Machine Learning for Combinatorial Optimization: a methodological tour d'horizon

Andrea Lodi
andrea.lodi@polymtl.ca

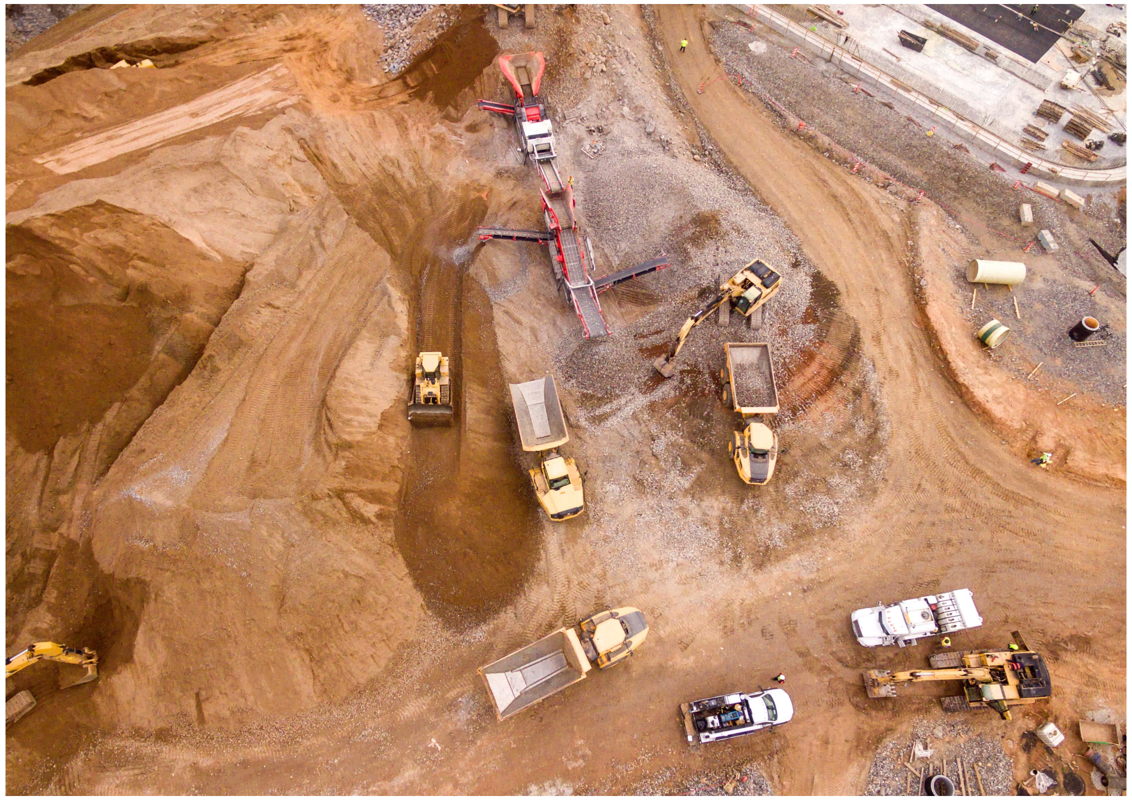
CO@Work - ZIB, Berlin - September 18, 2020

