IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Risk-Aware Routing for a Robot in a Shared Dynamic Environment

Elena Stracca, Giorgio Grioli, Lucia Pallottino, and Paolo Salaris

***Abstract*—This paper explores the challenge of optimal routing for a mobile robot navigating a dynamic and shared human environment. The primary goal is to minimize the risk of performance degradation during motion, such as delays in completing tasks due to the need for safe or acceptable human-robot encounters. The problem is formulated as a graph whose edge costs become progressively known only as the robot moves through the environment. We model this problem as a Markov Decision Process (MDP), enabling an offline evaluation of the expected cost of alternative routes based on statistical information about human spatial distributions and possible observations at each intersection. This compact state representation scales linearly with the number of intersections in the map. Since the memoryless property of the MDP may induce loops during online execution, we compute an offline policy and introduce an online policy adaptation mechanism to prevent cyclic behaviors. Extensive simulations across environments of different complexity, and using data collected from real-world experiments, demonstrate that our approach outperforms reactive and advanced state-of-the-art planners in terms of either performance or scalability.**

***Index Terms*—risk-aware routing, robot navigation, reactive path planning, stochastic shortest path with recourse**

I. Introduction

The employment of an Autonomous Mobile Robot (AMR) in dynamic environments shared with humans, such as logistic warehouses, hospitals, and industries, has considerably increased in recent years due to their flexibility and ability to increase productivity, while limiting related costs [1], [2].

Indoor environments typically have well-defined layouts and maps that allow for efficient path planning, routing, and navigation. However, a route or path that traverses dynamic, shared, and constrained spaces (e.g., corridors in a warehouse) can quickly degrade the robot's performance (e.g., delaying the arrival at the final destination) when these spaces are occupied by people or temporary obstructions whose location is often uncertain. On the other hand, AMRs are typically equipped with onboard sensors and detection modules that gather real-time data on congestion levels as the robot approaches critical decision points where multiple paths or routes to the final destination are available. This allows the robot to adjust the path/route online if less risky alternatives are available (Fig. 1). In this paper, we explore how to leverage the robot's ability to perform local observations, in conjunction with some prior knowledge, to anticipate situations that could potentially degrade the robot's overall performance (excluding safety concerns) during a routing problem.

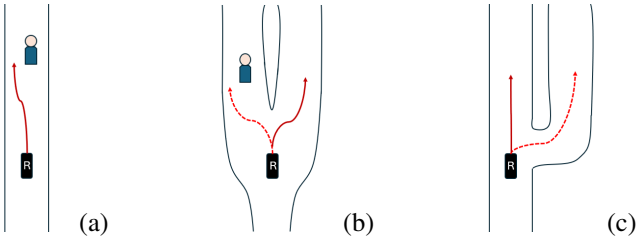


Fig. 1. A robot (the black rectangle) navigating different scenarios where humans may be present: **(a)**: In a long corridor, the robot must locally re-plan its path to avoid the human while staying in the same corridor. **(b)**: If a human is detected before an intersection, it may be advantageous to choose an alternate route. **(c)**: When two routes are available, selecting the one with higher visibility from the intersection point may be beneficial, as it reduces the likelihood of encountering someone if no one is currently detected.

In the context of risk management, risk is defined as an uncertain event that, if it occurs, negatively affects the success of a task. Risk is evaluated based on the combination of probability and severity [3]. Risks in robotics are typically associated with the hazard the robot poses to humans working in the same workspace. To move alongside humans, AMRs need to comply with safety norms [4]. This implies locally adjusting their trajectory near people, reducing their velocity, or changing route, resulting in some performance degradation, especially in confined spaces. Furthermore, collaborative robots should also ensure that the interaction is perceived as safe by the human during the encounter [5].

Typical risks may include delays in production schedules, increased stress for human operators, or even collisions. As a consequence, minimizing encounter risks is crucial for ensuring safe and efficient operations. Consequently, when an AMR enters a setting where humans may be present, it is prudent to recognize the potential risks associated with potential encounters throughout the various phases of motion planning.

Risk-aware motion planning aims to provide feasible and safe paths while trying to maintain high performance. In motion planning literature, risk is usually intended only as the probability of collision, without assessing the severity of the potential collision or other causes of task failure.

The problem of planning a reference path for AMRs in an

This work has received funding from the European Union's Horizon 2020 research and innovation program under agreement no. 101017274 (DARKO) and the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab FoReLab projects (Departments of Excellence). We acknowledge the support of the European Union by the Next Generation EU project ECS00000017 'Ecosistema dell'Innovazione' Tuscany Health Ecosystem (THE, PNRR, Spoke 4: Spoke 9: Robotics and Automation for Health).

The authors are with the Research Center E. Piaggio and Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Largo Lucio Lazzarino 1, Pisa, Italy , elena.stracca@phd.unipi.it, {lucia.pallottino, giorgio.grioli, paolo.salaris}@unipi.it

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

uncertain scenario usually starts by defining metrics whose expected cost should be minimized (e.g., length, distance to obstacles [6]), and the problem of changing the path if the cost changes is addressed with a real-time re-planning [7], [8]. In an asymmetrical environment, the path a robot selects to reach its objective will determine the cost of re-planning the path when the initial path choice leads the robot into an obstructed area due to an obstacle or a person. The single path associated with the minimum expected cost, however, is not the one that will necessarily lead to the minimum cost if the robot is able to update the selected path online once new sensory information is acquired. This problem is similar to the *Stochastic Canadian Traveler Problem (CTP)* [9]. In CTP, the routing problem is modeled as a graph, where some edges may be non-traversable with a certain probability. In this formulation, edge traversability becomes known when the agent reaches the starting node of the uncertain edge. Our formulation is inspired by CTP and, in particular, by its variant called *Stochastic Shortest Path with Recourse (SSPR)* [10]. In SSPR, new information on edge costs is acquired at runtime, but all the edges remain traversable even if at a higher cost.

Summary of contributions:
In this work, we integrate the risk inherent in human-robot interactions into the routing problem at a higher level of the motion planning pipeline compared to the state-of-the-art. We minimize the expected cost as a risk metric. To model the potential observations the robot might perceive from the intersections of homotopy classes, we employ an MDP formulation. These observations incorporated into the model will modify the prior probability of encountering a person in that corridor. Our objective is to address this problem using a computationally efficient and straightforward approach. The resulting policy will enable the robot to transition between homotopy classes, considering human position probability and potential encounter severity. This approach minimizes the risk associated with possible human-robot encounters.

The main innovative contributions of this work are as follows:

- We introduce in the motion planning pipeline a risk-aware decision-making module. By incorporating possible observations made at intersection points as part of the state, our method allows, for the first time, to compute a policy that explicitly accounts for re-planning possibilities, obstacles visibility, alternative route costs, and the severity of potential encounters.
- We demonstrate that the proposed decision-making module generates paths with superior performance compared to those generated by state-of-the-art path-planning algorithms that minimize the expected cost of the same metric without effectively incorporating observations into the model.
- We propose a formulation with low computational complexity, enabling its use in large and complex environments.
- We formalize an approach that enables a straightforward and rapid resolution, as well as an effortless online adaptation of the offline planned policy, upon the detection of humans.

The rest of the paper is structured as follows: Section II presents the state-of-the-art in risk-aware motion planning, stochastic shortest path with recourse, and Markov decision processes. Section III defines the problem and discusses the challenges in solving it. Section IV describes our approach to modeling the problem using an MDP. Section V applies our problem formulation and solves it explicitly for a dynamic shared warehouse scenario. Section VI tests the method on three maps of varying complexity, comparing our strategy with paths generated by several state-of-the-art algorithms. Finally, Section VII discusses the model's robustness.

II. Related Works

Risk-aware planners usually consider the current uncertainty in estimating possible human movement [11], dynamic obstacles position [12], or the current information on a hazardous source [13] to compute risk according to a coherent risk metric. In [14], authors propose a variant planning horizon method to evaluate different generated trajectories at each iteration on a metric depending on the estimated uncertainty of localization, the probability of collision, and the distance to the goal.

At the global planning level, only a limited number of works explicitly acknowledge risk in the path selection process. In [15], the authors present a risk-aware path planner, a modified A* algorithm, that utilizes a risk map to compute a safe path by balancing the path length and the risk cost. A risk-aware Markov Decision Process (MDP) was introduced in [16] to model the expected risk due to ocean currents probabilistically in an autonomous underwater vehicle scenario.

The problem of planning a minimum-cost strategy in a stochastic network where new information is acquired at runtime is known as the Stochastic Canadian Traveler Problem (CTP) [9]. CTP has recently been applied to robot routing in both outdoor [17] and indoor [18]–[20] environments. In our problem, however, human presence would only degrade the performance (e.g., by increasing the traversal time) of a selected route, so our problem formulation would be more similar to a CTP variant called *stochastic shortest path with recourse (SSPR)* [10]. In this version, all edges are always traversable, but the cost is stochastic a priori and becomes known when the agent reaches one of the edge starting nodes. However, in a shared dynamic environment, while real-time observations provide useful information, the edge cost would remain stochastic. To the best of the authors' knowledge, the only work that assumes an SSPR-like problem formulation in robotics routing is [21]. Nevertheless, the authors decompose the original network into a series of acyclic paths and solve at each iteration for the least risky path according to a metric that takes into account the expected cost and variance. This approach yields an approximate solution that is more similar to the tree search techniques employed for stochastic CTP classical problems. They also consider each edge cost to be independent and expressed by a normal distribution with different variances and expected costs. In contrast, this work focuses on performance degradation due to discrete

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

events (e.g., encountering a person), so we will have discrete probabilities associated with possible detrimental events and associated penalties. Furthermore, we will not decompose the network into a set of acyclic paths but propose a strategy to find a solution that accounts for the complete graph's structure.

SSPR formulations have been extensively applied to transportation networks [22], [23]. In this scenario, however, the problem formulation is different as edge costs vary over time. Nevertheless, real-time traffic information is consistently accessible for a specific set of edges within the network (not merely those adjacent to the current edge).

The stochastic nature of an environment concerning the presence of humans or other dynamic obstacles requires the robot to plan with a heightened awareness of uncertainty. In this regard, a popular way to address uncertainty in motion planning are Markov Decision Processes (MDPs). MDPs are usually used to acknowledge uncertainty in the obstacle position and the actuation control or to model and predict human movements [24]. The environment is divided into cells associated with a different occupancy probability, and the objective is to find an optimal policy to move across the grid map to accomplish the desired task. A constrained MDP was proposed in [25] to find paths in a warehouse. The objective of the MDP was to minimize a risk measure depending on the distance from static obstacles or the length of the path, imposing a constraint on the other objective.

In this work, we are interested in developing a higher-level routing problem by studying the set of routes that minimizes the risk originating from the possibility of re-planning online. For grouping similar paths we use the concept of homotopy classes, i.e., the union of all paths that connect two points such that the first path can be continuously transformed into the second in the free space (there are no obstacles between) [26]. The importance of keeping track of multiple path solutions belonging to different homotopy classes in dynamic environments is recognized in the literature, since affine paths are likely to degrade together [27]. Different methods have been proposed to detect homotopy classes in real-time and to quickly find a set of diverse paths to avoid local minima in online trajectory optimization methods (see e.g., [27], [28] and papers therein). An algorithm to switch between different homotopy classes in online motion planning, continuously evaluating the dynamic cost of paths in concurrent homotopy classes, was presented in [29]. They focus on the online modification of a dynamic graph of homotopy classes that can adapt to changes in the environment. However, they provide a reactive strategy to avoid humans locally, and the cost of possible alternative routes after making a first choice is never considered.

III. Problem Statement

Consider a robot that navigates through an indoor environment populated by obstacles that are possibly moving. Assuming that the environment map is static, we describe it in the form of a graph

$$\mathcal{G} = (V, E)\,, \tag{1}$$

that is a pair of two sets: the set V of nodes v, that represents *decision points* in the environment, and the set E of oriented edges $e = (v_i, v_j)$ connecting pairs of nodes from V, representing admissible *routes* from v_i to v_j. Here, the term *decision points* includes rooms and intersections, as well as the starting position of the robot - marked by node S, and the goal position of the robot - marked by node G. Whereas with *routes* we refer to corridors, doors, and in general passages connecting one decision point to another. More formally, each *route* represents the *homotopy class* of all the trajectories connecting two *decision points*, i.e., the union of all the possible *homotopic* paths connecting two points A and B. Here, given a continuous path connecting two points A and B, a second path is said to be homotopic to the first one if a continuous map exists that can transform the first path into the second without intersecting any obstacle (see also [26]).

Within this framework, a *path*

$$\mathcal{P} = (e_1, e_2, ..., e_n) \tag{2}$$

of connected[1] edges, describes a static high-level plan to navigate the robot from the first node of e_1 to the last node of e_n. We assume that the lower-level controller of the robot can always generate a feasible trajectory for a given *route* - and therefore for a given path - by searching in the space of its *homotopy class*. Notice that we assume this to be always true, even in the presence of obstacles, possibly at the price of the robot adopting a more costly trajectory (e.g. by stopping to let a moving obstacle walk by, or by navigating around a fixed obstacle). A consequence of this assumption is that the robot will never fail to navigate along a commanded *route*, i.e. it will never revert to the starting node of an edge once it starts walking along it.

All of that translates into the fact that each edge e_i has a stochastic navigation cost

$$C_i \in \{c_1, c_2, \ldots, c_{n_i}\} \quad \text{with} \quad \mathbb{P}(C_i = c_j) = p_j,$$

which is a discrete stochastic variable with possible outcomes c_j, each with probability p_j, reflecting the uncertainty in the environment, such as varying levels of congestion or the likelihood of an obstacle appearing. Within this framework, any given path $\mathcal{P}$ can be characterized by an expected cost

$$C(\mathcal{P}) \triangleq \mathbb{E}\left[\sum_j c_j\right] \text{ with } j | e_j \in \mathcal{P}\,. \tag{3}$$

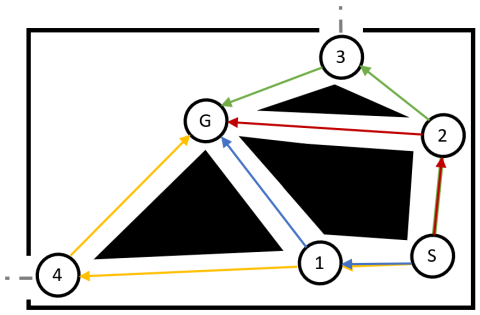
A. Stochastic shortest path problem

Consider now the problem of planning the best strategy for a robot to move from S to G. Assume, for simplicity, that the probability distributions of the costs of each edge are independent. Notably, the solution can be found with one of many algorithms from the literature [30] and is given by a static path *path* $\mathcal{P}^*$ from S to G, characterized by the minimum possible expected cost

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} C(\mathcal{P}). \tag{4}$$

[1]Each consecutive pair of edges e_i and e_{i+1} is connected by sharing a common vertex. Specifically, if $e_i = (u, v)$ and $e_{i+1} = (v, w)$, then the endpoint of e_i (vertex v) is the starting point of e_{i+1}.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.



