Machine Learning for Combinatorial Optimization

Andrea Lodi
andrea.lodi@cornell.edu

BUILding a DIgital Twin: requirements, methods, and applications

Roma, October 20, 2023

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CANADA
EXCELLENCE
RESEARCH
CHAIR

DATA SCIENCE
FOR REAL-TIME
DECISION-MAKING

CORNELL
TECH

HOME OF THE **JACOBS**
TECHNION-CORNELL
INSTITUTE

Preamble and Disclaimer

The use of Machine Learning (ML) for Combinatorial Optimization (CO) problems has been ubiquitous in the last 5 years at the very least.

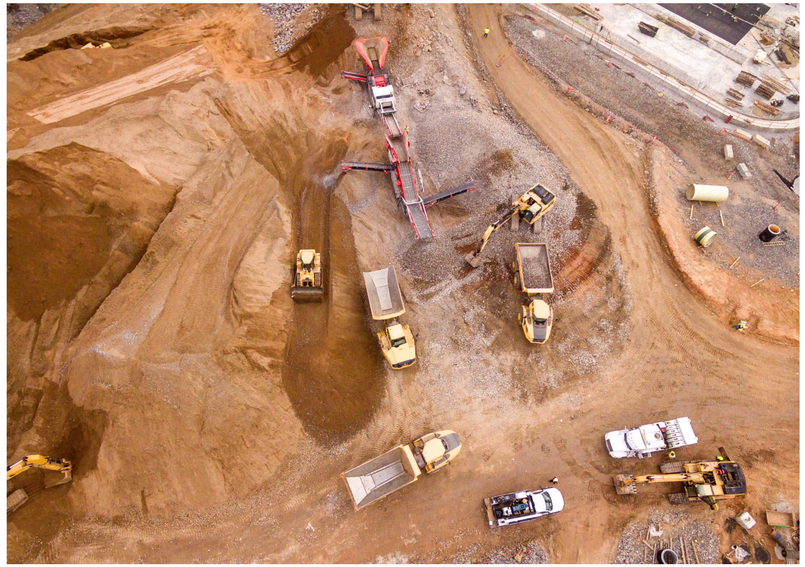
This is due to the incredible success of ML, especially deep learning, in beating human capabilities in image recognition, language processing and sequential games.

Those successes led to ask natural questions about using modern statistical learning in other disciplines, Combinatorial Optimization being one of them.

The new, very recent frontier of Generative Artificial Intelligence is yet another story and is **not** covered by this talk.

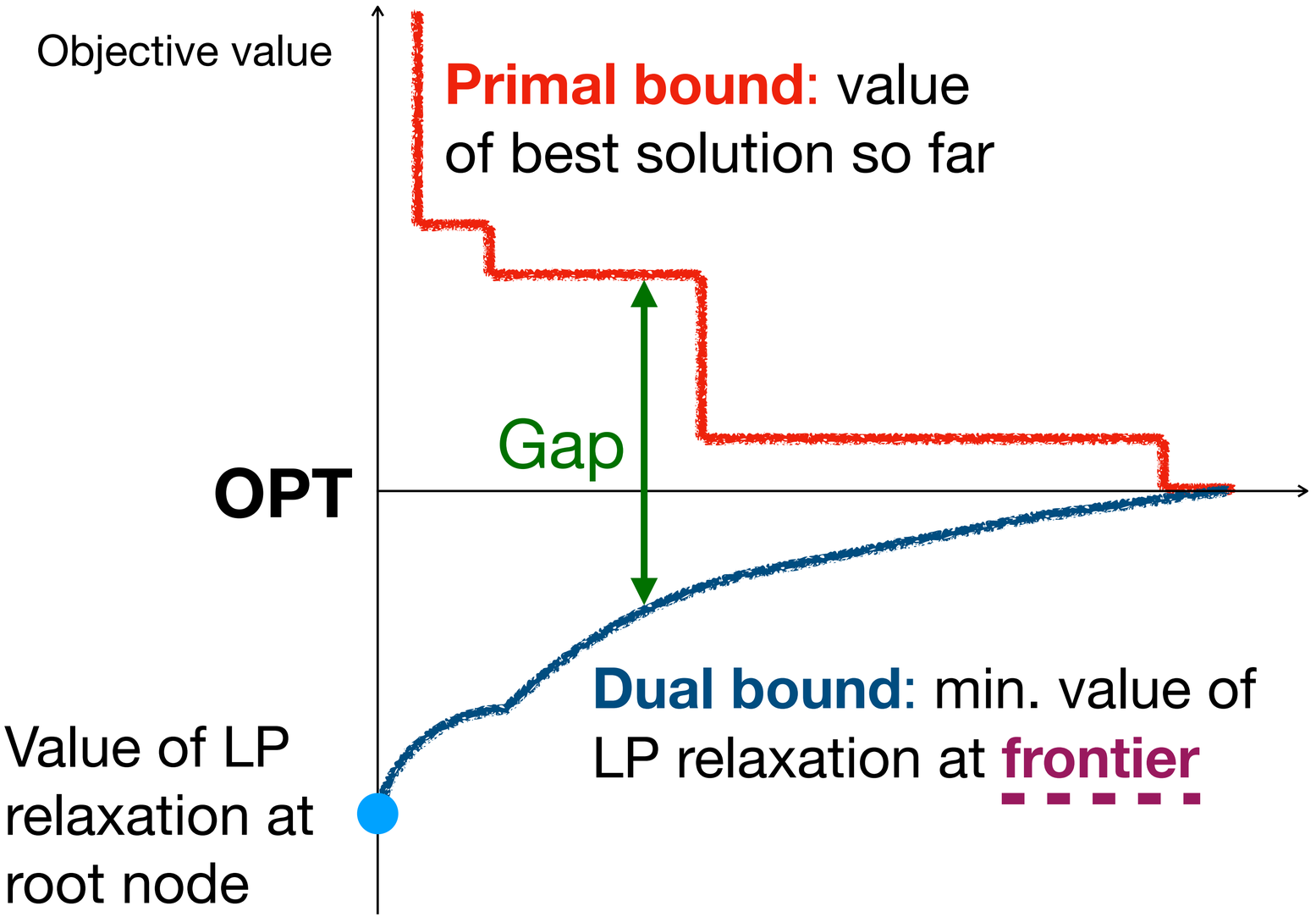
Discrete Optimization: where?

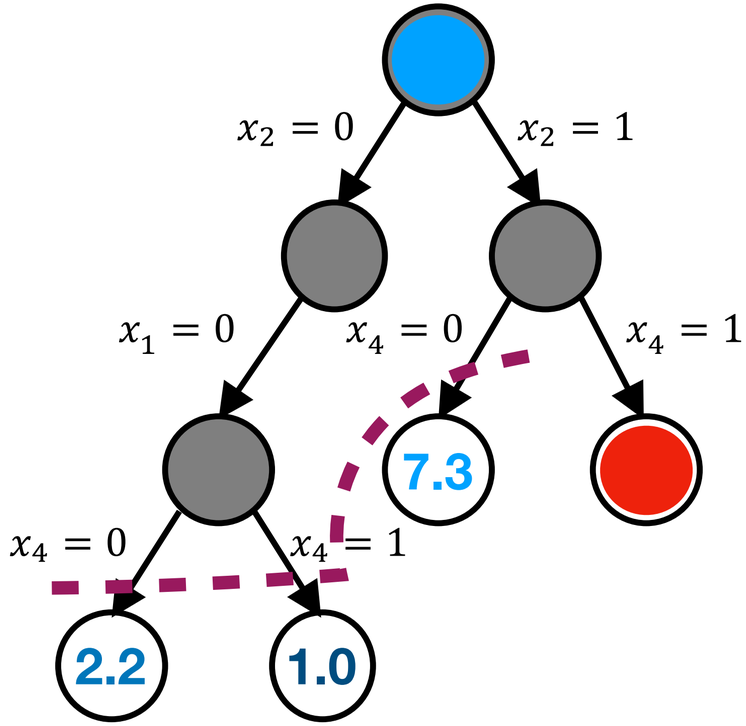

U A S C
HYUNDAI
COSCO
MAERSK
OOCL
CMA CGM
Hapag-Lloyd
CSAV



Discrete Optimization: how?

$$\min_x c^T x \text{ s.t. } Ax \leq b, x \in \{0,1\}^n$$





Search tree nodes

Slide courtesy of E. Khalil

Outline

The talk is divided in two parts:

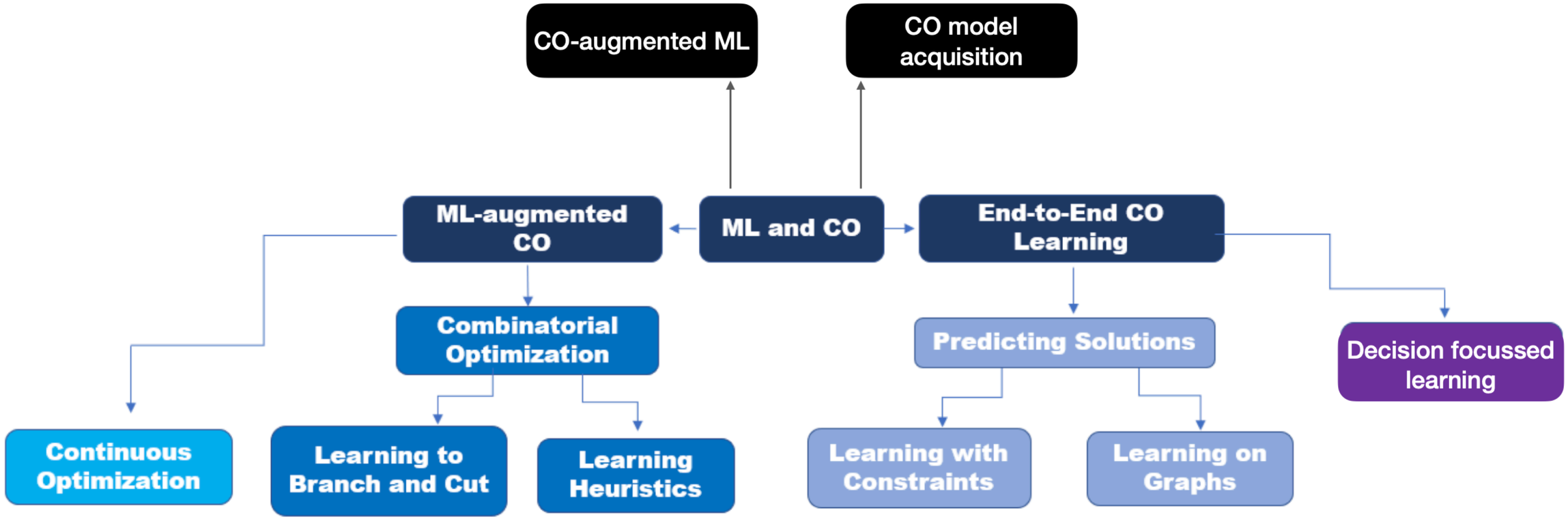
- Part 1 sets the ground of what using ML for Combinatorial Optimization means and discusses some of fast-growing literature in a methodological fashion.

- Part 2 introduces a business perspective on how ML could successfully help Combinatorial Optimization on applications by using an example in logistics.

Part 1: ML for Combinatorial Optimization

Schematic Overview

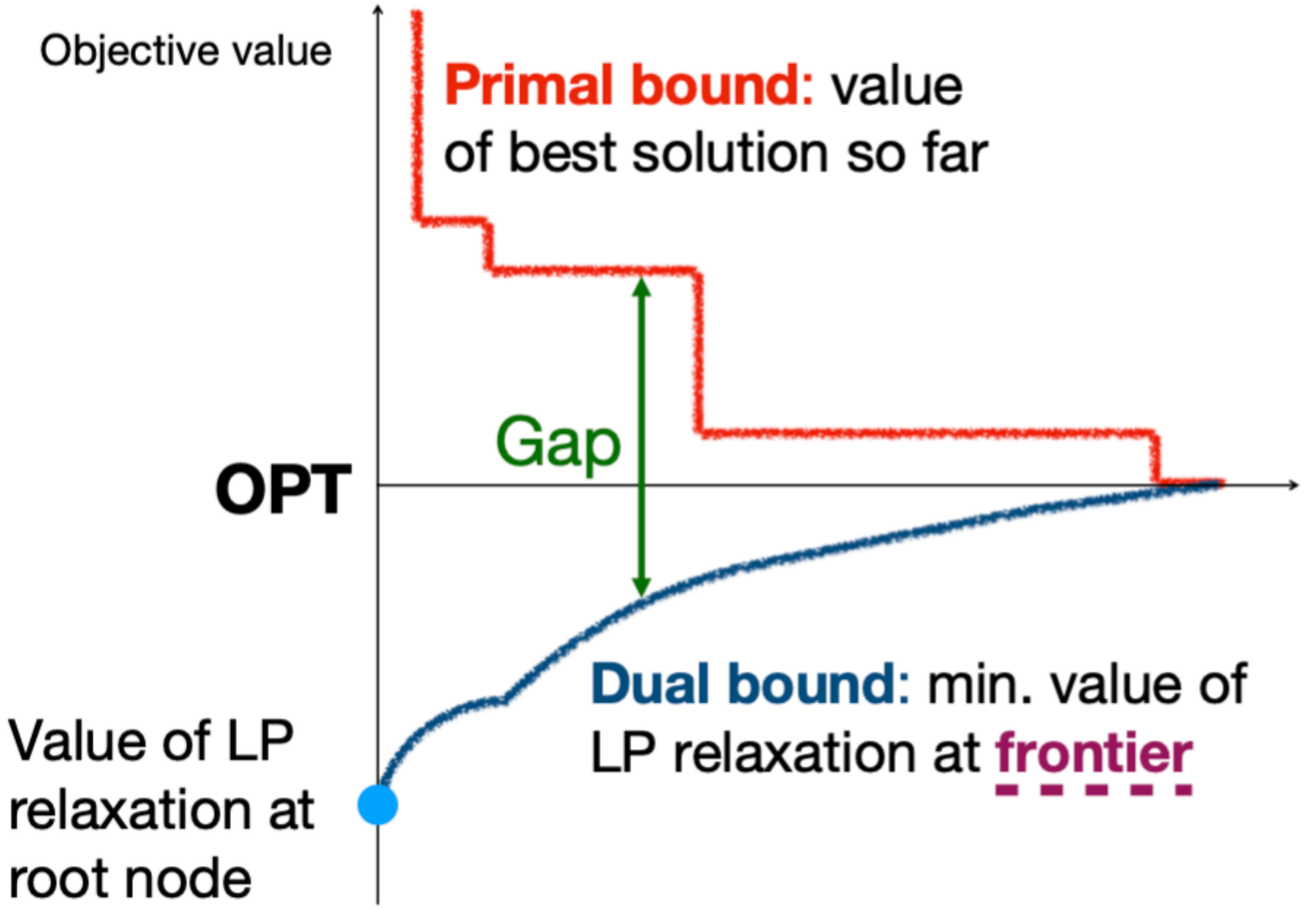
Slide courtesy of N. Yorke-Smith

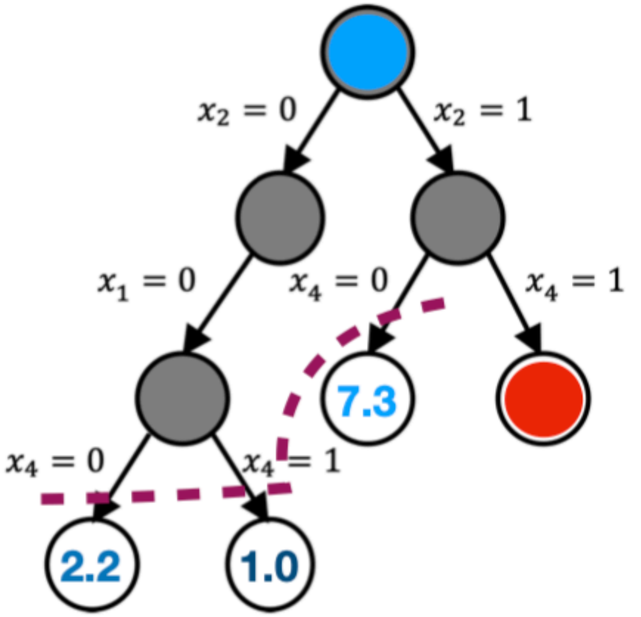


Y. Bengio, A. Lodi, A. Prouvost: Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon, EJOR 2021, 405-421

J. Kotary, F. Fioretto, P. Van Hentenryck, B. Wilder: End-to-End Constrained Optimization Learning: A Survey. IJCAI 2021: 4475-4482