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

Mila

POLYTECHNIQUE MONTRÉAL WORLD-CLASS ENGINEERING


MAERSK
HYUNDAI
CMA CGM
Hapag-Lloyd
OOCL



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

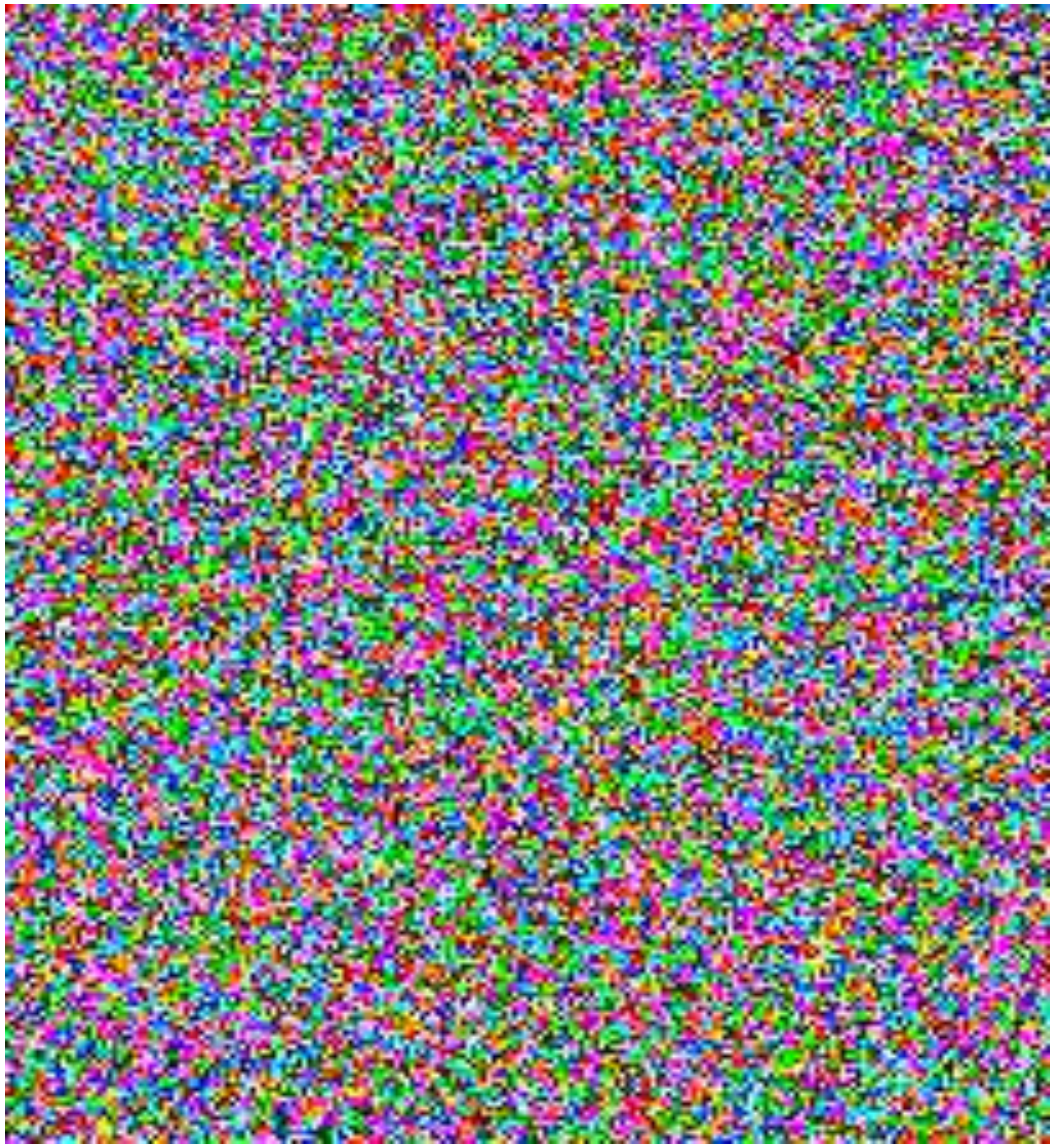
Requirement

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

The structure hypothesis

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

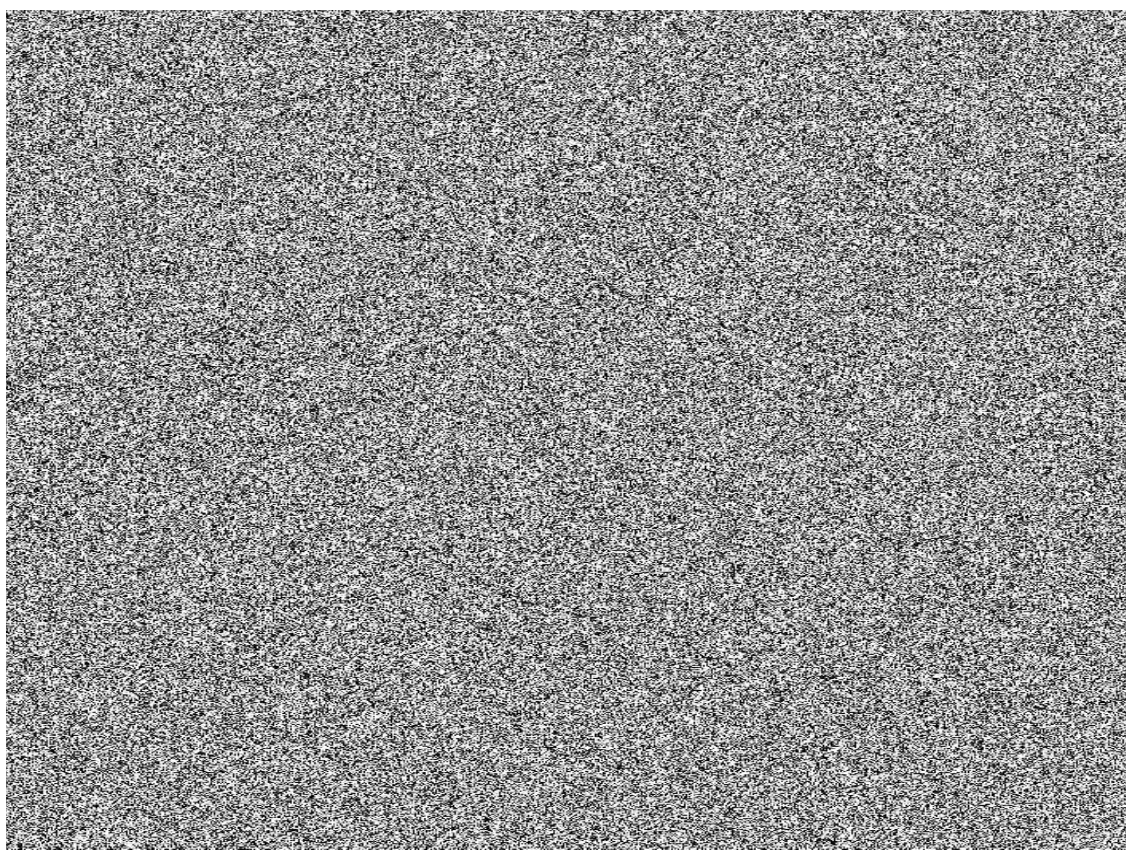
Random Images



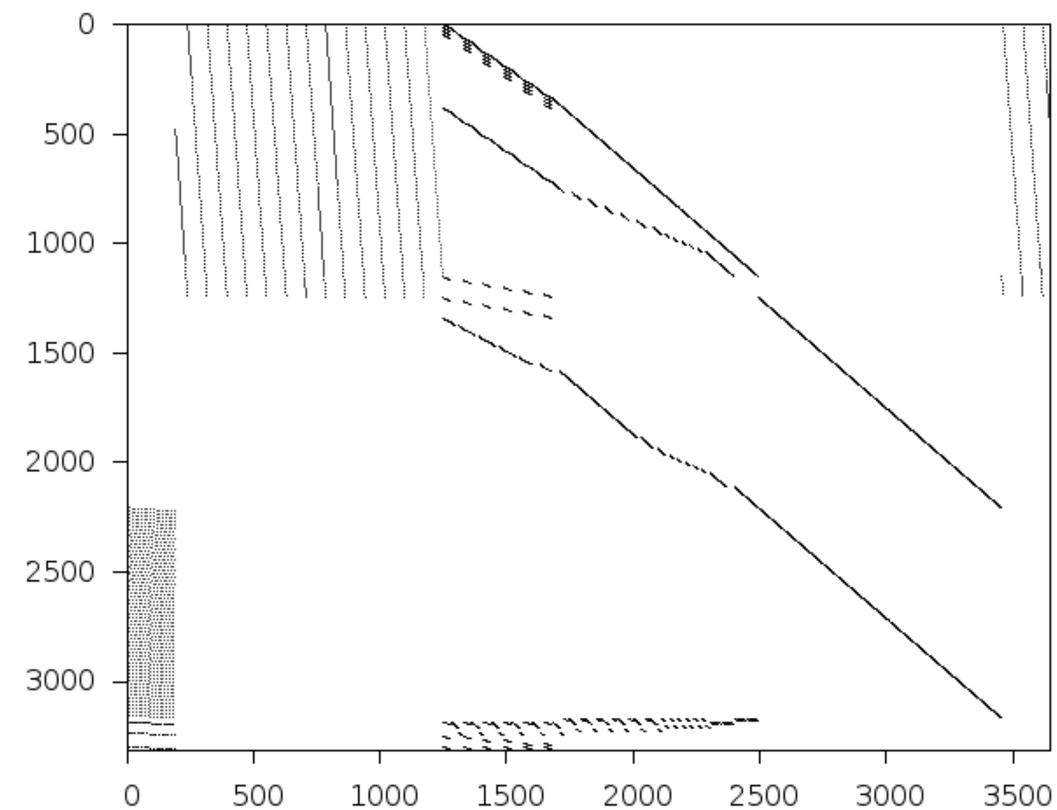
Random iid pixels

Random face (GAN)
thispersondoesnotexist.com

Random Instances



Random iid coefficients



a1c1s1 from MipLib 2017

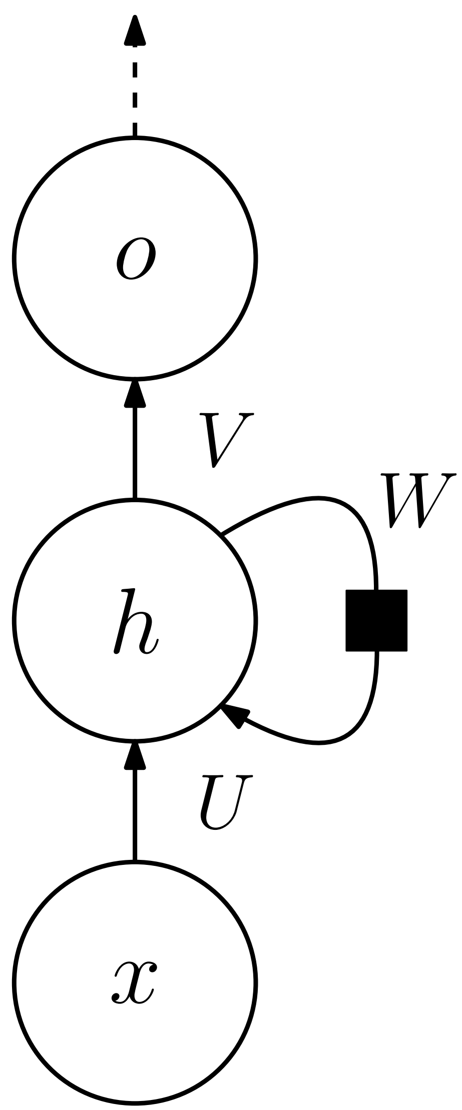
Business Applications

- 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

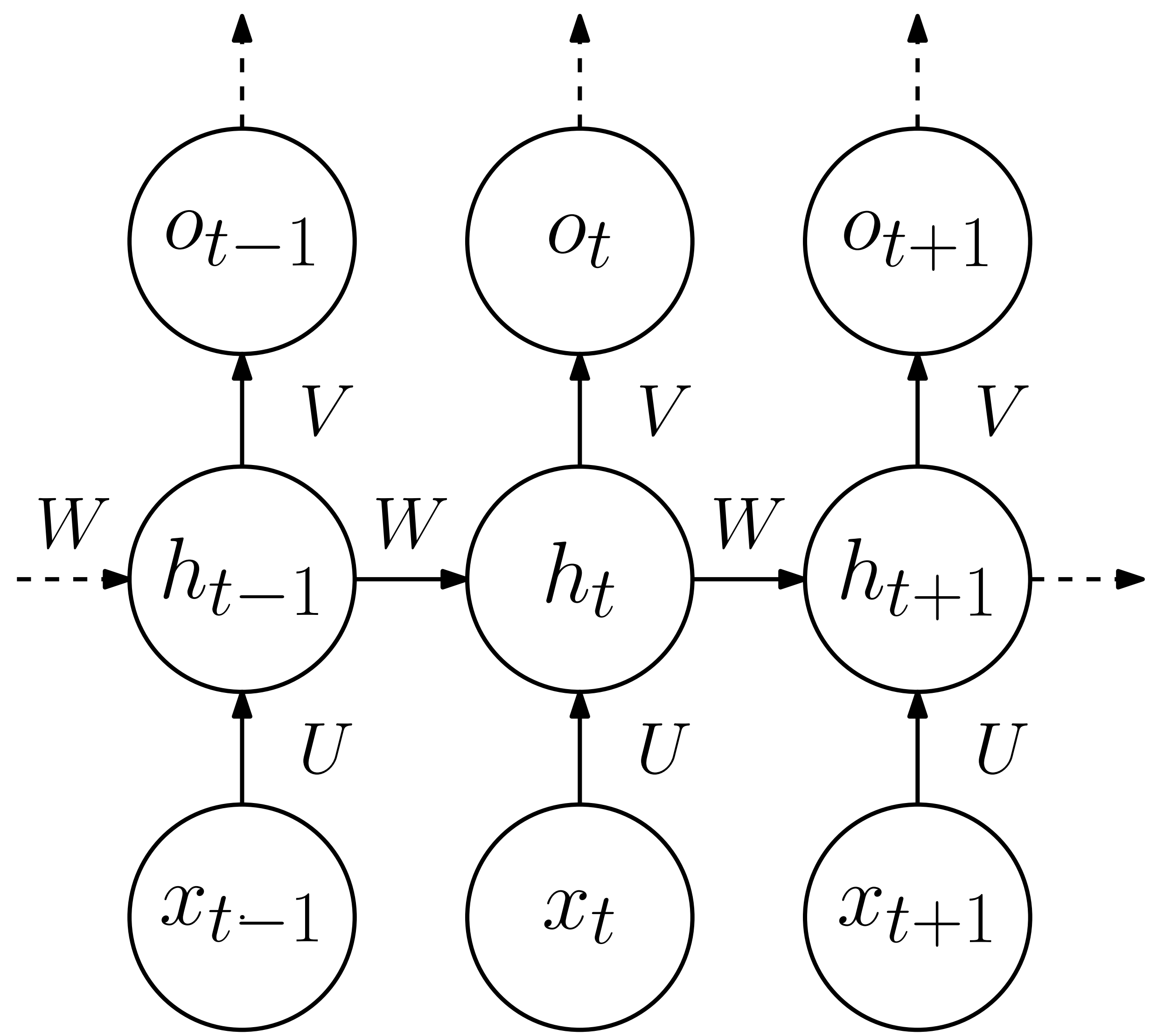
Imitation Learning Reminder

- Generally speaking, Machine Learning is a collection of techniques for
  - learning patterns in or
  - understanding the structure of **data**,
- often with the aim of performing data mining, i.e., recovering previously unknown, actionable information from the learnt data.
- Typically, in ML (IL in particular) one has to “learn'' from data (points in the so-called training set) a (nonlinear) function that predicts a certain score for new data points that are not in the training set.
- Each data point is represented by a set of features, which define its characteristics, and whose patterns should be learnt.
- The techniques used in ML are diverse, most recently artificial (deep) neural networks algorithmically boosted by first order optimization methods like gradient descent, etc.

