



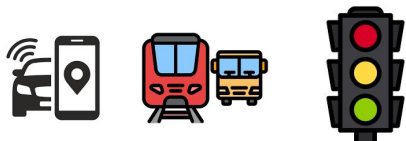
# Optimization in the Era of AI

Dr. Yining Ma ([yiningma@mit.edu](mailto:yiningma@mit.edu))

Laboratory for Information & Decision Systems (LIDS)



# Recap: Optimization is everywhere!



## Wake up...

Call a Uber (task **assignment**)

Take public transit (bus/metro **planning**)



# NP-Hard!!!

(scheduling)

(resource allocation)



Open  
Review  
.net



## Return home...

Amazon package (**facility location** + **packing** + **routing**)

Check NeurIPS reviews (paper-reviewer **matching**)

Check Stock Investment (**QCQP**)

# Recap: Optimization Solvers



**Exact Solvers**



**Heuristic Solvers**

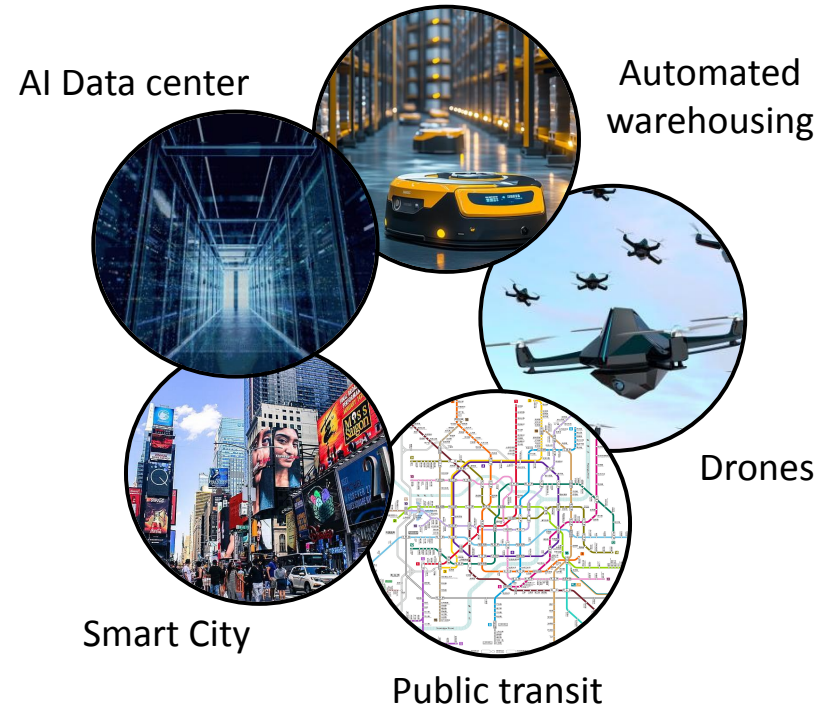


**Operations Research (OR)  
Human Solvers**

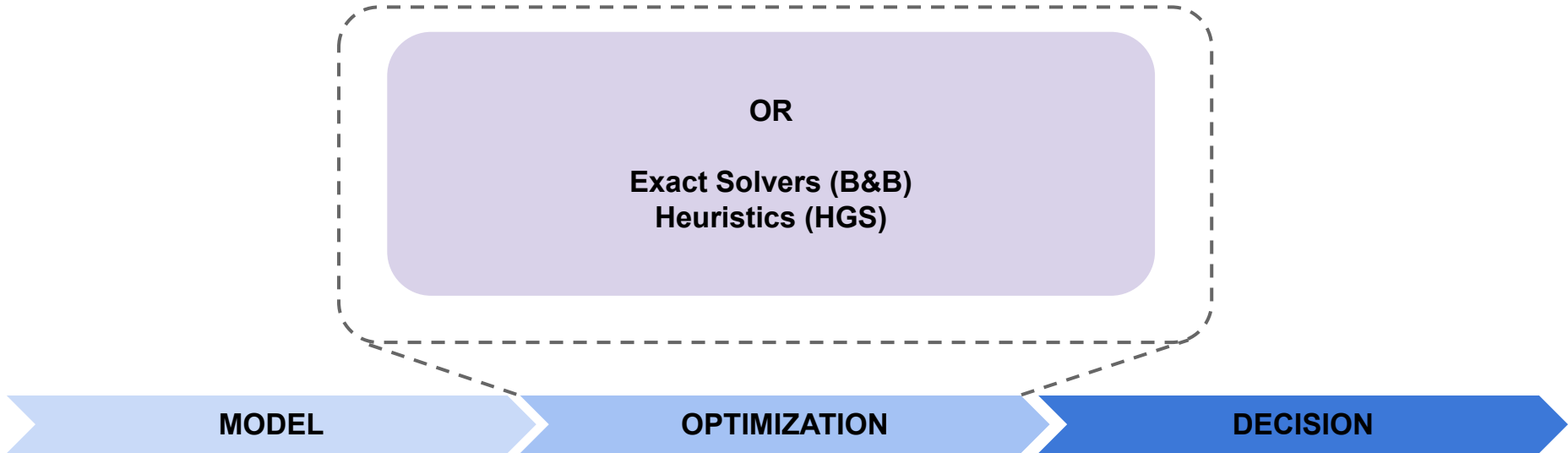
# Recap: Optimization is getting complex!

## Sources of complexity:

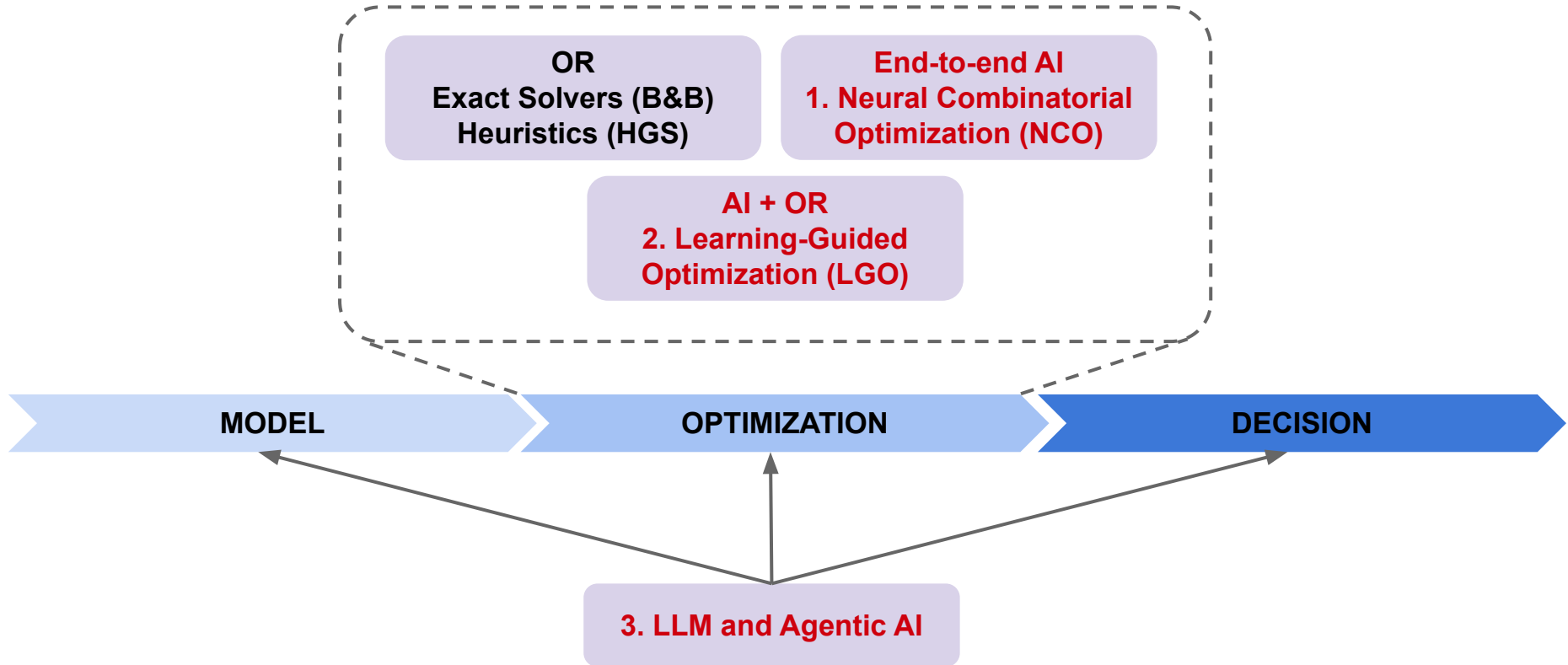
1. **Scalability**  
(millions to billions decision variables)
2. **Efficiency**  
(low cost, fast service, low latency)
3. **Variations**  
(new domains emerge, such as AI data center, drones, EVs, robot coordination, chip design, etc)
4. **Dynamics and Uncertainty**  
(hard to model constraints, underlying dynamics, long-term behaviors)
5. ...



# What we have so far

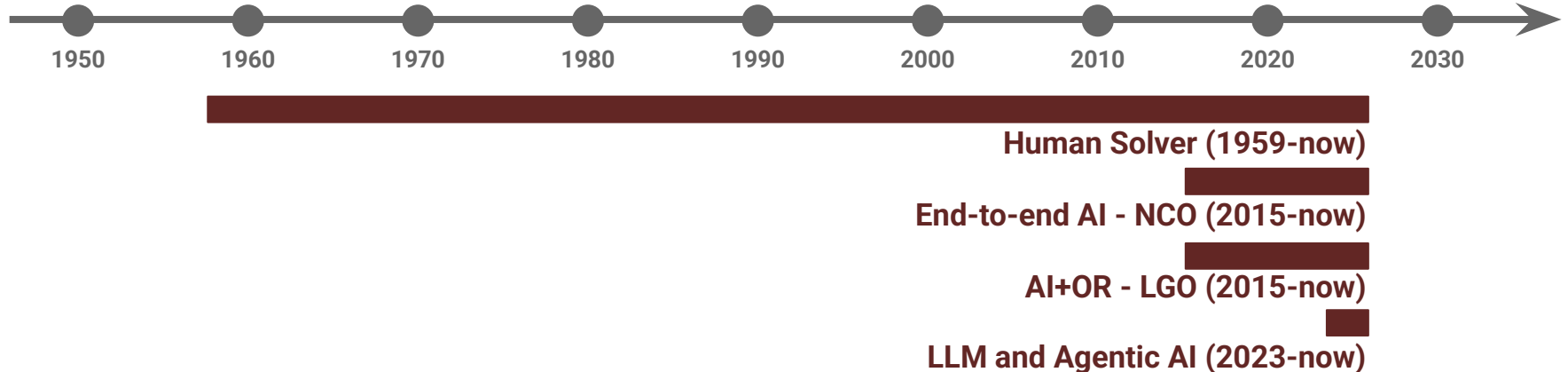


# Today's journey



# By the end of this lecture, you will be able to...

- Recognize the limitations of human solvers and motivations for AI solvers.
- Describe the methodology pipeline and research focus of NCO solves.
- Describe the real-world impact and research focus of LGO solvers.
- Summarize LLM's different roles for the next-generation decision pipeline.

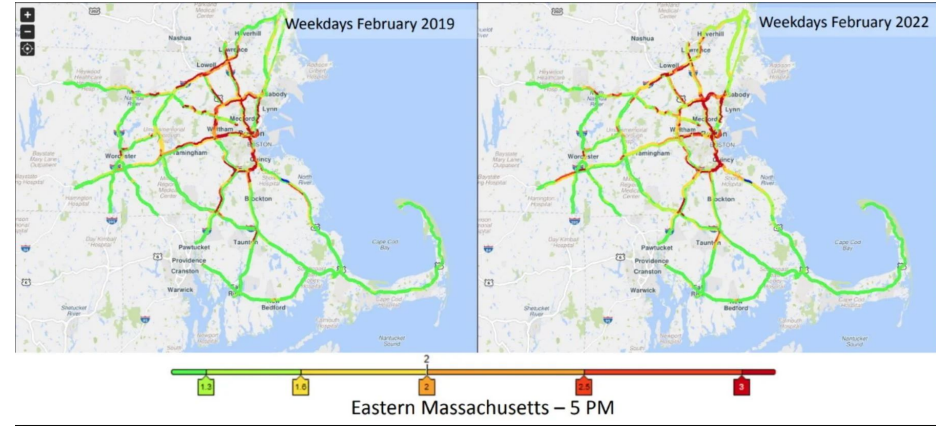
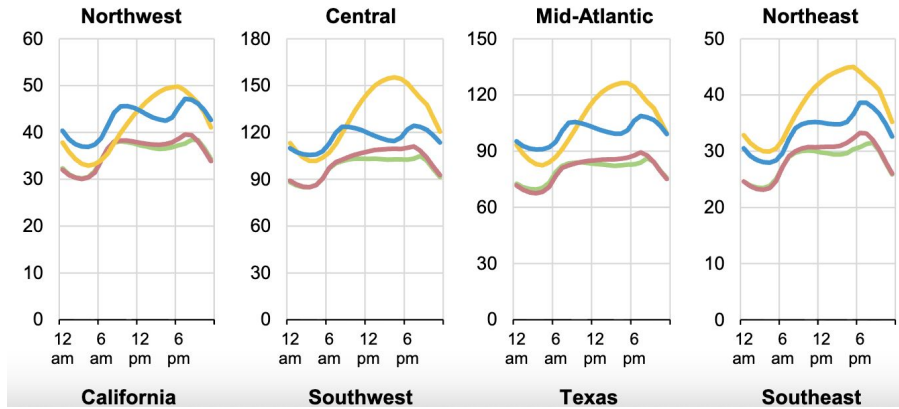


