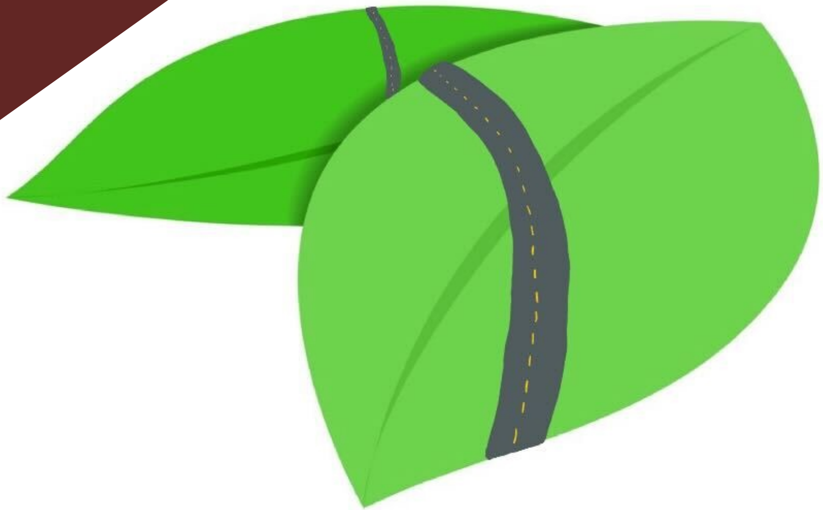
Towards Scalable Learning-based Mobile Coordination

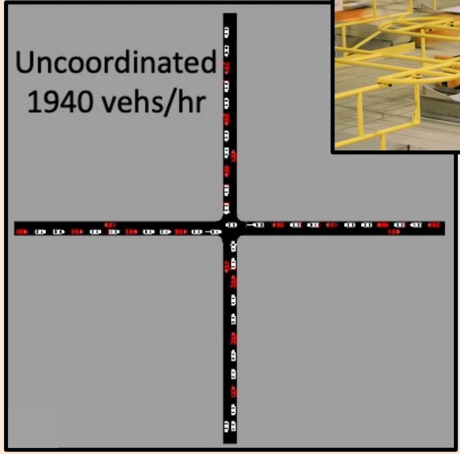
Cathy Wu | Assistant Professor

LIDS, CEE, IDSS



Learning for mobile coordination

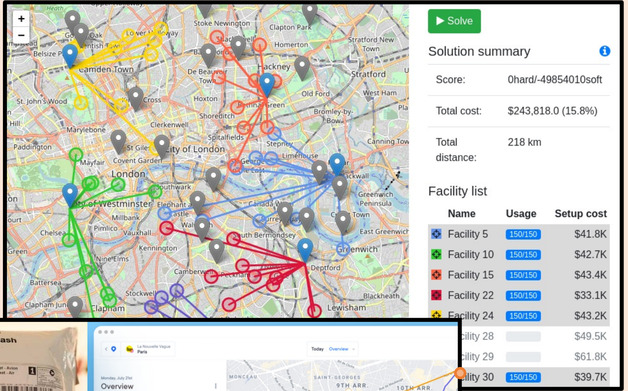
Scale | Topology | Degree of coordination | Type of decision | ...

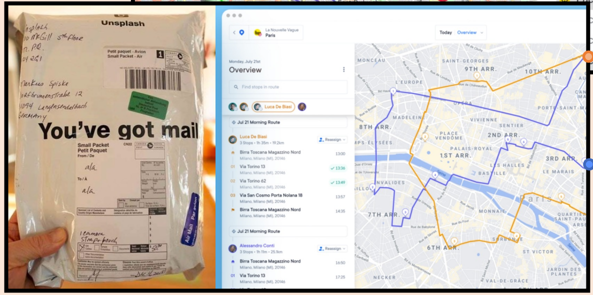


Warehouse automation

Intelligent transportation

Mixed integer linear programming





Vehicle routing

Kinematic coordination ⟷ Discrete coordination

Fundamental research question:

How to effectively design systems in the face of growing complexity?

Inspiration: Learning for decision making

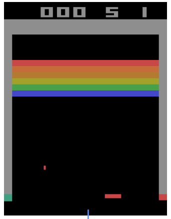
Atari 2600 games

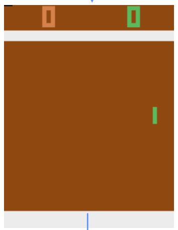
100+ games
- Not too similar
- Not too different

General purpose “meta” method
Deep Q Networks (DQN)

DQN DQN DQN DQN

Train





Breakout AI   Space Invaders AI   Pong AI   Montezuma’s Revenge AI

Specialized method for problem variant

V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, 2015.

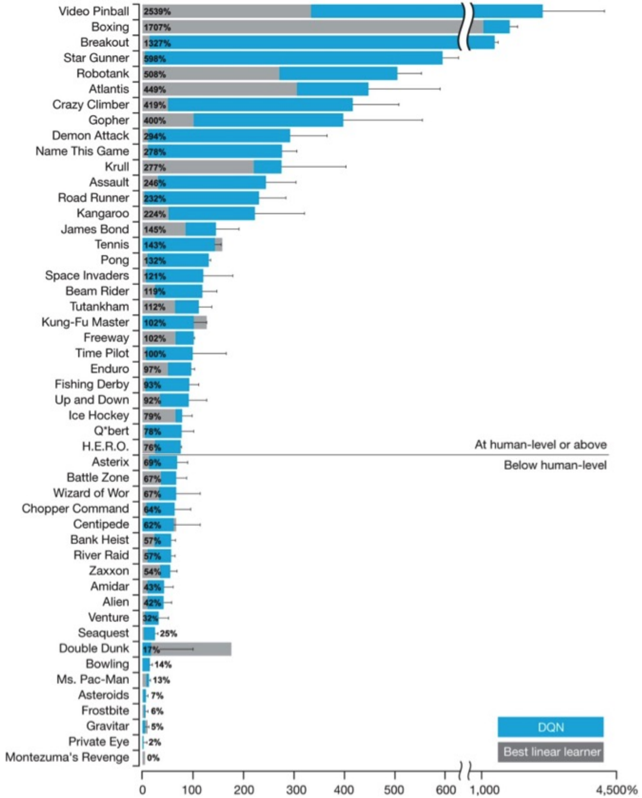


Figure: Automatically learned specialized method vs human player

Wu

Example: vehicle routing problems (VRPs)

VRP variants

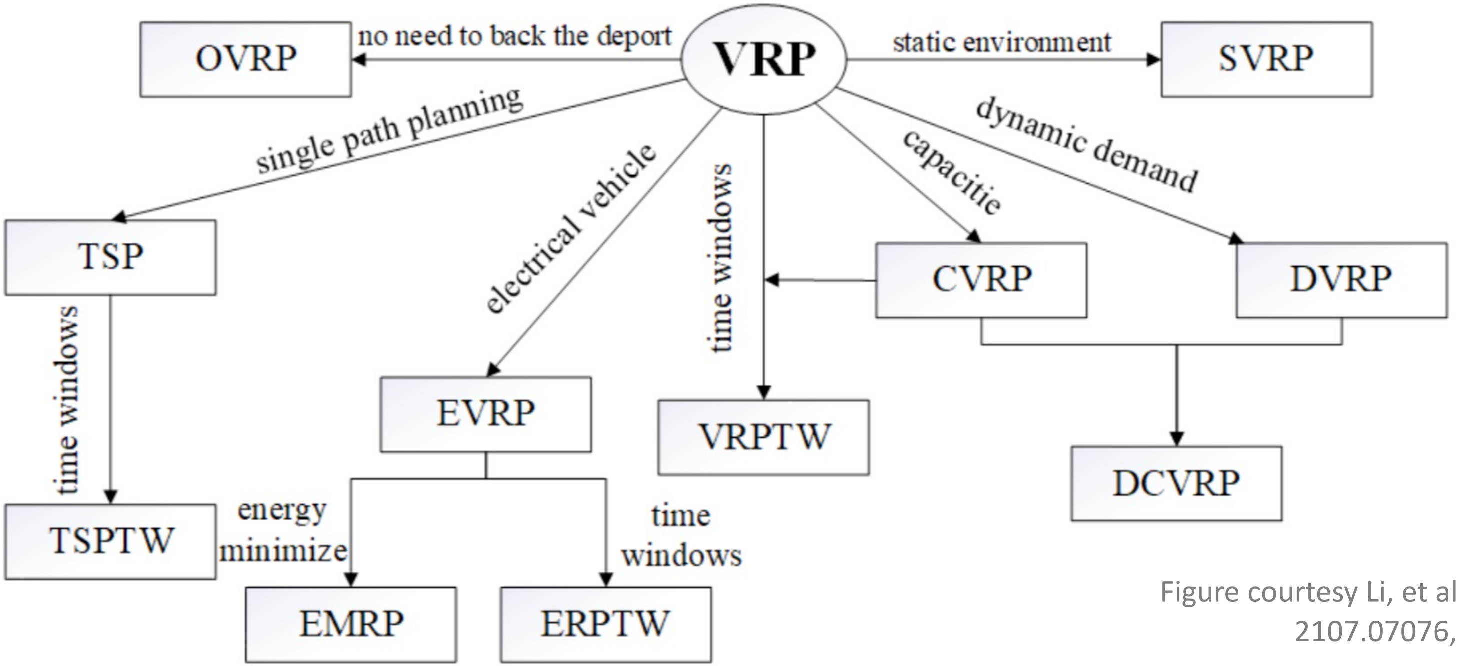
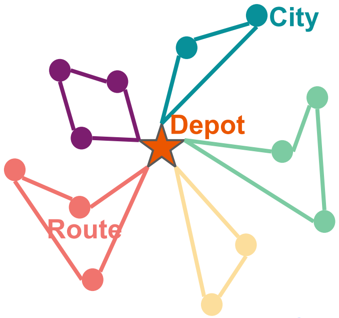


Figure courtesy Li, et al. arXiv 2107.07076, 2021.

Aim: Bring these principles to engineering systems

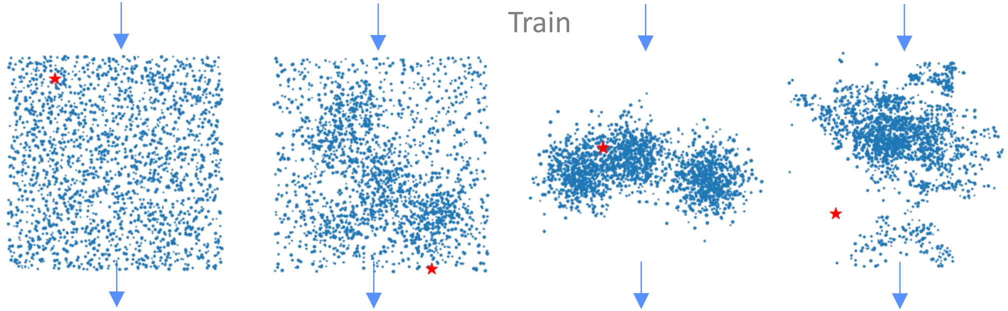


Vehicle routing problems (VRPs)

Dozens of variants
- Not too similar
- Not too different

General purpose “meta” method

??? ??? ??? ???



Capacitated VRP solver | CVRP with Time Windows solver | CVRP w/ Pickup and Delivery | …

Specialized method for problem variant

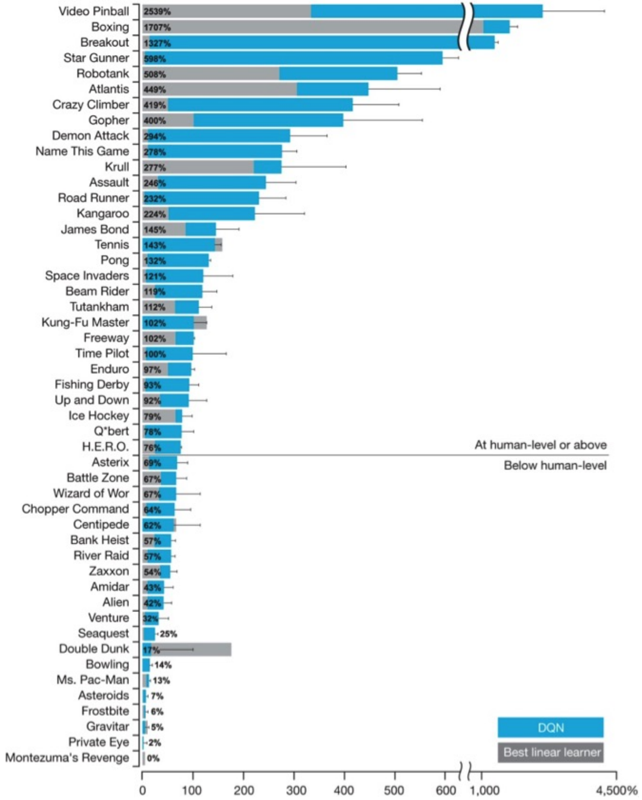
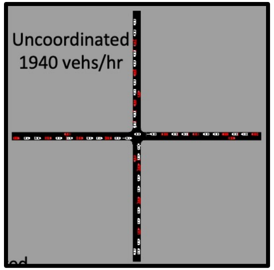


Figure: Automatically learned specialized method vs **heuristic solver**

Aim: Bring these principles to engineering systems



Autonomy-enabled traffic

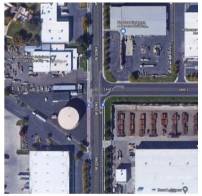
Countless variants

- Network topology
- Autonomy adoption
- Autonomy level
- …

General purpose "meta" method

??? ??? ??? ???