Deep Learning Reminder

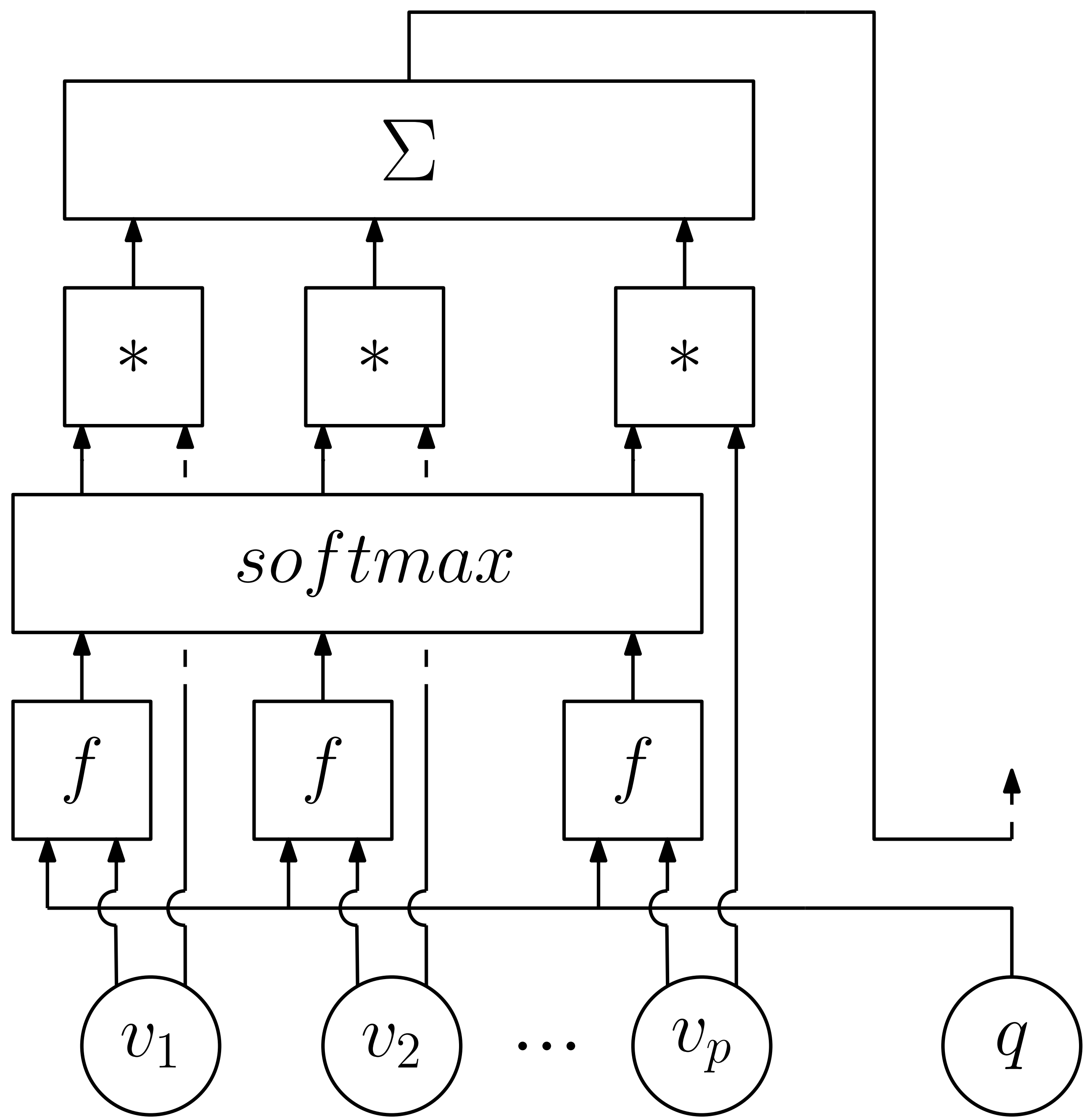


RNN folded



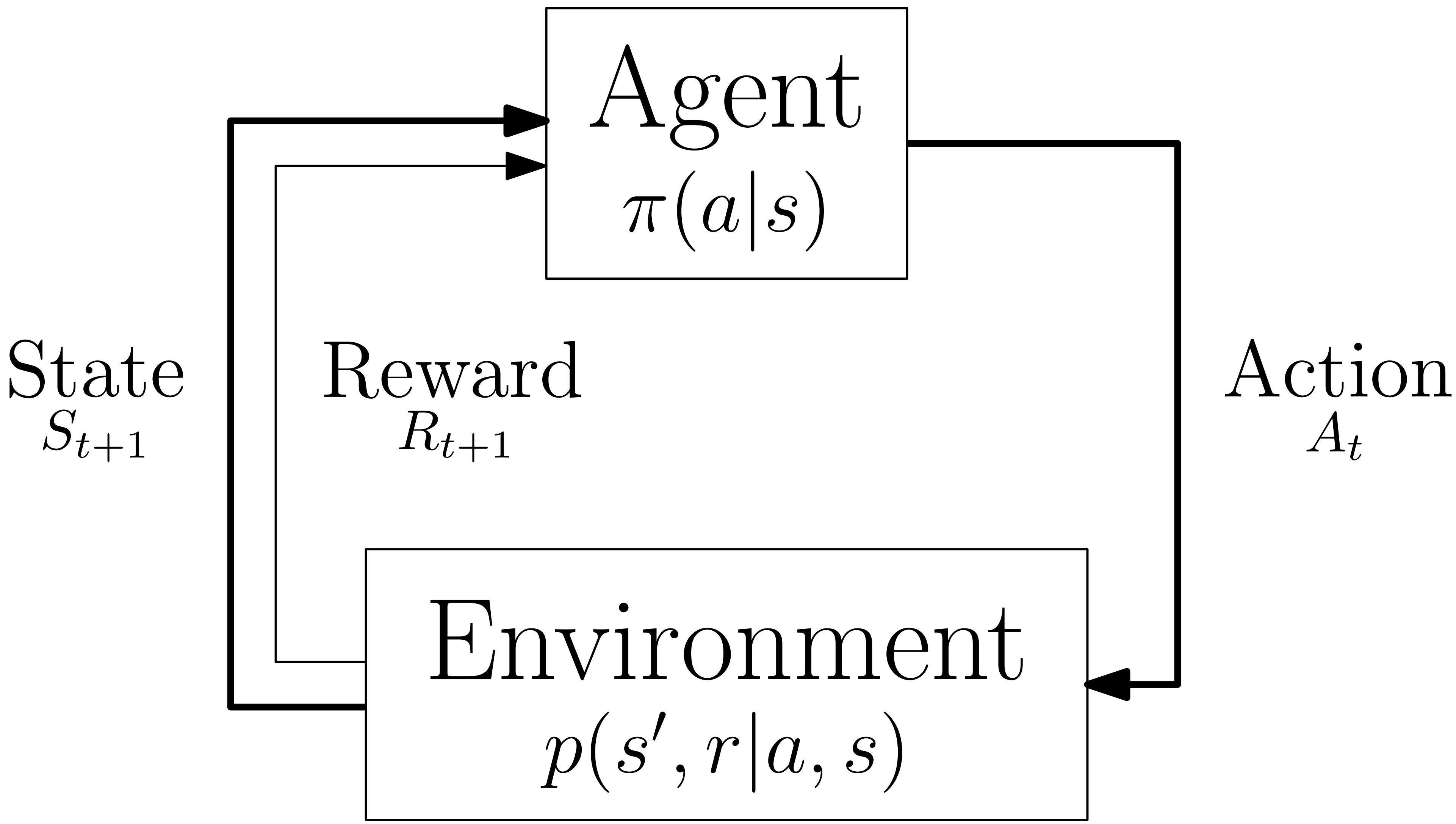
RNN unfolded

Deep Learning Reminder



Attention mechanism

Reinforcement Learning Reminder



Markov Decision Process for Reinforcement Learning

Learning Methods

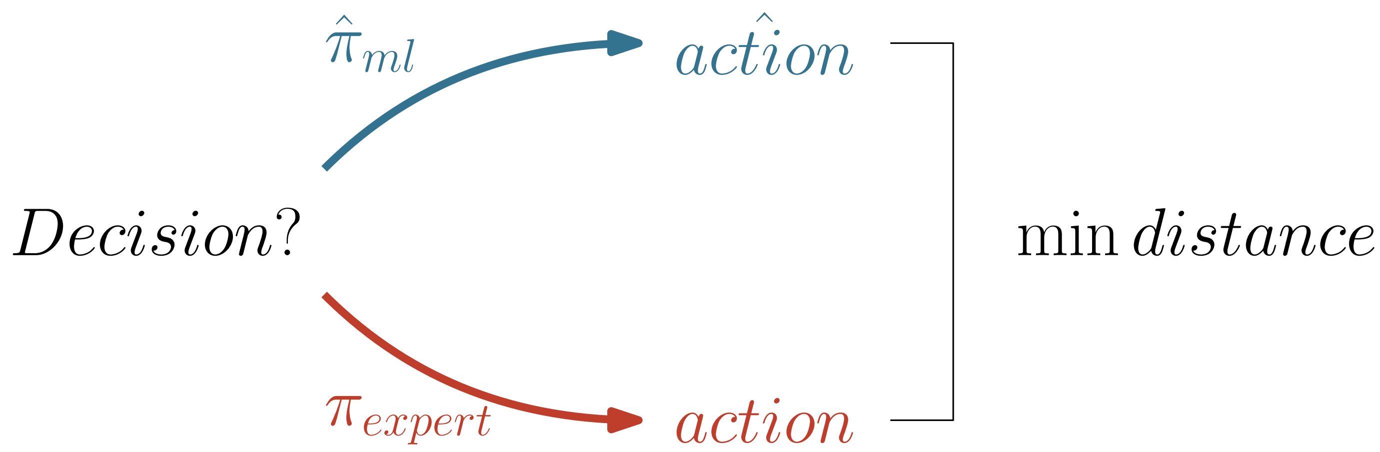
Demonstration

- 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

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

- 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

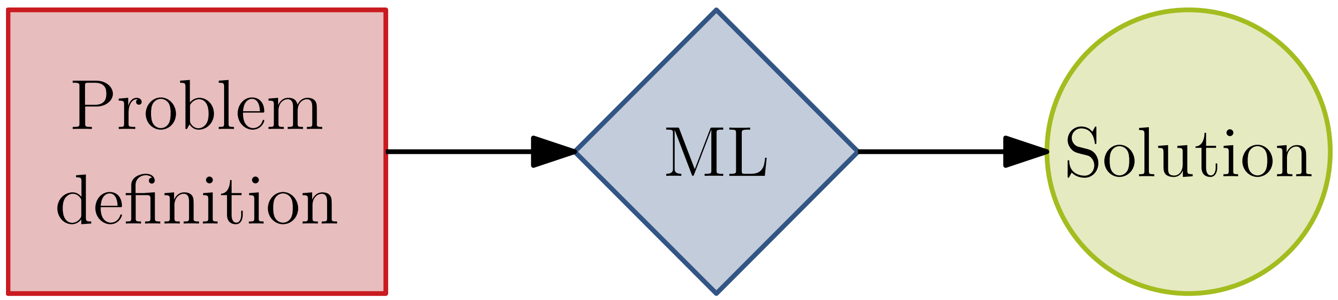
- Reinforcement can potentially discover better policies
- Harder, with local maxima (exploration difficult)
- Need to define a reward

Better **combined**!

