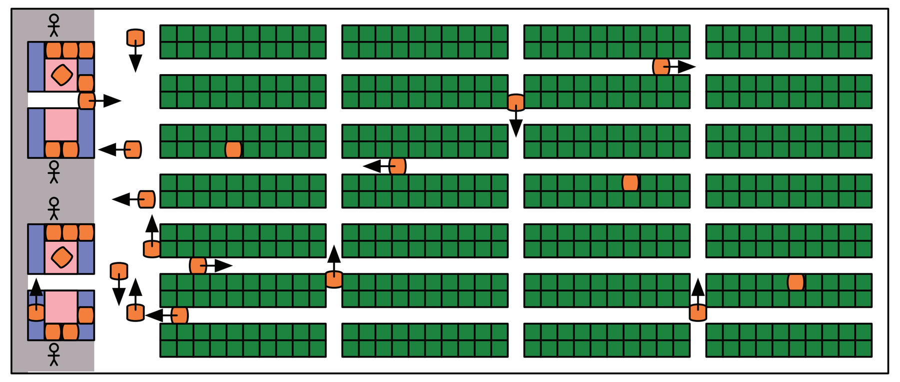
Machine Learning-Guided Search Algorithms for Multi-Agent Path Finding by Leveraging Domain Heuristics

Taoan Huang, University of Southern California (Joint work with Jiaoyang Li, Bistra Dilkina and Sven Koenig)

USC University of Southern California

SCAN ME

1 Multi-Agent Path Finding (MAPF) and Our Contribution



Input:
- An unweighted undirected graph.
- A set of agents, each with a start location and a goal location.

Output:
- A set of collision-free paths, one for each agent, that minimizes the sum of travel time.

We leverage domain heuristics and machine learning to improve different search algorithms for MAPF:

Optimal algorithms
- Conflict-based search: A complete tree search
- Learn to branch on conflicts by imitating an agent dependency heuristics

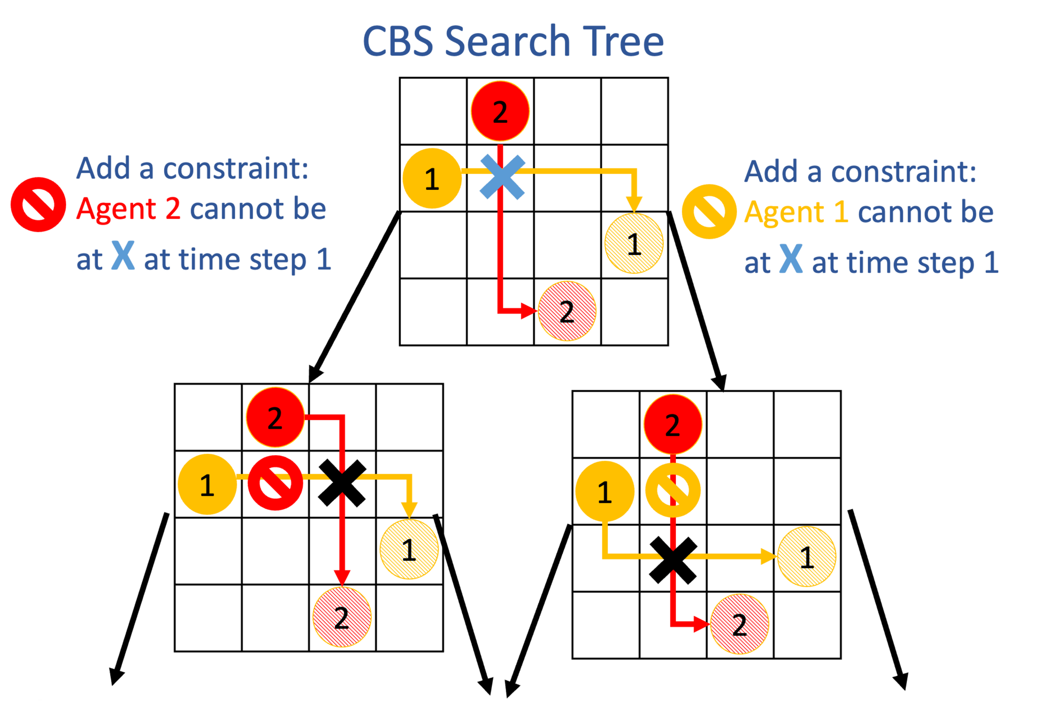
Bounded-suboptimal algorithms
- ECBS: A bounded suboptimal version of CBS
- Learn to improve node selection with imitation and curriculum learning based on a distance-to-goal heuristic