Edge	Cost
$(S,1)$	1.5
$(1,G)$	$\begin{cases} c_1 = 4,\ p_1 = 0.8 \\ c_2 = 40,\ p_2 = 0.2 \end{cases}$
$(1,4)$	$\begin{cases} c_1 = 6,\ p_1 = 0.7 \\ c_2 = 40,\ p_2 = 0.3 \end{cases}$
$(4,G)$	$\begin{cases} c_1 = 6,\ p_1 = 0.7 \\ c_2 = 40,\ p_2 = 0.3 \end{cases}$

Edge	Cost
$(S,2)$	2
$(2,G)$	$\begin{cases} c_1 = 5,\ p_1 = 0.75 \\ c_2 = 40,\ p_2 = 0.25 \end{cases}$
$(2,3)$	$\begin{cases} c_1 = 3,\ p_1 = 0.85 \\ c_2 = 40,\ p_2 = 0.15 \end{cases}$
$(3,G)$	$\begin{cases} c_1 = 3,\ p_1 = 0.85 \\ c_2 = 40,\ p_2 = 0.15 \end{cases}$

Fig. 2. Stochastic graph where the edge cost becomes known in the node at the tail of the arc. From node S to node G there are four different paths: the blue, the red, the green, and the yellow.

Consider, for example, the simple case in Fig. 2, where the graph represents the homotopy classes of a room with three obstacles, a start, a goal, and two doors, that get translated into five *decision points* and eight *edges*.

Starting from node S, four alternative paths exist to reach node G: the blue, the red, the green, and the yellow. By computing the expected cost ($\mathbb{E}[C]$) of the four paths we get

$$\mathbb{E}[C_{blue}] = c(S,1) + \mathbb{E}[C_{(1,G)}] = 12.7,$$
$$\mathbb{E}[C_{red}] = c(S,2) + \mathbb{E}[C_{(2,G)}] = 15.75,$$
$$\mathbb{E}[C_{green}] = c(S,2) + \mathbb{E}[C_{(2,3)}] + \mathbb{E}[C_{(3,G)}] = 19.1\,,$$
$$\mathbb{E}[C_{yellow}] = c(S,1) + \mathbb{E}[C_{(1,4)}] + \mathbb{E}[C_{(4,G)}] = 33.9\,.$$

And therefore the minimum cost is 12.7, which is obtained, on average, by the blue path.

B. Stochastic shortest path with recourse (SSPR)

Assume that when the robot approaches a *decision point*, its sensors provide some information y about the congestion of the adjacent *routes*. Consequently, it is reasonable that the probabilities p_j associated with the cost C_i assuming values c_j may change to some new values

$$\mathbb{P}(C_i = c_j | y) = \hat{p}_j\,. \tag{5}$$

As new information is acquired while moving the optimal *path* $\mathcal{P}^*$ won't be static, but would adapt online according to the performed observations. In this scenario, it is more appropriate to talk about policies. In general, a policy Π is a function

$$\Pi = \Pi(v) : V \to E\,, \tag{6}$$

that given the current configuration of the robot, represented by a node v, yields the next action, here represented by the edge e.

The solution of (4) can be extended to a static policy Π_s that assigns the action corresponding to the minimum prior expected cost at each node. However, this policy is, of course, not optimal as it doesn't take into account the new information acquired online.

In this case, a common approach [7] is to start to follow the optimal path, and then re-calculate a new static path $\mathcal{P}^*_y$, which is optimal based on the new information y. This generates a naive dynamic policy Π_d. However, such a policy is not always optimal. Consider again our toy example in Fig. 2, where each edge cost becomes known when the robot reaches its outgoing node. The optimal path is the blue one, so the robot would take edge $(S,1)$. It could happen that after the robot has executed the first step of the originally optimal path $\mathcal{P}^*$, it observes that an obstacle is present in the edge $(1,G)$, thus updating the graph costs, and, if edge $(1,4)$ is free (so its value is 6), it produces a novel policy Π^*_y for which the next optimal step is $(1,4)$. Such a strategy would lead to an expected cost

$$\mathbb{E}[C]_{\Pi_d} = c(S,1) + p_1(1,G)c_1(1,G) + p_2(1,G)[p_2(1,4)\cdot \\ \cdot c_2(1,G) + p_1(1,4)(c_1(1,4) + \mathbb{E}[C_{(4,G)}])] = 10.21.$$

However, the overall optimal dynamic policy (Π^*) for the toy example is as follows: the robot takes edge (S,2). Here, if the edge (2,G) has cost 5 or if both edges (2,G) and (2,3) have cost 40, the robot takes edge (2,G). In the other case, it takes edge (2,3). Indeed, computing the expected cost of this policy yields

$$\mathbb{E}[C]_{\Pi^*} = c(S,2) + p_1(2,G)c_1(2,G) + p_2(2,G)[p_2(2,3)\cdot \\ \cdot c_2(2,G) + p_1(2,3)(c_1(2,3) + \mathbb{E}[C_{(3,G)}])] = 9.70\,.$$

In general, finding a policy that is optimal under those assumptions is not a trivial task. Indeed, whenever the graph $\mathcal{G}$ contains cycles, which happens very often in practical cases (see e.g. Fig. 4), solving SSPR can be NP-Hard [10].

Provan demonstrates in [31] that solving SSPR for networks with cycles can be done with polynomial complexity only if the stochastic graph has the property that each time the agent returns to the same node, the cost of its outgoing star arcs is independent of previous realizations (otherwise the problem is, at best, NP-complete). Provan calls this the *reset* property. However, assuming *reset* for networks with cycles can lead to policies that get stuck in loops when the real use-case problem doesn't have such a property.

An example of such a problem is reported in Fig. 3. There, the only way to reach G from node 1 is to take the edge $(1,G)$, whereas, by assuming the reset property, it seems less costly to loop through $(1,3),(3,2),(2,0),(0,1)$ and back to node 1 "in hope" of a better observation. We will give a more formal proof of when a cycle can occur in our specific problem in section IV-A. This clearly leads to policies that are not useful in practice, e.g., in the case of static obstacles whose position is stochastically defined.

Therefore, our goal is to make such solutions usable also in those cases, by proposing a suitable mitigating strategy.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

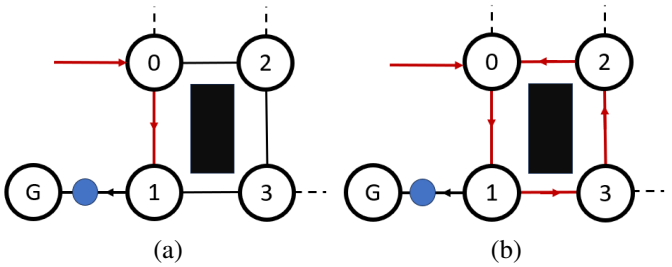


Fig. 3. Robot navigation on a graph with nodes $V = \{0, 1, 2, ..., G\}$ with uncertain edge costs. The robot movements are represented by red arrows, the blue dot is a static person on the edge $(1, G)$: (**a**) The robot arrives in 1 and observes the obstacle presence in $(1, G)$, and that $(1, 3)$ is free. If the expected cost of a strategy that from 1 executes $\{1 \to 3 \to 2 \to 0 \to 1\}$ to re-observe $(1, G)$ cost, is minor to the cost of crossing it with the obstacle, the theoretical optimal strategy (without considering the possibility of going back to the previous node) is to do the loop. (**b**) If the obstacle is static, however, this is not true, and repeating that loop creates an infinite cycle, since returning to node 1 yields no new information.

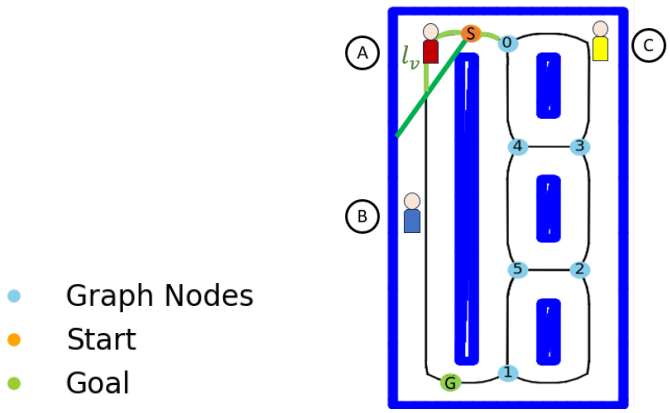


Fig. 4. Consider a scenario where 3 people $\{A, B, C\}$ are present in the robot environment. Our assumptions imply that if the robot is at node S: person A is observable, person B is not observable, and person C's presence is not considered in node S as they are not on an edge of the outgoing star of node S. Person C becomes observable from nodes $\{0, 3\}$.

Moreover, in the classical SSPR problem formulation, as in our toy example, the edge cost would become known when the agent reaches its tail $\mathbb{P}(C_i = c_j|y) = 1$. Instead, in a shared dynamic environment, this is not true as people may enter while the robot is already moving in the corridor, or might not be visible from the decision point. We will therefore need to compute explicitly $\mathbb{P}(C_i = c_j|y) = \hat{p}_j$.

C. Cost Metric

The cost metric we will consider in the following to evaluate the path resulting from a policy in a certain environment configuration is the path length, l_{path}, plus a severity component that estimates the performance degradation that happened along that path.

A risk is an uncertain event that can degrade the success of the task/project. In our problem, we can define the task success as a *safe and fast execution of the path that doesn't cause hazards for the other agents (e.g., human operators) or delays that can decrease productivity*. Because safety and collision avoidance always take priority over speed, the robot may need to slow down or stop when encountering an obstruction (e.g., a person) in a corridor. So, we can define a discrete penalty to be added to the path cost each time an obstruction is encountered. This penalty would be space-dependent, to capture how easy it is in that location to replan to avoid the partial obstruction. For simplicity, we consider it to be edge-dependent and denote it as $c_p[edge]$.

Therefore, the cost metric will be:

$$C(\mathcal{P}) = l_{path} + \sum_{h=0}^{k} (c_{p,h}[edge_h]) \tag{7}$$

where k is the total number of obstructions encountered while reaching the goal.

IV. Problem Formulation as an MDP

Without loss of generality, we consider in the following that partial obstructions are due to an unknown number $n \in \mathbb{N} \cap [0, N_p]$ of people in the environment. A complete representation of the problem would require including the possible number of people in each corridor in the state space and solving the problem as a Partially Observable Markov Decision Process (POMDP), where the robot position is known and the human positions are partially observable. However, this method would quickly become intractable. Even without discussing the computational complexities in solving a POMDP [32], consider an environment with 50 corridors. Assuming 20 people, the number of states (just considering the possible human distributions in the corridors) would be: $n_h = \binom{69}{49}$, giving a number of states of the order of 10^{17}.

Therefore, our idea is not to consider the number of people in each corridor as part of the state, but, instead, we propose to include in it what would have been the observations of the POMDP, namely, the possible observations the robot can make from each intersection point in the map. In this way, we obtain an MDP with a number of states that scales only linearly by increasing the number of intersection points and that does not depend on the number or possible distribution of people in the environment.

A Markov Decision Process (MDP) is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$ is a finite set of states and $\mathcal{A}$ is a finite set of actions. The transition model is encoded in a 3-dimensional tensor $P \in [0, 1]^{|\mathcal{A}|\times|\mathcal{S}|\times|\mathcal{S}|}$, where each element $P[a, s, s']$ represents the probability of transitioning from state s to state s' when action a is taken. The cost (or reward) structure is defined by a matrix $R \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, where each entry $R[s, a]$ denotes the expected cost incurred when executing action a from state s.

For simplicity, we consider that when the robot is in a node $v \in V$ it makes observations only in the adjacent corridors $(e \in E \mid e = (v, u)$ for some $u \in V)$ (Fig. 4). Suppose a person is in $l_v(v, e)$, the visible part of the corridor that the robot is observing from node v (there are no obstacles between the human position and that of the robot). In that case, it can detect the person correctly (Fig. 4). Moreover, to avoid a loop as in Fig. 3 can be generated by simply moving back and forward on the same edge, we impose that the robot can

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

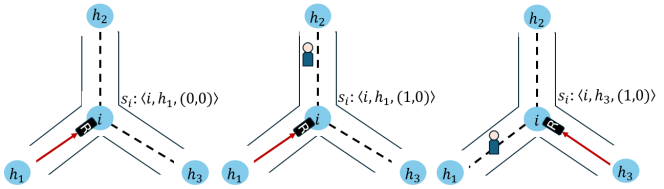


Fig. 5. Examples of different states the system can be when the robot is in node i.

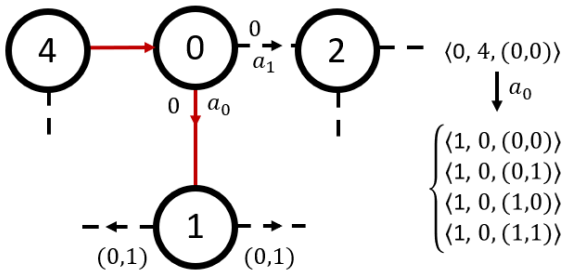


Fig. 6. Possible next states by taking action a_0 from state $\langle 0, 4, (0, 0)\rangle$.

change route at intersection points, but never turn back where it came from.

1) States: The state includes the intersection point at which the robot is making the decision and the incoming node from which it comes. To achieve a decision planner that takes into account the possibilities and quality of replanning routes, we then propose to add as part of the state a tuple representing the possible observations the robot can make in the current node. Observation "1" means that the robot detects at least one person in the observed edge of the homotopies graph, and observation "0" means that no people are seen in the corridor.

