# Learning-based Algorithms for Network Optimization

Nicola Di Cicco

*IIJ Research Laboratory Seminar*
*13/08/2024*

# About me

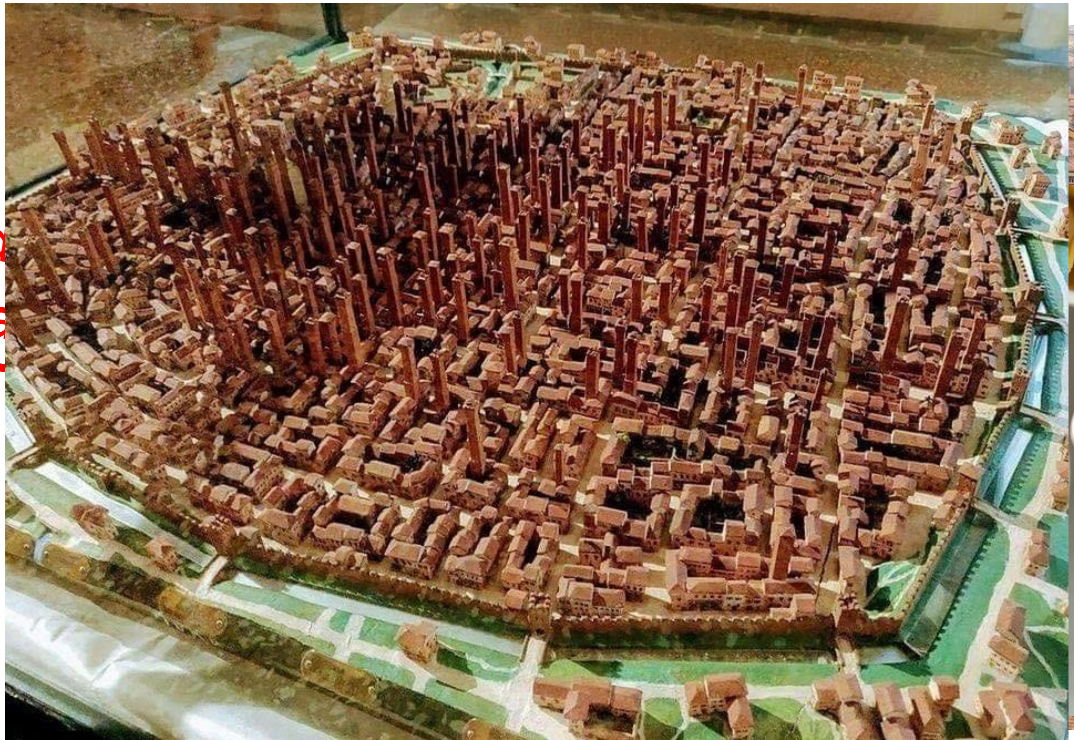
**Jeff set me up for this presentation. Help!**

# About me

I'm a 3rd year PhD student at Politecnico di Milano, Italy


I work here

I was born and live in Bologna

# TL;DR of this talk

- Optimization algorithms solve problem instances individually, **but instances are strongly correlated in practice!**
- We want to **discover and exploit these correlations** to develop **specialized and efficient solving algorithms**
- **Machine Learning** is a great hammer for this problem
- Illustrative numerical results look **pretty good**

# Outline of this talk (if you're still listening)

1) We will practically see why **Machine Learning for optimization makes actual sense** (it is not just drinking the 2024's Kool-aid)

2) We will see **general design paradigms** of ML for optimization, with **practical examples from recent networking papers**

3) We will discuss in greater technical detail **two contributions on the topic from my research group** at Politecnico di Milano, Italy

# “Why care about Machine Learning for Optimization?” *

* excerpt from a review I got for a submission at INFOCOM two years ago (the reviewer assigned me 1/5 and killed my paper)

# Brief rundown on “optimization”

- We want to find the values for **decision variables** that optimize an **objective function** while satisfying a set of **constraints**

Obj. function

Decision variables

$$\min_x \quad f(x)$$
$$\text{s.t.} \quad g_i(x) \in \mathcal{S}_i,\ i = 1, \ldots, n$$

Constraints

- Example: Knapsack problem
  - $x_i$ is 1 if object $i$ is in the knapsack
  - Maximize profit s.t. capacity constraint

$$\max \quad \sum_{i=1}^{n} p_i x_i$$
$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \le C$$
$$x_i \in \{0, 1\},\ i = 1, \ldots, n$$

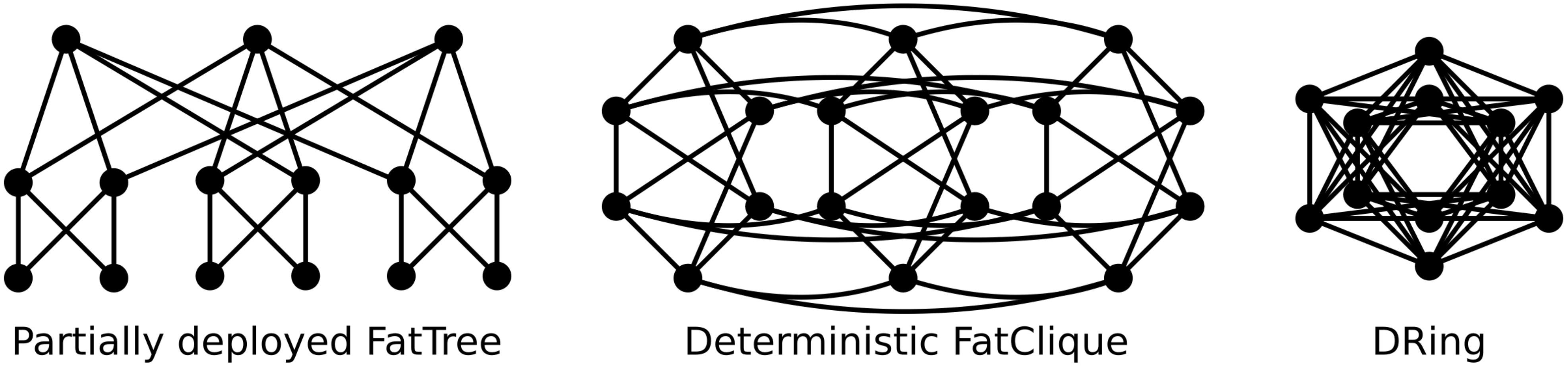
# Should you care about the general case?


- Suppose you're interested in solving a fancy new optimization problem…
- … but you find that **it is, in general, computationally intractable**

- However, if you **focus on instances with specific characteristics**…
- … you might be able to **discover interesting things!**

# Example: Oblivious Routing

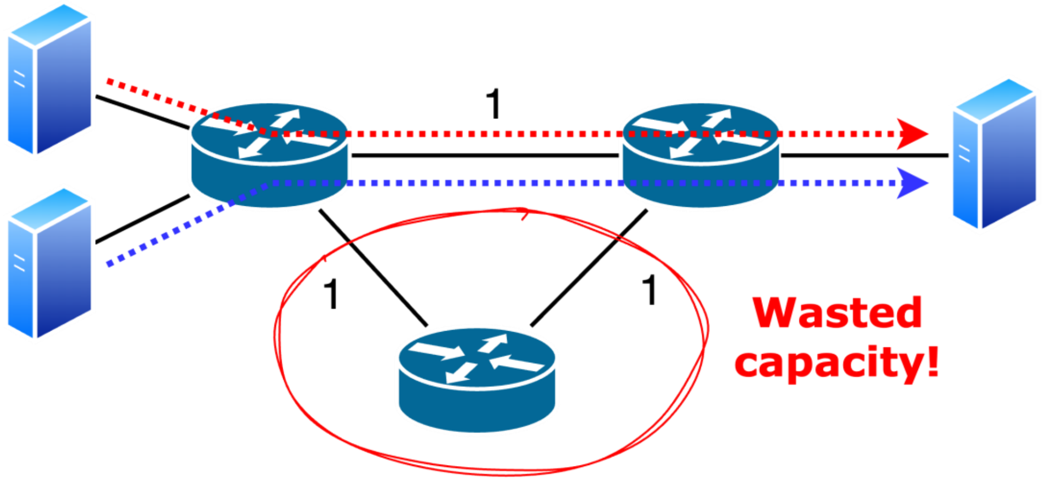
- **Problem**: find the best way to distribute network flows irrespectively from the actual demands (hence, “oblivious”)
- Unfortunately, **NP-Hard** in the general case…
- **… but can be made tractable for certain topologies** [1]

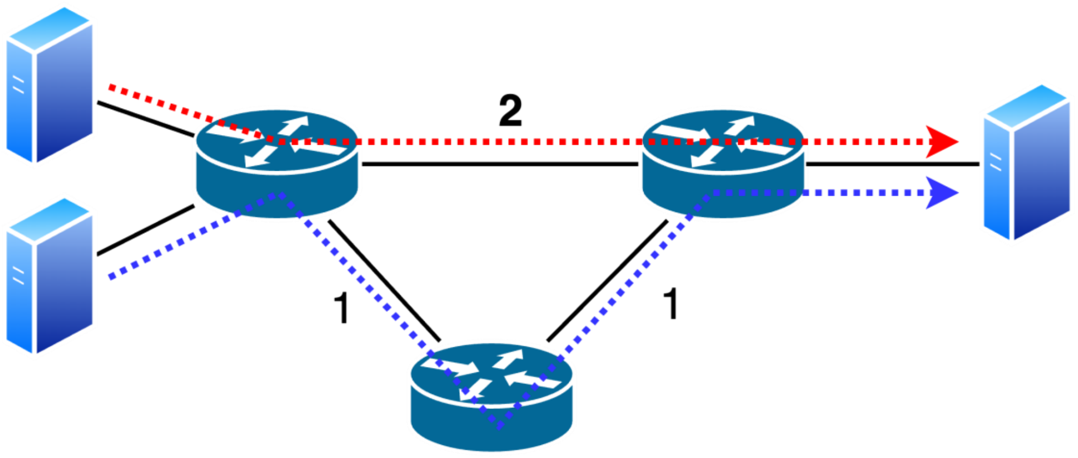



[1] S. Supittayapornpong et al., “Optimal Oblivious Routing for Structured Networks,” in INFOCOM, 2022

