2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Online Planning for Multi Agent Path Finding in Inaccurate Maps

Nir Malka[1], Guy Shani[1,2], Roni Stern[1]

Abstract— **In multi-agent path finding (MAPF), agents navigate to their target positions without conflict within an environment, typically represented as a graph. Traditionally, the input graph is assumed to be accurate. We investigate MAPF scenarios where the input graph may be inaccurate, containing non-existent edges or missing edges present in the environment. Agents can verify the existence or non-existence of an edge only by moving close to it. To navigate such maps, we propose an online approach where planning and execution are interleaved. As agents gather new information about the environment over time, they replan accordingly. To minimize replanning efforts, we developed methods to identify and replan only for agents affected by observed changes. To scale to larger problems, we defer conflicts resolution expected only in the distant future and adapt single-agent path-finding algorithms to account for map inaccuracies. Experimental results show impressive scalability, solving problems involving over 1000 agents in under 3 minutes.**

I. Introduction

The task in multi-agent path finding (MAPF) is to find conflict-free paths for a set of agents that must move from their current position to some target position. MAPF real-world applications from robotic arms, through robots in automated warehouses [1], to autonomous cars [2]. Most work on MAPF assumes the environment is known, i.e., the input specification (map) is accurate [3]. Yet, in many practical applications the input map might be inaccurate. For example, in an office building, the agents are well aware of the walls, but many doors may be either open or closed. The agents may have some prior information about which doors are likely to be open or closed, but there is no certainty. When the agents identify information that contradicts their early assumptions, they can *replan* given the new information.

We focus on inaccuracies about the edges of the input graph, i.e., some edges are believed to exist but are missing and some edges exist but are believed to missing. Formally, we define the MAPF with Imperfect Map (MAPF-IM), an extension of MAPF in which the planner receives as input a set of edges that are known to exist, a set of edges that are known to be missing, and a set of *uncertain edges* along with whether or not they are likely to exist. To identify the state of an uncertain edge, we assume the agents are equipped with sensing equipment that allows them to observe whether a particular uncertain edge exists, under some constraint. For example, it might be that the agents can only observe edges within some distance, as would a laser-based ranging sensor (LIDAR), or that agents can observe walls in front of them, as with image analysis. The agents can hence identify during runtime inaccuracies, and modify their paths accordingly.

Finding offline solutions for MAPF-IM problems with a relatively large number of uncertain edges results in huge plan trees, branching repeatedly as new information concerning the uncertain edges is revealed. Therefore, we suggest an *online* approach interleaving planning and execution. The agents start acting as if all uncertain edges follow the input assumption, and whenever new information becomes available, replan. This general replanning approach can have several different implementations. One could use an off-the-shelf classical MAPF (CMAPF) planner and replan for all agents whenever a change in the assumed map is observed. However, the observed new information may not be relevant to many agents. Replanning for these agents can be wasteful, both in terms of computation time as well as for system stability. It is therefore desirable to reuse some computations from the previous episode to the next one.

To this end, we propose a novel method for identifying which subset of the agents should replan with the newly obtained information about closed and open edges, and which should continue following their current plan. In MAPF-IM, knowledge of a closed edge may prevent the execution of an existing plan, while knowing that an edge is open may present an opportunity for faster plans to be found. For both cases, our method identifies the set of agents that are affected and should replan given the new information. This allows us, in many cases, to limit replanning to a subset of agents, thus reducing the computational burden. We call this method *Impact Detection (ImpD)*, as it was inspired by the Intepdence Detection framework for CMAPF [4]. We show how to incorporate ImpD method in two algorithms: Prioritized Planning (PP) [5] and Conflict-Based Search (CBS) [6]. Experimental results on standard grid maps demonstrate that ImpD method can greatly reduce computation time when coupled with each of these solvers. Integrating the Rolling Horizon Conflict Resolution (RHCR) framework [7] provides an additional boost in performance by ignoring conflicts that only may occur far in the future.

To further reduce the overhead incurred by replanning, we modify the lower-level single-agent path-finding algorithm to trade off optimality for certainty. This is done by penalizing risky moves, i.e., moves that traverse uncertain edges and may require replanning. We do this within the Explicit Estimation Search (EES) [8] search algorithm, allowing us to control the tradeoff between plan cost and risk.

We evaluate our algorithms and their different configurations experimentally over standard MAPF benchmarks, varying the number of agents and the quantity of possible uncer-

[1]Software and Information Systems Engineering, Ben-Gurion University, nirmalk@post.bgu.ac.il, {shanigu,sternron}@bgu.ac.il
[2]eBay Research, gshani@ebay.com

tain edges. Our findings demonstrate that using a suboptimal MAPF algorithm with our ImpD method, RHCR, and the risk-aware single-agent path-finding algorithm allows scaling to many agents while maintaining a minimal reduction in solution quality.

II. Background

A *classical MAPF* (CMAPF) problem is defined by a tuple $\langle G, A \rangle$ where G = (V, E) is a connected graph, also referred to as map, A = $\{a_1, a_2, \ldots, a_k\}$ is a set of k agents. Agent $a_i \in A$ has a unique start vertex $s_i \in V$ and a unique goal vertex $g_i \in V$. At each discrete timestep, an agent is allowed to perform a single action: either *move* to an adjacent vertex or *wait* in his current vertex.