The generic state of node i, with predecessor i_p, and observations $(i_o^1, i_o^2...)$ can be formally expressed as:

$$\mathcal{S}_i = \langle i, i_p, (i_o^1, i_o^2...)\rangle \tag{8}$$

For example, the states associated with node i having three incoming nodes $h_i, i = 1, 2, 3$ are:

$$\begin{aligned} i : \{&\langle i, h_1, (0,0)\rangle, \langle i, h_1, (0,1)\rangle, \langle i, h_1, (1,0)\rangle, \langle i, h_1, (1,1)\rangle, \\ &\langle i, h_2, (0,0)\rangle, \langle i, h_2, (0,1)\rangle, \langle i, h_2, (1,0)\rangle, \langle i, h_2, (1,1)\rangle, \\ &\langle i, h_3, (0,0)\rangle, \langle i, h_3, (0,1)\rangle, \langle i, h_3, (1,0)\rangle, \langle i, h_3, (1,1)\rangle\}. \end{aligned}$$

Where i is the current node, h_1, h_2, h_3 are the possible predecessors of node i, the combination of 0 and 1 is the possible observations the robot can make on the two remaining edges (not the predecessor). The elements of the observation vector are ordered by the indices of the neighboring nodes, starting from the smallest to the largest. Some examples are reported in Fig. 5.

The number of states will be for each node $u_i 2^{(u_i-1)}$, where u_i is the number of outgoing edges from the node. This applies to all nodes except for the start and goal. For the goal, we will only have one state, in which the process remains with probability 1 after the system reaches the goal. As the start and goal are added by identifying the edge on the homotopy classes graph on which they lie, they are each always connected to exactly two other nodes. For the start node, we will not have a predecessor node, so we will only have the four states $s : \{\langle s, (0,0)\rangle, \langle s, (0,1)\rangle, \langle s, (1,0)\rangle, \langle s, (1,1)\rangle\}$ associated with the two edges to which the start node is connected. The total number of states will be:

$$n_s = \sum_{i=1}^{N} (u_i 2^{u_i-1}) + 5, \tag{9}$$

where N is the number of nodes of the homotopy graph, excluding the start and goal. In this way, the number of states increases only linearly when augmenting the number of nodes (when the map complexity increases). Moreover, it does not depend on the number of people in the environment.

2) Actions: We consider as actions the choice to go through the different edges belonging to the outgoing star from each node (except for the one that returns to the predecessor node). The actions will not be the same for each state, a_0 would be the action that sends the robot to the minimum index node connected to the current node, a_1 to the second minimum index adjacent node, and so on. For example, in all the states of node "0" with predecessor node "4" in Fig. 6 we will have two actions:

$$0 : \{a_0 : \text{"To node 1"}, a_1 : \text{"To node 2"}\}.$$

The total number of actions will be $n_a = \max_{i \in V}\{u_i - 1\}$.

Fig. 6 represents which states may the robot reach by choosing action a_0 from state $\langle 0, 4, (0, 0)\rangle$. Once the action is chosen, the next node and the previous node are deterministically known; the uncertainty is in the possible observations the robot will make from the next node.

3) Transition probabilites: The P tensor encodes the probability of making a tuple of observations at the next node, starting from a node with known tuples of observations. We represent the generic element of the P tensor as: $P[a_z, \langle i, i_p, (i_o^1, i_o^2...)\rangle, \langle j, j_p, (j_o^1, j_o^2...)\rangle]$, where a_z is the considered action, $\langle i, i_p, (i_o^1, i_o^2...)\rangle$ is the state corresponding to the current node i, $\langle j, j_p, (j_o^1, j_o^2...)\rangle$ is the arriving state we may reach by taking action a_z.

The P computation algorithm is reported in algorithm 1. When we select an action, the next graph node is deterministic, so $P[a_z, \langle i, i_p, (i_o^1, i_o^2, \ldots)\rangle, \langle j, j_p, (j_o^1, j_o^2, \ldots)\rangle]$ would be 0 if $j \neq i(a_z)$ or $j_p \neq i$ (line 6). When $j = i(a_z)$ and $j_p = i$, $P[a_z, \langle i, i_p, (i_o^1, i_o^2, \ldots)\rangle, \langle j, i, (j_o^1, j_o^2, \ldots)\rangle] = \prod_{e=0}^{u_j} p_o(obs = j_o^e)$ (line 8), where e are all the u_j edges from the node j, p_o is the probability of making a certain observation, and obs $= j_o^e$ is the observation (0 or 1) that the robot will make from node j in that edge. By doing so, we are approximating observations on different edges as independent.

For every edge from j, the probability of making an observation equal to 1 is the probability that at least one person is in the visible part of the corridor looking from node j. For every possible number of people in the environment n, the probability that $obs = 0$ is the probability that all the n people are out of the visible region of the selected corridor.

Once the goal is reached, the process remains in the goal state with a probability of 1, so the probability to reach any other node is 0. (lines 10-11).

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Algorithm 1: P Computation

Data: actions, states, $p_o(obs)[node][edge]$
Result: $P(a \times s \times s)$

for $a_z \leftarrow 0$ **to** $actions$ **do**
  for i *in graph nodes* **do**
    if $i \neq goal$ **then**
      for j *in graph nodes* **do**
        if $j \neq i(a_z)$ ***or*** $j_p \neq i$ **then**
          $P[a_z, \langle i, i_p, (i_o^1, i_o^2, ..)\rangle, \langle j, j_p, (j_o^1, j_o^2, ..)\rangle] = 0$;
        else
          $P[a_z, \langle i, i_p, (i_o^1, i_o^2, ..)\rangle, \langle j, j_p, (j_o^1, j_o^2, ..)\rangle] = \prod_{e=0}^{u_j} p_o(obs = j_o^e)$;
    else
      $P[a_z, \langle i\rangle, \langle i\rangle] = 1$;
      $P[a_z, \langle i\rangle, \langle j, j_p, (j_o^1, j_o^2, \ldots)\rangle] = 0$;

4) Rewards matrix: The rewards matrix (R) with dimensions $(s \times a)$ will have the following form:

$$R = \begin{bmatrix} -c_{S_0,a_0} & -c_{S_0,a_1} & \dots & -c_{S_0,a_n} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{S_n,a_0} & -c_{S_n,a_1} & \dots & -c_{S_n,a_n} \\ 0 & 0 & \dots & 0 \end{bmatrix}, \tag{10}$$

where c_{S_i,a_j} represents the expected cost of the edge that we will choose from the node corresponding to state S_i if action a_j is selected. The last line, the one associated with the goal state, is 0 because it corresponds to the fact that, once the goal is reached, the system stays in a fictitious state with cost 0.

The edge cost if k humans are on it, $c_{k,e}$, according to eq. 7 is:

$$c_{k,e} = l_e + k\, c_p[edge]. \tag{11}$$

The expected edge costs that we need to compute as elements of the R matrix, will be:

$$c = l_e + \sum_{k=0}^{N_p} p(k|obs)\, k\, c_p[edge], \tag{12}$$

where l_e is the length of the path connecting the two nodes, $p(k|obs)$ is the probability that there are k people in the corridor given the observation obs made in the current node, $c_p[edge]$ is the severity coefficient for that edge (see section V-A).

In our scenario, given a current state and an action the edge to be traversed is deterministically identified, so the R matrix entries will not depend on the arriving state.

We show explicitly how matrices P and R can be computed in section V-B. Once the P and R matrices are computed, the *MDP* optimal policy can be found by solving the Bellman equation with classical methods, for example using the Value Iteration algorithm. The Bellman equation provides a recursive decomposition of the value function, which the Value Iteration algorithm uses to update the value of each state until convergence iteratively. We use $\gamma = 0,9999$ as the discount factor to ensure the long-term rewards are well considered.

A. Modifying the Optimal Policy Online

This formulation has undoubted advantages in terms of the tractability of the problem; however, it has some limitations. An MDP, by its inherent property, does not take past history into account (like the *reset* property by Provan).

This means that the probability of making a certain observation on an edge will not depend on the observations already made (even on the same edge). This implies that, if the severity of passing in an occupied edge is higher than the expected cost of returning to the same node by a different path, and here making a different observation, the *MDP* can enter a theoretical infinite loop during the path execution. The loop originates if a static person continues to occupy that homotopy class (Fig. 3). In the following, we will refer to these concepts:

- Spatial directed loop: a path in $G = (V, E)$ that starts and ends at the same *position* (i, i_p), that is, the same current node $i \in V$ and the same predecessor $i_p \in V$, possibly with a different observation vector. [2]
- State cycle: a spatial directed loop that returns to the identical decision state $s_i = \langle i, i_p, \mathbf{i}_o\rangle$ (node, predecessor, and observation vector unchanged).
- First-choice edge: for each node i, let $e^\star(i)$ be the edge that would be selected under optimal conditions, i.e. when all observations on edges incident to i are 0.

Let $c_e(0)$ and $c_e(1)$ be the traversal costs of an edge e under observations 0 and 1, respectively, and denote $\Delta c^\star = c_{e^\star}(1) - c_{e^\star}(0) > 0$ (the *over-cost due to bad observation* on the first-choice edge). In the following lemma, we will discuss the necessary conditions for a loop to be possible.

Lemma 1. *Consider the decision state $s_i = \langle i, i_p, \mathbf{i}_o\rangle$ at position (i, i_p).*
A spatial directed loop returning to (i, i_p) can belong to an optimal policy $\pi^\star$ only if all *the following conditions hold:*

i) The current observation on the first-choice edge is $i_o^\star = 1$;
ii) There exists a spatial directed loop γ such that $p\big[\,\gamma$ sets $i_o^\star : 1 \to 0\big] > 0$;
iii) The expected cost of the loop satisfies $\mathbb{E}[C_\gamma] < \Delta c^\star$.

If any one of these conditions is violated, no spatial directed loop can lower the expected cost-to-go from s_i, and therefore no such loop appears in $\pi^\star$.

Proof: (*Necessity*). Suppose a spatial directed loop γ returning to (i, i_p) is part of an optimal policy.
(i) If $i_o^\star = 0$, executing γ incurs the strictly positive cost $\mathbb{E}[C_\gamma]$ without any chance of reducing future cost on $e^\star$; the total cost-to-go therefore increases, contradicting optimality. Hence $i_o^\star = 1$ is necessary.

[2] A standard *spatial loop* may occur when the robot returns to the same node i independent from the predecessor. We provide this definition as the actions available from node i, and the associated observations, are conditioned on the predecessor i_p. Since i has a finite number of predecessors, any infinite loop returning to the same node i must also contain an infinite loop returning to the same (i, i_p) pair, ensuring our subsequent results apply generally.

(ii) If γ cannot flip $i_o^\star$ from 1 to 0 with positive probability, then regardless of the loop cost, the observation remains 1 and the cost-to-go is unchanged; the loop again yields a net loss, contradicting optimality.
(iii) Finally, if $\mathbb{E}[C_\gamma] \geq \Delta c^\star$, the expected saving obtained by potentially turning $i_o^\star$ to 0 is no greater than the cost of γ, so the loop is not convenient.
Since conditions (i)–(iii) are all necessary, the absence of any one of them rules out the inclusion of γ in an optimal policy. ∎

Corollary 1 (Existence of state infinite cycles)**.** *An* infinite state cycle *(i.e., a policy fragment that returns to the* same decision state $s_i = \langle i, i_p, \mathbf{i}_o \rangle$ *infinitely often) can appear in the optimal policy* $\pi^\star$ *computed* offline *only when*

i) conditions (i)–(iii) *of Lemma 1 are satisfied; and*
ii) the agent responsible for the observation 1 $(i_o^\star = 1)$ *on the first-choice edge* $e^\star(i)$ *is* static*, so that* $p\big[i_o^\star : 1 \to 0\big] = 0$.

Proof: Lemma 1 states that a spatial directed loop may belong to $\pi^\star$ only if (i)–(iii) hold. If, in addition, the person occupying $e^\star(i)$ never moves, condition (ii) is violated in practice because the observation cannot transit from 1 to 0. The Markov-state abstraction, however, considers in the state only the observations of the current node, being thus unaware of this persistence; hence, the offline solver still considers the loop as potentially beneficial and executes it. The loop is therefore executed indefinitely at run time, yielding an infinite state cycle. ∎

So, when the *MDP* is solved the first time before starting, and the optimal policy is computed, this policy can theoretically result in infinite state (and therefore spatial) loops.
To prevent the undesired behaviour of Corollary 1 we adopt the following *online policy–modification* strategy.

Definition 1 (Online policy modification)**.** *Whenever the robot arrives in a node* i*, and acquires the observation* $i_o(e) = 1$ *in* $e = (i, j)$*, it immediately:*

a) sets the corresponding edge–observation probability to 1*, i.e.* $p(i_o(e) = 1) = 1$ *for both incident nodes* i *and* j*. These probabilities will be kept fixed at* 1 *until the robot reaches the goal* g*. This is equivalent to removing from the transitions all terms that would consent to arrive in* i *from a neighbor node with an observation* $i_o(e) = 0$*, and thus disallowing the observation on* e *to switch* $1 \to 0$*.*
b) Resolves the MDP with the updated P*, obtaining a new optimal policy* $\pi^\star_{new}$*.*

The resulting control law is the concatenation of the sequence of local optimal policies $\pi^\star$, $\pi^\star_{new}, \ldots$ *generated in this way.*

Theorem 1 (Loop elimination and convergence)**.** *The policy obtained through Definition 1 (i) contains no infinite spatial cycles and (ii) reaches the goal node* g *with probability* 1.

Proof: Let the robot be at state $s_i = \langle i, i_p, \mathbf{i}_o \rangle$. We denote by $\mathcal{B}_t \subseteq E$ the (deterministic) set of edges that have been permanently labelled occupied (i.e. with observation 1 and consequently fixed to 1 by Definition 1 after the t-th replanning event). Observe that $\mathcal{B}_0 \subseteq \mathcal{B}_1 \subseteq \ldots$ and $|\mathcal{B}_t| \leq |E| < \infty$.
Step 1: no spatial cycles that return to an already *visited* node for the current policy in i. If no human is detected along the first-choice edge, the current policy coincides with the previous optimal policy and, by Lemma 1, cannot contain a beneficial loop returning to i. If a human *is* detected, denote the current replanning episode as τ and let π_τ be the policy in force between τ and the first subsequent detection of a new occupied edge. By construction, every edge in $\mathcal{B}_\tau$ is modelled with cost $c_e(\text{obs} = 1)$ *and* with all $(1 \to 0)$ transitions from neighbor nodes removed; hence condition (ii) of Lemma 1 is *false* for every edge in $\mathcal{B}_\tau$. Consequently, π_τ contains no possible spatial loop for all the states already visited.
Step 2: finiteness of the number of policy episodes. Each detection of a new blocked edge strictly increases $|\mathcal{B}_t|$. Because $|E|$ is finite, only finitely many replanning episodes can occur.
Step 3: connectivity guarantees reachability. Removing $1 \to 0$ transitions does *not* delete physical edges from G; it only blocks stochastic "reset" arcs in the MDP abstraction. The spatial graph, therefore, remains connected, and every local policy $\pi^\star_{\text{new}}$ retains at least one path from its starting state to the goal.
Step 4: convergence. Assume, by contradiction, that the overall execution visits some node i infinitely often. Because Step 1 forbids revisiting a node within the *same* policy episode, each return to i must be preceded by the detection of a *new* blocked edge. Step 2 implies that only finitely many such detections can happen, hence only finitely many returns to i are possible—a contradiction. Thus, no infinite spatial (and therefore state) cycle can arise.
Let T be the (finite) time of the last detection event. From T, the policy never changes, contains no loops (by Step 1 with $\tau = T$), and possesses a feasible path to the goal (Step 3). Standard MDP arguments then yield that the goal is reached in finite time with probability 1. ∎

This strategy is efficient in avoiding loops. However, in the policy computed offline, in the presence of the previously described conditions, the nodes' cost may be underestimated. This is because the expected cost of moving along the loop and performing a different observation is accounted for instead of passing on the occupied edge.

