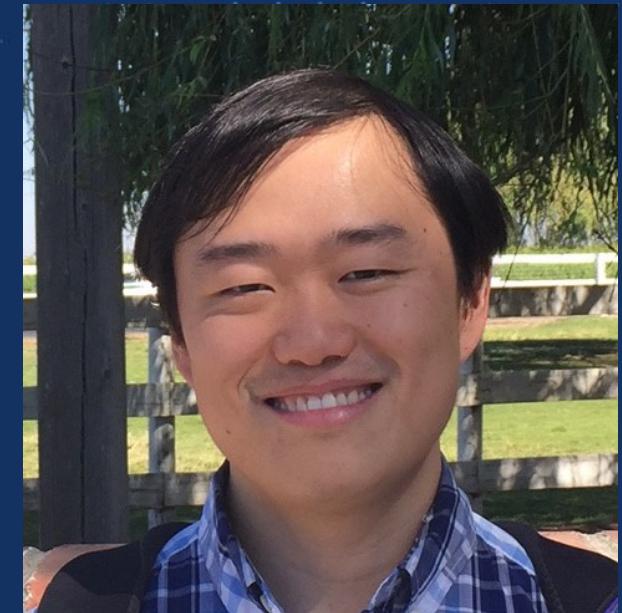


# Stronger Together: Marrying Deep Models with Traditional Symbolic Decision-Making

Yuandong Tian  
Research Scientist

Meta AI (FAIR)



# Large Language Models (LLMs)



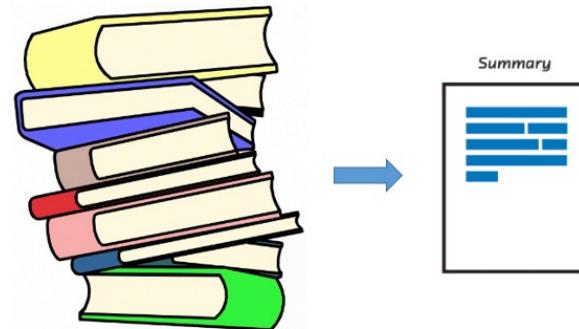
Content Creation



AI Agents



AI co-pilot



Summarization

# What LLMs cannot do well yet?



I'm going from **Seattle** to **California** from **November 6 to 10, 2023**. I have a **budget of \$6,000**. For lodging, I prefer an **entire room** and the accommodations must be **pet-friendly**.

Travel planning

# What LLMs cannot do well yet?



User

I'm going from **Seattle to California** from **November 6 to 10, 2023**. I have a **budget of \$6,000**. For lodging, I prefer an **entire room** and the accommodations must be **pet-friendly**.



## Information Collection

[Tool] **CitySearch[California]**

[Result] San Francisco, Los Angeles, ..., San Diego



[Tool] **FlightSearch[Seattle, San Francisco, 2023-11-06]**

[Result] No Flights.

[Tool] **FlightSearch[Seattle, Los Angeles, 2023-11-06]**

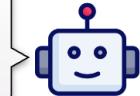
[Result] Flight Number: F123, 13:40-16:12, Cost: \$120

[Tool] **DistanceMatrix[Los Angeles, San Diego, taxi]**

[Result] Duration: 1 hour 57 mins, Distance: 193 km, Cost: \$200

## Planning

The plan must adhere to certain **constraints**, e.g., **user needs** and **commonsense**. It's also vital to ...



### User Needs (Hard Constraints)

1. Budget: \$6000
2. Room Type: Entire Room
3. Room Rule: Pet-friendly



### Commonsense Constraints

1. Reasonable City Route
2. Diverse Restaurants
3. Diverse Attractions
4. Non-conflicting Transportation
5. Accommodation meets Minimum Night



## Delivery Plan

2023-11-06	2023-11-07	2023-11-08	2023-11-09	2023-11-10
Seattle > Los Angeles Flight: F123 (13:40-16:12), Cost: \$120	Los Angeles • Breakfast: Chick-fil Minar • Lunch: Radhani Restaurant • Dinner: Domino's Pizza • Attractions: The Lost Mughal • Accommodation: Luxury building studio	Los Angeles -> San Diego • Take taxi to San Diego • Breakfast: Open Yard • Lunch: Harry's Bar • Dinner: Dragon Way • Attractions: La Jolla Shores Park; California Tower • Accommodation: East Side Apartment	San Diego • Breakfast: Baskin Robbins • Lunch: Harry's Bar • Dinner: Dragon Way • Attractions: La Jolla Shores Park; California Tower • Accommodation: East Side Apartment	San Diego -> Seattle Flight: F189 (7:59-10:56), Cost: \$300

2023-11-06

### Seattle -> Los Angeles

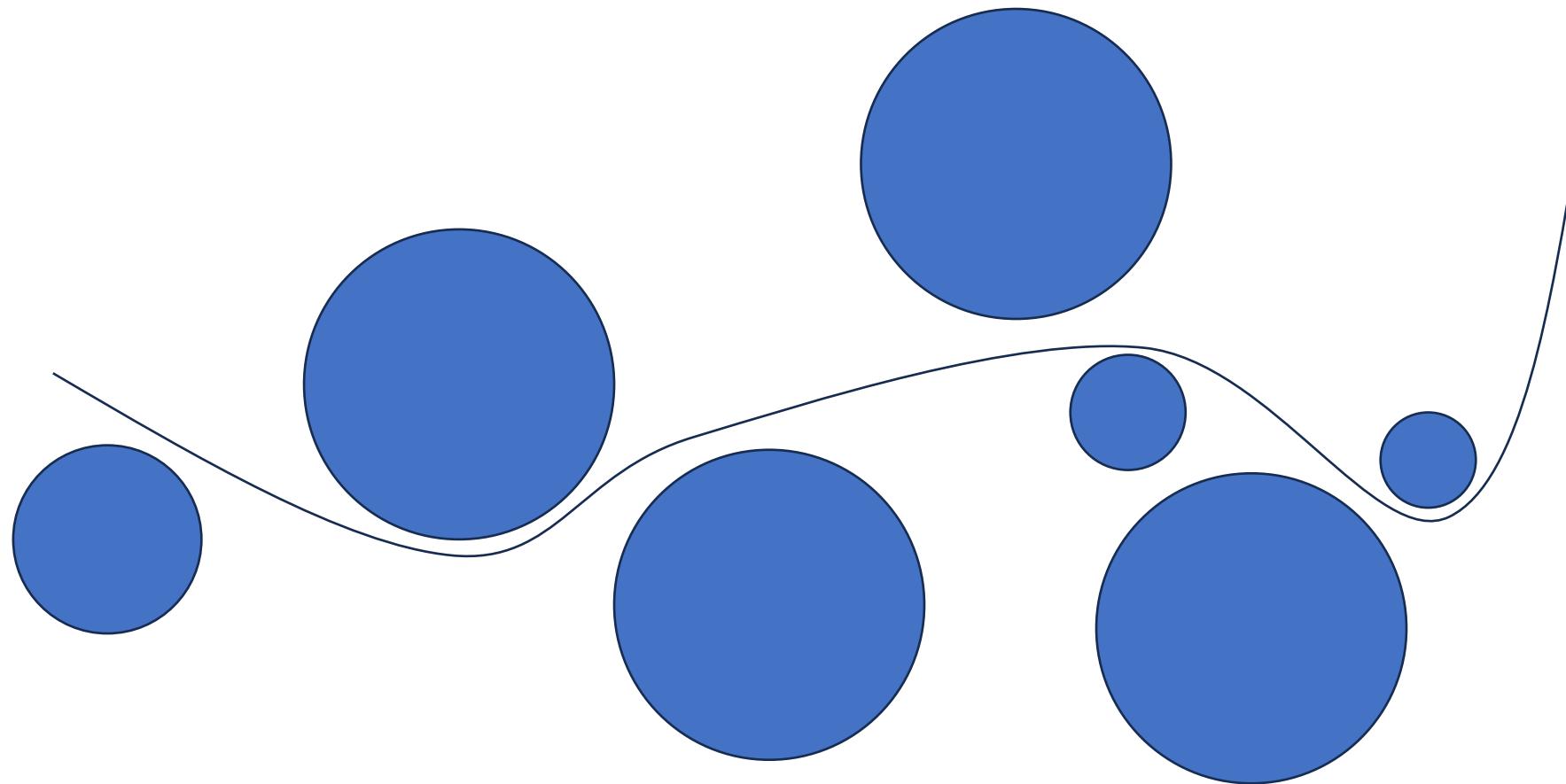
- Flight: F123, (13:40-16:12), Cost: \$120
- Accommodation: Luxury building studio
- Dinner: The Attraction

# Using SoTA LLMs for Travel Planning (not great)

Even SoTA LLMs struggle for such hard planning problems

		Validation (#180)						Test (#1,000)					
		Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
			Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
First tool use, Then plan the travel	Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
	<b>Two-stage</b>												
	Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
	Mixtral-8x7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
	Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
	GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
Ground-truth tool use, Then plan the travel	GPT-4-Turbo (OpenAI, 2023)	<b>89.4</b>	<b>61.1</b>	<b>2.8</b>	<b>15.2</b>	<b>10.6</b>	<b>0.6</b>	<b>93.1</b>	<b>63.3</b>	<b>2.0</b>	<b>10.5</b>	<b>5.5</b>	<b>0.6</b>
	<b>Sole-planning</b>												
	Direct <sub>GPT-3.5-Turbo</sub>	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
	CoT <sub>GPT-3.5-Turbo</sub>	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
	ReAct <sub>GPT-3.5-Turbo</sub>	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
	Reflexion <sub>GPT-3.5-Turbo</sub>	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
	Direct <sub>Mixtral-8x7B-MoE</sub>	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
	Direct <sub>Gemini Pro</sub>	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
	Direct <sub>GPT-4-Turbo</sub>	<b>100</b>	<b>80.4</b>	<b>17.2</b>	<b>47.1</b>	<b>22.2</b>	<b>4.4</b>	<b>100</b>	<b>80.6</b>	<b>15.2</b>	<b>44.3</b>	<b>23.1</b>	<b>4.4</b>

Hard to find solutions that satisfy all the constraints



# Can LLM solve Tic-tac-toe?

Claude 3 Opus

I'll place my O in the right-middle square:

| X | O  
---+---+---

| X | O  
---+---+---

O | X |

Now, the game has ended in a draw. There is no win for either player.

You placed an X in the top-middle square. Here's the updated board:

| X | O  
---+---+---  
| X |

---+---+---  
| |

I'll place my O in the bottom-left corner to block your potential win:

| X | O  
---+---+---  
| X |

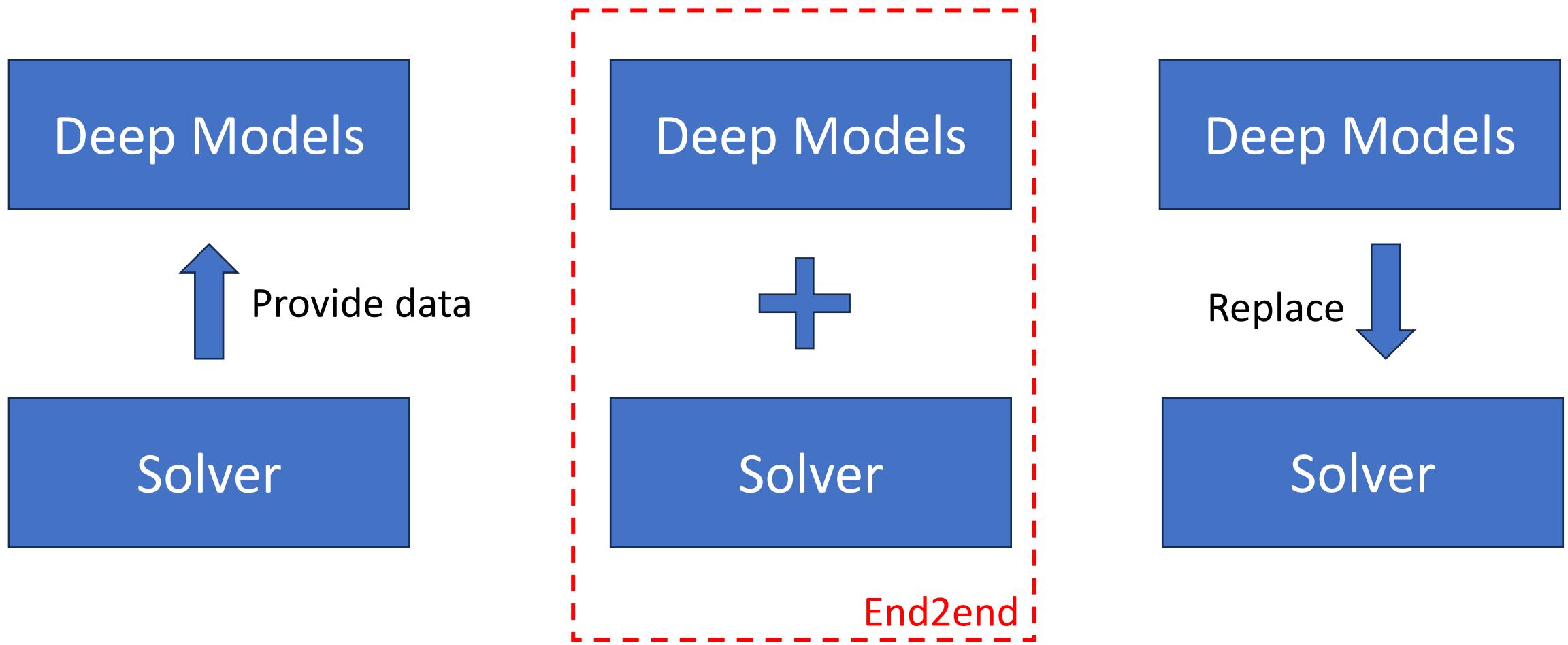
---+---+---  
O | |

It's your turn again. Where would you like to place your next X?



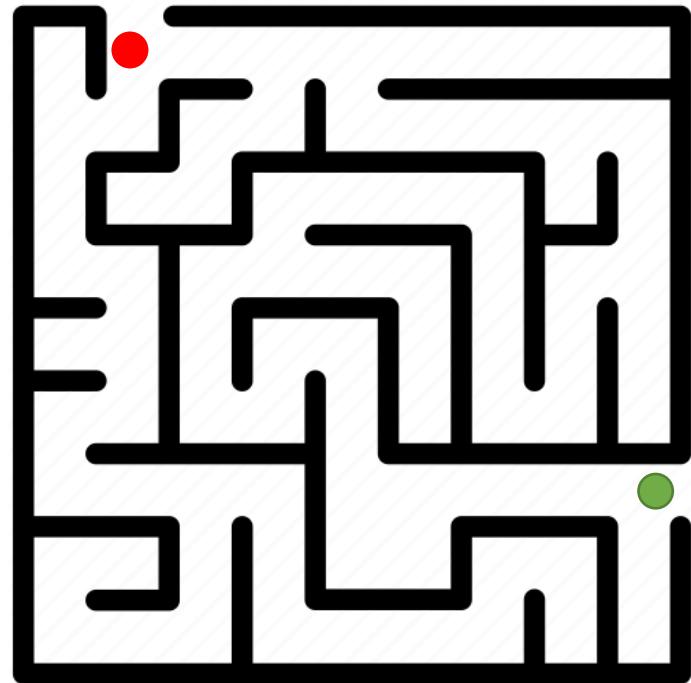
Do you think placing at the bottom-left corner is a good move?

# Overview

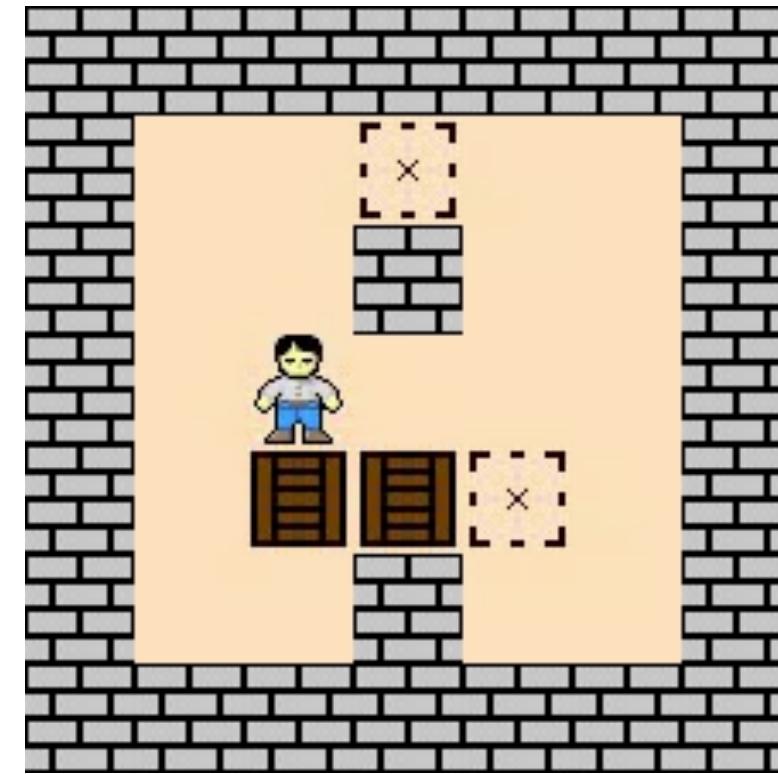


# Part I: Better Transformers with Data from Solvers

# Tasks

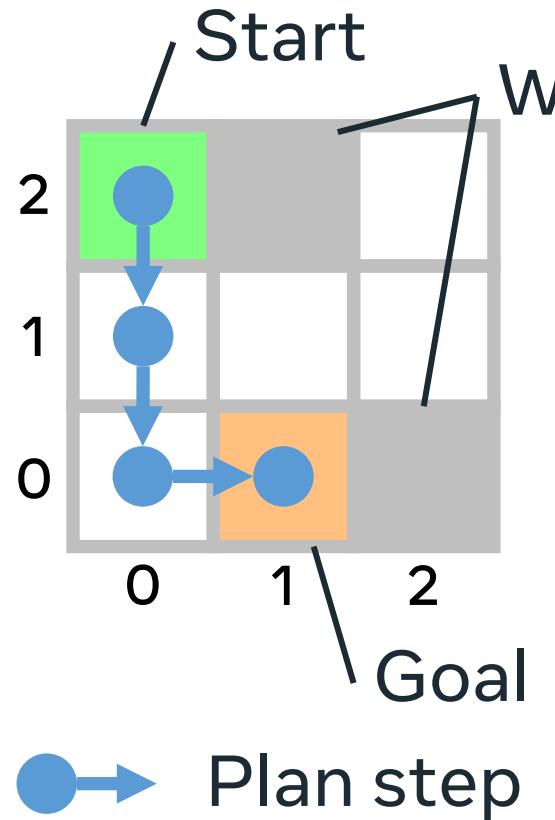


Maze Navigation



Sokoban (倉庫番)

# Planning and Sequential Decision Making



<prompt>

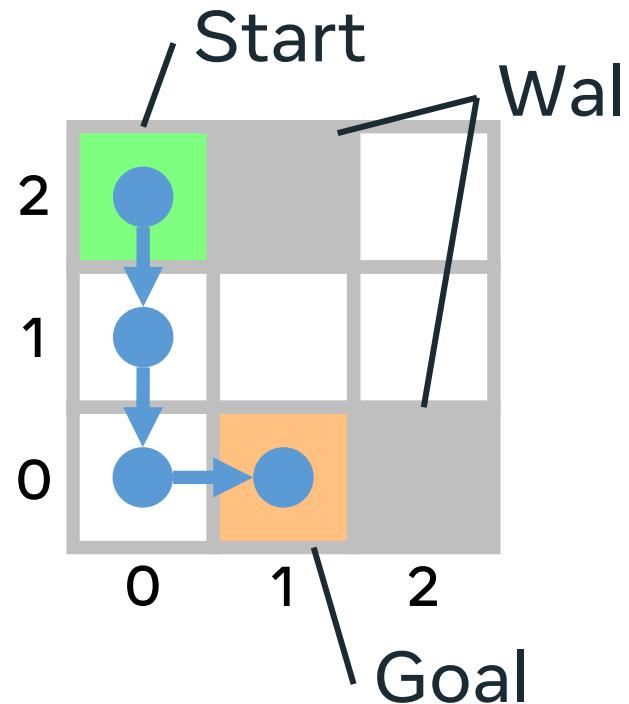
```
bos
start 0 2
goal 1 0
wall 1 2
wall 2 0
eos
```



<plan> (Task Solution)

```
bos
plan 0 2
plan 0 1
plan 0 0
plan 1 0
eos
```

# Computing a Plan with A\* Search



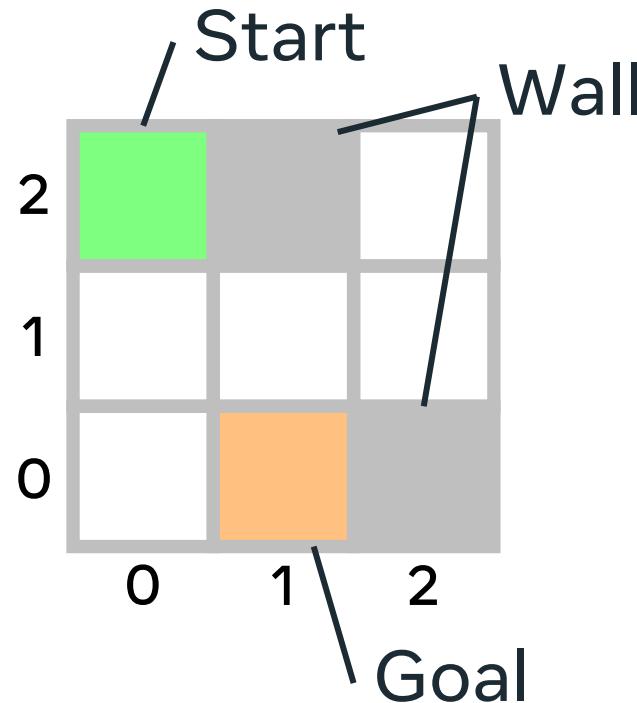
→ Plan step

● Frontier state

● Closed state

A\* computes a plan by manipulating two sets.