A *single-agent plan* for an agent a_i is a sequence of vertices $\pi_i = (v_1 = s_i, v_2, \ldots, v_m = g_i)$, where v_t is the location of agent a_i at timestep t. We denote by $\pi_i[t]$ the vertex that agent a_i intends to occupy at timestep t. A *vertex conflict* occurs when two agents occupy the same vertex at the same time, that is, $\pi_i[t] = \pi_j[t]$. An *swapping conflict* occurs when two agents traverse the same edge in opposite directions at the same time, i.e., $\pi_i[t] = \pi_j[t+1] \wedge \pi_i[t+1] = \pi_j[t]$. A solution to a CMAPF problem is a set of classical single-agent plan, $\Pi = \{\pi_1, \ldots, \pi_k\}$, in which there is no conflict between any pair of paths. We assume here that agents wait at their goal vertices permanently. *Sum of costs* (SOC) and *makespan* are common MAPF cost functions, corresponding to the sum or the maximum of the lengths of the agents' paths, respectively. We focus here on SOC.

A. Classical MAPF Algorithms

Prioritized Planning (PP) [5] is a common approach for solving CMAPF problems. In PP, the agents are ordered by some priority function. Then, the agents plan sequentially in order of decreasing priority. When an agent plans, it ignores lower-priority agents while ensuring collisions with plans of higher-priority agents are avoided.

Conflict Based Search (CBS) [6] is an optimal, two-level search algorithm for solving CMAPF. The high-level search of CBS is done over the *constraint tree(CT)*. A node N in the CT represents a set of *constraints* used to coordinate agents to avoid conflicts and a set of single-agent plans that satisfy these constraints. These plans may initially conflict. The cost of a node is the sum of costs of these plans

The root of the CT represents an empty set of constraints and a plan for each agent corresponding to following its shortest path to its goal while ignoring all other agents. At each iteration, CBS expands the lowest-cost node N in the CT that has not been expanded yet. If N has no conflicts, the plans in N form a valid MAPF solution, and the search halts. Otherwise, CBS expands N by generating two CT nodes. In each successor, we add a single constraint of the form (a, x, t) representing that agent a at time t cannot occupy vertex or edge x. The low-level search of CBS is invoked to replan for agent a, ensuring its plan satisfies all its constraints. CBS guarantees completeness by considering both ways of resolving each conflict. It guarantees optimality by performing best-first searches on either high or low levels.

Both PP and CBS require a low-level path-finding algorithm for finding single-agent plans that satisfy a given set of constraints (in PP the constraints are to avoid the plans of higher-priority agents). The state of the art in MAPF research is to use Safe Interval Path Planning (SIPP) [9] as the low-level search [10].

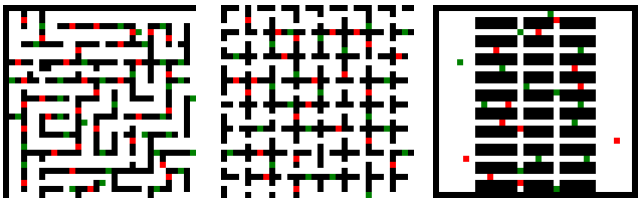
Several suboptimal CBS-based algorithms have also used a *Focal Search* algorithm as the low-level search algorithm [11], [12]. In this work, we utilized a specific focal search-based algorithm called EES [8]. EES is a bounded-suboptimal search algorithm that accepts a parameter $w \geq 1$ and maintains three lists: $cleanup$, $open$, and $focal$. $cleanup$ is sorted according to $f(v) = g(v) + h(v)$, where h is an admissible heuristic. $open$ is sorted according to $\hat{f}(v) = g(v) + \hat{h}(v)$, where $\hat{h}$ is an inadmissible heuristic. Let $best_f$ and $best_{\hat{f}}$ denote the node in $cleanup$ and $open$ with the minimal f and $\hat{f}$ value, respectively, breaking ties randomly. $focal$ includes every node n in $open$ where $\hat{f}(n) \leq w \cdot best_{\hat{f}}$. The nodes in $focal$ are sorted by an additional inadmissible heuristic $\hat{d}$. Let $best_{\hat{d}}$ be the node in $focal$ with the minimal $\hat{d}$. In every iteration EES expands $best_{\hat{d}}$, if $f(best_{\hat{d}}) \leq w \cdot f(best_f)$. Otherwise, it expands $best_{\hat{f}}$ if $f(best_{\hat{f}}) \leq w \cdot f(best_f)$. Otherwise, it expands $best_f$. EES offers great flexibility in the choice of $\hat{f}$ and $\hat{d}$, and is guaranteed to return a bounded-suboptimal solution.

B. Related Works

Cooperative multi-agent planning under partial observability is a well-studied problem. When communication between the agents is free and fast, planning can be centralized, modeling the problem as either a POMDP, in the stochastic case [13], [14], and contingent planning, for non-stochastic environments. Online approaches for solving these problems have been proposed [15]. Our online algorithm is similar in spirit, but leverages the particular settings of MAPF to scale to much larger domains, orders of magnitude beyond the ability of contingent planning solvers. Multi-agent planning when communication is unavailable or costly can be modeled as a Dec-POMDP [16], in the stochastic case, or multi-agent contingent planning [17], for non-stochastic settings. This setting is much harder than MAPF-IM, and we leave to future work extensions of our work to such settings.

MAPF under different forms of uncertainty have been studied before. In MAPF-TU [18], all edges are known but there are lower and upper bounds on the time it takes to traverse each edge. In MAPF-DP [19], [20], agents' actions may be delayed with some probability. In Online MAPF [21] new agents appear over time. Queffelec et al. [22] discuss MAPF in partially known environments, while Levy et a. [23] deal with agents that may stochastically move in a different direction than intended. These papers tackle different uncertainty than we do.

