Anytime Multi-Agent Path Finding with an Adaptive Delay-Based Heuristic

Thomy Phan[*1], **Benran Zhang**[*1], **Shao-Hung Chan**[1], **Sven Koenig**[2]

[1]University of Southern California
[2]University of California, Irvine
{thomy.phan,jimyzhan,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 of the solution. Current MAPF-LNS variants commonly use an adaptive selection mechanism to choose among multiple destroy heuristics. However, to determine promising destroy heuristics, MAPF-LNS requires a considerable amount of exploration time. As common destroy heuristics are stationary, i.e., non-adaptive, any performance bottleneck caused by them cannot be overcome by adaptive heuristic selection alone, thus limiting the overall effectiveness of MAPF-LNS. In this paper, we propose *Adaptive Delay-based Destroy-and-Repair Enhanced with Success-based Self-learning (ADDRESS)* as a single-destroy-heuristic variant of MAPF-LNS. ADDRESS applies restricted Thompson Sampling to the top-K set of the most delayed agents to select a seed agent for adaptive LNS neighborhood generation. We evaluate ADDRESS in multiple maps from the MAPF benchmark set and demonstrate cost improvements by at least 50% in large-scale scenarios with up to a thousand agents, compared with the original MAPF-LNS and other state-of-the-art methods.

Code — https://github.com/JimyZ13/ADDRESS

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 *Multi-Agent Path Finding (MAPF)* problem, where the goal is to find collision-free paths for multiple agents with each having an assigned start and goal location. Finding optimal solutions, w.r.t. minimal flowtime or makespan is NP-hard, which limits 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, according to an agent *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 like fast local repairing, integration of primal heuristics, machine learning-guided neighborhood selection, neighborhood size adaptation, and parallelism (Chan et al. 2024; Huang et al. 2022; Lam et al. 2023; Li et al. 2022; Phan et al. 2024b).

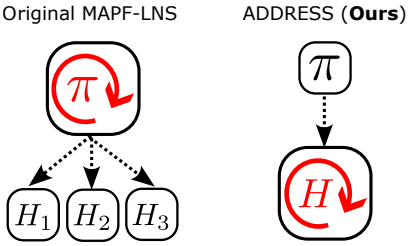

Figure 1: Scheme of our contribution. Instead of using an adaptive selection mechanism π to choose among multiple stationary destroy heuristics H_x (Li et al. 2021), ADDRESS (our approach) only uses a *single adaptive heuristic*.

Current MAPF-LNS variants use an *adaptive selection mechanism* π to choose from the set of destroy heuristics, as illustrated in Figure 1 (Ropke and Pisinger 2006). However, to determine promising destroy heuristics, MAPF-LNS requires a considerable amount of exploration time. As common destroy heuristics are *stationary*, i.e., non-adaptive (Li et al. 2021), any performance bottleneck caused by them cannot be overcome by the adaptive selection mechanism π alone, thus limiting the overall effectiveness of MAPF-LNS.

In this paper, we propose *Adaptive Delay-based Destroy-and-Repair Enhanced with Success-based Self-learning (ADDRESS)*, as a single-destroy-heuristic variant of MAPF-LNS, illustrated in Figure 1. ADDRESS applies restricted Thompson Sampling to the top-K set of the most delayed agents to select a seed agent for adaptive LNS neighborhood generation. Our contributions are as follows:

- We discuss a performance bottleneck of the current empirically most effective destroy heuristic in MAPF-LNS

[*]These authors contributed equally.

and its implications for large-scale scenarios.

- We define an adaptive destroy heuristic, called ADDRESS heuristic, to generate neighborhoods based on the top-K set of the most delayed agents, using multi-armed bandits like Thompson Sampling. We formulate a simplified variant of MAPF-LNS using only our ADDRESS heuristic, as illustrated in Figure 1.
- We evaluate ADDRESS in multiple maps from the MAPF benchmark set (Stern et al. 2019) and demonstrate cost improvements by at least 50% in large-scale scenarios with up to a thousand agents, compared with the original MAPF-LNS and other state-of-the-art methods.

While our paper focuses on MAPF, our ADDRESS heuristic can also be applied to other problem classes, where variables can be sorted by their cost contribution 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* $\mathcal{A} = \{a_1, ..., a_m\}$ 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 $p_i = \langle p_{i,1}, ..., p_{i,l(p_i)} \rangle$ per agent a_i, where $\langle p_{i,t}, p_{i,t+1} \rangle = \langle p_{i,t+1}, p_{i,t} \rangle \in \mathcal{E}$, $p_{i,1} = s_i$, $p_{i,l(p_i)} = g_i$, 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 does not have any 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* $c(P) = \sum_{p \in P} del(p)$. 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 monotonically improves 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 shown to scale up to large-scale scenarios with hundreds of agents (Huang et al. 2022; Li et al. 2021). Starting with an initial feasible plan P, e.g., found via *prioritized planning (PP)* from (Silver 2005), MAPF-LNS iteratively modifies P by destroying $N < m$ paths of the *neighborhood* $A_N \subset \mathcal{A}$. The destroyed paths $P^- \subset P$ are then repaired or replanned using PP to quickly generate new paths P^+. If the new cost $c(P^+)$ is lower than the previous cost $c(P^-)$, then P is replaced by $(P \backslash P^-) \cup P^+$, 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.