Train



… … … …

Specialized method for problem variant

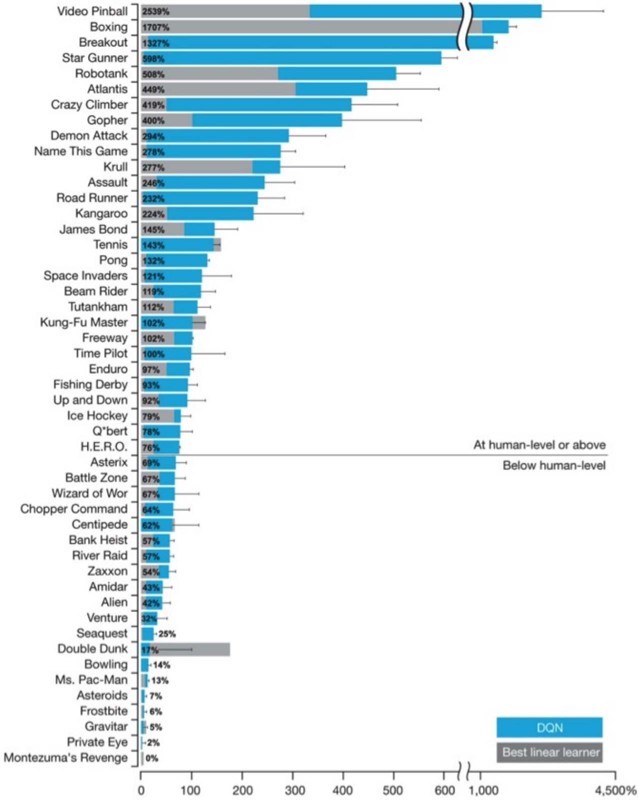
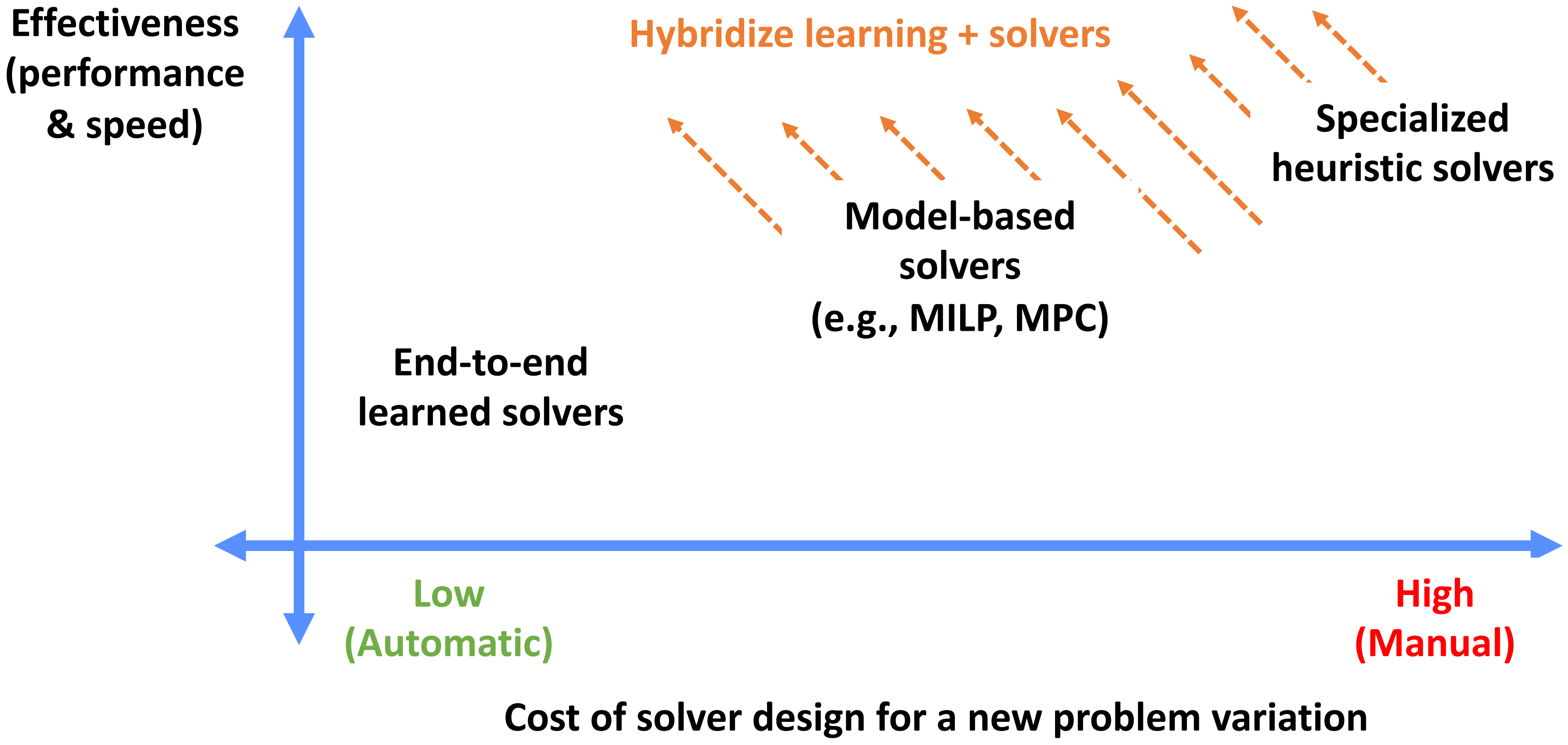
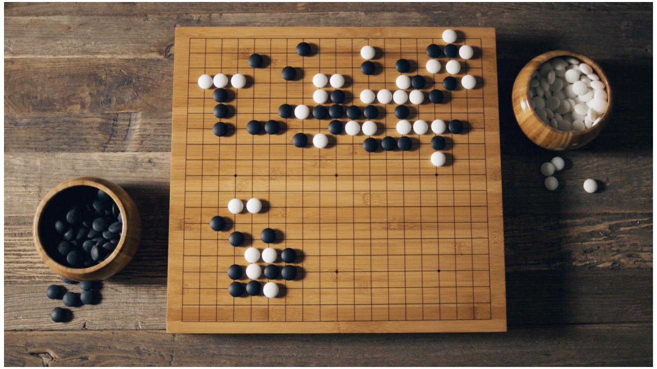


Figure: Automatically learned specialized method vs **heuristic solver**

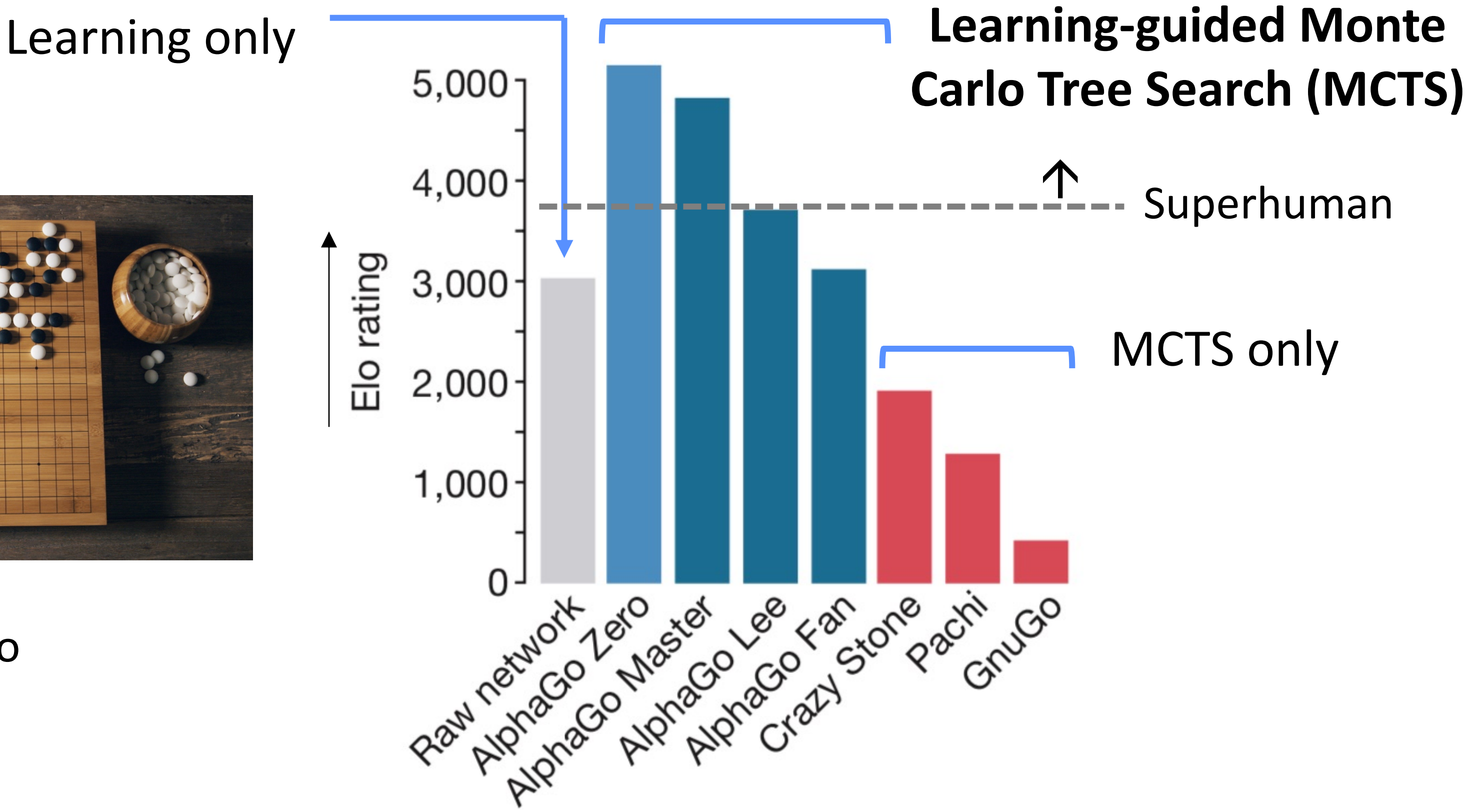
Background on solver design



Learning-guided search: hybridizing learning + solvers

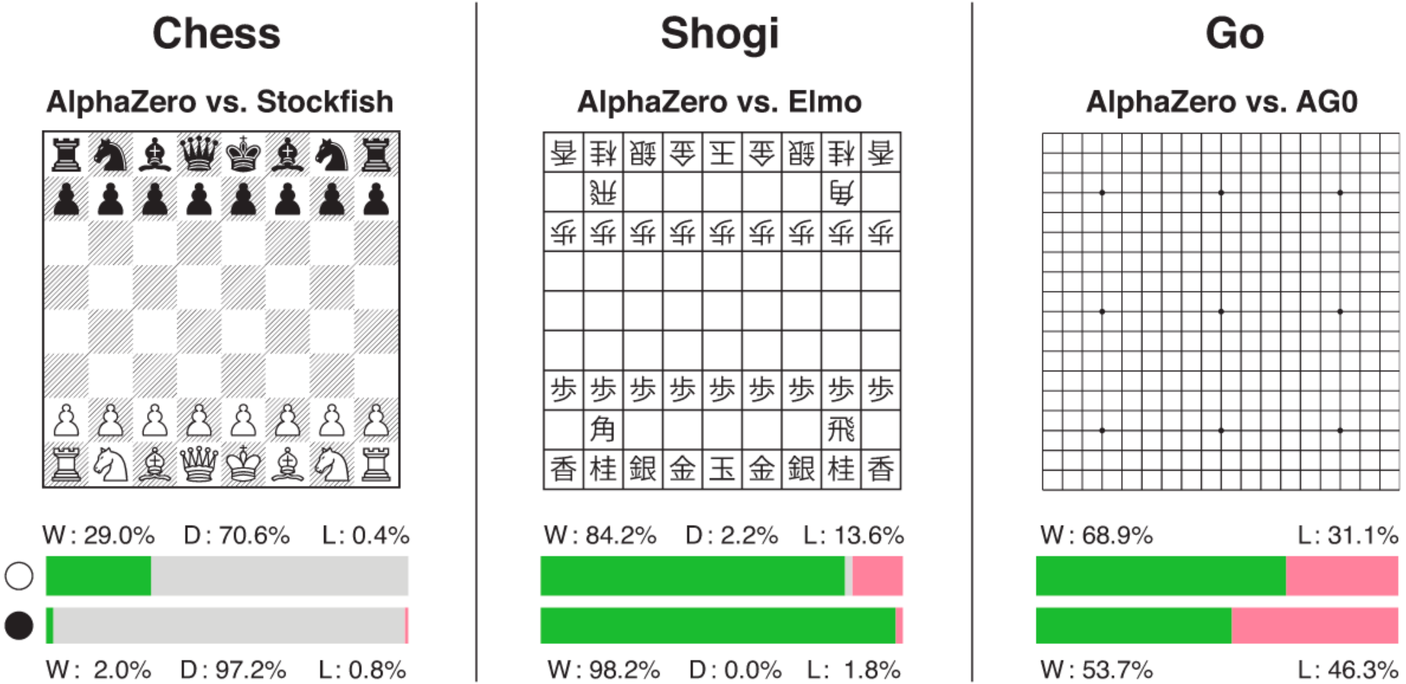


AlphaGo



D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, 2017.

Learning-guided search: hybridizing learning + solvers



- Same as AlphaGo Zero, but without exploiting Go-specific features:
  - No data augmentation due to rotational invariance
  - Allows for more than win/loss (0 = draw)

D. Silver *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play,” *Science*, 2018.

This talk

- **Vision: Cope with growing system complexity with meta-methods**

- Learning-guided search as meta-methods for transportation?
  - Vehicle routing problems
  - Multi-robot warehousing
  - Integer linear programming

- What about autonomous vehicles?

This talk

- Vision: Cope with growing system complexity with meta-methods

- **Learning-guided search as meta-methods for transportation?**
  - **Vehicle routing problems**
  - Multi-robot warehousing
  - Integer linear programming

- What about autonomous vehicles?