Closest to our MAPF-IM problem is recent work on MAPF under Obstacle Uncertainty (MAPF-OU) [24], where



(a) Maze (b) Rooms (c) Warehouse

Fig. 1: Illustrations of the domain types used in our experiments. Green cells denote $E_{o?}$ and red cells denote $E_{b?}$.

a bounded, relatively small, subset of edges may be blocked. MAPF-IM extends this problem setting in two ways. First, we assume some initial, possibly incorrect, knowledge about the state of each edge. Second, we assume that the set of uncertain edges is very large, and thus, that computing complete offline plan trees for every contingency is intractable. Third, the algorithms proposed for MAPF-OU where all offline approaches, creating plan trees with branches for every contingency. Consequently, their algorithms could not scale to more than 10 agents. In contrast, the MAPF-IM algorithms we propose in this work are online algorithms, and scale to dozens and in some cases hundreds of agents.

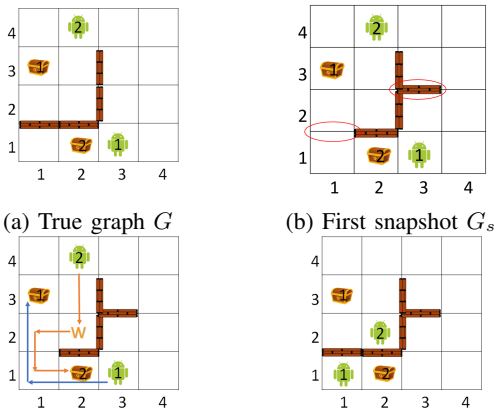
Our problem is different from multi-agent Canadian Travelers Problem [25], which do not consider conflicts between agent, and from Multi-Robot SLAM (MR-SLAM) [26], [27] where most of the environment is unknown and the objective is to map the environment.

III. Problem Definition

In MAPF-IM, a set of agents must navigate in a graph $G = \langle V, E, E_o, E_b \rangle$, where E is a set of edges, E_o is a set of open edges, while E_b is a set of blocked edges, where $E_o \cup E_b = E$ and $E_o \cap E_b = \emptyset$. The agents are given as input a graph specification $G_s = \langle V, E_{ko}, E_{kb}, E_{o?}, E_{b?} \rangle$. G denotes the true world graph, while G_s denotes the current knowledge that the agents have about G, called the *snapshot*. The set of vertices V is identical in G and G_s. E_{ko} denotes a set of edges that are known to be open (traversable), i.e., $e \in E_{ko} \implies e \in E_o$. E_{kb} denotes a set of edges that are known to be blocked (untraversable), i.e., $e \in E_{kb} \implies e \in E_b$. The sets of uncertain edges are denoted by $E_{o?}$ and $E_{b?}$. There may be some edges $e \in E_{o?} \cap E_b$, i.e., an edge the agents assume may be open, but may actually be missing from the graph. Similarly, there may be some edge $e' \in E_{b?} \cap E_o$, that is, an edge the agent assumes is blocked, yet it may exists in the underlying graph.

Let $O(v, e)$ be an observation relation between vertex and edge, representing that when an agent is at vertex v, it observes whether an edge e exists, i.e., $e \in E_o \vee e \in E_b$. Consider the case where, given the current snapshot G_s and the joint action, the agents arrive at the vertices $V_{new} = \langle v_1, \ldots, v_k \rangle$. Let $E_{new} = \{e \in E_{o?} \cup E_{b?} | \exists v \in V_{new}, O(v, e)\}$ be the set of uncertain edges that can be observed from any vertex in V_{new}. After the agents observe the edges in E_{new}, a new snapshot $G_{s'} = \langle V, E'_{ko}, E'_{kb}, E'_{o?}, E'_{b?} \rangle$ is obtained such that:

- $E'_{kb} = E_{kb} \cup (E_{new} \cap E_b)$. The set of non-existingng (blocked) edges that were detected from the new positions of the agents.
- $E'_{ko} = E_{ko} \cup (E_{new} \cap E_o)$. The set of existing (open) edges detected from the new positions of the agents.
- $E'_{b?} = E_{b?} \setminus E_{new}$. The set of all edges that the agents assumed were missing but were detected to exist.
- $E'_{o?} = E_{o?} \setminus E_{new}$. The set of all edges that the agents assumed to exist but were detected to be missing.



(a) True graph G (b) First snapshot G_s

(c) Snapshot optimal plan (d) Second snapshot $G_{s'}$

Fig. 2: Example of a 4×4 grid MAPF-IM problem. Bricks represent walls, and crates represent goals. Blue and orange lines denote plans. Erroneous edges are marked by red ovals.

It is desirable to reach the goal in a minimal amount of steps. However, agents may not know initially which path would yield the lower number of steps, as they have only partial information about open and blocked edges. We hence restrict our attention to *snapshot optimality*. A MAPF-IM algorithm is *snapshot optimal* if, given a snapshot G_s, the computed actions for the agents are a prefix of a minimal cost conflict-free plan, assuming that G_s is accurate. That is, assuming that the true graph is $G' = \langle V, E_{ko} \cup E_{o?} \rangle$.

Example 1: Figure 2 presents a MAPF-IM problem on a 4-neighborhood grid. Vertices V correspond to grid cells, while edges E connect adjacent cells in four cardinal directions. Examining the true graph G (Figure 2a), edges such as $\langle (1,1), (2,1) \rangle \in E$ link cells $(1,1)$ and $(2,1)$, and $\langle (1,1), (1,2) \rangle \in E$ link cells $(1,1)$ and $(1,2)$. Open edges (E_o) indicate no wall between cells, while blocked edges (E_b) indicate the presence of walls.

In the initial snapshot (Figure 2b), $E_{o?}$ mirrors E_o except for $\langle (1,1), (1,2) \rangle \in E_{o?} \setminus E_o$ where the agent incorrectly assumes no wall. Conversely, $E_{b?}$ includes E_b except for $\langle (3,2), (3,3) \rangle \in E_{b?} \setminus E_b$ where the agent erroneously assumes a wall. These discrepancies are highlighted with red ovals. The agents are unaware of these inaccuracies.