Unbounded-suboptimal algorithms
- Large Neighborhood Search for MAPF
- Learn a destroy heuristic by leveraging spatial-temporal information provided by hand-crafted heuristics

2 ML-Guided Conflict Based Search (CBS)

CBS

An optimal bi-level tree search:
- A tree node has a set of constraints, each prohibits an agent to travel along an edge or be at a vertex at time t
- High-level Search:
  - Pick a node with the minimum cost
  - Branch on a conflict between two agents in its current solution
  - Expand the tree by adding two children, each imposing a constraint for an agent resolving that conflict
- Low-level Search: Replan the optimal agents' path w.r.t. the imposed constraints



Weighted Dependency Graph Heuristic

- Effective: State-of-the-art heuristic to choose conflicts to branch on
- Not efficient: Solve a weighted vertex cover problem for each conflict

ML Framework

Learn to rank the conflicts as similarly as possible to the WDG heuristic, without actually computing it

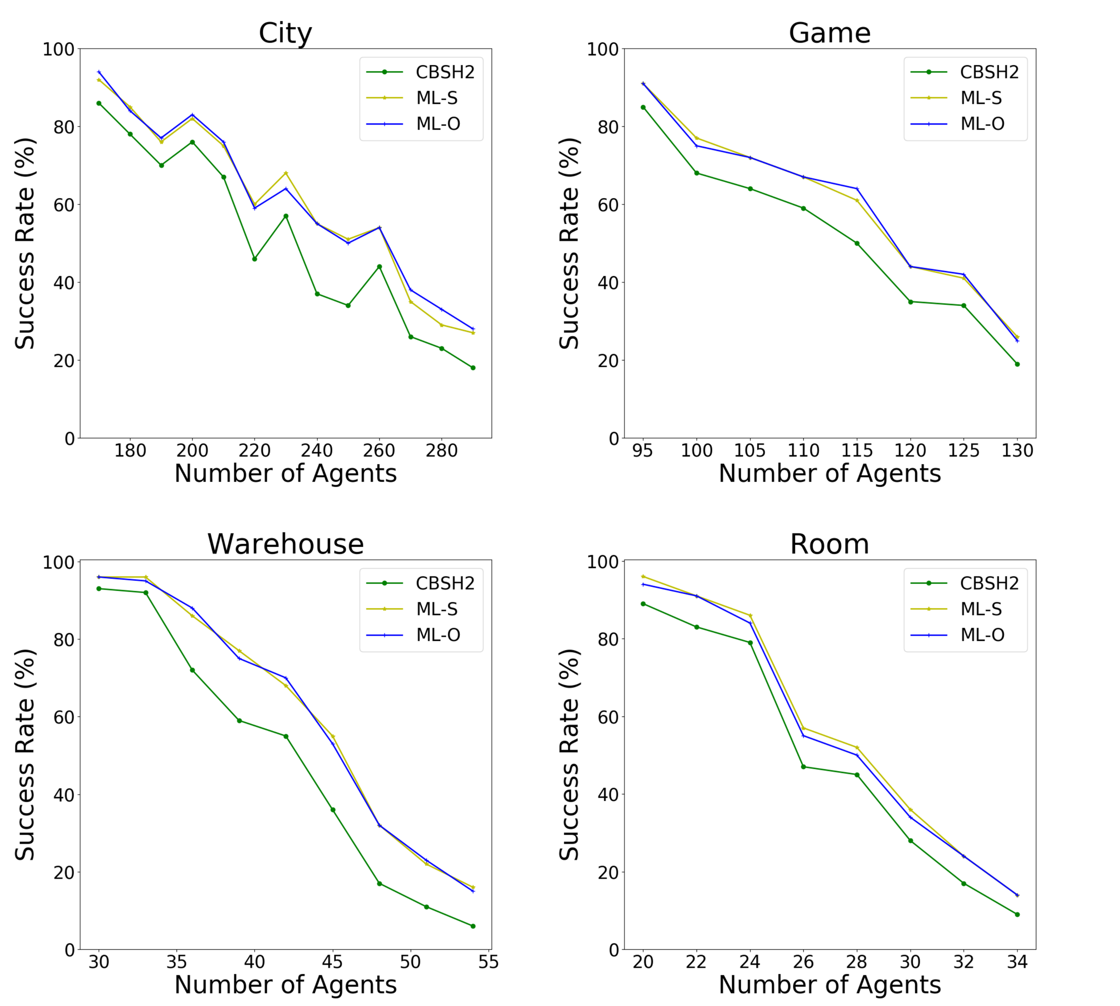
1. Data collection: Run CBS exhaustively with the WDG heuristic
2. Model learning: Imitate the heuristic's decision via learning to rank conflicts