The absence of memory of the MDP is also the reason why we introduced the hypothesis of not turning back at the previous edge. If we allow the robot to turn around and go back in the action selection, the minimum loop from Fig. 3 would simply be returning to the previous node and going back. This would therefore result in a higher underestimation of certain paths during the offline path selection.

B. Computational Complexity

In the worst-case scenario, considering the formula for the number of states (9), the state-space size grows linearly with the number of intersection points in the map, but exponentially with the maximum out-degree of the nodes $u_{\max}$, leading to:

$$n_s = \mathcal{O}\left(N\, u_{\max}\, 2^{u_{\max}}\right).$$

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

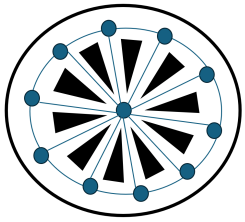


Fig. 7. Environment with high u_{max} that would make n_s grow as $u_{max}2^{u^{max}-1}$. Such topologies do not occur in typical indoor environments.

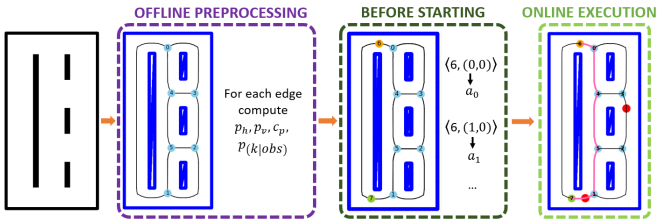


Fig. 8. Method overview: Starting from a black-and-white picture of the environment the homotopy classes graph is generated offline. In this step, we also compute graph edge properties and expected costs. When a start and goal are selected, the graph and graph properties are locally modified. The *MDP* resolution algorithm is run to find the optimal policy. Online, the robot moves with the precomputed policy, which is modified online when needed.

The computational cost of a single sweep of the value iteration algorithm is: $\mathcal{O}(n_s\, u_{\max}) = \mathcal{O}\left(N\, u_{\max}^2\, 2^{u_{\max}}\right)$. Given the known theoretical convergence bound for stochastic shortest-path MDPs with strictly positive costs, the number of sweeps required is at most $\mathcal{O}\left(n_s \log \frac{1}{\varepsilon}\right)$ for a given precision ε. This yields a worst-case overall runtime of: $\mathcal{O}(n_s^2\, u_{\max}) = \mathcal{O}\left(N^2\, u_{\max}^3\, 4^{u_{\max}}\right)$, thus being exponential in $u_{\max}$.
A pathological example of this behavior is illustrated in Fig. 7. Even though the homotopy classes graph includes only 11 nodes, the resulting MDP would have 5240 states. However, such high-degree nodes are rarely found in real-world indoor environments. Typical warehouse-like settings are big (high number of intersection points N), but don't have more than four or five outgoing edges from an intersection. Thus, our claim of low computational overhead refers to *empirical runtimes* observed in realistic scenarios, where $u_{\max}$ is small and the total number of states remains tractable.
However, performance may be affected in artificially "star-like" environments, leading to a limitation of our formulation.

V. Risk-aware routing in a shared dynamic environment

In the following, we present how our approach can be applied to account for the risk associated with human encounters in route selection in a shared dynamic environment. We outline a method to output the best strategy to follow starting from a black-and-white picture of the environment, and some information on human distributions.

The method is divided into four phases:

I. Homotopy classes Graph Computation: In this phase, from the image of the environment, we generate our homotopy classes graph and compute static properties (section V-A).
II. Adding Start and Goal: In this step, we add the start and goal and connect them as nodes to the previously computed static graph, modifying the properties of the edges involved accordingly.
III. Compute the optimal policy: Determining the homotopy class that minimizes the risk of encounters in the environment corresponds to finding the "best" policy along the homotopy classes graph. In this phase, we model the problem as an *MDP* (section V-B).
IV. Modifying Optimal Policy Online: As discussed in section IV-A the produced offline policy may be suboptimal and generate loops in certain conditions due to the reset property. So, in this phase, the optimal policy is modified online each time a human is seen.

The method overview is reported in Fig. 8.

A. Homotopy classes graph

To compute and characterize the homotopy classes graph, we first compute the edges of the generalized Voronoi diagram of the static map. Given a set $O = o_1, o_2, .., o_n$ of geometrical objects such as lines and polygons (representing walls and obstacles in our scenario), a generalized Voronoi diagram is a partitioning of the plane such that each region contains the points closest to a particular geometrical object o_i. A Voronoi edge is the boundary between two regions. We take the Voronoi edges from the generalized Voronoi diagram and use them to create a graph $\mathcal{G} = (V, E)$. To do so, we select the intersection points of three or more branches of the generalized Voronoi diagram. These intersection points, together with the goal and the starting position of the robot, will be the nodes in V of our homotopy classes graph. The edges E would result from the chains of the Voronoi edges that connect them. Using this graph, we consider the path with the lowest static collision risk (the path with the highest distance from obstacles) for each homotopy class. We will use this single candidate to compute all the static properties of the corridor (such as length and visibility) that will be needed to model the problem. However, we will assume the presence of a local planner (not part of this paper) able to locally modify the path inside the homotopy class to avoid the humans encountered.

We start by computing static properties depending on the homotopy's graph geometry.

1) p_h - Human Probability: The human probability, p_h, represents the relative probability that a person is on an edge, instead of that on the others. For example, assuming that people are uniformly distributed, it is:

$$p_h[e] = \frac{l_e}{l_t} \tag{13}$$

Where l_e is the homotopies graph edge length, and l_t is the length sum of all edges. However, any probability distribution can be used as long as $\sum_{e \in E} p_h[e] = 1$.

2) p_v - Visibility: When there are people on the edge, the probability of seeing them is equal to the probability that they are on the percentage of the edge visible from the node from which the robot makes the observation. A point from the edge is considered visible from the node when the line connecting them doesn't encounter any static obstacle (see Fig. 4 for $l_v[S][(S,G)]$ visualization). For example, in the case of uniform human presence probability along the corridor, the probability that a single person is in the visible region of the homotopy class would be:

$$p_v[v][e] = \frac{l_v(v,e)}{l_e}. \tag{14}$$

l_v is computed by discretizing the Voronoi chains and checking if each edge point is visible from the adjacent nodes.

3) c_p - Severity Coefficient: The severity coefficient c_p is the penalty we add in our metric each time an encounter between the robot and a human happens. To quantify this variable, we consider the consequences of a human-robot encounter. The possible consequences are:

- The robot must slow down or stop for safety reasons.
- If there is enough space, the robot can locally replan a new route in the current homotopy class.
- To pass, the robot has to enter the human personal space, causing stress and mental fatigue [5].
- If the robot is transporting a huge amount of load, the interaction can be perceived as not safe.

All these undesired events depend on the human-robot distance. Therefore, the performance and the possibility of performing a safe replanning are strictly connected with the width of the corridor.

The severity of encountering a human will not be the same for all corridors on the map. It will depend on the robot's dimensions, its load, and the geometric width of the corridor.

For this work, we consider a linear dependence of the severity coefficient with the corridor width and propose the following formulation:

$$c_p = \max\left(0, \min\left(-k_1(w_c - w_r) + k_2,\, k_3\right)\right) \tag{15}$$

Where w_c is the corridor's minimum width, w_r is the robot's width, and the coefficients are chosen according to the specific application scenario (robot mass, robot target speed, human familiarity with the scenario) so that the severity coefficient for each corridor spans between 0 and k_3. w_c is computed from the generalized Voronoi diagram by checking the distance to the nearest fixed obstacle on discretized points along the Voronoi graph. This severity criterion encompasses both the time the robot will incur by decelerating or halting to ensure a smooth and risk-free local re-planning, as well as the psychological strain experienced by the human operator.

B. MDP Formulation

We will now first compute P and R matrices assuming people are static during the robot route execution (e.g., they are working in specific locations of the environment), and then extend to a scenario where some people may enter the corridor while the robot is already moving in it. For simplicity, we approximate static and dynamic people in the environment modeled by two known independent distributions. We consider an encounter to happen when a person is already in the chosen corridor and when a human enters the corridor in the opposite direction while the robot moves in it. We will, in the following, denote the probabilities relative to the static people distribution as $p_s(\cdot)$, the ones relative to the moving people distribution as $p_m(\cdot)$, and the probabilities accounting for both as $p(\cdot)$.

For example, in case the presence of static humans in all the environment follows a binomial distribution, given n_{mean} mean number, N_p maximum number, and 0 minimum number of people in the environment, the probability that there are n static people is:

$$p_n = \binom{N_p}{n}\left(\frac{n_{mean}}{N_p}\right)^n\left(1 - \frac{n_{mean}}{N_p}\right)^{N_p - n}. \tag{16}$$

1) P matrix: To explicitly compute matrix P with algorithm (1) we need to compute $p_o(obs)$. The probability that all the n people are out ($obs = 0$), or at least one is in ($obs = 1$) the visible region of the corridor is:

$$p_{s,o}(obs) = \sum_{n=0}^{N_p} p_n((1 - p_v p_h)^n)^{1-obs}(1 - (1 - p_v p_h)^n)^{obs}. \tag{17}$$

2) R matrix: To compute the R matrix, according to eq. (12), we need to compute explicitly $p(k|obs)$. Applying Bayes rule $p(k|obs)$ becomes:

$$p(k|obs) = \frac{p(obs|k)p_k(k)}{p_o(obs)},$$

where $p_k(\cdot)$ represents the probability of encountering k people while going through the corridor.

The probability of making a certain observation from a node in an adjacent edge is the same as already computed in (17). The probability of making an observation of 0 or 1 when in the corridor there are k people, $p_s(obs|k)$, depends on the visibility probability computed in (14). If there are no people ($k = 0$) $p_s(obs = 0|k = 0) = 1$. Otherwise, ($k > 0$), $p_s(obs = 0|k)$ is the probability that all the people are in the not visible portion of the corridor. $p_s(obs = 1|k)$ is the probability that at least one person is visible. In general form, it would be:

$$p_s(obs|k) = (1 - p_v)^{(k(1-obs))}(1 - (1 - p_v)^k)^{obs}. \tag{18}$$

The probability that there are k people in the corridor would depend on the human presence probability (13). For each possible number of people in the environment n, we compute the probability that k of the n humans are in the corridor.

$$p_{s,k}(k) = \sum_{n=0}^{N_p} p_n \binom{n}{k}(p_h)^k(1 - p_h)^{(n-k)}. \tag{19}$$

3) Extension in presence of Moving People: In this paper, we model the presence of moving people alongside stationary ones by considering them as independent distributions. Let us denote p_e as the probability that in a time unit (e.g., $1s$), a person enters a corridor in the direction opposite to the robot's movement. Calling T the time (rounded to the nearest integer) the robot takes to traverse the corridor, the probability p_m that k people will enter the corridor while the robot is in it is:

$$p_{m,k}(k) = \binom{T}{k} p_e^k (1-p_e)^{T-k}. \tag{20}$$

The probability of making a certain observation from a node does not change if some people enter the corridor, as the observation is instantaneous and the visible portion of the corridor is the same. The P matrix will therefore maintain the same formulation, and:

$$p_o(obs) = p_{s,o}(obs). \tag{21}$$

To compute the total probability that the robot will encounter k people in the corridor, we notice that of these k people, a certain number, j will be static and already in the corridor when the robot chooses it (in the visible part or not), and $k-j$ would be not yet in the corridor, but will enter while the robot is traversing the corridor. This probability can be written as:

$$p_k(k) = \sum_{j=0}^{k} p_{s,k}(j) p_{m,k}(k-j). \tag{22}$$

The probability of making a certain observation knowing that we will encounter k people in the corridor $p(obs|k)$ takes into account the fact that, as we modeled the problem so far, the $(k-j)$ people not yet in the corridor are for sure not visible. The probability that if the robot encounters k people, j are static is:

$$p(j|k) = \frac{p(k|j)p(j)}{p(k)} = \frac{p_{m,k}(k-j)p_{s,k}(j)}{p_k(k)}. \tag{23}$$

So we have:

$$p(obs|k) = \sum_{j=0}^{k} p(j|k) p_s(obs|j). \tag{24}$$

Computed the P and R matrices, we solve the *MDP* with the Value Iteration algorithm as described in section IV.

VI. Validations

We tested our problem modeling in three different environments with increasing map complexity. Each map highlights different aspects of our formulation. The first and the second maps are used to compare our approach with more classical algorithms, and the effects on the performances by varying the severity coefficients c_p, human density, and moving people's presence.

To study the different effects on the c_p choice for the first two maps, we will use different values of c_p fixed for all the map corridors. Note that, as we consider people as uniformly distributed in the environment, the path with the minor probability of encountering someone, and so, in case of fixed c_p, the one with the minimum expected cost, is also the minimum length path. The third map, instead, is used to study the scalability of our approach and the effects of having different severity coefficients on portions of the environment. For the third map, the severity coefficient for each corridor is found using eq. 15. The framework has been implemented in Python on an i7 2000 MHz 12th Gen Intel processor with 16 GB of memory.

To obtain the generalized Voronoi diagram from the environment map picture, we use the code available in [33] [3]. The *MDP* is solved using the Value Iteration algorithm with the Markov Decision Process Toolbox for Python [4].

For each simulation, a random number n of people between 0 and N_p are considered present and static in the environment (e.g., performing a task in a certain position), with n_{mean} the mean number of static humans that will be present in the environment. From now on, this information will be expressed as $hum_{info} = (0, n_{mean}, N_p)$. The n people extracted are randomly located along the homotopy classes graph with uniform probability. For the simulations where we consider the moving people presence, to compute the probability that k humans will enter the corridor while the robot is in it, we use (20), taking p_e constant and equal for all the warehouse corridors. Robot velocity is considered $1m/s$, so T is each corridor's integer rounded path length. The performances of the three algorithms are compared for each random human configuration.