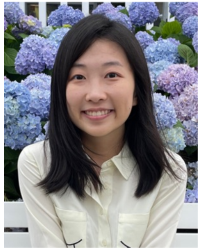
NEURAL INFORMATION PROCESSING SYSTEMS

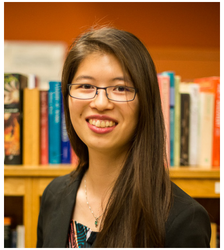
Learning to Delegate for Large-scale Vehicle Routing

NeurIPS 2021

Spotlight (<3%)

Sirui Li*, Zhongxia Yan*, **Cathy Wu**





Example: vehicle routing problems (VRPs)

VRP variants

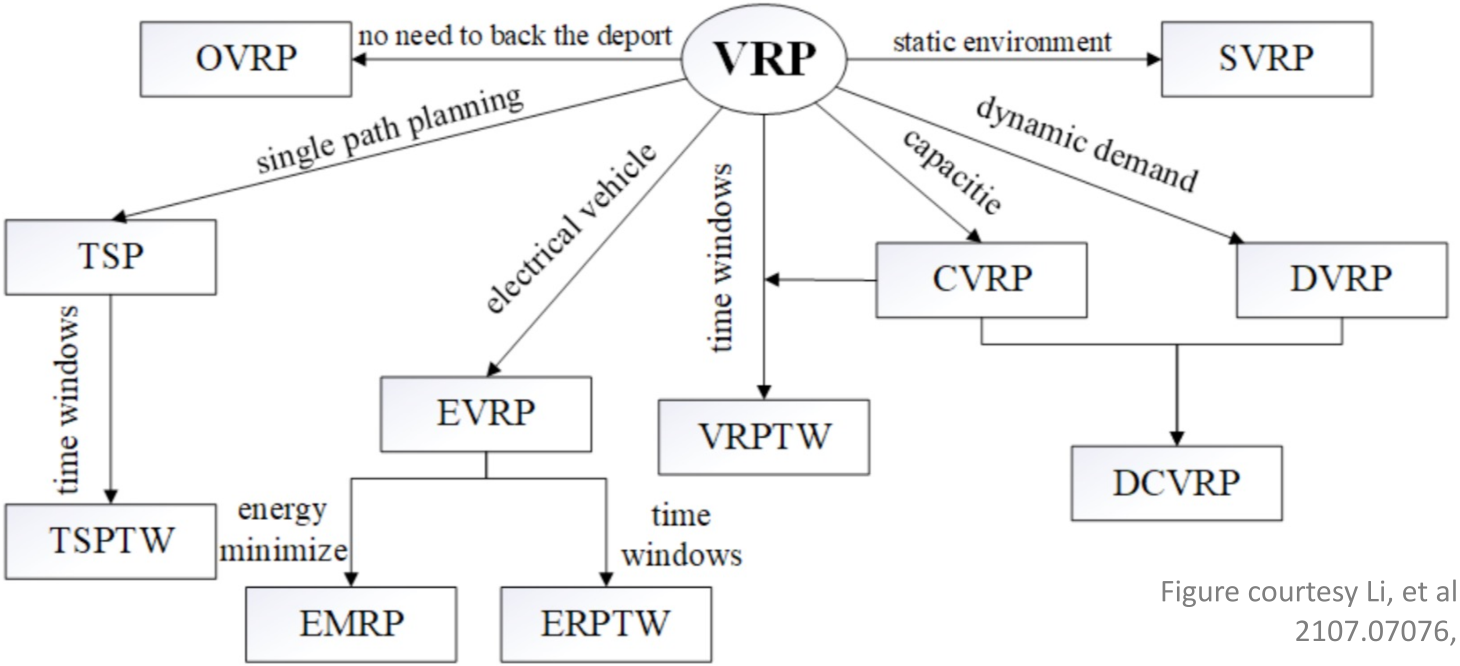


Figure courtesy Li, et al. arXiv 2107.07076, 2021.

General problem formulation: contextual decision

- Problem setting: contextual MDP, generalization of standard MDP, adding a context space $\mathcal{C}$ to capture heterogeneity within a problem class
- Desired:

$$\pi_c^*(\mathrm{s}) = \arg\max_{a \in A_c} Q_c^*(s, a),$$
$$c \in \mathcal{C}$$

Notation:
- Policy π
- State s, Action a
- Action space A
- Value function Q

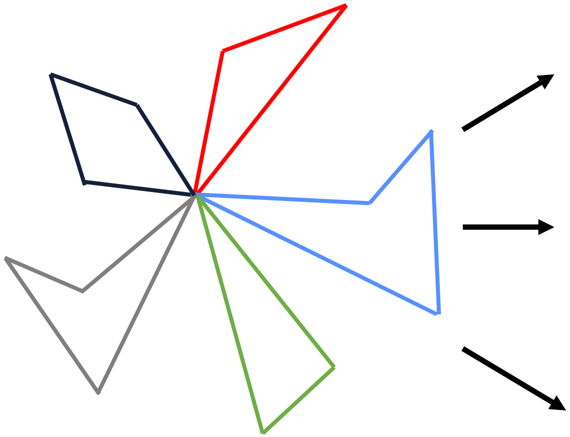
- Standard MDP: $M \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho)$ = (state space, action space, transition function, reward function, initial state distribution)
- Value function $Q_c^*(s, a)$ denotes the long-term reward of a state and action

Decomposition strategies

- Consider: Large transportation problems
- Large problems can often be decomposed into smaller problems
- MDP formulation
  - State s: customer locations & demands, current solution
  - Action a: solve subproblem with subsolver
    - $|A| \approx 200$
  - Context c: variation of VRP
    - Customer distributions
    - Constraints, e.g., capacity, time windows
- Ideal:

$$\pi^*(s) = \arg\max_{\boxed{a \in A(s)}} \boxed{Q^*}(s,a)$$

- Challenge: Expensive to evaluate



Candidate solution

Subproblem candidates

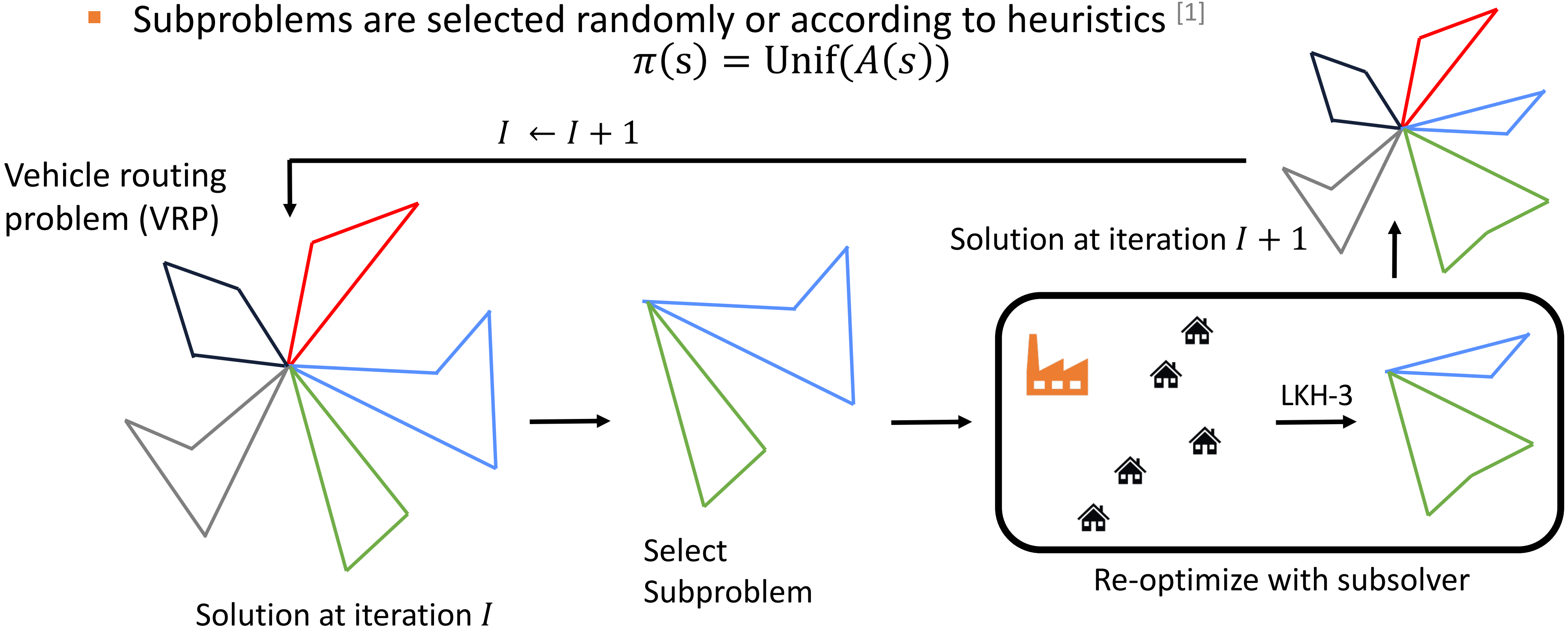
based on **spatial locality**

$O(2^N) \rightarrow O(N)$ subproblems

Large Neighborhood Search (LNS)

- Large problems often decompose into smaller problems
- LNS iteratively seek better solution by re-optimization of subproblems [1]
- Subproblems are selected randomly or according to heuristics [1]

$$\pi(s) = \mathrm{Unif}(A(s))$$

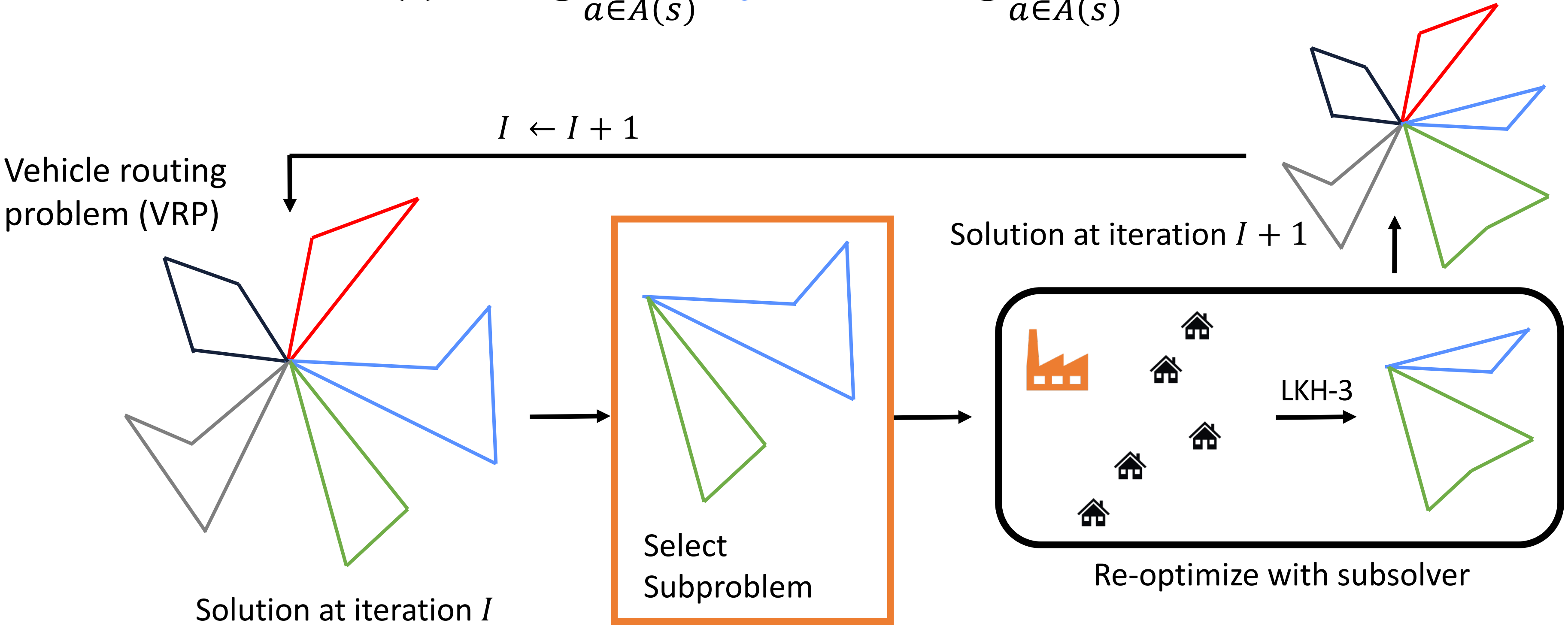


[1] Taillard et al. **Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC).** Springer, 2002

Learning-guided Large Neighborhood Search (LNS)

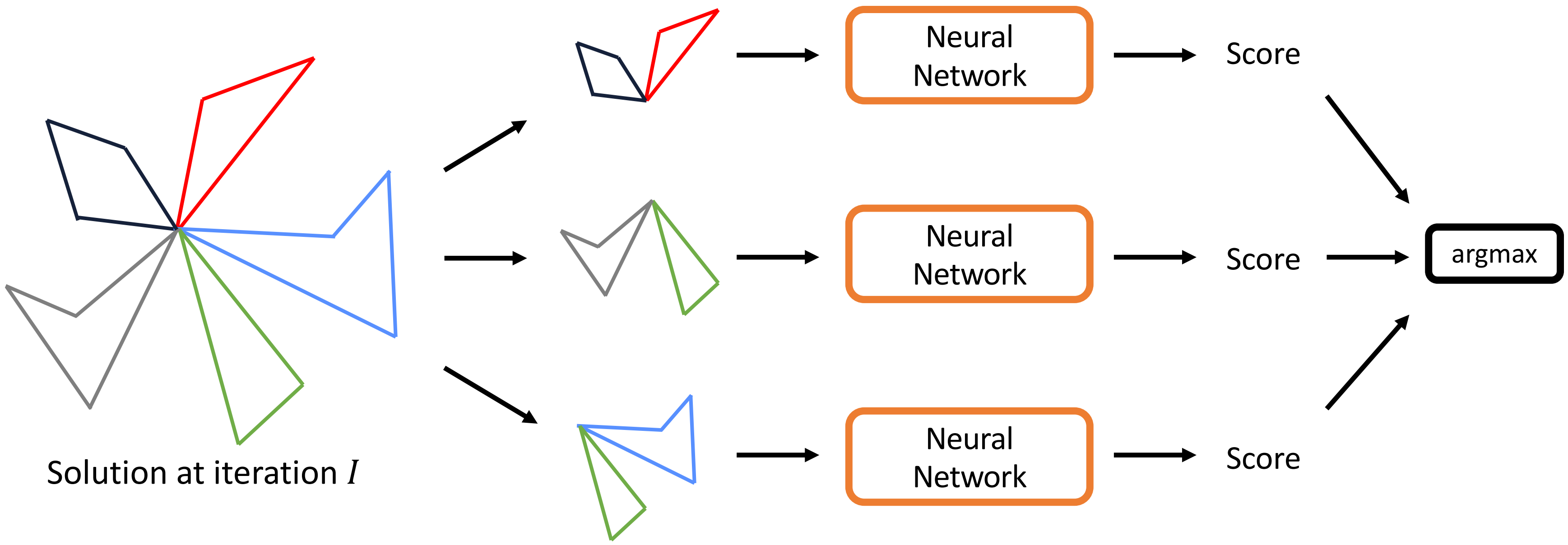
- Our work: **learn promising subproblems to select**

$$\pi(s) = \arg\max_{a \in A(s)} Q_\theta(s,a) \approx \arg\max_{a \in A(s)} Q^*(s,a),$$



Li*, Yan*, **Wu**. "Learning to Delegate for Large-scale Vehicle Routing." NeurIPS, 2021. Spotlight (<3%).