MAPF-LNS uses a set of three *destroy heuristics*, namely a *random uniform selection* of N agents, an *agent-based heuristic*, and a *map-based heuristic* (Li et al. 2021). The agent-based heuristic generates a neighborhood, including a *seed agent* a_j with the current maximum delay and other agents, determined via random walks, that prevent a_j from achieving a lower delay. The map-based heuristic randomly chooses a vertex $v \in \mathcal{V}$ with a degree greater than 2 and generates a neighborhood of agents moving around v. All heuristics are randomized but *stationary* since they do not adapt their rules and degree of randomization, i.e., the distributions, based on prior improvements made to the solution.

The original MAPF-LNS uses an adaptive *selection mechanism* π by maintaining selection weights to choose destroy heuristics P (Li et al. 2021; Ropke and Pisinger 2006).

2.3 Multi-Armed Bandits

Multi-armed bandits (MABs) or simply bandits are fundamental decision-making problems, where an *MAB or selection algorithm* π repeatedly chooses an *arm* j among a given set of arms or *options* $\{1, ..., K\}$ to maximize an expected *reward* of a stochastic reward function $\mathcal{R}(j) := X_j$, where X_j is a random variable with an unknown distribution f_{X_j} (Auer, Cesa-Bianchi, and Fischer 2002). To solve an MAB, one has to determine an *optimal arm* j^*, which maximizes the expected reward $\mathbb{E}\big[X_j\big]$. The MAB algorithm π has to balance between exploring all arms j to accurately estimate $\mathbb{E}\big[X_j\big]$ and exploiting its knowledge by greedily selecting the arm j with the currently highest estimate of $\mathbb{E}\big[X_j\big]$. This is known as the *exploration-exploitation dilemma*, where exploration can find arms with higher rewards but requires more time for trying them out, while exploitation can lead to fast convergence but possibly gets stuck in a poor local optimum. We will focus on *Thompson Sampling* and ϵ-*Greedy* as MAB algorithms and explain them in Section 4.2.

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; 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; Kuroiwa and Beck 2023). In most cases, a heavily engineered reward function with several weighted terms is used for training the MAB. Recently, a MAPF-LNS variant, called BALANCE, has been proposed to adapt the neighborhood size N along with the destroy heuristic choice using a bi-level Thompson Sampling approach (Phan et al. 2024b).

Instead of adapting the destroy heuristic selection, we propose a *single adaptive destroy heuristic*, thus *simplifying* the high-level MAPF-LNS procedure (Figure 1). Our destroy heuristic uses *restricted Thompson Sampling* with *simple binary rewards* to select a seed agent from the *top-K set of the most delayed agents* for LNS neighborhood generation, which can also be applied to other problem classes, such as TSP, MILP, or VRP (Pisinger and Ropke 2019).

3.2 Machine Learning in Anytime MAPF

Machine learning has been used in MAPF to directly learn collision-free path finding, to guide the node selection in search trees, or to select appropriate MAPF algorithms for certain maps (Alkazzi and Okumura 2024; Huang, Dilkina, and Koenig 2021; Kaduri, Boyarski, and Stern 2020; Phan et al. 2024a, 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 by stationary procedures, e.g., the destroy heuristics of (Li et al. 2021). The neighborhoods are then selected via 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 *adaptive* approaches to MAPF-LNS using *online learning via MABs*. Our destroy heuristic can adjust on the fly via *binary reward signals*, indicating a successful or failed improvement of the solution quality. The rewards are *directly* obtained from the LNS without any prior data acquisition or expensive feature engineering.

4 Adaptive Delay-Based MAPF-LNS

We now introduce *Adaptive Delay-based Destroy-and-Repair Enhanced with Success-based Self-learning (ADDRESS)* as a simplified yet effective variant of MAPF-LNS.

4.1 Original Agent-Based Destroy Heuristic

Our adaptive destroy heuristic is inspired by the agent-based heuristic of (Li et al. 2021), which is empirically confirmed to be the most effective standalone heuristic in most benchmark maps (Li et al. 2021; Phan et al. 2024b).

The idea is to select a *seed agent* $a_j \in \mathcal{A}$, whose path $p_j \in P$ has a high potential to be shortened, indicated by its delay $\mathit{del}(p_j)$. A random walk is performed from a random position in p_j to collect $N-1$ other agents a_i whose paths p_i are crossed by the random walk, indicating their contribution to the delay $\mathit{del}(p_j)$, to generate a neighborhood $A_N \subset \mathcal{A}$ of size $|A_N| = N < m$ for LNS destroy-and-repair.