# Example: Traffic Engineering with ECMP

- **Problem**: find the best integer edge weights, such that traffic is optimally distributed via equal-cost shortest paths



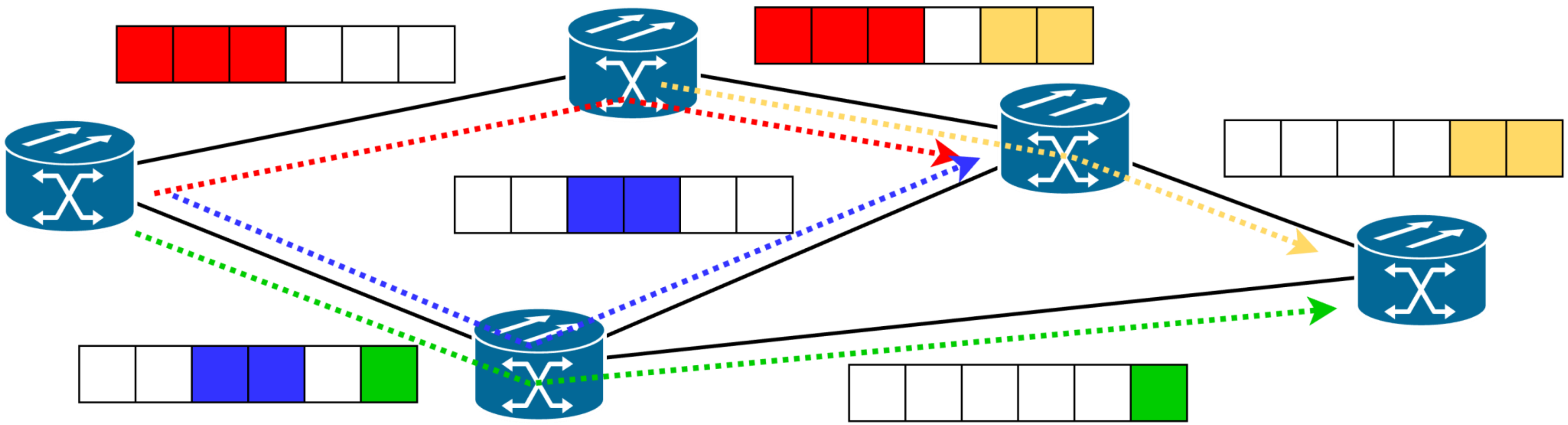




- Does **not** achieve optimal flow distribution in general…
- … but **it provably does for Clos networks** [2]

[2] M. Chiesa et al., “Traffic Engineering with ECMP: an Algorithmic Perspective,” in ToN, 2017

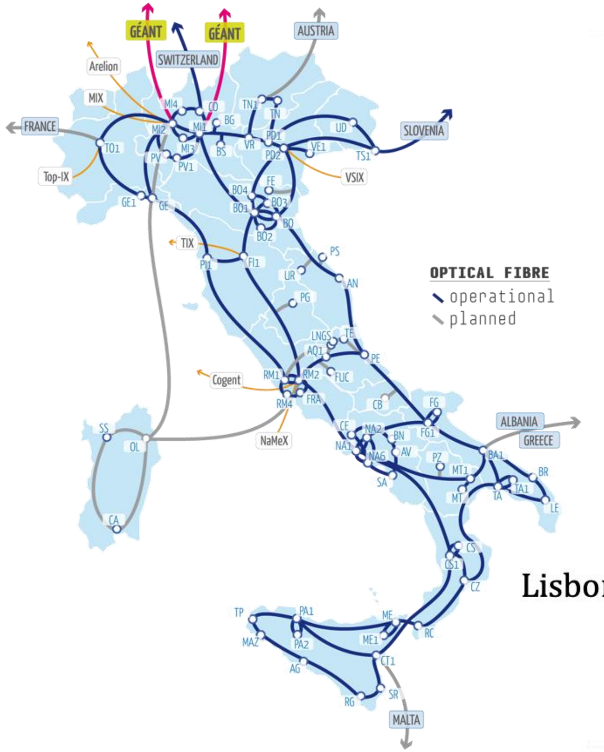
# Example: RSA in Optical Networks

- **Problem**: route network flows through optical wavelengths that must be contiguous over spectrum and continuous over paths

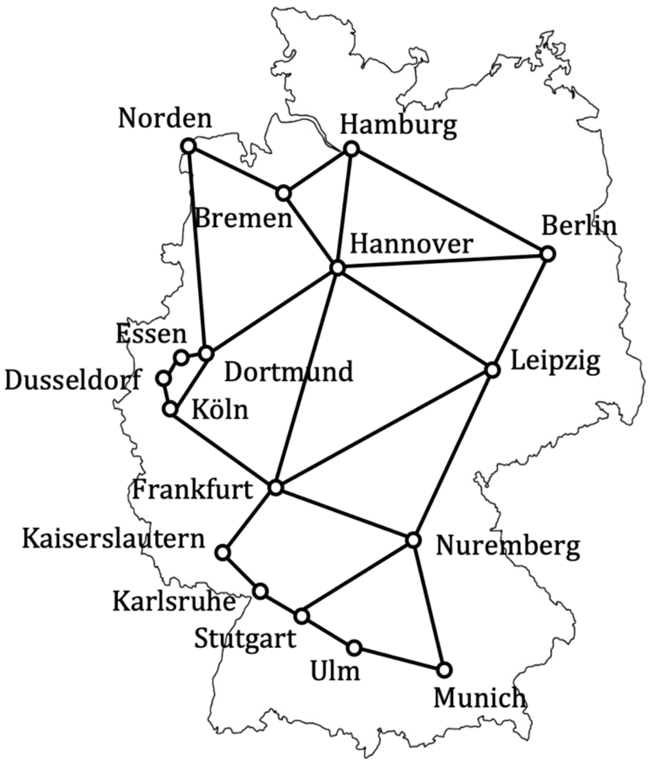


- Unfortunately, **NP-Hard** in the general case…

# Example: RSA in Optical Networks








- ... but optical networks have structure!
- **How do we systematically exploit this?** (Open question!)

# Your turn

**Take 30 seconds to think about another use-case where problem instances are, in practice, strongly correlated!**

# Main takeaway

**Real-world problem instances stem from data distributions!**

# State-of-the-Art in Optimization

- As of 2024, there exist a bunch of very efficient solvers…

- … but these solvers are tailored for the **general case**, and may not work well for the problems we're interested in solving!

# State-of-the-Art in Optimization

- Of course, we can leverage **specialized algorithms**…

**OR-Tools' Vehicle Routing Solver:**
A Generic Constraint-Programming Solver with Heuristic Search for Routing Problems (VRPs)

**Routing One Million Customers in a Handful of Minutes**

Luca Accorsi[1] and Daniele Vigo[2, 3]

**Concorde TSP Solver**

Concorde is a computer code for the symmetric traveling salesman problem (TSP) and some related network optimization problems. The code is written in the ANSI C programming language and it is available for academic research use; for other uses, contact William Cook for licensing options.

- … but designing them requires **extensive domain knowledge** for discovering the special characteristics of a problem!

# The big question

**Can machines autonomously learn specialized algorithms from data?**

# Learning for Optimization

# A Taxonomy of ML for Optimization

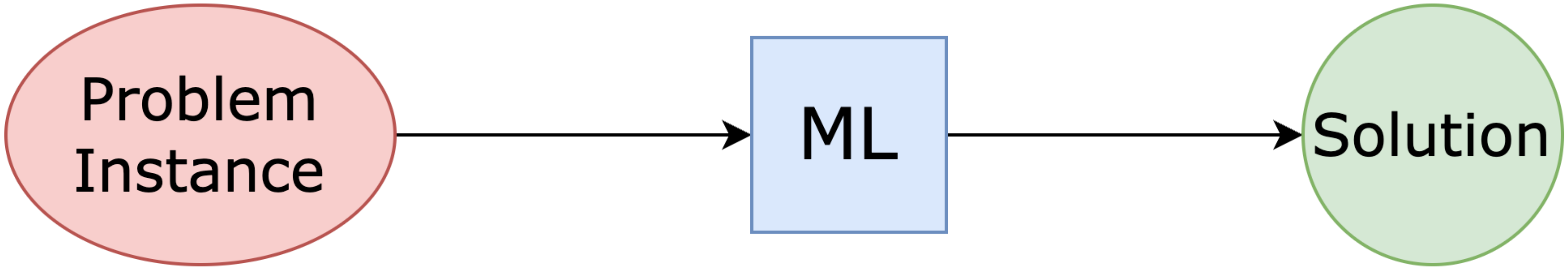
1. **End-to-End Learning**
   - We use ML to directly predict solutions for our problem

2. **Learning Algorithm Configurations**
   - We use ML to provide “information” to an optimization algorithm

3. **Learning Alongside Optimization**
   - We use ML to implement “subroutines” of a larger algorithm

[3] Y. Bengio et al., “ML for Combinatorial Optimization: A Methodological Tour D’Horizon” in EJOR, 2021

# End-to-End Learning




- **We learn statistical relations** between a problem instance and decision variables (<u>no optimization is happening</u>!)
  - e.g., in a Knapsack, high-value low-weight items are likely to be packed