Subproblem Selection with Learned Model



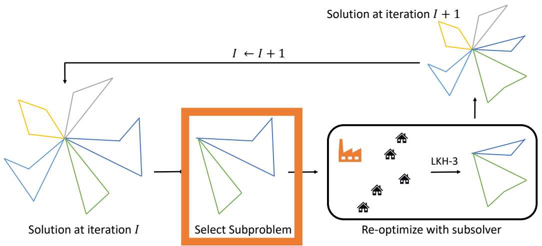
Subproblem candidates
based on **spatial locality**

$O(2^N) \rightarrow O(N)$
subproblems

Prediction
of **immediate improvement**

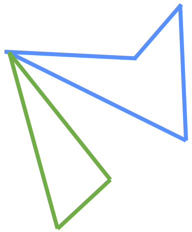
$Q_\theta(s,a) \approx \boldsymbol{Q_c^{\gamma=0}(s,a)} \approx Q_c^*(s,a)$

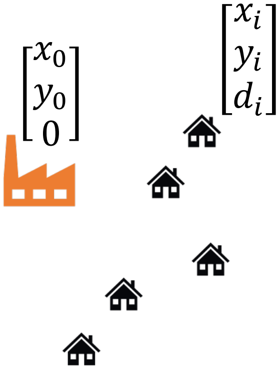
Supervision (training set) according to base solver LKH-3



Li*, Yan*, **Wu**. “Learning to Delegate for Large-scale Vehicle Routing.” NeurIPS, 2021. Spotlight (<3%).

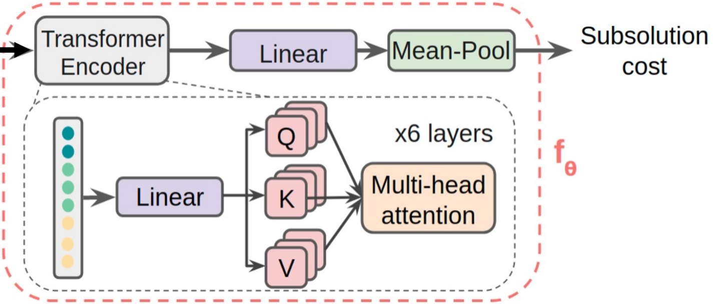
Transformer-based Neural Network





$$\begin{bmatrix} x_0 & y_0 & 0 \\ x_1 & y_1 & d_1 \\ x_2 & y_2 & d_2 \\ x_3 & y_3 & d_3 \\ x_4 & y_4 & d_4 \\ x_5 & y_5 & d_5 \end{bmatrix}$$

$$Q_\theta(s, a)$$



Subproblem Candidate

Remove Previous Routes

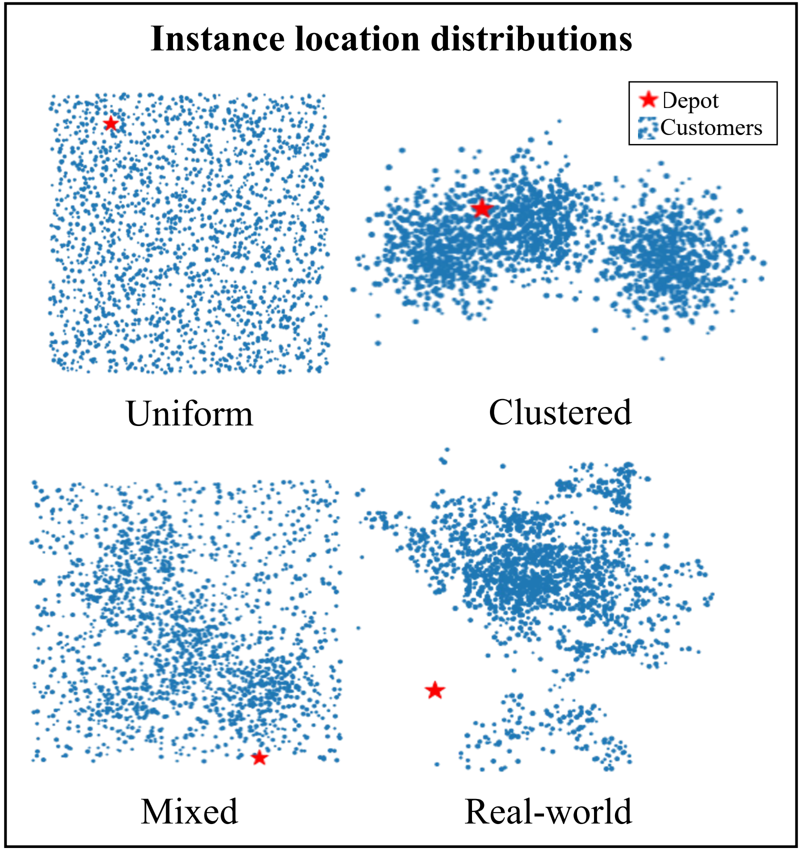
Unordered features

Neural network model (6-layer Transformer)

Score (via LKH-3 solver)

Li*, Yan*, **Wu**. “Learning to Delegate for Large-scale Vehicle Routing.” NeurIPS, 2021. Spotlight (<3%).

Results: Contexts - Location distributions



VRP with varying location distributions

Locations	Size N	Random	Ours	Ours / Random
Uniform	500	4x	7x	**1.8x**
	1000	6x	12x	**2x**
	2000	8x	15x	**1.9x**
Clustered	500	6x	40x	**7x**
	1000	8x	40x	**5x**
	2000	8x	18x	**2x**
Mixed	500	4x	11x	**3x**
	1000	6x	10x	**1.7x**
	2000	6x	14x	**2x**
Real-world	2000	8x	16x	**2x**

Table: Results. Speed-up to 95% LKH-3 30k solution quality

Finding: Up to 7x speed-up over random selection

Li*, Yan*, **Wu**. "Learning to Delegate for Large-scale Vehicle Routing." NeurIPS, 2021. Spotlight (<3%).

Results: Contexts - Classic VRP constraints

Variations

- **CVRP**: VRP with vehicle capacity
- **CVRPTW**: CVRP with time windows
- **VRPMPD**: CVRP with pickup and delivery

Finding: Effective across VRP constraints

VRP with varying constraints

Problem	Size N	Random	Ours	Ours / Random
CVRP	500	4x	7x	**1.8x**
	1000	6x	12x	**2x**
	2000	8x	15x	**1.9x**
CVRPTW	500	1.9x	2x	1x
	1000	2x	3x	**1.5x**
	2000	6x	8x	**1.3x**
VRPMPD	500	2x	4x	**2x**
	1000	6x	10x	**1.7x**
	2000	20x	30x	**1.5x**

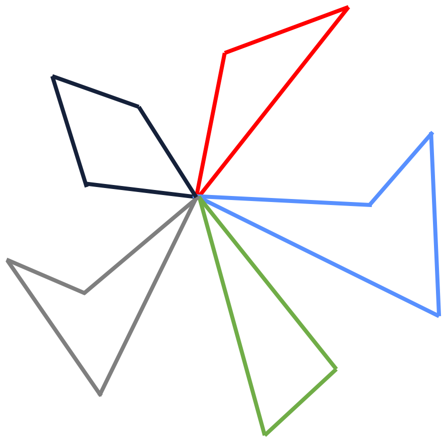
Table: Results. Speed-up to 95% LKH-3 30k solution quality

Li*, Yan*, **Wu**. "Learning to Delegate for Large-scale Vehicle Routing." NeurIPS, 2021. Spotlight (<3%).

Learning-guided LNS for large vehicle routing

- First learning-guided LNS method for VRPs
- **Fast inference** using $\boldsymbol{Q_\theta}(s,a) \approx Q_c(s,a)$
- Up to **7x faster** than random $a \in A_c$
- Takeaway: When inference is faster than solving, leverage learning to accelerate solving

Vehicle routing problem (VRP)



Notation:
- Policy π
- State s, Action a
- Action space A
- Value function Q

Li*, Yan*, Wu. "Learning to Delegate for Large-scale Vehicle Routing." NeurIPS, 2021. Spotlight (<3%).

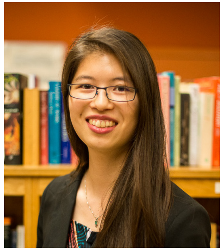
This talk

- Vision: Cope with growing system complexity with meta-methods
- **Learning-guided search as meta-methods for transportation?**
  - Vehicle routing problems
  - **Multi-robot warehousing**
  - Integer linear programming
- What about autonomous vehicles?

Neural Neighborhood Search for Multi-agent Path Finding

ICLR 2024

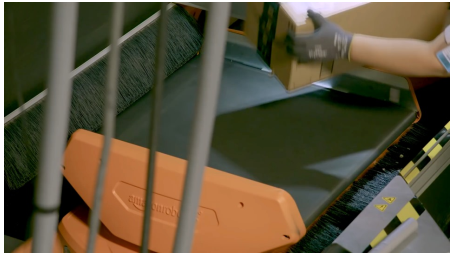
Zhongxia Yan, **Cathy Wu**



Motivation: large **real-time** applications

Robotic Warehouses

E.g. multi-robot transportation of packages in Amazon sortation centers



Induct: load robot with package

↑

Incoming truckloads of packages

Transport: robots move between induct stations and eject chutes

Eject: deposit package into chute

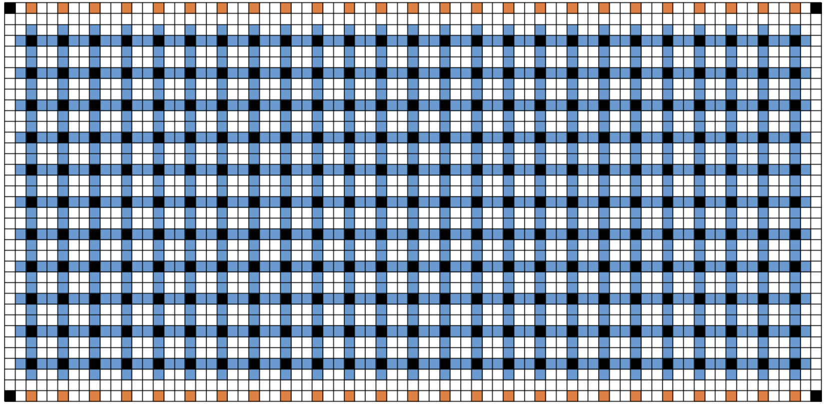
↓

Outgoing truckloads of packages

Background: Amazon Sortation

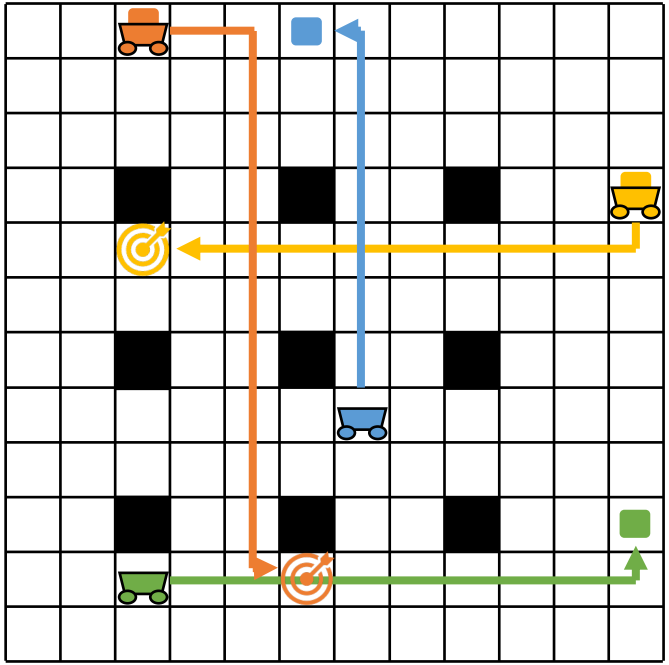
Multi-robot transportation of packages in Amazon sortation centers

- Fleet of ~800 robots ("drives")
- Stream of incoming packages into the warehouse
- Two types of robot tasks:
  - Loaded: transport package to eject
  - Unloaded: move to an induct
- Need to optimize **throughput** (#tasks completed / duration)!