The original destroy heuristic of (Li et al. 2021) greedily selects the seed agent with the maximum delay $\mathit{max}_{p_i \in P}\mathit{del}(p_i)$. To avoid repeated selection of the same agent, the original heuristic maintains a *tabu list*, which is emptied when all agents have been selected or when the current seed agent a_j has no delay, i.e., $\mathit{del}(p_j) = 0$. Therefore, the heuristic has to iterate over all agents $a_i \in \mathcal{A}$ in the worst case, which is time-consuming for large-scale scenarios with many agents, introducing a potential performance bottleneck. The original MAPF-LNS cannot overcome this bottleneck because it only adapts the high-level heuristic selection via π, as shown in Figure 1, and thus can only switch to other (less effective) destroy heuristics as an alternative.

4.2 ADDRESS Destroy Heuristic

Our goal is to overcome the limitation of the original agent-based destroy heuristic, and consequently of MAPF-LNS, using MABs. We model each agent $a_i \in \mathcal{A}$ as an arm i and maintain two counters per agent, namely $\alpha_i > 0$ for *successful cost improvements*, and $\beta_i > 0$ for *failed cost improvements*. Both counters represent the parameters of a Beta distribution $\mathit{Beta}(\alpha_i, \beta_i)$, which estimates the potential of an agent $a_i \in \mathcal{A}$ to improve the solution as a seed agent. $\mathit{Beta}(\alpha_i, \beta_i)$ has a mean of $\frac{\alpha_i}{\alpha_i+\beta_i}$ and is initialized with $\alpha_i = 1$ and $\beta_i = 1$, corresponding to an initial 50:50 chance estimate that an agent a_i could improve the solution if selected as a seed agent (Chapelle and Li 2011).

Since the number of agents m can be large, a naive MAB would need to explore an enormous arm space, which poses a similar bottleneck as the tabu list approach of the original agent-based heuristic (Section 4.1). Thus, we restrict the agent selection to the *top-K set* $\mathcal{A}_K \subseteq \mathcal{A}$ of the most delayed agents with $K \leq m$ to ease exploration.

The simplest MAB is *ϵ-Greedy*, which selects a random seed agent $a_i \in \mathcal{A}_K$ with a probability of $\epsilon \in [0, 1]$, and the agent with the highest expected success rate $\frac{\alpha_i}{\alpha_i+\beta_i}$ with the complementary probability of $(1 - \epsilon)$.

We propose a *restricted Thompson Sampling* approach to select a seed agent from $\mathcal{A}_K$. For each agent $a_i \in \mathcal{A}_K$ within the top-K set, we sample an estimate $q_i \sim \mathit{Beta}(\alpha_i, \beta_i)$ of the solution improvement rate and select the agent with the highest sampled estimate q_i. Thompson Sampling is a Bayesian approach with $\mathit{Beta}(1, 1)$ being the *prior distribution* of the improvement success rate and $\mathit{Beta}(\alpha_i, \beta_i)$ with updated parameters α_i and β_i being the *posterior distribution* (Chapelle and Li 2011; Thompson 1933).

Our destroy heuristic, called ADDRESS heuristic, first sorts all agents w.r.t. their delays to determine the top-K set $\mathcal{A}_K \subseteq \mathcal{A}$ of the most delayed agents. Restricted Thompson Sampling is then applied to the parameters α_i and β_i of all agents $a_i \in \mathcal{A}_K$ to select a seed agent a_j. An LNS neighborhood $A_N \subset \mathcal{A}$ is generated via random walks, according to (Li et al. 2021), by adding agents $a_i \in \mathcal{A}$ whose paths are crossed by the random walk. Note that these agents $a_i \in A_N \backslash \{a_j\}$ are not necessarily part of the top-K set $\mathcal{A}_K$.

The full formulation of our ADDRESS heuristic with Thompson Sampling is provided in Algorithm 1, where I represents the instance to be solved, P represents the current solution, K restricts the seed agent selection, and $\langle \alpha_i, \beta_i \rangle_{1 \leq i \leq m}$ represent the parameters for the corresponding Beta distributions per agent for Thompson Sampling.

4.3 ADDRESS Formulation

We now integrate our ADDRESS heuristic into the MAPF-LNS algorithm (Li et al. 2021). For a more focused search, we propose a simplified variant, called ADDRESS, which only uses our adaptive destroy heuristic instead of determin-

Algorithm 1: ADDRESS Destroy Heuristic

procedure *ADDRESSDestroy*$(I, P, K, \langle \alpha_i, \beta_i \rangle_{1 \leq i \leq m})$
  Sort all agents $a_i \in \mathcal{A}$ w.r.t. their delays *del*(p_i)
  Select the top-K set $\mathcal{A}_K \subseteq \mathcal{A}$ w.r.t. the delays
  for agent a_i in $\mathcal{A}_K$ **do**
    $q_i \sim$ *Beta*(α_i, β_i) ▷ Restr. Thompson Sampling
  end for
  $j \leftarrow$ *argmax*$_i q_i$ ▷ Select the seed agent index
  $A_N \sim$ *RandomWalkNeighborhood*(I, P, a_j) ▷ Random walk routine of (Li et al. 2021)
  return $\langle A_N, j \rangle$ ▷ Neighborhood and seed agent
end procedure

ing a promising stationary heuristic via time-consuming exploration, as illustrated in Figure 1.