# Computing a Plan with A\* Search



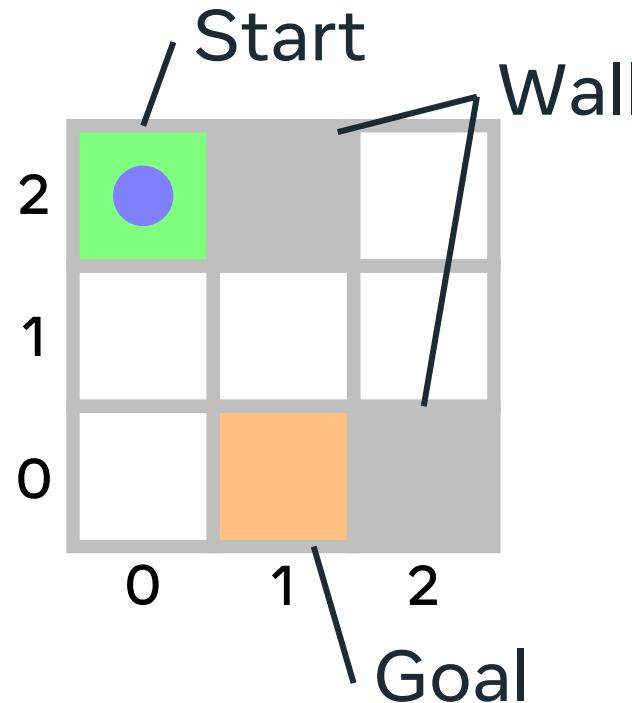
<trace><plan>

→ Plan step

● Frontier state

● Closed state

# Computing a Plan with A\* Search



→ Plan step

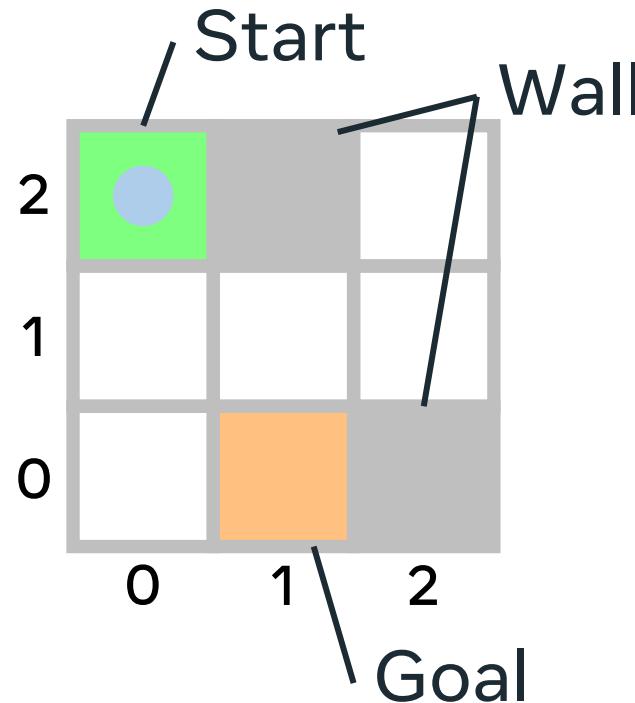
● Frontier state

● Closed state

<trace><plan>

create 0 2 c0 c3

# Computing a Plan with A\* Search



→ Plan step

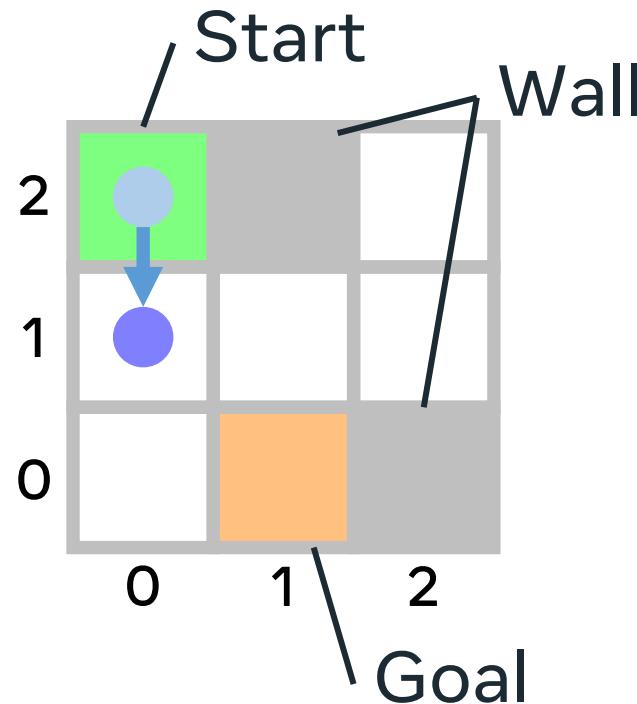
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3  
close 0 2 c0 c3
```

# Computing a Plan with A\* Search



→ Plan step

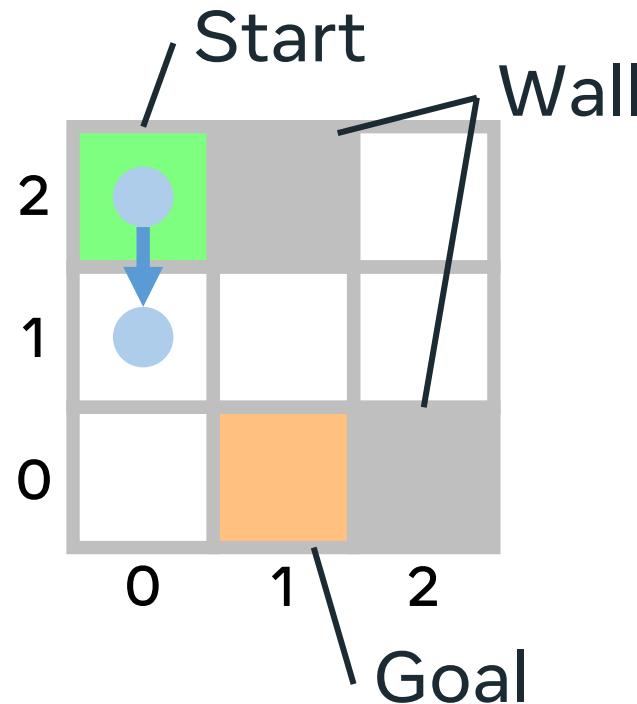
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3  
close 0 2 c0 c3  
create 0 1 c1 c2
```

# Computing a Plan with A\* Search



→ Plan step

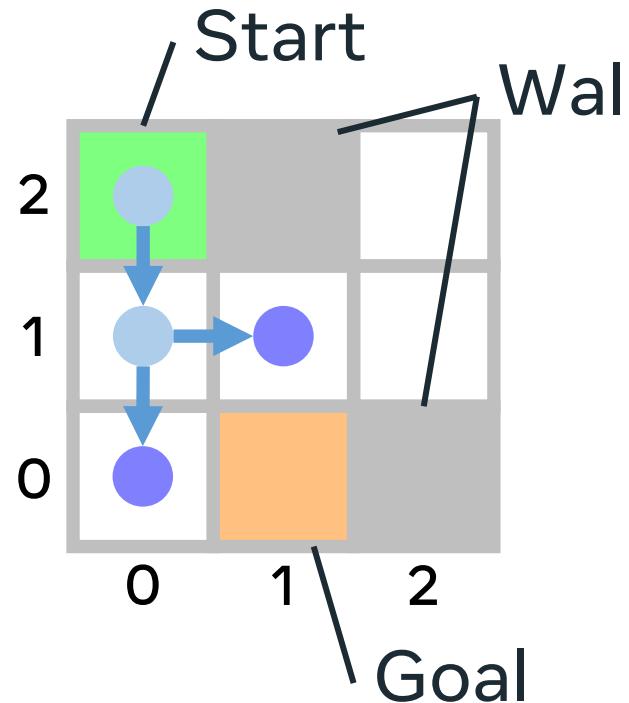
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
```

# Computing a Plan with A\* Search



→ Plan step

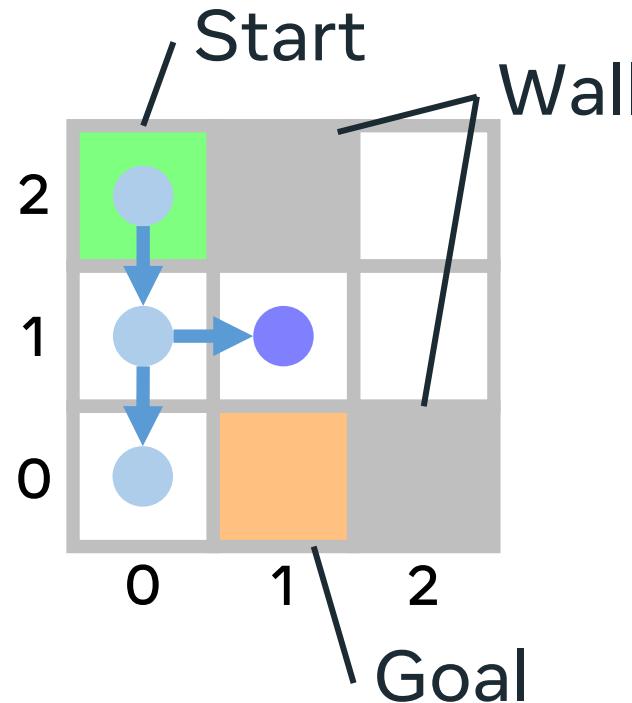
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
```

# Computing a Plan with A\* Search



→ Plan step

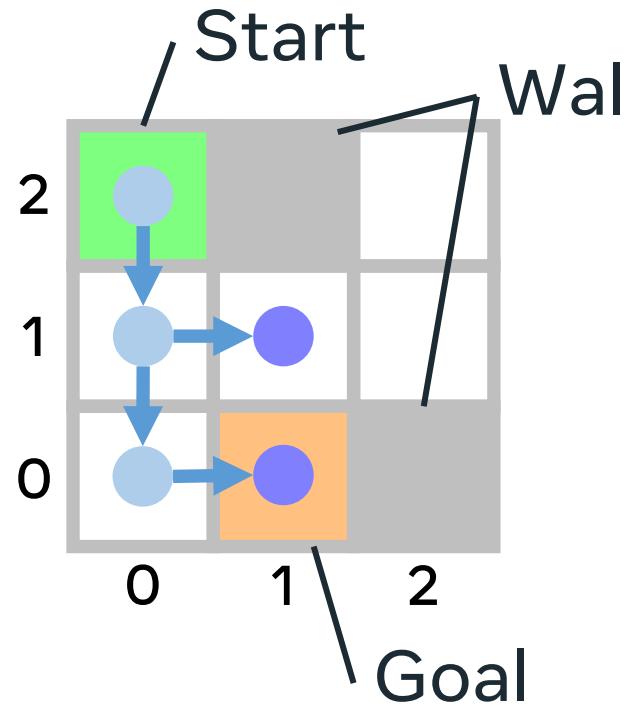
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
close 0 0 c2 c1
```

# Computing a Plan with A\* Search



→ Plan step

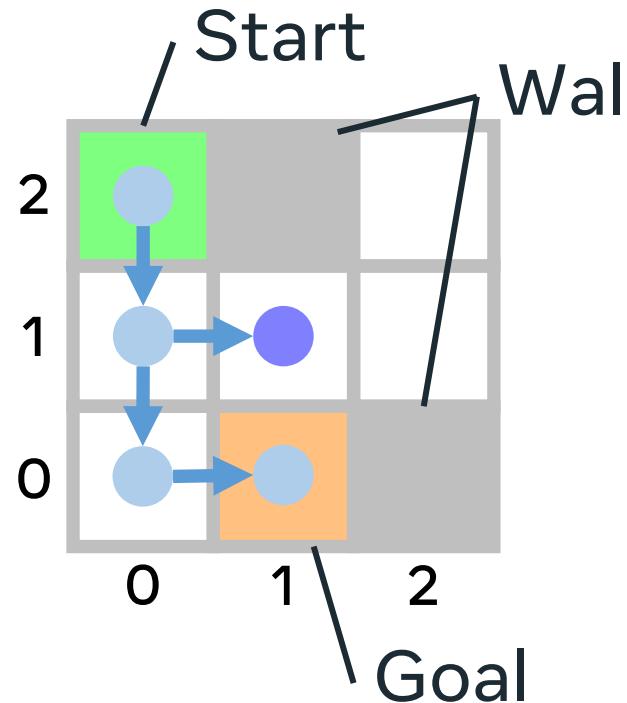
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
close 0 0 c2 c1
create 1 0 c3 c0
```

# Computing a Plan with A\* Search



→ Plan step

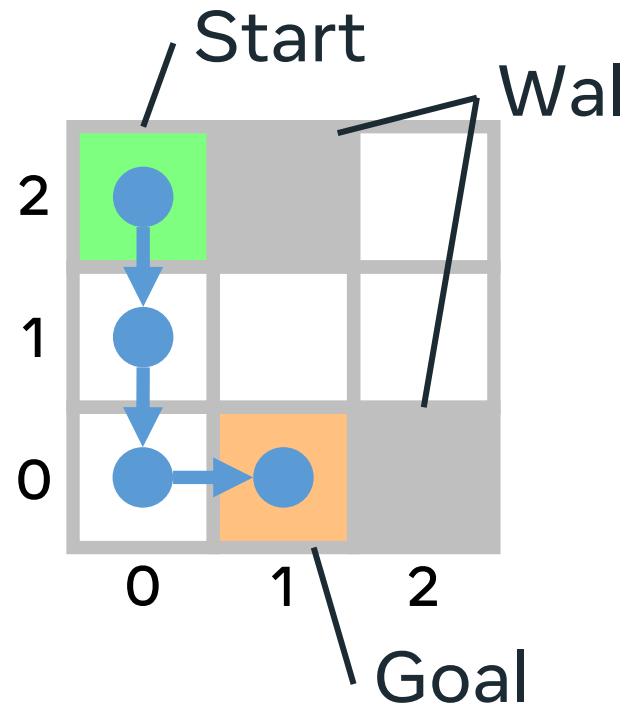
● Frontier state

● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
close 0 0 c2 c1
create 1 0 c3 c0
close 1 0 c3 c0
```

# Computing a Plan with A\* Search



→ Plan step

● Frontier state

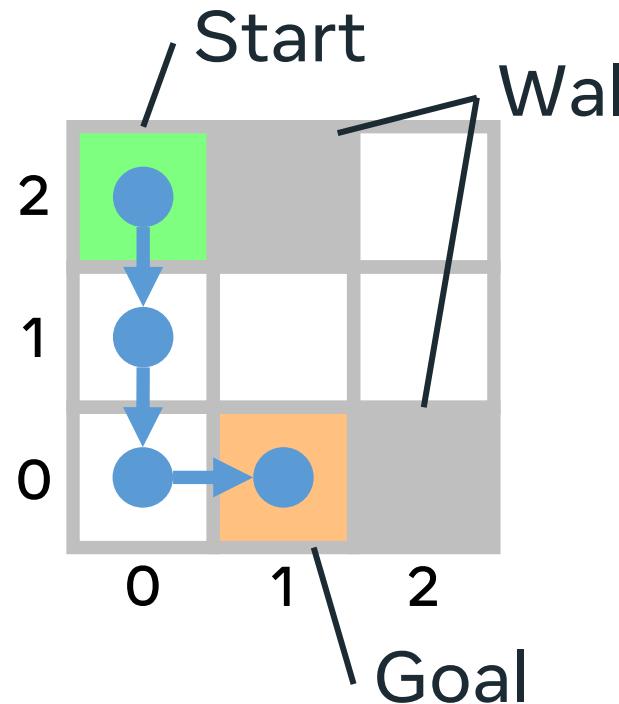
● Closed state

<trace><plan>

```
create 0 2 c0 c3
close 0 2 c0 c3
create 0 1 c1 c2
close 0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
close 0 0 c2 c1
create 1 0 c3 c0
close 1 0 c3 c0
```

Get optimal path from the search dynamics

# Imitating A\* Search as a Token Prediction Task



● → Plan step

● Frontier state

● Closed state

<prompt>