$$\min_x c^T x : Ax \leq b, x \in \{0,1\}^n$$





Search tree nodes

Slide courtesy of E. Khalil

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

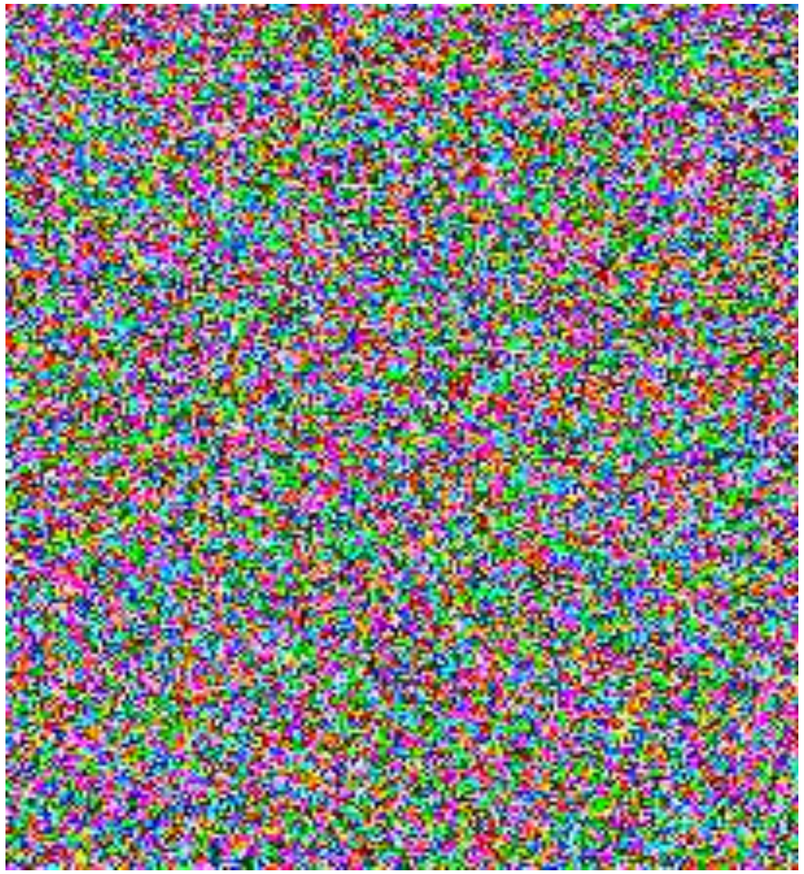
Question

- Can **Machine Learning** methods as Imitation Learning, Reinforcement Learning and all the recent powerful techniques (e.g., Deep Learning) and architectures (e.g., Graph Neural Networks) help **Combinatorial Optimization** algorithms by dealing with the issues above ("too slow" and / or "too heuristic")?

Requirement

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

Random Images

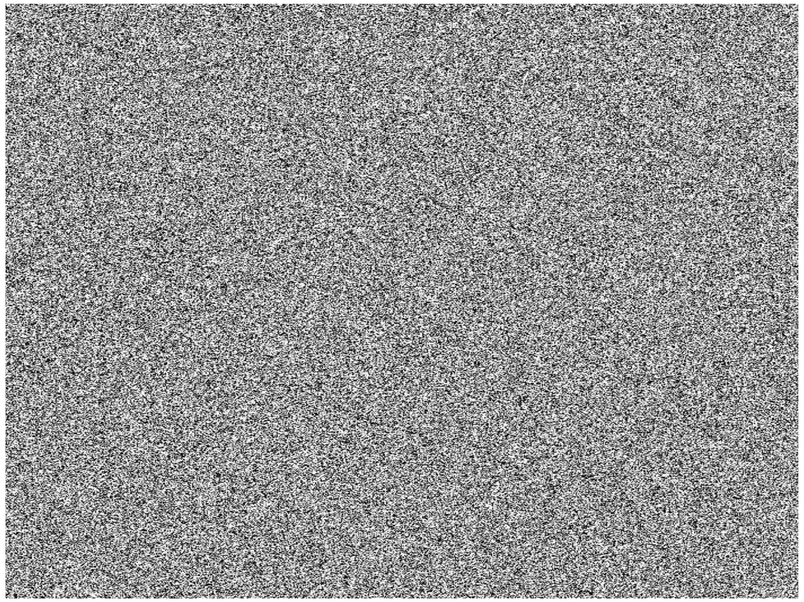


Random iid pixels

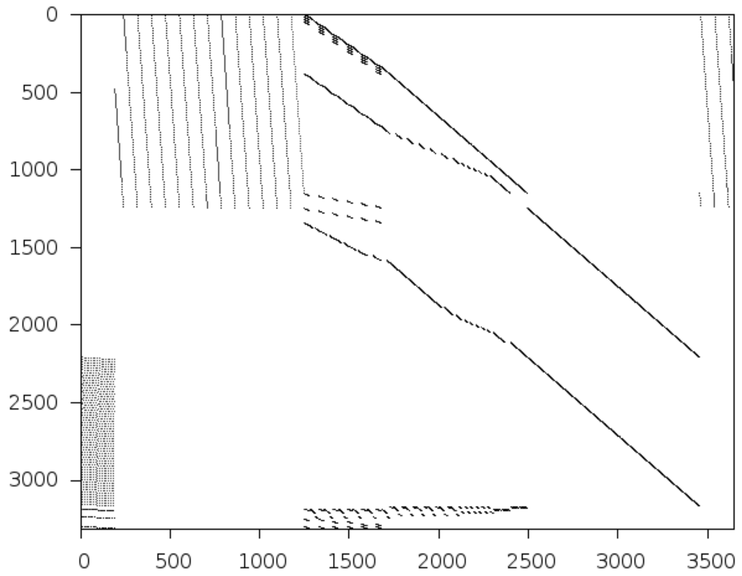
Random face (GAN)

