Truncated Counterfactual Learning for Anytime Multi-Agent Path Finding

Thomy Phan[1], **Shao-Hung Chan**[2], **Sven Koenig**[3,4]
[1]University of Bayreuth, Germany
[2]University of Southern California, USA
[3]University of California, Irvine, USA
[4]Örebro University, Sweden
thomy.phan@uni-bayreuth.de, shaohung@usc.edu, sven.koenig@uci.edu

Abstract

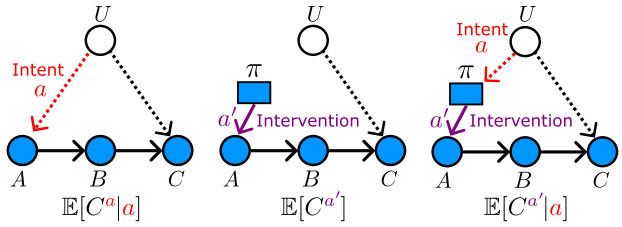
Anytime *multi-agent path finding (MAPF)* is a promising approach to scalable and collision-free path optimization in multi-agent systems. MAPF-LNS, based on *Large Neighborhood Search (LNS)*, is the current state-of-the-art approach where a fast initial solution is iteratively optimized by destroying and repairing selected paths, i.e., a *neighborhood*, of the solution. *Delay-based MAPF-LNS* has demonstrated particular effectiveness in generating promising neighborhoods via *seed agents*, according to their delays. Seed agents are selected using handcrafted strategies or online learning, where the former relies on human intuition about underlying structures, while the latter conducts black-box optimization, ignoring any structure. In this paper, we propose *Truncated Adaptive Counterfactual K-ranked LEarning (TACKLE)* to select seed agents via informed online learning by leveraging handcrafted strategies as human intuition. We show theoretically that TACKLE dominates its handcrafted and black-box learning counterparts in the limit. Our experiments demonstrate cost improvements of at least 60% in instances with one thousand agents, compared with state-of-the-art anytime solvers.

Code — github.com/thomyphan/counterfactual-mapf-lns

1 Introduction

A wide range of real-world applications, like goods transportation in warehouses, search and rescue missions, and traffic management, can be formulated as a *Multi-Agent Path Finding (MAPF)* problem, where the goal is to find collision-free paths for multiple agents with assigned start and goal locations. Finding optimal solutions, w.r.t. minimal flowtime or makespan, is NP-hard, which limits the scalability of most state-of-the-art MAPF solvers (Ratner and Warmuth 1986; Sharon et al. 2012; Yu and LaValle 2013).

Anytime MAPF based on *Large Neighborhood Search (LNS)* is a promising approach to finding fast and high-quality solutions to the MAPF problem within a fixed time budget (Li et al. 2021). Given an initial feasible solution and a set of destroy heuristics, LNS iteratively destroys and replans a fixed number of paths, i.e., a *neighborhood*, until the time budget runs out. MAPF-LNS represents the current state-of-the-art in anytime MAPF and has been experimentally shown to scale up to scenarios with hundreds of agents (Li et al. 2021). Due to its increasing popularity, several extensions have been proposed, such as fast local repair, integration of primal heuristics, learning mechanisms, and parallelism (Chan et al. 2024; Huang et al. 2022; Jiang et al. 2024; Lam et al. 2023; Li et al. 2022; Phan et al. 2024b).



(a) Observational (b) Interventional (c) Counterfactual

Figure 1: Causal graphs for delay-based MAPF-LNS with an unobserved confounder U, seed agent A, neighborhood paths B, and payoff C based on (Pearl 2010). In each model, blue nodes denote observable variables. Dashed lines illustrate the influence of the unobserved variable U, and solid lines illustrate the influence of observable variables. **(a)** Observational/intent selection via U. **(b)** Interventional seed agent selection via policy π. **(c)** Counterfactual seed agent selection via π conditioned on the pre-selection of (a).

Delay-based MAPF-LNS has demonstrated particular effectiveness in large-scale scenarios with many agents (Phan et al. 2025; Tan et al. 2025), where LNS neighborhoods are generated via *seed agents* according to their delays. The seed agent selection is important for the resulting solution quality and is commonly based on handcrafted strategies or online learning (Li et al. 2021; Phan et al. 2025; Tan et al. 2025). While the former relies on human intuition about underlying structures (Fig. 1a), the latter optimizes the selection in a black-box fashion, ignoring any structure (Fig. 1b). However, despite impressive results reported by the latter, no strategy dominates the other in general (Nozick 1969).

In this paper, we propose *Truncated Adaptive Counterfactual K-ranked LEarning (TACKLE)* using *causal multi-armed bandits with unobserved confounders (MABUC)* for delay-based MAPF-LNS. Our contributions are as follows:

- We formulate a causal model for seed agent selection in

delay-based MAPF-LNS and use counterfactual learning (Fig. 1c) with a truncated and non-stationary MABUC.

- We show theoretically that TACKLE dominates the effectiveness of its handcrafted and black-box learning counterparts in the limit.
- We empirically evaluate TACKLE in well-known benchmark maps (Stern et al. 2019) and demonstrate cost improvements of at least 60% in instances with one thousand agents, compared with state-of-the-art solvers.

While our paper focuses on MAPF, TACKLE can also be adjusted for other problem classes, where variables can be sorted by their contribution to the total cost in order to generate LNS neighborhoods (Pisinger and Ropke 2019).

2 Background

2.1 Multi-Agent Path Finding (MAPF)

We focus on *maps* as undirected, unweighted *graphs* $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where vertex set $\mathcal{V}$ contains all possible locations and edge set $\mathcal{E}$ contains all possible transitions or movements between adjacent locations. An *instance* I consists of a map G and a set of *agents* $\mathbb{A} = \{a_1, ..., a_m\}$ with each agent a_i having a *start location* $s_i \in \mathcal{V}$ and a *goal location* $g_i \in \mathcal{V}$. At every time step t, all agents can move along the edges in $\mathcal{E}$ or wait at their current location (Stern et al. 2019).

