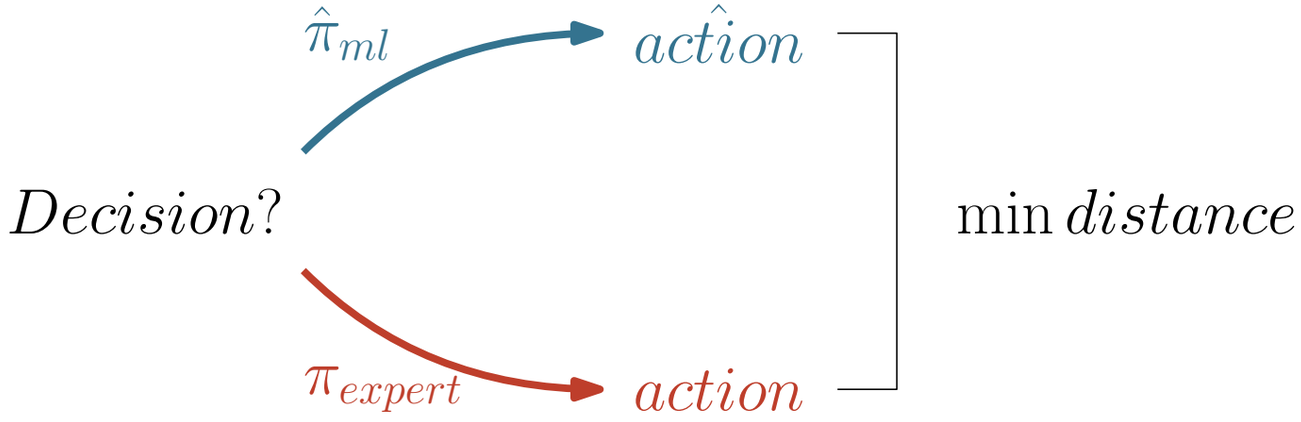
Demonstration

- An expert/**algorithm** provides a policy
- Assumes theoretical / empirical knowledge about the decisions
- Decisions are too long to compute
- Supervised imitation learning

Experience

- Learn and discover new policies (better hopefully)
- Unsatisfactory knowledge (not mathematically well-defined)
- Decisions are too heuristic
- Reinforcement learning

Demonstration



- Approximating strong branching
  [*Marcos Alvarez et al. 2014, 2016, 2017*][*Khalil et al. 2016*]

- Approximating lower bound improvement for cut selection
  [*Baltean-Lugojan et al. 2018*]

- Approximating optimal node selection
  [*He et al. 2014*]

Experience

- Learning greedy node selection (e.g. for TSP)
  [*Khalil et al 2017a*]

- Learning TSP solutions
  [*Bello et al. 2017*][*Kool and Welling 2018*][*Emami and Ranka 2018*]

Not mutually exclusive

Supervised

- Cannot beat the expert (an algorithm)
  → only interesting if the approximation is faster
- Can be unstable
- Cannot cope with equally good actions

Reinforcement

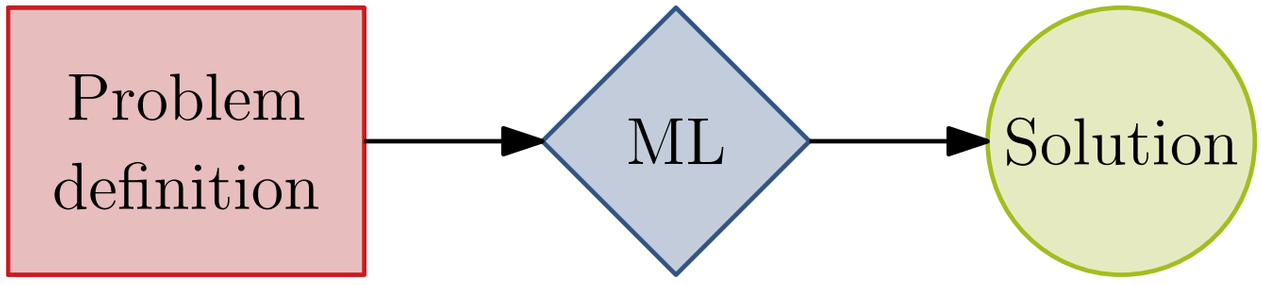
- Reinforcement can potentially discover better policies
- Harder, with local maxima (exploration difficult)
- Need to define a reward

Better **combined**!