# 01

## Limitation of Human Solvers

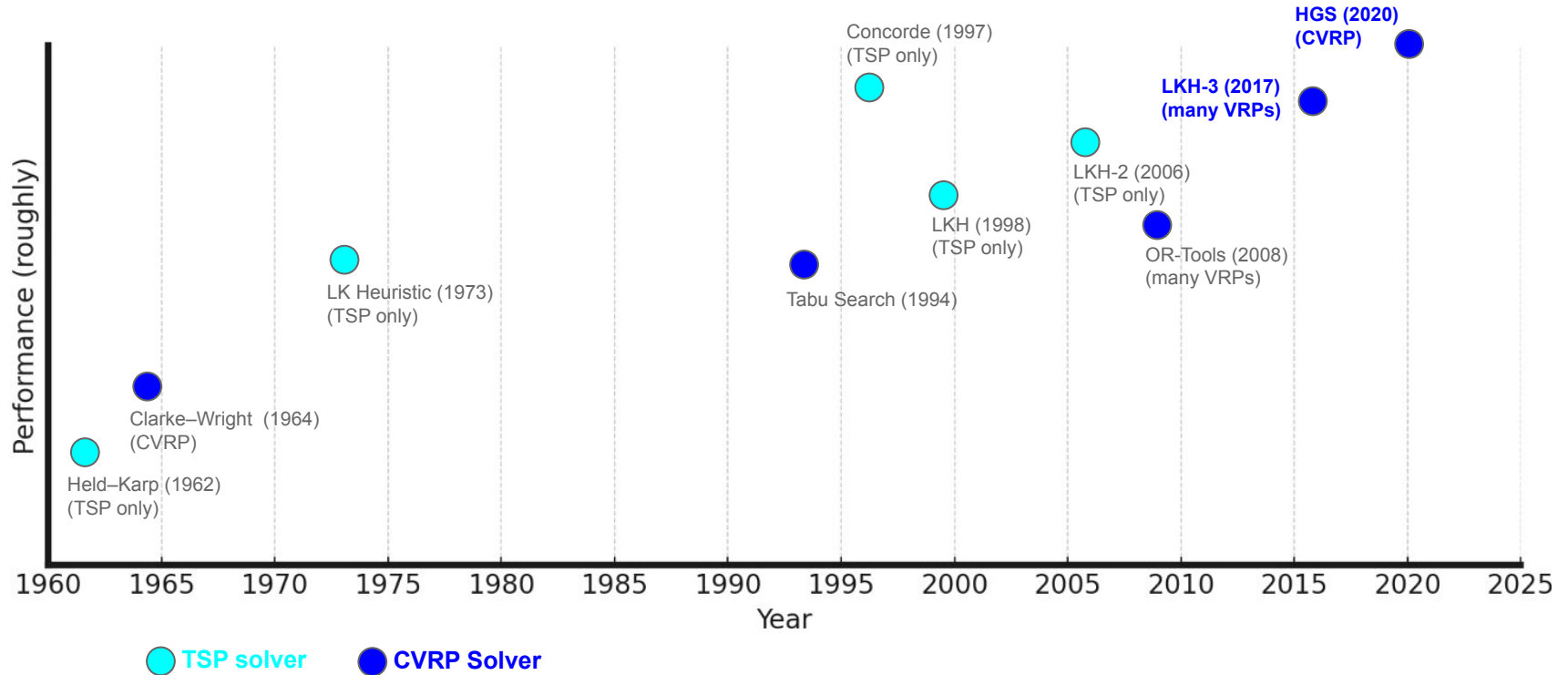


# Human solvers do individual optimization

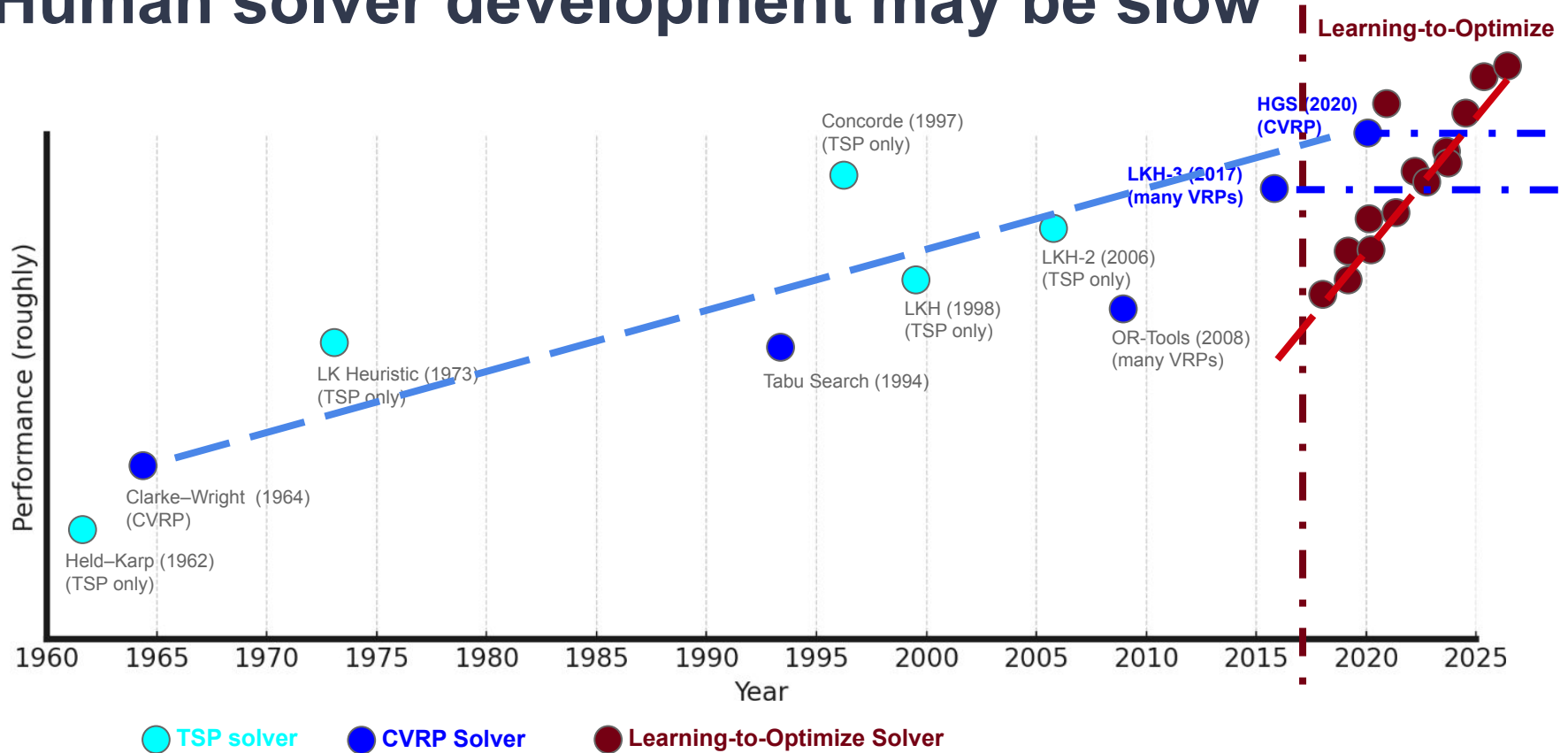


Why solve similar problems from scratch **individually**?

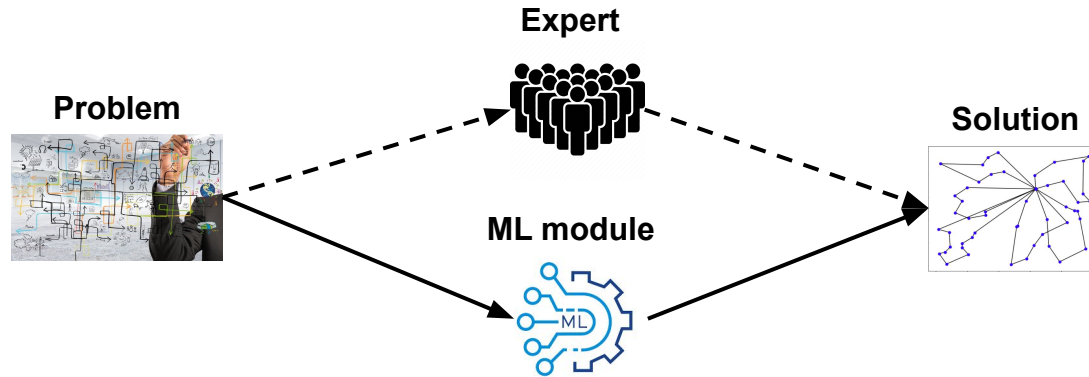
# Human solver development may be slow



# Human solver development may be slow



# AI for Optimization!



## Traditional Solver

- Heavy reliance on **human knowledge**
- Optimize each instances **individually**
- Relatively **slow** evolution due to the needs of trial and error and manpower

## AI for Optimization

- Automate **data-driven** pattern recognition
- Works for a **distribution** of instances
- Relatively **fast** evolution by running Reinforcement Learning/LLM 24/7

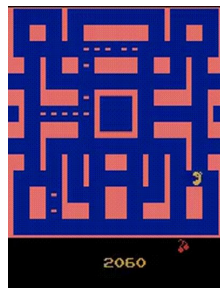
02

Neural Combinatorial Optimization (NCO)  
(2015 - now)

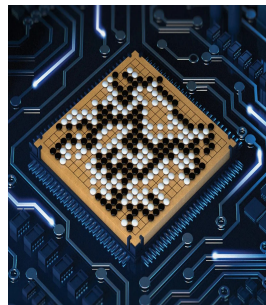
# The Belief of Super-human Deep Learning

## AI Community

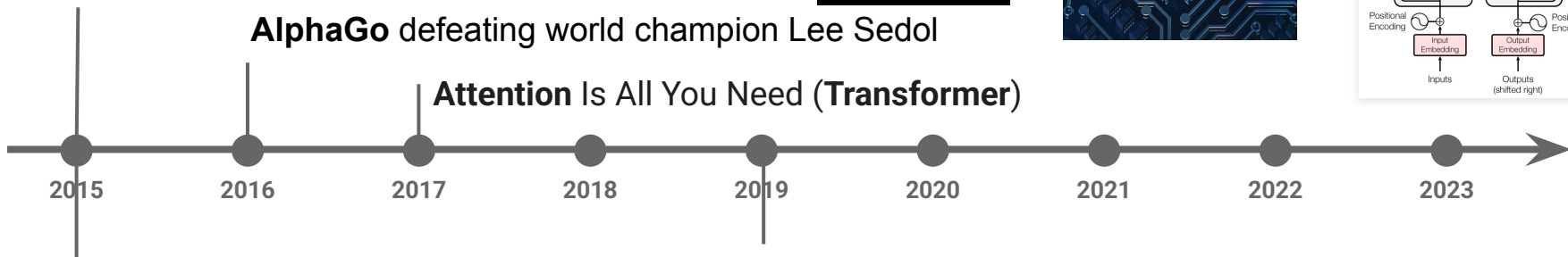
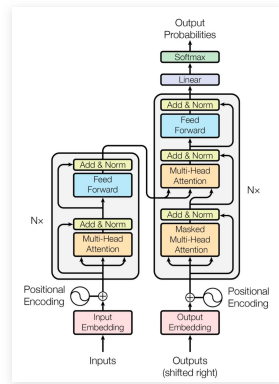
Nature: Deep Q-Network (DQN) for Atari Games



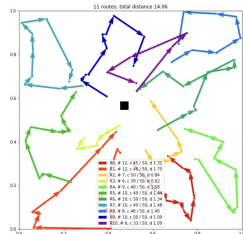
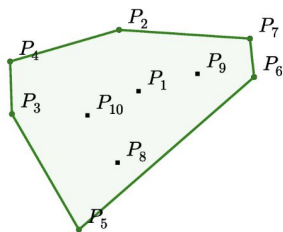
AlphaGo defeating world champion Lee Sedol



Attention Is All You Need (Transformer)

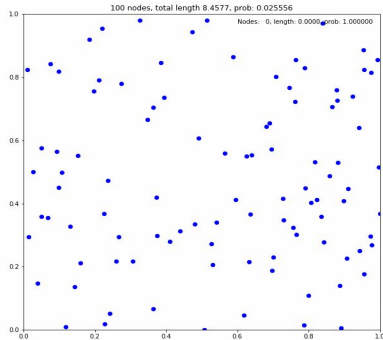
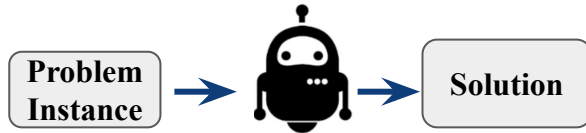