```
bos  
start 0 2  
goal 1 0  
wall 1 2  
wall 2 0  
eos
```

<trace><plan>

```
create 0 2 c0 c3  
close 0 2 c0 c3  
create 0 1 c1 c2  
close 0 1 c1 c2  
create 0 0 c2 c1  
create 1 1 c2 c1  
close 0 0 c2 c1  
create 1 0 c3 c0  
close 1 0 c3 c0
```

# Training Transformers to Solve Planning Tasks

Dataset

{<prompt><plan>}

{<prompt><trace><plan>}



Model

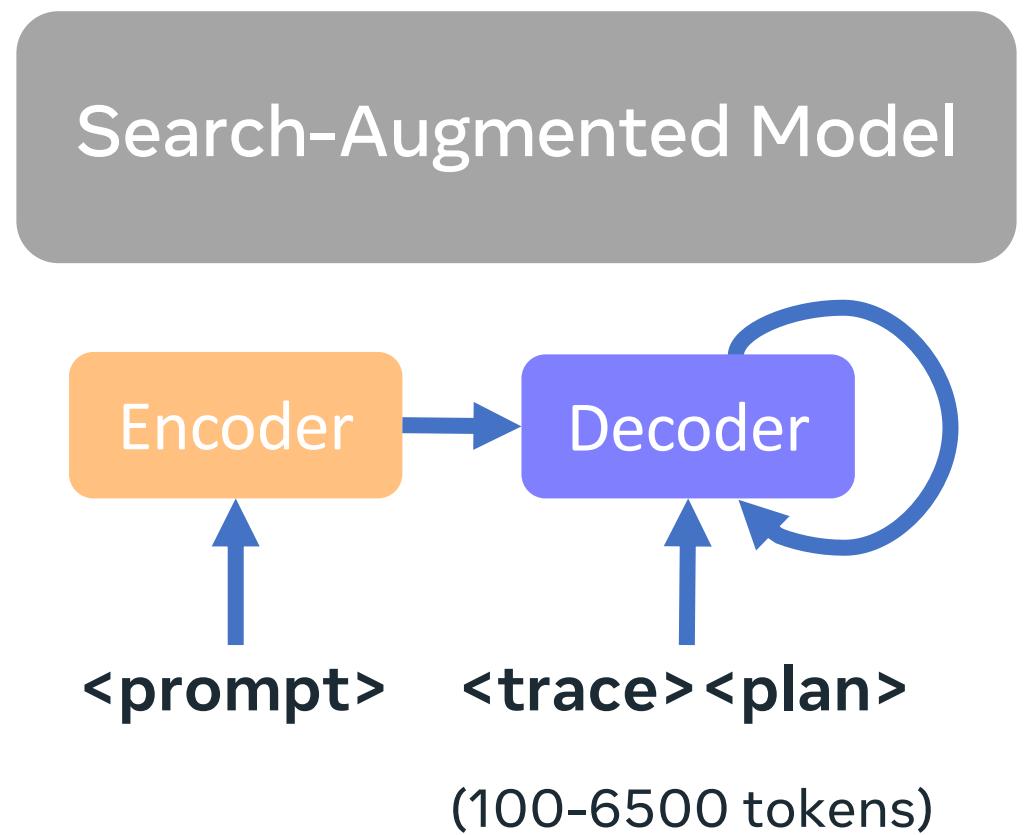
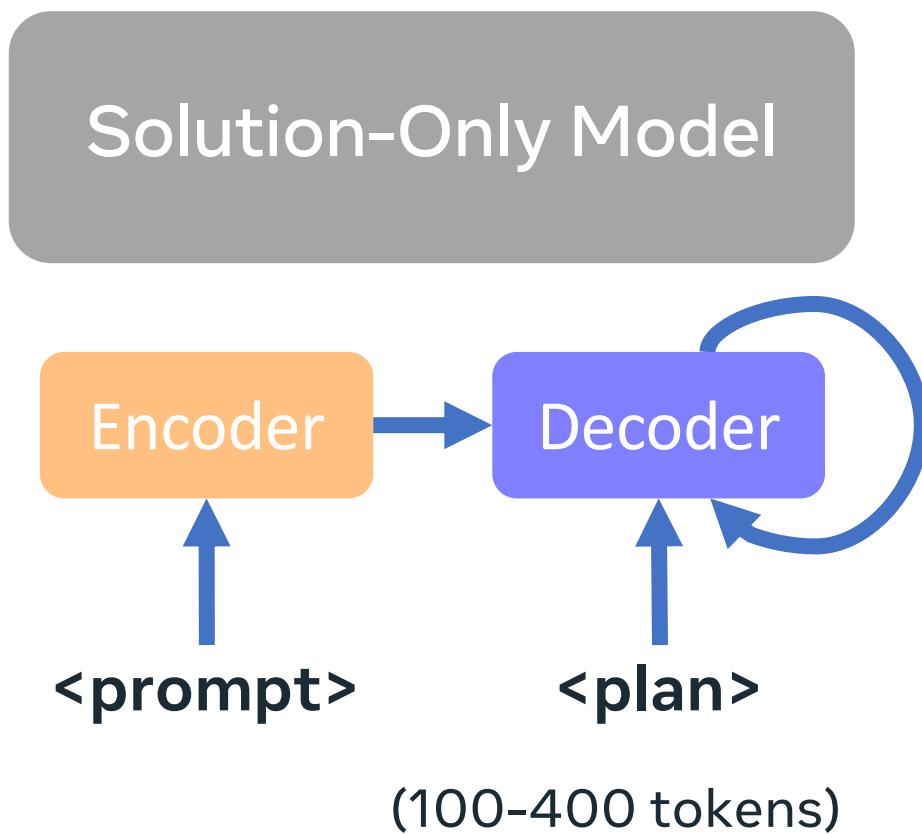
Solution-Only Model

Search-Augmented Model

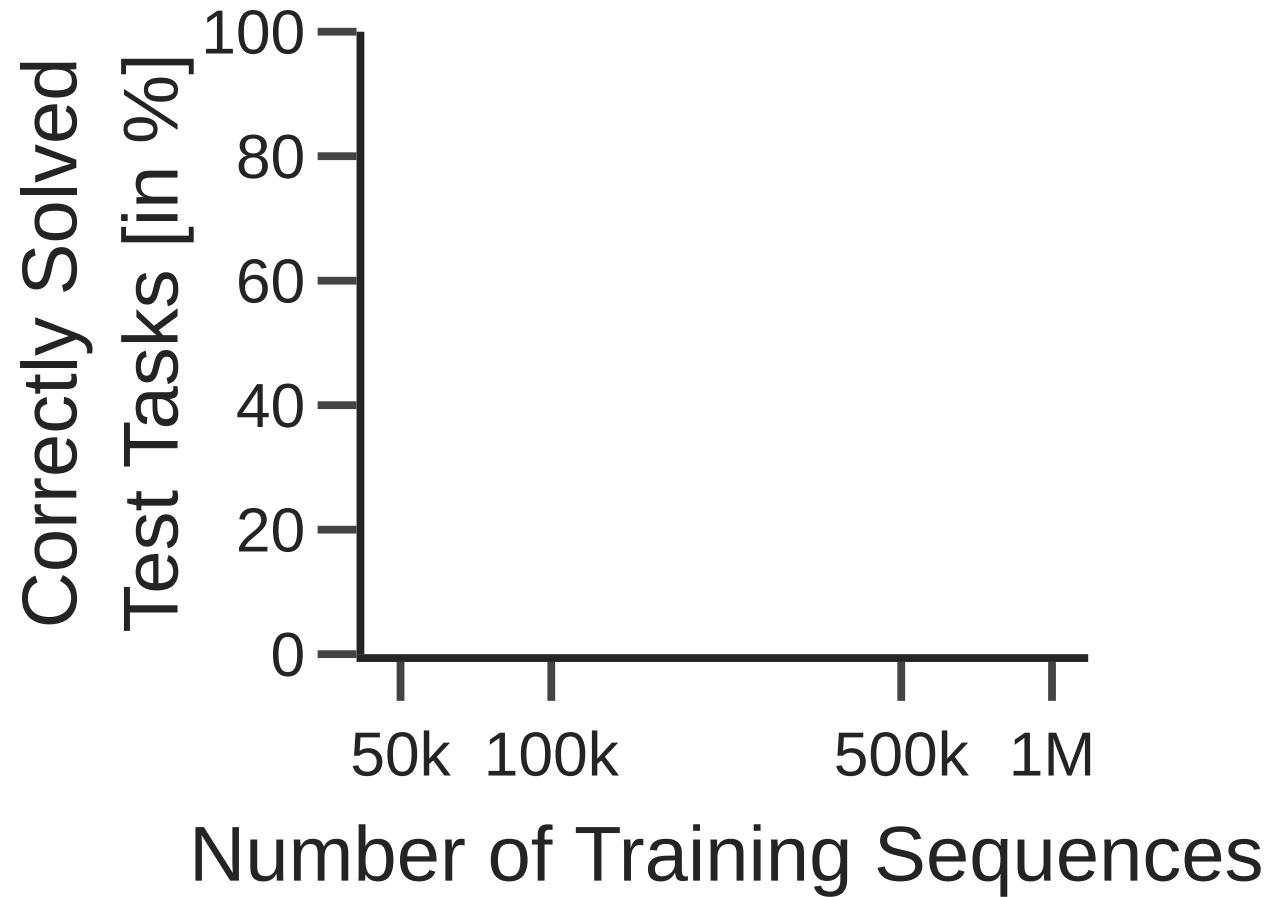
# Training Method

Train a Transformer to predict the next token via teacher forcing.

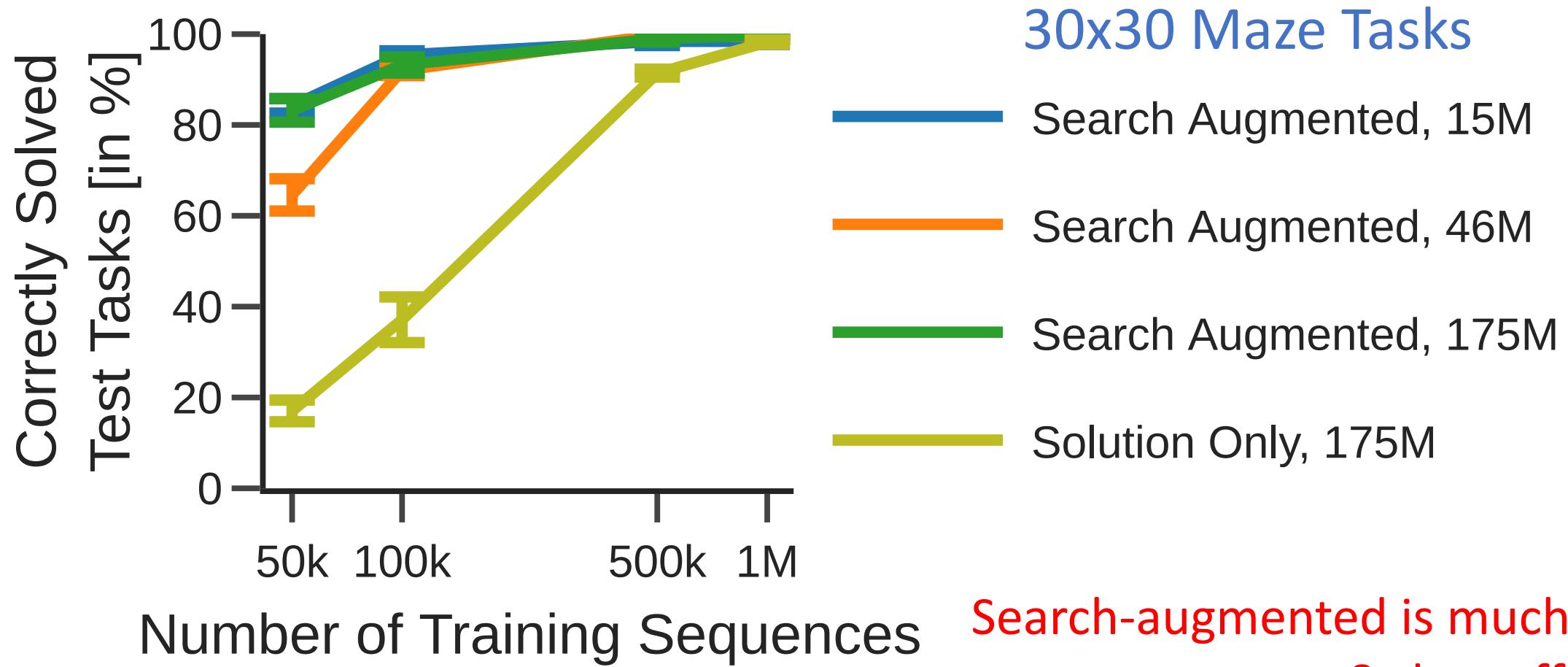
Model



# Search-Augmented vs. Solution-Only Models

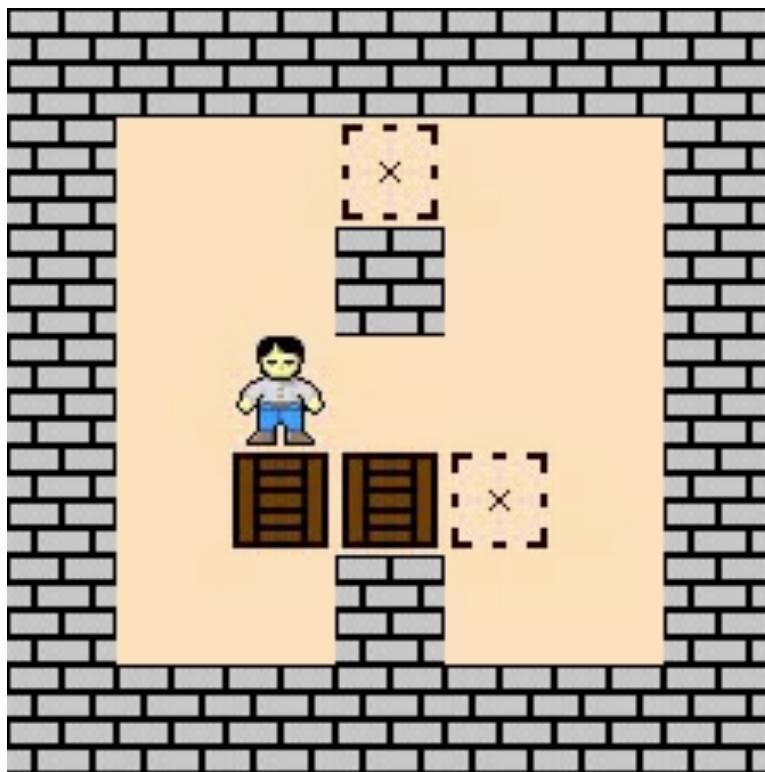


# Search-Augmented vs. Solution-Only Models

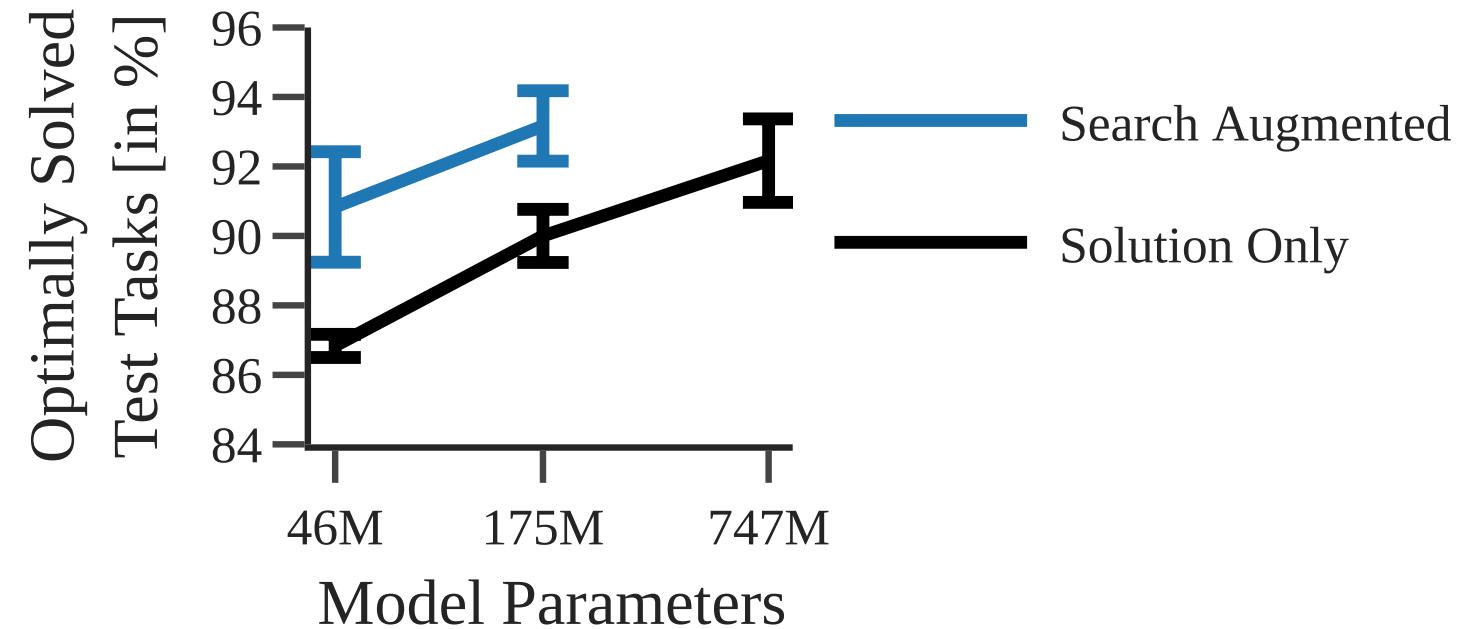


Search-augmented is much more parameter & data efficient!

# Sokoban Experiments



This animation was composed using image icons from  
<https://github.com/morenod/sokoban> (accessed 2023-11-21).



# How to go beyond?

Imitation Learning



Fine-tuning

Using solver's trace to train the Transformer with teacher forcing

Fine-tune the model to achieve **shorter** trace but still leads to **optimal** plan!  
(Reinforcement Learning task)

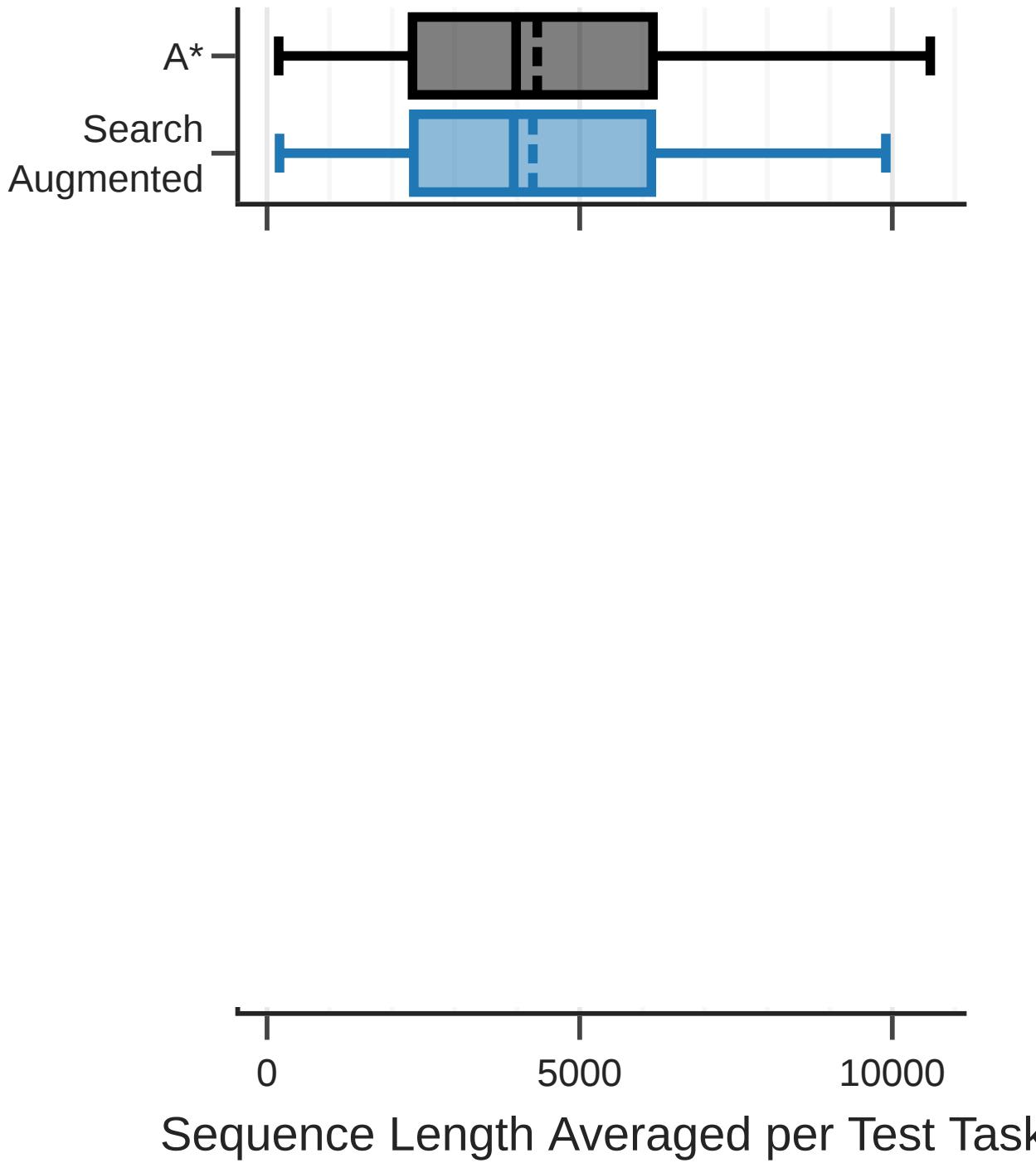


Search-augmented Models



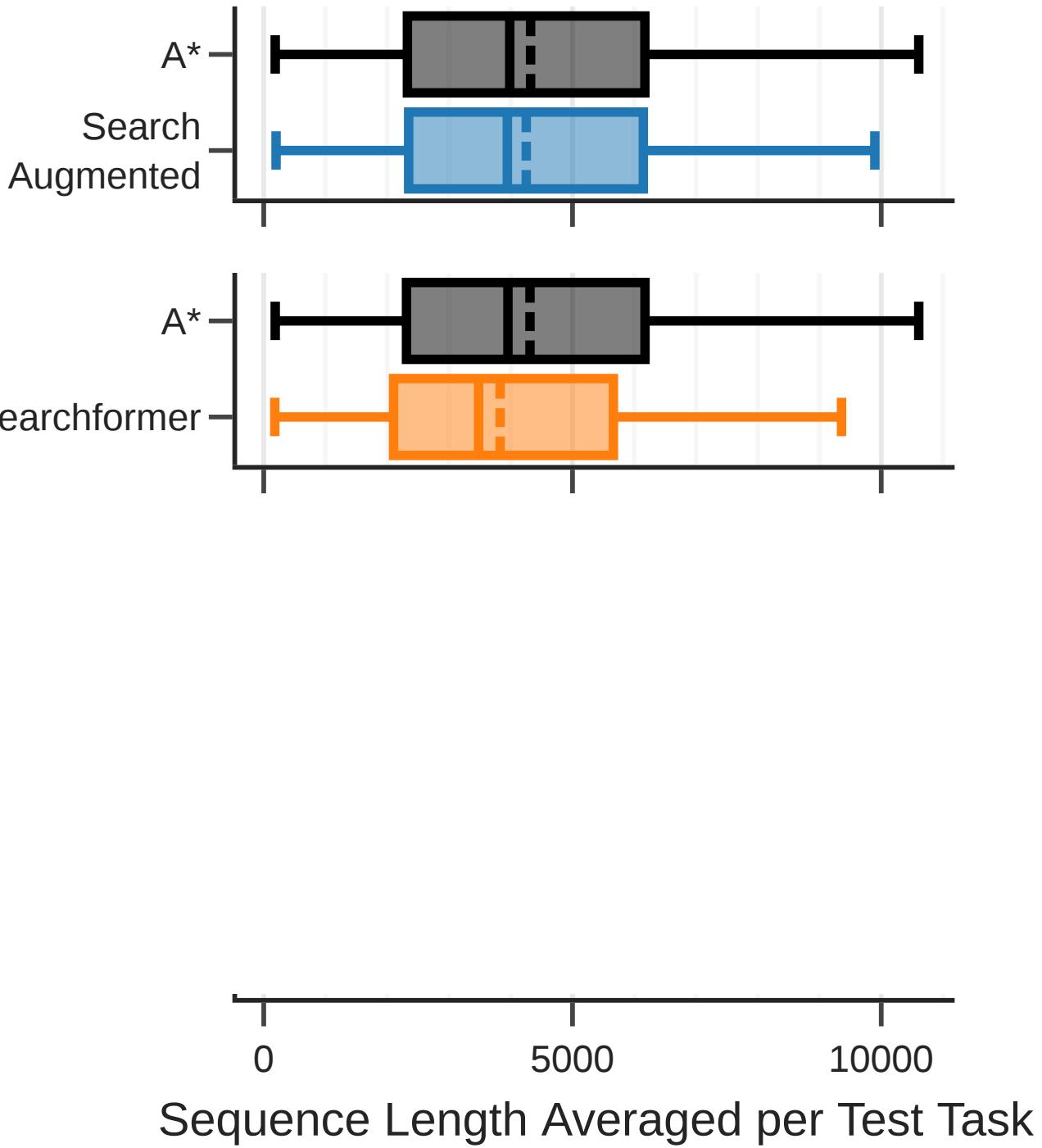
Searchformer

# Beyond A\*: Improving search dynamics via bootstrapping

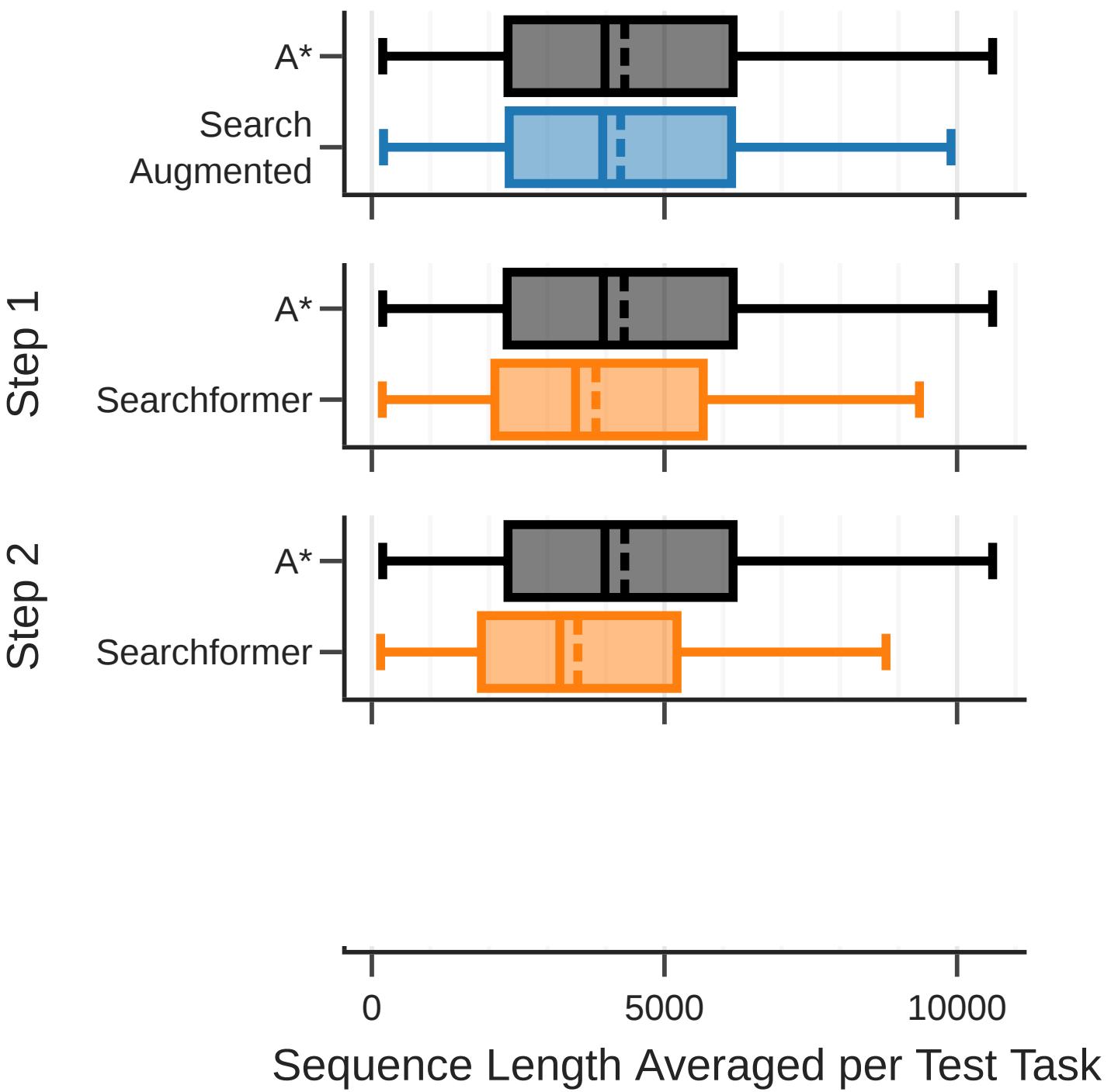


# Beyond A\*: Improving search dynamics via bootstrapping

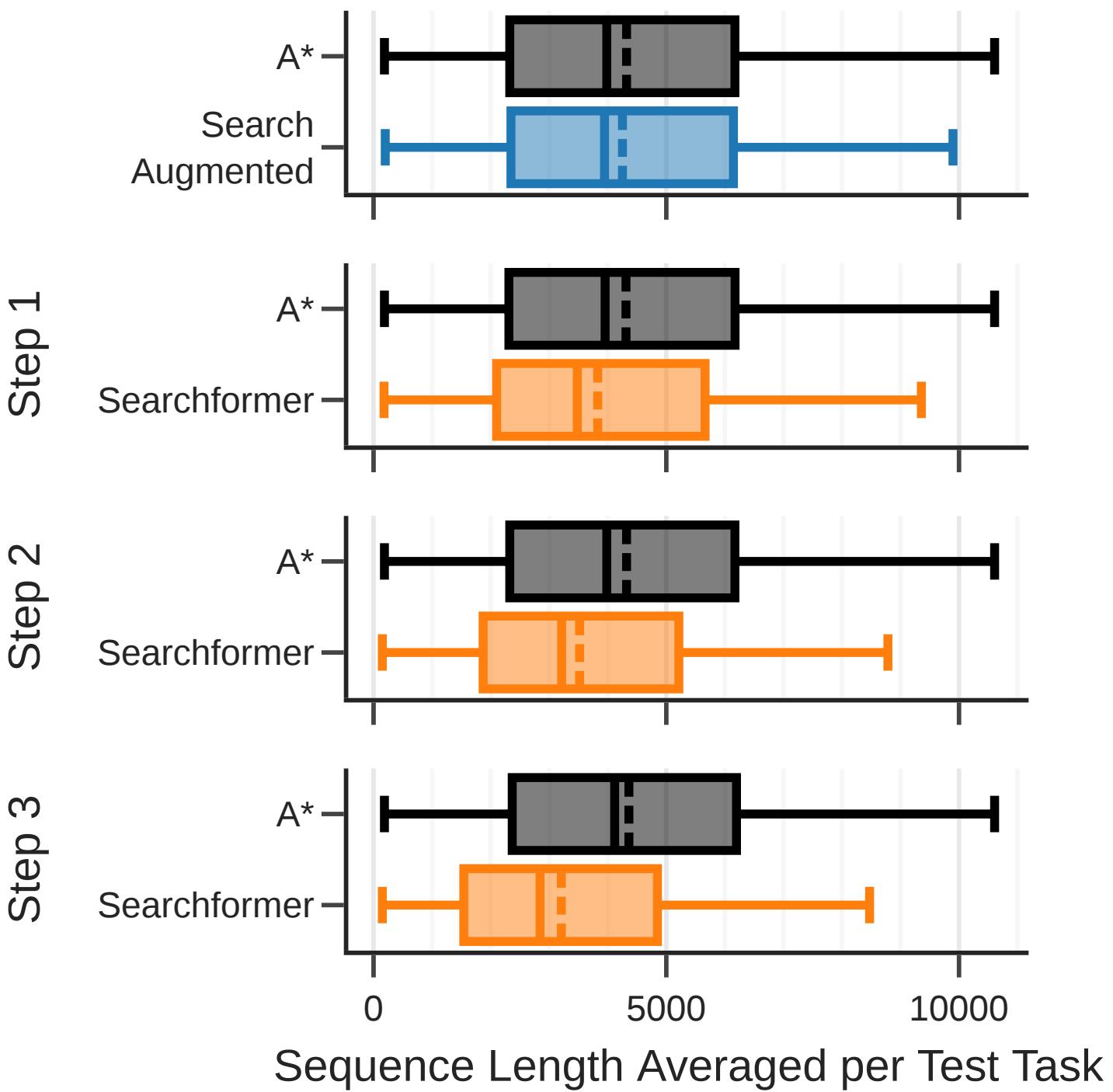
Step 1



# Beyond A\*: Improving search dynamics via bootstrapping



# Beyond A\*: Improving search dynamics via bootstrapping



# Improving search dynamics via bootstrapping

Params.	Model	Solved (%)	Optimal (%)
45M	Solution only	90.3 ±1.0	86.8 ±0.3
	Search augmented	92.5 ±1.0	90.8 ±1.6
	Searchformer, step 1	95.5 ±1.0	93.5 ±1.0
	Searchformer, step 2	96.0 ±0.5	93.4 ±0.6
	Searchformer, step 3	95.5 ±0.8	93.7 ±1.6
175M	Solution only	95.7 ±0.2	90.0 ±0.8
	Search augmented	95.2 ±0.9	93.2 ±1.0
757M	Solution only	96.5 ±0.1	92.2 ±1.2

# Improving search dynamics via bootstrapping

Params.	Model	Solved (%)	Optimal (%)	
45M	Solution only	90.3 ±1.0	86.8 ±0.3	Fine-tuning improves performance initially.
	Search augmented	92.5 ±1.0	90.8 ±1.6	
	Searchformer, step 1	95.5 ±1.0	93.5 ±1.0	
	Searchformer, step 2	96.0 ±0.5	93.4 ±0.6	
	Searchformer, step 3	95.5 ±0.8	93.7 ±1.6	
175M	Solution only	95.7 ±0.2	90.0 ±0.8	
	Search augmented	95.2 ±0.9	93.2 ±1.0	
757M	Solution only	96.5 ±0.1	92.2 ±1.2	

# Improving search dynamics via bootstrapping

Params.	Model	Solved (%)	Optimal (%)
45M	Solution only	90.3 ±1.0	86.8 ±0.3
	Search augmented	92.5 ±1.0	90.8 ±1.6
	Searchformer, step 1	95.5 ±1.0	93.5 ±1.0
	Searchformer, step 2	96.0 ±0.5	93.4 ±0.6
175M	Searchformer, step 3	95.5 ±0.8	93.7 ±1.6
	Solution only	95.7 ±0.2	90.0 ±0.8
	Search augmented	95.2 ±0.9	93.2 ±1.0
757M	Solution only	96.5 ±0.1	92.2 ±1.2

Searchformer  
outperforms largest  
solution-only model.

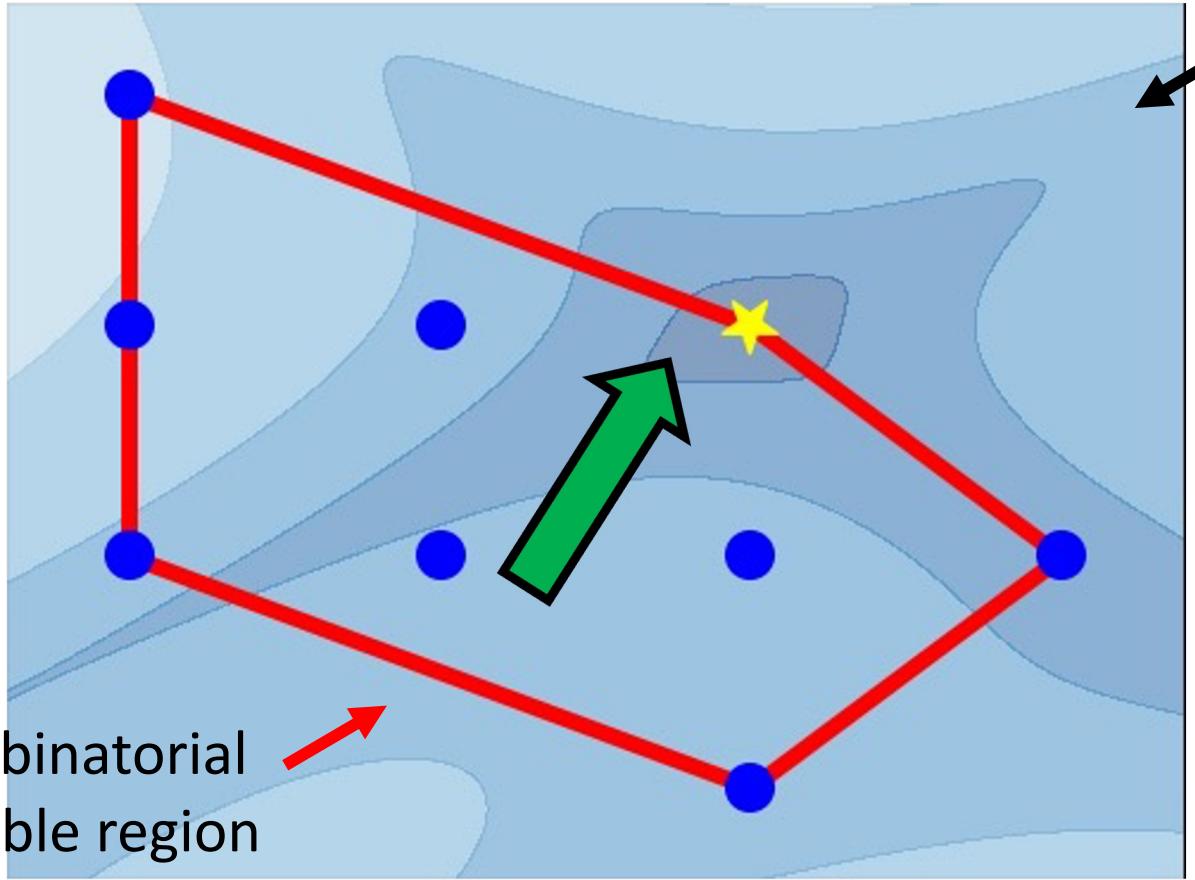
# Improving search dynamics via bootstrapping

Params.	Model	ILR-on-solved	ILR-on-optimal
45M	Solution only	—	—
	Search augmented	$0.908 \pm 0.020$	$0.919 \pm 0.019$
	Searchformer, step 1	$1.054 \pm 0.025$	$1.062 \pm 0.015$
	Searchformer, step 2	$1.158 \pm 0.025$	$1.181 \pm 0.012$
175M	Searchformer, step 3	$1.292 \pm 0.044$	$1.343 \pm 0.067$
	Solution only	—	—
757M	Search augmented	$0.925 \pm 0.010$	$0.933 \pm 0.011$