Algorithmic Structure

- *How do we build such algorithm? How do we mix OR with ML?*
- *How do we keep guarantees provided by OR algorithms (feasibility, optimality)?*

End to End Learning

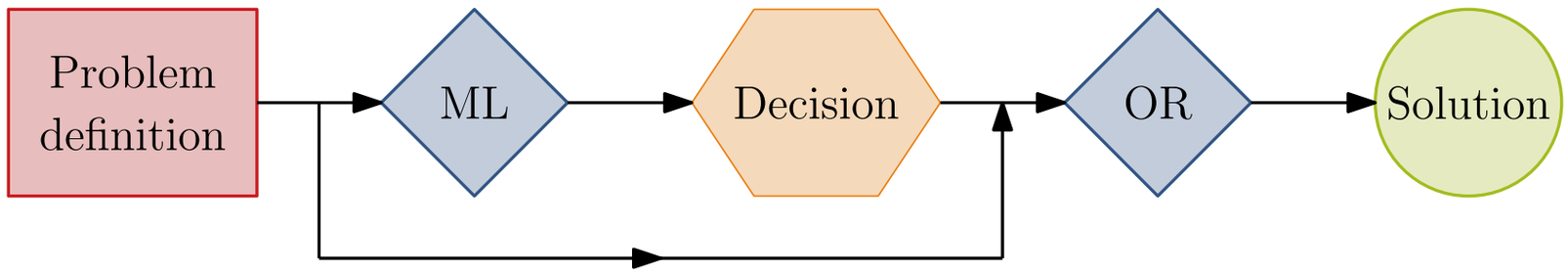


- Learning TSP solutions
  [*Bello et al. 2017*][*Kool and Welling 2018*][*Emami and Ranka 2018*]
  [*Vinyals et al. 2015*][*Nowak et al. 2017*]

- Predict aggregated solutions to MILP under partial information
  [*Larsen et al. 2018*]

- Approximate obj value to SDP (for cut selection)
  [*Baltean-Lugojan et al. 2018*]

Learning Properties



- Use a decomposition method
  [*Kruber et al. 2017*]

- Linearize an MIQP
  [*Bonami et al. 2018*]

- Provide initial cancer treatment plans to inverse optimization
  [*Mahmood et al. 2018*]

Learning Repeated Decisions

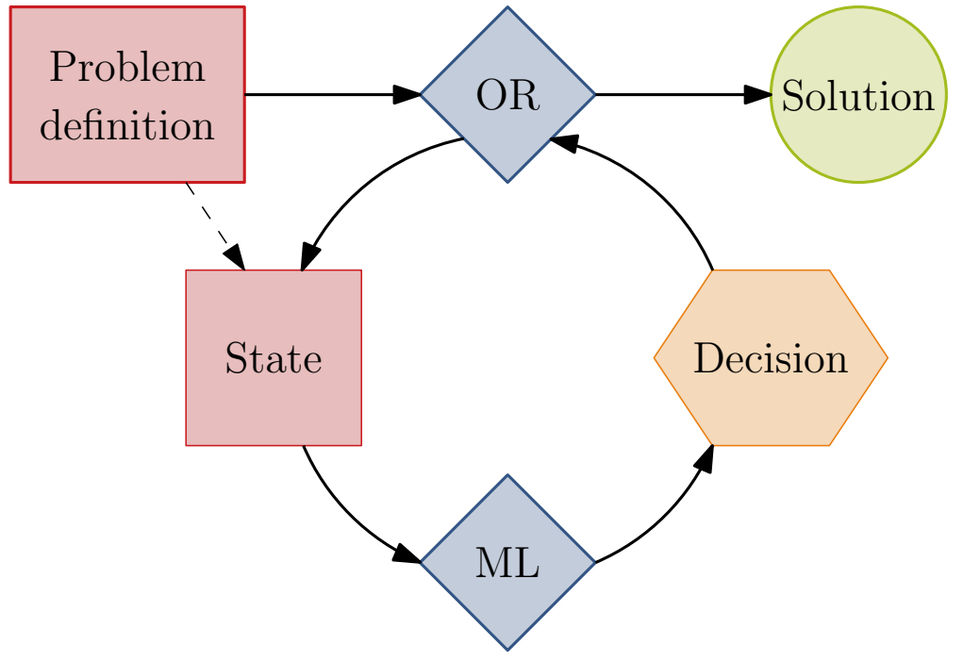
- Learning where to run heuristics in B&B
  [*Khalil et al. 2017b*]

- Learning to branch
  [*Lodi and Zarpellon 2017*] (survey)

- Learning gradient descent
  e.g. [*Andrychowicz et al. 2016*]

- Predict booking decisions under unperfected information
  [*Larsen et al. 2018*]

- Learning cutting plane selection
  [*Baltean-Lugojan et al. 2018*]



Just a question of viewpoint

Evaluation

- *What are our metrics?*
- *What instances do we want to generalize to*?
  - Instance specific policies should be easier to learn, but have to be re-learned every time
  - Policies that generalize can take some training offline (multi-task learning)
    → transfer learning, fine-tuning, meta-learning
- *What distribution of instances are we interested in?*

Challenges

- *Which models and DL/RL algorithms will perform well?*
- *How do we represent the data? Should we approximate it?*
- *Can we scale?*
  - In the computations?
  - Generalizing?
  - Learning?
- *How to anticipate learning? Which distribution? How to generate data?*

References and Paper

- Xavier, A. S., Qiu, F., & Ahmed, S. (2019). Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems. ArXiv:1902.01697 [Cs, Math, Stat]. Retrieved from http://arxiv.org/abs/1902.01697

- All other references are listed in the paper:
  Bengio, Lodi, Prouvost (2018) - Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon
  https://arxiv.org/abs/1811.06128