## NCO Community

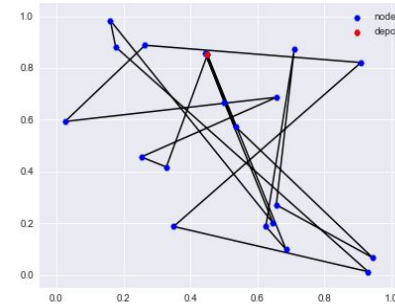
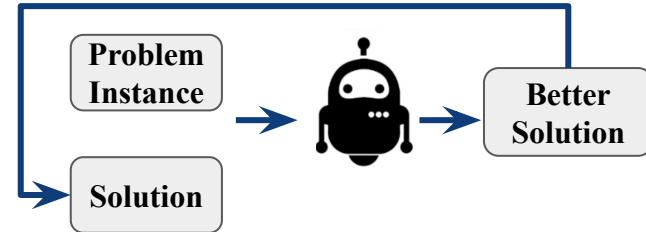


# Types of the NCO Solvers

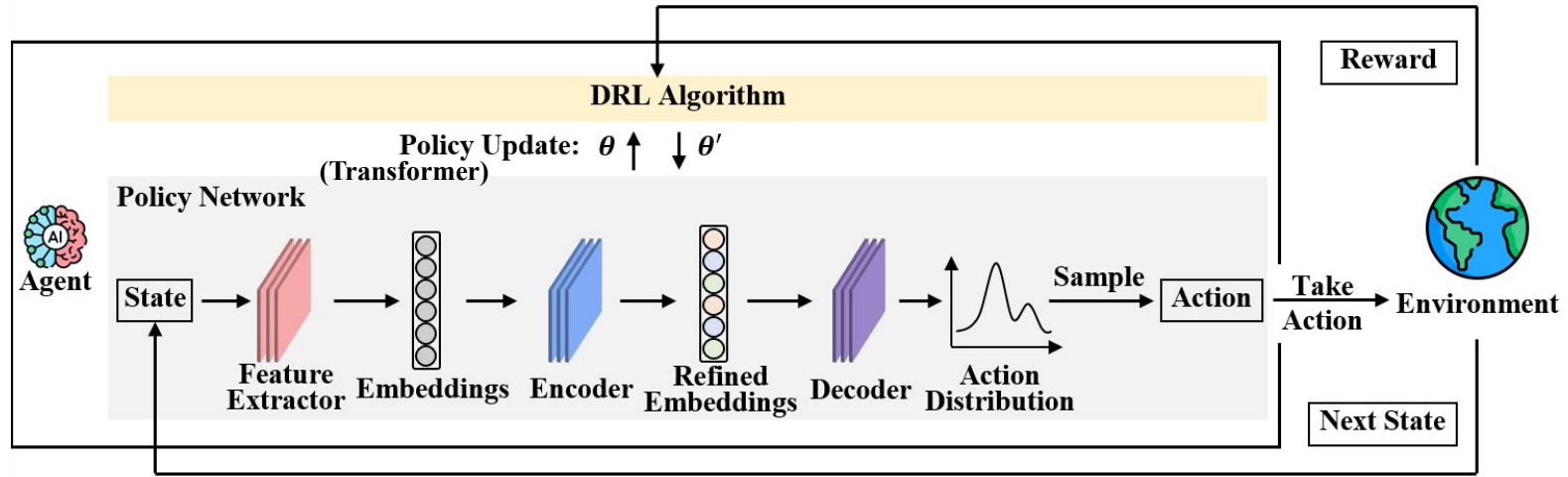
## Learning-to-Construct



## Learning-to-Search



# The Deep Reinforcement Learning pipeline



MDP

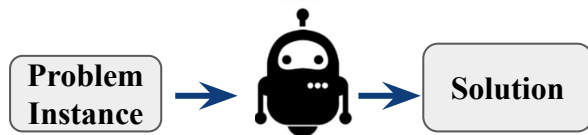
Policy Network

Constraint Handling

RL Training

Inference Method

# Learning-to-Construct



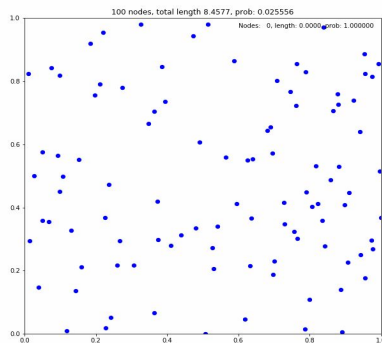
**MDP**

**State:** Node coordinate, first visited node, last visited node

**Action:** Select next node to visit

**Reward:** Negative total tour length

**Horizon:** Finite; terminate until all nodes are visited



**Policy Network**

Transformer (node embedding)

**Constraint Handling**

Invalid actions are masked out.

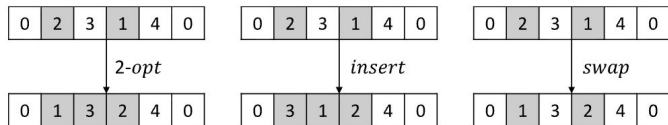
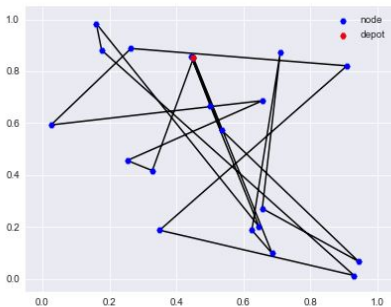
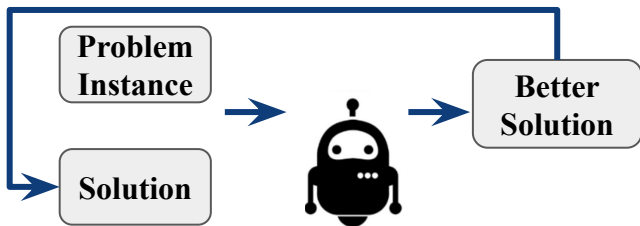
**RL Training**

REINFORCE (policy gradient)

**Inference Method**

Greedy, Sampling

# Learning-to-Search



MDP

**State:** Node coordinate, **current solution**  
**Action:** A particular 2-*opr*/swap/insert node pair  
**Reward:** Tour length reduction w.r.t best-so-far solution  
**Horizon:** **Infinite**; perform given local search steps

Policy Network

Transformer (node embedding) + **positional encoding**

Constraint Handling

Invalid actions are masked out.

RL Training

**A2C/PPO** (policy gradient with critic network)

Inference Method

Sampling

# NCO are keeping evolving rapidly...

Learning to **Construct** (around 2021)

	CVRP50		CVRP100	
	Gap	Time	Gap	Time
LKH3	-	7h	-	12h
<b>NCO (POMO)</b> [NeurIPS 2020]	<b>0.45%</b>	<b>26s</b>	<b>0.32%</b>	<b>2m</b>

Learning to **Construct** (around 2025)

	CVRP1k		CVRP50k	
	Gap	Time	Gap	Time
HGS	-	2.5m	-	8.1h
<b>NCO (SIL)</b> [ICLR 2025]	<b>2.72%</b>	<b>1.5m</b>	<b>-2.72%</b>	<b>49m</b>

Learning to **Search** (around 2021)

	PDTSP-LIFO-50		PDTSP-LIFO-100	
	Gap	Time	Gap	Time
LKH3	-	16m	-	67m
<b>NCO (N2S)</b> [IJCAI 2022]	<b>-0.55%</b>	<b>13m</b>	<b>-0.47%</b>	<b>54m</b>

Learning to **Search** (around 2025)

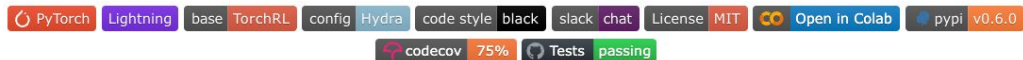
	VRPTW1k		VRPTW2k	
	Gap	Time	Gap	Time
HGS	-	120s	-	240s
<b>NCO (NDS)</b> [TMLR 2025]	<b>-3.14%</b>	<b>120s</b>	<b>-3.50%</b>	<b>240s</b>

[1] Kwon, Yeong-Dae, et al. "Pomo: Policy optimization with multiple optima for reinforcement learning." *Advances in neural information processing systems* 33 (2020): 21188-21198.

[2] Luo, Fu, et al. "Boosting neural combinatorial optimization for large-scale vehicle routing problems." *The Thirteenth International Conference on Learning Representations*. 2025.

[3] Ma, Yining, et al. "Efficient neural neighborhood search for pickup and delivery problems." *International Joint Conference on Artificial Intelligence*. 2022.

[4] Hottung, André, Paula Wong-Chung, and Kevin Tierney. "Neural deconstruction search for vehicle routing problems." *arXiv preprint arXiv:2501.03715* (2025).

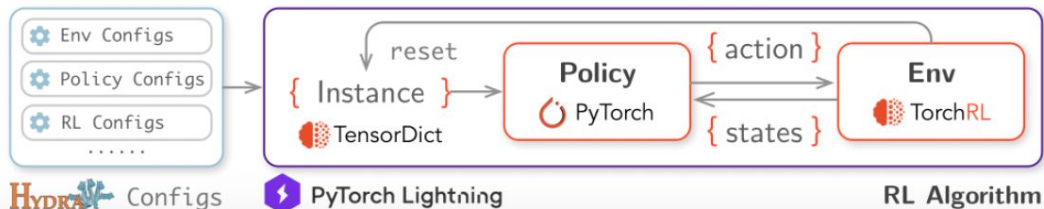


[Documentation](#) | [Getting Started](#) | [Usage](#) | [Contributing](#) | [Paper](#) | [Join Us](#)

An extensive Reinforcement Learning (RL) for Combinatorial Optimization (CO) benchmark. Our goal is to provide a unified framework for RL-based CO algorithms, and to facilitate reproducible research in this field, decoupling the science from the engineering.

RL4CO is built upon:

- [TorchRL](#): official PyTorch framework for RL algorithms and vectorized environments on GPUs
- [TensorDict](#): a library to easily handle heterogeneous data such as states, actions and rewards
- [PyTorch Lightning](#): a lightweight PyTorch wrapper for high-performance AI research
- [Hydra](#): a framework for elegantly configuring complex applications



We offer flexible and efficient implementations of the following policies:

v0.6.0 Latest  
on May 30, 2025

[+ 19 releases](#)

**Deployments** 88

✓ [github-pages](#) 3 months ago

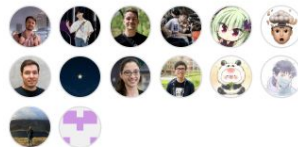
✓ [pypi](#) 11 months ago

[+ 86 deployments](#)

**Packages**

No packages published  
[Publish your first package](#)

**Contributors** 25



[+ 11 contributors](#)

**Languages**

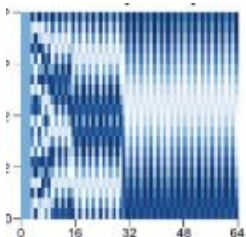
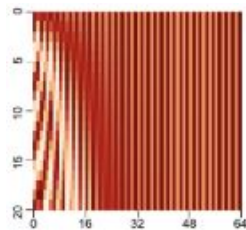
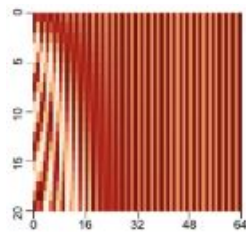
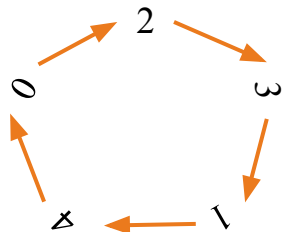
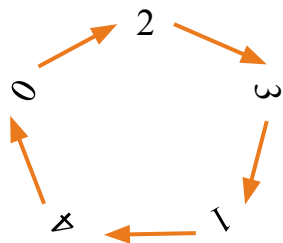
● Python 100.0%

<https://github.com/ai4co/rl4co/>

# NCO needs new designs (Network)

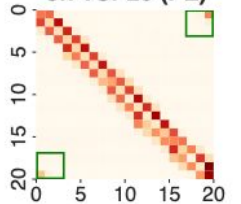
Case study: Position encoding from NLP assumes **linear** sequence

Tom → loves → movies → so → much.

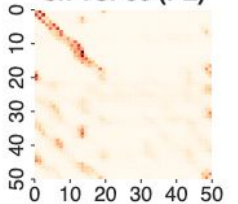


The NCO model memorizes the cyclic nature of the 1st and 20th nodes, preventing it from generalizing to larger graphs of 50 nodes.