ADDRESS iteratively invokes our proposed destroy heuristic of Algorithm 1 with the parameters $\langle \alpha_i, \beta_i \rangle_{1 \leq i \leq m}$ to select a seed agent $a_j \in \mathcal{A}$ and generate an LNS neighborhood $A_N \subset \mathcal{A}$ using the random walk procedure of the original MAPF-LNS (Li et al. 2021). Afterward, the standard destroy-and-repair operations of MAPF-LNS are performed on the neighborhood A_N to produce a new solution $P' = (P \backslash P^-) \cup P^+$. If the new solution P' has a lower cost than the previous solution P, then α_j is incremented and P is replaced by P'. Otherwise, β_j is incremented. The whole procedure is illustrated in Figure 2.

The full formulation of ADDRESS is provided in Algorithm 2, where I represents the instance to be solved and K restricts the seed agent selection. The parameters $\langle \alpha_i, \beta_i \rangle_{1 \leq i \leq m}$ are all initialized with 1 as a uniform prior.

4.4 Conceptual Discussion

ADDRESS is a simple and adaptive approach to scalable anytime MAPF. The adaptation is controlled by the learnable parameters α_i and β_i per agent a_i, and the top-K ranking of potential seed agents. Our ADDRESS heuristic can significantly improve MAPF-LNS, overcoming the performance bottleneck of the original agent-based heuristic of (Li et al. 2021) by selecting seed agents via MABs instead of greedily, and restricting the selection to the top-K set of the most delayed agents $\mathcal{A}_K$ to ease exploration.

The parameters α_i and β_i enable the seed agent selection via Thompson Sampling, which considers the improvement success rate under uncertainty via Bayesian inference (Thompson 1933). Unlike prior MAB-enhanced LNS approaches, ADDRESS only uses binary rewards denoting success or failure, thus greatly simplifying our approach compared to alternative MAB approaches (Chen et al. 2016; Chmiela et al. 2023; Hendel 2022; Phan et al. 2024b).

The top-K set enables efficient learning by reducing the number of options for Thompson Sampling, which otherwise would require exhaustive exploration of all agents $a_i \in \mathcal{A}$. The top-K set supports fast adaptation by filtering out seed agent candidates whose paths were significantly shortened earlier. While the top-K ranking causes some overhead due to sorting agents, our experiments in Section 5 suggest that the sorting overhead is outweighed by the performance gains regarding cost and AUC in large-scale scenarios.

Our single-destroy-heuristic approach enables a more focused search toward high-quality solutions without time-consuming exploration of stationary (and less effective) destroy heuristics. Due to its simplicity, our ADDRESS heuristic can be easily applied 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.

Algorithm 2: MAPF-LNS with our ADDRESS Heuristic

procedure *ADDRESS*(I, K)
  $\langle \alpha_i, \beta_i \rangle \leftarrow \langle 1, 1 \rangle$ for all agents $a_i \in \mathcal{A}$
  $P = \{p_1, ..., p_m\} \leftarrow$ *RunInitialSolver(I)*
  while runtime limit not exceeded **do**
    $B \leftarrow \langle \alpha_i, \beta_i \rangle_{1 \leq i \leq m}$ ▷ Distribution parameters
    $\langle A_N, j \rangle \leftarrow$ *ADDRESSDestroy*(I, P, K, B) ▷ See Algorithm 1
    $P^- \leftarrow \{p_i | a_i \in A_N\}$
    $P^+ \leftarrow$ *DestroyAndRepair*$(I, A_N, P \backslash P^-)$
    if $c(P^-) - c(P^+) > 0$ **then**
      $P \leftarrow (P \backslash P^-) \cup P^+$ ▷ Replace solution
      $\alpha_j \leftarrow \alpha_j + 1$ ▷ Success update
    else
      $\beta_j \leftarrow \beta_j + 1$ ▷ Failure update
    end if
  end while
  return P
end procedure

5 Experiments

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

Anytime MAPF Algorithms We implemented ADDRESS with Thompson Sampling and ϵ-Greedy, denoted by *ADDRESS (X)*, where *X* is the MAB algorithm. Our implementation is based on the public code of (Li et al. 2022; Phan et al. 2024b). We use the original MAPF-LNS, MAPF-LNS2, and BALANCE implementations from the respective code bases with their default configurations, unless stated otherwise. We also run LaCAM* from (Okumura 2023).