Figure 2c depicts the planned paths. To prevent a potential conflict at $(1,2)$, agent 2 waits at $(2,2)$. After the first joint action, no new information is revealed, so the snapshot remains unchanged. However, after the second action, agent a_1

reaches $(1,1)$ and detects the wall blocking $\langle(1,1),(2,1)\rangle$, confirming $\langle(1,1),(2,1)\rangle \in E_b$. This results in a new snapshot $G_{s'}$ (Figure 2d) where $\langle(1,1),(2,1)\rangle \in E'_{kb} \wedge \notin E'_{o?}$. Future plans, hence, avoid traversing this edge.

IV. Online Planning Framework

We now describe our online replanning approach for solving MAPF-IM problems. The agents act based on their current knowledge of the world, assuming that the given snapshot is accurate. Whenever new information becomes available to one agent, it communicates the information to all agents. Then, the agents may replan given the new information. We use an online replanning approach because an offline planner, that plans for all contingencies ahead before acting, must, in our case, create huge plan trees, branching at every step, given the multitude of possible observations. Below, we describe our online algorithms. We begin with a generic replanning algorithm, emphasizing the main components of all replanning methods. The algorithm begins with initializing the multi-agent plan. Then, the agents start acting based on this plan. At each step the agents update the current snapshot given the observation function O, that is, all edges that are observable from the current positions of the agents are observed. The agents communicate their observations to form a centralized new snapshot. If the snapshot is identical to the previous one, then the existing plan is still valid, and the agents can continue to act. The agents replan when the new snapshot is different, either because an edge deemed to be open is observed to be blocked or because an edge believed to be blocked is actually open. This continues until all agents reach their goals.

The efficiency of this generic replanning algorithm greatly depends on how to initialize the set of plans and how to replan given new observations. A naive approach, which we call *AlwaysReplan*, initializes the plans using any CMAPF solver and replans using the same CMAPF solver from scratch following every observation. If the CMAPF solver is optimal, AlwaysReplan ensures snapshot optimality, as the agents recompute their plans given the current snapshot. However, this approach can be wasteful since not every observation is relevant to all agents. In Section IV-A we present a novel method that we call *Impact Detection (ImpD)*, for identifying which agents whose plan may be or should be impacted by the new observation. Only these agents need to replan. This method is described in Section IV-A.

Another source for potential inefficiency is that a CMAPF solver resolves all conflicts before starting to act. However, in MAPF-IM, we expect that new information would cause the agents to alter their paths. Thus, possible conflicts that would occur far from the current positions of the agents may no longer be relevant, given the revised paths. Using this intuition, we propose a second approach that defers conflict resolution until a conflict is imminent, occurring within at most r timesteps from the current positions of the agents.

Algorithm 1: ImpD + CBS

```
1  Replan(G_s, Π, A, t)
       Input: : new snapshot G_s, current joint plan Π, set of
                agents A, current timestep t
2      G ← ⟨V, E_ko ∪ E_o?⟩
3      C ← {c ∈ C_π : t(c) > t}
4      groups ← Partition(C)
5      foreach g ∈ groups do
6          if ∃a_i ∈ g, affected(π_i, G_s) then
7              C ← C \ C_g
8              Π_g, C_g ← CBS(G, g)
9              Revise Π using Π_g
10             C ← C ∪ C_g
11     while ∃g_i, g_j ∈ groups with conflicting paths do
12         g ← g_i ∪ g_j
13         groups ← (groups ∪ g) \ g_i, g_j
14         C ← C \ C_g
15         Π_g, C_g ← CBS(G, g)
16         Revise Π using Π_g
17         C ← C ∪ C_g
18     C_π ← C
19     return Π
```

A. Reducing Replanning Computations

We propose two methods that aim to replan for only a subset of the agents: one based on Prioritized Planning (PP) and one based on CBS. Both methods first identify an initial set of agents that should consider modifying their plans. We referred to such agents as *affected agents*.

1) Identifying affected agents: If an edge e_1 considered to be open ($e_1 \in E_{o?}$) is now known to be blocked ($e_1 \in E_{kb}$), then all agents that planned to traverse e_1 are regarded as affected agents, as they must now modify their paths. Identifying these agents is trivial. If an edge e_2 that was considered to be blocked ($e_2 \in E_{b?}$) is now known to be open ($e_2 \in E_{ko}$), passing through e_2 may help some agents to shorten their paths.

Identifying which agent may gain from passing through $e_2 = (u, v)$ is more challenging. We consider agent a_i to be affected by e_2 only if the shortest path from its current location, denoted s_i, to its goal, denoted g_i, via e_2 is less than its incumbent path. That is, $d^*(s_i, u) + w(e_2) + d^*(v, g_i) < |\pi_i|$ where d^* is the minimum cost of traveling between two vertices, π_i is the agent's incumbent path, and $w(e_2)$ is the cost of traversing the newly discovered edge. To reduce computations, as $E_{b?}$ is known, we compute $d^*(u, v)$ for every $e = (u, v) \in E_{b?}$ only once during preprocessing, assuming all uncertain edges are passable. This reduces the agent relevance identification time, at the cost of unsoundness — possibly identifying an agent as affected when it is not. In our preliminary results, this tradeoff was justified.

2) ImpD+PP : starts by identifying the agents affected by the new snapshot. It then plans for these agents using PP, avoiding conflicts with unaffected agents by considering them to be of higher priority. There are two reasons why this call to PP may fail [28]: either all paths of one of the agents to its goal are blocked by the unaffected agents, or existing plans of the unaffected agents make collisions with an affected agent unavoidable. To mitigate such failures,

the algorithm includes a sample of interfering unaffected agents in the set of affected agents. This adjustment enables the algorithm to explore more flexible and feasible path solutions that were previously unattainable due to rigid priority constraints. Joint replanning allows for simultaneous pathfinding for both affected and selected unaffected agents, balancing their priorities and enhancing conflict resolution. Consequently, this approach results in a higher success rate by reducing instances of completely blocked paths and unavoidable collisions. This is reminiscent of the Local Neighborhood Search for MAPF [29].