TSP20 Policy tested on TSP20 (PE)

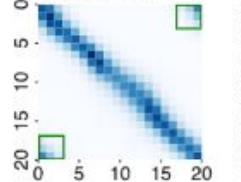


TSP20 Policy tested on TSP50 (PE)

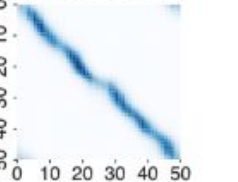


Gap: 7.93%

TSP20 Policy tested on TSP20 (CPE)



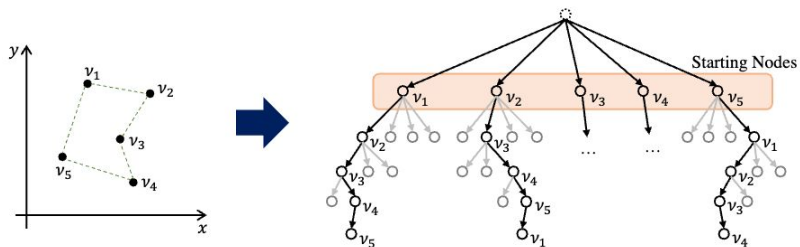
TSP20 Policy tested on TSP50 (CPE)



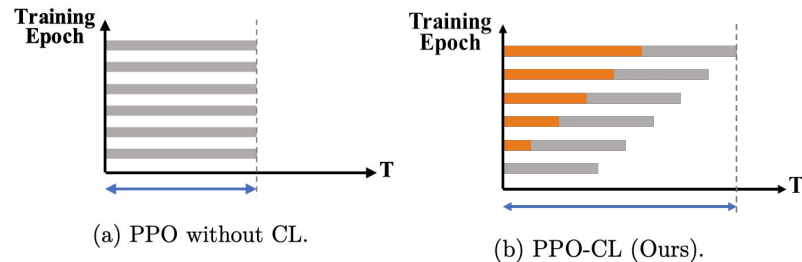
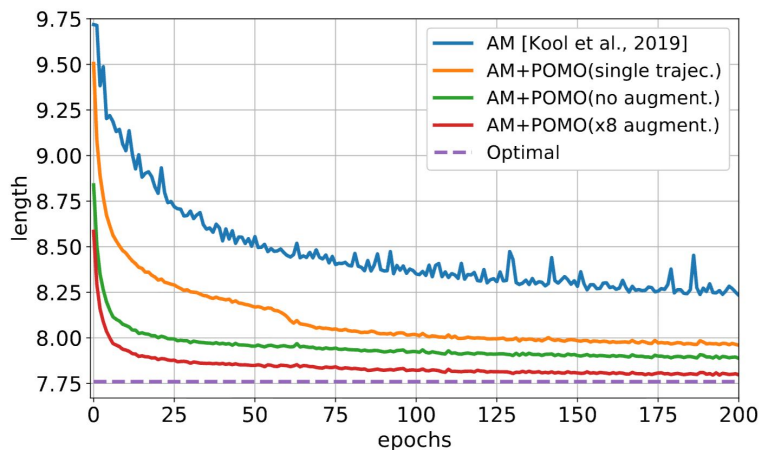
Gap: 2.98%

# NCO needs new designs (Training)

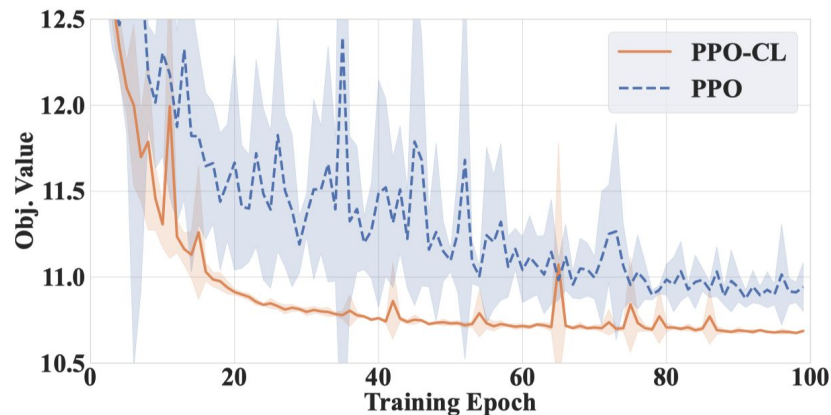
## Case study: Customize RL algorithms for NCO training



### Grouped Baseline for Better REINFORCE

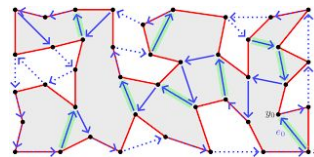
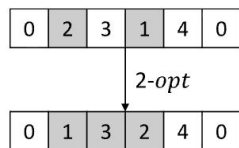
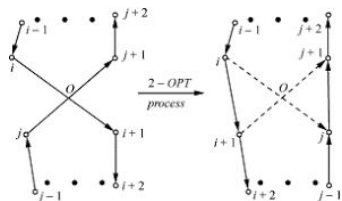


### Curriculum Learning for Better PPO



# NCO needs new designs (MDP)

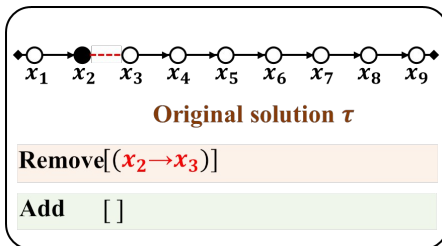
Case study: Action decomposition for exponentially increasing state/action space



**k-opt Factorization** - basis moves: Starting move (S-move), Intermediate move (I-move), Ending move (E-move)

## S-move

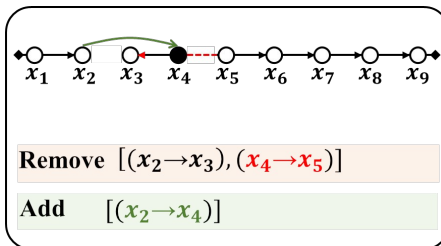
1. Remove one edge



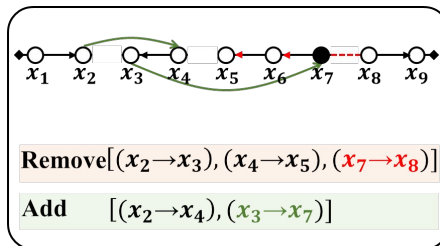
$S(x_2)$

## I-move

1. Add a new edge (starting from the first end-node)
2. Remove the corresponding conflicting edge
3. Reverse the edge directions in between



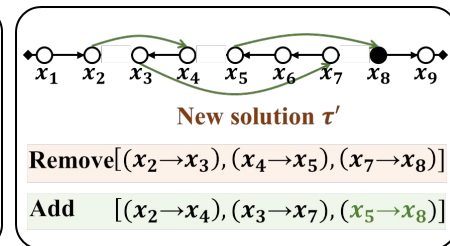
$I(x_4)$



$I(x_7)$

## E-move

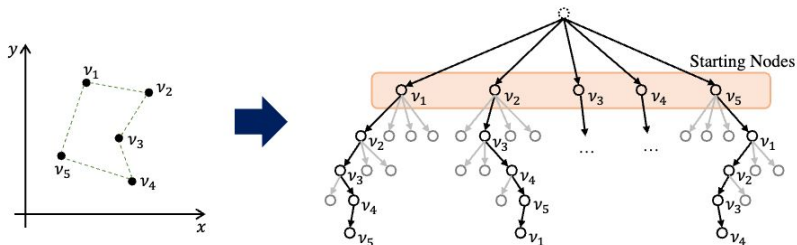
1. Close the loop



$E(x_8)$

# From “AI for CO” to “CO for AI”!

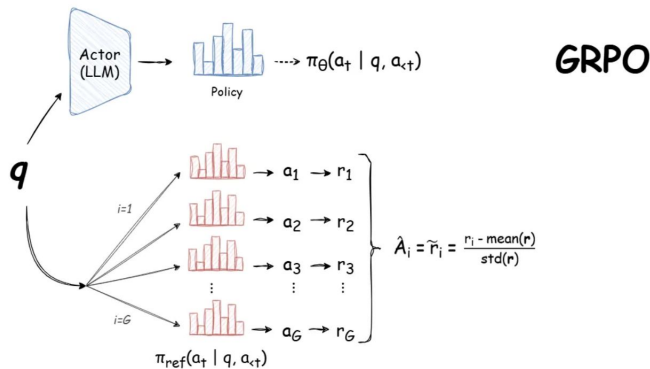
## Grouped Baseline for NCO [POMO, 2020]



## Position: Application-Driven Innovation in Machine Learning

David Rolnick, Alan Aspuru-Guzik, Sara Beery, Bistra Dilikina, Priya L. Donti, Marzyeh Ghassemi, Hannah Kerner, Claire Monteleoni, Esther Rolf, Milind Tambe, Adam White  
*Proceedings of the 41st International Conference on Machine Learning, PMLR 235:42707-42718, 2024.*

## Group Relative Policy Optimization (GRPO) [DeepSeek-AI, 2024]



### Abstract

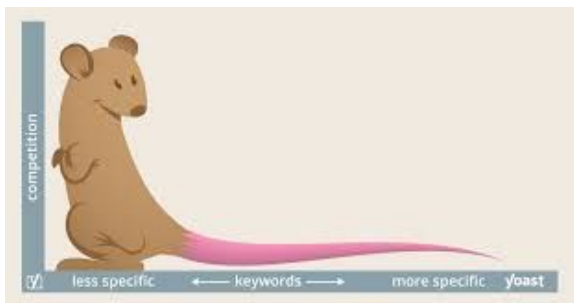
In this position paper, we argue that application-driven research has been systemically under-valued in the machine learning community. As applications of machine learning proliferate, innovative algorithms inspired by specific real-world challenges have become increasingly important. Such work offers the potential for significant impact not merely in domains of application but also in machine learning itself. In this paper, we describe the paradigm of application-driven research in machine learning, contrasting it with the more standard paradigm of methods-driven research. We illustrate the benefits of application-driven machine learning and how this approach can productively synergize with methods-driven work. Despite these benefits, we find that reviewing, hiring, and teaching practices in machine learning often hold back application-driven innovation. We outline how these processes may be improved.

[1] Kwon, Yeong-Dae, et al. "Pomo: Policy optimization with multiple optima for reinforcement learning." *Advances in neural information processing systems* 33 (2020): 21188-21198.

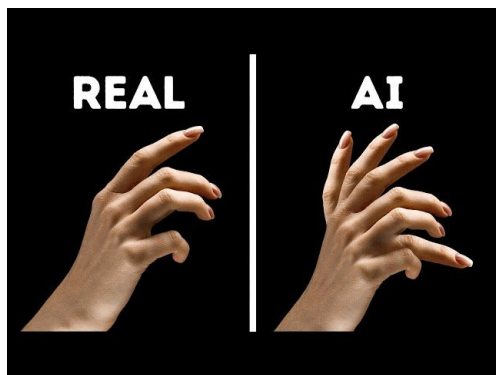
[2] Application-Driven Innovation in Machine Learning. David Rolnick, Alan Aspuru-Guzik, Sara Beery, Bistra Dilikina, Priya L. Donti, Marzyeh Ghassemi, Hannah Kerner, Claire Monteleoni, Esther Rolf, Milind Tambe, Adam White

# NCO is still not mature!

## Robustness to long-tail cases



## Failure cases

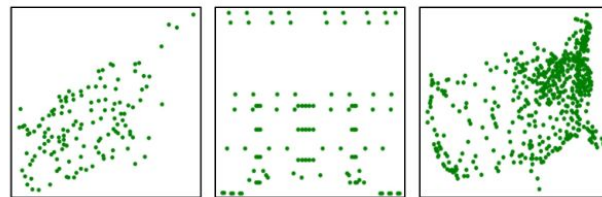


## Generalization performance

(reference AM performance: 3.72%)

Instance	Opt.	AM
X-n101-k25	27591	38264
X-n106-k14	26362	27923
X-n110-k13	14971	16320
X-n115-k10	12747	14055
X-n120-k6	13332	14456
X-n125-k30	55539	74329
X-n129-k18	28940	30869
X-n134-k13	10916	13952
X-n139-k10	13590	14893
X-n143-k7	15700	18251
Avg. Gap ( $n = 100-150$ )	0.00%	16.65%

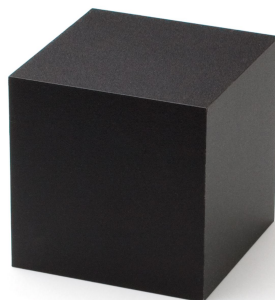
## Robustness to distribution-shift



## Real-world setups



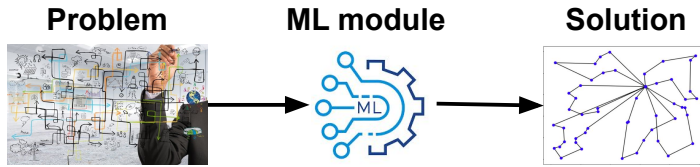
## Black-box decision



# 03

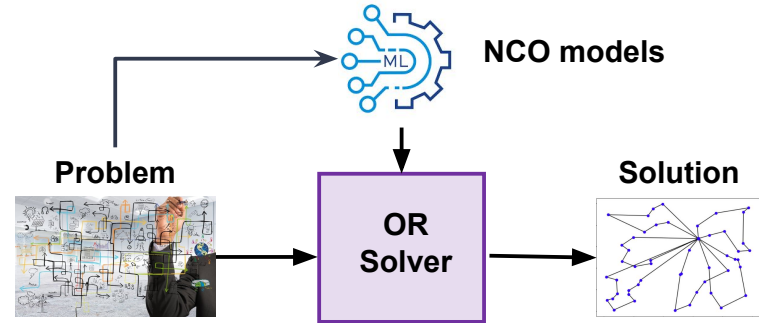
## Learning-Guided Optimization (LGO) (2015 - now)