MAPF aims to find a collision-free plan for all agents. A *plan* $P = \{p_1, ..., p_m\}$ consists of individual paths $p_i = \langle p_{i,0}, ..., p_{i,l(p_i)} \rangle$ per agent a_i, where $\langle p_{i,t}, p_{i,t+1} \rangle = \langle p_{i,t+1}, p_{i,t} \rangle \in \mathcal{E}$, $p_{i,0} = s_i$, $p_{i,l(p_i)} = g_i$, and $l(p_i)$ is the *length* or *travel distance* of path p_i. The *delay* $del(p_i)$ of path p_i is defined by the difference of path length $l(p_i)$ and the length of the shortest path from s_i to g_i w.r.t. map G.

In this paper, we consider *vertex conflicts* $\langle a_i, a_j, v, t \rangle$ that occur when two agents a_i and a_j occupy the same location $v \in \mathcal{V}$ at time step t and *edge conflicts* $\langle a_i, a_j, u, v, t \rangle$ that occur when two agents a_i and a_j traverse the same edge $\langle u, v \rangle \in \mathcal{E}$ in opposite directions at time step t (Stern et al. 2019). A plan P is a *solution*, i.e., *feasible* when it has no vertex or edge conflicts. Our goal is to find a feasible solution by minimizing the *flowtime* $\sum_{p \in P} l(p)$ equivalent to minimizing the *sum of delays* or *(total) cost* $SoC(P) = \sum_{p \in P} del(p)$. In the context of anytime MAPF, we also consider the *Area Under the Curve (AUC)* as a measure of how quickly we approach the quality of our final solution.

2.2 Anytime MAPF with LNS

Anytime MAPF searches for solutions within a given *time budget*. The solution quality improves monotonically with increasing time budget (Cohen et al. 2018; Li et al. 2021).

MAPF-LNS, based on *Large Neighborhood Search (LNS)*, is the current state-of-the-art approach to anytime MAPF and has been shown to scale up to large-scale scenarios with hundreds of agents (Huang et al. 2022). Starting with an initial feasible plan P, e.g., found via *prioritized planning (PP)* from (Silver 2005), MAPF-LNS iteratively modifies P by destroying up to $n < m$ paths, i.e., a *neighborhood* $B \subset P$. The destroyed paths B are repaired or replanned using PP to quickly generate new paths B'. If the new cost $SoC(B')$ is lower than the previous cost $SoC(B)$, then P is replaced by $(P \backslash B) \cup B'$, and the search continues until the time budget runs out. The result of MAPF-LNS is the last accepted solution P, with the lowest cost so far.

Since the number of possible neighborhoods scales exponentially w.r.t. m for $n > 1$ (Huang et al. 2022), MAPF-LNS employs multiple *destroy heuristics* to select promising neighborhoods B whose modifications have a high potential C of reducing the cost of the solution P (Li et al. 2021).

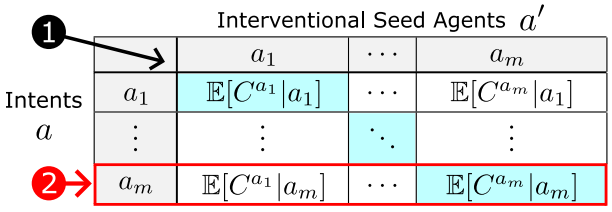


Figure 2: Counterfactual table as a cross of intent and interventional seed agents a and a'. Each cell stores the estimated counterfactual payoff $\mathbb{E}[C^{a'}|a]$. The MABUC **(1)** queries the intent agent a by selecting a row before **(2)** making a decision on the row entry a' as the actual seed agent (red).

2.3 Delay-Based Anytime MAPF-LNS

We focus on delay-based destroy heuristics, which are empirically the most effective in common benchmark maps (Li et al. 2021; Phan et al. 2025; Tan et al. 2025). The idea is to select a *seed agent* $A = a_j \in \mathbb{A}$, whose path $p_j \in P$ has a high potential to be shortened, indicated by its delay $del(p_j)$. A random walk is performed from a random position in p_j to collect up to $n - 1$ other agents a_i whose paths p_i are crossed by the random walk, indicating their contribution to $del(p_j)$, to determine a neighborhood $B \subset P$ of size $|B| \leq n < m$ for the LNS destroy-and-repair procedure. The seed agent selection scales linearly w.r.t. m and is important for reducing the cost (Fig. 1) (Phan et al. 2025).

2.4 Bandits with Unobserved Confounders

Multi-armed Bandits (MAB) *MABs or bandits* are basic decision-making problems with a set of choices $A = a \in \mathbb{A}$ (e.g., seed agents) and a stochastic payoff function $\mathcal{X}(a) := C^a$, where C^a is a random variable with an unknown distribution. The goal is to determine a choice a' that maximizes the expected payoff $\mathbb{E}[C^{a'}] \in [0, 1]$ (e.g., potential to reduce costs). The MAB has to balance between trying out choices to estimate the payoffs accurately and exploiting its knowledge by greedily selecting the choice with the highest payoff estimate. This is known as the *exploration-exploitation dilemma*, where exploration can lead to higher payoffs but is time-consuming, while exploitation can converge faster but possibly gets stuck in a local optimum. Our work focuses on *Thompson Sampling* (Thompson 1933).

MABs with Unobserved Confounders (MABUC) We assume a *Structural Causal Model (SCM)* (Pearl 2010) associated with a directed acyclic graph $\mathcal{M}$ (Fig. 1), a set

of *endogenous* (observed) variables $\mathbb{W}$, and a set of *exogenous* (unobserved) variables $\mathbb{U}$. Edges in $\mathcal{M}$ represent functional relationships between endogenous variables, i.e., $w_i \leftarrow f_i(PA_i, u_i) \in \mathbb{W}$, where $PA_i \subseteq \mathbb{W} \setminus \{w_i\}$ and $u_i \subseteq \mathbb{U}$ with probability $\mathbb{P}(u_i)$. Given two endogenous variables A and C, we define the *counterfactual* expression $c = C^a$, meaning "C would be c if A had been a" (Bareinboim, Forney, and Pearl 2015; Forney, Pearl, and Bareinboim 2017).

A MABUC is a generalized MAB, where for each trial, an *intent* $a \in \mathbb{A} \subset \mathbb{W}$ is naturally triggered by some unobserved confounder $U = u \subseteq \mathbb{U}$, as shown in Fig. 1a. Based on the intent, a *counterfactual policy* π selects a choice $A = a' = \pi(a)$, as shown in Fig. 1c. The goal is to find a choice $A = a'$ maximizing the *expected counterfactual payoff* $\mathbb{E}\left[C^{a'}|a\right]$ given an intent a caused by $u \in \mathbb{U}$ (Fig. 2).