3) ImpD+CBS: is based on CBS [6] and is designed to be snapshot optimal. Thus, it may replanning for more agents than the set of agents initially detected as affected. Algorithm 1 describes ImpD+CBS in more details. First, we identify a set of conflicts C that were computed for the previous plan Π, and are still relevant, i.e., occur after the current step t (line 3). Then, we partition the agents into disjoint groups (line 4) given C, where a pair of agents a_i, a_j are in the same group if there exists a conflict between them in the set C. For every such group of agents g, we call CBS on that group only if g includes an affected agent (lines 5-10). In addition, we replace the conflicts relevant for g with the new conflicts generated during replanning (lines 7,10). After planning for each group g independently, we check whether new plans for agents from different groups now conflict (line 11). As long as such two groups g_i, g_j exist, we join them into a new group g (lines 12-13), we remove the conflicts associated with the joint group g (line 14), and replan for g (line 15). We then check again whether a new conflict exists with another group.

4) Local Conflict Resolution (LCR): Limiting the horizon in which conflicts are considered during planning is effective in different MAPF contexts [5], [7]. This technique is expected to work especially well for MAPF-IM, since agents are likely to change their paths due to new information, and thus resolving conflicts that far in the future is wasteful. Specifically, when replanning we only consider conflicts that are at most r steps from the current state, where r is a parameter. To ensure a safe execution, we monitor the agents' location during execution and call for replanning if a conflict is about to occur within the next r steps. We refer to this approach as *Local Conflict Resolution* (LCR).

B. Uncertainty-Aware Low-Level Search Policies

The low-level search in MAPF algorithms such as PP and CBS usually aims to find the lowest cost path of a given agent to its goal subject to some constraints. SIPP [9] was used for this purpose in our experiments. Next, we examine three alternative *uncertainty-aware low-level search policies*, which are based on EES [8].

The *Risk-Averse* policy attempts to avoid replannings overheads by avoiding passing through edges that may be blocked as much as possible. If we knew the probability that each edge is open or blocked, we could quantify the risk of a given path. Instead, we assume risk is correlated with the amount of possible blocked edges in the plan and bias the search towards paths to avoid such risks by adjusting the ordering of nodes in EES's $focal$. Technically, when a vertex v is generated, we set $\hat{d}(v) = h(v) + penalty(p(v), v)$, where $p(v)$ is the immediate predecessor of v along the shortest path. Thus, we can encourage the search to avoid different types of edges.

The *Explorative* policy aims to gather more information about the uncertain edges, potentially reducing the overall solution cost. For this, we reward getting close to a possibly blocked edge during the execution of the incumbent plan. This is done by placing a negative penalty (=reward) over edge (u, v) when computing $\hat{d}(v)$ as noted above.

The *Hybrid* policy integrates the two approaches by penalizing traversing possibly open edges and rewarding getting adjacent to possibly blocked edges.

C. Theoretical Results

We say that an observation relation O is *truthful* if for every vertex v and pair of vertices (u, w) it holds that $(v, (u, w)) \in O$ if and only if $v = u$ and (v, w) is an edge in the true graph G. That is, once an agent observes the state of an edge, that state is indeed the state of the edge in the true graph.

Theorem 1: Given a truthful observation function, the Online Replanning framework demonstrates: (1) soundness, ensuring that the executed plan avoids conflicts and obstacles, contingent on the soundness of the CMAPF solver; and (2) completeness, assuring that a solution will eventually be found if one exists, provided the CMAPF solver is complete.

Proof: **Soundness.** Before taking any action, agents observe their environment. If new information is obtained, we adjust the plan using a sound CMAPF planner. Given the truthfulness of O, only valid actions are considered for the agents in the current iteration. Consequently, agents execute only legal actions at each iteration.

Completeness. Each planning phase generates a plan based on observed information. If the CMAPF fails to plan with the current knowledge, meaning all edges $e \in E_{ko} \cup E_{o?}$ are exhausted, we include the edges assumed to be blocked, i.e., we apply CMAPF search on a graph $G = \langle V, E_{ko} \cup E_{o?} \cup E_b \rangle$. Thus, we will eventually uncover the true graph. ■

By definition, Online Replanning is snapshot optimal when using *AlwaysReplan* and an optimal CMAPF algorithm such as CBS. Thus, using PP for replanning means losing the snapshot optimality property. Next, we prove that ImpD+CBS preserves this property.

Theorem 2: Online Replanning with ImpD+CBS is sound and snapshot optimal.

Proof: Let Π be a snapshot optimal plan, and let $a_1, a_2 \in A$ and $\pi_1, \pi_2 \in \Pi$ be a pair of agents and their corresponding plans. Suppose that a_1, a_2 were detected to conflict, either in a vertex conflict or in a swapping conflict, i.e., $\exists t' : (\pi_1[t'] = \pi_2[t']) \vee ((\pi_1[t'] = \pi_2[t'-1]) \wedge (\pi_1[t'-1] = \pi_2[t']))$. If $t' < t$ it follows that there are no conflicts between π_1, π_2 from the current timestep. Therefore, the

agents do not affect each other; If one of them vacates a location, it will not affect the other, as it already follows its shortest path. Yet, this is not the case when $t' > t$, since one of the agents was constrained to occupy some vertex that perhaps coerced him to deviate from his shortest path. It suffices to show that a group with no affected agent may follow its members' current plans. Let g be such a group, i.e., no affected agent belongs to g. By negation, we assume that the ongoing plan for g, denoted by Π_g, is either unsound or sub-snapshot-optimal. If Π_g is unsound, it implies that there exists an agent $\hat{a} \in g$ whose planned path traverses a blocked edge. Hence, by definition, a is affected. In contradiction to the initial assumption. If Π_g is not a snapshot-optimal plan w.r.t g, it follows that exists a plan $\hat{\Pi}$ such that $SOC(\hat{\Pi}) < SOC(\Pi)$. Thus, at least one agent $a \in g$ may shorten its current plan. Therefore, by definition, a is affected. Contradiction. ■ ■