# NCO v.s. LGO



## Neural Combinatorial Optimization

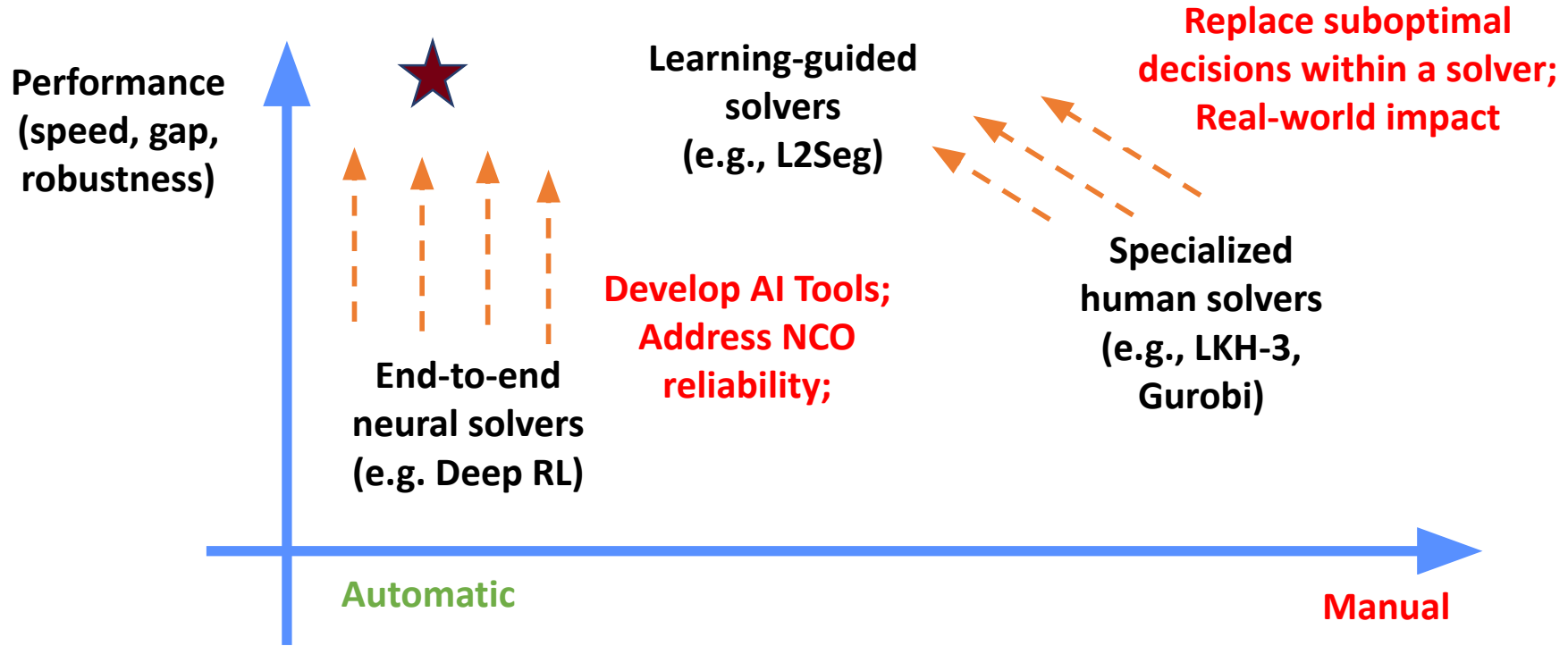
- Learning neural solver **from scratch**
- **Black box** solver - no guarantee
- Excel if no distribution shifts
- Impactful for AI technique developments
- **100% data-driven**



## Learning-Guided Optimization

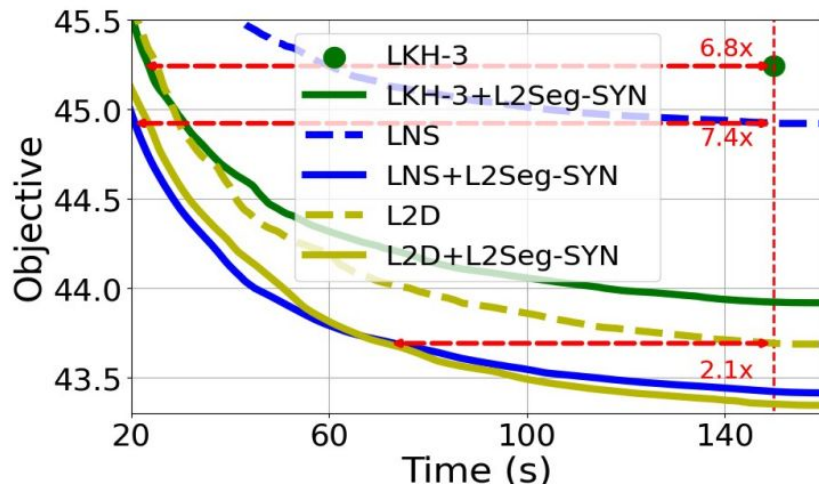
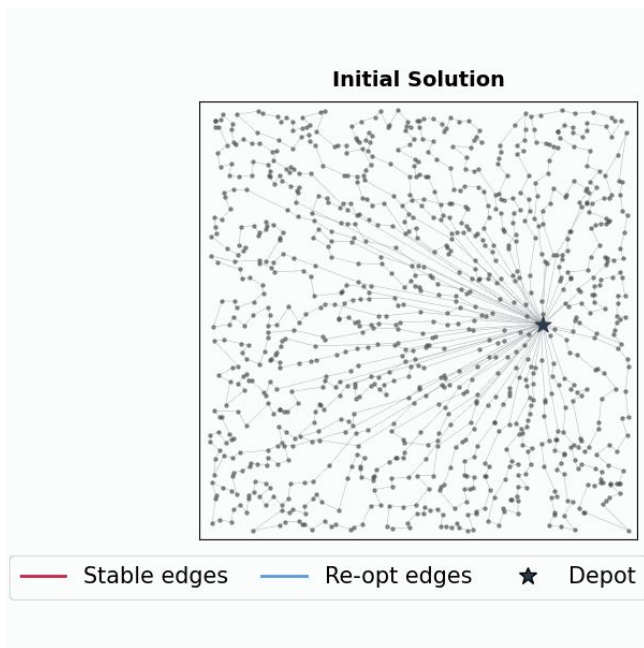
- Learning to **elevate existing solver**
- Still **retain the guarantee** of the OR solver
- **Generalize** better to distribution shifts
- Impactful for practical industry deployment
- $x\%$  data-driven +  $y\%$  human rules

# Connect the dots



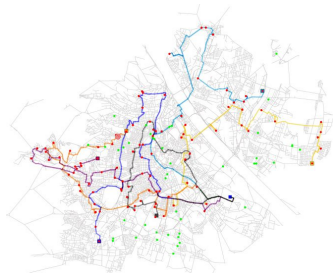
# LGO Example: Remove Redundant Search

**What to guide:** Subproblem Selection for large scale optimization

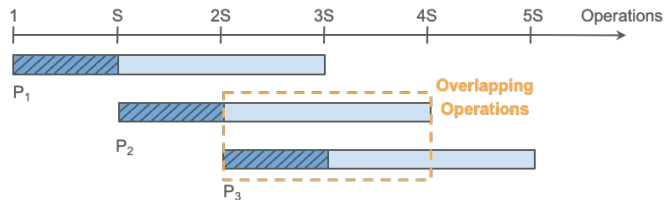


# LGO Example: Remove Redundant Search

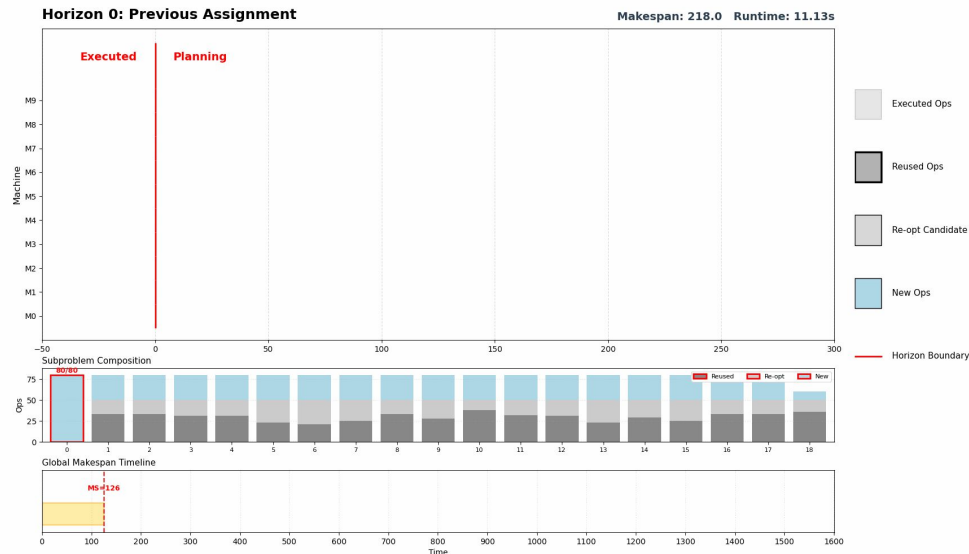
**What to guide:** Fine-grained Subproblem Selection for long-horizon optimization



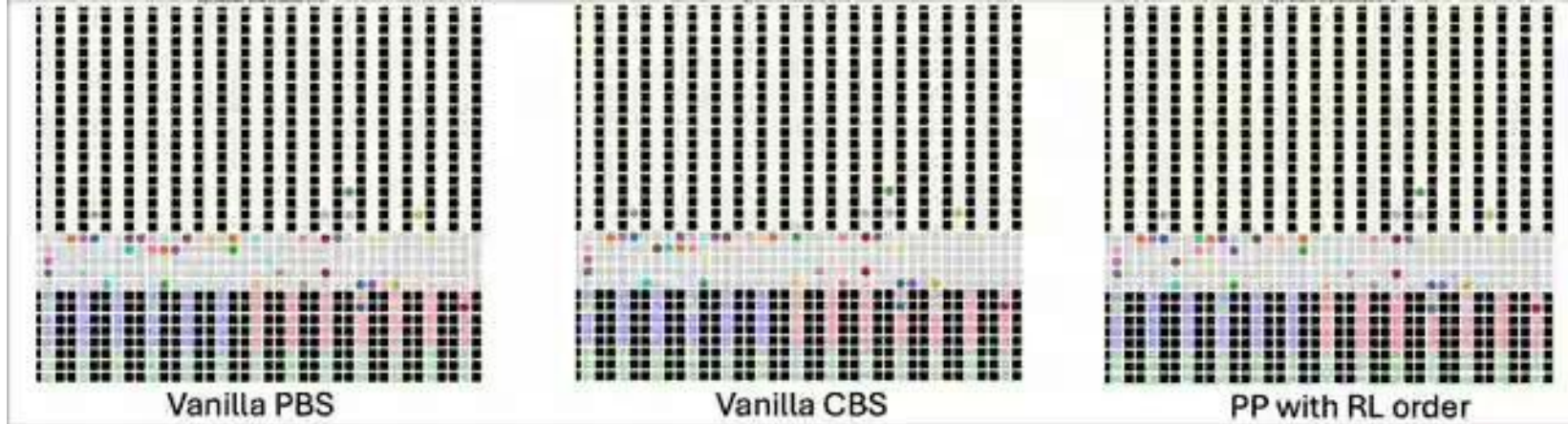
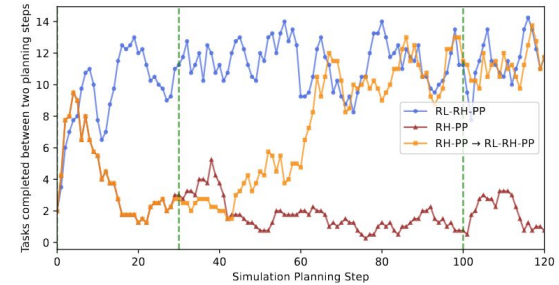
L2Seg: Spatial Decomposition



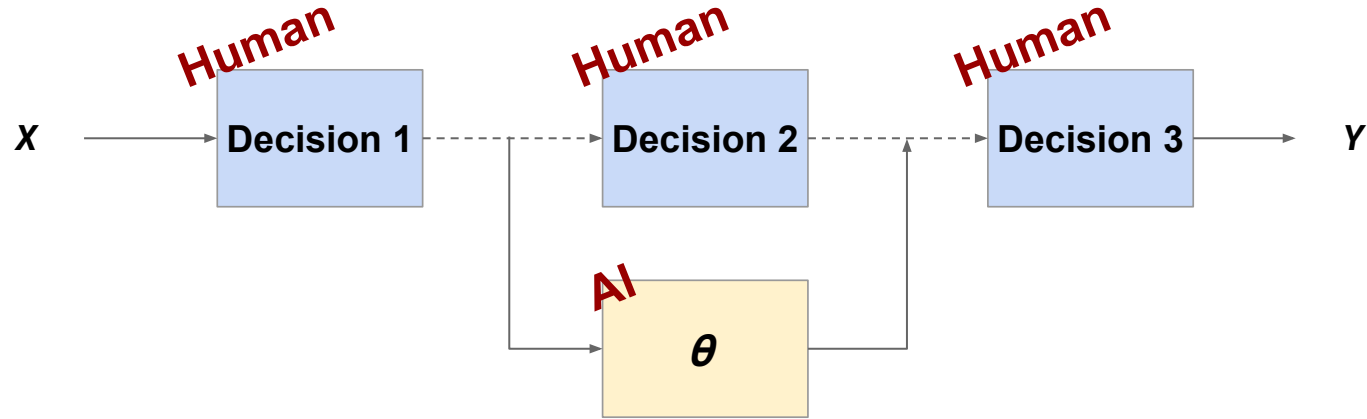
This work: Learn to accelerate Temporal Decomposition based optimization