Map source: Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph Durham, T. K. Satish Kumar, Sven Koenig. *Lifelong MAPF in Large-Scale Warehouses*. AAAI 2021

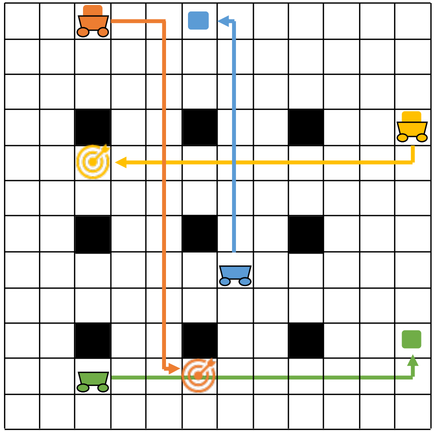
Multi-agent Path Finding (MAPF)



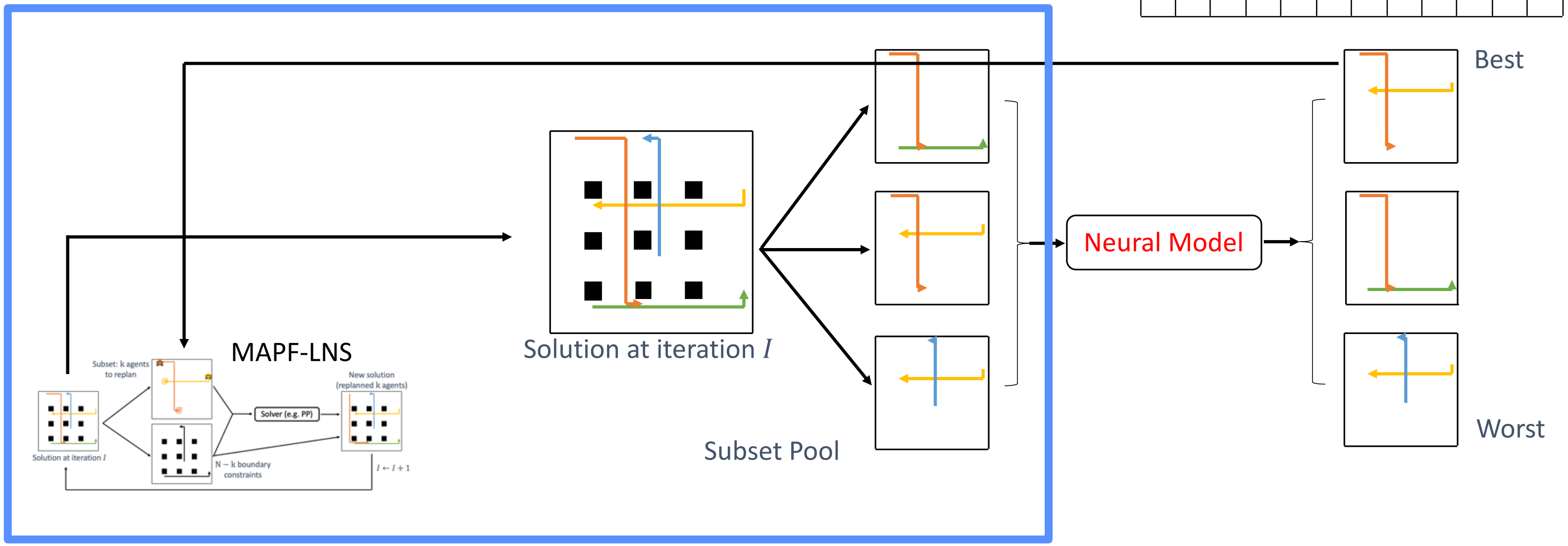
- A slice of the overall sortation problem (discretized)
- Agents $a \in \{1, \dots, N\}$
  - Start location s_a
  - Goal location g_a
- Solution: non-colliding ***space-time*** trajectories for all agents
- Cost: sum of trajectory lengths

Learning-guided LNS for MAPF

Problem illustration



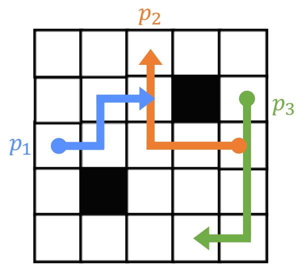
Solver (Non-neural) Runtime: 5-100ms



Yan, **Wu**. “Neural Neighborhood Search for Multi-agent Path Finding.” ICLR, 2024.

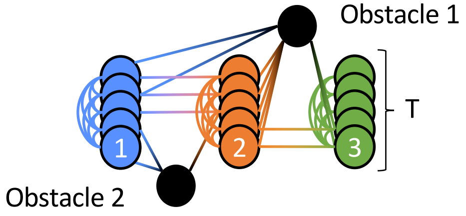
Challenges of Neural LNS for MAPF

- **Why did [Huang 2022] stop at linear features?**
- **Motion**: MAPF has agent-agent, agent-obstacle, and intra-agent interactions
  - Agent-to-agent and agent-obstacle: 3D convolution is suitable
  - Intra-agent interactions: how to represent interaction *along* an agent's path?
  - Encoding is much heavier than for vehicle routing problems (VRP)
- **Need to encode J (10-100) subsets each LNS step**
  - *Can we share computation?*
- LNS for MAPF requires total neural network inference time to be roughly $\leq$ 25ms
  - Difficult for 3D CNN



$A = \{1,2,3\}$

Desired agent interaction graph

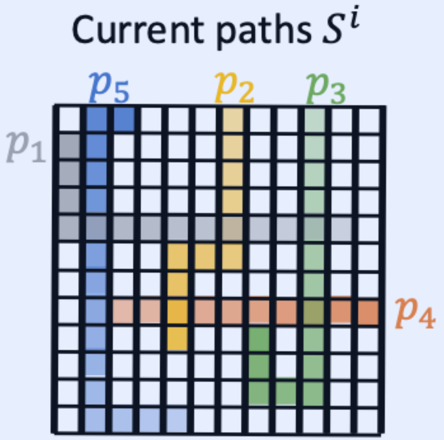


Linear NN: Taoan Huang, Jiaoyang Li, Sven Koenig, Bistra Dilkina. *Anytime MAPF via ML-guided LNS*. AAAI 2022
Deep NN (Ours): Zhongxia Yan, Cathy Wu. *Neural Neighborhood Search for MAPF*. ICLR 2024

Wu

Contribution: Architecture

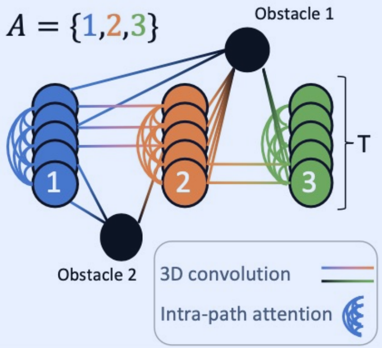
- New neural architecture (Multi-Subset) that efficiently encodes agent-agent relationships in dense constrained environments



Encode **local** spatiotemporal information using **3D convolutions**

Encode **non-local** trajectory-level information using **self-attention**

Desired agent interaction graph

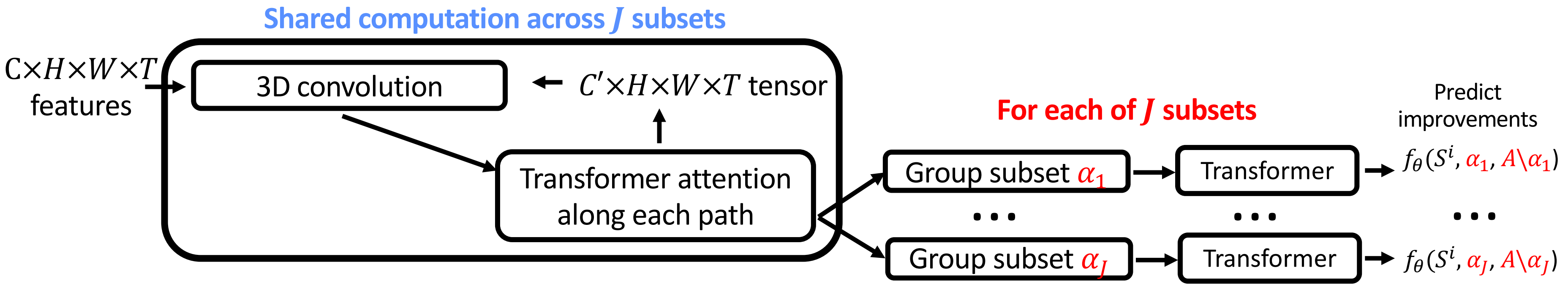


→ **agent-agent relationships**

Yan, **Wu**. "Neural Neighborhood Search for Multi-agent Path Finding." ICLR, 2024.

Our Proposal: Multi-Subset Architecture

- Summary



- Global representation of all agent paths shares heavy computation across all subsets
- Intra-agent transformer helps preserve agent entity
  - Subsets of agents can be grouped directly from global representation

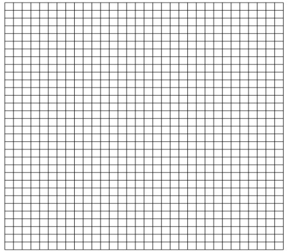
Large state $s \in \mathcal{S}_c$ ⇨ **Amortized inference across J subsets** using state encoding $E_\nu(s)$, s.t. $Q_\theta(\boldsymbol{E_\nu(s)}, a) \approx Q_c(s, a)$

Wu

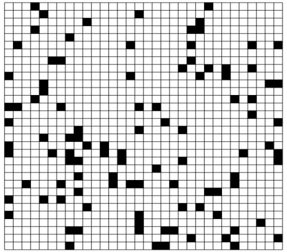
Zhongxia Yan, Cathy Wu. *Neural Neighborhood Search for Multi-agent Path Finding*. ICLR

Experimental Setup

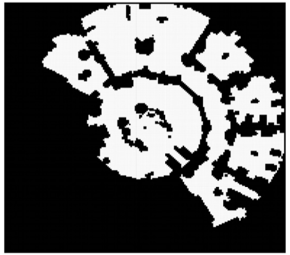
- MAPF Benchmark Suite
  [Stern 2019]
- Pre-apply spatial pooling for large floor maps
- PBS as subset solver
- LNS methods
  - Unguided (Baseline)
  - Linear (Baseline)
    - Hand-designed features
  - Per-Subset (Baseline)
  - **Multi-Subset**



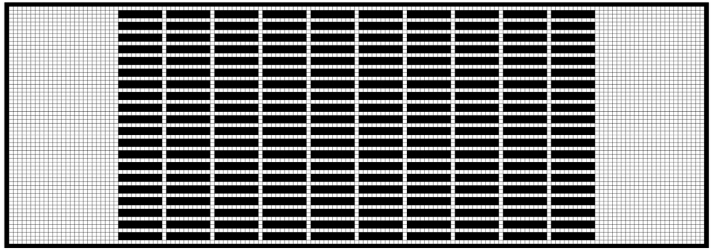
empty (32x32)
350 agents



random (32x32)
250 agents



ost003d (194x194)
400 agents



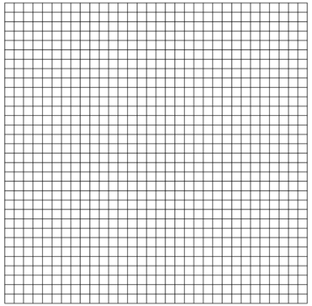
warehouse (161x63)
300 agents

den520d (256x257)
800 agents

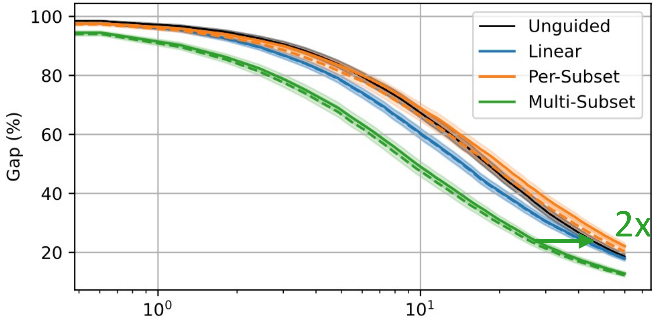
Zhongxia Yan, Cathy Wu. *Neural Neighborhood Search for Multi-agent Path Finding*. ICLR

Results: Cost (Gap) vs Time

Solid: with neural overhead Dashed: without neural overhead

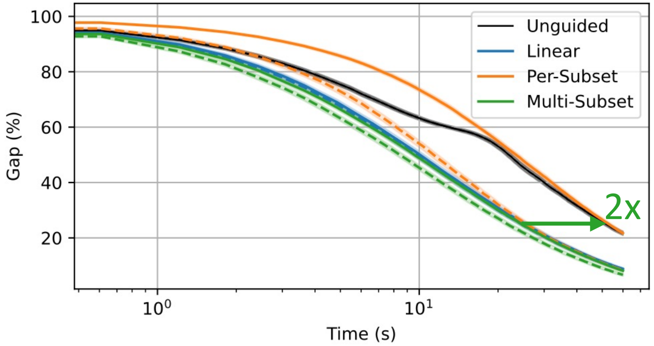


empty (32x32)
350 agents



Linear is similar to Unguided

den512d (256x257)
800 agents



Linear is similar to Multi-Subset

Zhongxia Yan, Cathy Wu. *Neural Neighborhood Search for Multi-agent Path Finding*. ICLR