\+ **Fast and scalable**

\- **No performance guarantees**

\- **Difficult to enforce hard constraints**

# Premise: Multi-Commodity Flow

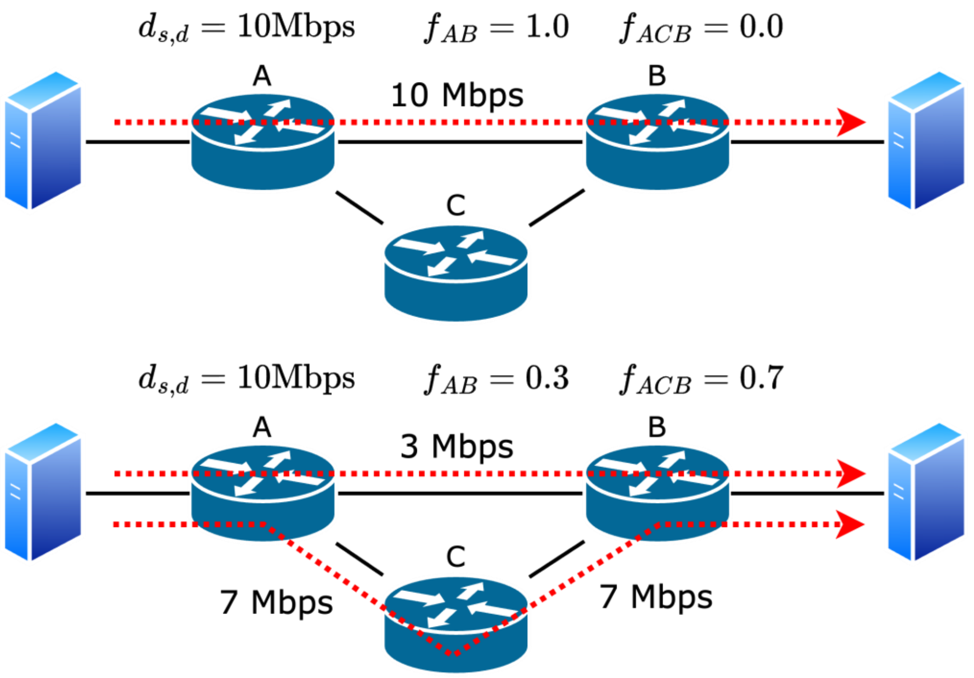
- **Problem**: given a **network**, a set of **source-destination paths** and a **traffic matrix**, decide how to **optimally distribute the traffic**
- A classical objective is **minimizing the maximum link utilization**

Per-path split ratios

Capacity requested by the $k$-th demand

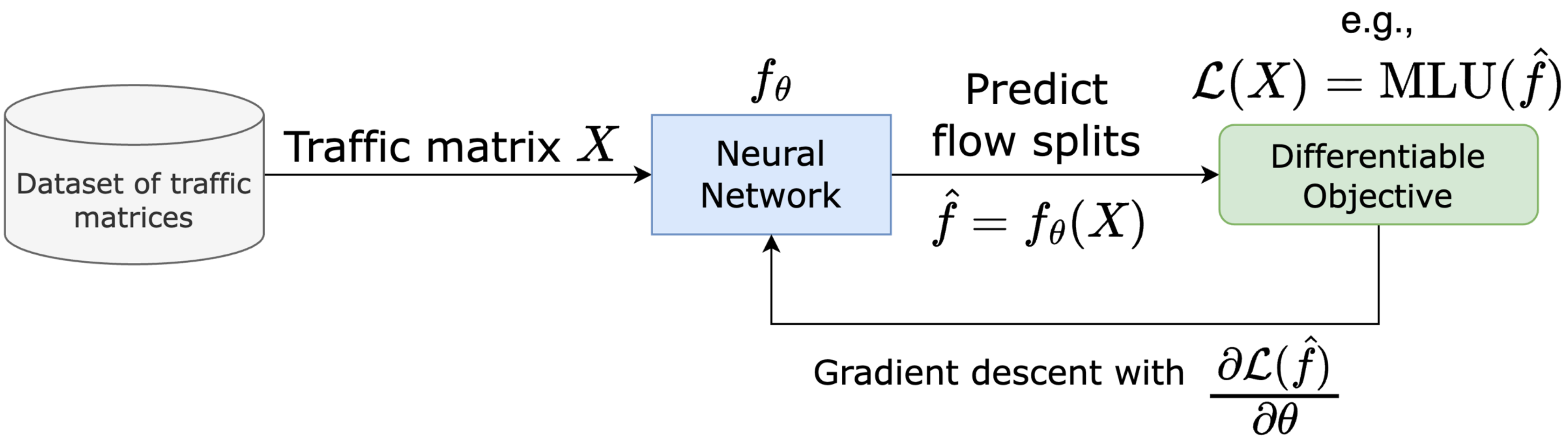
Capacity of link $e$

$$
\begin{aligned}
\min \quad & \alpha \\
\text{s.t.} \quad & \sum_{p \in P_k} f_p = d_k \quad \forall k \in K \\
& \sum_{k \in K} \sum_{p \in P_k : e \in p} f_p \leq \alpha C_e \quad \forall e \in E \\
& f_p \geq 0 \quad \forall k \in K, \forall p \in P_k \\
& \alpha \geq 0
\end{aligned}
$$



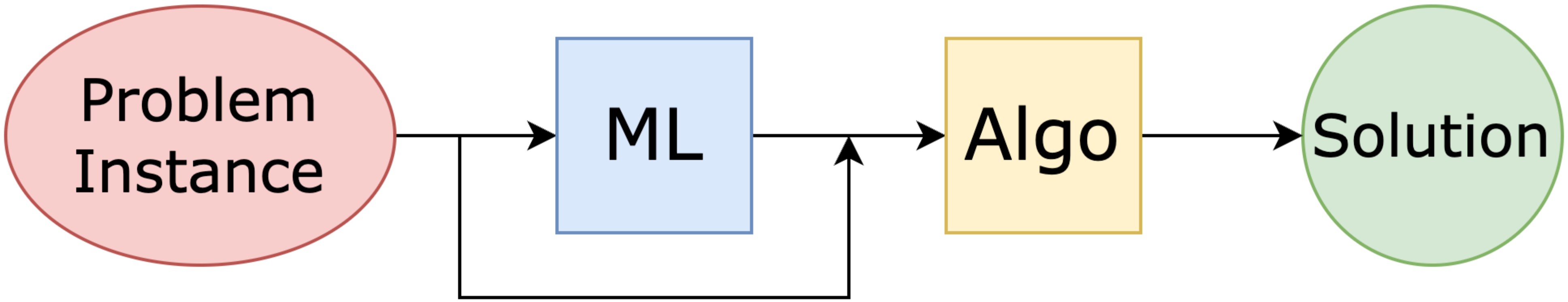
# Example: DOTE [4]

- **Goal**: solve large-scale multi-commodity flow problems
- **Idea**: leverage differentiability of the objective with respect to a solution to train a end-to-end a predictive neural network




[4] Y. Perry et al., "DOTE: Rethinking (Predictive) WAN Traffic Engineering" in NSDI, 2023

# Learning Algorithm Configurations



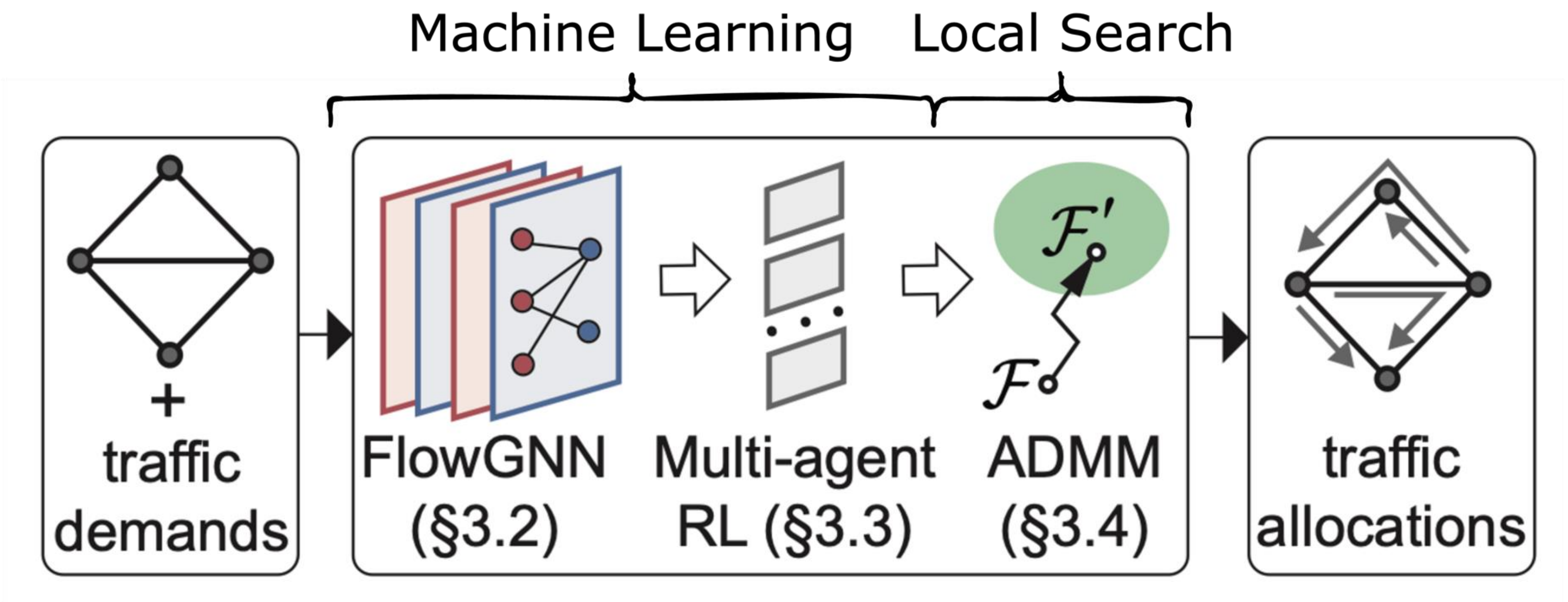

- Given a specific problem instance, **a ML model predicts the "optimal configuration"** of an optimization algorithm.
  - e.g., algo's hyperparameters, warm-starting solution, …