# LGO Example: Remove Myopic Decision



# How to LGO?



$RQ1 \rightarrow RQ2 \rightarrow RQ3 \rightarrow RQ4 \rightarrow$   
Stage 1: Pre-Learning Analysis

Find the target 'Decision 2'

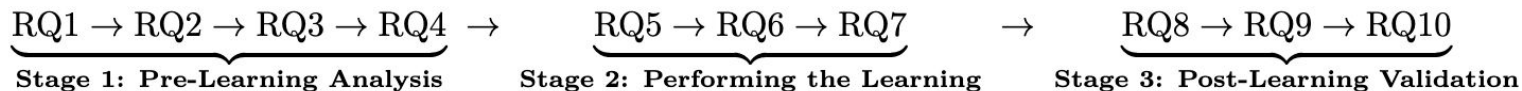
$RQ5 \rightarrow RQ6 \rightarrow RQ7 \rightarrow$   
Stage 2: Performing the Learning

Learn the neural network  
and replace 'Decision 2'

$RQ8 \rightarrow RQ9 \rightarrow RQ10$   
Stage 3: Post-Learning Validation

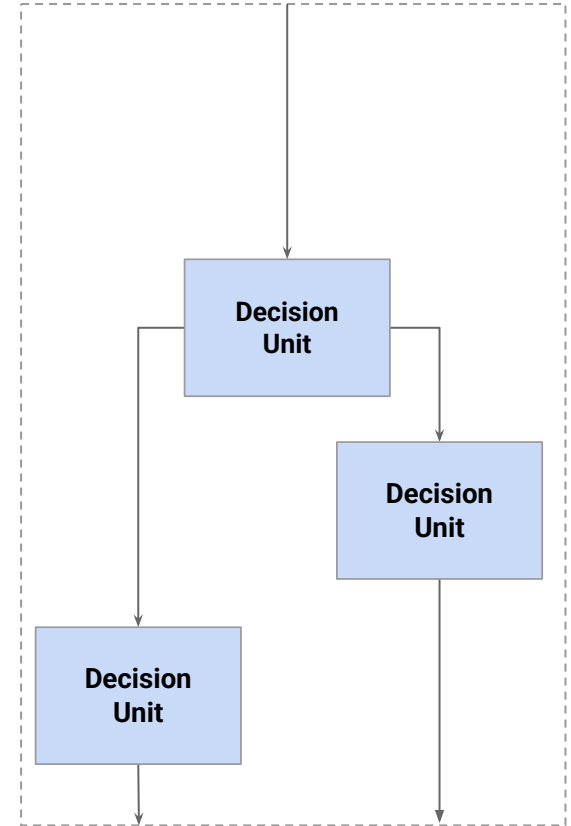
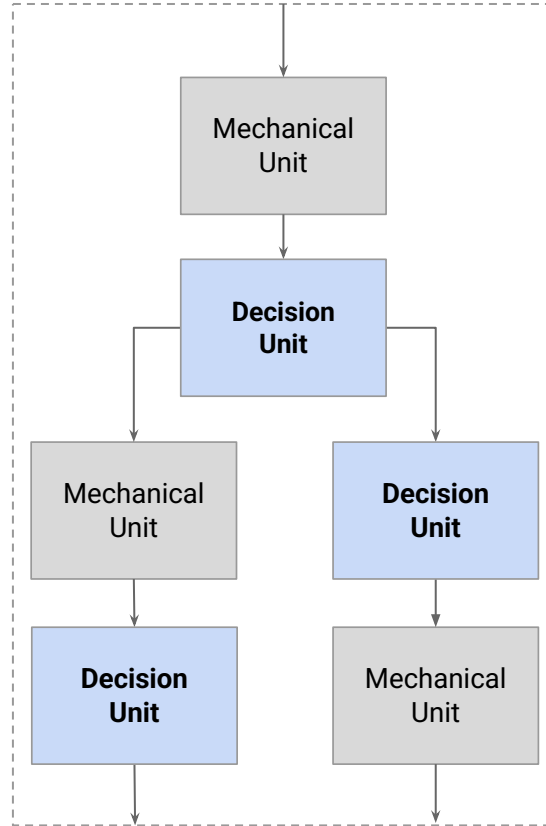
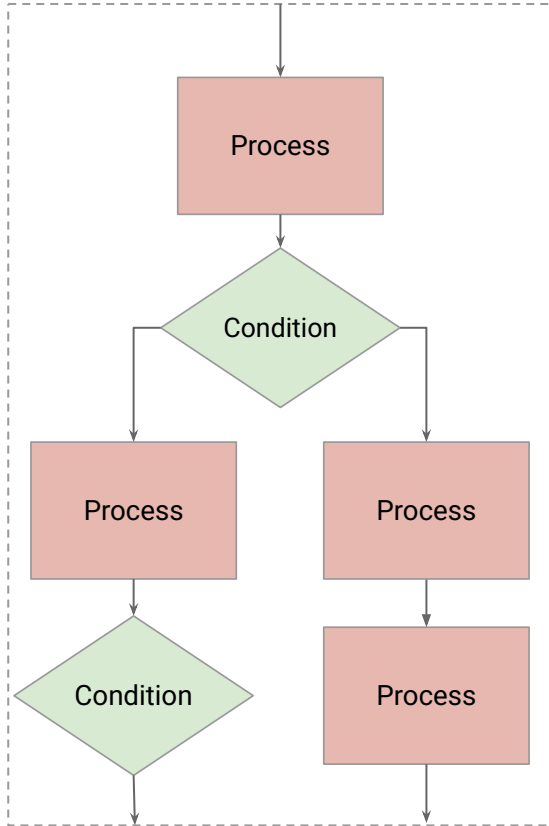
Analyze the impact after  
replacement

# How to LGO?

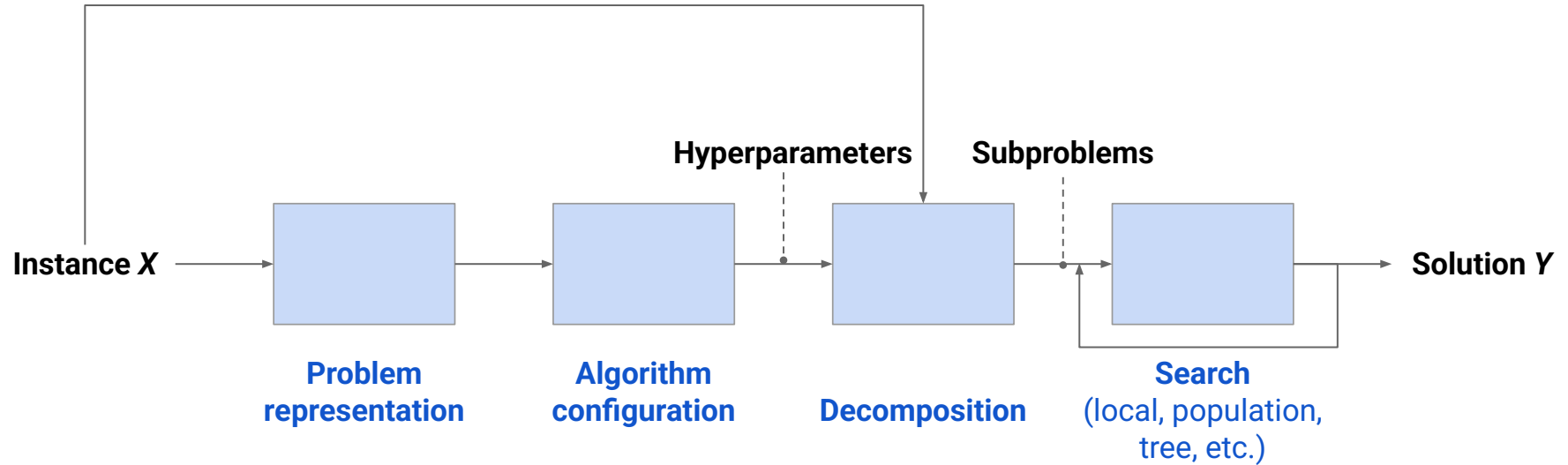


Stage 1	RQ1	What are the decisions within the solver?
	RQ2	Which decisions are potentially suboptimal?
	RQ3	How to formulate the ML task?
	RQ4	Is the ML task worth learning?
Stage 2	RQ5	How to collect training data?
	RQ6	How to design the Deep Learning pipeline?
	RQ7	How to design the Inference method?
Stage 3	RQ8	What has the model learned?
	RQ9	Why the learned model outperform?
	RQ10	What are the next steps for real-world deployment?

# RQ1 - What are the decisions within the solver?



# A typical discrete optimization solver



# RQ2 - Which decisions are potentially suboptimal?

## Decomposition



$$s_t \xrightarrow{o_t(\phi_t, \pi_t), \mathcal{T}} s_{t+1} = s_t \xrightarrow{\text{extract } \phi_t} z_t \xrightarrow{\text{decide } \pi_t} a_t \xrightarrow{\text{execute } \mathcal{T}} s_{t+1}$$

### Poor Feature Extraction $\phi$

- Overly simplistic

### Poor Decision Policy $\pi$

- Low expressiveness
- Myopic
- Too-specific

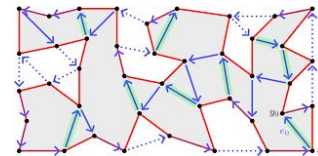
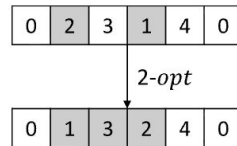
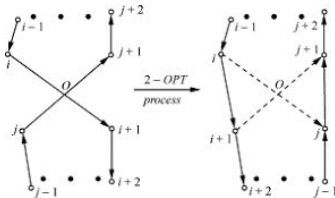
### Execution Bottlenecks $\mathcal{T}$

- Cause future computation bottleneck

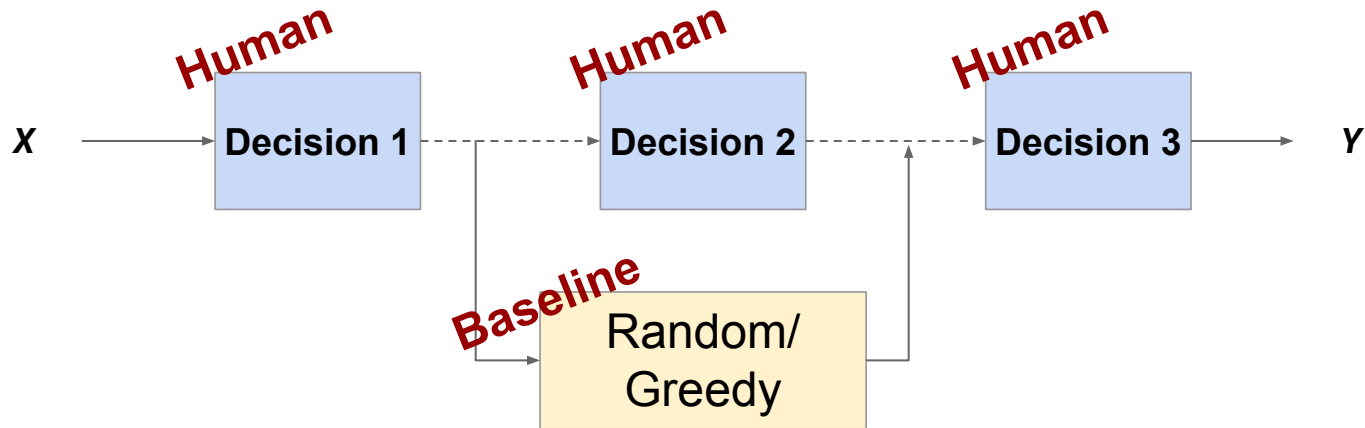
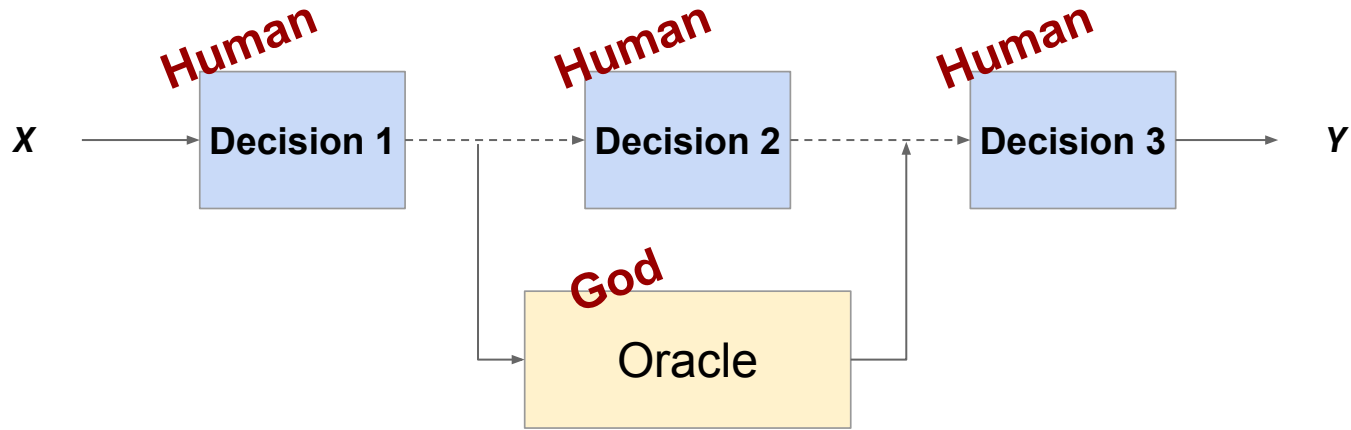
# RQ3 - How to formulate the ML task?