Algorithmic Structure

- *How do we build such algorithms? 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*]

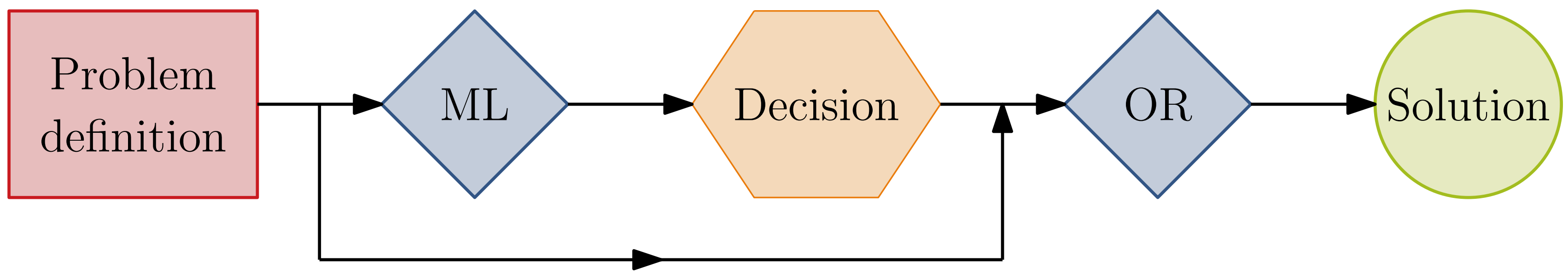
- Predict aggregated solutions to MILP under partial information (2nd stage stochastic optimization)
  [*Larsen et al. 2018*]

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

RL as a heuristic paradigm

- The End-to-End learning of the previous slide can be seen as a heuristic paradigm in itself.
- This is especially applied by exploiting Reinforcement Learning, that is certainly the ML paradigm that is closest to (discrete) optimization.
- Indeed, RL shares the same foundations of Approximate Dynamic Programming and has certainly things in common with some form of Metaheuristics.
  - The RL basic principles are exploitation and exploration, which remind of intensification and diversification.
  - The novelty certainly resides on the neural networks renewed effectiveness in dealing with / learning from data.

Learning Properties



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

- Linearize an MIQP
  [*Bonami et al. 2018, 2020*]

- Provide candidate cancer treatments to be refined by combinatorial optimization
  [*Mahmood et al. 2018*]

Learning Repeated Decisions

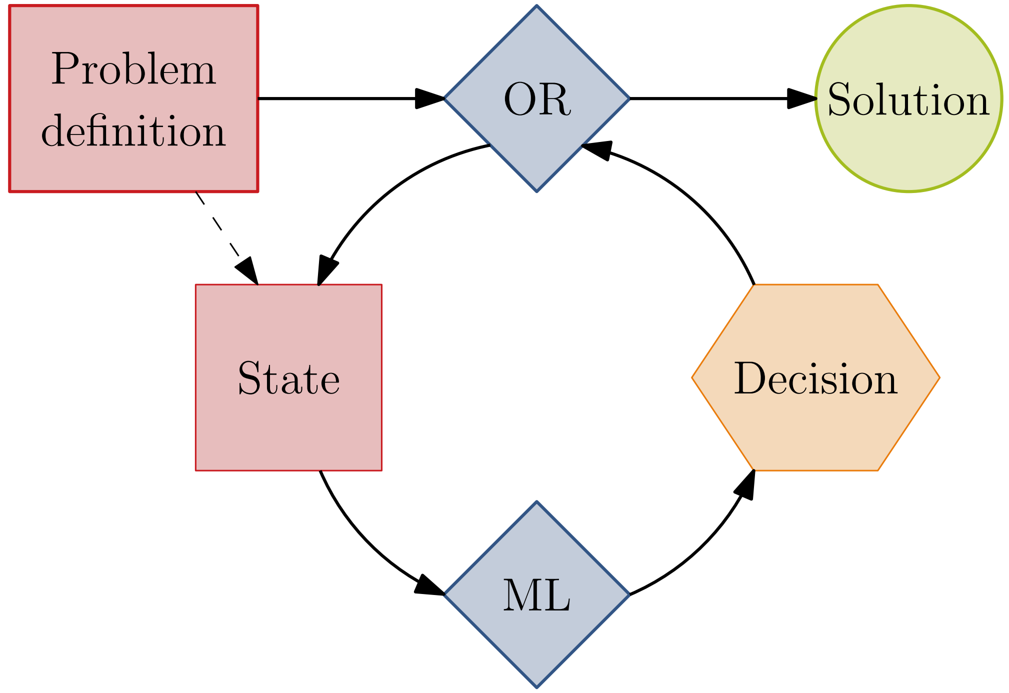
- Learning where to run heuristics in B&B
  [*Khalil et al. 2017b*]

- Learning to branch
  [*Lodi and Zarpellon 2017*] (survey)
  [Gasse et al. 2019]

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

- Predicting booking decisions under imperfect information
  [*Larsen et al. 2018*]

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



just a matter of viewpoint

Predicting tactical solutions to operational planning problems under imperfect information

E. Larsen, S. Lachapelle, Y. Bengio,
E. Frejinger, S. Lacoste-Julien & A. Lodi

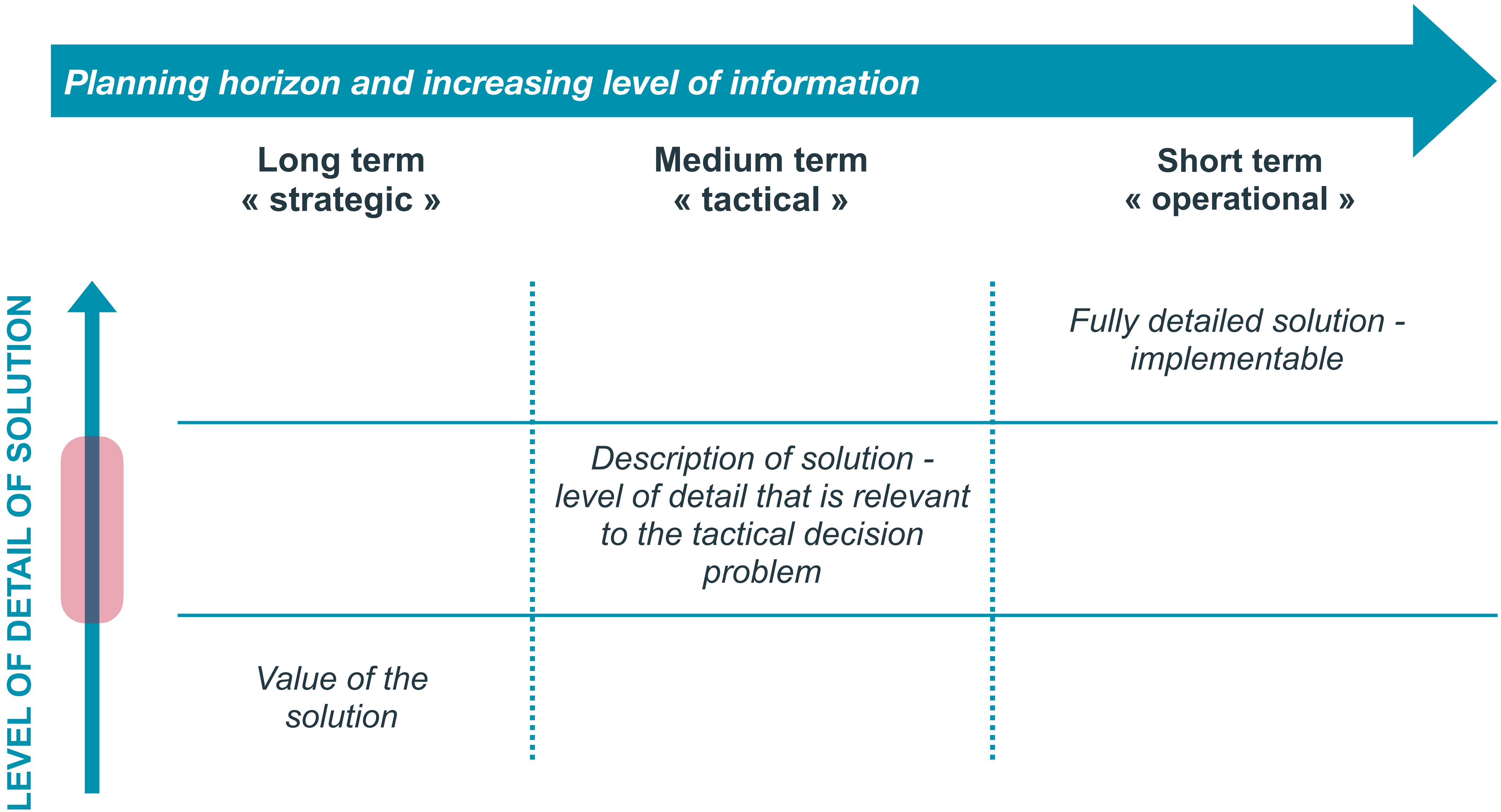
In brief:

Combine machine learning and discrete optimization to solve a problem that we could not solve with any existing methodology.

Challenges:

Very restricted computing time budget.
Imperfect information.

CONTEXT



Planning horizon and increasing level of information
Long term « strategic »
Medium term « tactical »
Short term « operational »
LEVEL OF DETAIL OF SOLUTION
Fully detailed solution - implementable
Description of solution - level of detail that is relevant to the tactical decision problem
Value of the solution

CONTEXT

Planning horizon and increasing level of information

	Medium term « tactical »	**Short term « operational »**
	Compute description of solution to operational problem under imperfect information	Operational problem of interest: Compute solution under perfect information
seconds to minutes		*Reasonable computing time - within the time budget for the operational problem*
milli-seconds	*Much shorter than the time it takes to solve the full problem under perfect information*	

COMPUTING TIME BUDGET

CONTEXT

Planning horizon and increasing level of information

Medium term
« tactical »

Compute description of solution to operational problem under imperfect information

Short term
« operational »

Operational problem of interest: Compute solution under perfect information

High-precision solution
Reasonable computing time

Solve deterministic optimization problem
mathematical programming

High-level solution
Very short computing time

Stochastic programming

Machine learning
predict the tactical solution descriptions

SOME NOTATION

Planning horizon and increasing level of information

	Medium term « tactical »		Short term « operational »	
Problem instance	Imperfect information	$\mathbf{x}_a$	Perfect information	$\mathbf{x} = [\mathbf{x}_a, \mathbf{x}_u]$
Solution		$\widehat{\mathbf{y}}^*(\mathbf{x}_a)$	Deterministic problem	$\mathbf{y}^*(\mathbf{x}) = \arg\min_{\mathbf{y} \in Y(\mathbf{x})} C(\mathbf{x}, \mathbf{y})$

Tactical solution description $\bar{\mathbf{y}}^* = g(\mathbf{y}^*(\mathbf{x}))$

APPLICATION - LOAD PLANNING

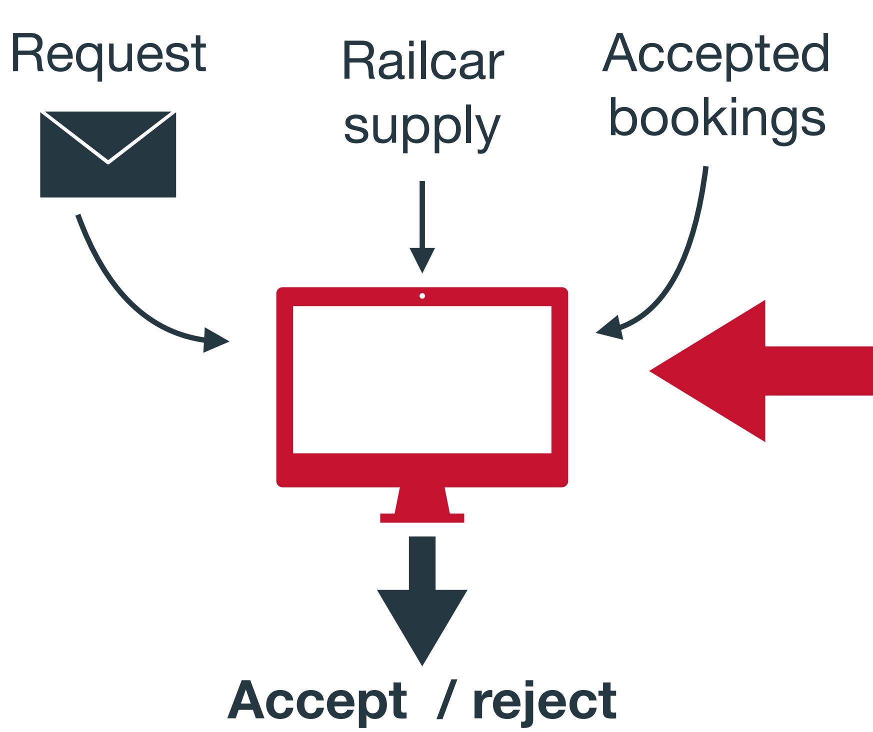
Planning horizon and increasing level of information

Medium term
« tactical »

Capacity management, e.g., bookings

Short term
« operational »

Load planning for double-stack trains



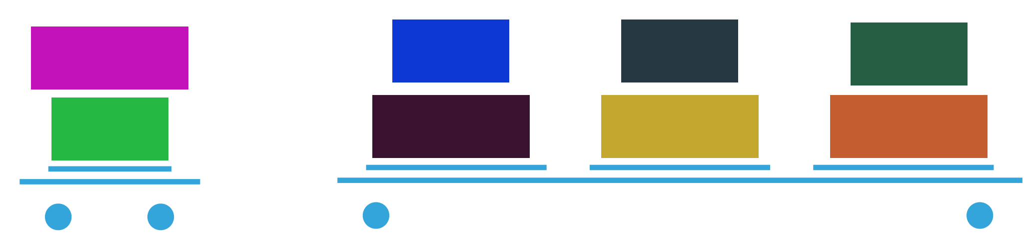
APPLICATION - LOAD PLANNING

Problem instance

$$\mathbf{x} = [\mathbf{x}_a, \mathbf{x}_u]$$

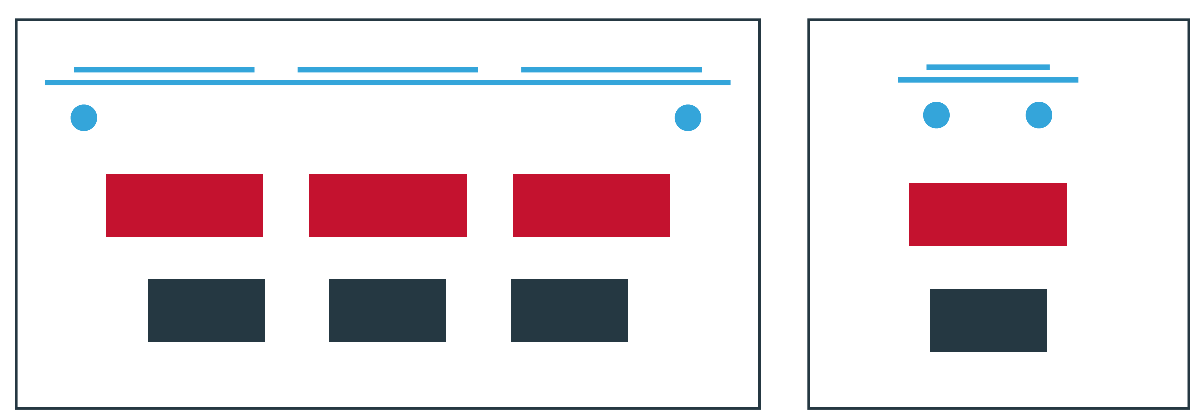
Operational solution

$$\mathbf{y}^*(\mathbf{x}) = \arg \min_{\mathbf{y} \in Y(\mathbf{x})} C(\mathbf{x}, \mathbf{y})$$