thispersondoesnotexist.com

Random Instances



Random iid coefficients

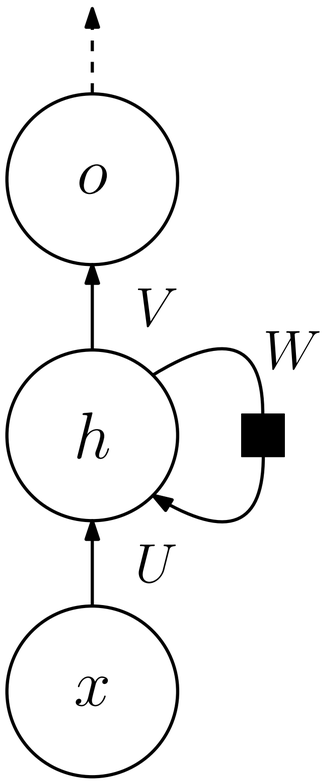


a1c1s1 from MipLib 2017

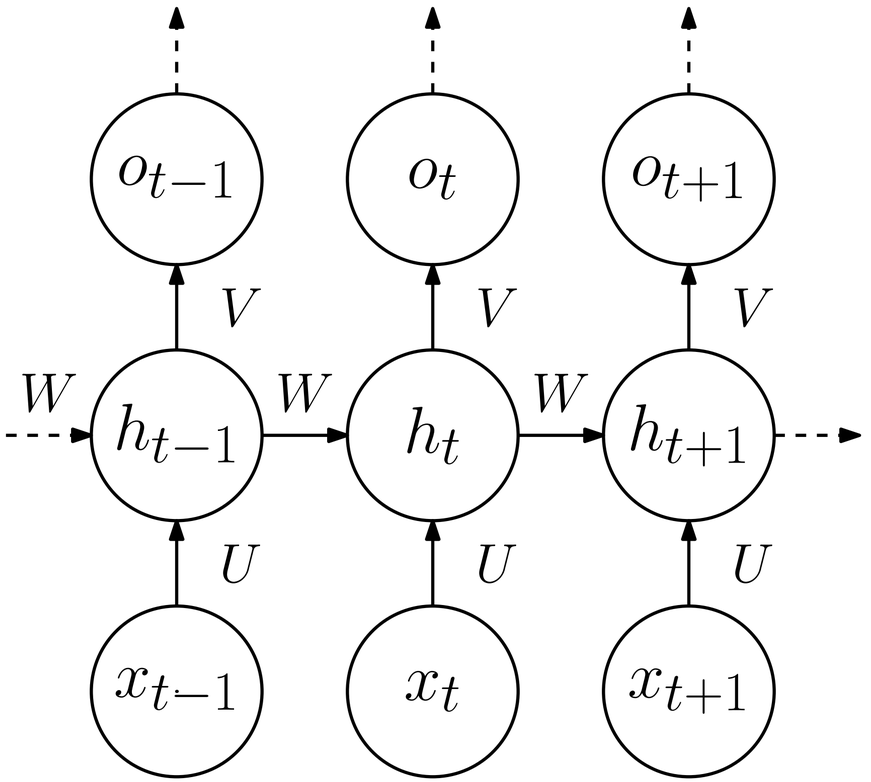
Imitation Learning in a nutshell

- 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 in a nutshell

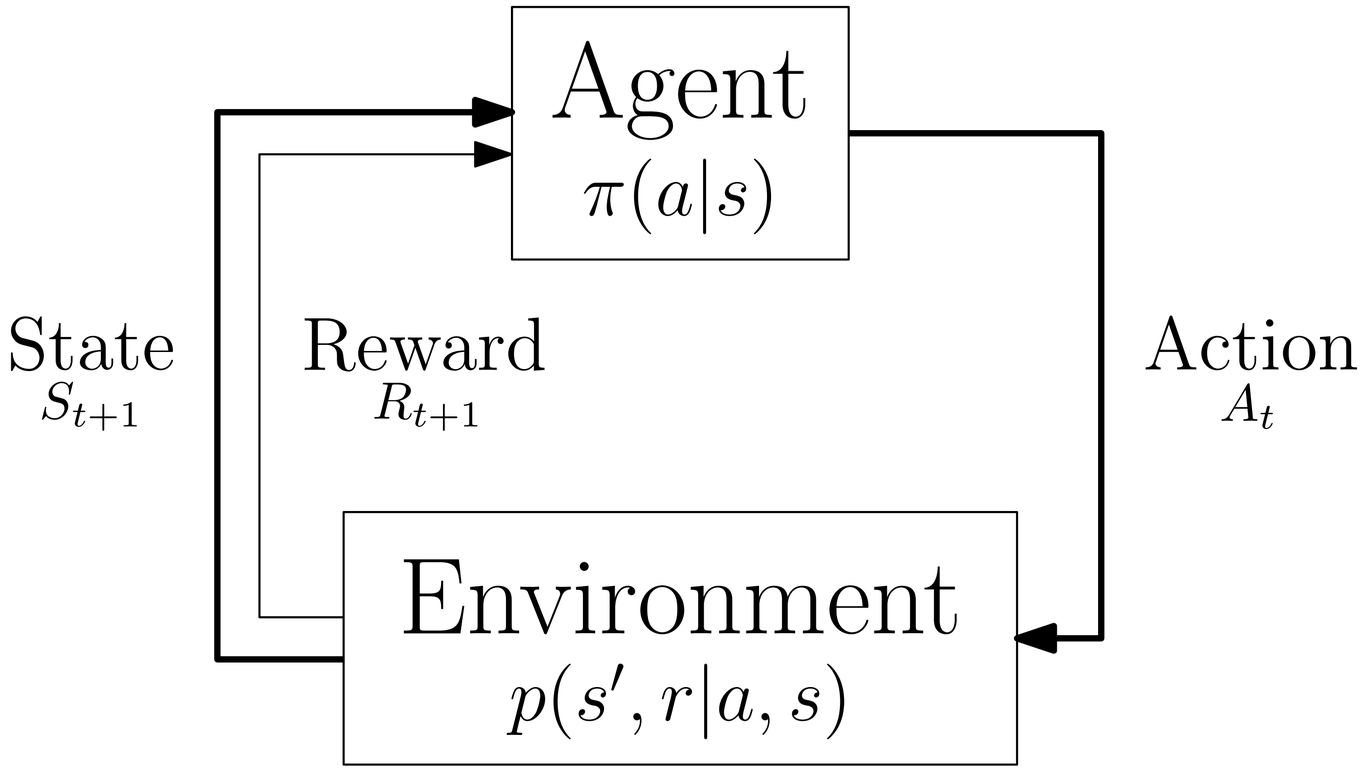


RNN folded



RNN unfolded

Reinforcement Learning in a nutshell



Markov Decision Process for Reinforcement Learning

Learning Methods

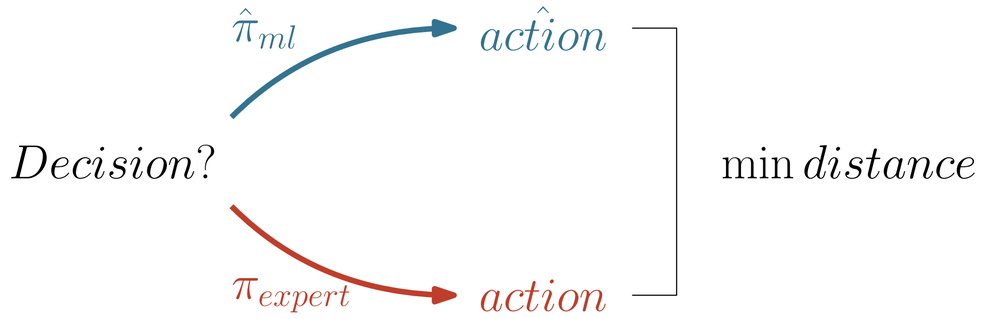
Demonstration

- 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 (hopefully better)
- 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*]

Demonstration vs Experience

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

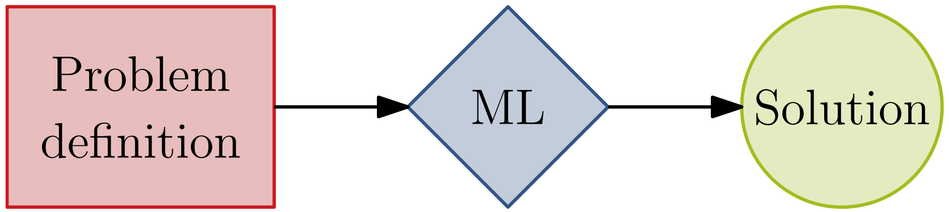
- Reinforcement can potentially discover better policies
- Harder, with local maxima (exploration difficult)
- Need to define a reward

Better **combined**!

Algorithmic Structure

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

End to End Learning

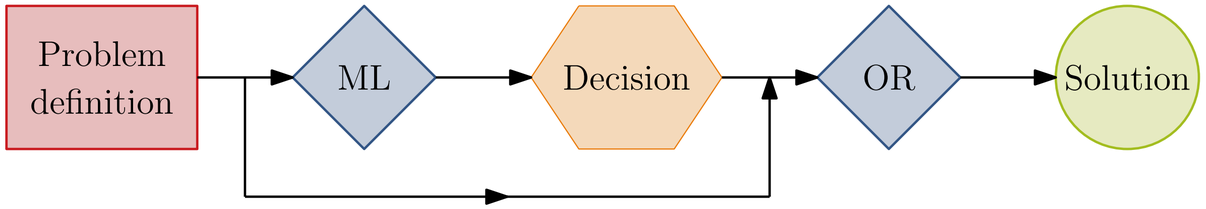


- Learning TSP solutions
  [*Bello et al. 2017*][*Kool and Welling 2018*][*Emami and Ranka 2018*]
  [*Vinyals et al. 2015*][*Nowak et al. 2017*]

- 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



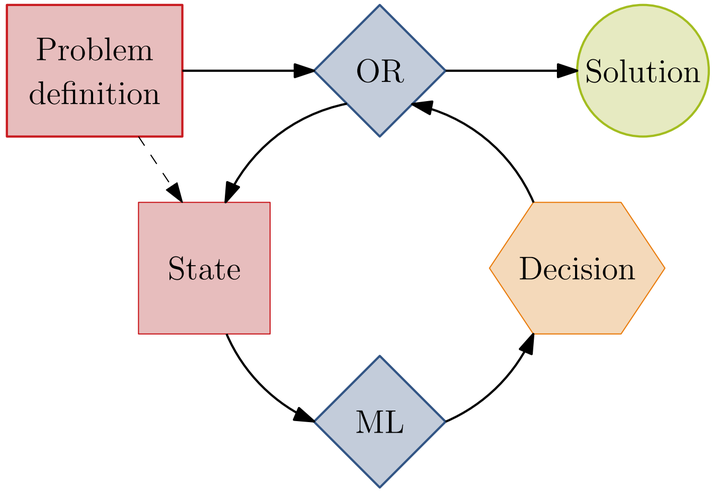
- Use a decomposition method
  [*Kruber et al. 2017*]

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

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

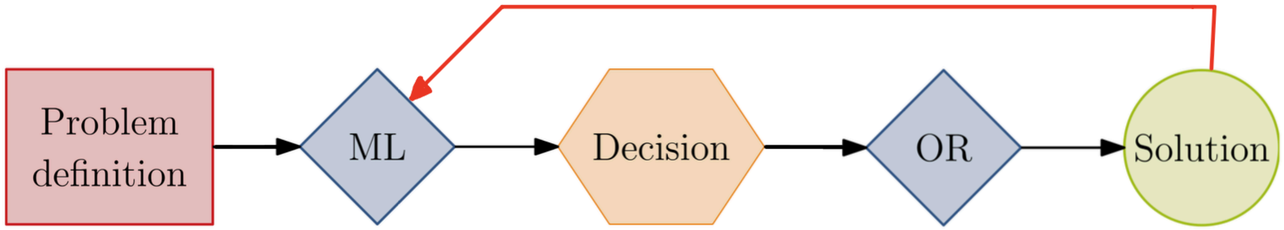
Learning Repeated Decisions

- 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*]
- Predicting booking decisions under imperfect information
  [*Larsen et al. 2018*]
- Learning cutting plane selection
  [*Baltean-Lugojan et al. 2018*]