Alternatives are *observational selection* (Fig. 1a), i.e., the intent $A = a = f(u)$, and *interventional selection* (Fig. 1b), where $A = a'$ is chosen by a policy independent of $U = u$.

The expected payoffs of committing to an observational, interventional, or counterfactual choice A, according to Fig. 1 and 2, are defined by J^{Obs}, J^{Inv}, and J^{Ctf}, respectively:

$$J^{Obs} = \sum_{a \in \mathbb{A}} \mathbb{P}(a)\mathbb{E}[C^a|a] \tag{1}$$

$$J^{Inv} = max_{a' \in \mathbb{A}} \left\{\mathbb{E}[C^{a'}]\right\} = max_{a' \in \mathbb{A}} \left\{\sum_{a \in \mathbb{A}} \mathbb{P}(a)\mathbb{E}[C^{a'}|a]\right\} \tag{2}$$

$$J^{Ctf} = \sum_{a \in \mathbb{A}} \mathbb{P}(a) max_{a' \in \mathbb{A}} \left\{\mathbb{E}[C^{a'}|a]\right\} \tag{3}$$

where $\mathbb{P}(a) = \mathbb{P}(f(u)) = \mathbb{P}(u)$ denotes the *observational distribution* induced by $U = u$. In heuristic search, $\mathbb{P}(a)$ can represent handcrafted strategies, i.e., human intuition about U (Phan, Chan et al. 2025). When $J^{Obs} \neq J^{Inv}$, then unobserved confounders exist, such that $\mathbb{U} \neq \emptyset$, justifying counterfactual learning (Bareinboim, Forney, and Pearl 2015).

3 Related Work

3.1 Multi-Armed Bandits for LNS

In recent years, MABs have been used to tune learning and optimization algorithms on the fly (Badia et al. 2020; Hendel 2022; Schaul et al. 2019). UCB1 and ϵ-Greedy are commonly used for LNS destroy heuristic selection in *traveling salesman problems (TSP)*, *mixed integer linear programming (MILP)*, and *vehicle routing problems (VRP)* (Chen et al. 2016; Hendel 2022). Some bandit-based MAPF-LNS variants have been proposed to adapt the neighborhood size n, e.g., *BALANCE* (Phan et al. 2024b), or the seed agent choice A, e.g., *ADDRESS* (Phan et al. 2025).

Since classic black-box MAB learning (Fig. 1b) does not dominate handcrafted strategies in general (Fig. 3), we use *counterfactual learning* (Fig. 1c) via truncated MABUCs.

3.2 Machine Learning in Anytime MAPF

Machine learning has been used in MAPF to directly learn collision-free path finding or to guide the node selection in search trees (Alkazzi and Okumura 2024; Huang, Dilkina, and Koenig 2021; Phan et al. 2024a; Phan, Phan, and Koenig 2025; Sartoretti et al. 2019). (Huang et al. 2022; Yan and Wu 2024) propose machine learning-guided variants of MAPF-LNS, where neighborhoods are generated via destroy heuristics (Li et al. 2021). The neighborhoods are then selected using an offline-trained model. Such methods cannot adapt during the search and require extensive prior efforts like data acquisition, model training, and feature engineering.

We focus on *online learning* in MAPF-LNS. Our approach can adjust via *binary payoff signals* on the fly, indicating a successful or failed cost improvement. Instead of acquiring data and engineering features, we leverage *existing handcrafted strategies* for informed online learning.

Model	Expected Payoff per Seed Agent Choice
Observational	$\mathbb{E}[C^a\|a] = \mathbb{E}_{\mathbb{P}(B=b\|A=a)}[C^a\|b,a]$
Interventional	$\mathbb{E}[C^{a'}] = \sum_a \mathbb{P}(a)\mathbb{E}_{\mathbb{P}(B=b\|A=a')}[C^{a'}\|b,a]$
Counterfactual	$\mathbb{E}[C^{a'}\|a] = \mathbb{E}_{\mathbb{P}(B=b\|A=a')}[C^{a'}\|b,a]$

Table 1: Different seed agent selection models w.r.t. Fig. 1.

4 Counterfactual Learning in MAPF-LNS

We now introduce *Truncated Adaptive Counterfactual K-ranked Learning and Evolution (TACKLE)* as a counterfactual learning approach to delay-based MAPF-LNS. Inspired by causal reasoning, e.g., (Bareinboim, Forney, and Pearl 2015), TACKLE first queries a handcrafted heuristic, e.g., from (Li et al. 2021; Tan et al. 2025), as a proxy for human intuition, to select an observational candidate. If the candidate belongs to the top-K set of the most delayed agents, a counterfactual table is queried to sample a seed agent, according to Fig. 2. Otherwise, the observational candidate directly serves as the seed agent for delay-based MAPF. If the time budget for planning were infinite, setting $K = m$ would converge to the best result. However, we assume an anytime setting, where the time budget is restricted. Therefore, we adopt the top-K truncation, suggested in (Phan et al. 2025). Fig. 4 provides an overview of TACKLE.

4.1 Causal Models for Seed Agent Selection

Based on the dependencies in delay-based MAPF-LNS, we can formulate a causal graph $\mathcal{M}$ (Fig. 1) as follows: We define a variable U, which represents all implicit structural properties and aspects of the current solution P, such as the map, start and goal locations, current paths, agent priorities, etc. The seed agent is represented by decision variable A, which may depend on U w.r.t. the selection model. The generated LNS neighborhood is represented by variable B depending on A to enable a selection with linear complexity w.r.t. the agent count m (Section 2.3). Variable C represents the cost improvement and depends on variables B and U.

In the following, we discuss common seed agent selection models and how they relate to our causal graph $\mathcal{M}$. The corresponding expected payoffs are listed in Table 1[1].

[1]For completeness, we also include B in the notation despite manipulating B only indirectly via A for tractability (Section 2.3). The condition $a = f(u)$ is an observable proxy of $u \sim \mathbb{P}(u)$ (Section 2.4) w.r.t. the back-door criterion in Fig. 1a (Pearl 2010).