We set the neighborhood size $N = 8$ (except for BALANCE, which automatically adapts N), $K = 32$, and use Thompson Sampling for ADDRESS and BALANCE, unless stated otherwise. ϵ-Greedy is used with $\epsilon = \frac{1}{2}$. All MAPF-LNS variants use PP to generate initial solutions and repair LNS neighborhoods (Li et al. 2021; Huang et al. 2022).

Compute Infrastructure All experiments were run on a high-performance computing cluster with CentOS Linux,

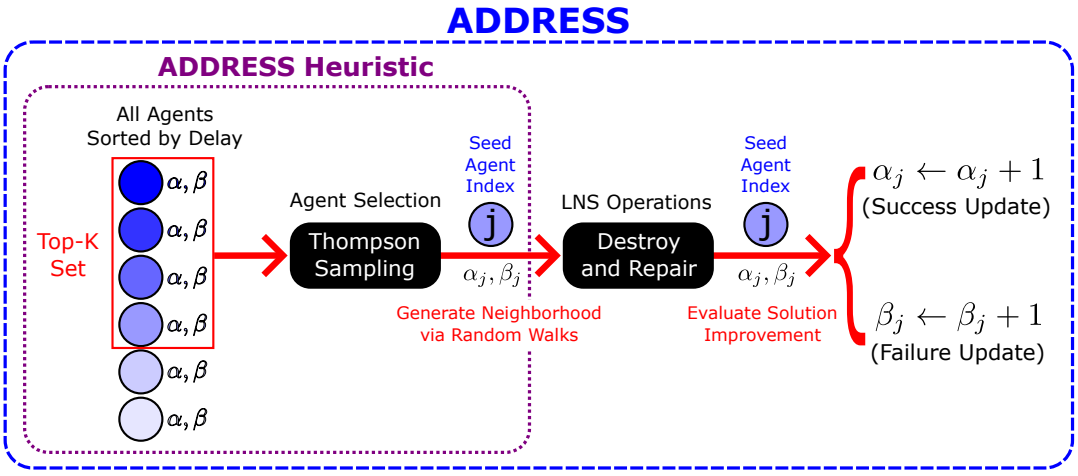

Figure 2: Detailed overview of ADDRESS. For each agent $a_i \in \mathcal{A}$, we maintain two parameters $\alpha_i, \beta_i > 0$. At each LNS iteration, all agents are sorted w.r.t. to their delays. A restricted Thompson Sampling approach is applied to the top-K set of the most delayed agents, according to their samples $q_i \sim Beta(\alpha_i, \beta_i)$, to choose a *seed agent index* j. The path of the seed agent a_j is used to generate an LNS neighborhood $A_N \subset \mathcal{A}$ via random walks. After running the LNS destroy-and-repair operations on A_N, the parameters α_j or β_j of the seed agent a_j are updated, depending on the cost improvement of the new solution.

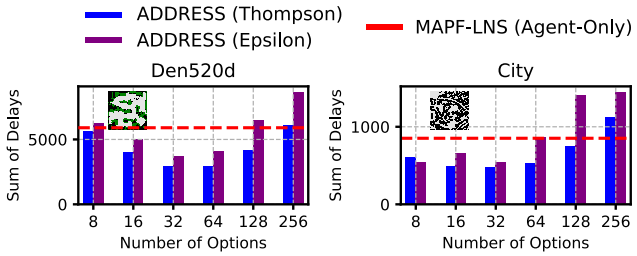

Figure 3: Sum of delays for ADDRESS (using ϵ-greedy or Thompson Sampling) compared with MAPF-LNS (using only the agent-based heuristic) for different numbers of options K with $m = 700$ agents, a time budget of 60 seconds, and $\epsilon = \frac{1}{2}$.

Intel Xeon 2640v4 CPUs, and 64 GB RAM.

5.1 Experiment – Choice of K

Setting We run ADDRESS with Thompson Sampling and ϵ-Greedy to evaluate different choices of $K \in \{8, 16, 32, 64, 128, 256\}$ on the `Den520d` and `City` map with $m = 700$ agents and a time budget of 60 seconds. The results are compared with MAPF-LNS using only the agent-based heuristic of (Li et al. 2021), as a stationary variant.

Results The results are shown in Figure 3. ADDRESS with Thompson Sampling always performs best when $K < 256$. However, ADDRESS is more sensitive to K when using ϵ-Greedy, which only outperforms the original agent-based heuristic, when $8 < K < 64$. In all our test maps, both ADDRESS variants work best when $K = 32$.

Discussion The results indicate that both ADDRESS variants with either Thompson Sampling or ϵ-Greedy can notably outperform the original agent-based heuristic of MAPF-LNS with sufficient restriction via $K < m$. Thompson Sampling is more robust regarding the choice of K.

5.2 Experiment – Delay-Based Heuristics

Setting Next, we evaluate the search progress of ADDRESS with Thompson Sampling and ϵ-Greedy for different time budgets on the `Den520d` and `City` map with $m = 700$ agents. The results are compared with MAPF-LNS using only the agent-based heuristic, as a stationary variant.