Experimental Results

Train two models for each map
- ML-S: Train on the data collected on the same map
- ML-O: Train on the data collected on the other maps

Improvement over CBSH2 (state-of-the-art CBS):
- Runtime: **10.3%-64.4%** faster
- Tree sizes: **13.0% to 68.2%** fewer nodes

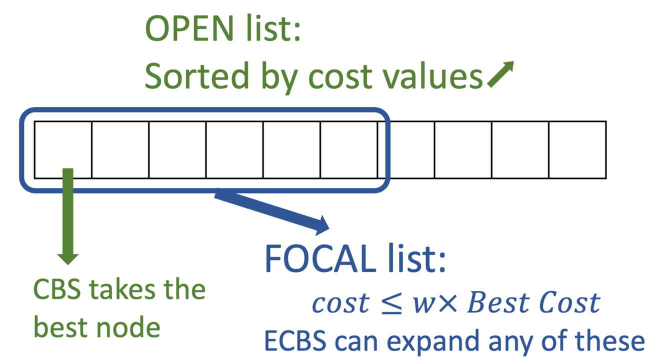
Success rates: the fractions of solved instances within time limit



3 ML-Guided Enhanced CBS (ECBS)

ECBS

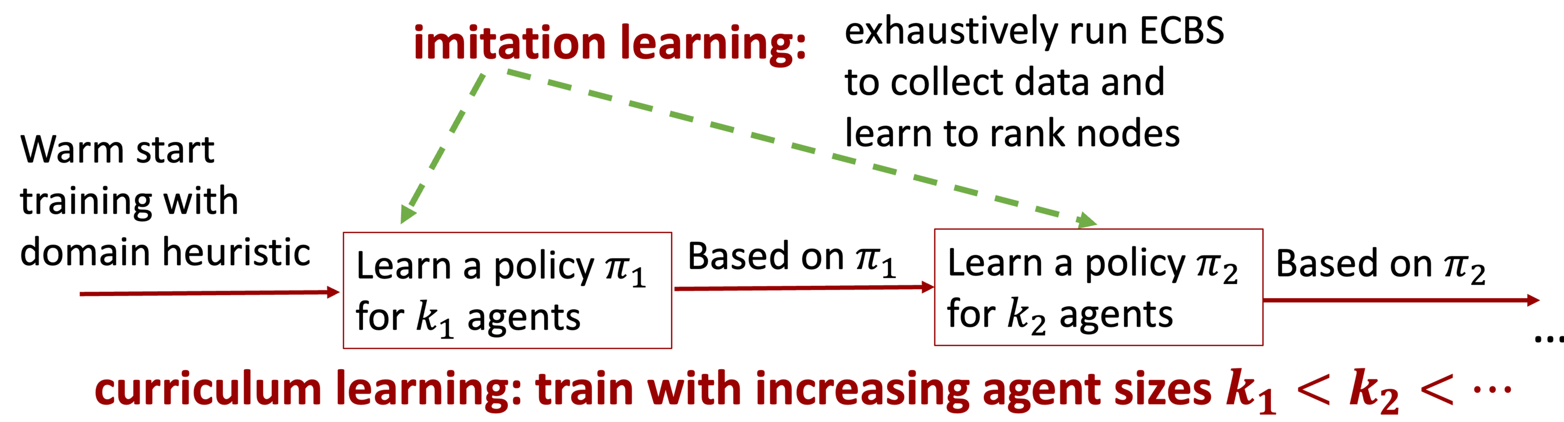
Bounded-suboptimal version of CBS:
- Find a solution with cost $\leq w\times$ optimal ($w \geq 1$)
- Node selection in ECBS vs CBS