Tactical solution

$$\bar{\mathbf{y}}^* = g(\mathbf{y}^*(\mathbf{x}))$$



APPLICATION - LOAD PLANNING

- Containers have different characteristics, for example:
  - Size
  - Weight
- The loading (operational problem) of the containers onto railcars crucially depends on weight
- Weight is unknown at the tactical level

IDEA IN BRIEF

- We know how to solve the deterministic problem - let’s use that!
  - Generate a lot of data and pretend that we have perfect information - solve the discrete optimization problem with an existing solver
- Let machine learning take care of the uncertain part: hide the information that is not available at prediction time - find best possible prediction of $\bar{\mathbf{y}}^*$

$$\widehat{\mathbf{y}}^*(\mathbf{x}_\mathrm{a}) \equiv f(\mathbf{x}_\mathrm{a}; \boldsymbol{\theta})$$

State-of-the-art ML model (↑ f) Parameters (↑ $\boldsymbol{\theta}$)

TACTICAL: MULTILAYER PERCEPTRON

Input

Fixed-size vector

$\mathbf{x}_a$

Output

Fixed-size vector

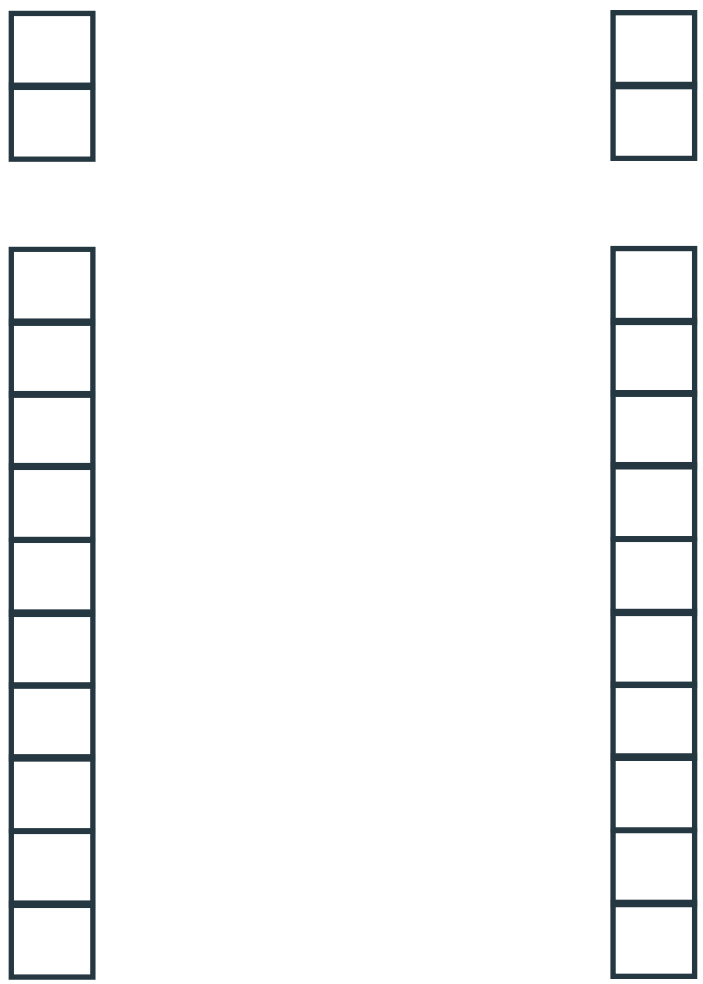
$\bar{\mathbf{y}}^*$

Nb of assignable containers of each type

Nb of containers of each type in the solution

Nb of of assignable railcars of each type

Nb of railcars of each type in the solution



TACTICAL: MULTILAYER PERCEPTRON

- Average performance of the MLP model is very good
  - **MAE** of only **2.1 containers/slots** for classes A, B and C (up to 100 platforms and 300 containers) with very small standard deviation (0.01)
- MLP results are considerably **better than benchmarks**
- The marginal value of using 100 times more observations is fairly small: modest increase in MAE from 0.985 to 1.304 on class A instances)
- **Prediction times are negligible**, milliseconds or less and with very little variation

TACTICAL: MULTILAYER PERCEPTRON

- The models trained and validated on simpler instances (A, B and C) generalize well to harder instances (D)
  - MAE of 2.85 (training on class A)
  - MAE of 0.32 (training on classes A, B and C)
  - Important variability across models with different hyper parameters when only trained on class A (MAE varies between 0.74 and 9.05)
- Numerical analysis of **feasibility**: there exists a feasible operational solution for a given predicted tactical solution in 96.6% of the instances (the share is much lower for the benchmarks)

TACTICAL: MULTILAYER PERCEPTRON

What if we solve a sample average approximation (SAA) of the two stage stochastic program?

- Class A instances
- The average absolute error of the SAA solution is similar to that of the ML algorithm: 0.82 compared to 0.985
- The computing times for SAA vary between 1 second to 4 minutes with an average of 1 minute

Exact Combinatorial Optimization with Graph Convolutional Neural Networks[1]

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi

POLYTECHNIQUE MONTRÉAL

GERAD
GROUPE D'ÉTUDES ET DE RECHERCHE EN ANALYSE DES DÉCISIONS

DATA SCIENCE FOR REAL-TIME DECISION-MAKING

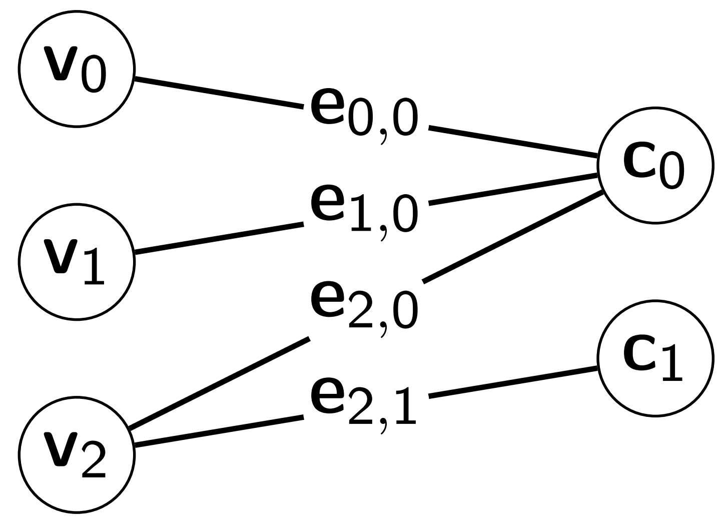
Mila

[1] In H. Wallach et al., Eds., Advances in Neural Information Processing Systems 32 (NIPS 2019), Curran Associates, Inc., 2019, 15554–15566.