\+ **Can guarantee feasibility (and possibly optimality)**

\- **Harder to train (Supervised Learning might not be possible)**

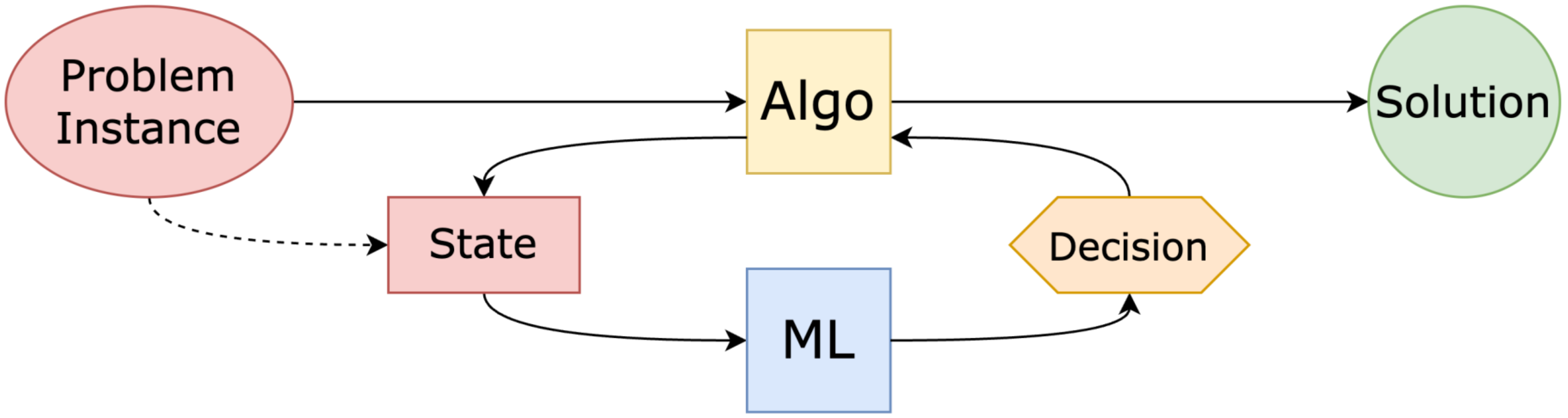
# Example: TEAL [5]

- **Problem**: solve large-scale multi-commodity flow problems
- **Idea**: use ML to predict a tentative solution, then do local search with ADMM to improve the solution and fix broken constraints




[5] Z. Xu et al., "Teal: Learning-Accelerated Optimization of WAN Traffic Engineering" in SIGCOMM, 2023

# Learning Alongside Optimization




- A ML model implements **subroutines** of a "master algorithm"
  - e.g., mutation function in a GA, neighbour selection policy in a LS

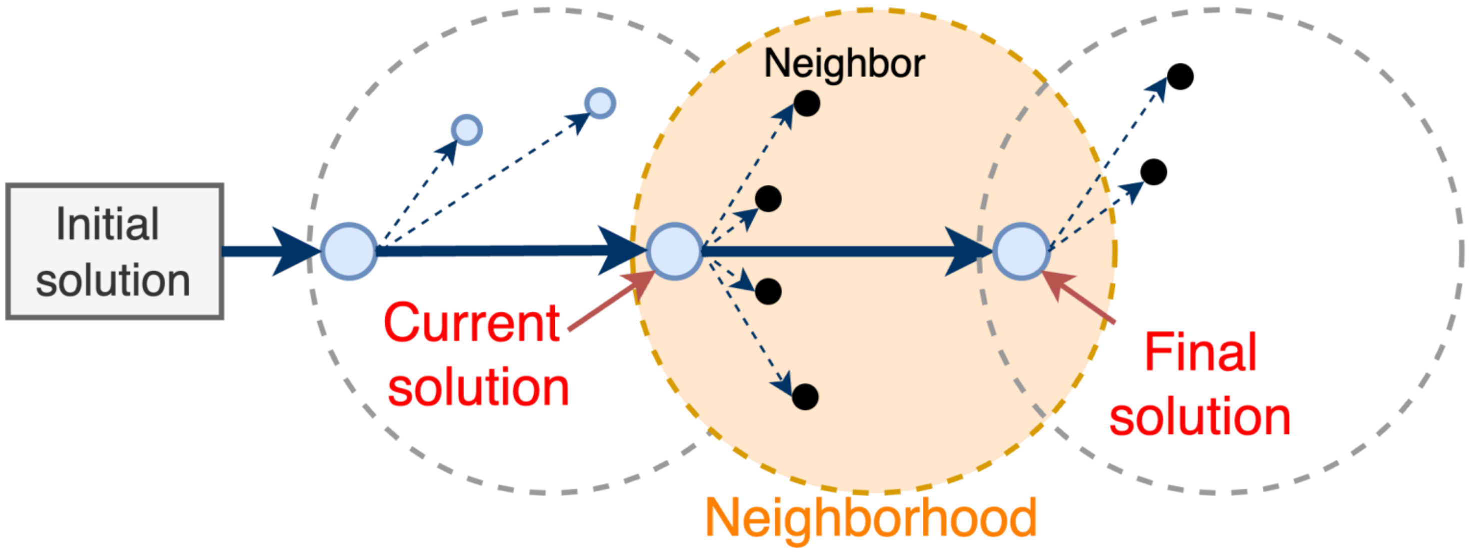
\+ **Very powerful**

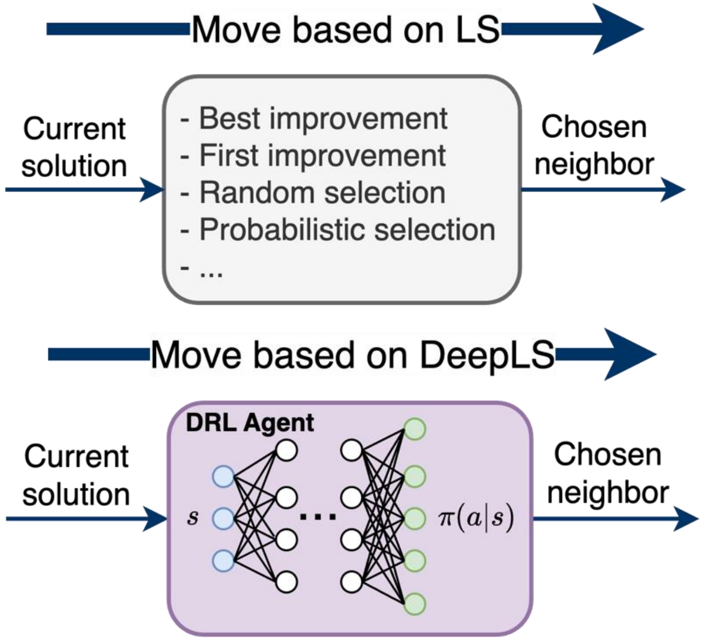
\+ **Can guarantee feasibility (and possibly optimality)**

\- **Harder to train (Supervised Learning might not be possible)**

# Example: DeepLS [6]

- **Problem**: learn problem-tailored neighbourhood selection policies for Local Search algorithms
- **Idea**: train a neural net with Reinforcement Learning to do it




[6] Di Cicco et al., "DeepLS: Local search for network optimization based on lightweight deep reinforcement learning" in TNSM, 2023

# Takeaways

ML for optimization can be roughly **divided in two**:

1. **Machine Learning only**
2. **Machine Learning integrated with classical algorithms**

- Using only Machine Learning allows us to trivially leverage **massive hardware acceleration** (GPUs, etc.), but **enforcing feasibility and performance guarantees is challenging**
- ML + optimization allows us to **build upon tens of years of literature**, at the price of a possibly **more complex training procedure and implementation**

# 1st Case Study:

# Augmenting Local Search with Reinforcement Learning

# Reference paper

108 IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 21, NO. 1, FEBRUARY 2024

## DeepLS: Local Search for Network Optimization Based on Lightweight Deep Reinforcement Learning

Nicola Di Cicco, *Graduate Student Member, IEEE*, Memedhe Ibrahimi, *Member, IEEE*, Sebastian Troia, *Member, IEEE*, and Massimo Tornatore, *Fellow, IEEE*

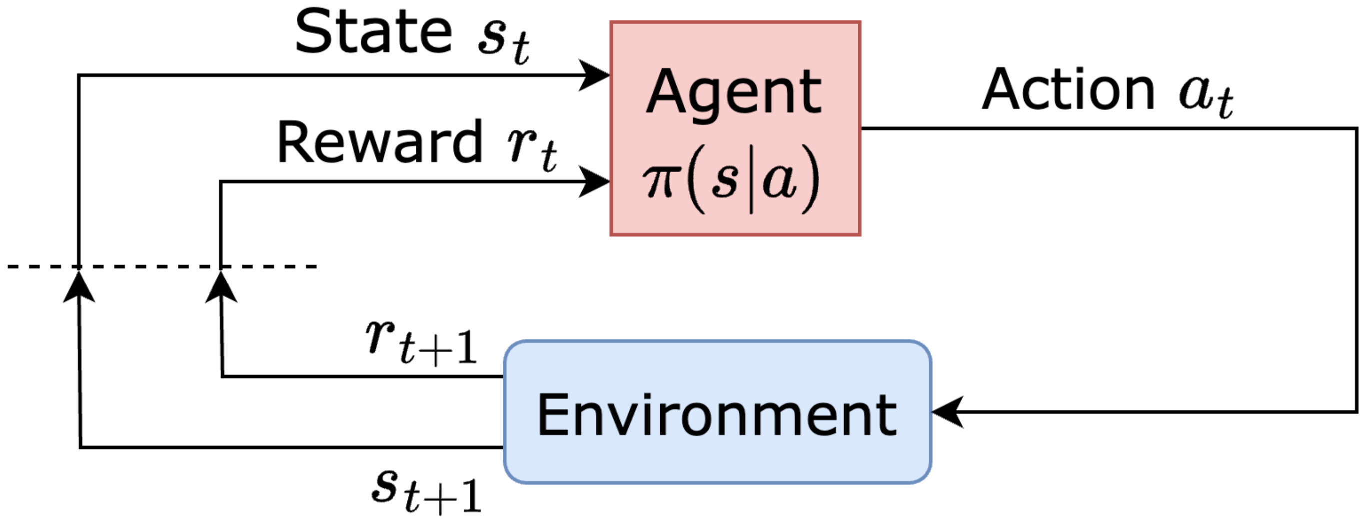
# What the paper does

1. **Learn problem-specific neighbor selection** policies in Local Search algorithms via **Deep Reinforcement Learning**

2. **Use a special type of neural network with symmetries** (equivariant nets) for scaling to larger instances than training

# Premise: Reinforcement Learning

- RL is a paradigm for solving generic decision-making problems




- The RL objective is maximizing the accumulation of rewards, i.e.,

$$\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T} r_t \,\middle|\, \pi\right]$$

# Premise: Deep Reinforcement Learning

- State and action spaces might be too large for exact algorithms
- **We can use NNs as universal function approximators!**
- DRL algorithms generally follow this workflow:
  1. Collect a bunch of experiences by running $\pi_\theta(a|s)$ on the env
  2. Update $\theta$ using some pseudo-gradient
- e.g., **Policy Gradient** algorithms