Node Selection Heuristics

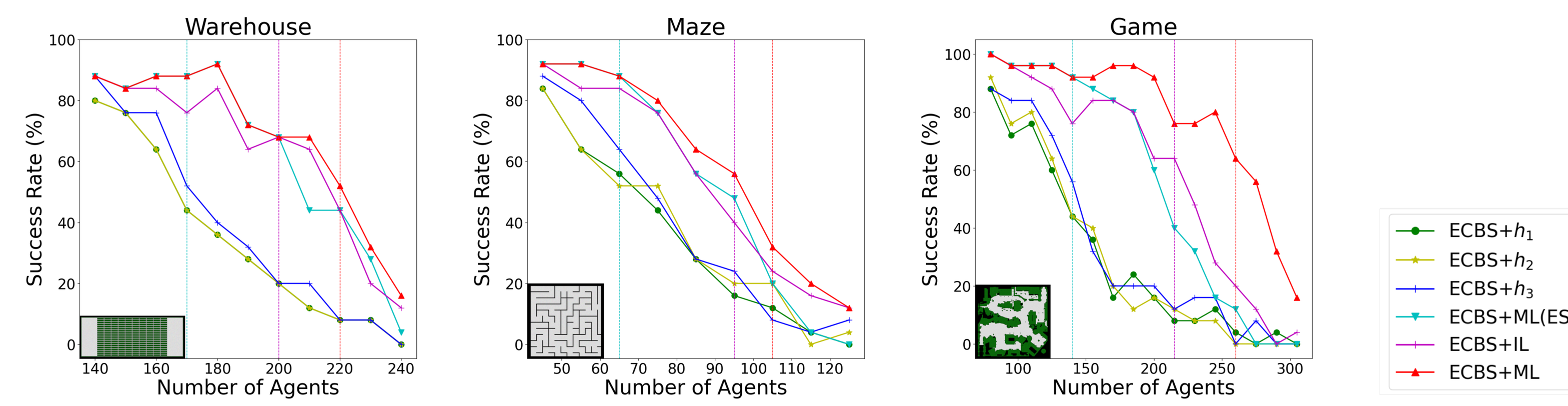
- Existing heuristics: choose the node with the fewest conflicts or conflicting (pairs of) agents
- Our method: retrospectively learn a better heuristic using the existing one as a starting point

ML Framework



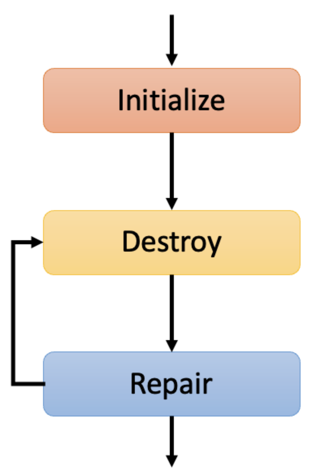
Experimental Results

ECBS+h_i: ECBS with domain heuristic h_i
ECBS+ML: Our method
ECBS+ML(ES): Our method with early stopping in curriculum learning
ECBS+IL: Our method with imitation learning but no curriculum learning