Action type	Effective space size	Example
Selection (1-of- $N$ )	$N$	Branching variable, neighborhood selection
Subset ( $k$ -of- $N$ )	$\binom{N}{k}$ , up to $2^N$	Destroy set in LNS, variable fixing
Ordering	$N!$	Priority ordering for planning
Value prediction	continuous (scalar)	Bound estimation, cost prediction
Generation	structured, exponential in length	Solution construction, cut generation

**Reminder: Action decomposition for exponentially increasing state/action space**



# RQ4 - Is the ML task worth learning?



# RQ4 - Is the ML task worth learning?

## Look-ahead Oracle

- Starting from the decision points
- Oracle computes:
  - Enumerate all possible decisions and observe the consequence
  - Get the best decision
- Execute the Oracle action and discard any time/resources used

You may also inject errors to the Oracle to understand the future learning performance

	<b>Oracle (LNS) + perfect recall &amp; TNR</b>	<b>Oracle (LNS) + 95% recall &amp; 95% TNR</b>	<b>Oracle (LNS) + 90% recall &amp; 90% TNR</b>	<b>Oracle + 70% recall &amp; 70% TNR</b>	<b>Ref (L2Seg-SYN-LNS)</b>
<b>Obj.</b>	56.02	56.01	56.02	56.04	56.08
<b>Time</b>	39s	62s	119s	324s	241s

# 04

## LLM and Agentic AI (2023 - now)

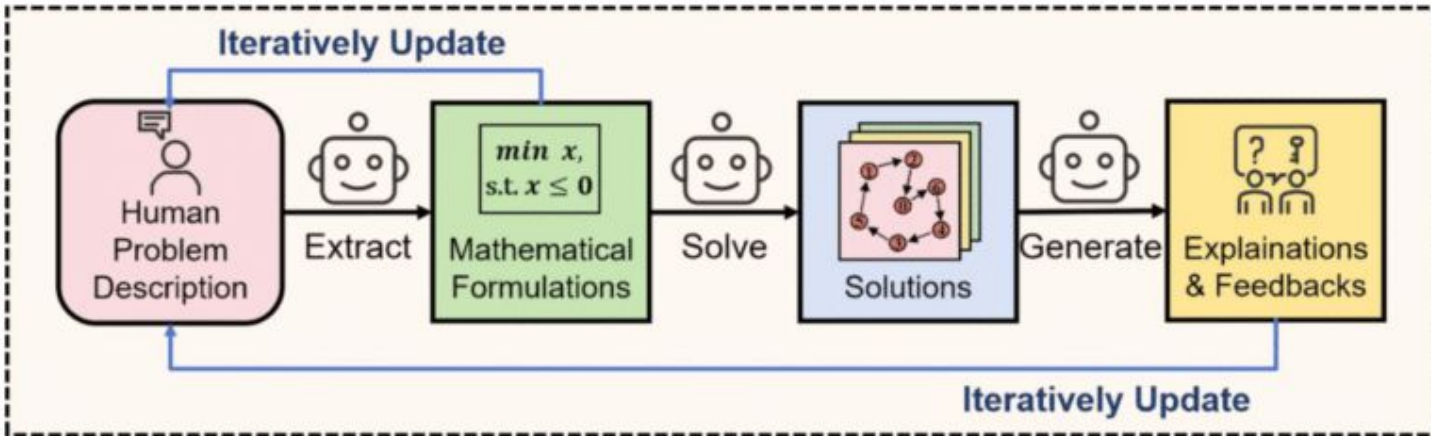
# Towards ChatOpt

MODEL

OPTIMIZATION

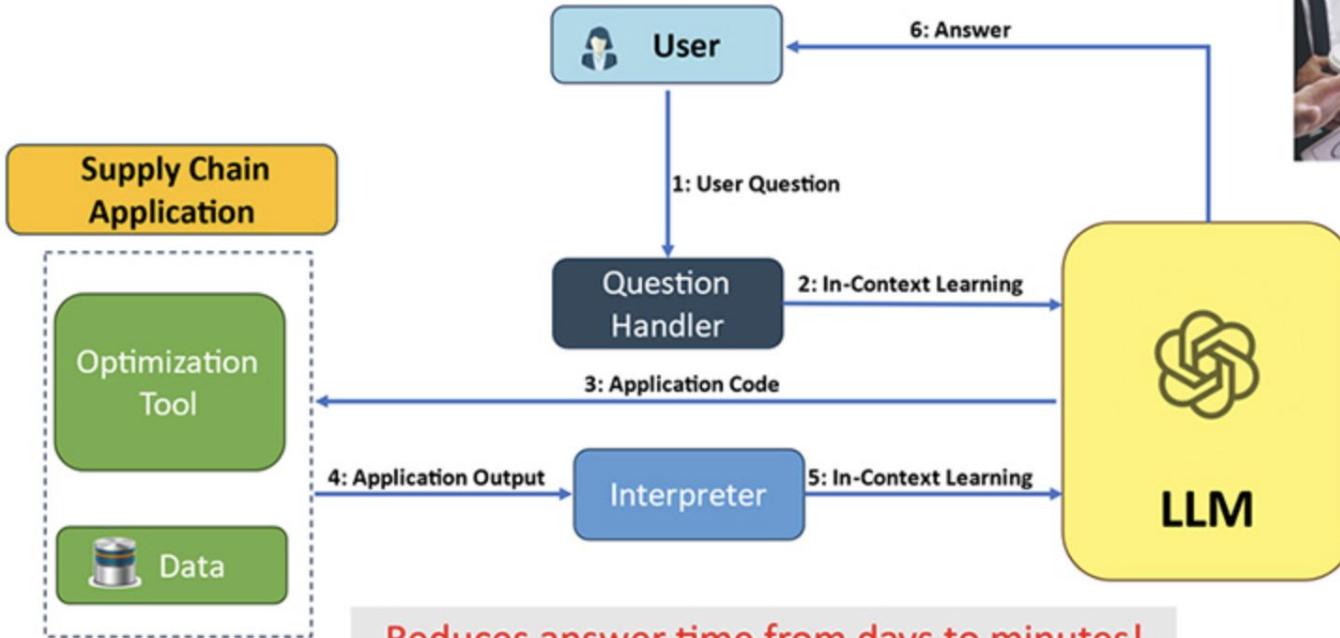
DECISION

3. LLM and Agentic AI



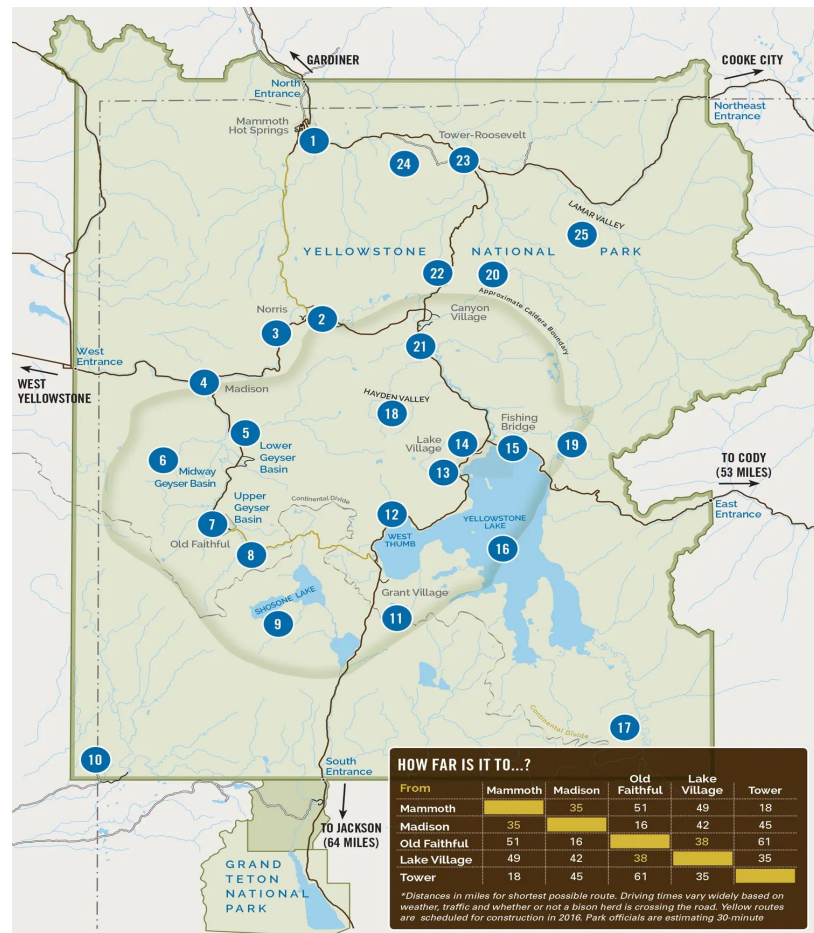
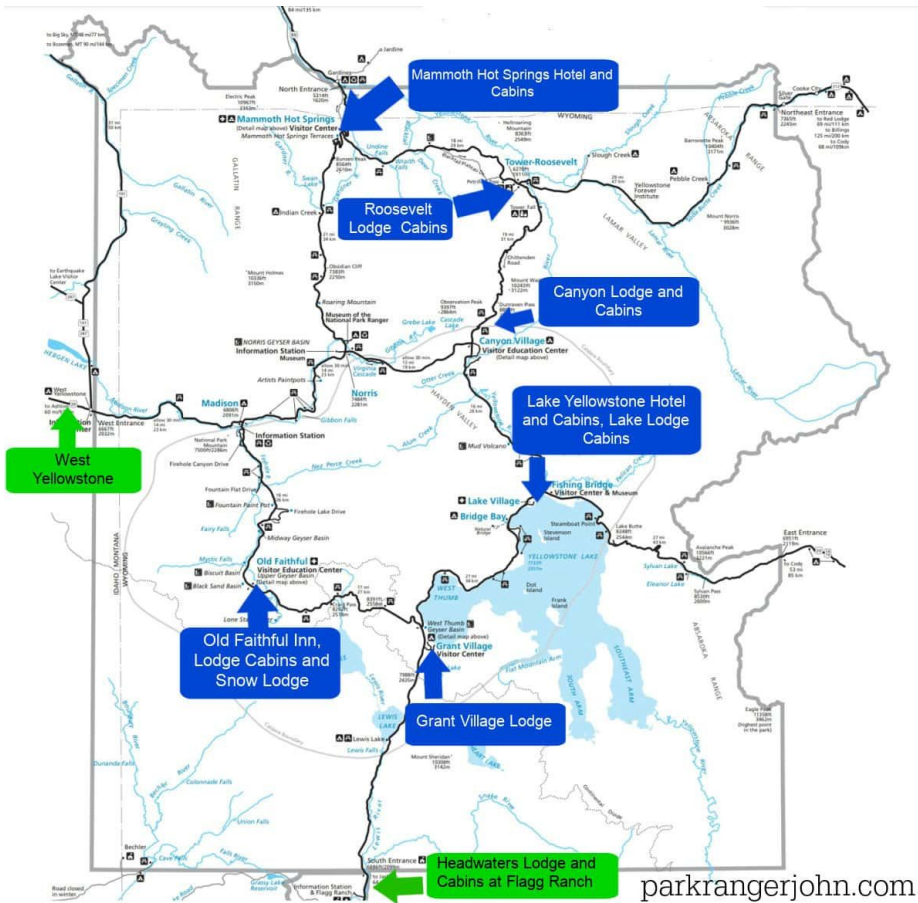
# OptiGuide by Microsoft Research

- Why is demand X not fulfilled?
- What if we use supplier Y for demand X?



**Reduces answer time from days to minutes!**

# LLM Models Your Yellowstone Trip



# LLM Models Your Yellowstone Trip

## Gemini Prompt

Please assist me in formulating and solving a mathematical model for an M-day travel itinerary within Yellowstone National Park. I will provide you with the instance data:

1. Node IDs: Points of Interest (POIs) + two depot locations + Hotels
2. Service Time (at each POI).
3. Travel Distance (simplified using the K-Nearest Neighbors approach).
4. Daily Hotel Rate

The itinerary must satisfy the following constraints:

1. each day involves visiting a specific number of POIs before returning to the hotel for rest; on subsequent days, travel starts from the same hotel;
  2. the journey begins by entering through one designated depot and concludes by exiting through the other.
  3. the total daily travelling time must be less than x hours
- The objective is to perform a single-objective optimization based on a weighted average of the total travel distance (shortest path) and the total cost (lowest price). Please provide me with a detailed modeling scheme.

## Gemini Replies

This is a classic variation of the **Tourist Trip Design Problem (TTDP)**, which is fundamentally rooted in the Multi-Depot Vehicle Routing Problem with Time Windows and Intermediate Stops.