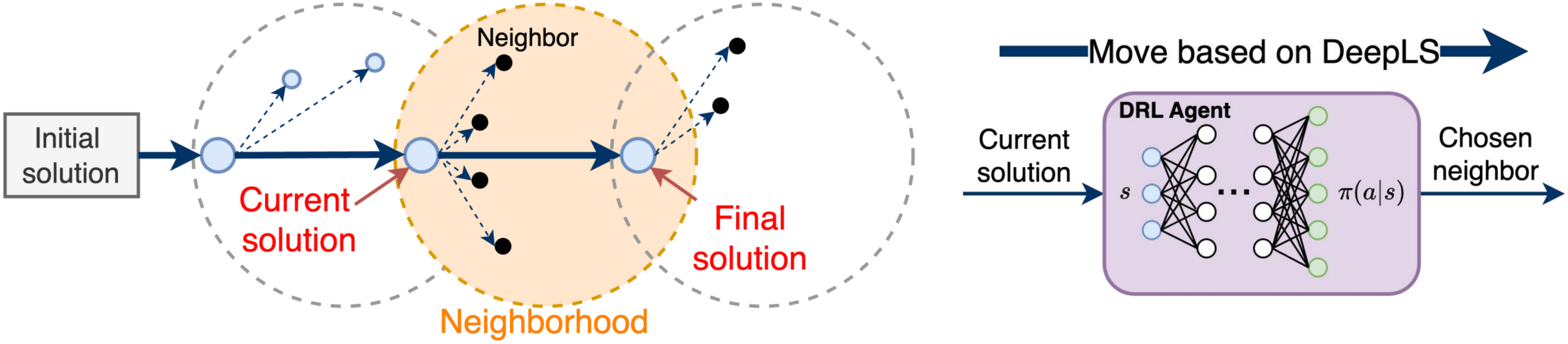
$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right],$$

where $\Psi_t$ may be one of the following:

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action $a_t$.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual.

[7] J. Schulman et al., “High-Dimensional Continuous Control Using Generalized Advantage Estimation” in ICLR, 2016

# Local Search as a DRL problem



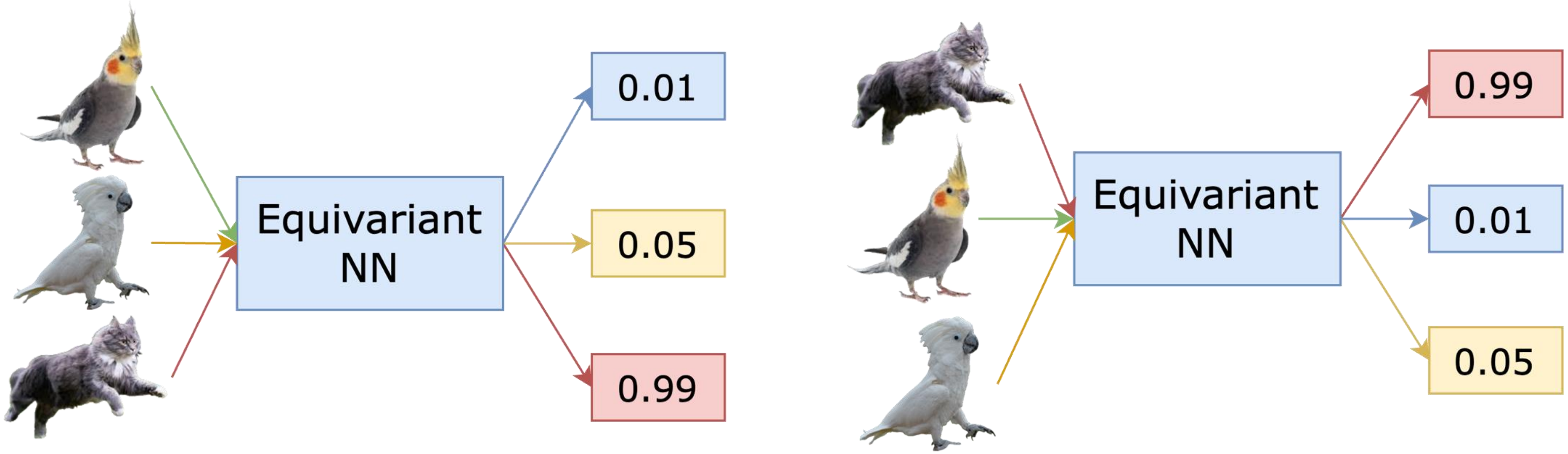

- **State**: value of each decision variable + variable-specific features
  - e.g., if variables are link weights, include link centrality and bandwidth
- **Action**: select which variable to perturb
- **Reward**: change in the objective function's value

# Permutation-equivariant neural nets

- Equivariant NNs are **NNs with symmetry**
- A set function $f: X^M \to Y^M$ is **permutation-equivariant** if:

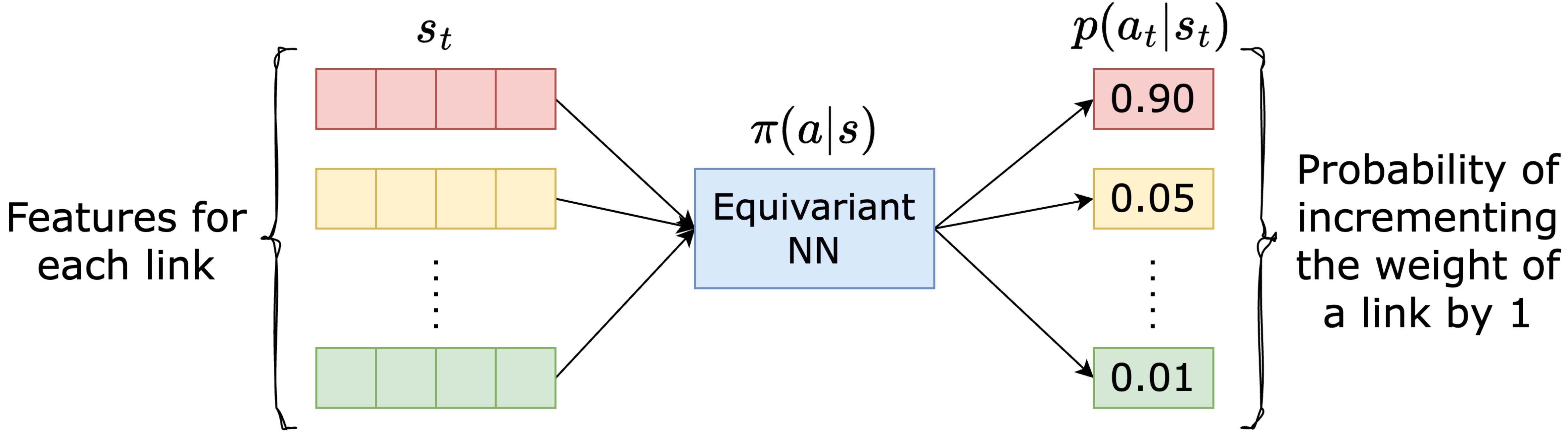
$$\pi\big(f(x)\big) = f\big(\pi(x)\big) \text{ for every permutation } \pi$$

- i.e., if we permute the ordering of the inputs, the same permutation is reflected in the ordering of the outputs



# Equivariant NNs for Local Search

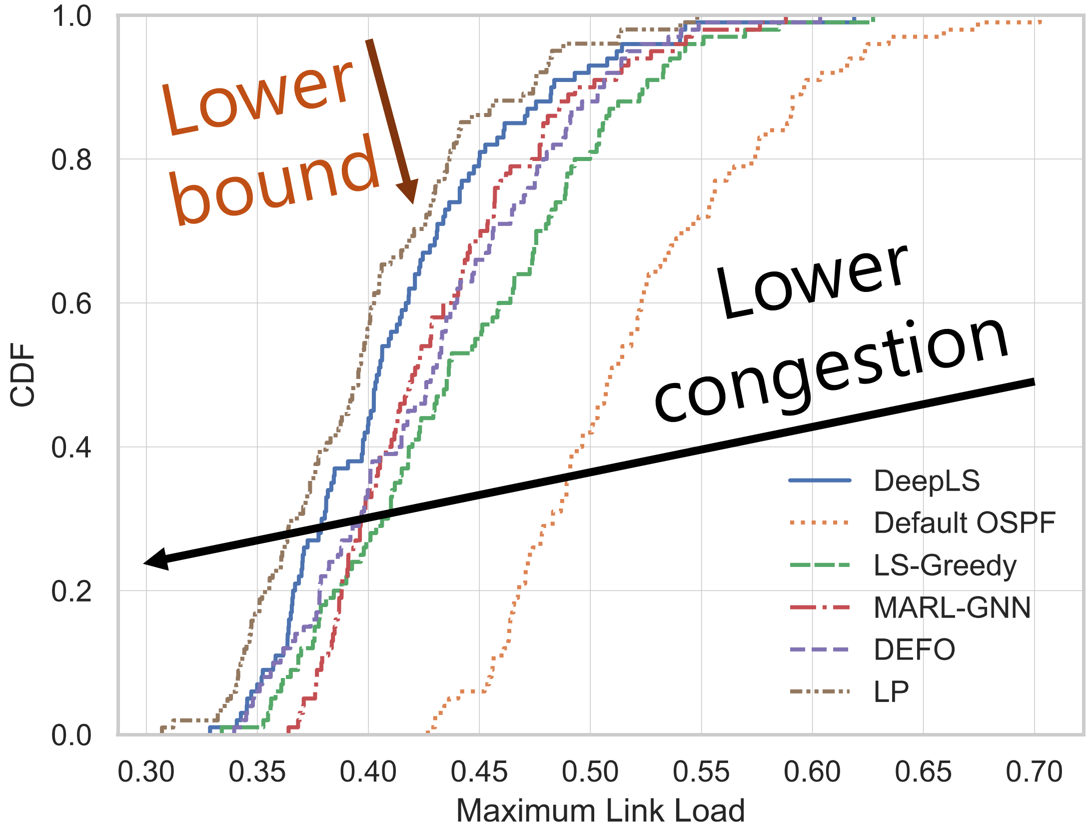
- **Problem**: traffic engineering with ECMP
- **Decision variables**: link weights