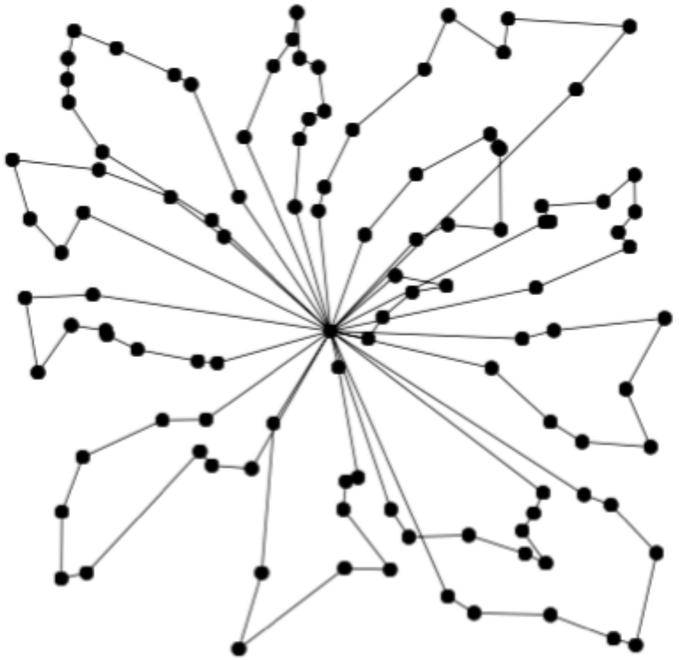
just a matter of viewpoint

Decision-aware learning



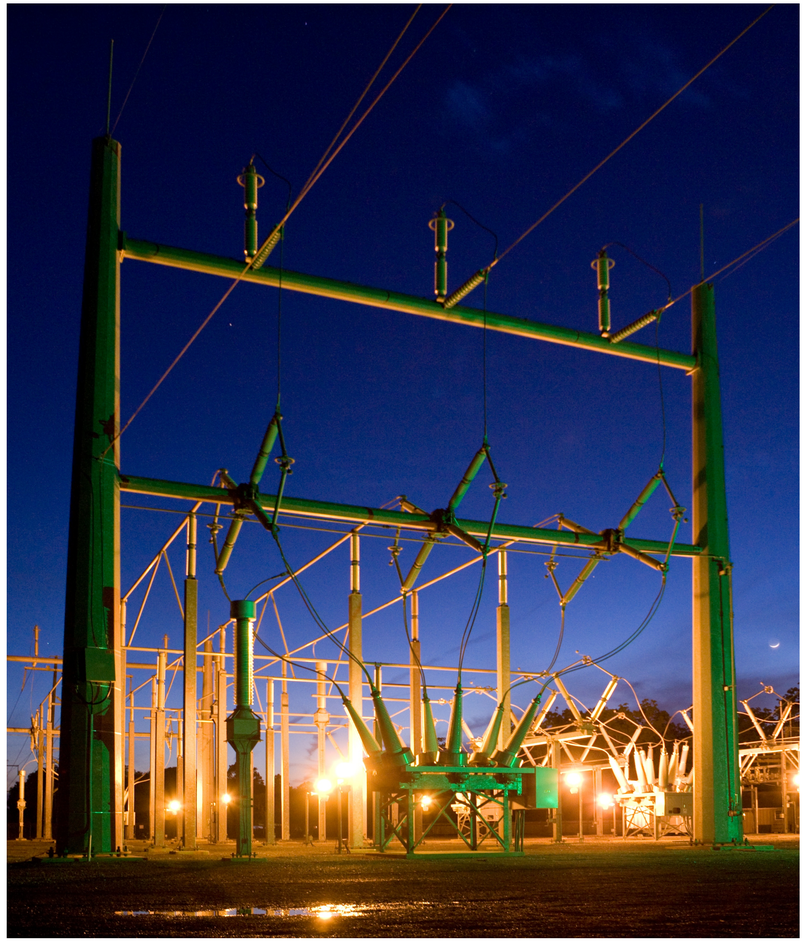
- Quadratic Programming
  [Amos and Kolter 2017]
- Novel loss functions development
  [Elmachtoub and Grigas 2022] [Mandi et al. 2022]
- Perturbation or penalization methods
  [Mandi and Guns 2020] [Blondel et al. 2020] [Wilder et al. 2019]

Part 2: a logistics-flavored business perspective



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



Learning to repeatedly solve routing problems

M. Morabit, G. Desaulniers, A. Lodi, Learning to repeatedly solve routing problems, arXiv:2212.08101, 2022.

Slides based on the PhD defense of M. Morabit

Motivation

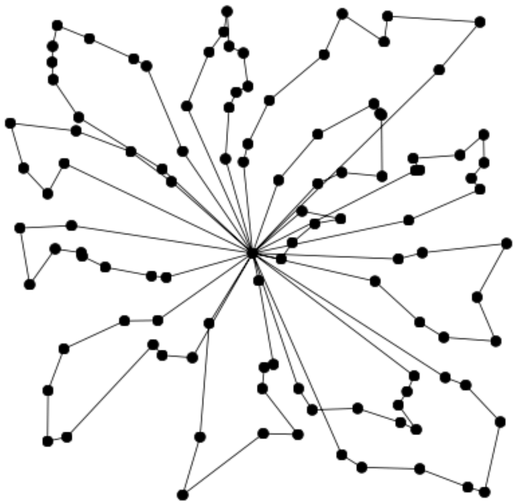
Goal

Accelerate the reoptimization of repeatedly solved problems after a slight change in the problem data

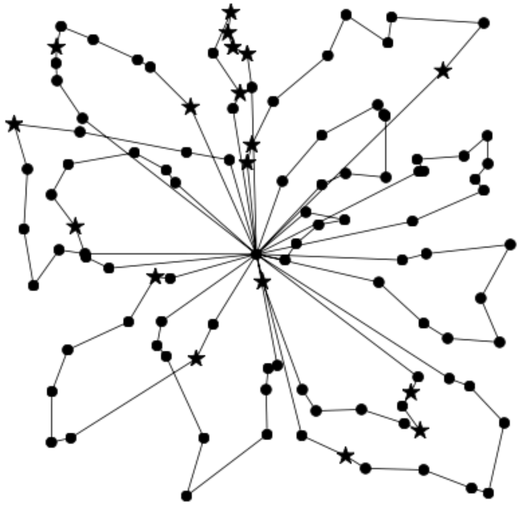
- A focus on the Capacited Vehicle Routing Problem (CVRP) Costa et al. 2019, Pessoa et al. 2020
- Locations of the clients are the same
- Slightly different demands

Objective

- $\mathcal{P}_o$: a problem instance and its solution $\mathcal{S}_o$
- $\mathcal{P}_m$: obtained from $\mathcal{P}_o$ after some changes of the client demands