Observational Selection is based on human intuition or knowledge about U (Fig. 1a) using handcrafted strategies or imitation learning (Huang et al. 2022). In delay-based MAPF, observational methods select seed agents via *Roulette Wheel* sampling proportional to their delays (Tan et al. 2025) or just greedily using *Tabu Lists* (Li et al. 2021). We define $\mathbb{P}(a_j) = \frac{del(p_j)}{\sum_{q \in P} del(q)}$ based on Roulette Wheel.

Interventional Selection ignores U and uses black-box strategies, such as classic MABs explained in Section 2.4 (Fig. 1b). Thompson Sampling, as used in ADDRESS, outperforms observational selection in most benchmark maps, by optimizing $\mathbb{E}[C^{a'}]$ over $a' \in \mathbb{A}$ (Phan et al. 2025).

Algorithm 1: TACKLE for Delay-Based MAPF-LNS

procedure *TACKLE*$(I, K, \mathbb{P})$
  $\mathbb{A}_K^{old} \leftarrow \emptyset$
  $P = \{p_1, ..., p_m\} \leftarrow$ *RunInitialSolver(I)*
  while runtime limit not exceeded **do**
    $a \sim \mathbb{P}(a)$ ▷ Query intent/observational choice
    Select the top-K set $\mathbb{A}_K \subseteq \mathbb{A}$ w.r.t. the delays
    if *orderedList*$(\mathbb{A}_K) \neq$ *orderedList*$(\mathbb{A}_K^{old})$ **then**
      $\langle \alpha_i^a, \beta_i^a \rangle \leftarrow \langle 1, 1 \rangle$ for all table entries
      $\mathbb{A}_K^{old} \leftarrow \mathbb{A}_K$ ▷ Adapt to non-stationarity
    end if
    if $a \in \mathbb{A}_K$ **then**
      for agent a_i in $\mathbb{A}_K$ **do**
        $q_i \sim$ *Beta*(α_i^a, β_i^a)
      end for
      $j \leftarrow$ *argmax*$_i q_i$ ▷ Max. chance to improve
    else
      Use index of the intent a as j
    end if
    $B \sim$ *RandomWalkNeighborhood*(I, P, a_j)
    $B' \leftarrow$ *DestroyAndRepair*(I, P, B)
    if *SoC*$(B) >$ *SoC*(B') **then**
      $P \leftarrow (P \backslash B) \cup B'$ ▷ Replace solution
      $\alpha_j^a \leftarrow \alpha_j^a + 1$ ▷ Success update (if $a \in \mathbb{A}_K$)
    else
      $\beta_j^a \leftarrow \beta_j^a + 1$ ▷ Failure update (if $a \in \mathbb{A}_K$)
    end if
  end while
  return P
end procedure

4.2 MABUC for Seed Agent Selection

Observational and interventional selection are common in delay-based MAPF-LNS, but in general, no strategy dominates the other w.r.t. J^{Obs} and J^{Inv}, as illustrated in Fig. 3 (Nozick 1969). Counterfactual learning always converges to the maximum $J^{Ctf} = 1$ (Eq. 3), and provably dominates both common strategies (Bareinboim, Forney, and Pearl 2015).

For counterfactual learning via MABUCs in delay-based MAPF-LNS, we leverage prior handcrafted strategies, e.g., Roulette Wheel, to sample an intent seed agent $a \in \mathbb{A}$, i.e., a row in Fig. 2 via $\mathbb{P}(a)$. Each row of an intent a represents a classic Thompson Sampling MAB, where each

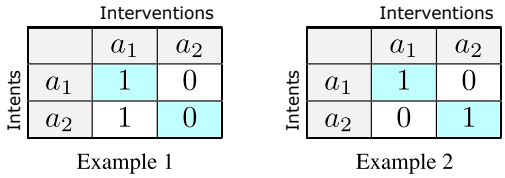


Figure 3: **Example 1**: Interventional dominates; $J^{Inv} = 1 \geq J^{Obs}$. **Example 2**: Observational dominates; $J^{Obs} \geq J^{Inv}$.

row entry represents an interventional seed agent choice $a_j \in \mathbb{A}$. For each choice a_j, we maintain two counters, namely $\alpha_j^a > 0$ for *successful* and $\beta_j^a > 0$ for *failed cost improvements*, respectively. Both counters represent a Beta distribution *Beta*(α_j^a, β_j^a), which models the potential of an agent $a_j \in \mathbb{A}$ to reduce the solution cost as a seed agent, conditional on intent a. *Beta*(α_j^a, β_j^a) has a mean of $\frac{\alpha_j^a}{\alpha_j^a + \beta_j^a}$ and is initialized with $\alpha_j^a = 1$ and $\beta_j^a = 1$, i.e., a 50:50 estimated chance of reducing the solution cost (Chapelle and Li 2011).

4.3 TACKLE: Truncation and Non-Stationarity

Despite counterfactual learning eventually converging to optimal choices, it has to explore all seed agent candidates sufficiently to learn accurate payoff estimates. In large-scale anytime MAPF, where the number of agents m is high and the time budget for learning is restricted, exhaustive exploration can lead to a poor solution quality (Phan et al. 2025).

Inspired by ADDRESS (Phan et al. 2025), we consider the *top-K set* $\mathbb{A}_K \subseteq \mathbb{A}$ of the most delayed agents. If the sampled intent $a \notin \mathbb{A}_K$ is not included in the top-K set, the TACKLE policy adopts it directly as the actual seed agent choice $A = a$ without further intervention. We define the *observational coverage* $R_K = \sum_{a \in \mathbb{A}_K} \mathbb{P}(a) \in [0, 1]$, depending on K and $\mathbb{P}(a)$. The modified expected interventional and counterfactual payoffs J_K^{Inv} (ADDRESS) and J_K^{Ctf} (TACKLE), respectively, are defined by:

$$J_K^{Inv} = max_{a' \in \mathbb{A}_K} \left\{ \sum_{a \in \mathbb{A}_K} \mathbb{P}(a) \mathbb{E}[C^{a'} | a] \right\} + J_{Rest}^{Obs} \quad (4)$$

$$J_K^{Ctf} = \sum_{a \in \mathbb{A}_K} \mathbb{P}(a) max_{a' \in \mathbb{A}_K} \left\{ \mathbb{E}[C^{a'} | a] \right\} + J_{Rest}^{Obs} \quad (5)$$

where $J_{Rest}^{Obs} = \sum_{b \in \mathbb{A} \backslash \mathbb{A}_K} \mathbb{P}(b) \mathbb{E}[C^b | b] \leq (1 - R_K)$.