Results

Multi-Subset outperforms baselines

This is despite a 4-10x added overhead vs Linear

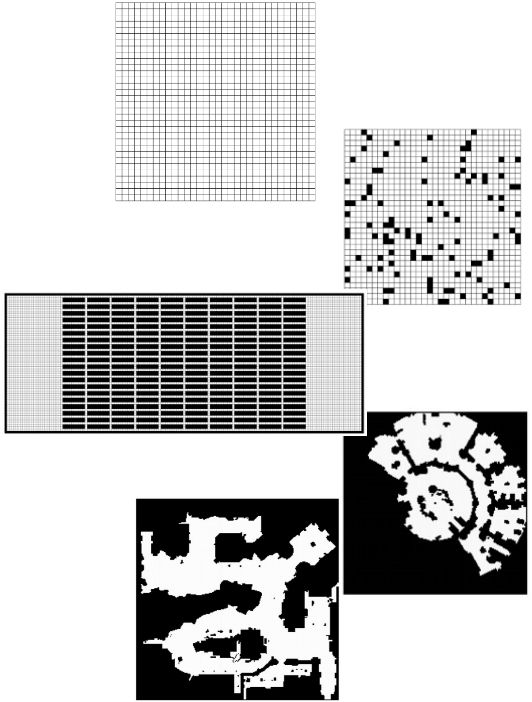


Table 1: **Performance and overhead of all methods.**

Setting	Metric	Unguided	Linear	Per-Subset	Multi-Subset
empty (32x32)	Average Gap (%)	42 ± 1	39 ± 1	46 ± 1	**31 ± 1**
$\|A\| = 350$	Win / Loss	0 / 0	136 / 50	57 / 129	**178 / 8**
Uniform $k = 50$	Final Gap (%)	19 ± 1	18 ± 0.8	22 ± 1	**13 ± 0.7**
	Overhead (s)	0.1	+0.002	+0.03	+0.013
random (32x32)	Average Gap (%)	5.3 ± 0.2	5.4 ± 0.3	6.2 ± 0.2	**5 ± 0.2**
$\|A\| = 250$	Win / Loss	0 / 0	98 / 100	36 / 162	**116 / 82**
Agent-local $k = 25$	Final Gap (%)	**0.51 ± 0.03**	0.7 ± 0.05	0.72 ± 0.05	0.69 ± 0.04
	Overhead (s)	0.079	+0.0018	+0.03	+0.012
warehouse (161x63)	Average Gap (%)	14 ± 0.6	15 ± 0.7	20 ± 0.8	**12 ± 0.6**
$\|A\| = 300$	Win / Loss	0 / 0	82 / 103	6 / 179	**155 / 30**
Agent-local $k = 25$	Final Gap (%)	1.7 ± 0.2	2.5 ± 0.3	2.5 ± 0.3	**1.6 ± 0.2**
	Overhead (s)	0.21	+0.0025	+0.05	+0.025
ost003d (194x194)	Average Gap (%)	43 ± 1	36 ± 1	35 ± 0.9	**22 ± 1**
$\|A\| = 400$	Win / Loss	0 / 0	180 / 20	188 / 12	**200 / 0**
Agent-local $k = 10$	Final Gap (%)	24 ± 1	15 ± 1	16 ± 0.8	**6.5 ± 0.7**
	Overhead (s)	0.063	+0.002	+0.09	+0.013
den520d (256x257)	Average Gap (%)	45 ± 0.7	30 ± 0.7	48 ± 0.7	**29 ± 0.7**
$\|A\| = 800$	Win / Loss	0 / 0	**200 / 0**	47 / 153	**200 / 0**
Agent-local $k = 25$	Final Gap (%)	22 ± 0.7	8.8 ± 0.4	22 ± 0.6	**8.2 ± 0.5**
	Overhead (s)	0.15	+0.006	+0.19	+0.025

Yan, **Wu**. “Neural Neighborhood Search for Multi-agent Path Finding.” ICLR, 2024.

Results

Multi-Subset reduces added overhead by 2-8x vs naïve architecture (Per-Subset)

Per-Subset generally underperforms baselines

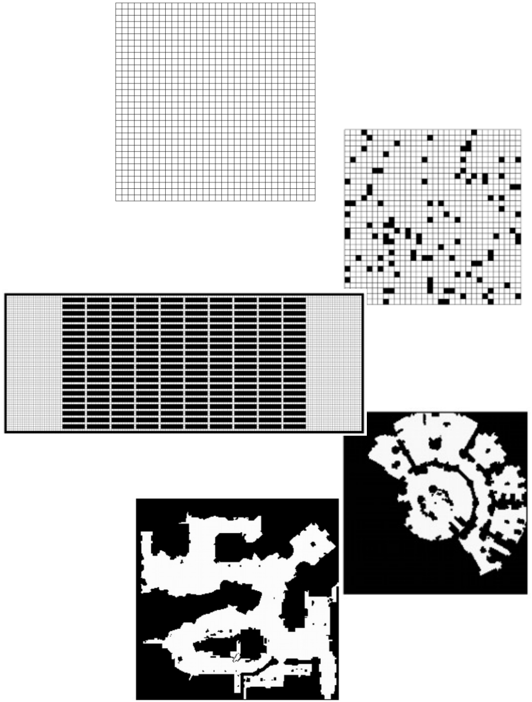


Table 1: **Performance and overhead of all methods.**

Setting	Metric	Unguided	Linear	Per-Subset	Multi-Subset
empty (32x32) $\|A\| = 350$ Uniform $k = 50$	Average Gap (%)	42 ± 1	39 ± 1	46 ± 1	**31 ± 1**
	Win / Loss	0 / 0	136 / 50	57 / 129	**178 / 8**
	Final Gap (%)	19 ± 1	18 ± 0.8	22 ± 1	**13 ± 0.7**
	Overhead (s)	0.1	+0.002	+0.03	+0.013
random (32x32) $\|A\| = 250$ Agent-local $k = 25$	Average Gap (%)	5.3 ± 0.2	5.4 ± 0.3	6.2 ± 0.2	**5 ± 0.2**
	Win / Loss	0 / 0	98 / 100	36 / 162	**116 / 82**
	Final Gap (%)	**0.51 ± 0.03**	0.7 ± 0.05	0.72 ± 0.05	0.69 ± 0.04
	Overhead (s)	0.079	+0.0018	+0.03	+0.012
warehouse (161x63) $\|A\| = 300$ Agent-local $k = 25$	Average Gap (%)	14 ± 0.6	15 ± 0.7	20 ± 0.8	**12 ± 0.6**
	Win / Loss	0 / 0	82 / 103	6 / 179	**155 / 30**
	Final Gap (%)	1.7 ± 0.2	2.5 ± 0.3	2.5 ± 0.3	**1.6 ± 0.2**
	Overhead (s)	0.21	+0.0025	+0.05	+0.025
ost003d (194x194) $\|A\| = 400$ Agent-local $k = 10$	Average Gap (%)	43 ± 1	36 ± 1	35 ± 0.9	**22 ± 1**
	Win / Loss	0 / 0	180 / 20	188 / 12	**200 / 0**
	Final Gap (%)	24 ± 1	15 ± 1	16 ± 0.8	**6.5 ± 0.7**
	Overhead (s)	0.063	+0.002	+0.09	+0.013
den520d (256x257) $\|A\| = 800$ Agent-local $k = 25$	Average Gap (%)	45 ± 0.7	30 ± 0.7	48 ± 0.7	**29 ± 0.7**
	Win / Loss	0 / 0	**200 / 0**	47 / 153	**200 / 0**
	Final Gap (%)	22 ± 0.7	8.8 ± 0.4	22 ± 0.6	**8.2 ± 0.5**
	Overhead (s)	0.15	+0.006	+0.19	+0.025

Yan, **Wu**. "Neural Neighborhood Search for Multi-agent Path Finding." ICLR, 2024.

Conclusion

- We propose an architecture for efficiently representing three-dimensional trajectories of hundreds of agents
  - Has applications beyond LNS for MAPF
- We show that this architecture can practically accelerate the state-of-the-art LNS-based multi-path planning solvers by 1.5-4x
  - A milestone for learning-guided LNS for these spatiotemporal problems
  - *Though we still require a GPU, while Unguided does not*

This talk

- Vision: Cope with growing system complexity with meta-methods

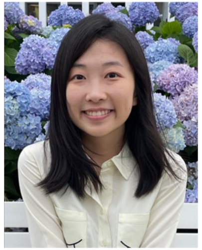
- **Learning-guided search as meta-methods for transportation?**
  - Vehicle routing problems
  - Multi-robot warehousing
  - **Integer linear programming**

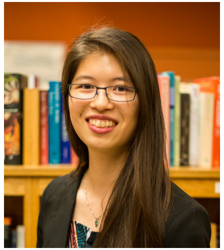
- What about autonomous vehicles?

Learning to Configure Separators in Branch-and-Cut

NeurIPS 2023

Sirui Li*, Wenbin Ouyang*, Max B. Paulus, **Cathy Wu**



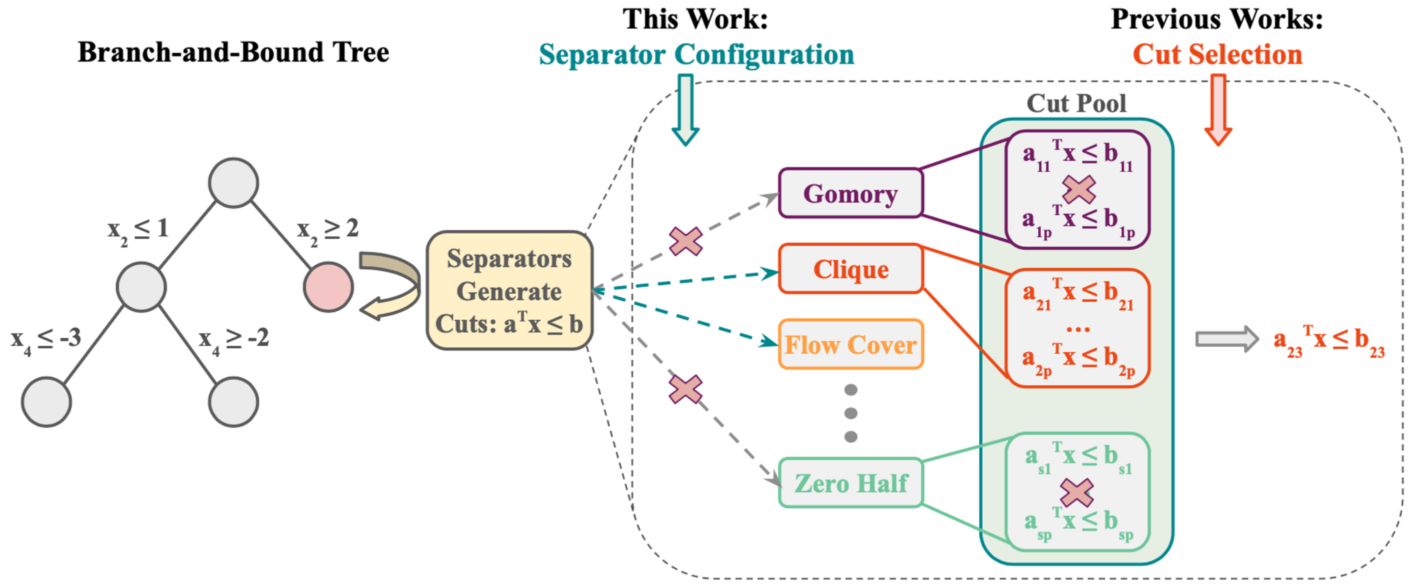



The Separator Configuration Task in Branch-and-Cut

Mixed integer linear programs (MILPs)

$\min_x$	$c^T x$	objective
s.t.	$Ax \leq b$	constraints
	$x_j \in \mathbb{Z}, \forall j \in \mathcal{J}$	integrality

This work introduces a new machine learning task to accelerate solving MILPs.



Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

Learning-to-Separate: Configuring cutting planes for MILPs

Main challenge: Configuration space $|A| = 2^M \approx 130K$ is huge! ($M \approx 17$ is # of separators)

Unlike in VRP, no "spatial locality" for configurations

Theorem: Optimal predictor performance $f_{\bar{A}}$ is **submodular** in $\bar{A} \subseteq A$.

Implication: Can greedily construct $\bar{A} \ll A$; in practice $|\bar{A}| \approx 20$.

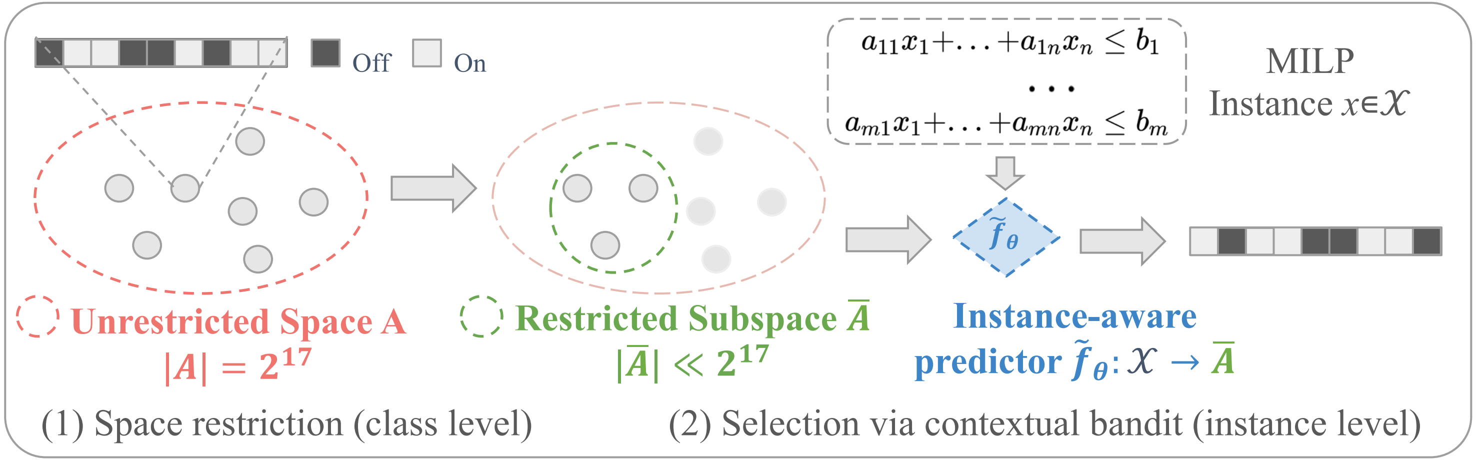


Figure: Learning-to-separate method

Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

Results: Standard MILP benchmarks

Performance:
median
(std)

Table 1: **Tang et al. and Ecole.** Absolute solve time of SCIP default and relative time improvement (median and standard deviation) of different methods (higher the better, best are bold-faced).