For each scenario varying hum_{info}, p_e, c_p we perform 10000 simulations when only static people presence is considered, and 20000 when we account for people entering the corridor while the robot traverses it (in this case, the degree of freedom in each simulation is higher).

A. Algorithms selected for comparison

1) A and Modified A*:* To compare our approach with reactive strategies, we define an expected cost graph $\mathcal{G}_{EC}(V, E)$, having the same nodes and edges of $\mathcal{G}$, but the weight of each edge is given by:

$$c_e = l_e + \sum_{k=0}^{N_p} p_k(k)\, k\, c_p[edge]. \tag{25}$$

We can compute the minimum expected cost path, approximating edge costs as independent, for example, by applying an A* algorithm to $\mathcal{G}_{EC}$ considering the Euclidean distance between nodes as heuristic.

We also use as a comparison a modified A* algorithm that every time a $\mathcal{G}_{EC}$ node is reached (including S), the cost of its outgoing edges star are updated as:

$$c'_e = l_e + \sum_{k=0}^{N_p} p(k|obs)\, k\, c_p[edge], \tag{26}$$

and the new minimum-expected cost path is computed.

[3] https://github.com/ross1573/generalized_voronoi_diagram

[4] https://pymdptoolbox.readthedocs.io/en/latest/

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Unlike our MDP approach, the Modified A* algorithm is allowed to turn back and take the same node it came from at intersection points if convenient.

2) RAGS: The *Risk-Aware Graph Search (RAGS)* algorithm, originally introduced by Chung et al. [21], addresses graph search problems characterized by uncertain edge costs, modeled as independent Gaussian random variables. In their original formulation, edge costs are initially unknown, and the true cost of each edge is dynamically revealed upon reaching its originating node. In our scenario, however, the edge costs are represented by discrete probability mass functions (PMFs), rather than Gaussian distributions. Furthermore, unlike the original setting, upon reaching a node, the actual cost of outgoing edges is not immediately revealed. Instead, the initial discrete distribution $p(k)$ is dynamically updated into the conditional distribution $p(k|obs)$, where $obs \in \{0, 1\}$. Below, we detail how we adapted the RAGS algorithm to make it suitable for our scenario and to facilitate a fair comparison with it.

The first phase of the RAGS algorithm is the *initial sweep*, whose purpose is to identify a set of acyclic, non-dominated paths connecting the start and goal nodes. A path A is said to dominate a path B when:

$$p(c(A) < c(B)) > d_{\text{thr}}, \quad d_{\text{thr}} \in [0.5, 1), \tag{27}$$

where d_{thr} is a tunable parameter, a greater d_{thr} increases the computational cost of the initial sweep while increasing the chance of including in the set the one that in the real world would turn out to be the minimum cost path.

In our implementation with discrete probability distributions, the dominance criterion between two paths A and B can be computed as:

$$p(c(A) < c(B)) = \sum_x p(c(A) = x) \sum_{y>x} p(c(B) = y), \tag{28}$$

where $x, y \in \mathcal{K}$ represent the discrete cost values.

The initial sweep essentially performs an A^*-like search, maintaining an open set of paths that any other path has not yet dominated.

To compute the discrete probability distribution of the path resulting from the union of two or more edges, we use:

$$p(Z = z) = \sum_{x+y=z} p(X = x)\, p(Y = y), \tag{29}$$

where X and Y represent independent discrete edge costs, and Z their combined path cost.

Moreover, each edge explicitly represents the probabilities associated with costs corresponding to a discrete number of people, from 0 to N_p. Consequently, when combining the distributions, we enforce this constraint by limiting the convolution to cases where the sum of the corresponding indices remains within the allowable maximum:

$$p(Z = z) = \sum_{\substack{i+j \le N_p \\ x_i + y_j = z}} p(X = x_i)\, p(Y = y_j). \tag{30}$$

The online phase of the RAGS algorithm dynamically selects, at each decision node, the successor node with the highest probability of belonging to the lowest-cost non-dominated path towards the goal. Formally, the algorithm would choose node v over node u if:

$$p(c_{v,\min} < c_{u,\min}) \ge 0.5, \quad \forall u \in V \setminus \{v\}, \tag{31}$$

where $c_{v,\min}$ and $c_{u,\min}$ are the costs of the minimum cost path from v and u.

In our discrete implementation, this criterion becomes:

$$p(c_{v,\min} < c_{u,\min}) = \sum_x p(c_{v,\min} = x) \sum_{y>x} p(c_{u,\min} = y), \tag{32}$$

where $x, y \in \mathcal{K}$, the support of the discrete cost values.

To have a fair comparison, we made RAGS leverage the results in Section V-A to compute p_k and $p(k|obs)$. Since RAGS strongly depends on d_{thr}, this parameter determines which possible paths to the goal survive the initial sweep selection. We compare in the following validations with both RAGS $d_{\text{thr}} = 0.6$ and RAGS $d_{\text{thr}} = 0.8$.

In section VI-F, we compare the expected cost of the different policies with the costs resulting from the simulations. The *A** expected cost is computed as the length of the minimum-cost path in $\mathcal{G}_{EC}(V, E)$. Denoting as $c_o[s_i]$ the cost each state s_i of the S node converges to after the value iteration procedure, the *MDP* expected cost is:

$$\mathbb{E}[C]_{\Pi_{\text{MDP}}} = \sum_{o_i} \sum_{o_j} p_o\big(\langle S, (o_i, o_j)\rangle\big) c_o\big[\langle S, (o_i, o_j)\rangle\big], \quad \text{with } o_i, o_j \in \{0, 1\}. \tag{33}$$

As reactive strategies, the *A* mod* and *RAGS* policies don't have a prior estimated cost.

Unless otherwise specified, the results in the tables are presented as the sample mean $\pm 1.96SE$, where SE is the standard error of the mean. This interval corresponds to a 95% confidence level, assuming the sampling distribution of the mean is approximately Gaussian, in accordance with the Central Limit Theorem. To statistically validate that the *MDP* algorithm leads to a lower mean cost according to the metric in Eq. 7, we perform a paired permutation test for each tested scenario. Specifically, we compare the costs of *MDP* against *A**, *A* mod*, and *RAGS*, maintaining the pairing of costs for the same human configuration. The test evaluates whether the observed mean difference between the two distributions (*MDP* cost - *A*/A* mod/RAGS* cost) is significantly smaller than the differences obtained by resampling under the null hypothesis. The null hypothesis assumes that the expected costs of the two algorithms are equal. The null hypothesis would be H_0: *the mean difference (MDP cost - A*/A* mod cost/RAGS) is greater than or equal to zero*, and the alternative hypothesis H_1: *the mean difference is less than zero*. Discarding the null hypothesis indicates that *MDP* achieves a lower expected cost.

B. Small Map - Replanning Possibilities and Visibility

In this map, we will consider fixed start and goal positions. The path associated with the minimum expected cost according to the A* algorithm is the one that directly connects the start and end nodes. However, this path does not present escaping possibilities when a person is in it. Moreover, from

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

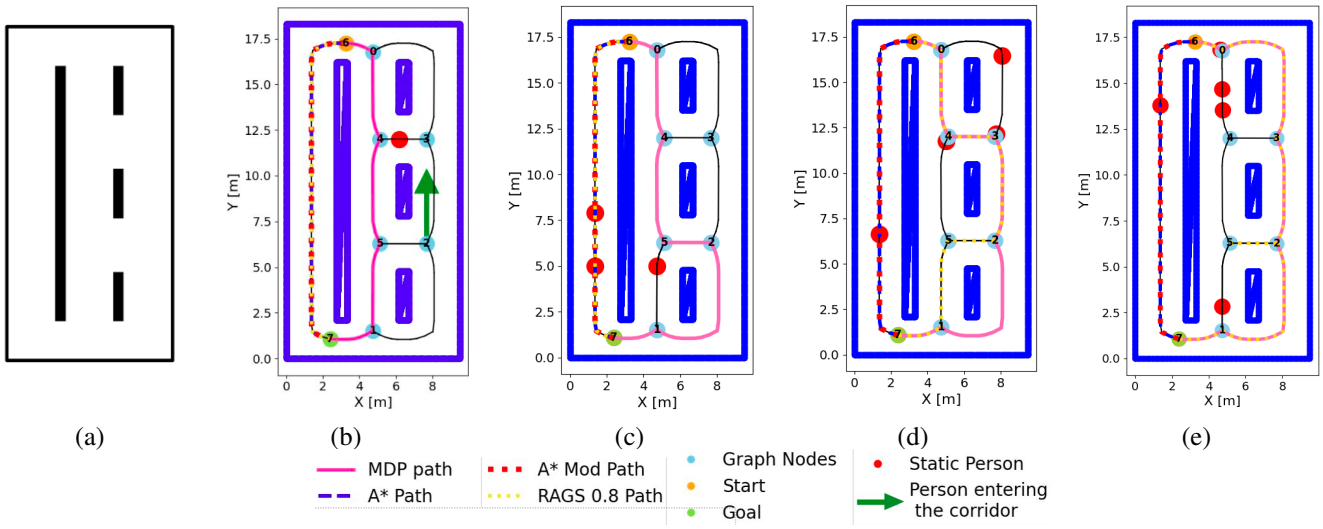


Fig. 9. **9a)**: Map of the small warehouse. **9b-9c)**: Examples of path results in the small map from a fixed starting position (in orange) to a fixed goal (in green). Severity: $c_p = 30$, $hum_{info} = (0, 2, 5)$, $p_e = 0.005$. **9d-9e)**: Severity: $c_p = 30$, $hum_{info} = (0, 4, 8)$, $p_e = 0$. Path colors: MDP (Pink), A* (Blue), A* mod (Red), RAGS ($d_{\text{thr}} = 0.8$) (Yellow). Static people are presented as red dots, moving people entering the corridor while the robot is in it are presented as a green arrow indicating the moving direction.

TABLE I
SMALL MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 10000 SIMULATIONS, $p_e = 0$ AND $hum_{info} = (0, 2, 10)$

	$c_p = 5$			$c_p = 20$			$c_p = 40$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP	$\mathbf{20.57 \pm 0.06}$	0.41 ± 0.01	18.50 ± 0.01	$\mathbf{25.84 \pm 0.20}$	0.21 ± 0.01	21.66 ± 0.06	$\mathbf{29.76 \pm 0.38}$	0.20 ± 0.01	21.66 ± 0.06
A*	21.06 ± 0.07	0.55 ± 0.01	18.32 ± 0.00	29.59 ± 0.29	0.56 ± 0.01	18.32 ± 0.00	40.84 ± 0.56	0.56 ± 0.01	18.32 ± 0.00
A* mod	$\mathbf{20.57 \pm 0.06}$	0.41 ± 0.01	18.50 ± 0.01	26.28 ± 0.24	0.37 ± 0.01	18.81 ± 0.03	33.92 ± 0.47	0.38 ± 0.01	18.81 ± 0.03
RAGS ($d_{\text{thr}} = 0.8$)	$\mathbf{20.57 \pm 0.06}$	0.41 ± 0.01	18.50 ± 0.01	26.29 ± 0.24	0.37 ± 0.01	18.80 ± 0.03	33.99 ± 0.47	0.38 ± 0.01	18.79 ± 0.03
RAGS ($d_{\text{thr}} = 0.6$)	21.06 ± 0.07	0.55 ± 0.01	18.32 ± 0.00	29.59 ± 0.29	0.56 ± 0.01	18.32 ± 0.00	40.84 ± 0.56	0.56 ± 0.01	18.32 ± 0.00

TABLE II
SMALL MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 10000 SIMULATIONS, $p_e = 0$ AND $c_p = 30$.

	$hum_{info} = (0, 1, 3)$			$hum_{info} = (0, 2, 5)$			$hum_{info} = (0, 4, 8)$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP	$\mathbf{22.71 \pm 0.16}$	0.06 ± 0.01	20.81 ± 0.05	$\mathbf{27.49 \pm 0.27}$	0.19 ± 0.01	21.70 ± 0.06	$\mathbf{40.58 \pm 0.46}$	0.60 ± 0.01	22.66 ± 0.06
A*	26.83 ± 0.30	0.28 ± 0.01	18.32 ± 0.00	34.91 ± 0.41	0.55 ± 0.01	18.32 ± 0.00	51.63 ± 0.57	1.11 ± 0.02	18.32 ± 0.00
A* mod	24.26 ± 0.25	0.19 ± 0.01	18.50 ± 0.02	29.86 ± 0.34	0.37 ± 0.01	18.82 ± 0.03	42.24 ± 0.50	0.76 ± 0.02	19.52 ± 0.05
RAGS ($d_{\text{thr}} = 0.8$)	26.83 ± 0.30	0.28 ± 0.01	18.32 ± 0.00	29.85 ± 0.34	0.37 ± 0.01	18.79 ± 0.03	41.32 ± 0.47	0.62 ± 0.02	22.76 ± 0.07
RAGS ($d_{\text{thr}} = 0.6$)	26.83 ± 0.30	0.28 ± 0.01	18.32 ± 0.00	34.91 ± 0.41	0.55 ± 0.01	18.32 ± 0.00	51.63 ± 0.57	1.11 ± 0.02	18.32 ± 0.00

the start node, only a portion of the corridor is visible, so even if the observation is 0, there is still a high probability of making an encounter. The map has eight nodes, six from the homotopy classes graph, plus the start and goal, for a total of 77 states and 2 actions.

The map is presented in Fig. 8a, together with its homotopy classes graph. It has dimensions 10 x 17.5 m. Some path results for the MDP, A*, A* Mod, and RAGS ($d_{\text{thr}} = 0.8$) algorithms in different conditions and human configurations are reported in Fig. 9. We don't report the results for RAGS ($d_{\text{thr}} = 0.6$), as in all the studied scenarios, the only path found by the initial sweep is the A* path, and so RAGS ($d_{\text{thr}} = 0.6$) cannot replan online.

In TABLE I we report the results obtained with 10000 simulations varying the severity coefficient c_p for $hum_{info} = (0, 2, 5)$ and $p_{mov} = 0$. For a low severity value ($c_p = 5$), the algorithm derived from the *MDP* formulation behaves like the *A* mod* algorithm, passing on the left side of the big wall (unless a person is present in the visible part of the corridor). In this scenario, the paths resulting from the two algorithms are the same. When severity increases, the *MDP policy* is to pass on the central corridor, even if the length is higher. This increases the possibility of avoiding an encounter, resulting, on average, in a lower cost. The initial sweep for RAGS ($d_{\text{thr}} = 0.8$) includes the paths on the right side of the big wall. However, since RAGS doesn't directly depend on severity, for $hum_{info} = (0, 2, 10)$, the strategy that maximizes the probability of containing the minimum cost path is to pass on the left, regardless of the c_p value.