Since $\mathbb{A}_K$ includes all agents a with the highest observational probability $\mathbb{P}(a)$, we can focus on very few choices ($K \ll m$) in practice to ease exploration (Phan et al. 2025).

With Eq. 4 and 5, we can show that TACKLE dominates ADDRESS and its underlying observational model $\mathbb{P}(a)$.

Theorem 1. *Given a stationary observational distribution $\mathbb{P}(a)$ and $K \geq 1$, TACKLE eventually converges to a counterfactual policy with an expected payoff J_K^{Ctf} that dominates J_K^{Inv} and J^{Obs} (Eq. 1), i.e., $J_K^{Ctf} \geq J_K^{Inv}$ and $J_K^{Ctf} \geq J^{Obs}$.*

Proof. Proving $J_K^{Ctf} \geq J_K^{Inv}$ is analogous to $J^{Ctf} \geq J^{Inv}$, according to (Bareinboim, Forney, and Pearl 2015). $J_K^{Ctf} \geq$

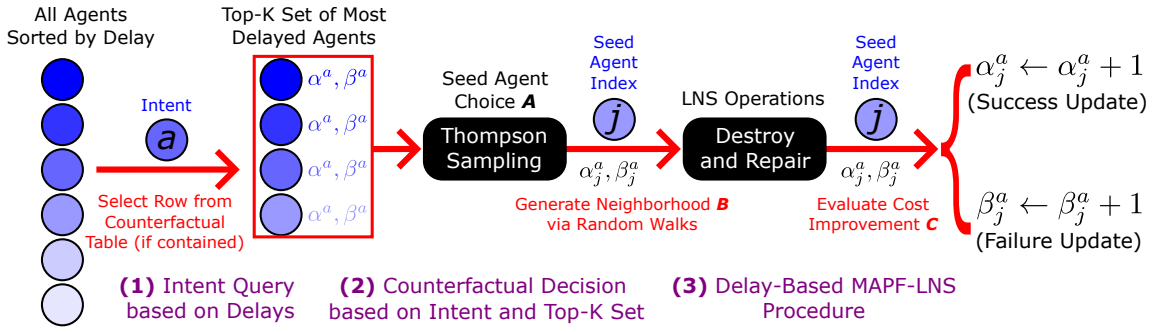


Figure 4: Detailed overview of TACKLE w.r.t. the counterfactual model in Fig. 1c. **(1)** An intent $a \in \mathbb{A}$ is queried using an observational model $\mathbb{P}(a)$ based on human intuition, e.g., Roulette Wheel w.r.t. delays. If $a \in \mathbb{A}_K$, i.e., the top-K set of the most delayed agents, a row is selected from the truncated $K \times K$ counterfactual table, containing the parameters $\alpha_j^a, \beta_j^a > 0$ for Thompson Sampling. **(2)** By sampling $q_j \sim \mathit{Beta}(\alpha_j^a, \beta_j^a)$, a counterfactual seed agent $A = a_j$ is selected. If intent $a \notin \mathbb{A}_K$, then the seed agent is just $A = a$. **(3)** The path of the seed agent A is used to generate an LNS neighborhood $B \subset P$. After running the LNS destroy-and-repair operations on B, the parameters α_j^a or β_j^a are updated w.r.t. the cost improvement (C).

J^{Obs} can be proven by decomposing J_K^{Ctf} into a counterfactual (i.e., $\sum_{a \in \mathbb{A}_K} \mathbb{P}(a) max_{a' \in \mathbb{A}_K} \left\{ \mathbb{E}[C^{a'} | a] \right\}$) and an observational (i.e., J_{Rest}^{Obs}) part, where the analogous parts of J^{Obs} (Eq. 1) have either equal or less expected payoffs. □

To adapt to non-stationarity, TACKLE can leverage the top-K truncation. As the effectiveness of our MABUC depends on the underlying solution structure (U), any change in $\mathbb{A}_K$, e.g., a replacement or swapped ordering, would indicate a substantial change in the solution cost and, therefore, in the solution structure U and distribution $\mathbb{P}(a)$. Thus, all parameters α_i^a and β_i^a are reset to 1 when $\mathbb{A}_K$ has changed.

The full formulation of TACKLE is provided in Algorithm 1 and illustrated in Fig. 4, where I represents the instance to be solved, K truncates the seed agent choices, and $\mathbb{P}$ is the observational distribution over intents $a \in \mathbb{A}$.

4.4 Conceptual Discussion

TACKLE is a counterfactual approach to delay-based MAPF-LNS using human intuition for informed online learning. It can be seen as a generalization of prior delay-based MAPF-LNS variants, where $K = 0$ reduces TACKLE to its observational counterpart, i.e., $\mathbb{P}(a)$. Replacing $\mathbb{P}(a)$ in Line 5 of Algorithm 1 with a static deterministic choice or a distribution that is statistically independent of U, e.g., a random uniform distribution $\mathbb{Q}(a) = \frac{1}{m}$, reduces TACKLE to its interventional counterpart, i.e., ADDRESS.

While TACKLE dominates its interventional (and truncated) counterpart ADDRESS with a $K \times K$ counterfactual table, it is not guaranteed to dominate any non-truncated interventional/counterfactual variant with a full $m \times m$ table, when the optimal[2] choice $a^* \notin \mathbb{A}_K$ is not included in the top-K set. However, in large-scale anytime MAPF, where m is large and the time budget is restricted, non-truncated variants often fail to converge to high-quality solutions.

[2]In the LNS context, optimality refers to the *potential of reducing costs* via seed agent a^*. It does *not* refer to an optimal plan.

TACKLE can serve as a general framework for MAPF-LNS destroy heuristics that rely on alternative decision variables, such as locations and visitation counts (Chen et al. 2024; Li et al. 2021). Investigating analogous counterfactual extensions would be an exciting direction for future work.

TACKLE can also be easily adjusted to other problem classes, such as TSP, MILP, or VRP, when using so-called *worst* or *critical destroy heuristics*, focusing on high-cost variables that "spoil" the structure of the solution (Pisinger and Ropke 2019). We defer such applications to future work.