4 ML-Guided Large Neighborhood Search (LNS)

MAPF-LNS

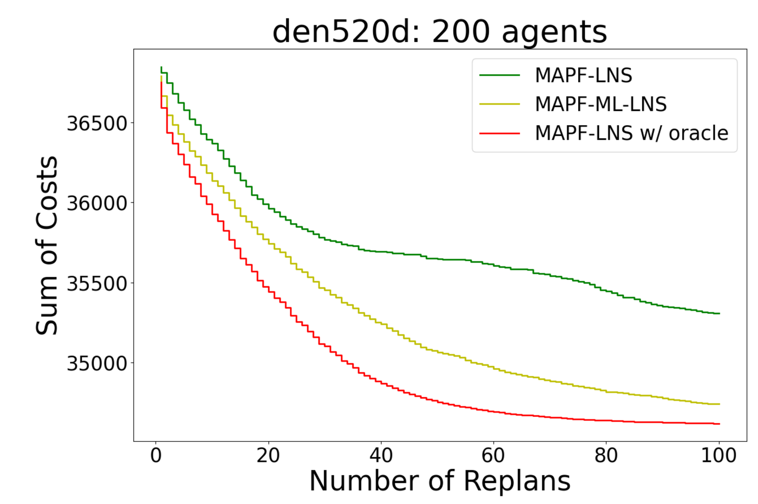


1. Find an initial solution via a non-optimal MAPF solver.
2. Select a subset of agents.
3. Replan their paths so that they are collision-free with each other and with the paths of the other agents.

Destroy Heuristics

MAPF-LNS uses randomized domain heuristics.
MAPF-ML-LNS (our method) imitates an oracle:
- The oracle samples 20 candidate agent subsets using domain heuristics, replans them exhaustively and chooses the best one
- During testing, we replace the oracle with the learned model which is a lot faster

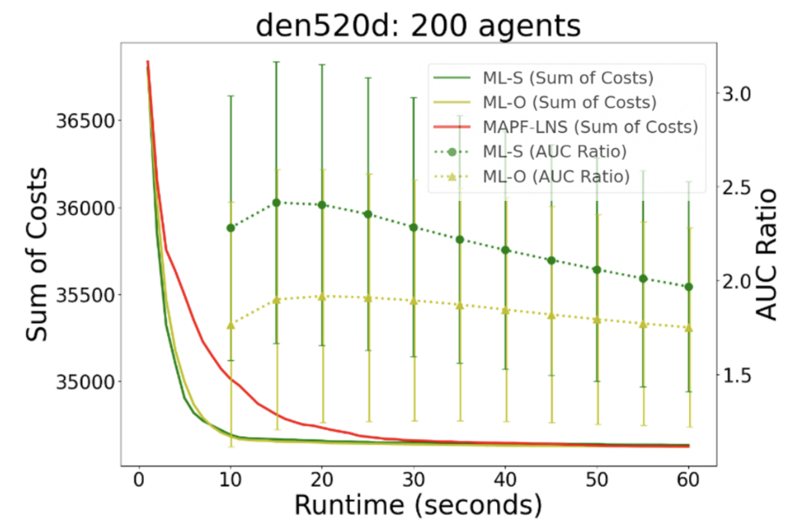
What is the best we could possibly achieve?



Experimental Results

Speed of improving solutions: We measure the area under the curve (AUC) and compare against MAPF-LNS by taking the ratios (**>1 means we are better**).
- ML-S and ML-O use models trained on the same map and unseen maps, respectively.
- k is the number of agents.

	k	AUC Ratio ML-S	AUC Ratio ML-O
den520d	200	**1.97±0.56**	**1.75±0.53**
	300	**1.62±0.55**	**1.45±0.43**
	400	**1.65±0.54**	**1.31±0.30**
	500	**1.25±0.35**	**1.13±0.22**
	600	**1.10±0.15**	**1.10±0.08**
	700	**1.07±0.06**	**1.05±0.06**



	k	Sum of Agents' Delay (Suboptimality) MAPF-LNS	ML-S	ML-O
den520d	200	64 (1.00)	65 (1.00)	66 (1.00)
	300	400 (1.01)	**298 (1.00)**	328 (1.01)
	400	1,327 (1.02)	**778 (1.01)**	**1,121 (1.01)**
	500	3,616 (1.04)	**2,676 (1.03)**	**3,281 (1.03)**
	600	8,134 (1.08)	**6,654 (1.06)**	**6,967 (1.07)**
	700	12,558 (1.10)	**11,785 (1.10)**	**11,535 (1.09)**