In TABLE II, we present the results of 10000 simulations varying the hum_{info} parameters. The mean cost obtained using the *MDP* policy is always slightly better than the cost

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE III
SMALL MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 20000 SIMULATIONS, $c_p = 30$ AND $hum_{info} = (0, 2, 5)$

	$p_e = 0.005$			$p_e = 0.01$			$p_e = 0.05$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP	$\mathbf{30.64 \pm 0.33}$	0.30 ± 0.01	21.71 ± 0.06	$\mathbf{34.16 \pm 0.39}$	0.41 ± 0.01	21.72 ± 0.06	$\mathbf{57.81 \pm 0.65}$	1.30 ± 0.02	18.79 ± 0.03
A*	37.92 ± 0.45	0.65 ± 0.02	18.32 ± 0.00	40.37 ± 0.48	0.73 ± 0.02	18.32 ± 0.00	62.12 ± 0.68	1.46 ± 0.02	18.32 ± 0.00
A* mod	32.86 ± 0.39	0.47 ± 0.01	18.82 ± 0.03	35.39 ± 0.42	0.55 ± 0.01	18.84 ± 0.03	$\mathbf{57.78 \pm 0.65}$	1.30 ± 0.02	18.79 ± 0.03
RAGS ($d_{\text{thr}} = 0.8$)	32.91 ± 0.39	0.47 ± 0.01	18.78 ± 0.03	35.40 ± 0.42	0.55 ± 0.01	18.82 ± 0.03	61.79 ± 0.69	1.36 ± 0.02	20.90 ± 0.05
RAGS ($d_{\text{thr}} = 0.6$)	37.92 ± 0.45	0.65 ± 0.02	18.32 ± 0.00	40.37 ± 0.48	0.73 ± 0.02	18.32 ± 0.00	62.12 ± 0.68	1.46 ± 0.02	18.32 ± 0.00

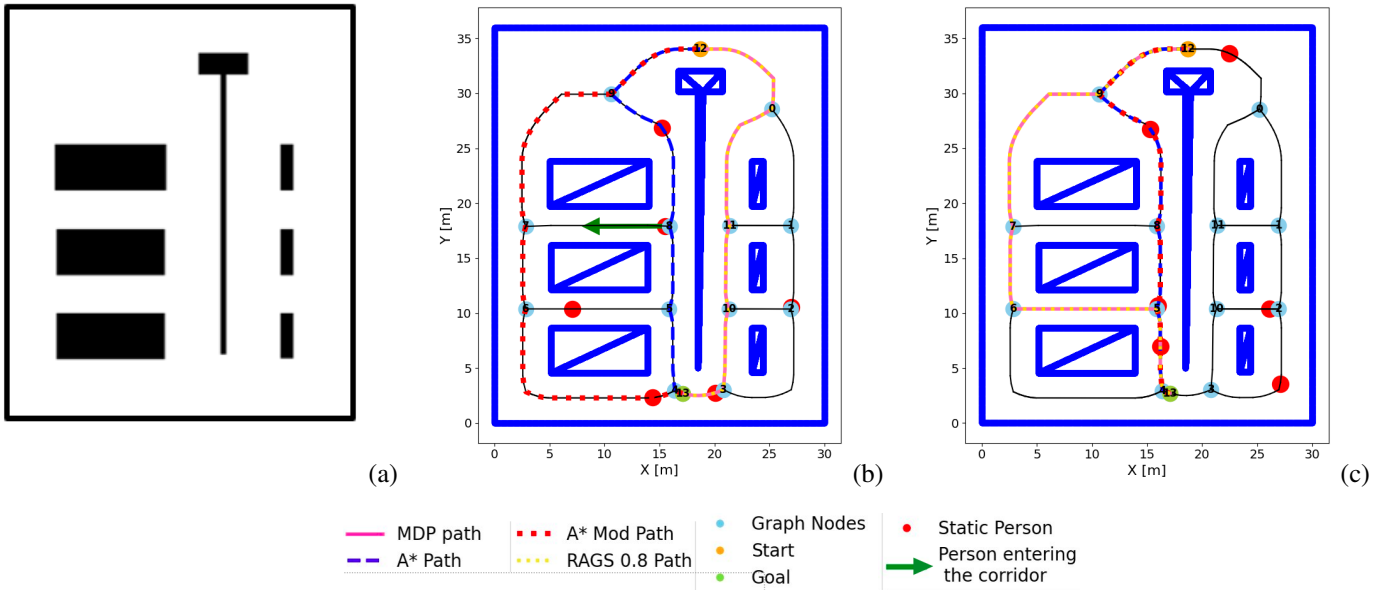


Fig. 10. **10a)**: Map of the medium warehouse. Path colors: MDP (Pink), A* (Blue), A* mod (Red). Static people are represented as red dots, and the entrances of moving people are represented as green arrows. **10b)**: Example of path results in the medium map from a fixed start position (in orange) and goal (in green). Severity: $c_p = 30$, $hum_{info} = (0, 6, 12)$, $p_e = 0.005$. **10c)** For $c_p = 20$, $hum_{info0} = (0, 6, 12)$, $p_e = 0$, the A* mod algorithm does not replan even if a human is seen in (9,8) corridor.

obtained with the reactive strategy of the *A* mod* algorithm. In this case RAGS ($d_{\text{thr}} = 0.8$) passes on the right side for $hum_{info} = (0, 4, 8)$. However, it still performs slightly worse than MDP, as in node 2 it chooses node 5 instead of 1, thus exploiting less the observable part of the path in node 2 (see Fig. 9.d,e). In TABLE III we show the results in the presence of moving people in the environment, performing 20000 simulations for each scenario. For $p_e = 0.05$, there is a high probability that people enter the corridors while the robot is traversing them. This means that the value of observations is reduced, and so, for this map, it is more convenient to pass as first-choice on the left side of the wall, resulting in the same policy of the *A* mod* algorithm. In the other tested scenarios, the *MDP* approach performs better.

C. Medium Map - Cost of the Alternative Routes

With this map, we test the ability of our approach to account for the expected cost of the alternative routes. The map is shown in Fig. 10, along with some path results varying human configurations and parameters. Again, we don't show the RAGS ($d_{\text{thr}} = 0.6$) path as it is the same as A*. The map has 12 nodes plus the start and goal, for a total of 149 states and 2 actions. The start and goal are fixed, and in this scenario, the single path associated with the minimum expected cost is the one passing left of the long obstacle in the middle. However, by passing right, the length of the alternative routes in case the first choice corridor is occupied is minor. The environment has dimensions 30 x 35 m, and c_p is kept constant for all the map corridors. In TABLES IV, V are reported the results for 10000 simulations, varying human densities and the severity coefficient. In table VI, the results of 20000 simulations varying the entering probability p_e are presented.

In this scenario, the *MDP* algorithm performs as the *A* mod* only for the scenario with $c_p = 5$, in the other case studies it leads to a smaller mean cost. Depending on the parameter choice the *A* mod* algorithm may choose to remain on the *A** path even if a person is seen as the estimated expected cost of the best alternative route is higher than the cost to make the encounter on the first-choice path (Fig. 9c). However, this is not necessarily the best option, as more replanning possibilities are present after the first rerouting edge (9,7) is taken. In the scenario in the picture, the *MDP* policy takes the path on the left from 12 as (12,0) is occupied and then decides to take (9,7) to avoid the person in (9,8).

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE IV
MEDIUM MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 10000 SIMULATIONS, $p_e = 0$ AND $hum_{info} = (0, 6, 12)$

	$c_p = 5$			$c_p = 20$			$c_p = 40$		
	Cost	Enc.	Len.	Cost	Enc.	Len.	Cost	Enc.	Len.
MDP	**45.44 ± 0.09**	0.95 ± 0.02	40.68 ± 0.02	**56.31 ± 0.32**	0.54 ± 0.01	45.47 ± 0.11	67.24 ± 0.60	0.53 ± 0.01	46.14 ± 0.12
A*	46.27 ± 0.10	1.22 ± 0.02	40.18 ± 0.00	64.22 ± 0.41	1.20 ± 0.02	40.18 ± 0.00	88.78 ± 0.82	1.21 ± 0.02	40.18 ± 0.00
A* mod	**45.44 ± 0.09**	0.95 ± 0.02	40.68 ± 0.02	59.24 ± 0.37	0.90 ± 0.02	41.21 ± 0.06	70.42 ± 0.63	0.60 ± 0.02	46.33 ± 0.20
RAGS ($d_{\text{thr}} = 0.8$)	45.54 ± 0.09	1.01 ± 0.02	40.51 ± 0.01	56.51 ± 0.32	0.52 ± 0.01	46.05 ± 0.12	**66.96 ± 0.60**	0.49 ± 0.01	47.45 ± 0.16
RAGS ($d_{\text{thr}} = 0.6$)	46.27 ± 0.10	1.22 ± 0.02	40.18 ± 0.00	64.22 ± 0.41	1.20 ± 0.02	40.18 ± 0.00	88.78 ± 0.82	1.21 ± 0.02	40.18 ± 0.00

TABLE V
MEDIUM MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 10000 SIMULATIONS, $p_e = 0$ AND $c_p = 30$

	$hum_{info} = (0, 2, 5)$			$hum_{info} = (0, 6, 15)$			$hum_{info} = (0, 10, 15)$		
	Cost	Enc.	Len.	Cost	Enc.	Len.	Cost	Enc.	Len.
MDP	**46.48 ± 0.14**	0.09 ± 0.00	43.89 ± 0.06	**61.90 ± 0.46**	0.55 ± 0.01	45.53 ± 0.11	82.22 ± 0.67	1.20 ± 0.02	46.20 ± 0.11
A*	52.37 ± 0.25	0.41 ± 0.01	40.18 ± 0.00	76.92 ± 0.62	1.22 ± 0.02	40.18 ± 0.00	100.56 ± 0.78	2.01 ± 0.03	40.18 ± 0.00
A* mod	47.72 ± 0.18	0.17 ± 0.01	42.55 ± 0.09	64.29 ± 0.48	0.62 ± 0.02	45.80 ± 0.16	85.67 ± 0.70	1.36 ± 0.02	44.76 ± 0.13
RAGS ($d_{\text{thr}} = 0.8$)	49.66 ± 0.22	0.31 ± 0.01	40.36 ± 0.01	62.05 ± 0.46	0.49 ± 0.01	47.35 ± 0.16	**81.80 ± 0.66**	1.17 ± 0.02	46.65 ± 0.13
RAGS ($d_{\text{thr}} = 0.6$)	52.37 ± 0.25	0.41 ± 0.01	40.18 ± 0.00	76.92 ± 0.62	1.22 ± 0.02	40.18 ± 0.00	100.56 ± 0.78	2.01 ± 0.03	40.18 ± 0.00

TABLE VI
MEDIUM MAP: COSTS, ENCOUNTERS AND LENGTHS FOR THE PATHS OBTAINED WITH 20000 SIMULATIONS, $c_p = 30$ AND $hum_{info} = (0, 6, 12)$

	$p_e = 0.005$			$p_e = 0.01$			$p_e = 0.05$		
	Cost	Enc.	Len.	Cost	Enc.	Len.	Cost	Enc.	Len.
MDP	**68.22 ± 0.38**	0.76 ± 0.01	45.49 ± 0.08	75.77 ± 0.44	1.01 ± 0.01	45.48 ± 0.08	**129.25 ± 0.69**	2.90 ± 0.02	42.37 ± 0.03
A*	82.95 ± 0.48	1.43 ± 0.02	40.18 ± 0.00	88.55 ± 0.51	1.61 ± 0.02	40.18 ± 0.00	138.36 ± 0.73	3.27 ± 0.02	40.18 ± 0.00
A* mod	71.88 ± 0.41	0.93 ± 0.01	44.09 ± 0.09	78.57 ± 0.46	1.15 ± 0.01	44.05 ± 0.09	131.13 ± 0.70	3.01 ± 0.02	40.70 ± 0.02
RAGS ($d_{\text{thr}} = 0.8$)	68.64 ± 0.38	0.73 ± 0.01	46.80 ± 0.11	**75.53 ± 0.43**	0.99 ± 0.01	45.69 ± 0.08	129.86 ± 0.70	2.90 ± 0.02	43.01 ± 0.05
RAGS ($d_{\text{thr}} = 0.6$)	82.95 ± 0.48	1.43 ± 0.02	40.18 ± 0.00	88.55 ± 0.51	1.61 ± 0.02	40.18 ± 0.00	138.36 ± 0.73	3.27 ± 0.02	40.18 ± 0.00

The RAGS algorithm with $d_{\text{thr}} = 0.8$ always takes the path to the right of the long wall in this map, except when $hum_{info} = (0, 2, 5)$, in which case it passes on the left. When it goes right (towards node 0), it follows a policy similar to that of the MDP, with the key difference that RAGS is allowed to backtrack to the previous decision node. Under certain parameter settings, this flexibility allows RAGS to achieve slightly better performance than the MDP approach.

D. Big Map - Scalability and Severity

In this case study, we test our approach on a complex map of big dimensions. The picture of the environment is taken from [25], and represents a real warehouse (real dimensions are not provided). We consider the map to have dimensions 210 x 35 m. The map has 157 nodes, for a total of 2499 states, and 3 actions. We will use this map to study the scalability of the proposed approach and how severity influences the policies, starting from various starting positions. Severity is static and obtained using equation 15. The coefficients are chosen so that if the corridor is wider than $w_r + 5\,m$, the severity is 0; if it is narrower than $w_r + 1\,m$, the severity is 50. In this way, we have: $k_1 = 12.5, k_2 = 62.5, k_3 = 50$. We take $w_r = 1\,m$.

At each simulation, a random starting position is chosen, the start and goal are added locally modifying the graph, the properties of the newly added edges are computed, and the *MDP* optimal policy is found. Fig. 11 shows the map picture and an example of path results in the previously described conditions, starting from different starting positions in the environment (in the same randomly extracted human presence configuration). We can see that the *MDP* algorithm prefers slightly longer paths that choose the wider corridors, where the severity, in case an encounter happens, is lower.