5 Experiments

Maps We evaluate TACKLE on five maps from the MAPF benchmark set of (Stern et al. 2019), namely **(1)** a `Warehouse` map (*Warehouse-20-40-10-2-2*), **(2)** a `City` map (*Paris_1_256*), **(3)** two `Game` maps *Ost003d* and **(4)** *Den520d*, and **(5)** a `Random` map (*Random-32-32-20*). All maps have different sizes and structures. We conduct all experiments on all publicly available 25 random scenarios per map and report the averages and 95% confidence intervals.

Anytime MAPF Solvers Our implementation of TACKLE is based on the public code of (Li et al. 2022; Phan et al. 2025). We use the notation *TACKLE*(X), where X is an observational strategy as a proxy for human intuition (Fig. 1a and Section 4.1) with X = *Roulette*(*Wheel*) as the default. We use the original implementations of MAPF-LNS, ADDRESS (top-K-truncated), BALANCE, and LaCAM* (Okumura 2023) for comparison.

We always set the maximum neighborhood size to $n = 8$ (except for BALANCE, which automatically adapts n) and $K = 32$ unless stated otherwise. All MAPF-LNS variants use PP to generate initial solutions and repair LNS neighborhoods, as suggested in (Li et al. 2021; Huang et al. 2022).

Compute Infrastructure All experiments were run on a high-performance computing cluster with CentOS Linux, Intel Xeon 2640v4 CPUs, and 64 GB RAM.

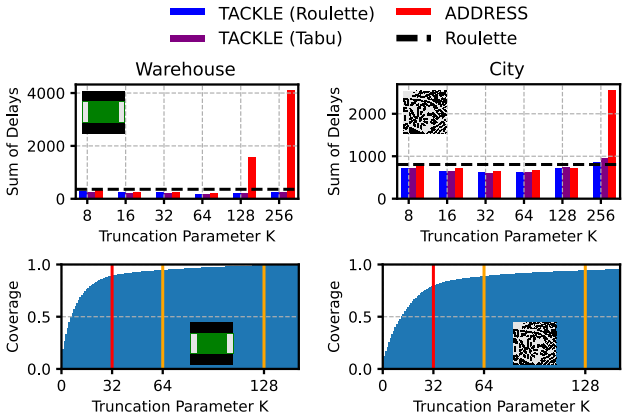


Figure 5: **Top**: Sum of delays for TACKLE with *Roulette Wheel* and *Tabu List*, as well as ADDRESS (as an interventional variant), and *Roulette* as a standalone delay-based MAPF-LNS variant for different values of K with $m = 800$ agents in both maps, a time budget of 60 seconds. **Bottom**: The corresponding observational coverages $R_K = \sum_{a \in \mathbb{A}_K} \mathbb{P}(a)$ of different K w.r.t. the delays of the initial solution.

5.1 Experiment – Truncation Parameter K

Setting We run TACKLE with *Roulette Wheel* and *Tabu List*, as well as ADDRESS (Phan et al. 2025), to evaluate different values of $K \in \{8, 16, 32, 64, 128, 256\}$ on the `Warehouse` and `City` maps with $m = 800$ agents and a time budget of 60 seconds. The results are compared with *Roulette* as a standalone delay-based MAPF-LNS variant.

Results The results are shown in Figure 5. *TACKLE*(*Roulette*) and *TACKLE*(*Tabu*) perform similarly, with $K = 32$ offering a good runtime-quality tradeoff in most scenarios. ADDRESS is more sensitive w.r.t. K and only outperforms *Roulette Wheel* when $K \in [16, 64]$ while performing worst when $K \geq 128$. The bottom row of Figure 5 shows the average observational coverage R_K of different K w.r.t. the delays of the initial solution. In both maps, intervening on $K = 32$ seed agent candidates covers approximately 80% of the observational choices.

Discussion Truncation is useful when the delays are concentrated in very few agents, as represented by the top-K set $\mathbb{A}_K$, which is indicated by the average observational coverages R_K. Due to the restricted time budget, TACKLE and ADDRESS are less effective with large K due to more exploration time – despite higher observational coverage.

5.2 Experiment – Delay-Based MAPF-LNS Variants

Setting Next, we evaluate the search progress of TACKLE, as well as ADDRESS, *Roulette Wheel*, and *Tabu List*, for different time budgets on the `Warehouse` and `City` maps with $m = 800$ agents. The results are compared with a stationary variant of TACKLE without MABUC resets (skipping Lines 7–10 in Algorithm 1).

	TACKLE	LaCAM*
`Random`	$\mathbf{2,988.4 \pm 106}$	$10,146.4 \pm 1399$
`Ost003d`	$\mathbf{6,938.4 \pm 567}$	$38,632 \pm 3925$
`Den520d`	$\mathbf{1490.7 \pm 154}$	$33,604.4 \pm 3823$
`Warehouse`	$\mathbf{411.1 \pm 37}$	$70,107.8 \pm 911$
`City`	$\mathbf{1180.8 \pm 117}$	$48,760.6 \pm 614$

Table 2: Average sum of delays of TACKLE and LaCAM* with 95% confidence intervals, with a time budget of 60 seconds, and the maximum number of agents per map evaluated in Figure 7. The best performance is highlighted in boldface.

Results The results are shown in Figure 6. Both *TACKLE*(*Roulette*) and *TACKLE*(*Tabu*) progress fastest, with ADDRESS being able to keep up in `Warehouse`. The observational variants, *Roulette Wheel* and *Tabu List*, progress notably slower than TACKLE and ADDRESS, indicating the presence of unobserved confounders, according to Section 2.4. *TACKLE*(*Stationary*) performs worst, always achieving the highest sum of delays and AUC.

Discussion The results confirm empirically that conditioning the seed agent selection on observational choices can outperform black-box interventions, e.g., ADDRESS, as well as observational alternatives, e.g., *Roulette Wheel* or *Tabu List*, especially when the time budget is less than 30 seconds. The similar performance of *TACKLE*(*Roulette*) and *TACKLE*(*Tabu*) indicates that both delay-based strategies are suitable to serve as observational models for counterfactual learning. Adapting to non-stationarity, e.g., by resetting the MABUC, is important as the observational distribution $\mathbb{P}(a)$ changes substantially during the search.