Results The results are shown in Figure 4. Both ADDRESS variants outperform the agent-based MAPF-LNS by always achieving lower sums of delays and AUC values, which indicate that ADDRESS always improves faster than the original agent-based heuristic. Thompson Sampling always performs at least as well as ϵ-Greedy.

Discussion The results demonstrate the potential of both ADDRESS variants to improve MAPF-LNS over the original agent-based heuristic for any time budget w.r.t. solution cost and speed of cost improvement. This confirms that the combination of MABs and the top-K set can overcome the performance bottleneck of the original agent-based heuristic (Section 4.1) with negligible overhead (Section 4.4).

5.3 Experiment – ADDRESS and MAPF-LNS

Setting We compare ADDRESS with the original MAPF-LNS using all stationary destroy heuristics of (Li et al. 2021), as described in Section 2.2, for different time budgets on the `Den520d`, `Warehouse`, and `City` map with $m = 700$ agents. To evaluate the dominance of our ADDRESS heuristic over all stationary heuristics, we introduce a MAPF-LNS variant including all commonly used destroy heuristics, as well as our own.

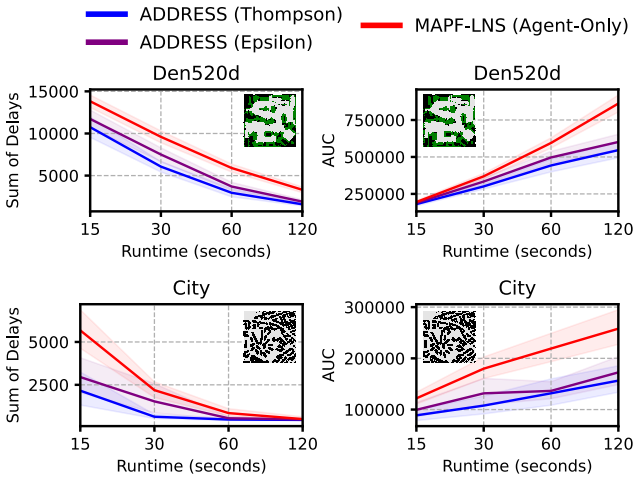

Figure 4: Sum of delays and AUC for ADDRESS (using ϵ-greedy or Thompson Sampling) compared with MAPF-LNS (using only the agent-based heuristic) for different time budgets (starting from 15 seconds) with $m = 700$ agents and $\epsilon = \frac{1}{2}$. Shaded areas show the 95% confidence interval.

Results The results are shown in Figure 5. ADDRESS outperforms both MAPF-LNS variants. The MAPF-LNS variant with our ADDRESS heuristic performs second best in `Den520d` and generally in the other maps with a maximum time budget of 30 seconds. Using our ADDRESS heuristics always leads to a lower average AUC when the time budget is lower than 120 seconds. The selection weights of MAPF-LNS indicate that our ADDRESS heuristic is the dominant destroy heuristic, as it is quickly preferred over all other heuristics.

Discussion The results confirm that our ADDRESS heuristic is more effective than the other heuristics in large-scale scenarios with $m = 700$ agents (Li et al. 2021), as it is consistently preferred by the original MAPF-LNS within less than 10 seconds of runtime. MAPF-LNS, with our ADDRESS heuristic, generally underperforms ADDRESS since it additionally explores the less effective destroy heuristics, whereas ADDRESS directly optimizes the seed agent selection for LNS neighborhood generation.

5.4 Experiment – State-of-the-Art Comparison

Setting Finally, we compare ADDRESS with the original MAPF-LNS, MAPF-LNS2 (which finds feasible solutions by minimizing collisions), BALANCE, and LaCAM*. We run all algorithms on the `Random`, `Ost003d`, `Den520d`, `Warehouse`, and `City` maps with different numbers of agents m and a time budget of 60 seconds.

Results The results with ADDRESS, MAPF-LNS, MAPF-LNS2, and BALANCE are shown in Figure 6. ADDRESS significantly outperforms all other approaches except in `Random`. BALANCE slightly outperforms MAPF-LNS and MAPF-LNS2 in `Den520d` and `Warehouse` with $m \geq 600$. Due to the large performance gap, we report the sum of delays of LaCAM* and ADDRESS separately in Table 1 for the maximum number of agents per map tried in this experiment. ADDRESS (and all other baselines) clearly outperforms LaCAM*.

	ADDRESS	LaCAM*
`Random`	$\mathbf{3,343.52 \pm 120}$	$10,146.4 \pm 1399$
`Ost003d`	$\mathbf{10,788.4 \pm 1219}$	$38,632 \pm 3925$
`Den520d`	$\mathbf{2,646.4 \pm 433}$	$33,604.4 \pm 3823$
`Warehouse`	$\mathbf{3,047.8 \pm 2165}$	$70,107.8 \pm 911$
`City`	$\mathbf{2,645.1 \pm 772}$	$48,760.6 \pm 614$

Table 1: Average sum of delays of ADDRESS and LaCAM* with 95% confidence intervals with a time budget of 60 seconds and the maximum number of agents per map evaluated in Figure 6. The best performance is highlighted in boldface.