TABLE VII reports the computational times for the *MDP* formulation for the Before Starting, *MDP* resolution, and online modification parts, for the three maps. The data are expressed as mean ± standard deviation in this case. The "Before Starting" phase includes the local modification of the static graph and its properties to include the start and goal and the first *MDP* solving (not the offline preprocessing phase). For the small and medium maps, adding the start and goal to the map is also done by randomly selecting a different start in each simulation. Most of the time is spent computing the visibility property for the modified edges, so it is strictly dependent on the length of the edge to split. The "*MDP* Resolution" column highlights the computational time needed to solve the *MDP* using the Value Iteration algorithm. "Online Modification" indicates the time required to modify the policy online by adjusting the P matrix and rerun the *MDP* online when a human is seen.

TABLE VIII reports the order of magnitude of the time required to perform the RAGS initial sweep, for $d_{\text{thr}} = 0.6$ and $d_{\text{thr}} = 0.8$. While the time for $d_{\text{thr}} = 0.6$ scales well, the time required for $d_{\text{thr}} = 0.8$ explodes when increasing the map dimensions. However, we saw in the previous tables that the initial sweep with $d_{\text{thr}} = 0.6$ only finds the minimum expected cost path; thus, during online execution, it behaves exactly like

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

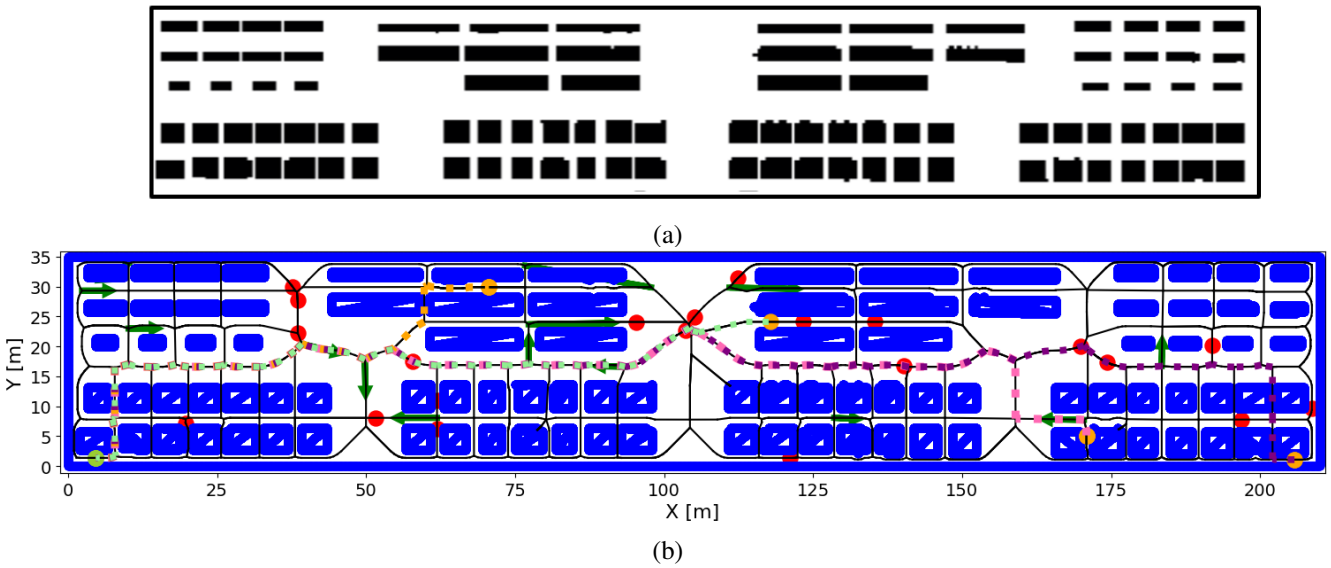


Fig. 11. **11a)**: Map of the big warehouse. **11b)**: Path results in the big map with random starting positions (in orange) and a fixed goal point (in green). Different path colors represent different paths founded by the MDP policy, changing the starting position in the same human configuration. Static people are represented as red dots, and the entrances of moving people are represented as green arrows. Results obtained for $hum_{info} = (0, 20, 30)$, $p_e = 0.005$.

TABLE VII
COMPUTATIONAL TIMES FOR THE CASE STUDY MAPS, 100 TRIALS.

	$time\,[s]$		
	Before Starting	MDP Resolution	Online Modification
Small Map	$10^{-2}(2.5 \pm 0.7)$	$10^{-4}(4.0 \pm 1.2)$	$10^{-3}(1.3 \pm 0.1)$
Medium Map	$10^{-1}(1.6 \pm 0.1)$	$10^{-4}(7.8 \pm 7.6)$	$10^{-3}(7.0 \pm 4.3)$
Big Map	$10^{0}(1.5 \pm 0.3)$	$10^{-2}(3.2 \pm 1.3)$	$10^{-1}(2.7 \pm 0.8)$

TABLE VIII
INITIAL SWEEP TIME [S] FOR RAGS WITH DIFFERENT d_{THR} VALUES.

	$d_{\text{thr}} = 0.8$	$d_{\text{thr}} = 0.6$
Small Map	10^{-3}	10^{-3}
Medium Map	10^{-2}	10^{-3}
Big Map	10^{2}	10^{-2}

the A* planner, never triggering any replanning.
The reason why, in our discrete scenario, the initial sweep of RAGS demands substantially higher computational effort for the large map compared to the normal distribution case is that the discrete representation requires explicitly propagating complete distributions of costs for each path. Operations like dominance checks and distribution comparisons become significantly more computationally expensive than simply manipulating mean and variance, as done in the normal distributions case.

E. Online Policy Modification Impact

We will now show some scenarios where, with only the policy computed offline, the robot would be stuck in a loop until the people occupying the corridor move. We consider, as an example, the small map with $c_p = 40$.
Figures 12 a) and c) report what happens when the online policy modification is disabled. In both cases, the robot repeatedly visits the same nodes, hoping for a different observation outcome, due to the lack of memory regarding previously made observations. Figures 12 b) and d) show the path resulting from the online strategy in the same scenarios. Now the strategy avoids the loop, passes through an occupied edge, and reaches the goal.

F. Discussion

Our approach leads to lower (or at worst equal) mean costs than the *A** and *A* mod* approaches in all the studied scenarios. RAGS performs slightly better than our MDP-based approach in a few scenarios on the Medium Map, but it performs worse in others. Moreover, the computational cost of RAGS grows significantly with map size, making it impractical for large-scale environments. In TABLES IX, X, we report the estimated costs by the *MDP* resolution and *A**, and the actual costs obtained by simulation. We made two assumptions to treat the problem as discussed in section IV: we approximated edge costs and observations in different edges as independent, and assumed the *reset* property. The first assumption comports a small overestimation of a path strategy cost, as for each edge, the (very slight) possibility that all the people are on that edge is considered. Instead, the *reset* property assumption would comport a policy cost underestimation of the *MDP* in the conditions described in section III-B. This behavior can be seen in the small map (TABLE IX) for $c_p = 40$. Our approach leads to estimated costs that may slightly differ from the real cost, but that are significantly smaller than the ones obtained with the *A** algorithm. Having an estimated expected cost, in contrast to a reactive strategy, is an advantage, as this information can be used, for example, for task planning algorithms.

As shown in TABLE VII, the approach scales well and is also usable in big, complex maps.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

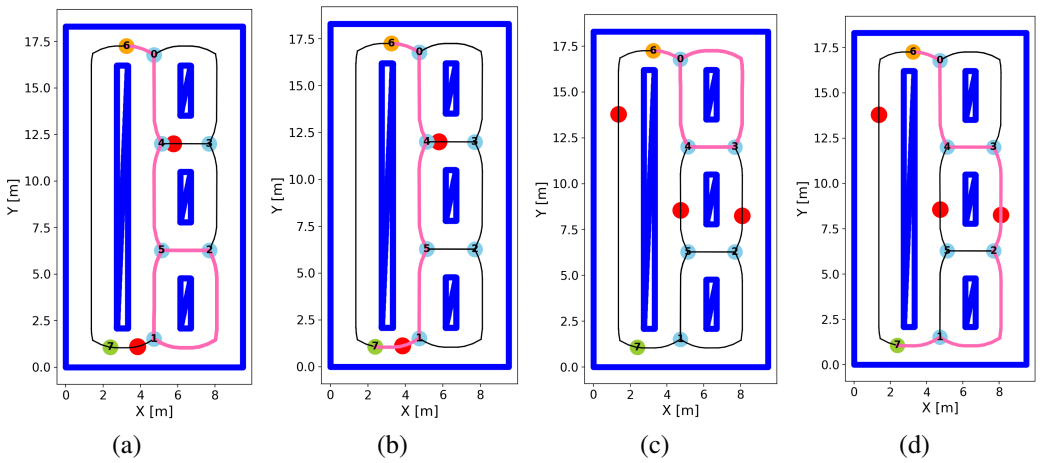


Fig. 12. Online policy modification impact, small map with $c_p = 40$. Path results from a start node (in orange) and a fixed goal (in green). The path in pink is the path obtained from the MDP policy (with the online policy modification off or on). Static people are presented as red dots. **12a,c)**: Offline policy only: the robot continues to move into a cycle without ever reaching the goal. **12b,d)**: When the online policy modification is on, in the same environment's configurations, the robot reaches the goal correctly.

TABLE IX
EXPECTED COSTS VARYING c_p, SMALL MAP. $hum_{info} = (0, 2, 10)$, $p_e = 0$. REAL MEAN COST RESULTS FOR 10000 SIMULATIONS.

Small Map	MDP		A*	
c_p	Estimated Cost	Real Cost	Estimated Cost	Real Cost
5	20.65	20.57 ± 0.06	21.11	21.06 ± 0.07
20	25.76	25.84 ± 0.20	29.50	29.59 ± 0.29
40	28.55	29.76 ± 0.38	40.67	40.84 ± 0.56

TABLE X
EXPECTED COSTS VARYING c_p, MEDIUM MAP. $hum_{info} = (0, 6, 12)$, $p_e = 0$. REAL MEAN COST RESULTS FOR 10000 SIMULATIONS.

Medium Map	MDP		A*	
c_p	Estimated Cost	Real Cost	Estimated Cost	Real Cost
5	45.44	45.44 ± 0.09	46.25	46.27 ± 0.10
20	56.71	56.31 ± 0.32	64.47	64.22 ± 0.41
40	67.48	67.24 ± 0.60	88.75	88.78 ± 0.82

The paired permutation test shows that the mean difference between the costs of the two policies is statistically significant (with $p_{\text{value}} < 0.0008$) in all scenarios where the *MDP* policy achieves a lower cost. Conversely, for parameter sets where the *MDP* policy has an equal or higher cost compared to the *A* mod* or *RAGS* policies, the difference is, as expected, not significant ($p_{\text{value}} = 1$).

VII. VALIDATIONS WITH MODEL PERTURBATIONS

Until now, our algorithm has been tested in ideal scenarios where we have complete knowledge of the underlying model. However, in real-world applications, this is not the case. In this discussion, we explore the robustness of our approach in determining the optimal strategy to achieve the goal when the parameters used to define the model are incorrect.

A. Experimental evaluation of the real-world delays

To validate our approach using a physically meaningful metric, we conducted experiments to measure the real-world delays caused by human encounters.

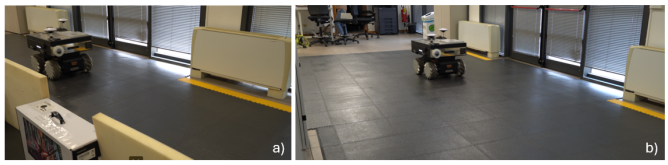


Fig. 13. Data collection setting for the time needed to traverse the corridor when no humans are present. **a)** Narrow corridor, **b)** Wide corridor.

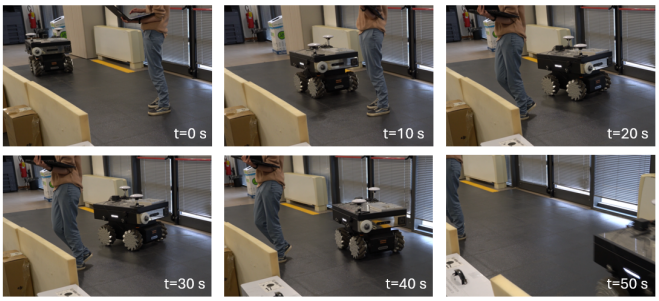


Fig. 14. Single data collection run in the narrow corridor when a person is present. The person needs to move to make the robot pass.

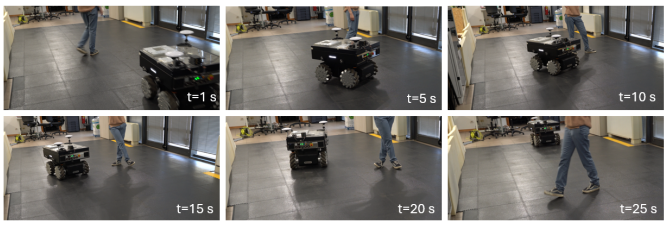


Fig. 15. Single data collection run in the wide corridor when a person is present.

Experimental Setup: We chose as a mobile robot the Robotnik XL-Steel, equipped with two SICK s300 lidars [5], having $w_r = 0.776\,m$. The robot software architecture is based on ROS Melodic. To collect data, we consider two corridors of different widths. The first corridor is narrower $w_c \approx 2.25\,m$, and the second one is wider $w_c \approx 3.75\,m$. Both corridors have length $l_e \approx 7\,m$ To detect the presence of people in the environment, we rely on the open-source package `obstacle_detector`[6] [34], which uses LiDAR data to detect distant obstacles. Although not explicitly designed for human detection, this method allows us to identify people by selecting objects with a radius compatible with that of a human body (we consider in this scenario that the only obstacles not known offline are people). Additionally, we exclude static obstacles by rejecting all detections that correspond to occupied areas in the known offline map of the environment. As a local planner, we use TEB (Timed Elastic Band) in the `move_base` framework, with a custom velocity scaling module to ensure adaptive and safe behavior near obstacles. We designed such velocity scaling according to the following specifications:

- The Euclidean distance from the robot to each human is continuously monitored.
- **Fast mode** when no humans are detected ($v_{\max} = 0.9\,m/s$).
- **Slow mode** is triggered whenever an obstacle is closer than 2 m in front of the robot or closer than 1 m behind it ($v_{\max} = 0.1\,m/s$).
- **Medium mode** is triggered under the same conditions with extended thresholds of 3 m (front) and 1.5 m (back), only if no slow condition is active ($v_{\max} = 0.4\,m/s$).
- To prevent sudden re-acceleration due to momentary loss of detection (e.g., sensor dropout), hysteresis is applied: once the robot enters a slower mode, it remains in that mode if discontinuous jumps in the distance to the closest human are detected.

We performed 15 runs for the scenarios where the corridor is free (Fig. 13), and 50 runs (for each corridor) for the scenarios where a person is present in the homotopy class space (Fig. 15).