		Tang			Ecole		
	Method	Bin. Pack.	Max. Cut	Pack.	Comb. Auc.	Indep. Set	Fac. Loc.
	Default Time (s)	0.076s (0.131s)	1.77s (0.56s)	8.82s (25.46s)	2.73s (4.43s)	8.21s (114.15s)	61.1s (55.37s)
Heuristic Baselines	Default	0%	0%	0%	0%	0%	0%
	Random	-23.4% (153.8%)	-108.4% (168.2%)	-91% (127.8%)	-48.6% (159.0%)	-5% (161.4%)	-33.3% (157.2%)
	Prune	13.9% (27.0%)	2.7% (26.2%)	6.6% (45.0%)	12.3% (24.2%)	18.0% (24.2%)	24.7% (47.9%)
Ours Heuristic Variants	Inst. Agnostic Configuration	33.7% (36.6%)	69.8% (10.5%)	20.1% (38.0%)	60.1% (27.6%)	57.8% (29.5%)	11.5% (21.8%)
	Random within Restr. Subspace	26.9% (33.6%)	68.0% (11.0%)	18.8% (38.7%)	58.1% (28.7%)	57.4% (75.8%)	17.7% (33.0%)
Ours Learned	L2Sep	**42.3% (34.2%)**	**71.9% (11.3%)**	**28.5% (39.3%)**	**66.2% (26.2%)**	**72.4% (27.8%)**	**29.4% (39.6%)**

Similar gains over base Gurobi MILP solver of **12-56%**

Subspace restriction **without** learning

with learning

Relative time improvements of **29-72%**

Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

Results: Real-World MILP benchmarks

Performance: median (std)

Table 3: **Real-world MILPs.** Absolute solve time of SCIP default and relative time improvement (median and standard deviation) of different methods (higher the better, best are bold-faced).

Methods	Default Times (s)	Heuristic Baselinses			Ours Heuristic Variants		Ours Learned
		Default	Random	Prune	Inst. Agnostic Configuration	Random within Restr. Subspace	L2Sep
MIPLIB	25.08s (57.05s)	0%	-149.1% (149.7%)	4.8% (107.6%)	5.5% (71.5%)	1.9% (74.9%)	**12.9% (73.1%)**
NN Verification	31.42s (22.44s)	0%	-300.0% (152.3%)	31.5% (36.3%)	31.4% (38.3%)	30.7% (34.1%)	**37.5% (33.9%)**
Load Balancing	31.86s (7.07s)	0%	-300.1% (129.5%)	21.1% (150.8%)	10.4% (8.5%)	10.0% (31.5%)	**21.2% (20.3%)**

Challenging heterogeneous benchmark (mixed MILP classes)

Subspace restriction **without** learning

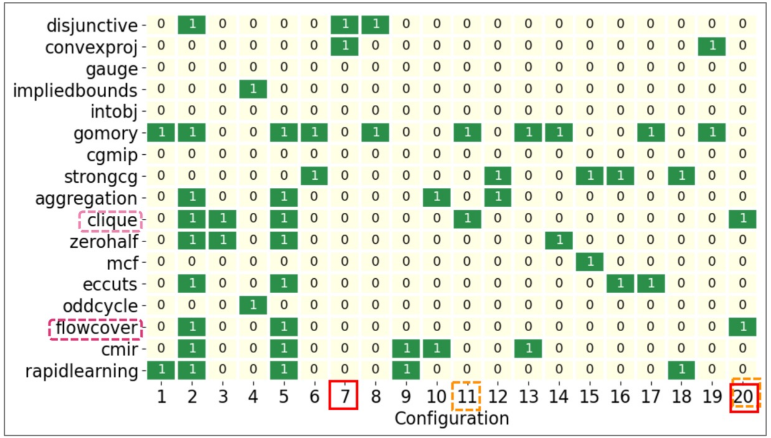
with learning

Relative time improvements of **13-37%**

Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

Learning-to-Separate: Interpretation analysis

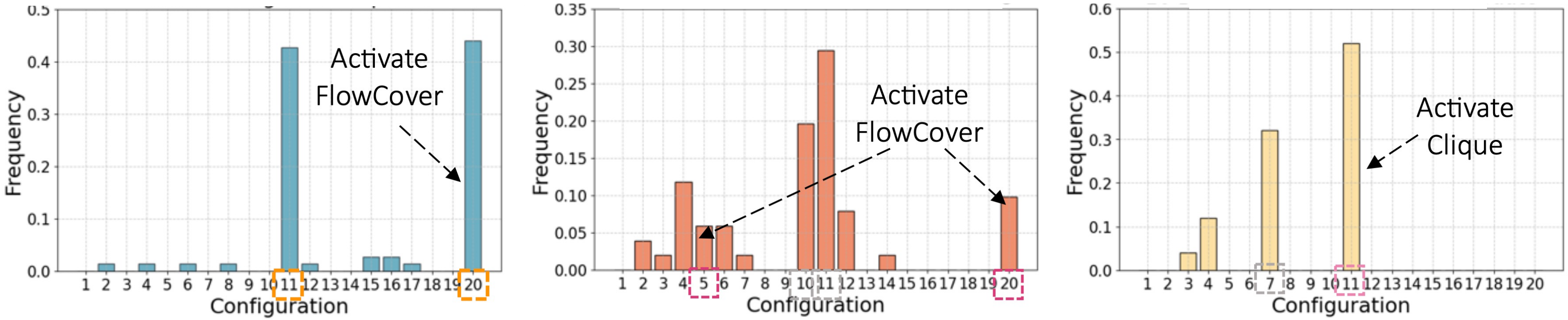
- The learned model recovers known facts from operations research about separators.
- **Example: Bin Packing**
  - Many bins ≈ Bipartite Matching problem
    → Flowcover cuts effective
  - Few bins ≈ Knapsack problem
    → Clique cuts effective



Many bins (66 bins)

Moderate bins (33 bins)

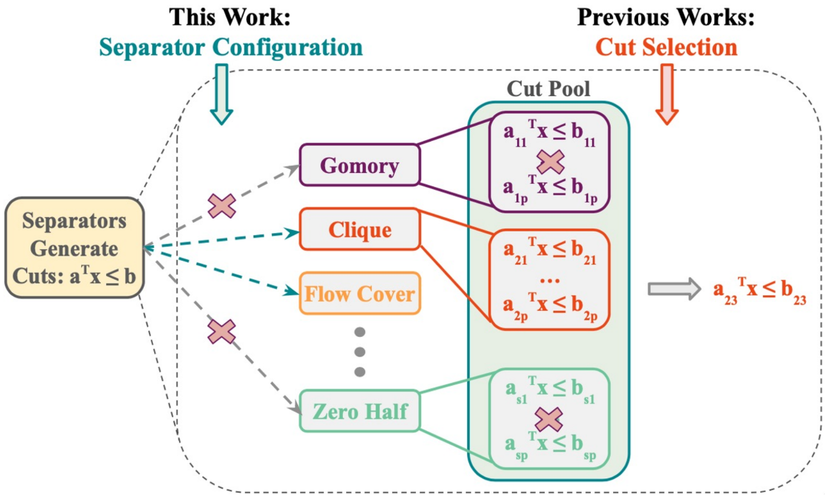
Few bins (16 bins)



Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

Conclusions: Learning-to-separate

- First method to effectively configure BnC separators for wide range of MILPs
- Up to 3.5x faster than SCIP & 2x faster than Gurobi
- Takeaway: When there are too many actions to learn effectively, leverage submodularity to restrict action space



Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.

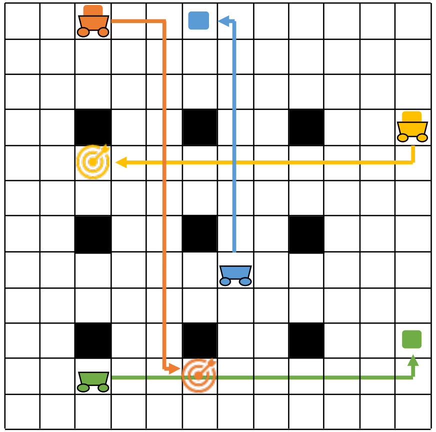
This talk

- Vision: Cope with growing system complexity with meta-methods

- Learning-guided search as meta-methods for transportation?
  - Vehicle routing problems
  - Multi-robot warehousing
  - Integer linear programming

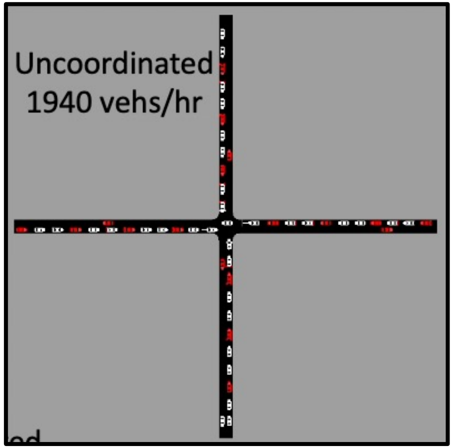
- **What about autonomous vehicles?**

Better Search at Autonomous Intersections?



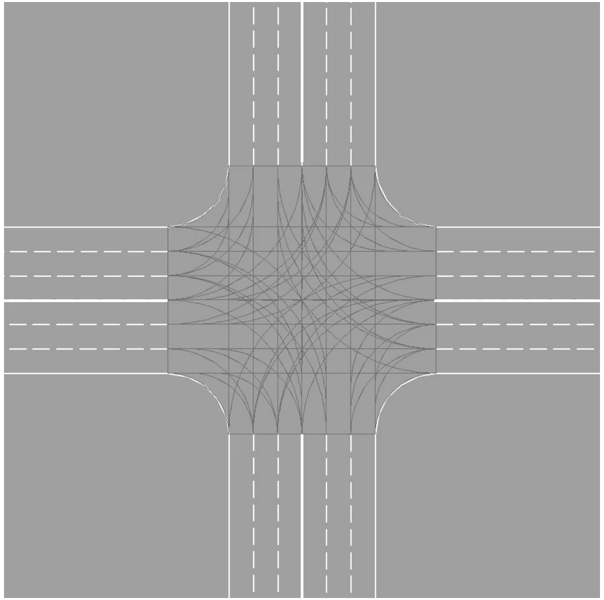
Multi-agent path finding

+



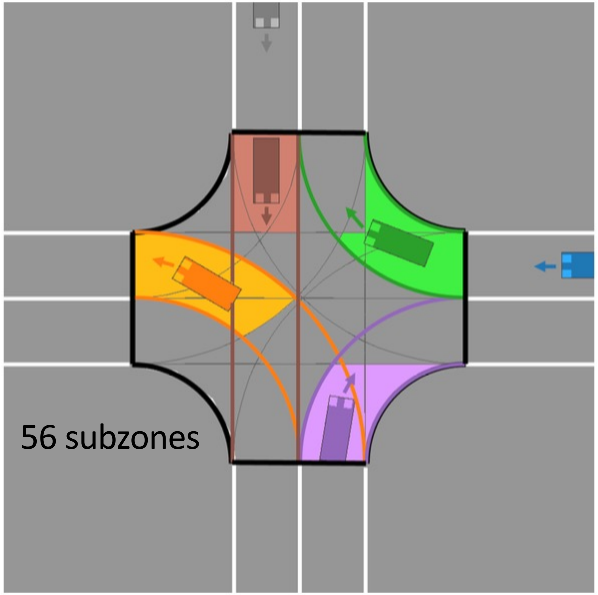
Vehicle kinematics

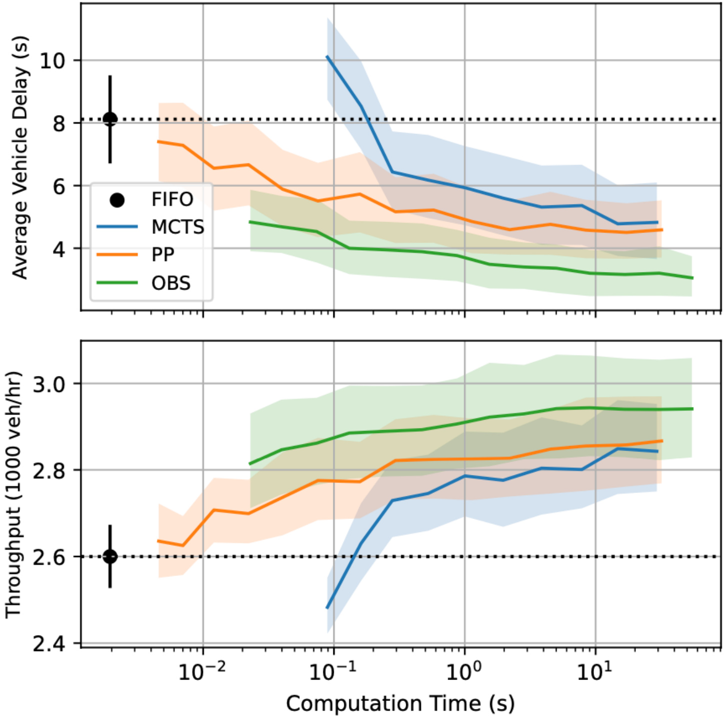
=



Zhongxia Yan, Han Zheng, Cathy Wu. *Multi-agent Scheduling of Intersection Crossings for Cooperative Autonomous Driving*. ICRA 2024. Under review at IEEE T-RO.

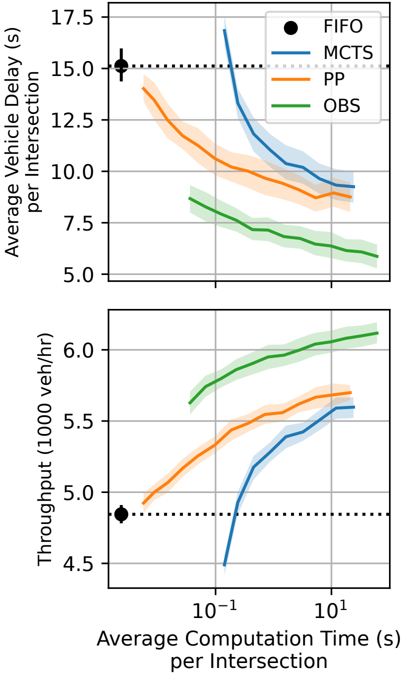
Results: Single Intersection



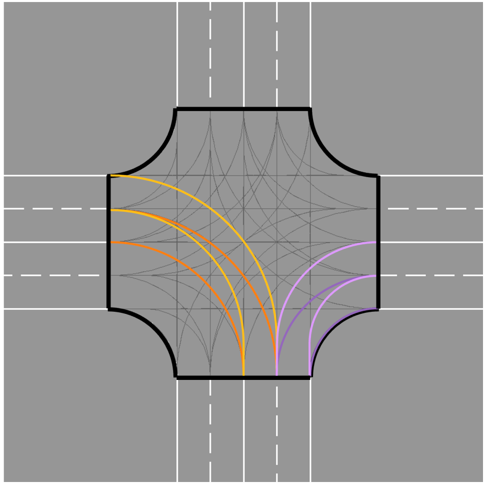


Zhongxia Yan, Han Zheng, Cathy Wu. *Multi-agent Scheduling of Intersection Crossings for Cooperative Autonomous Driving*. ICRA 2024. Under review at IEEE T-RO.