Discussion The experiment demonstrates the ability of ADDRESS to outperform the state-of-the-art in large-scale scenarios with up to a thousand agents like in the `Warehouse` or `City` map. The high-level simplification of MAPF-LNS allows ADDRESS to focus its runtime on optimizing seed agents for neighborhood generation without **(1)** exploring less effective destroy heuristics or **(2)** iterating through the whole agent set $\mathcal{A}$, unlike the original agent-based destroy heuristic, used in MAPF-LNS and BALANCE. However, ADDRESS does not outperform the baselines in smaller scenarios, according to $|\mathcal{V}|$ (Figure 6), e.g., in the `Random` map. In this case, the overhead caused by agent sorting and Thompson Sampling outweighs the benefits of ADDRESS. In contrast, MAPF-LNS and BALANCE resort to the map-based heuristic, which is the dominant heuristic in the `Random` map (Li et al. 2021).

6 Conclusion

We presented ADDRESS as a single-destroy-heuristic variant of MAPF-LNS. ADDRESS applies restricted Thompson Sampling to the top-K set of the most delayed agents to select a seed agent for adaptive LNS neighborhood generation. Therefore, ADDRESS avoids time-consuming exploration of several stationary destroy heuristics.

Our experiments show that ADDRESS significantly outperforms state-of-the-art anytime MAPF algorithms like the original MAPF-LNS, MAPF-LNS2, BALANCE, and LaCAM* in large-scale scenarios with up to a thousand agents. The effectiveness of our destroy heuristic is confirmed by its lower costs and AUC compared with the original agent-based destroy heuristic in MAPF and the strong preference by the original MAPF-LNS over all other commonly used destroy heuristics. The combination of Thompson Sampling and the top-K ranking of the most delayed agents enables efficient learning and a stronger focus on promising seed agent candidates through fast adaptation and filtering of agents whose paths were significantly shortened over time. ADDRESS with ϵ-Greedy can also outperform state-of-the-art anytime MAPF with slightly weaker performance than Thompson Sampling, indicating that other MAB algorithms could be used, which we want to investigate in the future.