We present the results of our data collection in Fig. 16, which illustrates the distributions of the additional navigation time required in narrow and wide corridors when a human is present. Specifically, each data point represents the difference between the navigation time measured with a person present and the average navigation time observed in scenarios without human presence for that corridor. As expected, the wide corridor is associated with a lower average additional cost compared to the narrow corridor. However, both distributions have substantial variance and cannot be considered normal distributions. This behavior occurs primarily because, in some scenarios, the robot becomes temporarily stuck (more frequently and severely in the narrow corridor) and consequently requires considerably more time to pass by the person. Moreover, the delay strongly depends on the human behaviour during the human-robot interaction.

[5]Robotnik xl-steel simulator, https://github.com/RobotnikAutomation/summit_xl_sim

[6]https://github.com/tysik/obstacle_detector

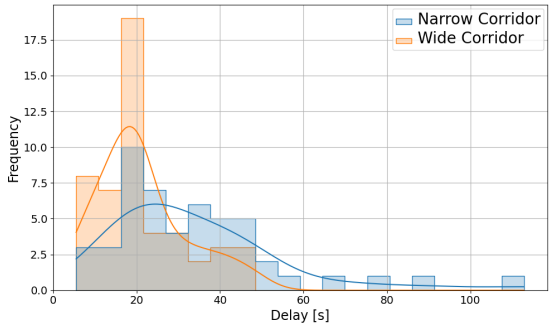


Fig. 16. Distributions of the additional navigation time required in narrow and wide corridors under human presence conditions. The distributions are the residuals between the time measured with a person and the average time without obstructions.

B. Perturbed Severity Coefficient (Delay)

As we discussed in the previous section, the actual value of the delay when a person is encountered is highly variable. To demonstrate that our approach remains valid even in such conditions, we conducted additional validations for the *Small* and *Medium* maps. We sampled a different delay from the collected data during each simulation encounter with a person. Table XI shows the results obtained using the experimental data instead of $c_p = 30$, for small map ($hum_{\text{info}} = (0, 2, 5), p_e = 0.005$), for medium map ($hum_{\text{info}} = (0, 2, 5), p_e = 0$). For each scenario, 20000 simulations are performed.
As can be seen from the table, even in the presence of high variance in delays (and therefore in the severity coefficient), the MDP maintains the policy that leads to the lowest average cost compared to the other planners evaluated.

C. Perturbed Humans Model

In this work, we assume prior knowledge about p_h, hum_{info}, and p_e. In real-world scenarios, these parameters could be estimated and updated over time. In this context, we discuss qualitatively the robustness of our method against potential inaccuracies or uncertainties in these parameters.

- p_h - *Human probability*: This parameter significantly impacts the optimal policy. In this paper, we deliberately adopted a uniform distribution of human presence throughout the environment, viewing it as a neutral and conservative initial assumption. Incorrect prior assumptions could result in suboptimal route selection. While more precise estimations could be achieved by learning patterns of human distribution over time (e.g., higher occupancy in specific zones at particular times), implementing such learning mechanisms is left to future work.
- hum_{info} - *Human Info*: Inaccuracies in hum_{info} could cause discrepancies between the expected and actual costs, but, at least in the scenarios we tested, they do not change the policy selected by the MDP. Therefore, the model would be quite robust against this parameter.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE XI
VALIDATION USING EXPERIMENTAL DELAYS INSTEAD OF FIXED $c_p = 30$, 20000 SIMULATIONS.

Map	Scenario	MDP	A*	A*-mod	RAGS $d_{\text{thr}} = 0.8$	RAGS $d_{\text{thr}} = 0.6$
Small Map	Narrow Corridors	**32.23 ± 0.31**	40.22 ± 0.42	34.47 ± 0.36	34.48 ± 0.36	40.13 ± 0.42
	Wide Corridors	**28.22 ± 0.19**	32.04 ± 0.25	28.62 ± 0.22	28.66 ± 0.22	32.11 ± 0.25
Medium Map	Narrow Corridors	**46.74 ± 0.17**	53.73 ± 0.34	48.25 ± 0.24	50.56 ± 0.29	53.76 ± 0.33
	Wide Corridors	**45.73 ± 0.12**	48.69 ± 0.20	46.18 ± 0.16	46.89 ± 0.18	48.74 ± 0.20

TABLE XII
COST COMPARISON OF PLANNERS WITH PERTURBED ENTERING PROBABILITY (p_e), 20000 SIMULATIONS.

Map	Assumed p_e	Actual p_e	MDP	A*	A* mod	RAGS $d_{\text{thr}} = 0.8$	RAGS $d_{\text{thr}} = 0.6$
Small	0.005	0.02	40.96 ± 0.34	46.04 ± 0.38	41.22 ± 0.35	41.25 ± 0.35	46.04 ± 0.38
Small	0.05	0.02	41.26 ± 0.35	45.82 ± 0.38	41.26 ± 0.35	42.62 ± 0.36	45.82 ± 0.38
Medium	0.005	0.02	89.73 ± 0.52	101.41 ± 0.57	92.26 ± 0.54	90.67 ± 0.54	101.41 ± 0.57
Medium	0.05	0.02	90.88 ± 0.52	101.23 ± 0.57	94.04 ± 0.55	91.04 ± 0.52	101.23 ± 0.57

- p_e - *Entering Probability* Moving humans entering corridors while the robot is already traversing them are treated as a stochastic disturbance in this paper. Our primary objective is not to accurately model human movements between homotopy classes, but rather to estimate the frequency of such disturbances. This estimation helps us determine the probability of encountering a human, even when the initial observation at the corridor entrance indicates no occupancy. Since our estimated value of p_e may be inaccurate, we qualitatively examine robustness by discussing the implications of both underestimating and overestimating this parameter. For simplicity, we assume p_e remains constant across all corridors.

To illustrate the sensitivity to perturbations in p_e, Table XII presents simulation results comparing the performance of our planners on the small and medium maps. Specifically, we assume planners use incorrect entering probabilities ($p_e = 0.005$ and $p_e = 0.05$), while in reality, the true value during execution is in both cases $p_e = 0.02$.

From Table XII, we can see that, even if the real entering probability is the same, the obtained costs for the same algorithm may vary. For example, for the small map, the MDP policy passes on the right of the big wall when estimated $p_e = 0.005$, and on the left when $p_e = 0.05$, behaving in this case, as A* mod. RAGS with $d_{\text{thr}} = 0.8$, instead, make the opposite decision, as a higher estimated p_e makes the observation in edge $(6, 7)$ less valuable, increasing the chances that the best path is one of the multiple paths on the right. For the medium map, instead, both RAGS $d_{\text{thr}} = 0.8$ and the MDP pass on the right side, resulting in comparable performance.

VIII. CONCLUSIONS AND FUTURE WORKS

In this paper, we have explored the advantages of considering the risk of performance degradation from unknown obstacles. We have formulated the problem as a stochastic shortest path with recourse problem and presented it as an augmented Markov Decision Process to identify the optimal policy with the lowest risk. Our approach incorporates prior knowledge of human probability distributions, visibility, and the severity of encounters based on space width constraints. We have evaluated it on diverse map sizes and compared it with a homotopy classes graph with an A* algorithm that minimizes expected cost. We have also implemented a modified A* planner that dynamically updates paths based on available information. Our approach consistently results in lower costs than reactive planners across various scenarios. We have finally compared it with the Risk-Aware Graph Search (RAGS) algorithm. Our approach performs comparably or better, requiring significantly less computational time compared to RAGS's initial sweep, especially on large maps.

Furthermore, we have integrated our decision-making framework with a local planner based on a modified Timed Elastic Band (TEB) approach, which reduces the robot's speed whenever the distance to the nearest person falls below a certain threshold. We have collected a dataset to quantify the actual delay introduced by this local planner, validating our assumptions regarding severity and robustness to the high variance present in the real-world replannings.

Future work activities will include learning human distributions from previous experiments to adjust the MDP matrices accordingly in the long run. Additionally, we plan to design a local planner explicitly aimed at minimizing the expected time needed for replanning by exploiting the full width of the corridor while complying with the ISO 15066 regulations, particularly the Speed and Separation Monitoring (SSM) requirements.

REFERENCES

[1] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, "Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics," *Annals of operations research*, vol. 308, no. 1, pp. 125–143, 2022.

[2] G. Fragapane, R. De Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *European Journal of Operational Research*, vol. 294, no. 2, pp. 405–426, 2021.

[3] A. Ahmed, B. Kayis, and S. Amornsawadwatana, "A review of techniques for risk management in projects," *Benchmarking: An Int. Journal*, 2007.

[4] I. O. for Standardization, "Robots and robotic devices – collaborative robots," Geneva, CH, Standard ISO/TS 15066:2016, 2016.

[5] M. Rubagotti, I. Tusseyeva, S. Baltabayeva, D. Summers, and A. Sandygulova, "Perceived safety in physical human–robot interaction—a survey," *Robotics and Autonomous Systems*, vol. 151, p. 104047, 2022.

[6] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using voronoi diagram and fast marching," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2376–2381.

[7] D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using rrt," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 1429–1434.

[8] C. Tonola, M. Faroni, M. Beschi, and N. Pedrocchi, "Anytime informed multi-path replanning strategy for complex environments," *IEEE Access*, vol. 11, pp. 4105–4116, 2023.

[9] A. Bar-Noy and B. Schieber, "The canadian traveller problem," in *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*. Citeseer, 1991, pp. 261–270.

[10] G. H. Polychronopoulos and J. N. Tsitsiklis, "Stochastic shortest path problems with recourse," *Networks: An International Journal*, vol. 27, no. 2, pp. 133–143, 1996.

[11] J. Chu, F. Zhao, S. Bakshi, Z. Yan, and D. Chen, "Risk-aware path planning with uncertain human interactions," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 4225–4230.

[12] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[13] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes, "Risk-aware motion planning in partially known environments," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 5220–5226.

[14] K. Cai, C. Wang, S. Song, H. Chen, and M. Q.-H. Meng, "Risk-aware path planning under uncertainty in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–15, 2021.

[15] S. Primatesta, G. Guglieri, and A. Rizzo, "A risk-aware path planning strategy for uavs in urban environments," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 629–643, 2019.

[16] A. A. Pereira, J. Binney, G. A. Hollinger, and G. S. Sukhatme, "Risk-aware path planning for autonomous underwater vehicles using predictive ocean models," *Journal of Field Robotics*, vol. 30, no. 5, pp. 741–762, 2013.

[17] H. Guo and T. D. Barfoot, "The robust canadian traveler problem applied to robot routing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5523–5529.

[18] L. Nardi and C. Stachniss, "Long-term robot navigation in indoor environments estimating patterns in traversability changes," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 300–306.

[19] C. Cao, X. Xu, X. Gong, B. Lu, W. Chi, and L. Sun, "Anmip: Adaptive navigation based on mutual information perception in uncertain environments," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2023, pp. 1–7.

[20] F. Tsang, T. Walker, R. A. MacDonald, A. Sadeghi, and S. L. Smith, "Lamp: Learning a motion policy to repeatedly navigate in an uncertain environment," *IEEE transactions on robotics*, vol. 38, no. 3, pp. 1638–1652, 2021.

[21] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 182–195, 2019.

[22] S. Gao and I. Chabini, "Optimal routing policy problems in stochastic time-dependent networks," *Transportation Research Part B: Methodological*, vol. 40, no. 2, pp. 93–122, 2006.

[23] S. Kim, M. E. Lewis, and C. C. White, "Optimal vehicle routing with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178–188, 2005.

[24] A. Asahara, K. Maruyama, A. Sato, and K. Seto, "Pedestrian-movement prediction based on mixed markov-chain model," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2011, pp. 25–33.

[25] S. Feyzabadi and S. Carpin, "Risk-aware path planning using hirerachical constrained markov decision processes," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2014, pp. 297–303.

[26] M. Werner and S. Feld, "Homotopy and alternative routes in indoor navigation scenarios," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014, pp. 230–238.

[27] L. Palmieri, A. Rudenko, and K. O. Arras, "A fast random walk approach to find diverse paths for robot navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 269–276, 2016.

[28] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[29] K. Kolur, S. Chintalapudi, B. Boots, and M. Mukadam, "Online motion planning over multiple homotopy classes with gaussian process inference," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2358–2364.

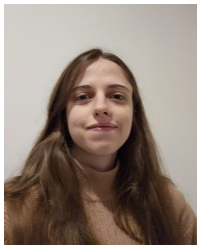
[30] H. Frank, "Shortest paths in probabilistic graphs," *operations research*, vol. 17, no. 4, pp. 583–599, 1969.

[31] J. S. Provan, "A polynomial-time algorithm to find shortest paths with recourse," *Networks: An International Journal*, vol. 41, no. 2, pp. 115–125, 2003.

[32] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[33] J. Lee, T.-W. Kang, Y.-S. Choi, and J.-W. Jung, "Clearance-based performance-efficient path planning using generalized voronoi diagram," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 23, no. 3, pp. 259–269, 2023.

[34] M. Przybyła, "Detection and tracking of 2d geometric obstacles from lrf data," in *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2017, pp. 135–141.



Elena Stracca received the M.S. degree in robotics and automation engineering from the University of Pisa, Pisa, Italy, in 2022. She is currently a Ph.D. student in information engineering at the University of Pisa, Pisa, Italy. Her research interests include risk-aware motion planning and decision-making under uncertainty.

Giorgio Grioli (Senior Member, IEEE) is a Assistant Professor at Università di Pisa (Pisa, Italy) and Visiting Scientist at Istituto Italiano di Tecnologia (Genova, Italy), where he investigates the design, modeling, and control of soft robotics systems applied to augmentation of, rehabilitation of and interaction with the human. His research interest span from collaborative industrial robotics to prosthetics.

Lucia Pallottino (Senior Member, IEEE) received the "Laurea" degree in mathematics and the doctoral degree in robotics and industrial automation from the University of Pisa. She is currently Professor with the Centro di Ricerca "E. Piaggio" and the Dipartimento di Ingegneria dell'Informazione, University of Pisa. Her main research interests include motion planning and control, optimal control, coordination of multirobot systems, and distributed algorithms.

Paolo Salaris earned his Ph.D. in automation and bioengineering from the University of Pisa in 2011. From 2015 to 2019, he was a Permanent Researcher at Inria Sophia Antipolis Mediterranee in France. From 2019 to 2022, he was an Assistant Professor at the University of Pisa, and since 2022, he's an Associate Professor. His research interests include optimal motion planning and control, active sensing, sensor placement, estimation, and multirobot and nonholonomic systems.