V. EMPIRICAL EVALUATION

We implemented the proposed algorithms in C++ [1] and compared them experimentally on five diverse grid maps [2] taken from the standard grid MAPF benchmark [3], namely `maze-128-128-10` (Maze), `warehouse-20-40-10-2-2` (Warehouse), `Paris-1-256` (Paris), `den520d`, and `ost003d` for problems with up to 2000 agents. All experiments were conducted on a Ubuntu 22.04.2 Intel Core i7, 2.8GHz, 16GB RAM machine.

A. Experimental Setup

The MAPF benchmark provides 25 scenario files for each grid specifying start and goal locations for different numbers of agents. To define MAPF-IM problems, we created the true graph by adding and removing edges from a given grid. For each grid G we created 50 different initial snapshots G_s^i where i denotes the size of $E_{o?}$, $E_{b?}$. Each snapshot was generated by randomly sampling edges for $E_{o?}$, and $E_{b?}$. CBS-based planners were evaluated on four different initial snapshots, where $i \in 25, 50, 75, 100$ and the number of agents was limited to 100. PP based planners, that scale better, were evaluated on more complex grids where i ranges from 40 to 1000 and the number of agents was up to 2000. In all experiments, we assume that $E_{ko} = \emptyset$ initially, while E_{kb} contains all non-neighbor pairs of positions in the grid. That is, we assume that all edges are uncertain to begin with, except for shortcuts between remote positions that we know do not exist. The uncertain edges were strategically placed to affect the agents' paths to their goals. Figure 1 shows grid examples where green and red cells denote possibly blocked and traversable edges, respectively.

We evaluated every combinations of algorithms presented in this paper [3], namely, using PP or CBS as the underlying MAPF solver, with or without ImpD, Full conflict resolution or LCR, and SIPP or EES. For EES, we experimented with the three uncertainty-aware low-level search policies we proposed — Risk-averse, Explorative, and Hybrid. The main metrics we consider are (1) success rate, i.e., the ratio of problems solved within the 3-minute time limit; and (2) sum of costs (SOC).

[1]MAPF-IM GitHub Repository
[2]Moving AI Benchmarks
[3]All experiment results are available in the GitHub repository

B. Results

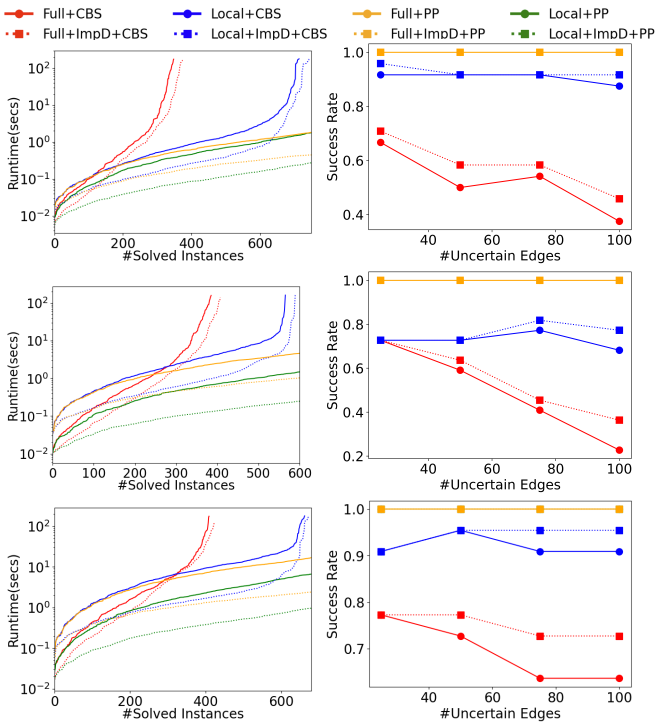


TABLE I: Evaluating the effects of LCR framework and ImpD on ost00d(top), den520(middle), Paris(bottom) maps.

The plots presented in Table I (left) depict the relationship between the number of solved problems (x-axis) and runtime (y-axis) for each algorithm, showcasing several key trends across different grid types. Most notably, our ImpD method (dashed line), which efficiently identifies agents that do not require replanning, consistently demonstrates a significant performance advantage across all configurations. This enables a greater number of problems to be solved within the same runtime, highlighting the method's effectiveness. The impact of ImpD is particularly pronounced in the den520d and Paris maps, where the Local+ImpD+CBS configuration solved all instances approximately five times faster than Local+CBS without ImpD. Furthermore, this performance boost extends to PP-based planners, where the incorporation of ImpD yields similarly substantial improvements.

PP-based solvers demonstrated superior performances by sequentially handling agents, reducing search space and computational time. In contrast, CBS-based algorithms face exponential runtime growth with increasing conflicts, resulting in significant computational overhead.