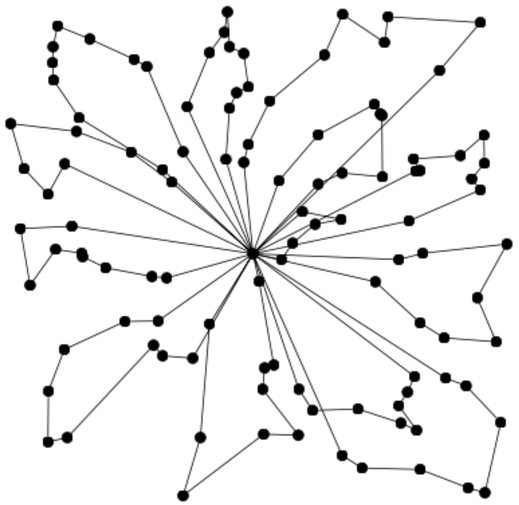


Solution of $\mathcal{P}_o$

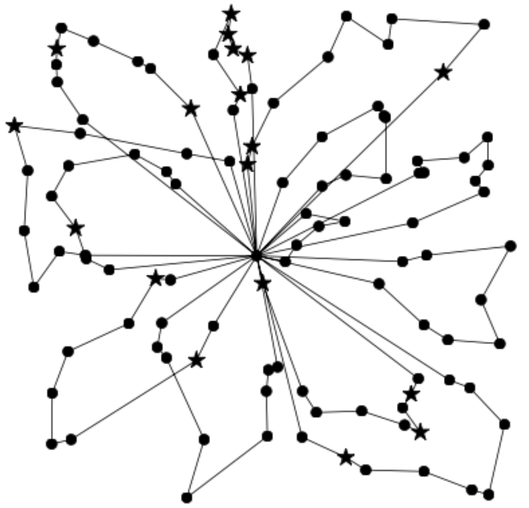


Solution of $\mathcal{P}_m$

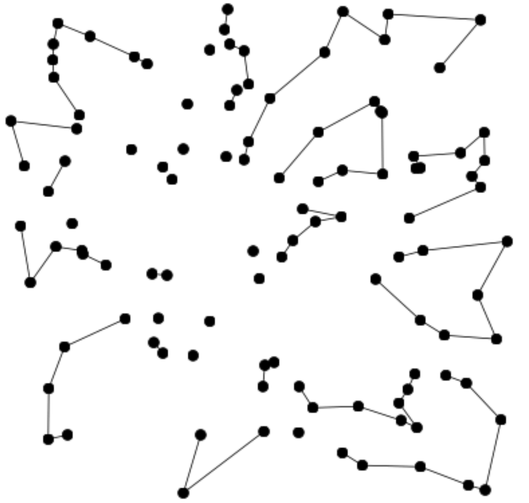
Objective



Solution of $\mathcal{P}_o$



Solution of $\mathcal{P}_m$



Edges to predict

Objective

Instead of solving $\mathcal{P}_m$ from scratch, identify and fix the parts of $\mathcal{S}_o$ that have a high probability of remaining unchanged

Data collection

- Let $E(\mathcal{S})$ denote the set of edges of a solution $\mathcal{S}$
- Given a tuple of original and modified instances with their solutions $(\mathcal{P}_o, \mathcal{S}_o, \mathcal{P}_m, \mathcal{S}_m)$
  A labeled dataset $\mathcal{D} = \{(\boldsymbol{x_e}, y_e) | \forall e \in E(\mathcal{S}_o)\}$ is built

Data collection - Labels

Labels y_e

The labels are assigned by checking the overlapping edges between $\mathcal{S}_o$ and $\mathcal{S}_m$

$$y_e = \begin{cases} 1 & \text{if } e \in E(\mathcal{S}_o) \cap E(\mathcal{S}_m) \\ 0 & \text{otherwise} \end{cases}, \qquad e \in E(\mathcal{S}_o). \tag{6}$$

Data collection - Features

Features x_e

For each edge $e = \langle i, j \rangle$ in $\mathcal{S}_o$

- (x, y) coordinates of nodes i and j
- c_e edge cost
- old and new demands of nodes i and j
- distances between the depot and nodes i and j, respectively
- binary value indicating if the edge is a depot edge
- binary value indicating whether the client i or j, or both, have a changed demand
- The rank of i with respect to j (and vice versa) according to the neighbor distances

ML-based edge fixing

Given $(\mathcal{P}_o, \mathcal{S}_o, \mathcal{P}_m)$

- After training, the predictive model is used to obtain predictions
- For each edge in the original solution, the edge variable is set to 1 depending on the predictions

$$x_e = 1, \quad \forall e \in E(\mathcal{S}_o) \mid \hat{y}_e = 1. \tag{7}$$

Infeasibility case

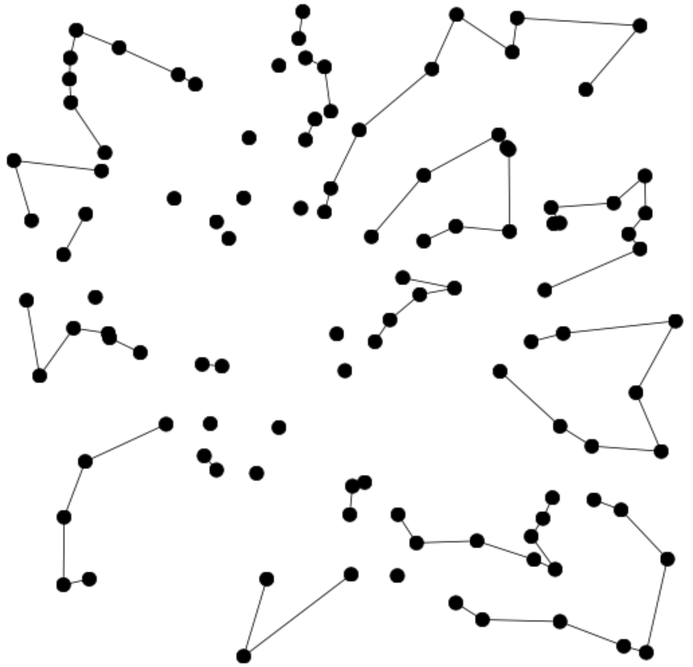
- Fixed sequences of edges exceeding vehicle capacity

- Let $\hat{p}_e$ be the probability estimate of edge e
- For each fixed sequence exceeding the capacity
  $\Rightarrow$ Unfix the edge with the lowest $\hat{p}_e$

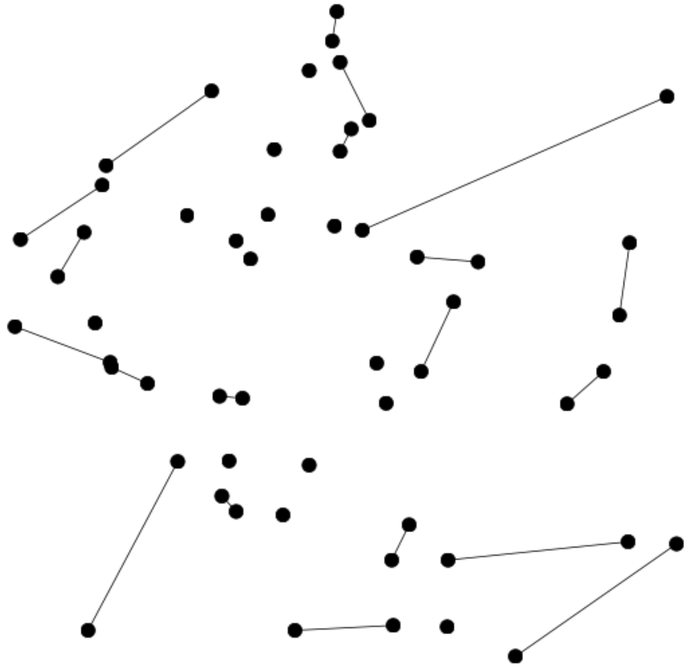
Infeasibility check

1: **for** `each sequence` $s = (v_i, v_2, \ldots, v_{|s|})$ **do**
2: **if** $\sum_{i=1}^{|s|} d_{v_i} > Q$ **then**
3: $e = argmin_{e \in E(p)} \hat{p}_e$
4: $\hat{y}_e = 0$
5: **end if**
6: **end for**

Network contraction



Edges to fix



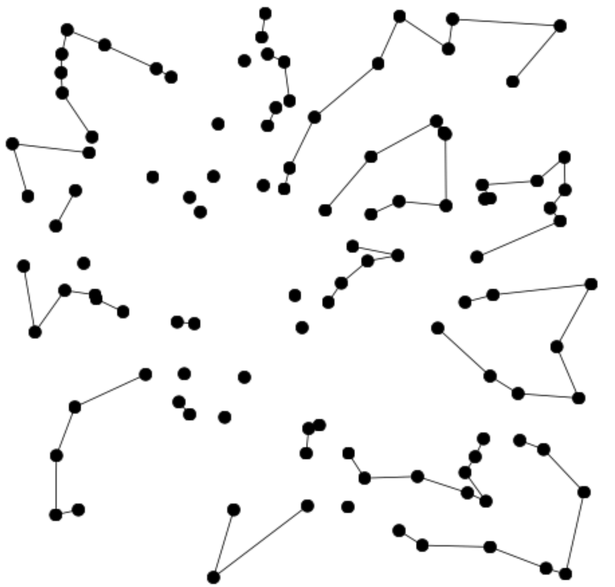
Edges after contraction

$110 \rightarrow 54$ nodes

Network contraction

For each sequence $s = (v_i, v_2, \ldots, v_{|s|})$ with $|s| \geq 3$

- Remove all intermediate nodes $v_2, \ldots, v_{|s|-1}$
- Add the edge $e_{new} = \langle v_1, v_{|s|} \rangle$
- Update the cost $c_{e_{new}} = \sum_{e \in E(s)} c_e$
- Update the demands by adding $\sum_{i=2}^{|s|-1} d_{v_i}$
- Fix the new edge $x_{e_{new}} = 1$



CVRP instances

- X-benchmark instances from the CVRPLIB (Uchoa et al. 2017)
- The instances have different characteristics

Instance name	Depot position	Client positioning	Demand distribution
X-n101-k25	R	RC	$[1 - 100]$
X-n106-k14	E	C	$[50 - 100]$
X-n110-k13	C	R	$[5 - 10]$
X-n125-k30	R	C	*Quadrant*
X-n129-k18	E	RC	$[1 - 10]$
X-n134-k13	R	C	*Quadrant*
X-n139-k10	C	R	$[5 - 10]$
X-n143-k07	E	R	$[1 - 100]$

CVRP instances - demand modifications

For each instance

- $N_c\%$ of the client demands are changed, $N_c = \{10, 20, 30\}$
- The new demand of each client i is chosen randomly in the interval $[d_i - \Delta_d, d_i + \Delta_d]$

Demand	Δ_d **(interval size)**		
distribution	**S**	**M**	**L**
$[1 - 100]$	5	10	15
$[50 - 100]$	5	10	15
$[5 - 10]$	1	2	3
Quadrant	5	10	15
$[1 - 10]$	2	3	4

$\Rightarrow$ 9 different scenarios (10% Small, 10% Medium, etc.)