	Solution only	—	—

Repeated bootstrapping increases the  
*Improved Length Ratio (ILR)*

# Part II: Train Deep Models with Solvers

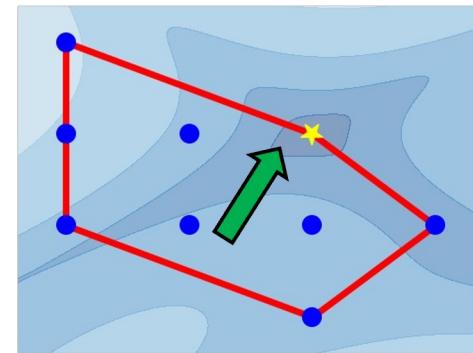
# Nonlinear objective with combinatorial constraints



Nonlinear + differentiable  
objective  $f(\mathbf{x})$

- Real-world domains:
  - Computer system planning
  - Designing photonic devices
  - Throughput optimization
  - Antenna design
  - Energy grid

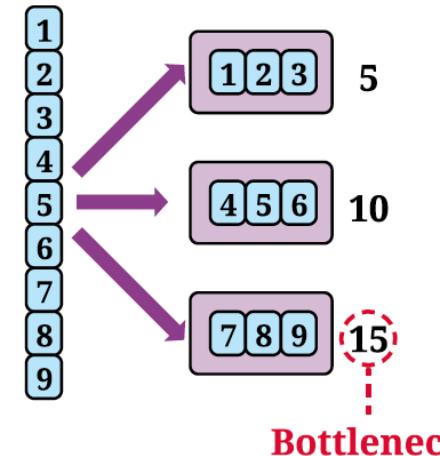
# Example: Embedding Table Placement



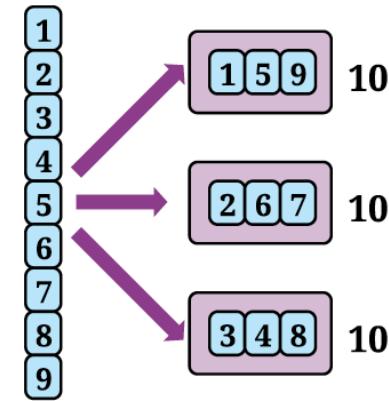
Given:

- $k$  tables
- $n$  identical devices
- Table  $i$  has memory requirement  $m_i$
- Device  $j$  has memory capacity  $M_j$

**Naive Sharding**



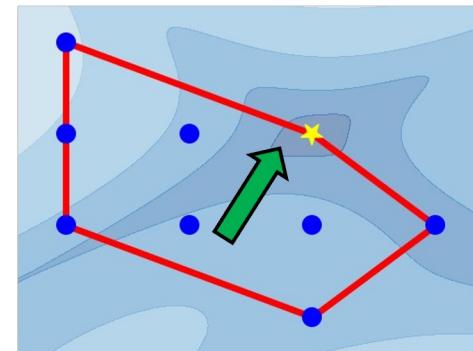
**Balanced Sharding**



Find

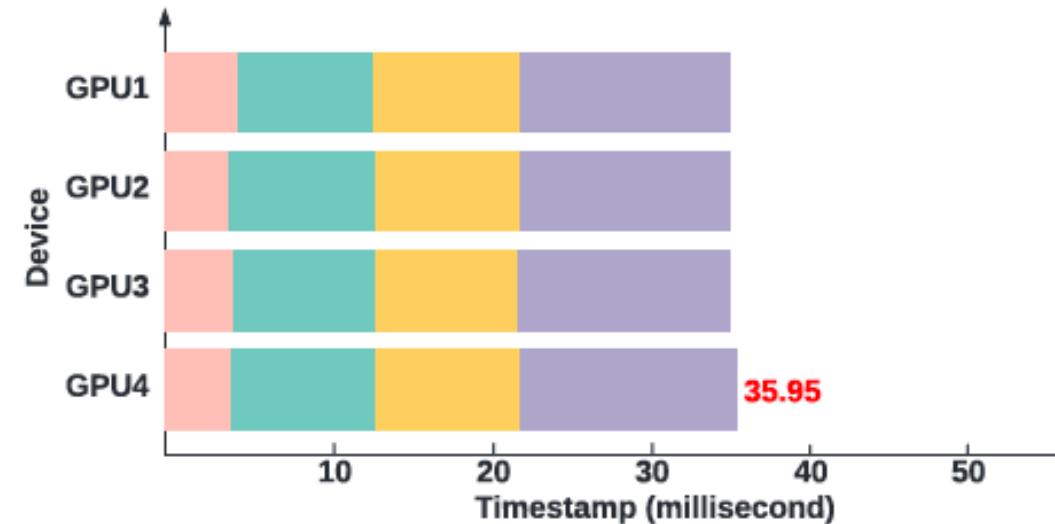
- Allocation of tables to devices observing device memory limits
- Minimize latency which is **estimated by a neural network** (capturing nonlinear interactions)

# Example: Embedding Table Placement



Given:

- $k$  tables
- $n$  identical devices
- Table  $i$  has memory requirement  $m_i$
- Device  $j$  has memory capacity  $M_j$



Formulation

$$\text{Min}_x \textcolor{green}{L}(\{x_{ij}\}) \quad \text{s.t.} \quad \sum_i x_{ij} m_i \leq M_j, \quad \sum_j x_{ij} = 1, \quad x_{ij} \in \{0,1\}$$

$\textcolor{green}{L}$  is nonlinear due to system issues (e.g., batching, communication, etc)

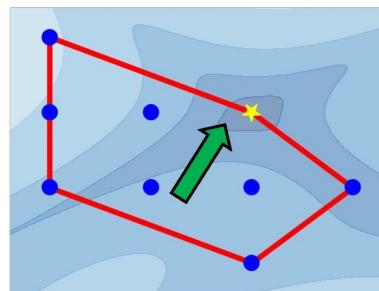
# Idea: Find a Linear Surrogate

- Learn a MILP objective whose optimal solution  $x^*$  solves the nonlinear problem

Originally

Nonlinear optimization with  
combinatorial constraints

$$\begin{aligned} \min_x f(x; y) \\ \text{s.t } x \in \Omega = & \quad \text{combinatorial} \\ & \quad \text{constraints} \end{aligned}$$



Now

Surrogate optimization

$$\begin{aligned} x^*(y) = \operatorname{argmin}_x c(y)^T x \\ \text{s.t } x \in \Omega \end{aligned}$$

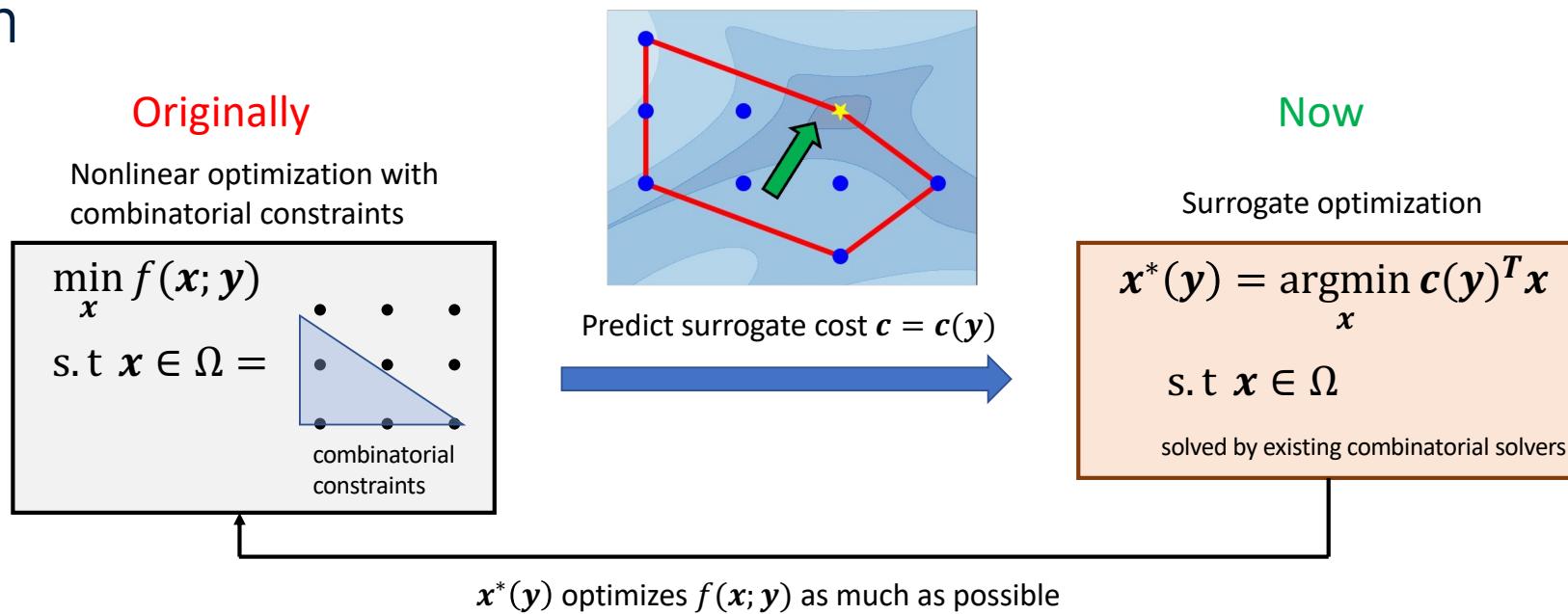
solved by existing combinatorial solvers

Predict surrogate cost  $c = c(y)$

$x^*(y)$  optimizes  $f(x; y)$  as much as possible

# Idea: Find a Linear Surrogate

- Learn a MILP objective whose optimal solution  $x^*$  solves the nonlinear problem

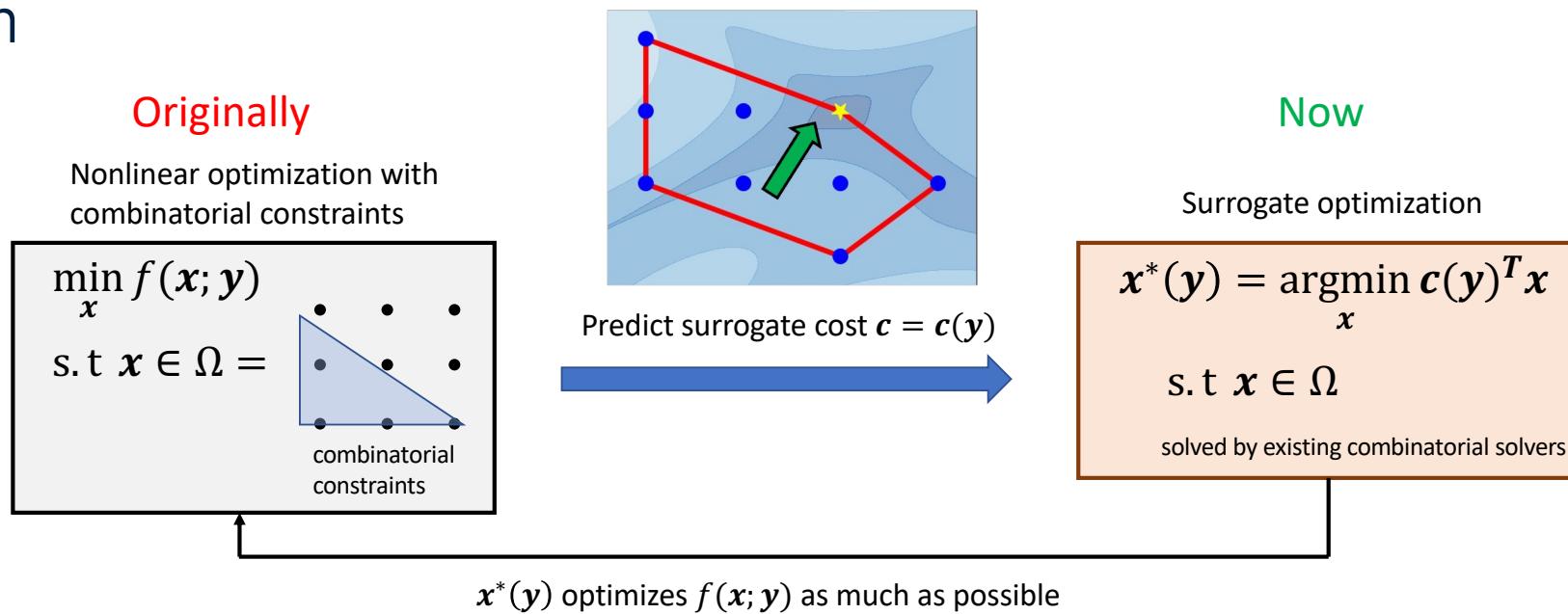


**Challenge:** how to find the right objective?

[A. Ferber et al, *SurCo: Learning Linear Surrogates For Combinatorial Nonlinear Optimization Problems*, ICML'23  
and **outstanding paper** in SODS workshop]

# Idea: Find a Linear Surrogate

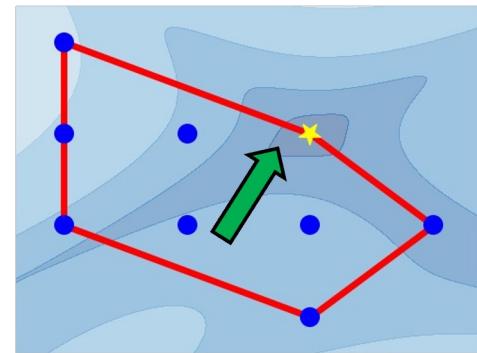
- Learn a MILP objective whose optimal solution  $x^*$  solves the nonlinear problem



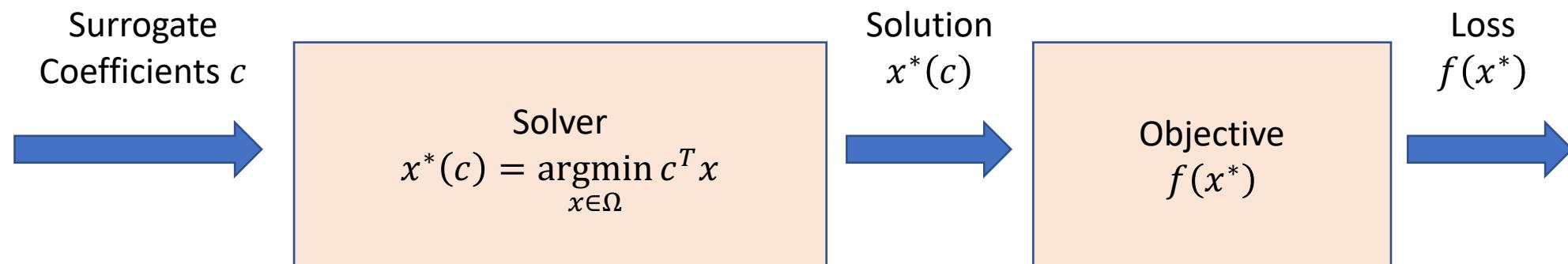
**Proposal:** gradient-based optimization

[A. Ferber et al, *SurCo: Learning Linear Surrogates For Combinatorial Nonlinear Optimization Problems*, ICML'23 and **outstanding paper** in SODS workshop]

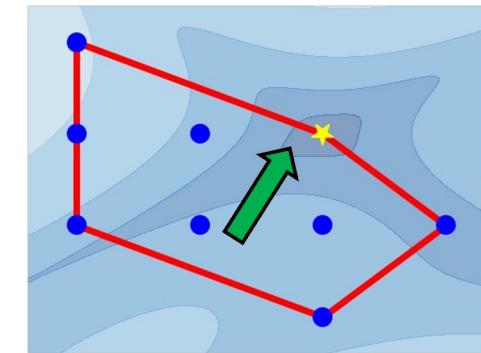
# Proposal: surrogate learning



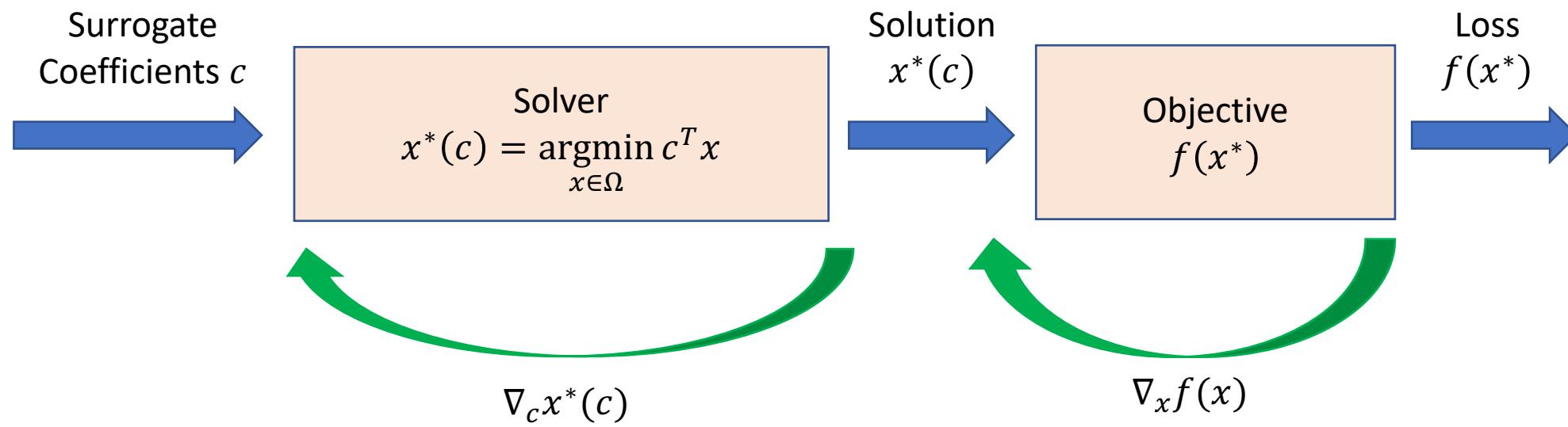
- Use surrogate MILP to solve original problem
- Find linear coefficients  $c$  such that  $\underset{x \in \Omega}{\operatorname{argmin}} f(x) = \underset{x \in \Omega}{\operatorname{argmin}} c^T x$



# SurCo-zero: gradient-based optimization



- **Iterative** solver based on linear surrogate guided by **gradient updates**
- Update linear coefficients  $c$  such that  $x^*(c)$  improves objective  $f(x^*(c))$



Recent work on differentiable optimization

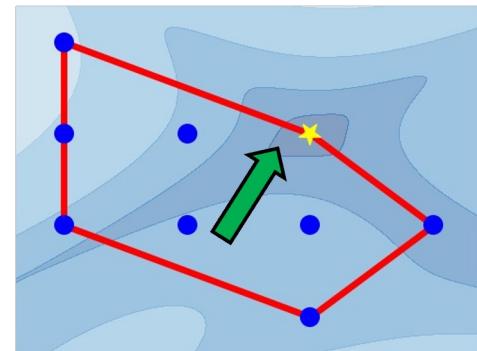