Local conflict resolution is significantly more efficient in terms of runtime compared to full conflict resolution. However, its short-term focus might lead to more frequent

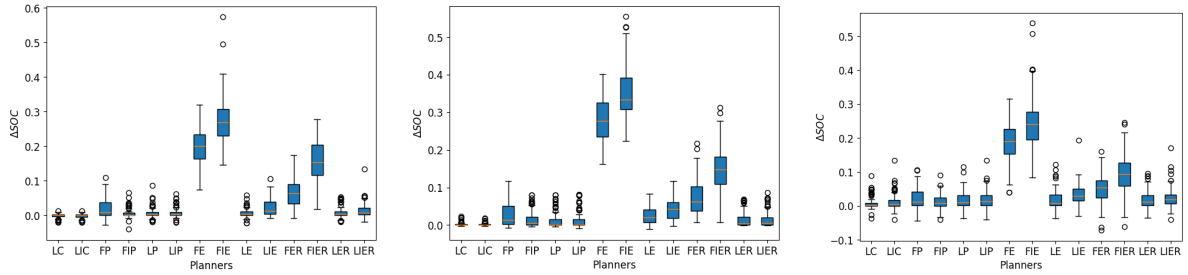


Fig. 3: Δ SOC results for ost003d (left), den520d (middle), and Maze (right).

replanning, consuming time and computational resources that could otherwise be used to solve more problems under a full replanning approach.

Example 2: Figure 4 illustrates a scenario where full conflict resolution is better than local resolution. Two agents aim to reach their respective treasure boxes, with their intended paths shown as solid lines. Employing local conflict resolution with a horizon of $r = 1$ leads to eight CT node expansions, as one agent continually yields to the other to achieve its goal. Conversely, full conflict resolution results in merely one CT node expansion. Here, the potential conflict at (3, 3) is anticipated and resolved early, prompting one agent to take an alternative route (the dashed line).

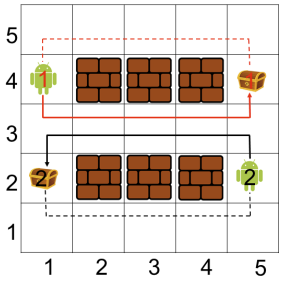


Fig. 4: Full vs. local conflict resolution example

Next, we analyze the impact of the number of agents and uncertain edges on algorithm performance. Table I(middle) shows success rates as uncertain edges increase, and Table II(middle) does the same for agents. In each snapshot, we incrementally added uncertain edges or agents and tested the algorithms. Each snapshot was built by strategically placing uncertain edges in either $E_{o?}$ or $E_{b?}$ (Figure 1). The effect of ImpD and Local is clear; both reduce runtime and increase success rate.

Table II shows the benefits of using our modified EES as the low-level search. All tested algorithms were integrated with the LCR framework, PP and ImpD. Regardless of the policy, all EES-SIPP low-level planners demonstrate at least a twofold improvement in speed compared to the baseline (red line). The ImpD-enhanced EES variants consistently exhibit superior performance, not only in terms of computational efficiency but also in their ability to scale effectively across various problem configurations. Overall, the risk-averse policy consistently outperformed the other policies. In the maps used for our experiments, avoiding uncertain edges proved to be a more effective strategy, as it reduced the need for replanning.

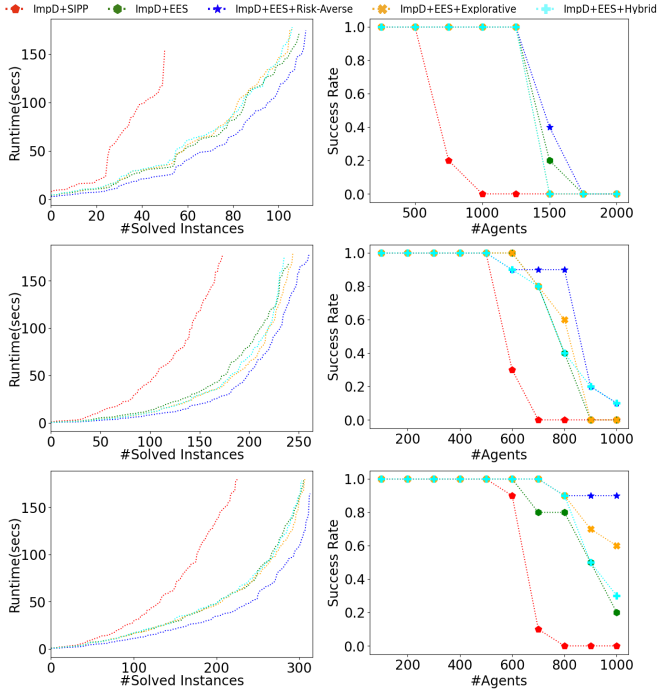


TABLE II: Evaluating the impact of various low-level search algorithms' policies on the Warehouse (top), den520 (middle), and Paris (bottom) maps.

We also analyze the cost (SOC) the solutions returned by each algorithm. Preliminary results showed that, as expected, the snapshot optimal algorithms (Full+CBS, Full+ImpD+CBS) yield the lowest SOC. Thus, we report the difference between the SOC they obtain and the SOC of Full+CBS, normalized by the SOC of Full+CBS. We denote this by ΔSOC. Figure 3 shows a Box and Whisker plot of the ΔSOC for each grid and algorithm, where F=full, L=local, C=CBS, P=PP, I=ImpD, E=EES, R=Risk-Averse, e.g., FIER represents the Full+ImpD+EES+Risk-Averse algorithm. We do not show results for Full+CBS and Full+ImpD+CBS as they yield the reference snapshot optimal SOC value (i.e., ΔSOC=0).

In all grids Local+CBS and Local+ImpD+CBS yield better (i.e., lower) SOC than the PP-based algorithm. This is because CBS is geared towards finding joint lowest-cost

solutions, while in PP each agent aims to optimize its own plan. Interestingly, the Local algorithm either matches or surpasses the performance of the Full algorithm in terms of solution quality. In the Local algorithm an agent alters its path from the shortest one only upon detecting an imminent collision within a predefined distance r. The Full algorithm employs a proactive approach, whereby agents are initially rerouted from their shortest paths to preemptively mitigate potential collisions. However, given the dynamic nature of route changes, this preemptive deviation from the shortest paths in the Full algorithm appears to be superfluous.