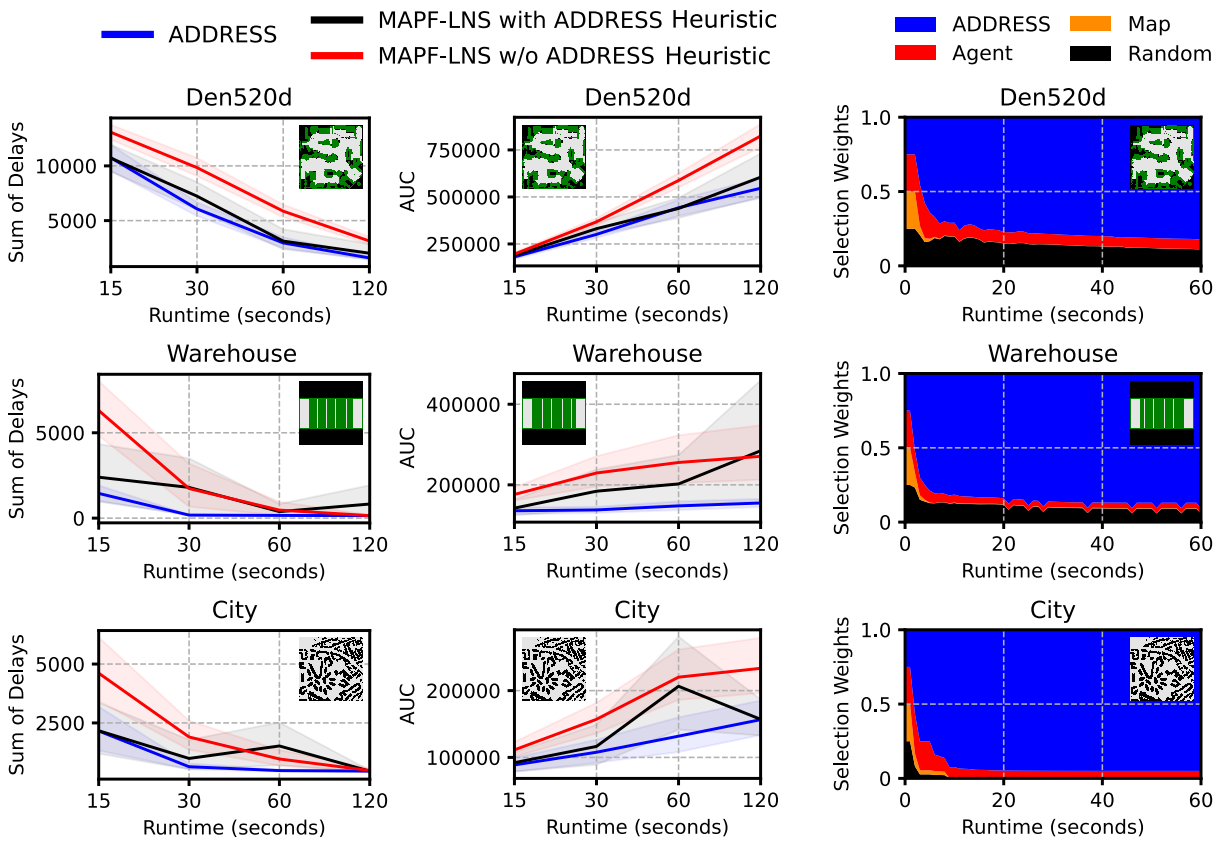

Figure 5: Sum of delays (**left**) and AUC (**middle**) for ADDRESS compared with the original MAPF-LNS (with and without our ADDRESS heuristic) for different time budgets (starting from 15 seconds) with $m = 700$ agents in all maps. Shaded areas show the 95% confidence interval. **Right**: Evolution of the selection weights of MAPF-LNS with our ADDRESS heuristic over time.

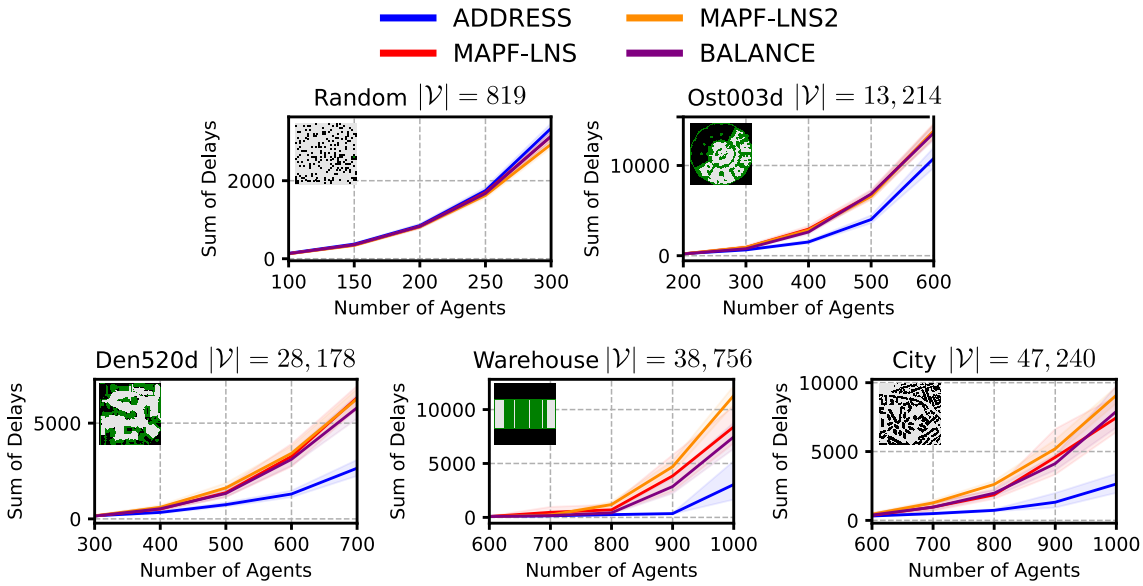

Figure 6: Sum of delays for ADDRESS compared with the original MAPF-LNS (without our ADDRESS heuristic), MAPF-LNS2, and BALANCE for different numbers of agents m and a time budget of 60 seconds. Shaded areas show the 95% confidence interval. The legend at the top applies across all plots. A comparison with LaCAM* is shown in Table 1. $|\mathcal{V}|$ denotes the corresponding map size, i.e., the number of occupiable locations, according to (Stern et al. 2019).

Acknowledgements

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1817189, 1837779, 1935712, 2121028, 2112533, and 2321786 as well as a gift from Amazon Robotics. 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.

References

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

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine learning*, 47(2-3): 235–256.

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.

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.

Chmiela, A.; Gleixner, A.; Lichocki, P.; and Pokutta, S. 2023. Online Learning for Scheduling MIP Heuristics. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 114–123. Springer.

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.

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.

Kaduri, O.; Boyarski, E.; and Stern, R. 2020. Algorithm Selection for Optimal Multi-Agent Pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 161–165.

Kuroiwa, R.; and Beck, J. C. 2023. Large Neighborhood Beam Search for Domain-Independent Dynamic Programming. In *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

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.

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

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.; Driscoll, J.; Romberg, J.; and Koenig, S. 2025. Confidence-Based Curricula for Multi-Agent Path Finding via Reinforcement Learning. *Preprint at Research Square*.

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.

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.

Ropke, S.; and Pisinger, D. 2006. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation science*, 40(4): 455–472.

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.

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.