Differentiation of blackbox optimizers

CVXPYLayers

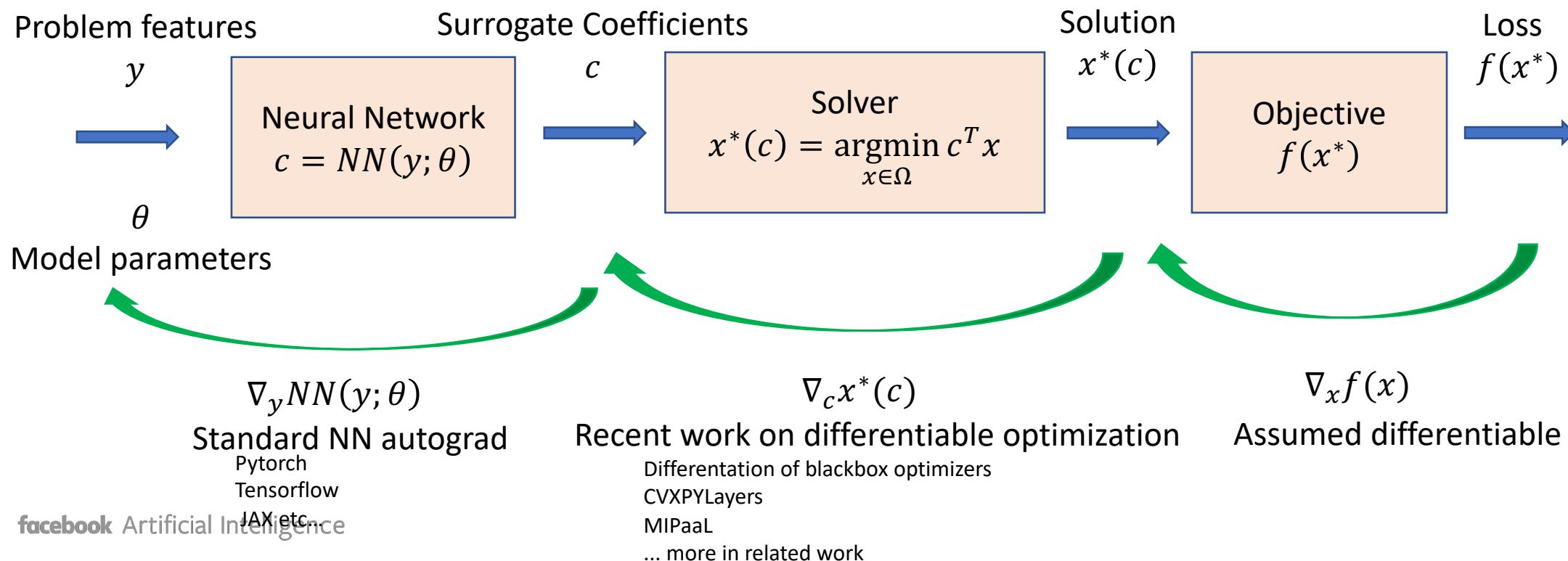
MIPaaL

Assumed differentiable

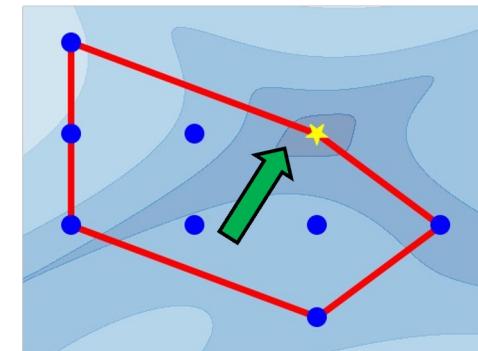
# SurCo-prior: distributional learning



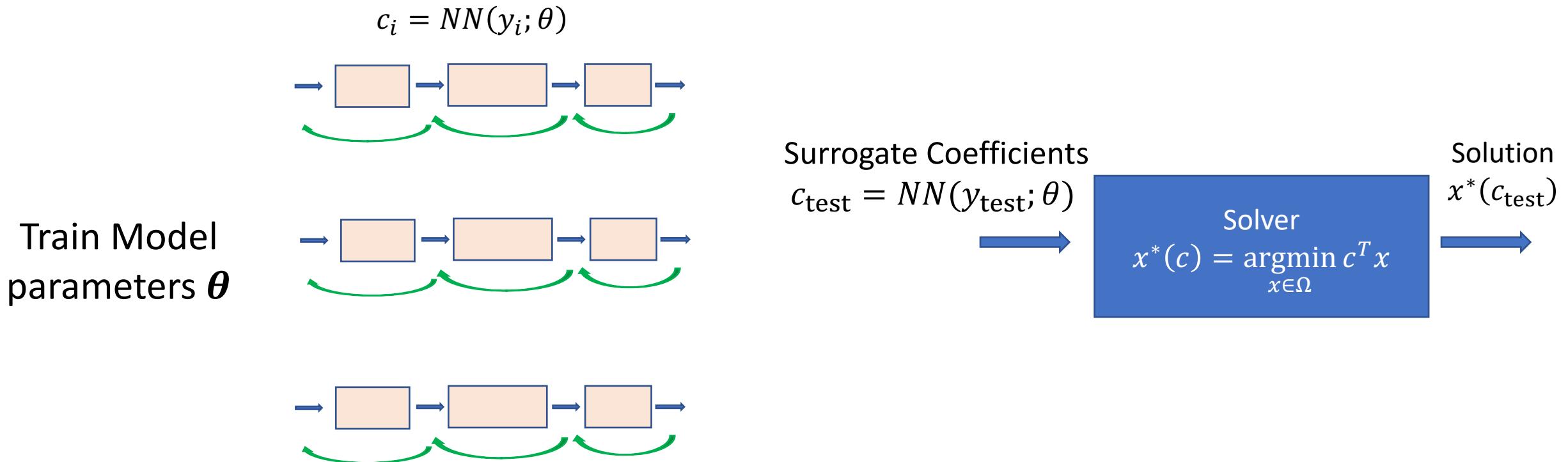
- One pass solver based on model **learned offline**
- Use neural model based on **problem features** to predict linear coefficients



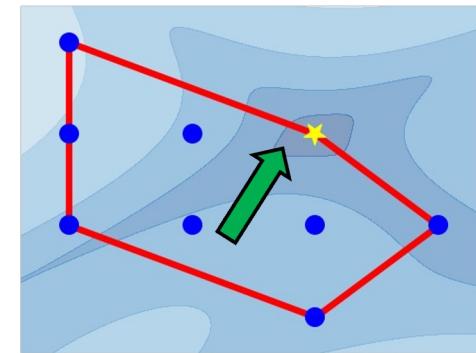
# SurCo-prior: distributional learning



- Update neural network parameters from training dataset



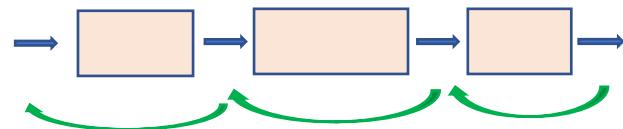
# SurCo-hybrid: fine-tuning from trained model



Update neural network parameters from training dataset

**Fine-tune surrogate on-the-fly**

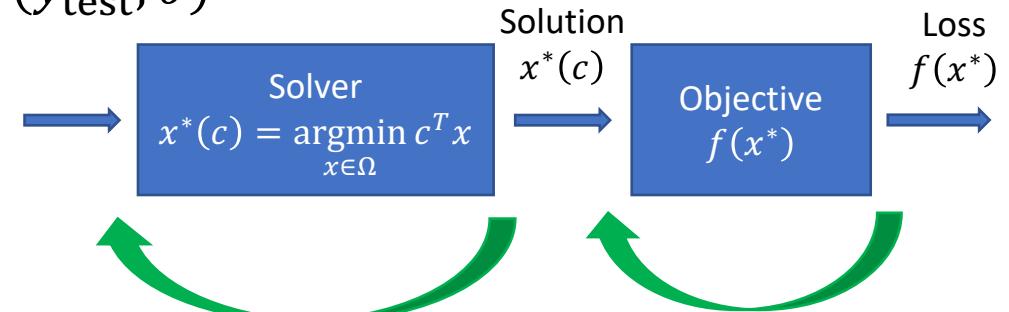
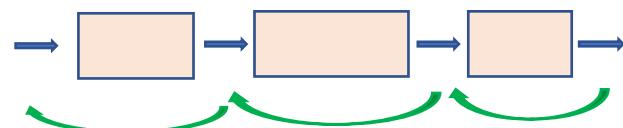
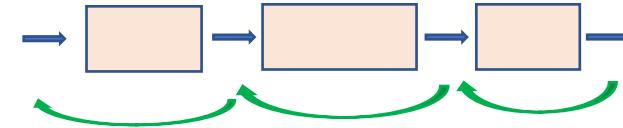
$$c_i = \text{NN}(y_i; \theta)$$



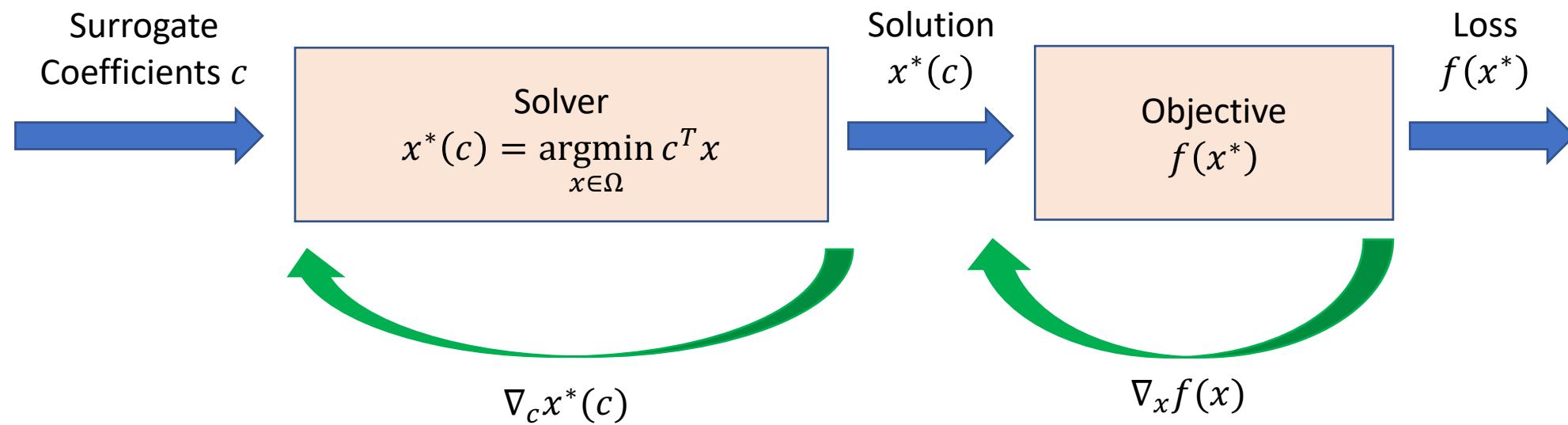
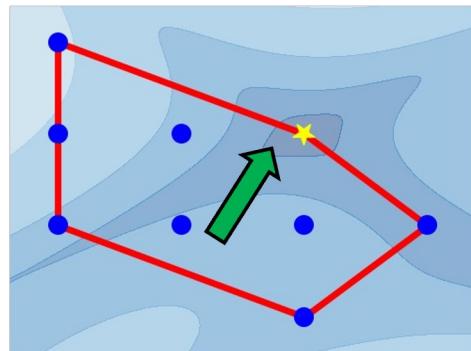
Initial Surrogate Coefficients

$$c_0 = \text{NN}(y_{\text{test}}; \theta)$$

Train Model parameters  $\theta$

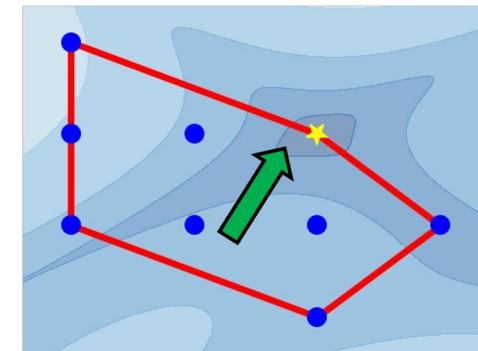


# SurCo-zero

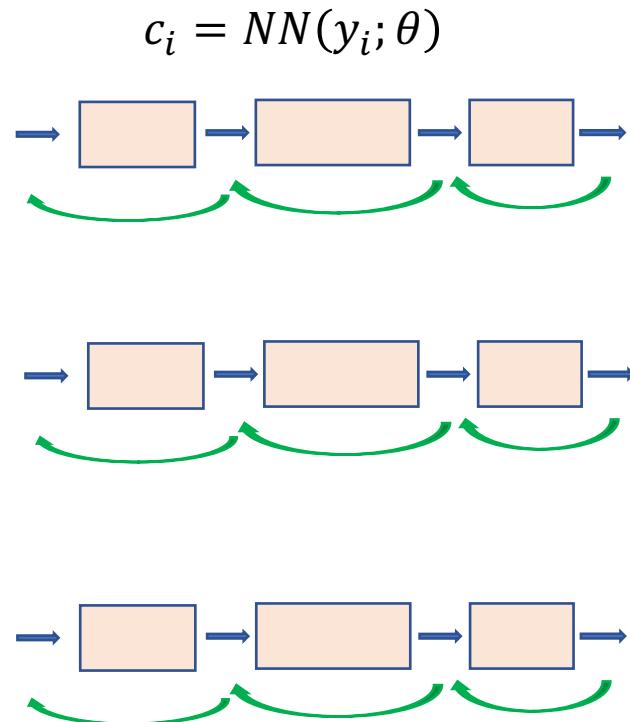


No offline training data, just solve a single problem instance on-the-fly

# SurCo-prior



Train Model  
parameters  $\theta$



Surrogate Coefficients

$$c_{\text{test}} = NN(y_{\text{test}}; \theta)$$



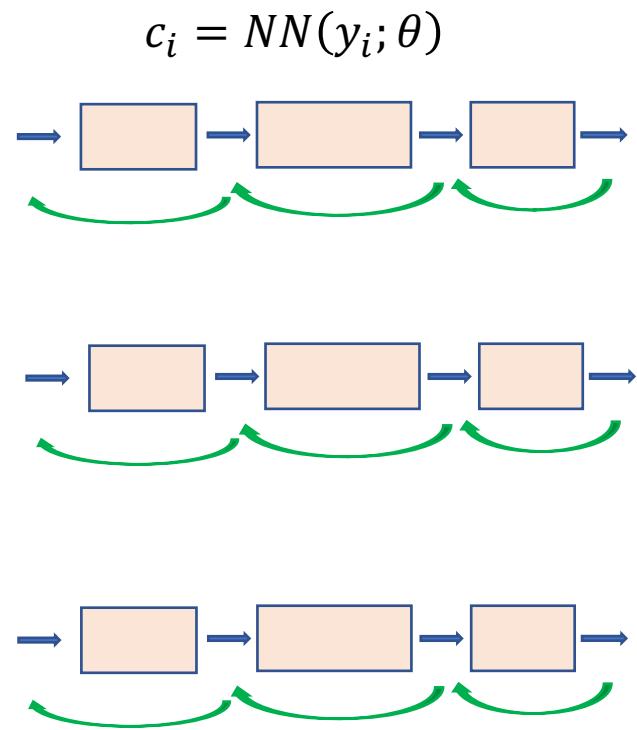
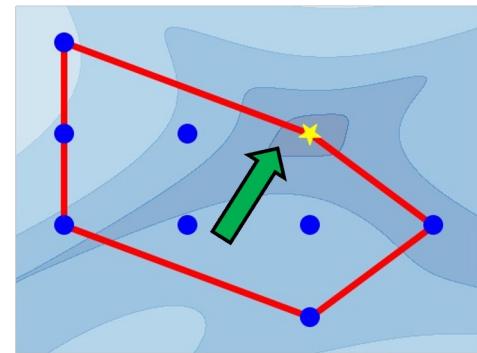
Solver  
 $x^*(c) = \operatorname{argmin}_{x \in \Omega} c^T x$

Solution  
 $x^*(c_{\text{test}})$

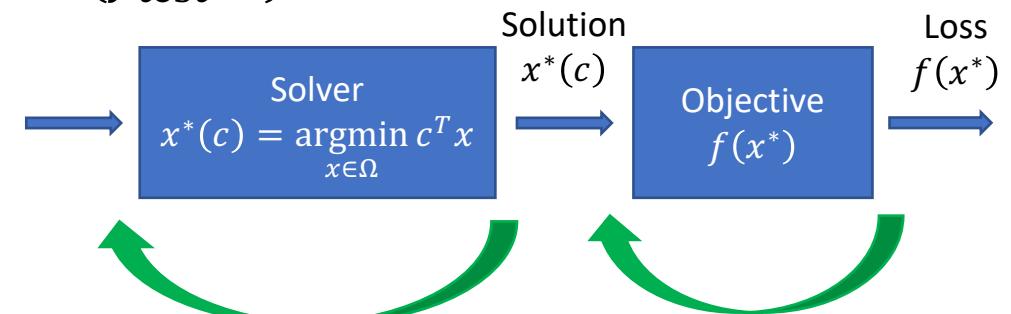


Uses offline training data to quickly solve problems at test time with just one solver call

# SurCo-hybrid



Initial Surrogate Coefficients  
 $c_0 = NN(y_{\text{test}}; \theta)$



Offline train + on-the-fly fine-tuning the surrogate

# Embedding Table Sharding

Used in large-scale deep learning systems: recommendation systems, knowledge graph

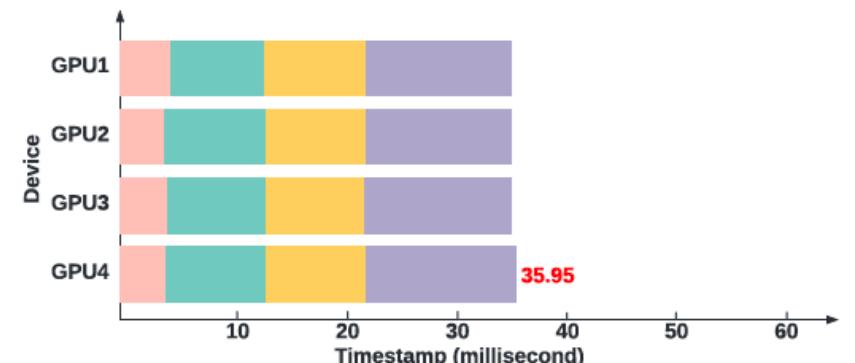
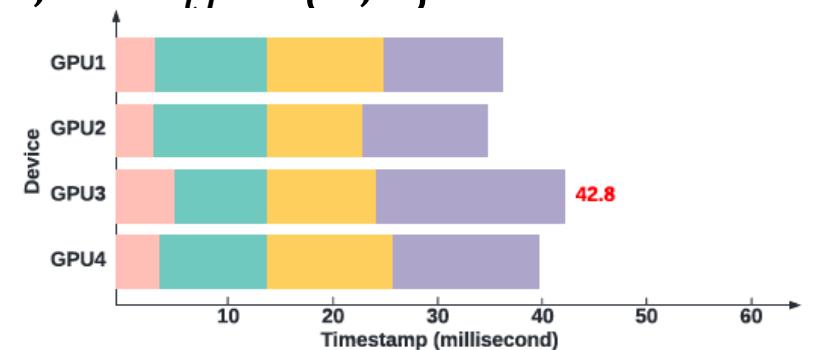
Place N “tables” (with known memory need  $m_i$ ) on K devices ( $x_{ij} = 1$ : table  $i$  assigned to device  $j$ )

$$\text{Min}_x L(\{x_{ij}\}) \quad \text{s.t. } \sum_i x_{ij} m_i \leq M_j, \quad \sum_j x_{ij} = 1, \quad x_{ii} \in \{0,1\}$$

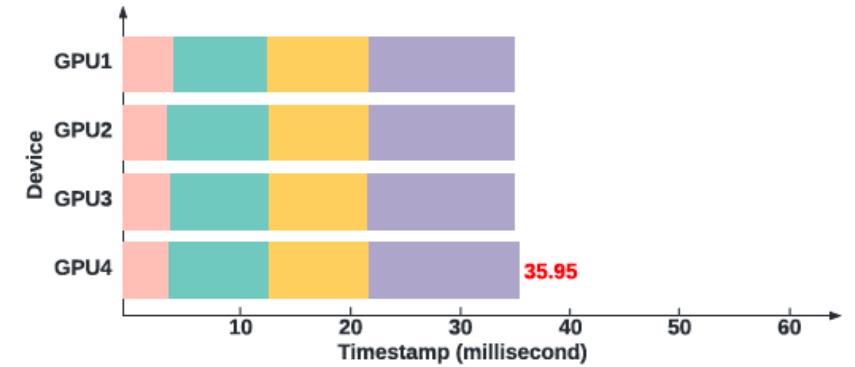
$L$  : Runtime bottleneck  $f(x)$  estimated by NN (longest-running device)

$L$  is nonlinear due to system issues  
(e.g., batching, communication, etc.)

$c(y; \theta)$  gives surrogate “per-table cost”  $c_{ij}$   
(and  $\sum_i c_{ij} x_{ij}$  is the surrogate latency objective)



# Embedding Table Sharding

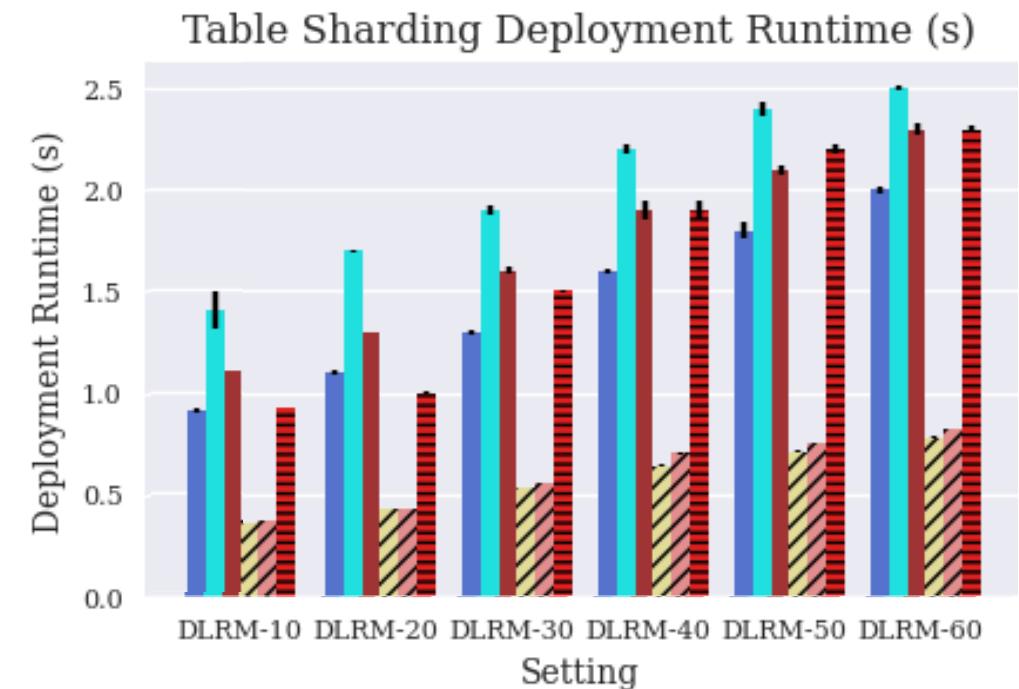
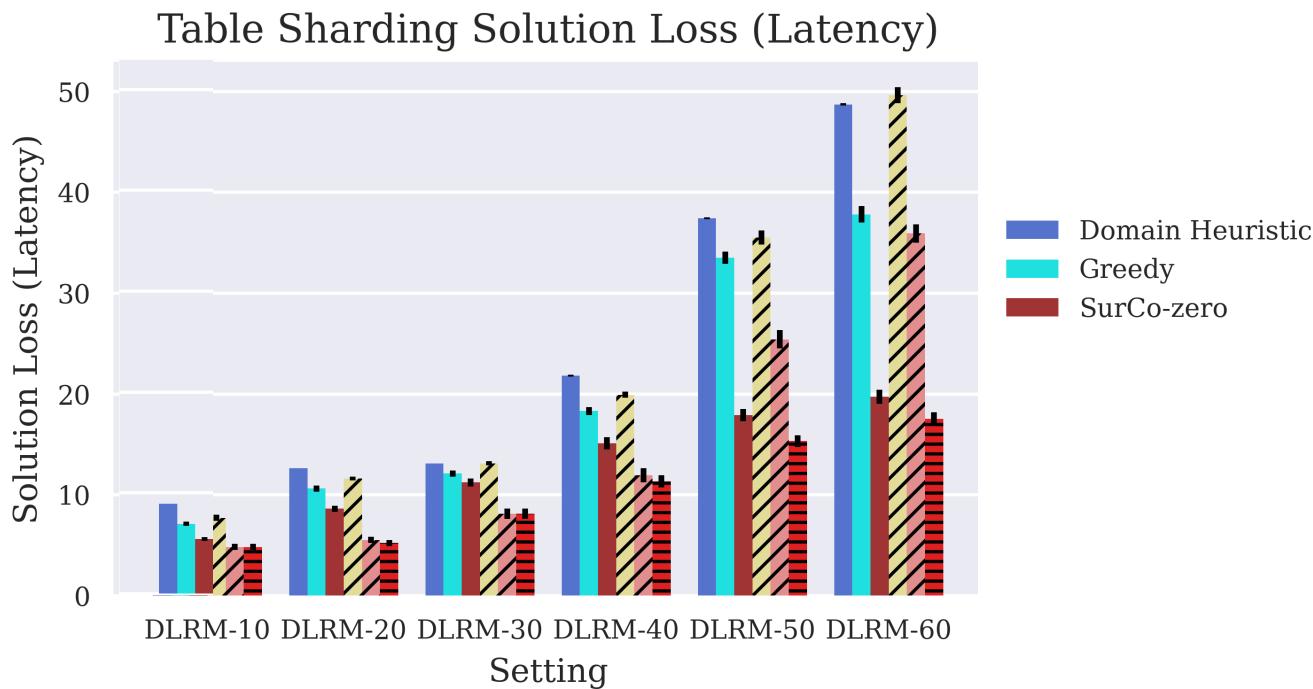


- Public Deep Learning Recommendation Model (DLRM dataset) placing between 10 to 60 tables on 4 GPUs
- Baseline: Greedy
- SoTA: RL approach Dreamshard<sup>1</sup>
- SurCo: Surrogate NN model learned via CVXPYLayers (differentiable LP Solver)

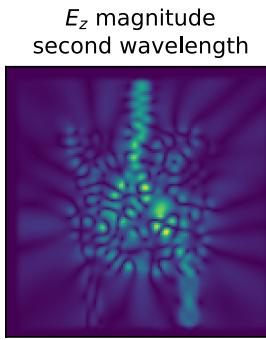
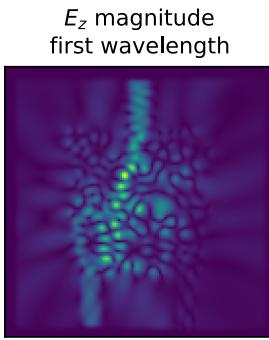
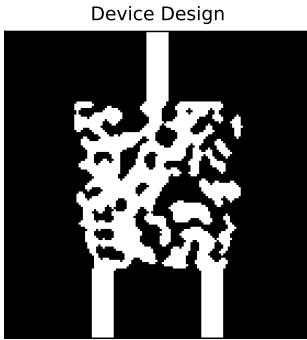
<sup>1</sup> Zha et al. NeurIPS 2022

Dataset: [https://github.com/facebookresearch/dlrm\\_datasets](https://github.com/facebookresearch/dlrm_datasets)

# Results – Table Sharding



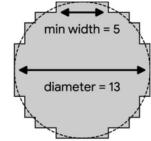
# Inverse Photonic Design



- Design physically-viable devices that take light waves and routes different wavelengths to correct locations

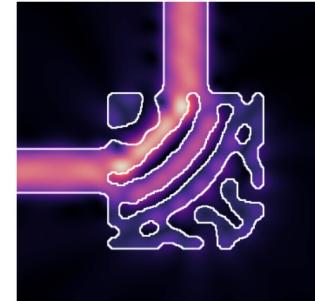