Incorporating ImpD in most cases did not affect solution cost, as expected, except for configurations of Full, PP and EES, where using ImpD had a negative impact on the solution cost. ImpD+PP implicitly assigns lower priorities to affected agents by considering only their paths for replanning. Consequently, these agents may choose long unnecessary detours, increasing the SOC. This could be mitigated through alternative agent prioritization strategies, particularly by incorporating all agents' plans into the replanning process.

VI. Conclusion and Future Work

This paper studies the MAPF-IM problem, where the problem solver has limited knowledge about the traversability of certain edges. We proposed an online replanning framework for solving MAPF-IM problems and explore several ways to implement it. To reuse existing plans as much as possible while adapting to observed changes, we propose a novel method that identifies which agents should not replan as they are likely unaffected by the observed changes. We also proposed a way to consider the uncertainty in a lower-level path planning procedure. We evaluate these methods experimentally, showing that they greatly reduce runtime. Our study paves the way for other research inquiries, such as addressing the challenges of solving MAPF-IM in settings characterized by communication delays or high costs, exploration-exploitation tradeoff, and more. We propose a specific approach for reusing information collected in previous replanning episodes by intelligently selecting which agents should replan. Orthogonality, one may use incremental search methods such as D^*-lite [30] and *LPA** [31] for the individual planning. We leave the integration of these ideas to future research.

Acknowledgements

This paper is partially supported by the ISF fund, grants #964/22 and #1238/23, by the Helmsley Charitable Trust through the ABC fund, and by the KAMIN program case no. 81085.

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[2] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *IJCAI*, 2015.

[3] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *SoCS*, pp. 151–158, 2019.

[4] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *AAAI*, pp. 173–178, 2010.

[5] D. Silver, "Cooperative pathfinding," in *AIIDE*, pp. 117–122, 2005.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[7] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *AAAI*, pp. 11272–11281, 2021.

[8] J. T. Thayer and W. Ruml, "Bounded suboptimal search: A direct approach using inadmissible estimates," in *IJCAI*, vol. 2011, pp. 674–679, 2011.

[9] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *ICRA*, pp. 5628–5635, 2011.

[10] S. Chan, R. Stern, A. Felner, and S. Koenig, "Greedy priority-based search for suboptimal multi-agent path finding," in *SoCS*, pp. 11–19, 2023.

[11] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *ECAI*, pp. 961–962, 2014.

[12] S.-H. Chan, J. Li, G. Gange, D. Harabor, P. J. Stuckey, and S. Koenig, "Flex distribution for bounded-suboptimal multi-agent path finding," in *AAAI*, pp. 9313–9322, 2022.

[13] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based pomdp solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, pp. 1–51, 2013.

[14] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[15] R. Brafman and G. Shani, "A multi-path compilation approach to contingent planning," in *AAAI*, pp. 1868–1874, 2012.

[16] C. Amato, G. Chowdhary, A. Geramifard, N. K. Üre, and M. J. Kochenderfer, "Decentralized control of partially observable markov decision processes," in *IEEE (CDC)*, pp. 2398–2405, 2013.

[17] S. Shekhar, R. I. Brafman, and G. Shani, "A factored approach to deterministic contingent multi-agent planning," in *ICAPS*, pp. 419–427, 2019.

[18] T. Shahar, S. Shekhar, D. Atzmon, A. Saffidine, B. Juba, and R. Stern, "Safe multi-agent pathfinding with time uncertainty," *JAIR*, vol. 70, pp. 923–954, 2021.

[19] G. Wagner and H. Choset, "Path planning for multiple agents under uncertainty," in *ICAPS*, vol. 27, pp. 577–585, 2017.

[20] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *ICAPS*, vol. 30, pp. 29–37, 2020.

[21] J. Švancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták, "Online multi-agent pathfinding," in *AAAI*, pp. 7732–7739, 2019.

[22] A. Queffelec, O. Sankur, and F. Schwarzentruber, "Complexity of planning for connected agents in a partially known environment," *Theoretical Computer Science*, vol. 941, pp. 202–220, 2023.

[23] E. Levy, G. Shani, and R. Stern, "An online approach for multi-agent path finding under movement uncertainty," in *SoCS*, pp. 299–301, 2022.

[24] B. Shofer, G. Shani, and R. Stern, "Multi agent path finding under obstacle uncertainty," in *ICAPS*, pp. 402–410, 2023.

[25] D. Shiri and F. S. Salman, "On the online multi-agent o–d k-canadian traveler problem," *Journal of Combinatorial Optimization*, vol. 34, pp. 453–461, 2017.

[26] J. Kshirsagar, S. Shue, and J. M. Conrad, "A survey of implementation of multi-robot simultaneous localization and mapping," in *SoutheastCon*, pp. 1–7, IEEE, 2018.

[27] M. A. Abdulgalil, M. M. Nasr, M. H. Elalfy, A. Khamis, and F. Karray, "Multi-robot slam: An overview and quantitative evaluation of mrgs ros framework for mr-slam," in *Robot Intelligence Technology and Applications 5*, pp. 165–183, 2019.

[28] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE (T-ASE)*, vol. 12, no. 3, pp. 835–849, 2015.

[29] J. Li, Z. Chen, D. Harabor, P. Stuckey, and S. Koenig, "Anytime multi-agent path finding via large neighborhood search," in *IJCAI*, 2021.

[30] S. Koenig and M. Likhachev, "D* lite," in *AAAI-02*, pp. 476–483, 2002.

[31] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.