Node state encoding

Natural MILP representation : constraint / variable bipartite graph

$$
\begin{aligned}
\arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\
& \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\
& \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.
\end{aligned}
$$

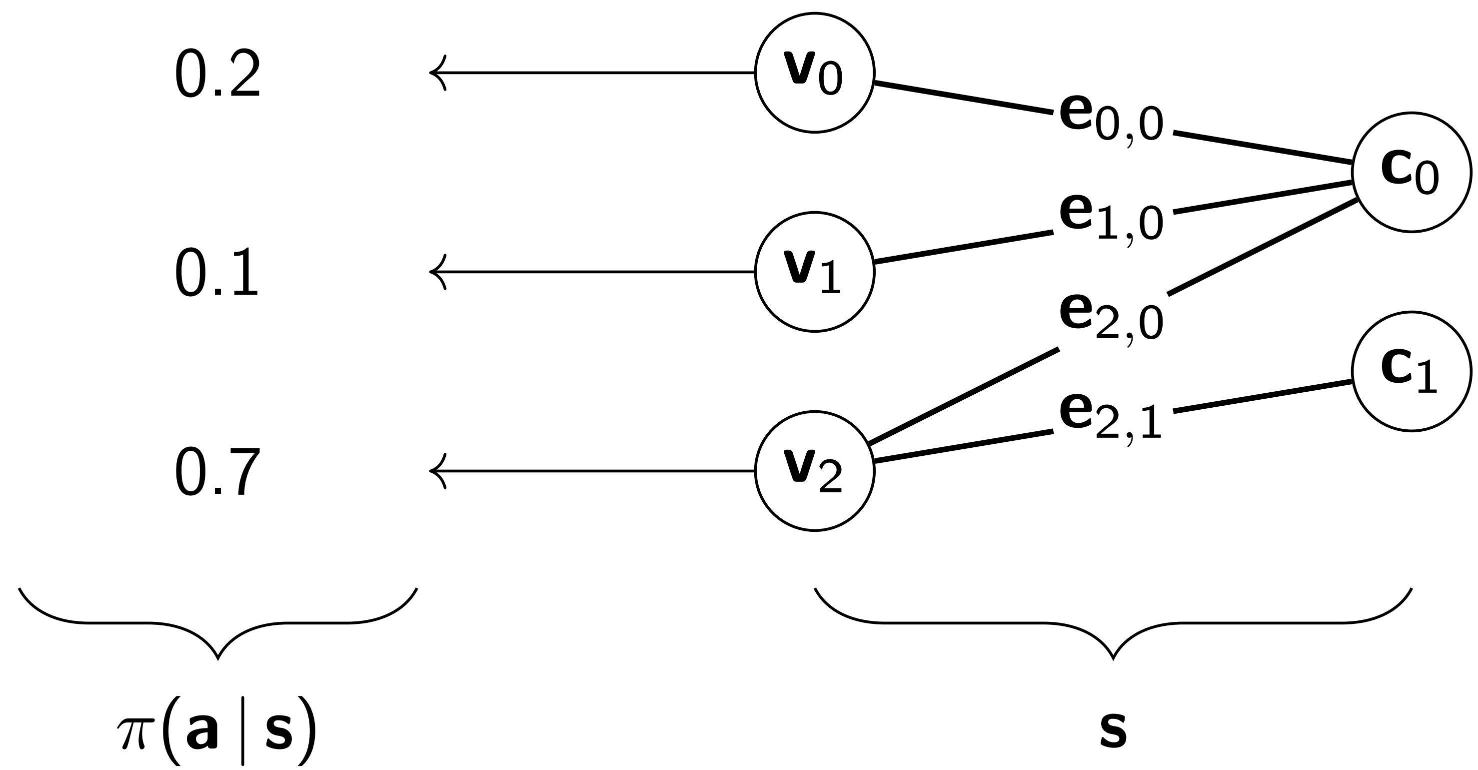


- $\mathbf{v}_i$: variable features (type, coef., bounds, LP solution...)
- $\mathbf{c}_j$: constraint features (right-hand-side, LP slack...)
- $\mathbf{e}_{i,j}$: non-zero coefficients in $\mathbf{A}$

D. Selsam et al. (2018). Learning a SAT Solver from Single-Bit Supervision.

Branching policy as a GCNN model

Neighbourhood-based updates: $\mathbf{v}_i \leftarrow \sum_{j \in \mathcal{N}_i} \mathbf{f}_\theta(\mathbf{v}_i, \mathbf{e}_{i,j}, \mathbf{c}_j)$



Natural model choice for graph-structured data

- permutation-invariance
- benefits from sparsity

T. N. Kipf and M. Welling (2016). Semi-Supervised Classification with Graph Convolutional Networks.

Strong Branching approximation

Full Strong Branching (FSB): good branching rule, but computationally expensive. Can we learn a fast, good-enough approximation ?

Imitation learning

- collect $\mathcal{D} = \{(\mathbf{s}, a^\star), \dots\}$ from an expert agent (FSB) using SCIP[2]
- estimate $\pi^\star(a \,|\, \mathbf{s})$ from $\mathcal{D}$
- \+ no reward function, supervised learning, well-behaved
- − will never surpass the expert...

Not a new idea

- Alvarez et al. (2017) predict SB scores with an XTrees model
- Khalil et al. (2016) predict SB rankings with an SVMrank model
- Hansknecht et al. (2018) do the same with a λ-MART model

[2] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 6. Technical Report. Optimization Online.

Minimum set covering[3]

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	20.19	0 / 100	16	282.14	0 / 100	215	3600.00	0 / 0	
RPB	13.38	1 / 100	63	66.58	9 / 100	2327	1699.96	27 / 65	51 022
XTrees	14.62	0 / 100	199	106.95	0 / 100	3043	2726.56	0 / 36	58 608
SVMrank	13.33	1 / 100	157	89.63	0 / 100	2516	2401.43	0 / 48	42 824
λ-MART	**12.20**	**59** / 100	161	72.07	12 / 100	2584	2177.72	0 / 54	48 032
GCNN	12.25	39 / 100	**130**	**59.40**	**79** / 100	**1845**	**1680.59**	**40** / 64	**34 527**

3 problem sizes

- 500 rows, 1000 cols (easy), training distribution
- 1000 rows, 1000 cols (medium)
- 2000 rows, 1000 cols (hard)

Pays off: better than SCIP's default in terms of solving time.
Generalizes to harder problems !

[3] E. Balas and A. Ho (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. In: Combinatorial Optimization. Springer, pp. 37–60.

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?*

Summary

- Overall we start to have (mild) evidence that such an integration approach could be effective.
- The needs for discrete optimization are clear:
  - dealing with big (and uncertain) data
  - introduce in the process more modern statistics
  - “learn” to repeatedly solve the “same instance”
- The additional challenge is how to integrate all that in a new sound methodology, including technology issues like the CPU vs GPU relationship *[Gupta et al. 2020]*.

References

- Y. Bengio, A. Lodi, A. Prouvost (2018) - *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon.* https://arxiv.org/abs/1811.06128 (EJOR 2020)

- A.S. Xavier, F. Qiu, S. Ahmed (2019) - *Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems.* https://arxiv.org/abs/1902.01697

- E. Larsen, S. Lachapelle, Y. Bengio, E. Frejinger, S. Lacoste-Julien, A. Lodi (2019) - *Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information.* https://arxiv.org/abs/1901.07935

- M. Gasse, D. Chételat, N. Ferroni, L. Charlin, A. Lodi (2019) - *Exact Combinatorial Optimization with Graph Convolutional Neural Networks* In H. Wallach et al., Advances in Neural Information Processing Systems 32 (NIPS 2019), Curran Associates, Inc., 2019, 15554--15566.