$$\mathcal{L}(S) = \left( \left\| \text{softplus} \left( g \frac{|S|^2 - |S_{\text{cutoff}}|^2}{\min(w_{\text{valid}})} \right) \right\|_2 \right)^2$$

- Device design misspecification loss  $f(x)$  computed by differentiable electromagnetic simulator
- Feasible solution: the design must be the union of brush pattern
  - $x = \text{binary\_opening}(x, \text{brush})$
  - $x = \sim \text{binary\_opening}(\sim x, \text{brush})$

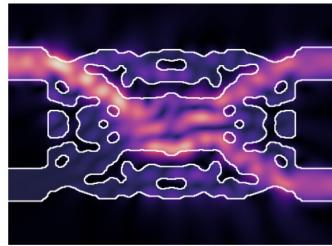


# Inverse Photonic Design

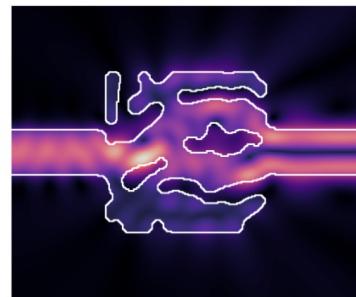
- Dataset: Ceviche Challenges<sup>1</sup>
- Most baselines don't work here due to combinatorial constraints
- SoTA: Brush-based algorithm <sup>1</sup>
- SurCo: Surrogate learned via blackbox differentiation<sup>2</sup> of brush solver



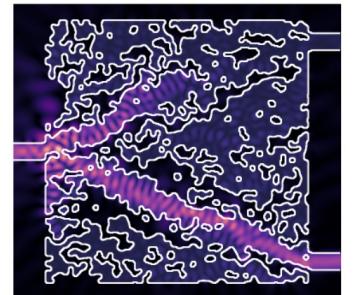
Waveguide bend



Beam splitter



Mode converter



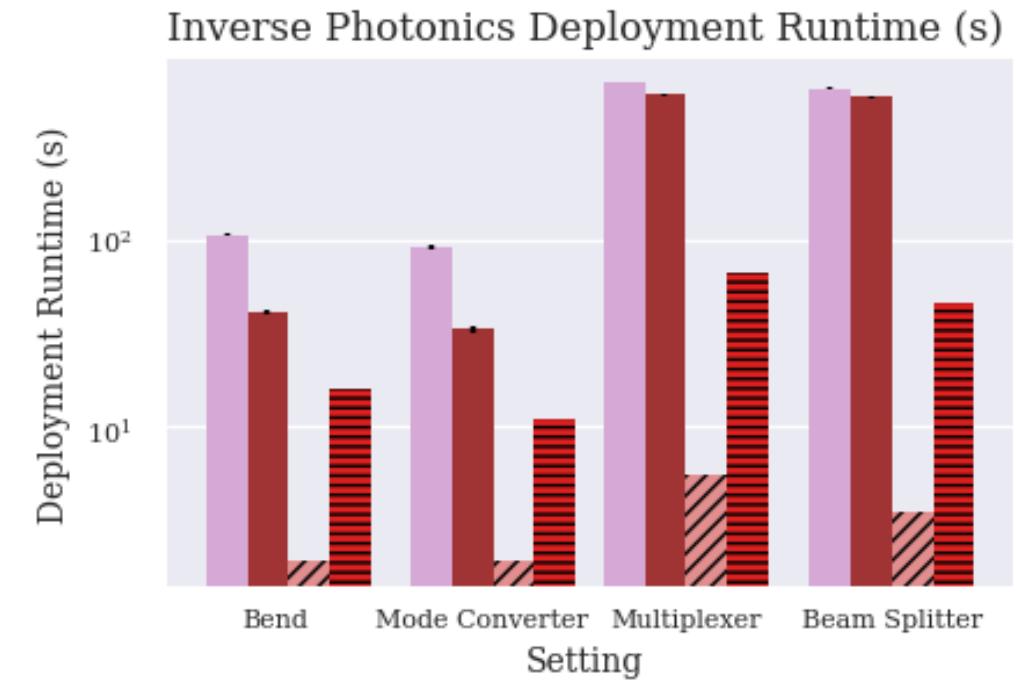
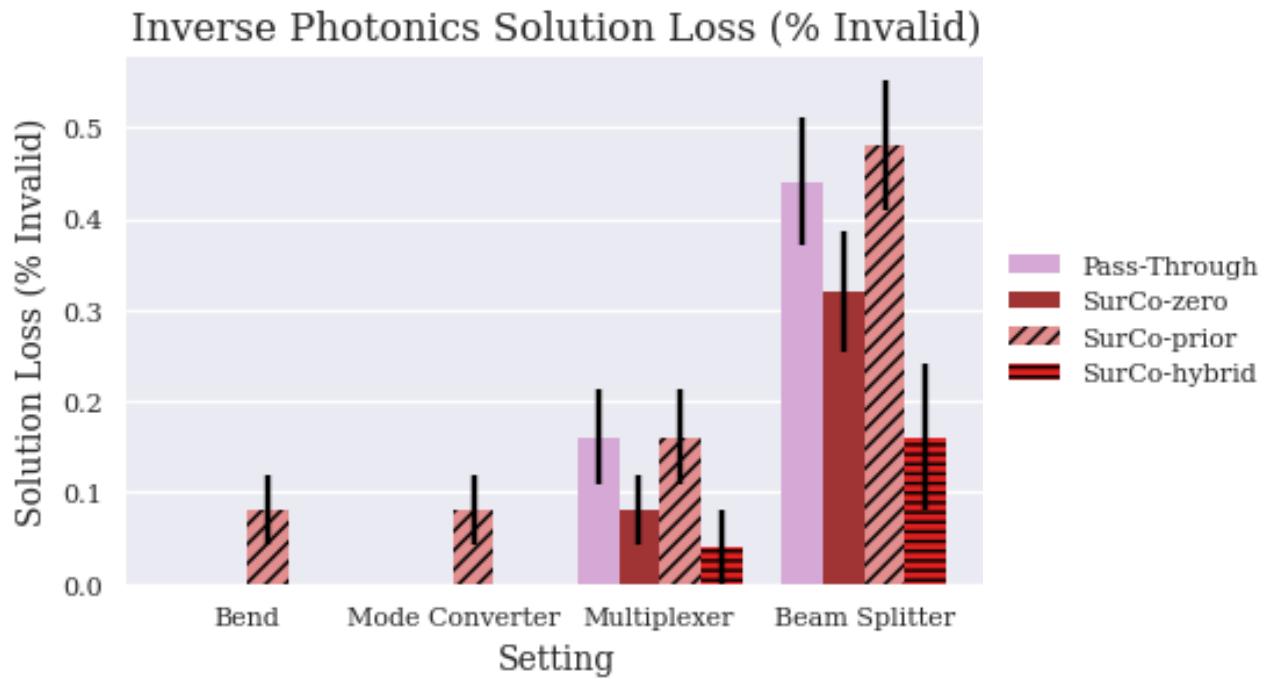
Wavelength division multiplexer

<sup>1</sup>Schubert et al. ACS Photonics 2022

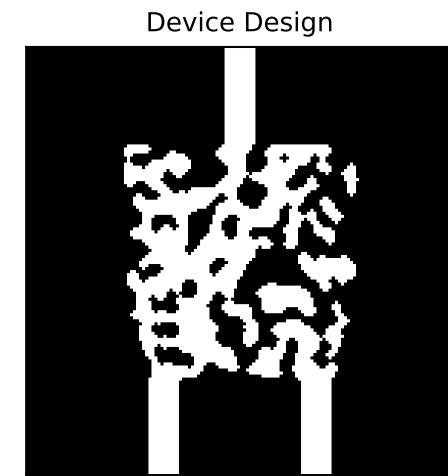
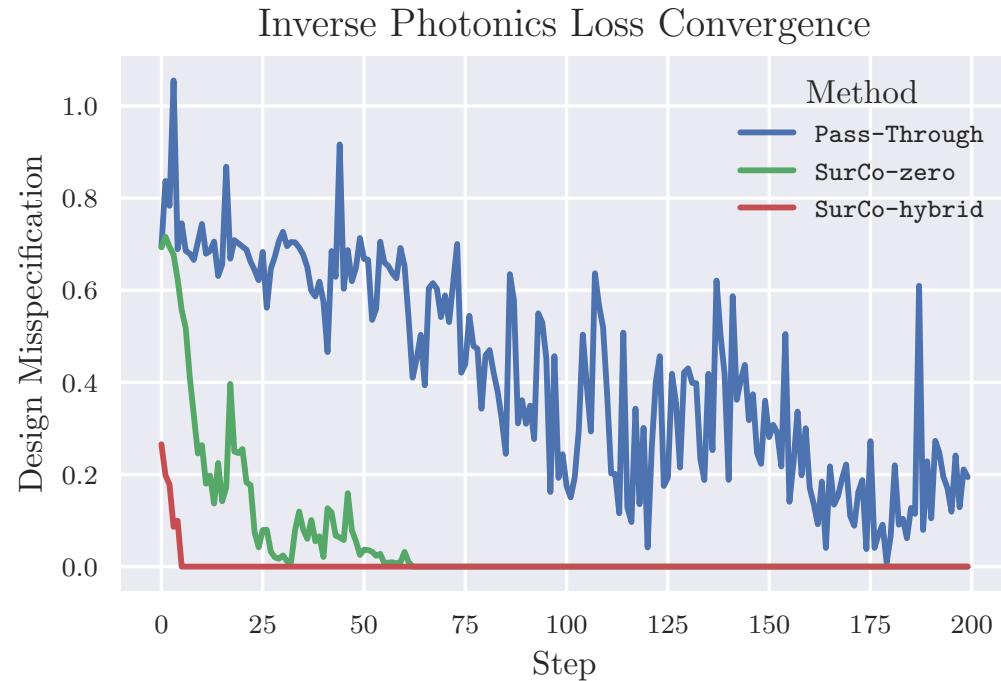
<sup>2</sup>Vlastelica et al. ICLR 2019

Dataset: <https://github.com/google/ceviche-challenges>

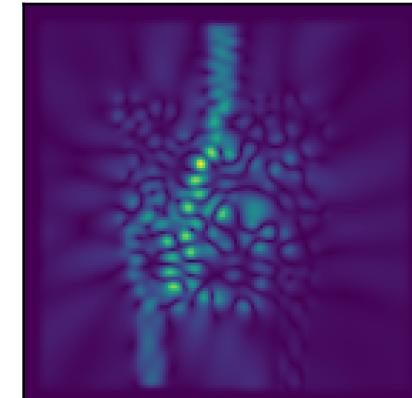
# Results – Inverse Photonics



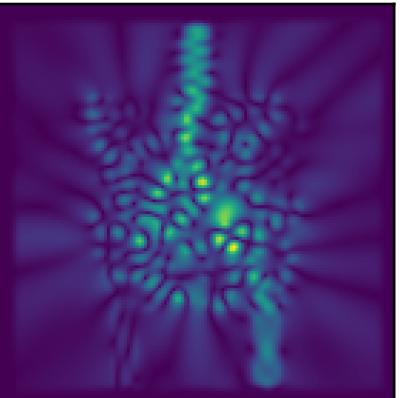
# Inverse photonics Convergence comparison + Solution example



$E_z$  magnitude  
first wavelength



$E_z$  magnitude  
second wavelength

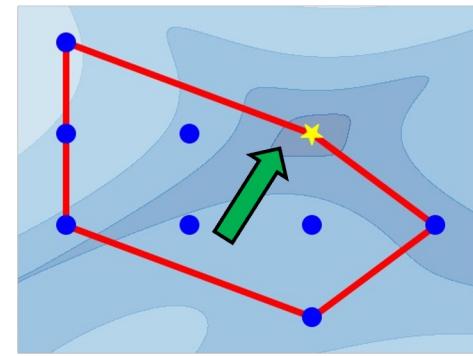


Wavelength division multiplexer

## Takeaways:

- SurCo-Zero finds loss-0 solutions quickly
- SurCo-Hybrid uses offline training data to get a head start

# Conclusion



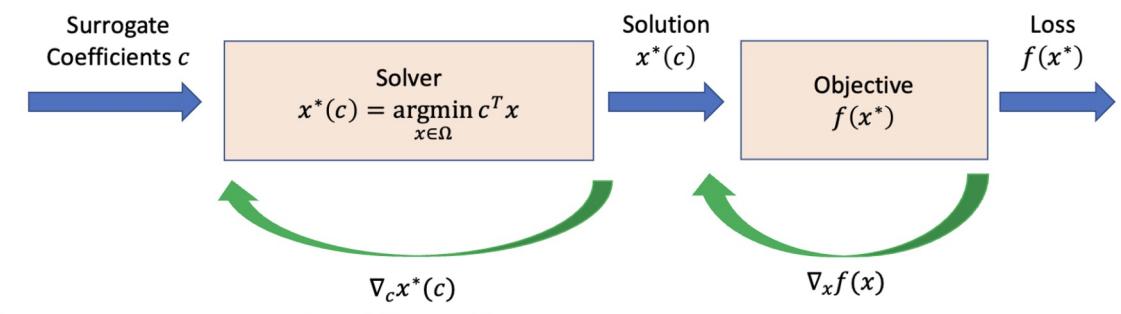
- Handle industrial applications with differentiable optimization
- High-quality solutions to combinatorial nonlinear optimization by finding linear surrogates
  - Sometimes we can find “easier” surrogate problems that solve much more difficult instances
- SurCo works in several data settings
  - Zero-shot vs Offline training
  - One step inference vs fine-tuning

# Limitation of SurCo

Recall **SurCo**: Update linear coefficients  $\mathbf{c}$  such that  $x^*(\mathbf{c})$  improves objective  $f(x^*(\mathbf{c}))$

$$\min_{\theta} \mathcal{L}(Y, Z) := \sum_{i=1}^N f(\mathbf{g}_{\theta}(\mathbf{y}_i); \mathbf{z}_i)$$

$$\mathbf{g}_{\theta}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \Omega} \mathbf{x}^\top \mathbf{c}_{\theta}(\mathbf{y})$$



- Requires  $\nabla_x f(x) \rightarrow$  Does not applicable with “black-box” functions
- Requires  $\nabla_c g_{\theta}(c) \rightarrow$  Solver is backpropagatable

# Using Surrogate Models $\mathcal{M}$

- **LANCER:** Learn a landscape surrogate  $\mathcal{M}$  that approximates  $f \circ g$  and minimize  $\mathcal{M}$  instead.
  - $f$  and  $g$  may not be differentiable, but  $\mathcal{M}$  is differentiable

$$\min_{\theta} \mathcal{L}(Y, Z) := \sum_{i=1}^N f(\mathbf{g}_{\theta}(\mathbf{y}_i); \mathbf{z}_i)$$
$$\mathbf{g}_{\theta}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \Omega} \mathbf{x}^\top \mathbf{c}_{\theta}(\mathbf{y})$$

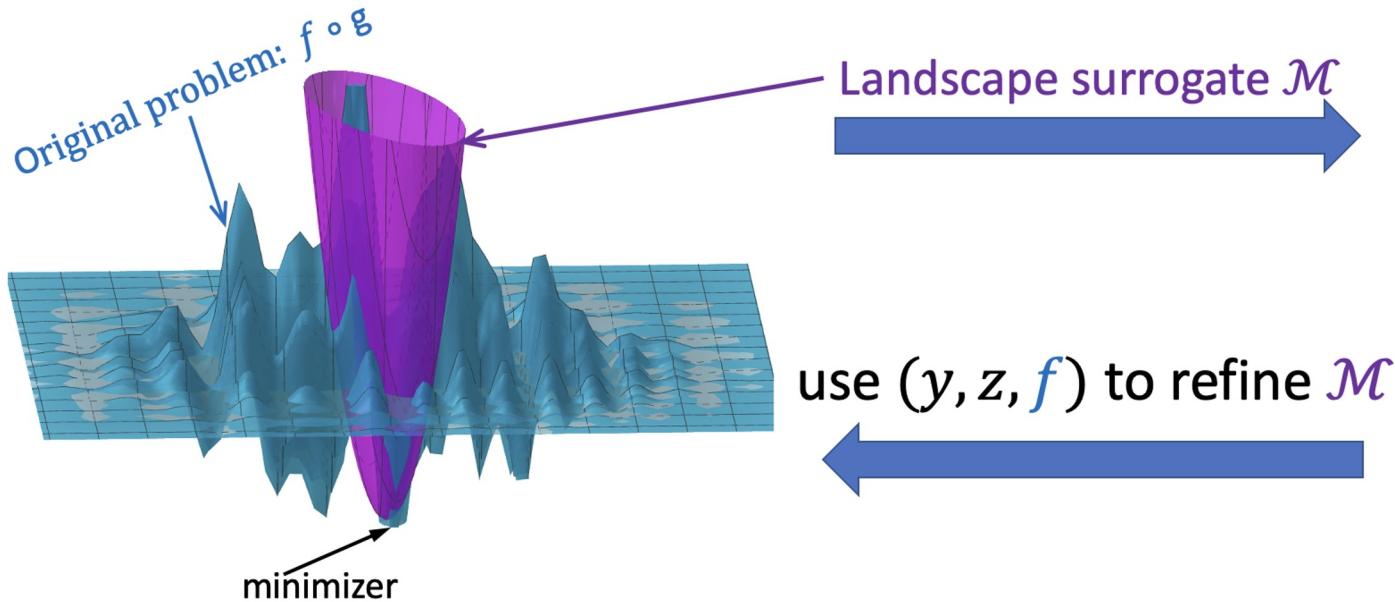


$$\min_{\theta} \mathcal{M}(Y, Z) := \sum_{i=1}^N \mathcal{M}(\mathbf{c}_{\theta}(\mathbf{y}_i); \mathbf{z}_i).$$

# How to learn surrogate loss $\mathcal{M}$ ?

$$\begin{aligned} & \min_{\mathbf{w}} \|\mathcal{M}_{\mathbf{w}}(\mathbf{c}_{\theta^*}(\mathbf{y}_i), \mathbf{z}_i) - f(\mathbf{g}_{\theta^*}(\mathbf{y}_i); \mathbf{z}_i)\| \\ \text{s.t. } & \theta^* \in \operatorname{argmin}_{\theta} \mathcal{M}_{\mathbf{w}}(\mathbf{c}_{\theta}(\mathbf{y}_i), \mathbf{z}_i). \end{aligned}$$