- n. of links is arbitrary; n. of features per link is fixed

# Illustrative numerical results

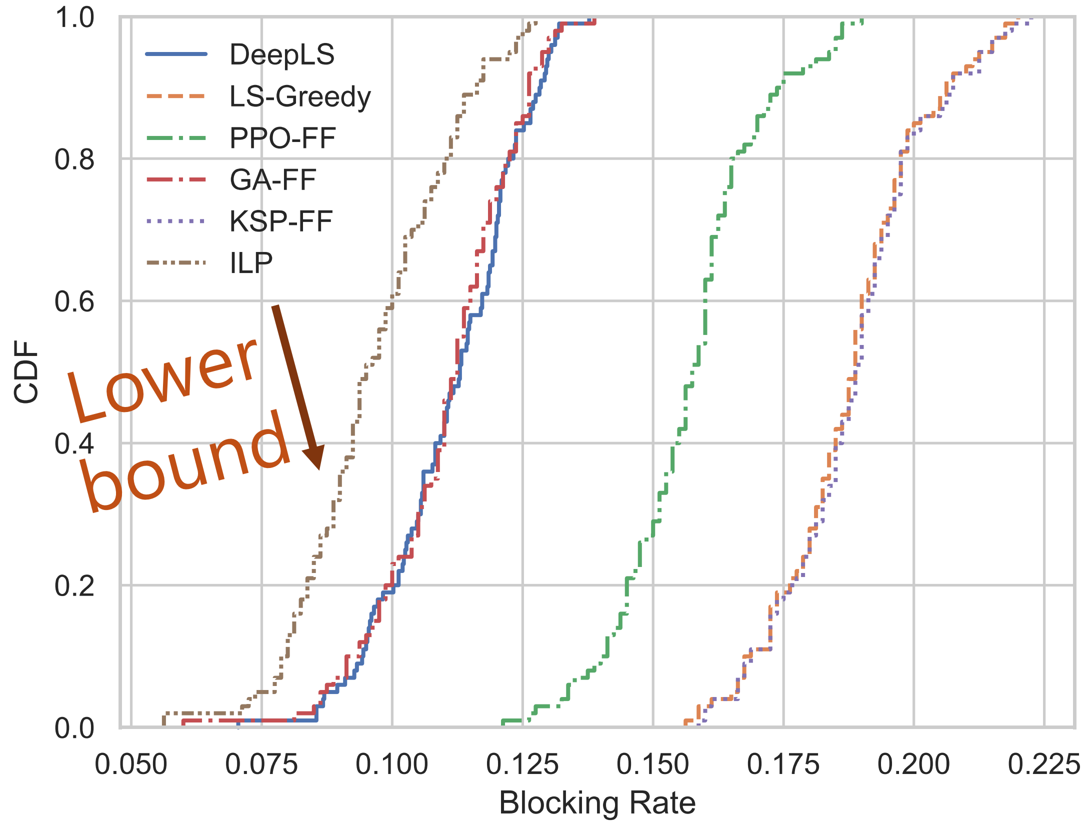
- **Problem**: traffic engineering with ECMP
- **Objective**: minimize the maximum link load




- DeepLS scales to instances up to **~4x larger than training**
- DeepLS outperforms competitive ML-based (MARL-GNN) and non (DEFO) algorithms

# Illustrative numerical results

- **Problem**: Routing and Spectrum assignment in optical networks
- **Objective**: maximize the n. of admitted requests




- DeepLS scales to instances up to **~5x larger than training**
- DeepLS is **on-par with a heavily customized Genetic Algorithm for RSA** (GA-FF), at a fraction of the complexity

# Takeaways

- The performance of Local Search algorithms strongly depends on the considered **neighbour selection policy**
- We can use Reinforcement Learning and simple neural networks to **autonomously learn the best neighbourhood selection policy for a specific problem class**
- Numerical results illustrate **better or comparable performance with respect to to handcrafted solutions**

# 2nd Case Study:

# Machine Learning for Low-Margin Optical-Network Planning

# Reference paper

IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 31, NO. 3, JUNE 2023 1293

## Dual-Stage Planning for Elastic Optical Networks Integrating Machine-Learning-Assisted QoT Estimation

Matteo Salani, Cristina Rottondi, *Senior Member, IEEE*, Leopoldo Ceré, and Massimo Tornatore, *Senior Member, IEEE*

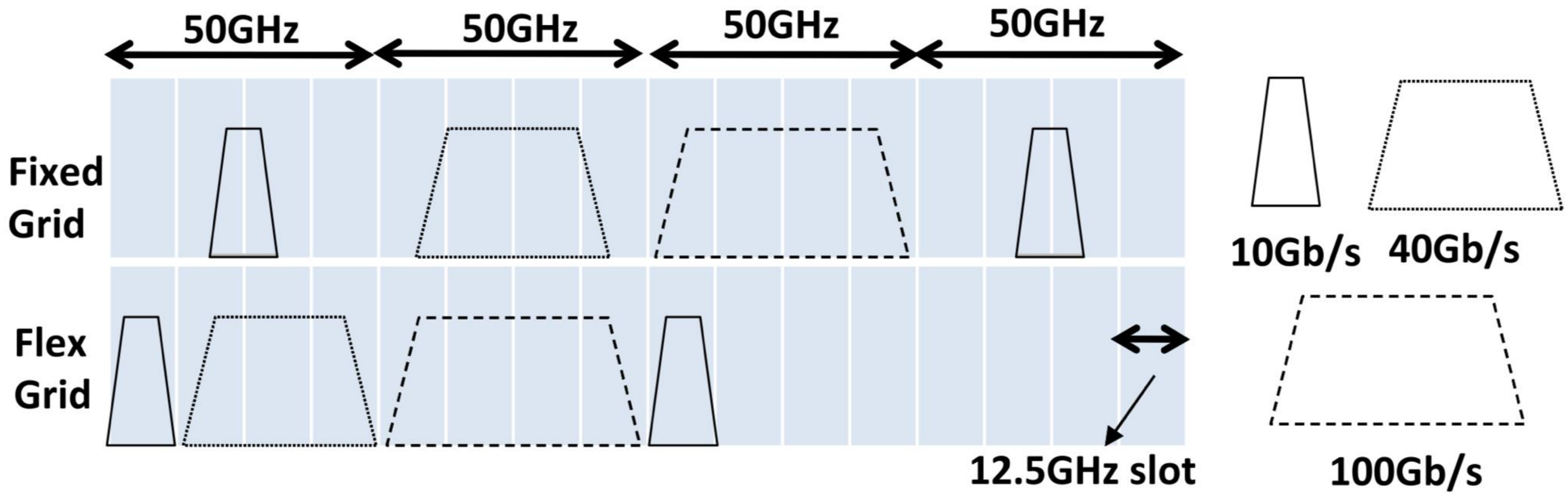
# What the paper does

1. Use Machine Learning to model the **Bit-Error-Rate** of optical signals in Elastic Optical Networks

2. Integrate the ML model with an ILP-based optimization algorithm to **forbid BER-unfeasible solutions**

# Premise: Elastic Optical Networks

- **Wavelength-Division Multiplexing** (WDM) optical networks use a **fixed spectrum grid** -> spectrum underutilization
- **Elastic Optical Networks** use finer (e.g., 12.5 GHz) subcarriers, with the possibility of using **multiple modulation formats**




[8] M. Imran et al., “A Survey of Optical Carrier Generation Techniques for Terabit Capacity Elastic Optical Networks” in COMST, 2017

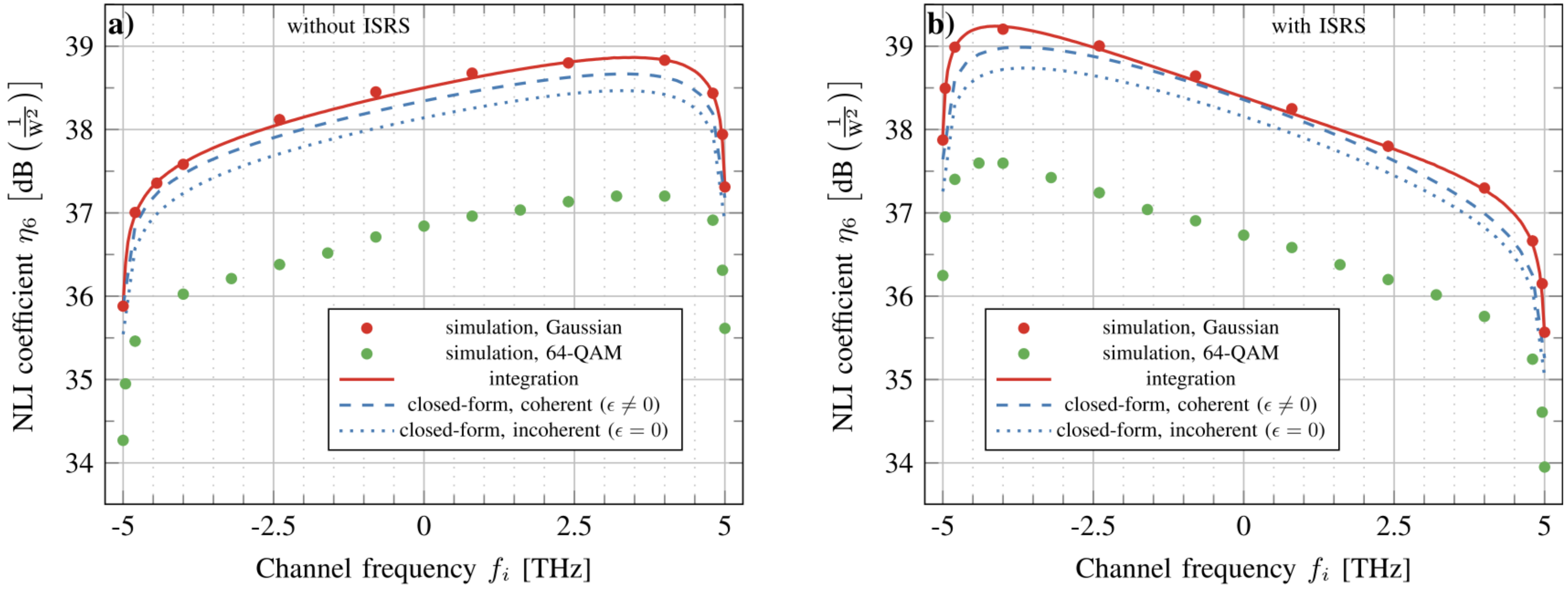
# Premise: Quality-of-Transmission (QoT)