5.3 Experiment – State-of-the-Art Comparison

Setting We now compare TACKLE with the original MAPF-LNS, BALANCE, and LaCAM*. We run all solvers on the `Ost003d`, `Den520d`, `Warehouse`, and `City` maps with different numbers of agents m and a time budget of 60 seconds. We also include *Roulette Wheel* as a standalone delay-based MAPF-LNS variant for comparison.

Results The results are shown in Figure 7. TACKLE outperforms all other approaches. BALANCE and *Roulette Wheel* outperform MAPF-LNS. For $m = 1000$, TACKLE improves the cost by at least 76% in `Warehouse` and 60% in `City`, compared with the second-best anytime solver for these maps, i.e., *Roulette Wheel*. Due to the large performance gap, we report the sum of delays for LaCAM* and TACKLE separately in Table 2 for the maximum number of agents per map tested in this experiment. TACKLE (and all other baselines) clearly outperforms LaCAM*.

Discussion Counterfactual learning enables TACKLE to find promising neighborhoods B more efficiently to reduce the solution cost by considering the underlying structure U via the observational model $\mathbb{P}(a)$. In contrast, black-box approaches like the original MAPF-LNS and BALANCE require a considerable amount of exploration time to select suitable destroy heuristics, thus being less effective.

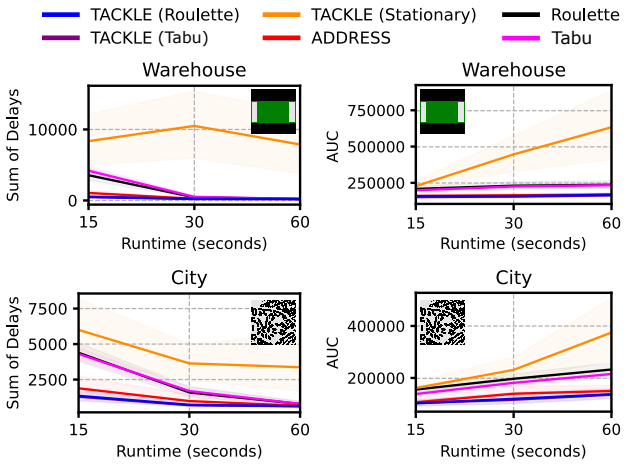


Figure 6: Sum of delays and AUC for TACKLE and a stationary variant without MABUC resets (skipping Lines 7–10 in Algorithm 1) compared with ADDRESS, *Roulette Wheel*, *Tabu List*, as standalone delay-based MAPF-LNS variants (interventional and observational counterparts, according to Section 4.1) for different time budgets (starting from 15 seconds) with $m = 800$ agents. Shaded areas show the 95% confidence interval.

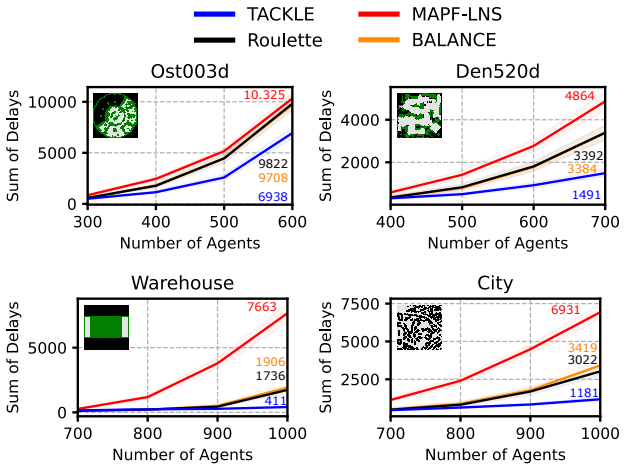


Figure 7: Sum of delays for TACKLE compared with the original MAPF-LNS, BALANCE, and *Roulette Wheel* as a standalone delay-based MAPF-LNS variant for different numbers of agents m and a time budget of 60 seconds. Shaded areas show the 95% confidence interval. The average sum of delays for the maximum number of agents per map is displayed on the right of each plot (colored by solver and sorted by effectiveness). A separate comparison with LaCAM* is provided in Table 2.

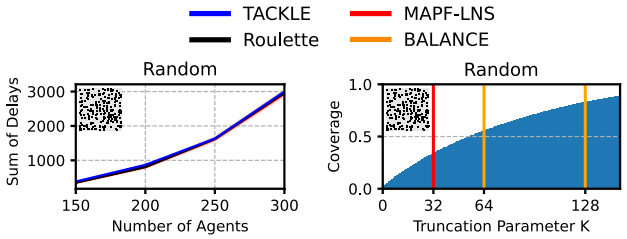


Figure 8: **Left**: Sum of delays for TACKLE compared with the original MAPF-LNS, BALANCE, and *Roulette Wheel* as a standalone delay-based MAPF-LNS variant for different numbers of agents m and a time budget of 60 seconds in the `Random` map. Shaded areas show the 95% confidence interval. **Right**: The observational coverage $R_K = \sum_{a \in \mathbb{A}_K} \mathbb{P}(a)$ of different K w.r.t. the initial delays for $m = 300$.

5.4 Experiment – Limitations of TACKLE

Setting Finally, we compare all solvers from Section 5.3 on the `Random` map with different numbers of agents m and a time budget of 60 seconds.

Results The results are shown in Figure 8, where TACKLE does not outperform any baseline. The average observational coverage R_K w.r.t. the initial delays for $m = 300$ shows that $K = 32$ covers clearly less than 50%.

Discussion Due to the low observational coverage in the `Random` map, TACKLE has less room to dominate its observational counterpart and black-box MAPF-LNS variants strictly. While the top-K truncation is useful in large-scale scenarios, it becomes a limiting factor in small-scale scenarios, as the sorting procedure causes more overhead relative to its potential to reduce costs (Phan et al. 2025).

6 Conclusion

We presented TACKLE, a counterfactual approach to delay-based MAPF-LNS. TACKLE is based on a causal model to select seed agents using human intuition for informed online learning. TACKLE dominates the effectiveness of its handcrafted and black-box learning counterparts in the limit.