Data generation

For each original instance and for each scenario

- 100 modified instances are generated
  - 95 for the ML phase
  - 5 remaining for the optimization phase
- Each instance is solved using FILO (Fast Iterated Localized Search Optimization - Accorsi and Vigo 2021)
- One ML-model is trained for each instance and scenario
  $\Rightarrow$ (Artificial) Neural Network models

Two questions

The computational investigation aims at answering two questions:

1. Can we effectively learn the structure of the solution of the "master", and use that structure to devise faster algorithms?
   - Of course, since we are learning probabilistically, the algorithm will be heuristic. But will it be a good one?

2. Is there value in trying to learn the difference between solutions of slightly modified instances?
   - Maybe it would be better to just learn the solution itself, especially considering that the instances are very similar, thus we are really within distribution!
   - We compare with *Dual-Aspect Collaborative Transformer algorithm* (DACT - Ma et al. (2021))

Computational results - $N_c = 10\%$

Interval	$\mathcal{P}_o$	$\mathcal{S}_m$ cost	$sim(\mathcal{S}_o, \mathcal{S}_m)$	ML model metrics			Edge-fixing			Exact B&P		DACT	
				TNR	TPR	Accuracy	Cost	Time (s)	Gap	Cost	Time (s)	Cost	Gap
Small	X-n101-k25	27637	84%	71%	70%	70%	27718	13	0.29%	27635	211	28184	1.98%
	X-n106-k14	26376	85%	89%	60%	75%	26436	25	0.23%	26376	962	26897	1.98%
	X-n110-k13	14987	93%	100%	66%	83%	14987	8	0.00%	14987	265	15159	1.15%
	X-n125-k30	55613	63%	76%	73%	75%	55683	235	0.13%	-	-	58581	5.34%
	X-n129-k18	28765	62%	78%	72%	75%	28982	78	0.75%	28765	3479	29697	3.24%
	X-n134-k13	10888	80%	89%	54%	72%	10917	77	0.27%	-	-	11284	3.64%
	X-n139-k10	13590	85%	92%	81%	86%	13599	26	0.06%	-	-	13846	1.88%
	X-n143-k07	15722	81%	87%	88%	88%	15726	54	0.02%	-	-	16245	3.32%
Average		**23886**	**79%**	**85%**	**71%**	**78%**	**24256**	**65**	**0.22%**	-	-	**24987**	**2.82%**
Medium	X-n101-k25	27606	83%	77%	63%	70%	27736	18	0.47%	27606	324	28298	2.51%
	X-n106-k14	26358	66%	95%	73%	84%	26380	14	0.08%	26358	355	26871	1.95%
	X-n110-k13	14971	87%	98%	67%	82%	14993	12	0.15%	14969	283	15137	1.11%
	X-n125-k30	55713	60%	74%	73%	74%	55758	198	0.08%	55655	5156	58735	5.42%
	X-n129-k18	28862	60%	74%	75%	75%	29124	105	0.91%	-	-	29801	3.26%
	X-n134-k13	10888	63%	74%	73%	74%	11024	320	1.25%	-	-	11304	3.82%
	X-n139-k10	13601	90%	85%	76%	81%	13608	27	0.05%	-	-	13863	1.92%
	X-n143-k07	15707	85%	97%	79%	88%	15710	63	0.02%	-	-	16200	3.14%
Average		**24213**	**74%**	**84%**	**72%**	**78%**	**24292**	**95**	**0.38%**	-	-	**25026**	**2.89%**
Large	X-n101-k25	27651	70%	63%	77%	70%	28125	122	1.71%	27648	399	28142	1.98%
	X-n106-k14	26412	65%	84%	74%	79%	26557	250	0.54%	-	-	26846	1.64%
	X-n110-k13	15030	75%	86%	73%	80%	15107	35	0.51%	15030	2283	15187	1.04%
	X-n125-k30	55733	55%	81%	75%	78%	55825	422	0.16%	-	-	58500	4.97%
	X-n129-k18	28755	62%	78%	73%	75%	28972	171	0.76%	28748	2457	29546	2.75%
	X-n134-k13	10908	69%	72%	68%	70%	10908	164	0.66%	-	-	11267	3.29%
	X-n139-k10	13600	83%	85%	75%	80%	13635	47	0.26%	-	-	13850	1.84%
	X-n143-k07	15716	88%	97%	73%	85%	15717	62	0.00%	-	-	16265	3.49%
Average		**24226**	**71%**	**81%**	**73%**	**77%**	**24365**	**159**	**0.58%**	-	-	**24950**	**2.62%**

Computational results - $N_c = 20\%$

Interval	$\mathcal{P}_o$	$\mathcal{S}_m$ cost	$sim(\mathcal{S}_o, \mathcal{S}_m)$	ML model metrics			Edge-fixing			Exact B&P		DACT	
				TNR	TPR	Accuracy	Cost	Time (s)	Gap	Cost	Time (s)	Cost	Gap
Small	X-n101-k25	27486	83%	73%	75%	74%	27628	29	0.51%	27486	231	28220	2.67%
	X-n106-k14	26338	72%	90%	71%	81%	26401	30	0.24%	26336	1147	26845	1.93%
	X-n110-k13	14980	77%	86%	74%	80%	15024	9	0.29%	14980	429	15152	1.15%
	X-n125-k30	55493	63%	85%	68%	77%	55509	233	0.03%	-	-	58372	5.19%
	X-n129-k18	29020	56%	71%	78%	74%	29408	143	1.33%	29009	5791	29703	2.35%
	X-n134-k13	10909	73%	85%	69%	77%	10943	150	0.32%	-	-	11323	3.80%
	X-n139-k10	13613	87%	91%	81%	86%	13616	103	0.02%	-	-	13870	1.89%
	X-n143-k07	15715	86%	90%	86%	88%	15748	30	0.21%	-	-	16261	3.47%
Average		**24194**	**75%**	**84%**	**75%**	**80%**	**24284**	**91**	**0.37%**	-	-	**24968**	**2.81%**
Medium	X-n101-k25	27512	68%	74%	72%	73%	27694	32	0.66%	27512	518	28076	2.05%
	X-n106-k14	26306	64%	87%	71%	79%	26400	20	0.36%	26306	5257	26785	1.72%
	X-n110-k13	14983	73%	86%	77%	82%	15078	27	0.64%	14983	548	15217	1.56%
	X-n125-k30	55503	54%	82%	74%	78%	55579	233	0.14%	-	-	58528	5.45%
	X-n129-k18	29171	57%	73%	76%	74%	29655	315	1.66%	-	-	30001	2.85%
	X-n134-k13	10877	55%	82%	80%	81%	10956	127	0.73%	-	-	11258	3.50%
	X-n139-k10	13605	72%	87%	80%	84%	13631	205	0.20%	-	-	13855	1.84%
	X-n143-k07	15708	81%	80%	83%	82%	15745	49	0.24%	-	-	16233	3.35%
Average		**24208**	**65%**	**81%**	**77%**	**79%**	**24342**	**126**	**0.58%**	-	-	**24991**	**2.79%**
Large	X-n101-k25	27645	59%	70%	73%	72%	27871	56	0.81%	27645	466	28244	2.17%
	X-n106-k14	26413	61%	79%	74%	77%	26555	100	0.54%	-	-	26834	1.59%
	X-n110-k13	15034	68%	81%	79%	80%	15161	23	0.84%	15034	487	15216	1.21%
	X-n125-k30	55958	58%	79%	70%	74%	56168	433	0.37%	-	-	58629	4.77%
	X-n129-k18	29123	55%	77%	74%	75%	29462	386	1.16%	29092	3106	29848	2.49%
	X-n134-k13	10909	55%	82%	80%	81%	11051	254	1.31%	-	-	11317	3.75%
	X-n139-k10	13585	73%	88%	77%	83%	13619	139	0.25%	-	-	13788	1.50%
	X-n143-k07	15743	84%	79%	79%	79%	15775	41	0.20%	-	-	16332	3.74%
Average		**24301**	**64%**	**79%**	**76%**	**78%**	**24458**	**179**	**0.69%**	-	-	**25026**	**2.65%**