- A lightpath must be of sufficient “quality” to function properly
- Generally, we want an **Optical Signal-To-Noise Ratio** (OSNR) achieving a **low-enough Bit-Error-Rate** (e.g., $10^{-3}$ pre-FEC)

- The OSNR at the receiver depends upon many things, including:
  - Total path length and fiber types
  - Nonlinear noise caused by the fiber’s Kerr effect
  - Non-flatness of the amplifier’s gain profile
  - Filtering penalties
  - Connector losses

# Premise: The Gaussian-Noise (GN) Model

- The GN Model is the current SoTA for optical-network planning
- Accurate and fast (it is just computing a formula!), but…

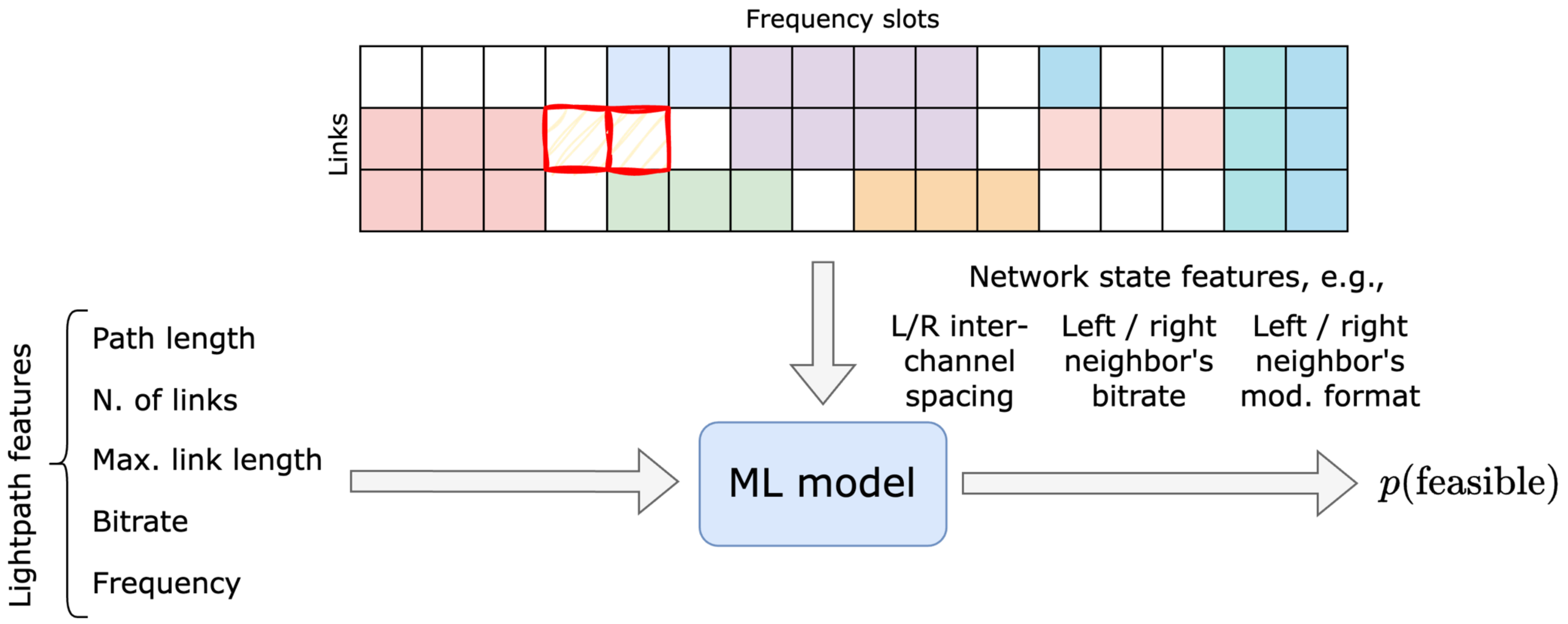



- … it's a conservative approximation, resulting in large margins

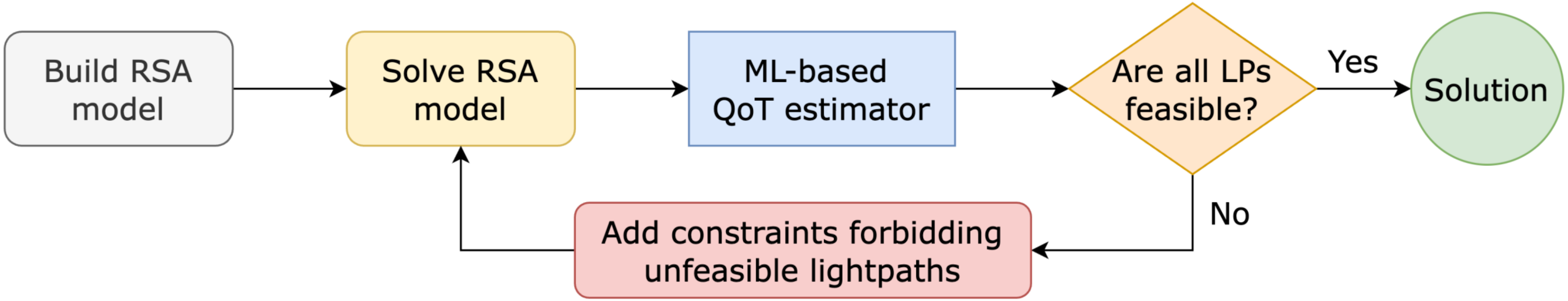
[9] D. Semrau et al., "A Closed-Form Approximation of the Gaussian Noise Model …", in JLT, 2019

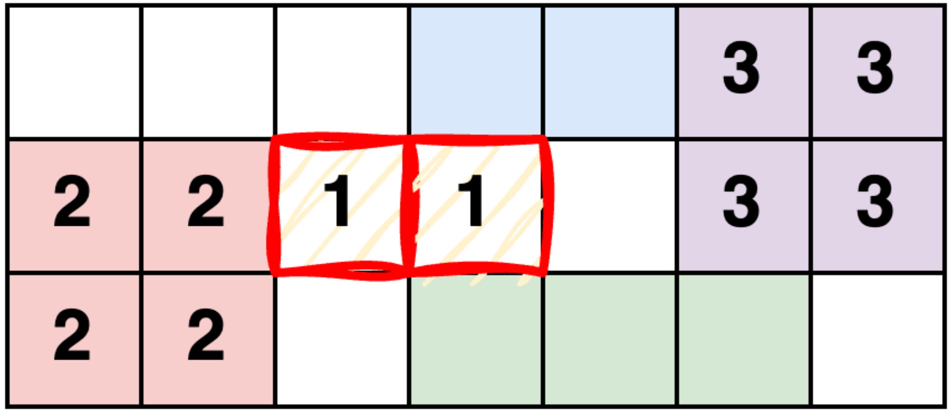
# Machine Learning for QoT Estimation

- We treat **QoT estimation as a classification problem**
- We predict whether a lightpath will be OSNR-feasible or not based on a general set of lightpath features