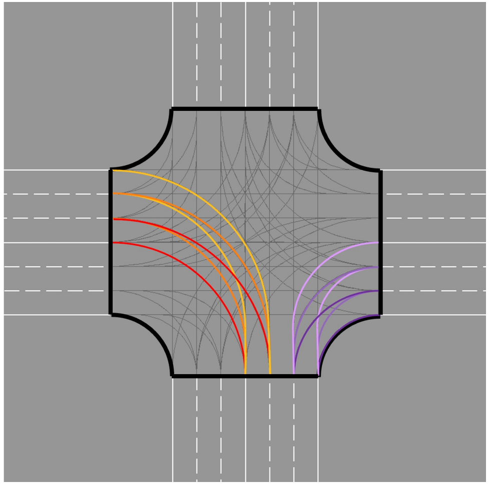
Results: Multi-lane Intersections

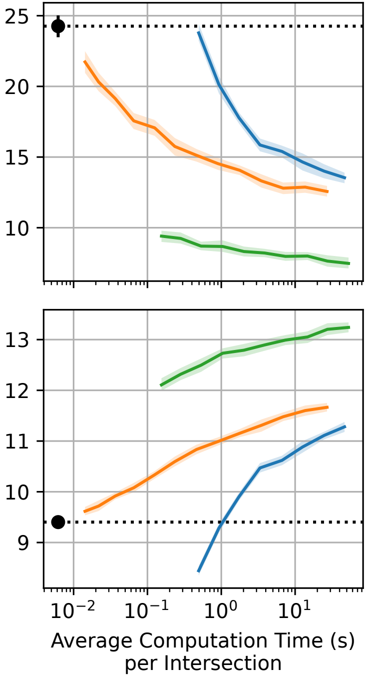


Two-lanes: 230 subzones

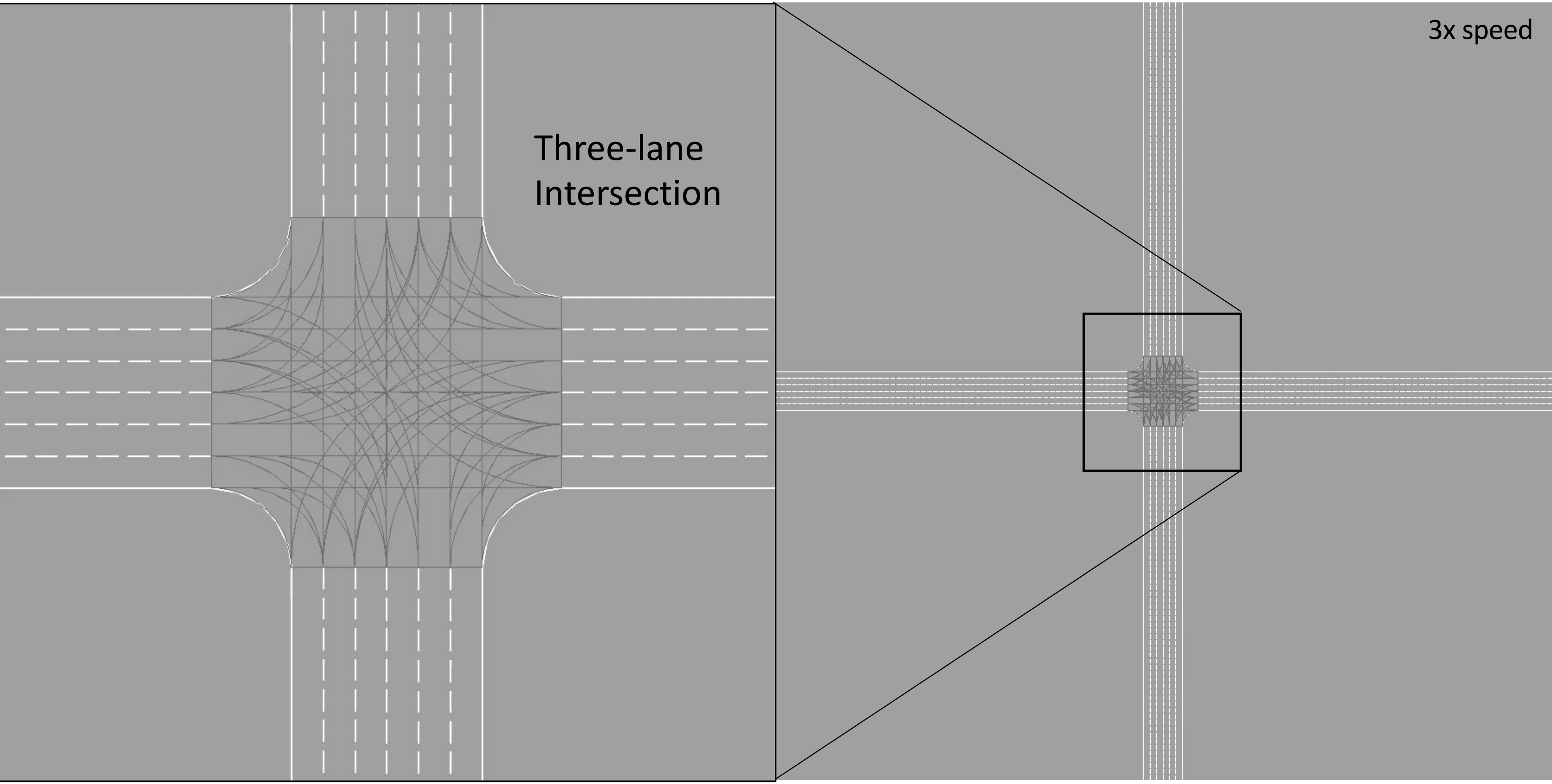


Three-lanes: 524 subzones



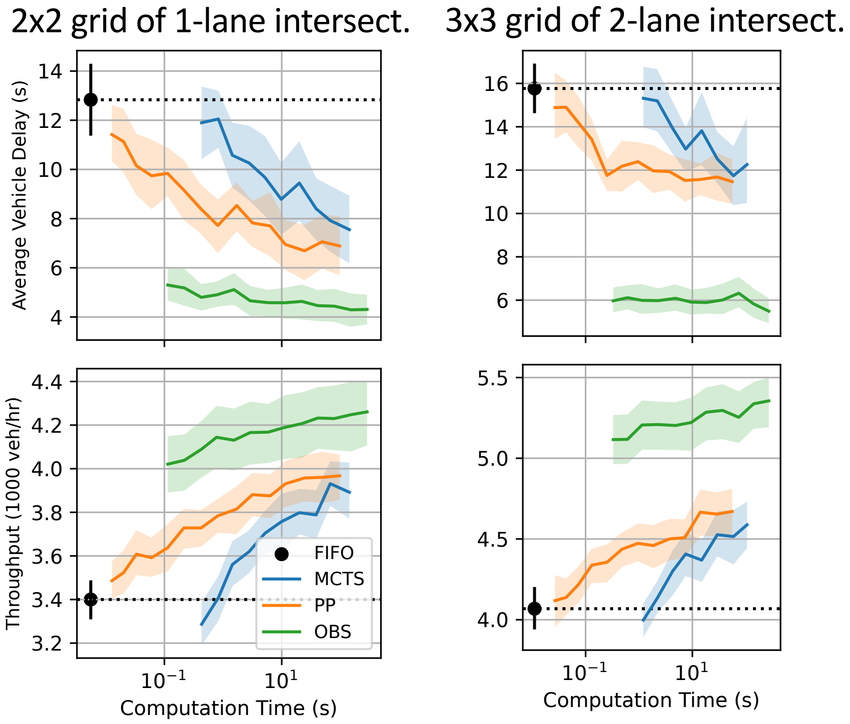


Zhongxia Yan, Han Zheng, Cathy Wu. *Multi-agent Scheduling of Intersection Crossings for Cooperative Autonomous Driving*. ICRA 2024. Under review at IEEE T-RO.

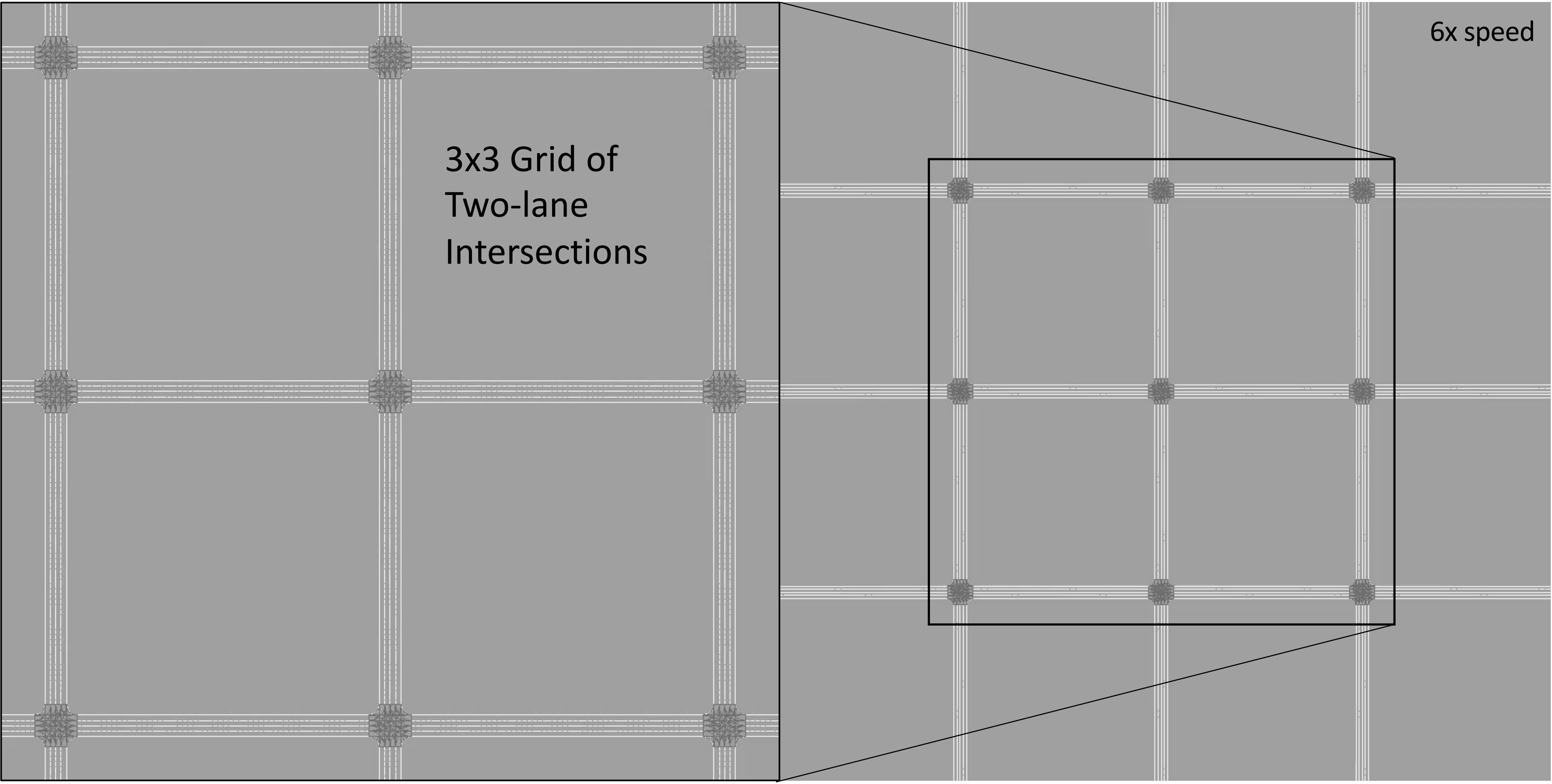

Three-lane
Intersection
3x speed

Results: Multiple Intersections

- Crossing order does not make sense across multiple intersections!
- Thus, we optimize each intersection's crossing orders independently of its neighboring intersection
- OBS is just as effective!



Zhongxia Yan, Han Zheng, Cathy Wu. *Multi-agent Scheduling of Intersection Crossings for Cooperative Autonomous Driving*. ICRA 2024. Under review at IEEE T-RO.


3x3 Grid of
Two-lane
Intersections
6x speed

This talk: Towards Scalable Learning-based Mobile Coordination

- Vision: Cope with growing system complexity with meta-methods
- Learning-guided search as meta-methods for transportation?
  - Vehicle routing problems [1]
  - Multi-robot warehousing [2]
  - Integer linear programming [3]
- What about autonomous vehicles? [4]

[1] Li*, Yan*, **Wu**. "Learning to Delegate for Large-scale Vehicle Routing." NeurIPS, 2021. Spotlight (<3%).
[2] Yan, **Wu**. "Neural Neighborhood Search for Multi-agent Path Finding." ICLR, 2024.
[3] Li*, Ouyang*, Paulus, **Wu**. "Learning to Configure Separators in Branch-and-Cut." NeurIPS, 2023.
[4] Z. Yan, H. Zheng, and **C. Wu**, "Multi-agent Path Finding for Cooperative Autonomous Driving," in *ICRA*, 2024.