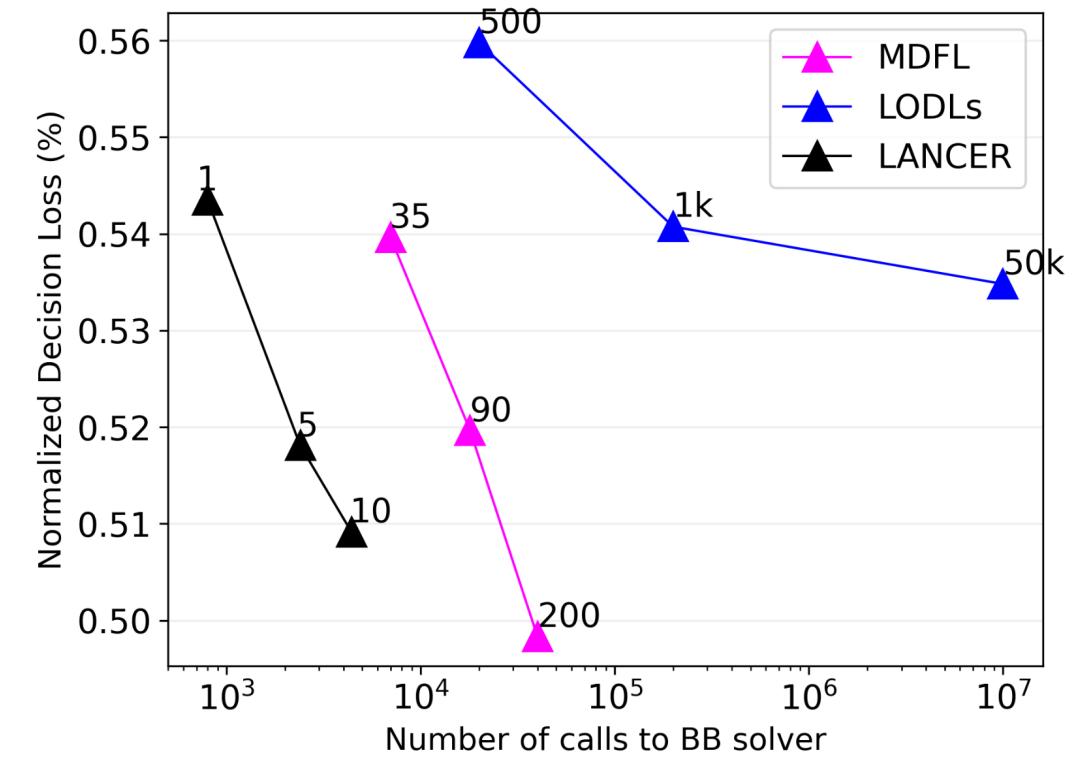
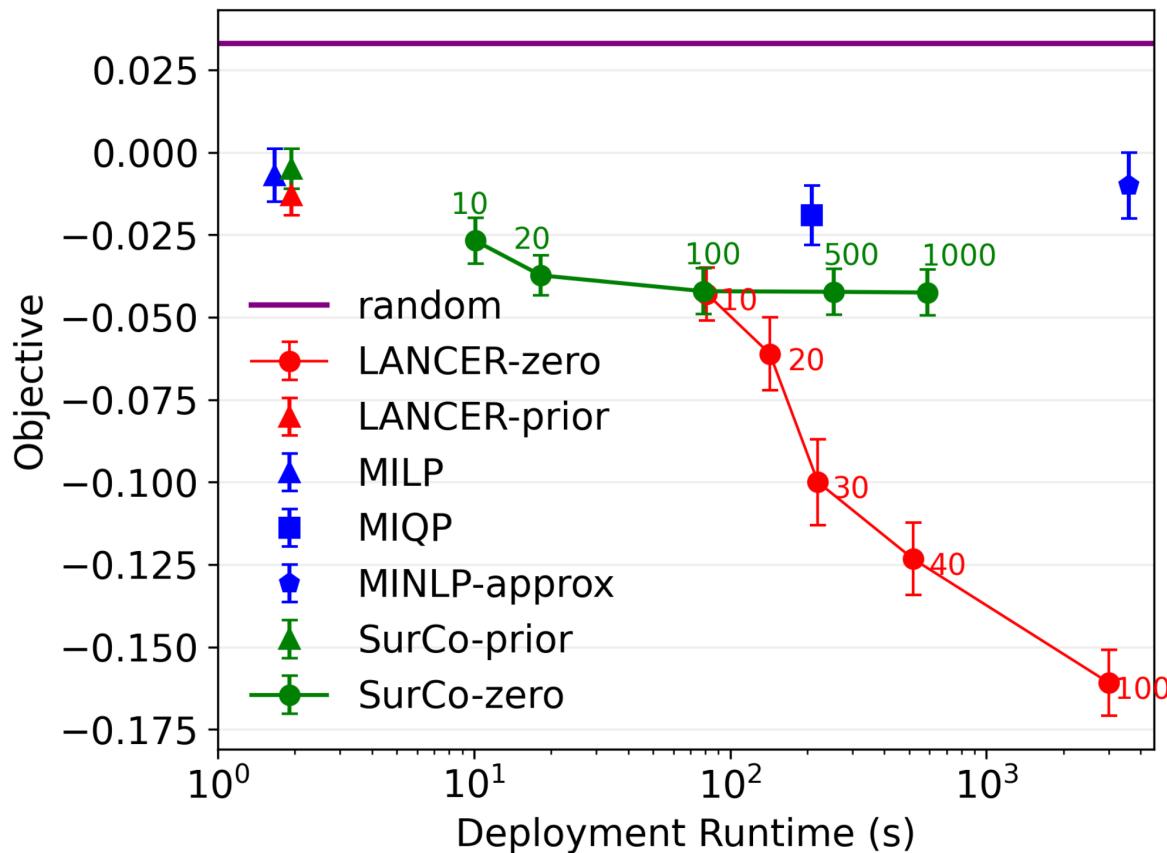
Good approximation only around optimal  $\theta$



learn  $c_{\theta}$ :  
$$\min_{\theta} \mathcal{M}(c_{\theta}(y), z)$$

# Experiments: Portfolio optimization

**Task:** Markowitz' portfolio selection problem but more complex objective and some variables are forced to be discrete. This is mixed-integer nonlinear program (MINLP).  
**Dataset:** Historical data on market prices from QuandlWIKI [2].



# Generating Diverse Solutions

For design problems, we want to have **diverse** solutions that optimize the **nonlinear** objective with **combinatorial** constraints.

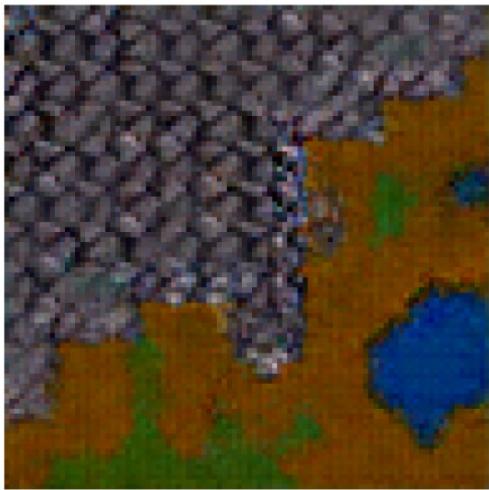
$$\min_{\theta} \mathcal{L}(\mathcal{X}) + \gamma \sum_j \mathcal{D}(\mathbf{x}_j) \quad \text{s.t. } \mathbf{x}_j = \mathbf{g}(G_{\theta}(\epsilon_j))$$

Group loss                                      Individual loss                              Combinatorial solver                              Generative models in the latent space

Where  $\mathcal{X} = \{\mathbf{x}_j\}$  are a set of solutions.

# Example Tasks

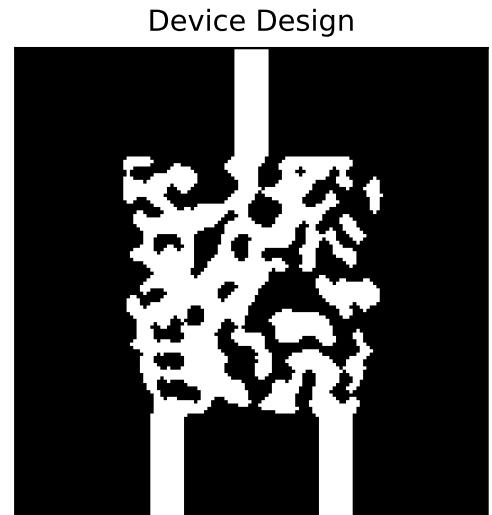
Experiment	Sol $x$	Feas $\Omega$	Group Loss $\mathcal{L}$	Ind. Loss $\mathcal{D}$	Latent $c$	Solver $g$
Game Design	Game Level	Playability	WGAN (comb)	–	Soft sol.	Gurobi (ILP)
Path Planning	Min Path	Path	WGAN	Min path (comb)	RGB map	Gurobi (LP)
Photonic Device	0/1 Grid	Manufacturing	VQVAE (comb)	Maxwell's sim. (comb)	Soft sol.	Domain spec.



Warcraft map generation



Game level design



Inverse photonics

# Results on Game Level Design

GAN + MILP



GenCo + fixed Adv



GenCo + upd Adv



# Numerical Performance

## Game Level Design

Approach	% Unique ↑	Density ↑	Coverage ↑	GAN loss ( $\mathcal{L}$ ) ↓	GenCO adversary ↓
GAN + MILP fix (previous)	0.52	<b>0.07</b>	0.94	0.22	0.24
GenCO - Fixed Adversary	0.22	0.05	<b>0.98</b>	-1.45	-0.85
GenCO - Updated Adversary	<b>0.995</b>	0.06	0.82	<b>-10.10</b>	<b>-4.49</b>

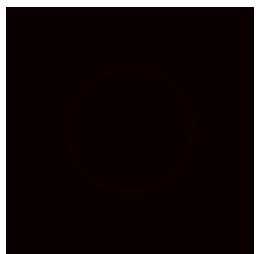
## Inverse Photonics Design

Approach	% Unique ↑	Density ↑	Coverage ↑	Avg Solution Loss ↓
VQVAE + postprocess	30.6%	0.009	0.006	1.244
GenCO (reconstruction only)	<b>100%</b>	0.148	0.693	1.155
GenCO (objective only)	46.6%	0.013	0.036	<b>0</b>
GenCO (reconstruction + objective)	<b>100%</b>	<b>0.153</b>	<b>0.738</b>	<b>0</b>

Constraints are  
strictly satisfied

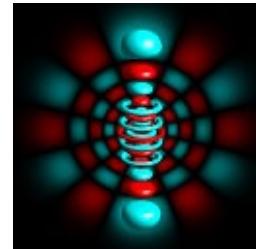
# Part III: Replace solvers with Deep Models

# Solving the linear PDE $\frac{\partial^n \psi}{\partial t^n} = F(\psi, \nabla_x \psi, \dots; \mathbf{h})$



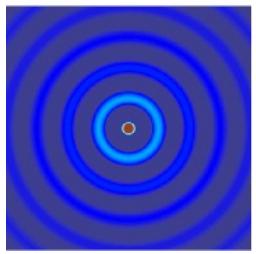
$$\frac{\partial \psi}{\partial t} = \nabla^2 \psi$$

Heat equation



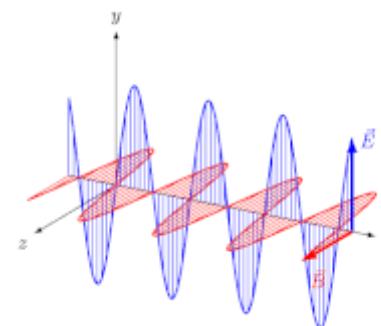
$$i\hbar \frac{\partial \psi}{\partial t} = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V \right] \psi$$

Schrodinger's Equation



$$\frac{\partial^2 \psi}{\partial t^2} = c^2 \nabla^2 \psi$$

Wave equation



$$\begin{aligned}\nabla \cdot E &= \frac{\rho}{\epsilon_0}, \nabla \cdot B = 0 \\ \nabla \times E &= -\frac{\partial B}{\partial t}, \nabla \times B = \mu_0 j + \frac{1}{c^2} \frac{\partial E}{\partial t}\end{aligned}$$

Maxwell's equation

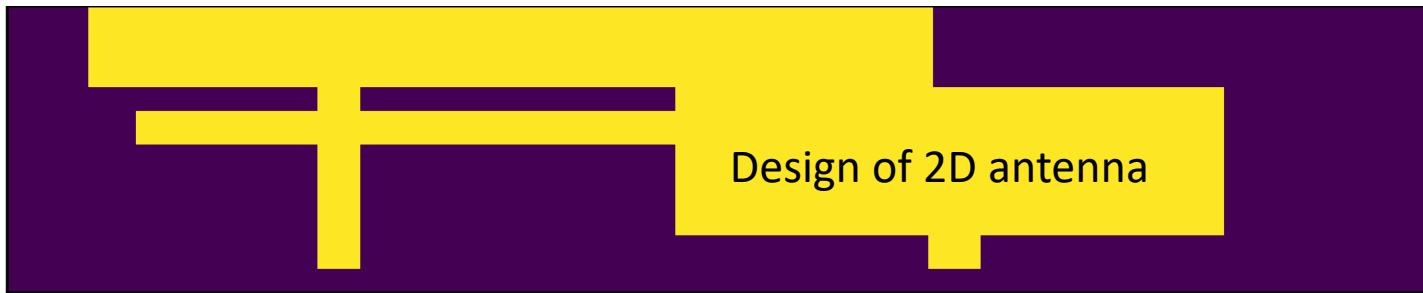
Tricky to simulate accurately and efficiently → Can we do better?