# Iterative RSA with Machine Learning



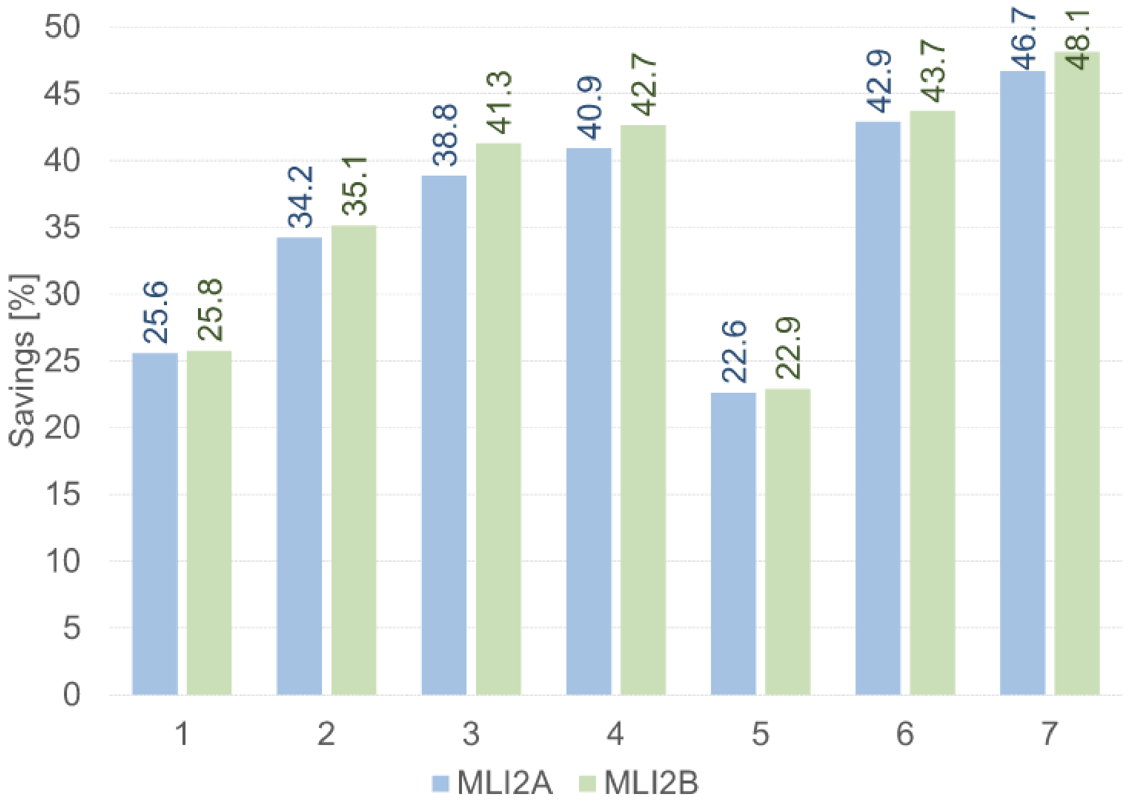



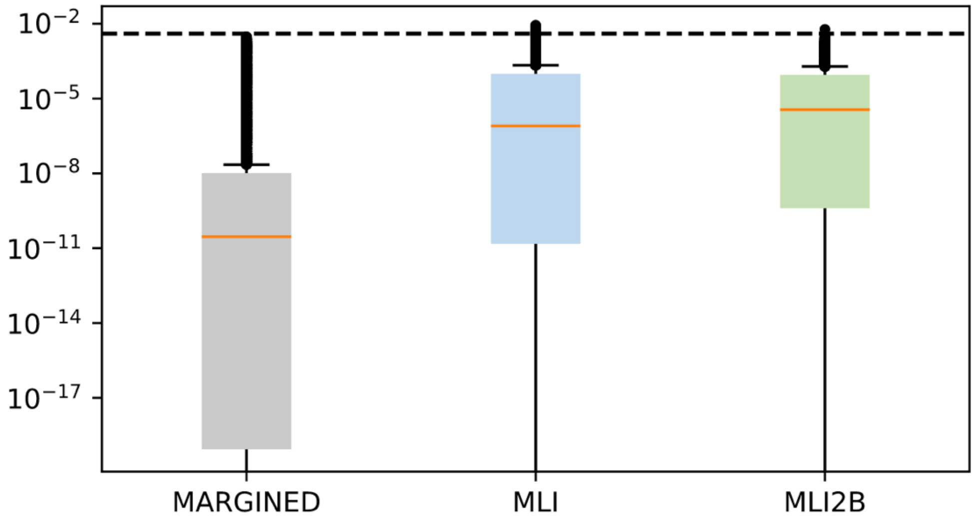

e.g., suppose that LP 1 is unfeasible

- We impose that LP 1 and its closest neighbors (LPs 2 and 3) cannot be simultaneously chosen
- Do the same for all unfeasible LPs

# Illustrative numerical results

- **Baseline**: margined reach tables for each path-modformat




| Framework | Total Inst. | Inf. Inst. | Inf. Inst. [%] | Inf. Lightp. [%] | Average MAX BER | Average BER |
|---|---|---|---|---|---|---|
| MARGINED | 13 | 0 | 0.00 | 0.00 | 1.91E-03 | 3.14E-05 |
| MLI | 78 | 19 | 21.59 | 0.21 | 5.59E-03 | 2.28E-04 |
| MLI2B | 80 | 1 | 1.11 | 0.01 | 3.32E-03 | 1.63E-04 |

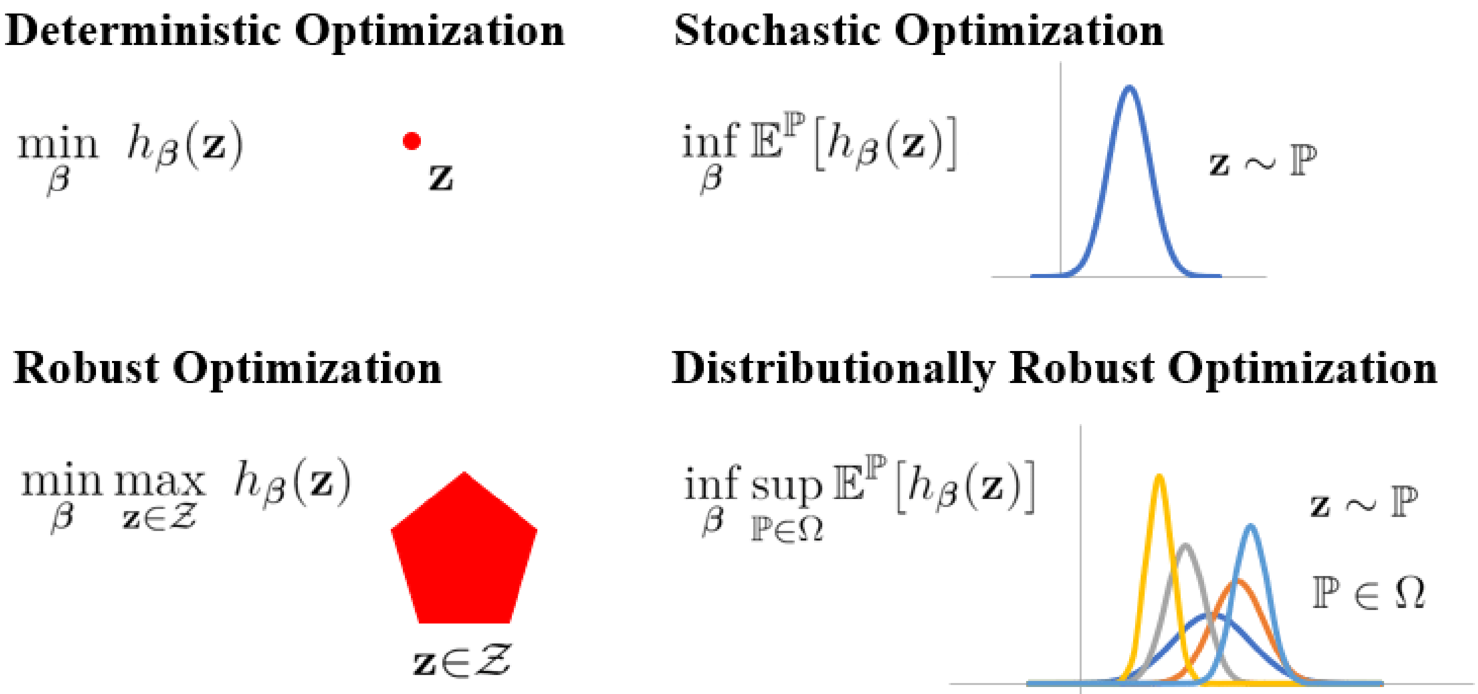
# Takeaways

- In practice, optical networks operate subject to **large QoT margins**, which can result in **spectrum underutilization**
- We can leverage data (field or testbed measurements) and ML for accurately estimating the lightpaths' QoT, enabling **near-zero margin optical-network planning**
- Numerical results illustrate **~35% spectrum savings** w.r.t. conventional margined network planning approaches

# Challenges & Research Opportunities

# Optimization Under Uncertainty

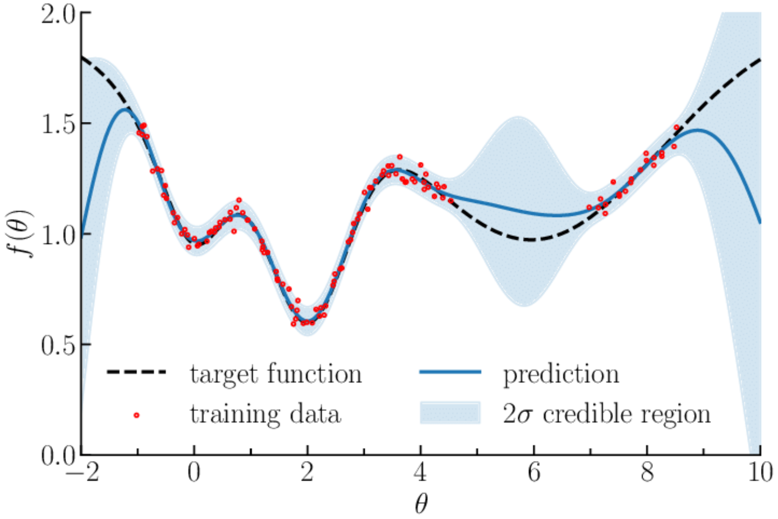
- Optimization under uncertainty is very well-studied

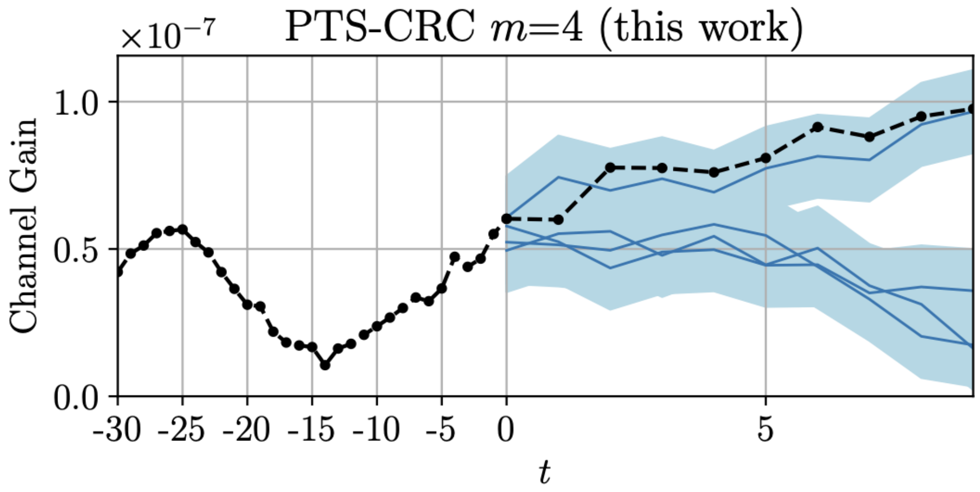



- The effectiveness of all of the above methodologies heavily relies on a **suitable model of parameter uncertainty**

# Uncertainty Quantification

- Many ML algorithm can estimate predictive uncertainties
  - Gaussian Processes, Quantile Regression, Conformal Prediction…




**How to best integrate ML uncertainty quantification with methods from optimization under uncertainty?**

[9] M. Zecchin et al., "Forking Uncertainties: Reliable Prediction…" in arXiv preprint, 2023

# Generalization

- “Learning without generalization is pointless” [3]
- It is expected that a ML-based optimization algorithm is able to generalize to problem instances “different” than training

- One aspect of generalization: **scaling to large instances**

**How to design ML models and input representations that enable consistent scalability to instances larger than training?**

[3] Y. Bengio et al., “ML for Combinatorial Optimization: A Methodological Tour D’Horizon” in EJOR, 2021

# Generalization

- Generalization is closely related to **instance distribution**

**How do we generate training data that covers the distribution of instances we are interested in?**

**How can we enable generalization / fast adaptation to instances from a data distribution different than training?**

# Interpretability

- Operators are reluctant about using technologies they cannot satisfactorily explain “why they work” to stakeholders

- Machine Learning models not only are black boxes, but they can perform arbitrarily bad under ”unlucky” or adversarial inputs

**How to distill human-understandable, actionable algorithmic insights from ML-based optimization algorithms?**

that's all folks