By applying a K-Nearest Neighbors (K-NN) filter, you are effectively using **graph sparsification**. This is an excellent strategy because it drastically reduces the number of decision variables, making the Mixed-Integer Linear Programming (MILP) model much faster for a solver (like Gurobi or CPLEX) to compute.

Here is the detailed mathematical modeling scheme for your Yellowstone itinerary.

### 1. Sets and Indices

- $M$ : Total number of days for the trip. The day index is  $m \in \{1, 2, \dots, M\}$ .
- $P$ : Set of all Points of Interest (POIs).
- $H$ : Set of all candidate hotels.
- $d_{in}$ : The starting depot (entry point on Day 1).
- $d_{out}$ : The ending depot (exit point on Day  $M$ ).
- $N$ : Set of all nodes in the network, where  $N = P \cup H \cup \{d_{in}, d_{out}\}$ .
- $i, j$ : Node indices, where  $i, j \in N$ .

### 2. Parameters

- $t_{ij}$ : Travel distance/time from node  $i$  to node  $j$ . (Due to your K-NN simplification, if  $j$  is not in the K-NN list of  $i$ ,  $t_{ij} = \infty$ , or the edge is simply excluded from the model).
- $s_i$ : Service time (visiting duration) at node  $i$ . For hotels and depots,  $s_i = 0$ .
- $C_h$ : The daily rate (price) of hotel  $h \in H$ .
- $T_{max}$ : The maximum allowable daily travel time ( $x$  hours).
- $V_m$ : The specific number of POIs you wish to visit on day  $m$ .
- $\alpha, \beta$ : User-defined weights for the multi-objective function ( $\alpha$  for distance,  $\beta$  for cost).
- $M_{big}$ : A sufficiently large positive constant (Big-M) used for linearizing constraints.

### 4. Objective Function

The goal is to minimize the weighted sum of the total travel distance and the total hotel cost. To ensure the solver treats these fairly, you may need to normalize the distance and cost values beforehand so they exist on a similar scale.

$$\min Z = \alpha \sum_{m=1}^M \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij}^m + \beta \sum_{m=1}^{M-1} \sum_{h \in H} C_h y_h^m$$

### 5. Constraints

#### Constraint 1: Daily POI Visitation Count

Ensure that on each day  $m$ , exactly  $V_m$  POIs are visited.

$$\sum_{i \in P} \sum_{j \in N} x_{ij}^m = V_m \quad \forall m \in \{1, \dots, M\}$$

#### Constraint 2: Unique POI Visitation

Ensure that no POI is visited more than once across the entire trip (assuming you do not want to revisit the same attraction).

$$\sum_{m=1}^M \sum_{j \in N} x_{ij}^m \leq 1 \quad \forall i \in P$$

#### Constraint 3: Flow Conservation

If a traveler arrives at a POI on day  $m$ , they must also depart from that same POI on day  $m$ .

$$\sum_{i \in N} x_{ij}^m = \sum_{k \in N} x_{jk}^m \quad \forall j \in P, \forall m \in \{1, \dots, M\}$$

#### Constraint 4: Start and End Depots

The journey must start at  $d_{in}$  on Day 1 and strictly end at  $d_{out}$  on Day  $M$ .

$$\sum_{j \in P \cup H} x_{d_{in},j}^1 = 1$$

$$\sum_{i \in P \cup H} x_{i,d_{out}}^M = 1$$



[solver] Status: OPTIMAL

=====  
Day 1  
=====

#	Location	Arrive	Stay	Depart	Drive
1	west_entrance	0.00h	-	0.00h	0.90h
2	norris	0.90h	2.0h	2.90h	0.70h
3	mammoth	3.60h	1.8h	5.35h	0.60h
4	tower_falls	5.95h	0.8h	6.70h	0.05h
5	roosevelt_lodge_dining	8.45h	1.5h	9.95h	0.05h
6	roosevelt_lodge	10.00h	-	10.00h	

Day 1 total drive: 2.30h  
Hotel tonight: roosevelt\_lodge (\$450.00/night)

=====  
Day 2  
=====

#	Location	Arrive	Stay	Depart	Drive
1	roosevelt_lodge	0.00h	-	0.00h	0.65h
2	yellowstone_canyon	0.65h	3.0h	3.65h	0.55h
3	mud_volcano	4.20h	1.0h	5.20h	0.15h
4	fishing_bridge	7.15h	0.8h	7.90h	0.70h
5	west_thumb	8.60h	1.2h	9.85h	0.15h
6	grant_village	10.00h	-	10.00h	

Day 2 total drive: 2.20h  
Hotel tonight: grant\_village (\$640.00/night)

=====  
Day 3  
=====

#	Location	Arrive	Stay	Depart	Drive
1	grant_village	0.00h	-	0.00h	0.05h
2	grant_village_lake_house	0.05h	1.5h	1.55h	0.70h
3	old_faithful_snow_lodge_dining	2.25h	1.5h	3.75h	0.05h
4	old_faithful_basin	4.00h	4.0h	8.00h	0.50h
5	river_downstream	8.50h	1.0h	9.50h	0.50h
6	south_entrance	10.00h	-	10.00h	

Day 3 total drive: 1.80h

=====  
Summary  
=====

Total travel time : 6.30h  
Total hotel cost : \$1090.00  
Objective value : 115.30

[solver] Status: OPTIMAL

=====  
Day 1  
=====

#	Location	Arrive	Stay	Depart	Drive
1	west_entrance	0.00h	-	0.00h	0.50h
2	river_downstream	0.50h	1.0h	1.50h	0.50h
3	old_faithful_basin	2.00h	4.0h	6.00h	0.05h
4	old_faithful_snow_lodge_dining	7.85h	1.5h	9.35h	0.65h
5	grant_village	10.00h	-	10.00h	

Day 1 total drive: 1.70h  
Hotel tonight: grant\_village (\$640.00/night)

=====  
Day 2  
=====

#	Location	Arrive	Stay	Depart	Drive
1	grant_village	0.00h	-	0.00h	0.05h
2	grant_village_lake_house	0.05h	1.5h	1.55h	0.20h
3	west_thumb	1.75h	1.2h	3.00h	0.70h
4	fishing_bridge	3.70h	0.8h	4.45h	0.15h
5	mud_volcano	4.80h	1.0h	5.80h	0.55h
6	yellowstone_canyon	6.35h	3.0h	9.35h	0.65h
7	roosevelt_lodge	10.00h	-	10.00h	

Day 2 total drive: 2.30h  
Hotel tonight: roosevelt\_lodge (\$450.00/night)

=====  
Day 3  
=====

#	Location	Arrive	Stay	Depart	Drive
1	roosevelt_lodge	0.00h	-	0.00h	0.05h
2	roosevelt_lodge_dining	0.05h	1.5h	1.55h	0.05h
3	tower_falls	1.60h	0.8h	2.35h	0.60h
4	mammoth	4.65h	1.8h	6.40h	0.70h
5	norris	7.10h	2.0h	9.10h	0.90h
6	south_entrance	10.00h	-	10.00h	

Day 3 total drive: 2.30h

=====  
Summary  
=====

Total travel time : 6.30h  
Total hotel cost : \$1090.00  
Objective value : 115.30

# LLM Improves Your Solvers

The screenshot displays the 'Automated Discovery' interface for the 'agile-fox' benchmark. The top navigation bar includes 'LIVE', 'IDEAS', 'DIVERSITY', and 'BENCHMARK' tabs. The main dashboard shows the current score of 6791.2, which is a 0.29119% improvement over the previous best and a 1.67% improvement over the Best Known Solution (BKS) of 6679.8. A network graph visualizes the route structure. The 'BENCHMARK PROGRESS' chart shows the performance of various agents over time. The 'LIVE FEED' section lists several proposals from 'crystal-forge', such as 'Cross-exchange segment swap (1-2 len)' and 'Route elimination via cheapest reinsertion'. The 'LEADERBOARD' lists seven agents: agile, prima, silve, sharp, prima, bright, and vivid.

Automated Discovery **LIVE** IDEAS DIVERSITY BENCHMARK

AGENTS 29 EXPERIMENTS 230 IMPROVEMENT -28.7%

ROUTES  
agile-fox

BEST 25/25 · LATEST  
C2\_4\_2 (4/24)

SCORE  
**6791.2**  
-0.29119% vs prev best  
+1.67% vs BKS (6679.8)

ROUTE DISTANCE  
**3985.9**

LIVE FEED

- 21:50:57 crystal-forge proposed: "Cross-exchange segment swap (1-2 len)"
- 21:50:57 crystal-forge proposed: "Route elimination via cheapest reinsertion"
- 21:50:57 crystal-forge proposed: "ALNS ruin-and-recreate with regret-2"
- 21:50:57 crystal-forge proposed: "Multi-operator ALNS (shaw/random/worst)"
- 21:50:57 crystal-forge proposed: "ALNS with SA acceptance + greedy insertion"
- 21:50:57 crystal-forge proposed: "Stronger polish in ALNS loop"
- 21:50:57 crystal-forge proposed: "Adaptive operator weights (roulette ALNS)"
- 21:50:57 crystal-forge proposed: "Pool logging + or-opt seg 1-3 enabled"
- 21:50:57 crystal-forge proposed: "Higher SA temp + longer ALNS + diversification"

BENCHMARK PROGRESS

GLOBAL

CODE DIVER:

- agil
- prim
- silv
- shar
- prim
- brig

LEADERBOARD:

#	AGENT
1	agile
2	prima
3	silve
4	sharp
5	prima
6	bright
7	vivid

## Join the Swarm

Help a swarm of AI agents collaboratively optimize vehicle routes in real time.

OPEN CLAUDE CODE AND PASTE:

```
Clone https://github.com/SteveDiamond/tig-swarm-demo, read the CLAUDE.md, and start contributing
```

Click anywhere to close · press J to reopen

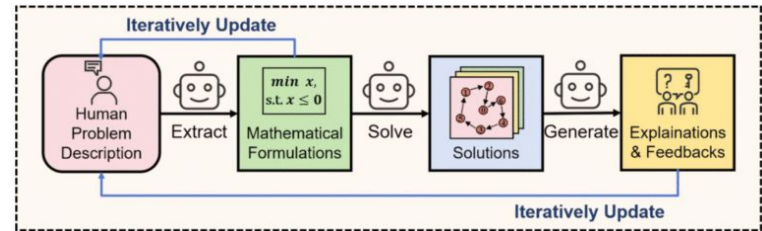
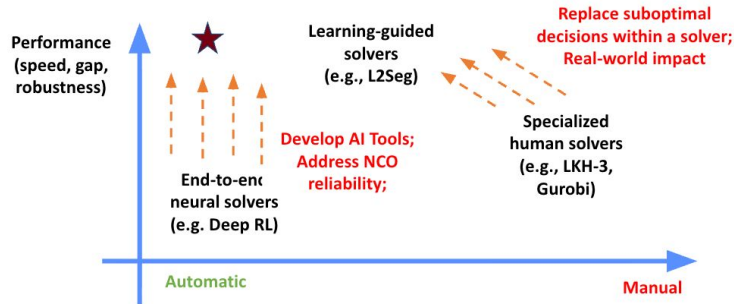
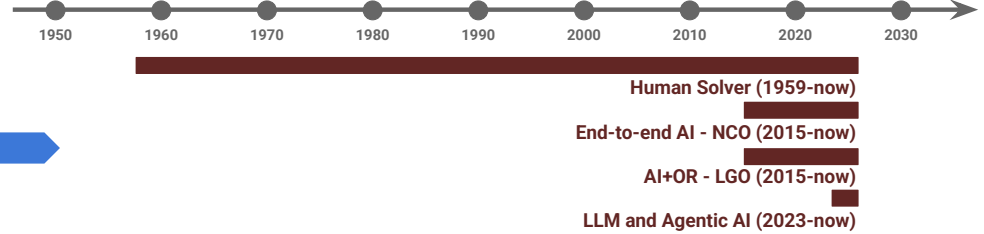
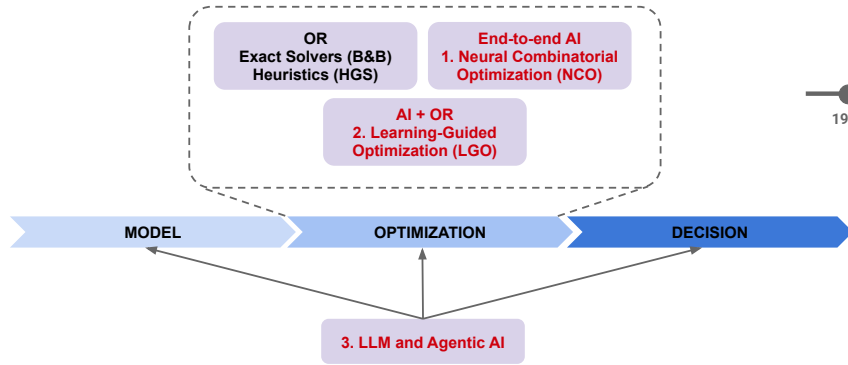
# LLM Explains Your Solver Results



RouteExplainer: Application Demo

- Interactive Tourist Route Generation -

# Takeaways





# Thank you!!

Dr. Yining Ma ([yiningma@mit.edu](mailto:yiningma@mit.edu))

Laboratory for Information & Decision Systems (LIDS)