Computational results - $N_c = 30\%$

Interval	$\mathcal{P}_o$	$\mathcal{S}_m$ cost	$sim(\mathcal{S}_o, \mathcal{S}_m)$	ML model metrics			Edge-fixing			Exact B&P		DACT	
				TNR	TPR	Accuracy	Cost	Time (s)	Gap	Cost	Time (s)	Cost	Gap
Small	X-n101-k25	27562	75%	80%	70%	75%	27669	37	0.39%	27562	178	28219	2.38%
	X-n106-k14	26383	65%	89%	69%	79%	26438	148	0.21%	26378	1577	26833	1.70%
	X-n110-k13	15005	74%	92%	77%	84%	15083	32	0.52%	15005	555	15157	1.01%
	X-n125-k30	55776	53%	77%	73%	75%	55834	259	0.10%	-	-	58601	5.06%
	X-n129-k18	29414	58%	80%	74%	77%	29778	237	1.24%	29405	3742	30086	2.28%
	X-n134-k13	10925	77%	90%	62%	76%	10952	140	0.25%	-	-	11361	4.00%
	X-n139-k10	13622	83%	91%	73%	82%	13692	154	0.51%	-	-	13829	1.52%
	X-n143-k07	15754	83%	79%	87%	83%	15822	56	0.43%	-	-	16328	3.65%
Average		**24305**	**71%**	**85%**	**73%**	**79%**	**24409**	**133**	**0.46%**	-	-	**25052**	**2.70%**
Medium	X-n101-k25	27654	63%	68%	73%	71%	27804	51	0.54%	27651	243	28277	2.25%
	X-n106-k14	26437	59%	75%	70%	73%	26615	247	0.67%	-	-	26780	1.30%
	X-n110-k13	15058	77%	87%	72%	79%	15095	14	0.24%	15058	269	15292	1.55%
	X-n125-k30	55925	50%	82%	78%	80%	56009	451	0.15%	-	-	58859	5.25%
	X-n129-k18	29221	54%	79%	75%	77%	29698	137	1.63%	29215	3574	29849	2.16%
	X-n134-k13	10926	56%	83%	80%	81%	11048	129	1.12%	-	-	11301	3.43%
	X-n139-k10	13640	75%	89%	78%	83%	13708	256	0.49%	-	-	13926	2.09%
	X-n143-k07	15782	80%	78%	85%	82%	15870	49	0.55%	-	-	16423	4.06%
Average		**24330**	**64%**	**80%**	**76%**	**78%**	**24481**	**167**	**0.67%**	-	-	**25088**	**2.76%**
Large	X-n101-k25	27617	66%	76%	72%	74%	27929	185	1.12%	27617	339	28207	2.13%
	X-n106-k14	26440	54%	79%	68%	74%	26637	275	0.75%	-	-	26913	1.79%
	X-n110-k13	15090	66%	82%	72%	77%	15211	88	0.80%	15090	1120	15318	1.51%
	X-n125-k30	56228	52%	84%	71%	78%	56221	475	-0.01%	-	-	58725	4.44%
	X-n129-k18	29674	58%	75%	73%	74%	30179	75	1.69%	29670	4914	30554	2.97%
	X-n134-k13	10950	53%	83%	82%	82%	11029	145	0.72%	-	-	11423	4.31%
	X-n139-k10	13616	73%	91%	76%	84%	13635	128	0.14%	-	-	13868	1.85%
	X-n143-k07	15884	79%	82%	83%	83%	15941	313	0.36%	-	-	16326	2.79%
Average		**24437**	**63%**	**82%**	**75%**	**78%**	**24598**	**211**	**0.70%**	-	-	**25167**	**2.72%**

Summary

- Discrete Optimization is one of the tools of choice for a variety of applications, including those of interest for this workshop.

- The needs for making combinatorial optimization competitive in many applied contexts include:
  - dealing with big (and uncertain) data
  - “learn” to repeatedly solve the “same instance”

- Overall we start to have (solid) evidence that ML / CO integration approaches could be effective to deal with those needs and challenges.

SCIP

Solving Constraint Integer Programs

https://www.scipopt.org/

- One of the fastest non-commercial solvers for MIP
- ~800k lines of code; many advanced features and extensions
-

Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers

Antoine Prouvost
Mila, Polytechnique Montréal

Justin Dumouchelle
Polytechnique Montréal

Lara Scavuzzo
Technische Universiteit Delft

Maxime Gasse
Mila, Polytechnique Montréal

Didier Chételat
Polytechnique Montréal

Andrea Lodi
Mila, Polytechnique Montréal

https://www.ecole.ai/

NeurIPS 2021 competition: ML4CO

Machine Learning for
Combinatorial Optimization
——COMPETITION 2021——

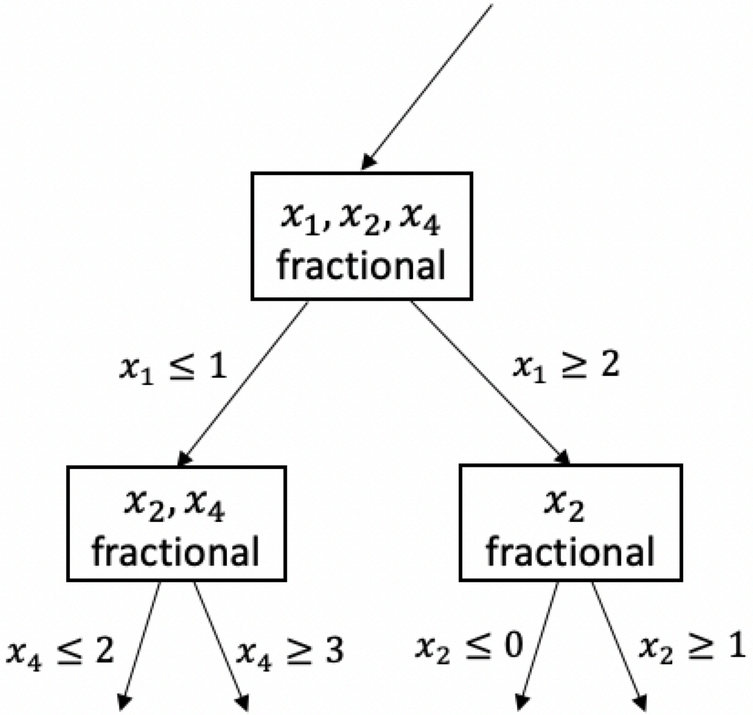
Mixed Integer Linear Programming (MILP)

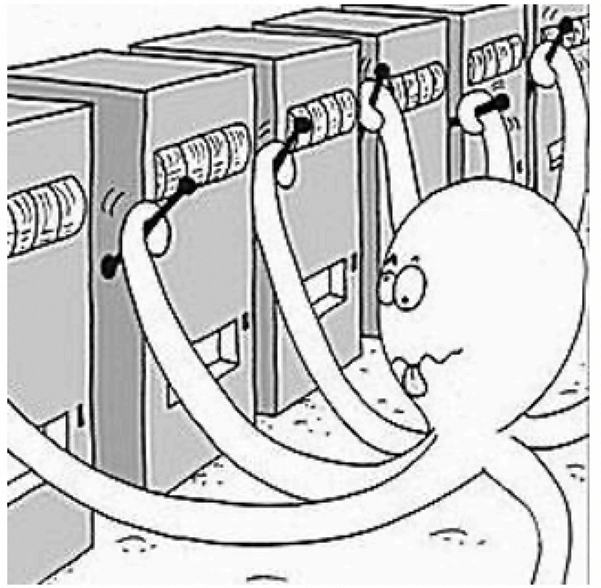
$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}^\top \mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

Machine Learning for Combinatorial Optimization (ML4CO)
NeurIPS 2021 competition (Submission deadline: Oct 31 2021)
https://www.ecole.ai/2021/ml4co-competition/

Three Tasks

Finding feasible solutions of MILP





Primal Task | **Dual Task** | **Configuration Task**