# Antenna Design problem

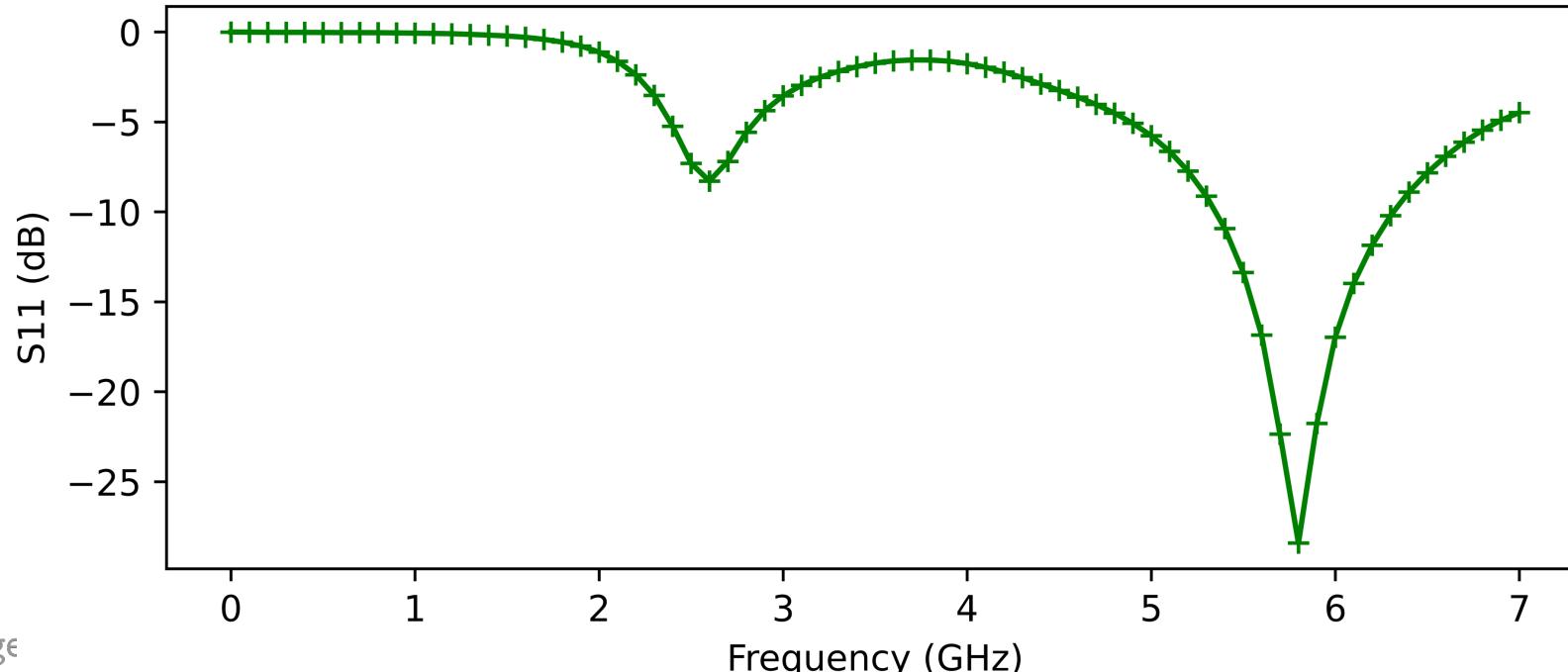
**Goal:**

find the right design to achieve  
the right frequency response

$h =$



$S_{11}(\psi) =$

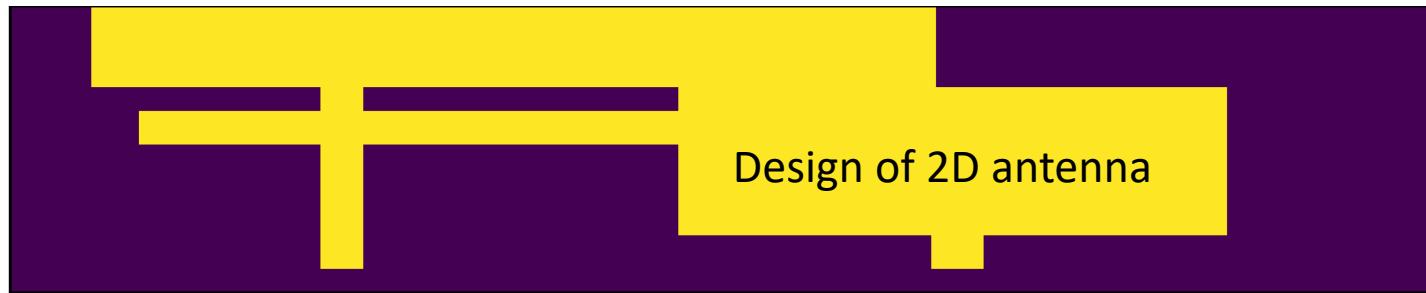


# Antenna Design problem

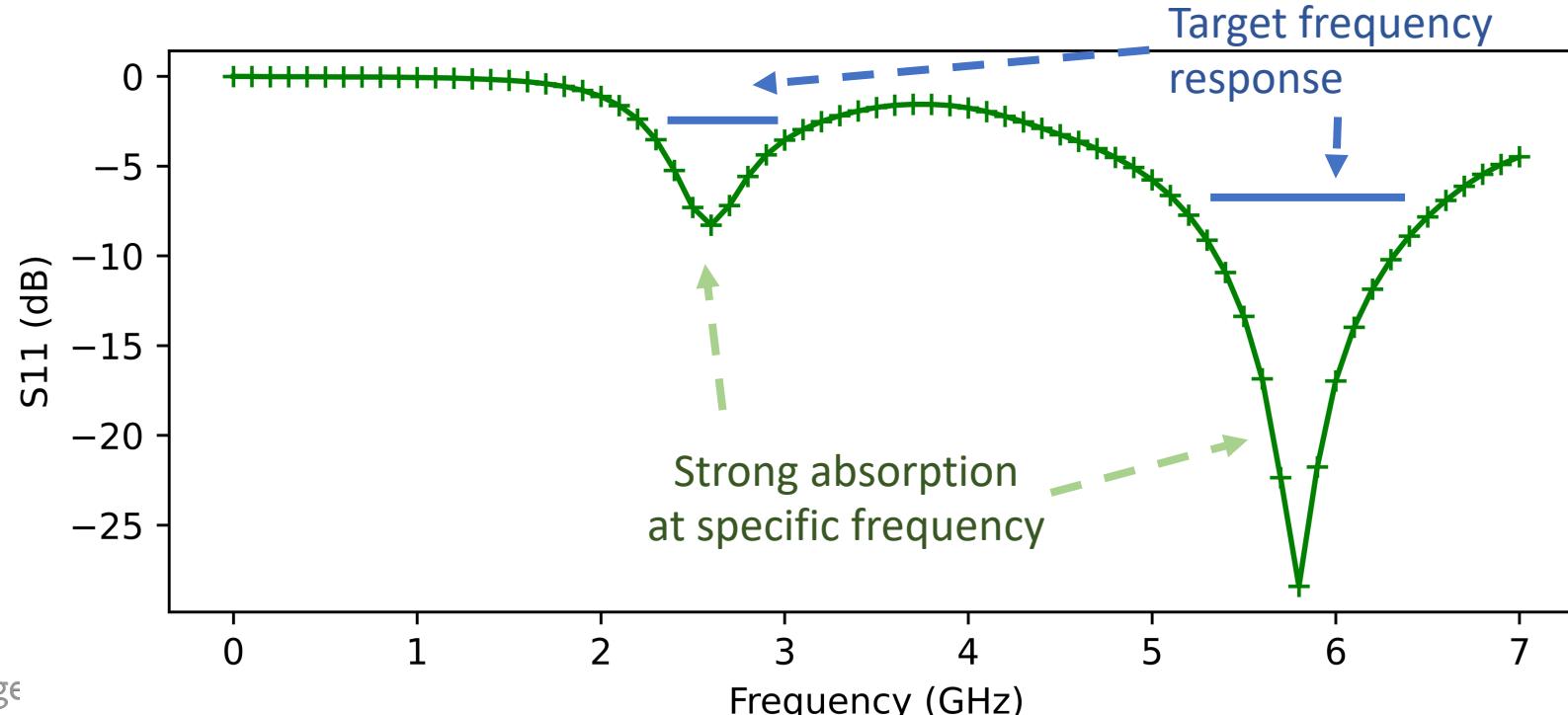
**Goal:**

find the right design to achieve the right frequency response

$h =$



$S_{11}(\psi) =$



# Parametric formula for Linear PDEs

**Theorem:** For any linear coefficients  $\mathbf{b}_1$  and  $\mathbf{b}_2$ :

$$\frac{\mathbf{b}_1^T \hat{\boldsymbol{\phi}}(\omega)}{\mathbf{b}_2^T \hat{\boldsymbol{\phi}}(\omega)} = c_0(\mathbf{h}) \prod_{k=1}^{K_1} (\omega - z_k(\mathbf{h})) \prod_{k=1}^{K_2} (\omega - p_k(\mathbf{h}))^{-1}$$

where the constant  $c_0(\mathbf{h})$ , zeros  $z_k(\mathbf{h})$  and poles  $p_k(\mathbf{h})$  are complex functions of the design choice  $\mathbf{h}$

**Proof idea:** Linear ODE theory gives us the analytic form of the solution  $\boldsymbol{\phi}(t) = e^{At} \boldsymbol{\phi}(0)$ . Fourier Transform yields  $\hat{\boldsymbol{\phi}}(\omega)$  as a rational function of complex polynomials w.r.t. frequency  $\omega$ .

# For Antenna Optimization

The *Scattering Coefficients*  $S_{11}(\omega)$ :

$$S_{11}(\omega) = \frac{Z_{\text{in}}(\omega) - Z_0}{Z_{\text{in}}(\omega) + Z_0}$$

$Z_{\text{in}}(\omega)$ : Input *Impedance*. Impedance  $Z(\omega) := V(\omega)/I(\omega)$

  
Voltage  
(in Fourier domain)      Current  
(in Fourier domain)

Both are linear function w.r.t. signal  $\hat{\phi}(\omega)$

# Parameter form of $\log|S_{11}(\omega)|$

$$\log|S_{11}(\omega)| = \log|c_o(\mathbf{h})| + \sum_{k=1}^K \log \frac{|\omega - z_k(\mathbf{h})|}{|\omega - p_k(\mathbf{h})|}$$

where the constant  $c_0(\mathbf{h})$ , zeros  $z_k(\mathbf{h})$  and poles  $p_k(\mathbf{h})$  are complex functions of the design choice  $\mathbf{h}$ .

# CZP model architecture

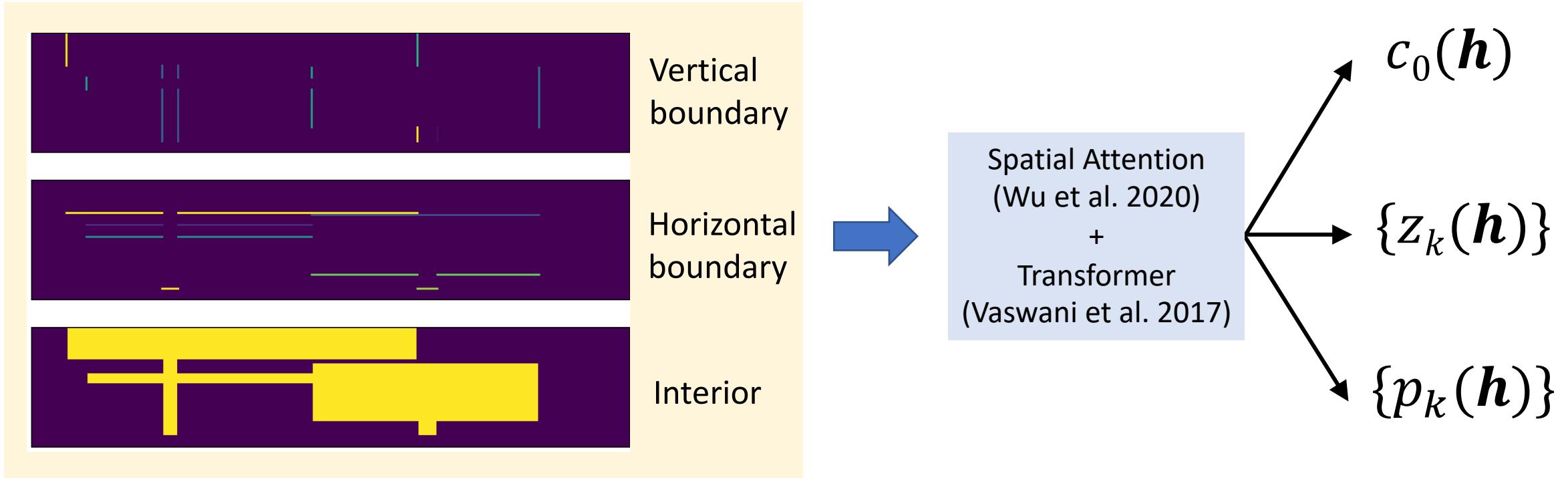
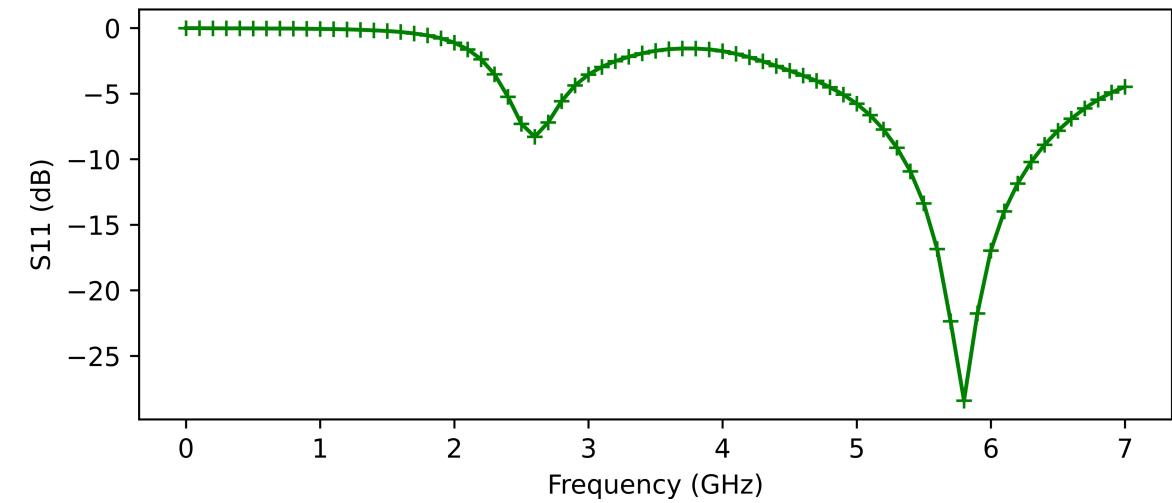
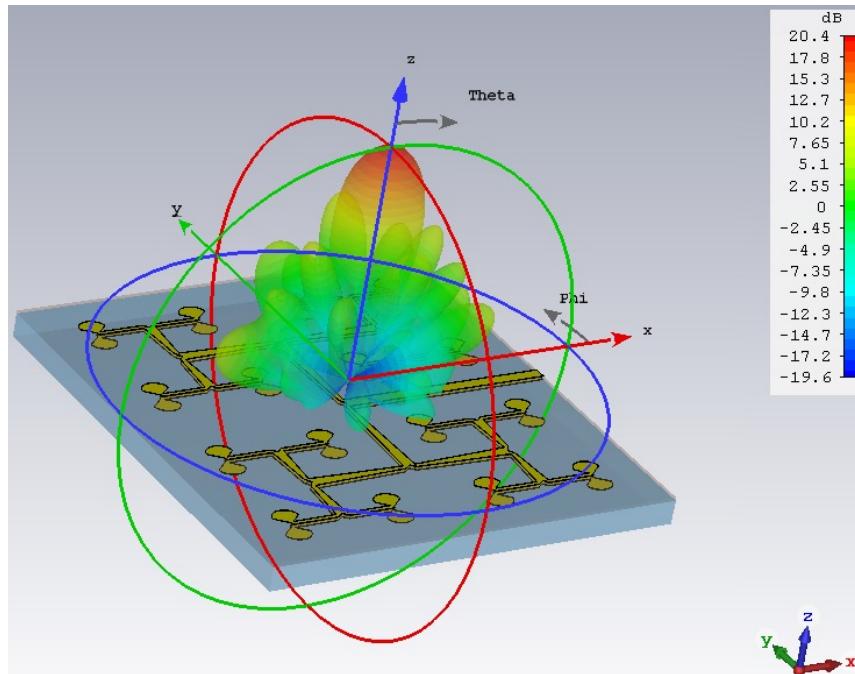


Image-based Representation  
of design choice  $h$

Predict the constant, zeros and poles from an  
image representation of an antenna

# Data Collection

Dataset is collected from commercial simulators (e.g., CST)  
Numerical simulation of Electromagnetic wave dynamics

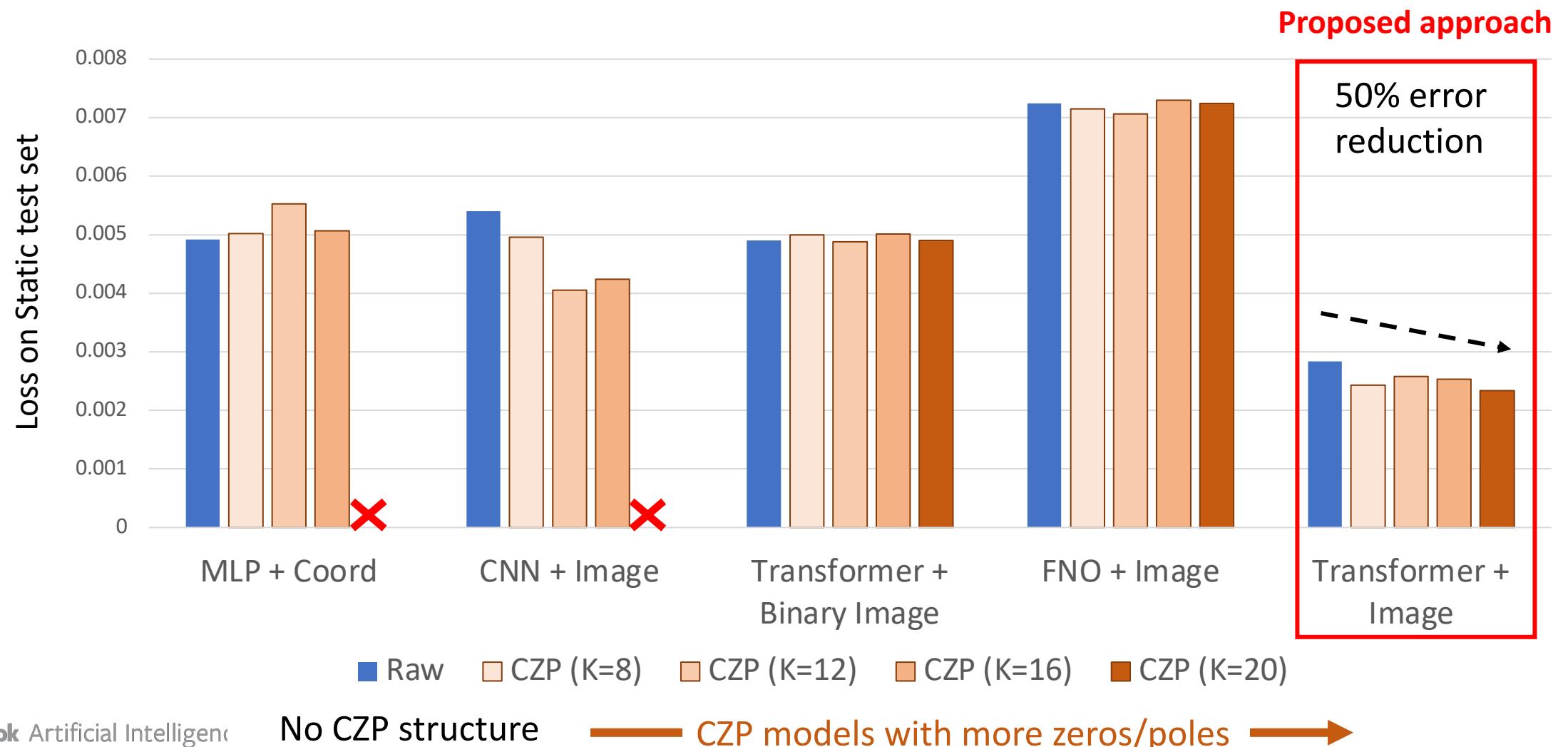


It takes minutes (or even hours) to get one simulation data point.  
Dataset size = 48k

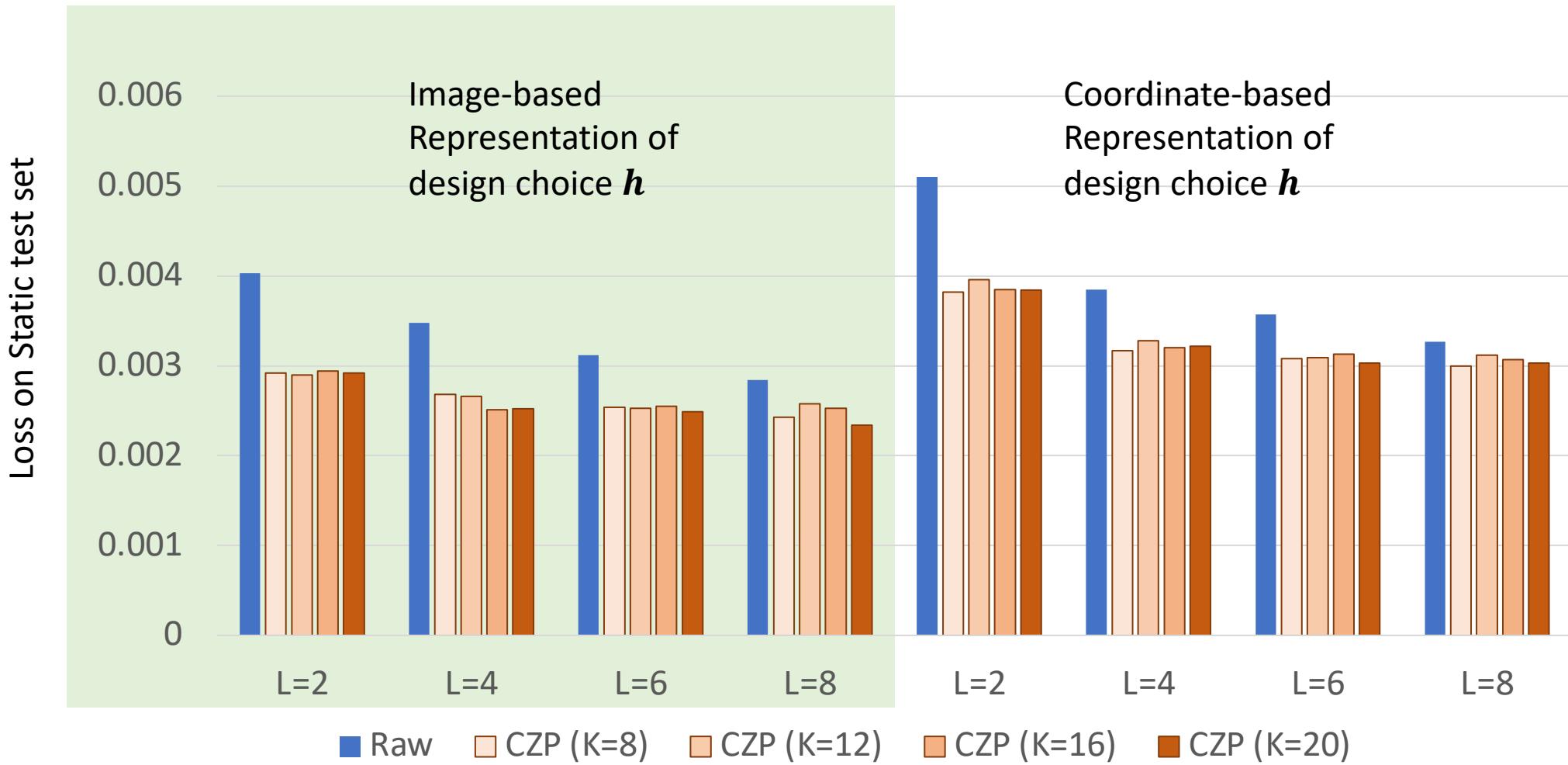
# Model Evaluation

- Static Evaluation
  - On a held-out test set, compute the loss
  - Loss = gap between surrogate models and ground truth (commercial software)
- Dynamic Evaluation
  - Use the surrogate model to search a good design
  - Evaluate the design in ground truth (commercial software)

# Static Evaluation: Surrogate Model Test Loss

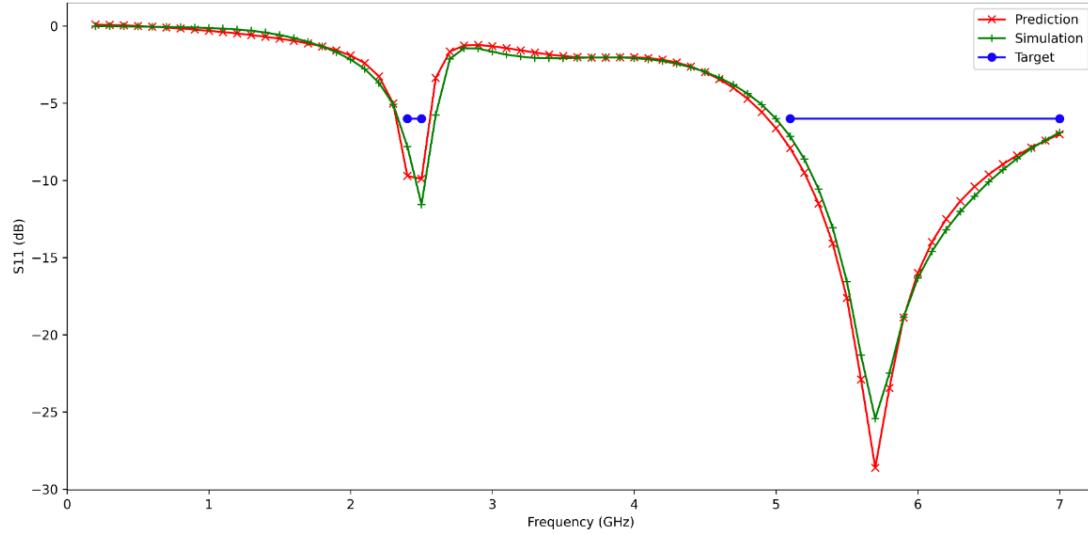
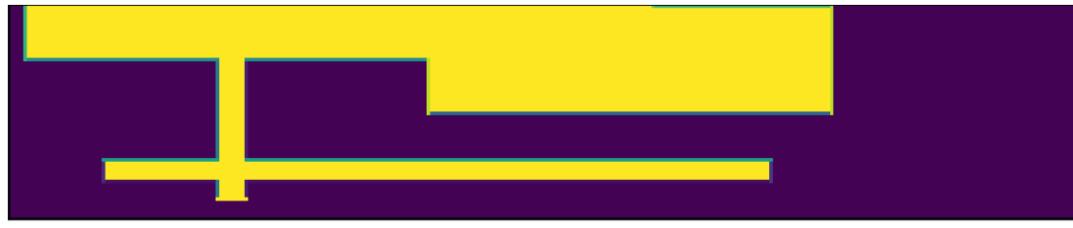


# Static Evaluation: Surrogate Model Test Loss

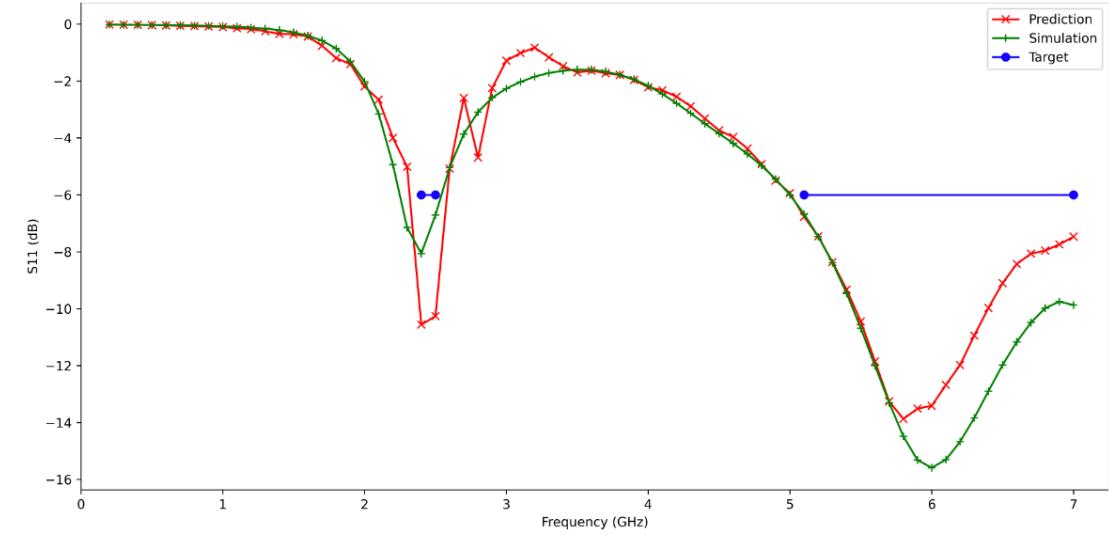
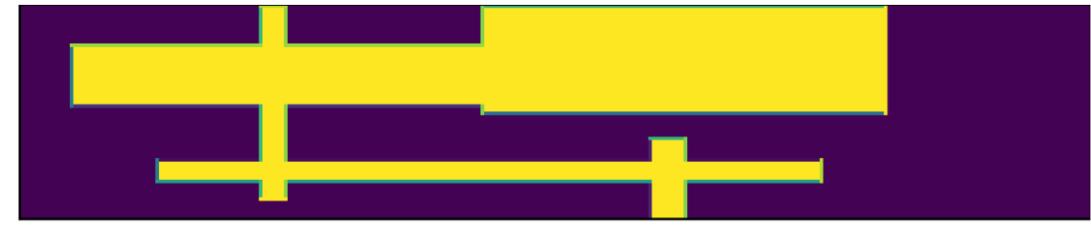


# Visualization

Our CZP model captures the smooth structure of scattering coefficients  $S_{11}(\omega)$



(a) CZP



(b) Raw

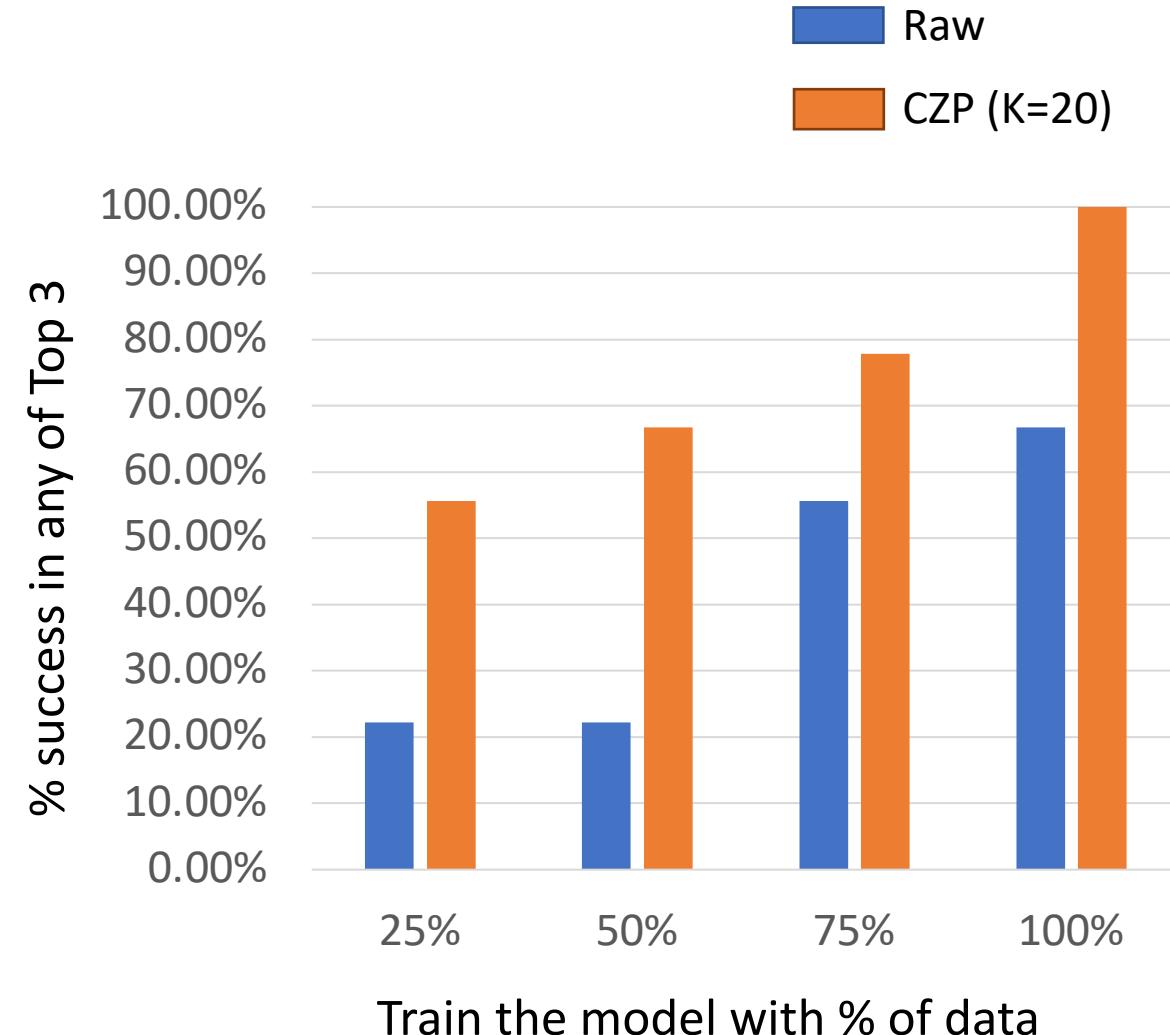
# Dynamic Evaluation: CZP model with Search

We use Soft Actor-Critic  
as the specific search technique.

**Goal:** to find a solution to satisfy the  
frequency constraints (verified with CST)

3 models trained x 3 search attempts using  
different random seed

For each attempt, check top-3 solutions



# Thanks!