Our experiments show that TACKLE significantly outperforms state-of-the-art anytime MAPF solvers like the original MAPF-LNS, ADDRESS, BALANCE, and LaCAM* in large-scale scenarios with up to a thousand agents. The effectiveness of our counterfactual approach is confirmed by its lower costs and AUC compared with alternative delay-based MAPF-LNS variants using handcrafted or black-box learning strategies. TACKLE is most effective in structured and large-scale scenarios, where a small number of sorted seed agent candidates offers sufficient observational coverage for efficient counterfactual online learning.

In the future, we want to include additional variables in our causal graphs to provide explanations and predictions based on data, e.g., about the exact cause of delays. The extensions can also be used for other destroy heuristics leveraging alternative decision variables, e.g., locations and visitation counts (Chen et al. 2024; Li et al. 2021), as well as other NP-hard problem classes, such as TSP, MILP, or VRP.

Acknowledgments

The research at the University of California, Irvine and the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 2544613, 2434916, 2321786, 2112533, as well as gifts from Amazon Robotics and the Donald Bren Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government. Sven Koenig was awarded a WASP Distinguished Guest Professorship at Örebro University.

References

Alkazzi, J.-M.; and Okumura, K. 2024. A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. *IEEE Access*.

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z. D.; and Blundell, C. 2020. Agent57: Outperforming the Atari Human Benchmark. In *International conference on machine learning*, 507–517. PMLR.

Bareinboim, E.; Forney, A.; and Pearl, J. 2015. Bandits with Unobserved Confounders: A Causal Approach. *Advances in Neural Information Processing Systems*, 28.

Chan, S.-H.; Chen, Z.; Lin, D.-L.; Zhang, Y.; Harabor, D.; Koenig, S.; Huang, T.-W.; and Phan, T. 2024. Anytime Multi-Agent Path Finding using Operation Parallelism in Large Neighborhood Search. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2183–2185.

Chapelle, O.; and Li, L. 2011. An Empirical Evaluation of Thompson Sampling. In *Advances in neural information processing systems*, 2249–2257.

Chen, Y.; Cowling, P. I.; Polack, F. A. C.; and Mourdjis, P. 2016. A Multi-Arm Bandit Neighbourhood Search for Routing and Scheduling Problems.

Chen, Z.; Harabor, D.; Li, J.; and Stuckey, P. J. 2024. Traffic Flow Optimisation for Lifelong Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18): 20674–20682.

Cohen, L.; Greco, M.; Ma, H.; Hernández, C.; Felner, A.; Kumar, T. S.; and Koenig, S. 2018. Anytime Focal Search with Applications. In *IJCAI*, 1434–1441.

Forney, A.; Pearl, J.; and Bareinboim, E. 2017. Counterfactual Data-Fusion for Online Reinforcement Learners. In *International Conference on Machine Learning*, 1156–1164. PMLR.

Hendel, G. 2022. Adaptive Large Neighborhood Search for Mixed Integer Programming. *Mathematical Programming Computation*, 1–37.

Huang, T.; Dilkina, B.; and Koenig, S. 2021. Learning Node-Selection Strategies in Bounded Suboptimal Conflict-Based Search for Multi-Agent Path Finding. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Huang, T.; Li, J.; Koenig, S.; and Dilkina, B. 2022. Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, 9368–9376.

Jiang, H.; Zhang, Y.; Veerapaneni, R.; and Li, J. 2024. Scaling Lifelong Multi-Agent Path Finding to More Realistic Settings: Research Challenges and Opportunities. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, 234–242.

Lam, E.; Harabor, D.; Stuckey, P. J.; and Li, J. 2023. Exact Anytime Multi-Agent Path Finding Using Branch-and-Cut-and-Price and Large Neighborhood Search. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021. Anytime Multi-Agent Path Finding via Large Neighborhood Search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 4127–4135.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 10256–10265.

Nozick, R. 1969. Newcomb's Problem and Two Principles of Choice. In *Essays in Honor of Carl G. Hempel: A Tribute on the Occasion of His Sixty-Fifth Birthday*, 114–146. Springer.

Okumura, K. 2023. Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Pearl, J. 2010. The Foundations of Causal Inference. *Sociological Methodology*, 40(1): 75–149.

Phan, T.; Chan, S.-H.; et al. 2025. Counterfactual Online Learning for Open-Loop Monte-Carlo Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25): 26651–26658.

Phan, T.; Driscoll, J.; Romberg, J.; and Koenig, S. 2024a. Confidence-Based Curriculum Learning for Multi-Agent Path Finding. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 1558–1566.

Phan, T.; Huang, T.; Dilkina, B.; and Koenig, S. 2024b. Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 38(16): 17514–17522.

Phan, T.; Phan, T.; and Koenig, S. 2025. Generative Curricula for Multi-Agent Path Finding via Unsupervised and Reinforcement Learning. *Journal of Artificial Intelligence Research*, 82: 2471–2534.

Phan, T.; Zhang, B.; Chan, S.-H.; and Koenig, S. 2025. Anytime Multi-Agent Path Finding with an Adaptive Delay-Based Heuristic. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 39, 23286–23294.

Pisinger, D.; and Ropke, S. 2019. Large Neighborhood Search. *Handbook of metaheuristics*, 99–127.

Ratner, D.; and Warmuth, M. 1986. Finding a Shortest Solution for the NxN Extension of the 15-Puzzle is Intractable. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI'86, 168–172. AAAI Press.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. S.; Koenig, S.; and Choset, H. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters*, 4(3): 2378–2385.

Schaul, T.; Borsa, D.; Ding, D.; Szepesvari, D.; Ostrovski, G.; Dabney, W.; and Osindero, S. 2019. Adapting Behaviour for Learning Progress. *arXiv preprint arXiv:1912.06910*.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2012. Conflict-Based Search For Optimal Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1): 563–569.

Silver, D. 2005. Cooperative Pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 1(1): 117–122.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, 151–158.

Tan, J.; Luo, Y.; Li, J.; and Ma, H. 2025. Reevaluation of Large Neighborhood Search for MAPF: Findings and Opportunities. *18th International Symposium on Combinatorial Search (SoCS)*.

Thompson, W. R. 1933. On the Likelihood that One Unknown Probability exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4): 285–294.

Yan, Z.; and Wu, C. 2024. Neural Neighborhood Search for Multi-Agent Path Finding. In *The 12th International Conference on Learning Representations*.

Yu, J.; and LaValle, S. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1): 1443–1449.