Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Path Planning for Multiple Agents Under Uncertainty

Glenn Wagner, Howie Choset
Carnegie Mellon University
500 Forbes Ave, Pittsburgh PA 15213

Abstract

Multi-agent systems in cluttered environments require path planning that not only prevents collisions with static obstacles, but also safely coordinates the motion of many agents. The challenge of multi-agent path finding becomes even more difficult when the agents experience uncertainty in their pose. In this work, we develop a multi-agent path planner that considers uncertainty, called uncertainty M* (UM*), which is based on a prior multi-agent path approach called M*. UM* plans a path through the belief space for each individual agent and then uses a strategy similar to M* that coordinates only agents that are "likely" to collide. This approach has the same scalability advantages as M*. We then introduce an extension called Permuted UM* (PUM*) that uses randomized restarts to enhance performance. We finish by presenting a belief space representation appropriate for multi-agent path planning with uncertainty and validate the performance of UM* and PUM* in simulation and mixed-reality experiments.

Introduction

Many applications, such as robotic warehouses, must coordinate large numbers of agents simultaneously to carry out their specific tasks. In practice, robots will not execute planned paths exactly due to unmodeled dynamics as well as localization and sensing errors, leading to a need for plans that are robust to significant uncertainty in robot pose. In this paper, we address the *multi-agent path finding with uncertainty* (MAPFU) problem by combining the M* *multi-agent path finding* (MAPF) algorithm with belief space planning (Bry and Roy 2011)(Gonzalez and Stentz 2005)(Patil, Van Den Berg, and Alterovitz 2012)(Prentice and Roy 2009)(Sun and Torres 2013) to produce uncertainty M* (UM*).

Belief space planning functions by first choosing a simple, parameterized representation of the probability distribution of the system's configuration, often a Gaussian for single agent systems, and then computing a single trajectory through the resulting belief space. At the intuitive level, belief space planning can be thought of as computing a single trajectory for the nominal configuration of the system, e.g., the center of a Gaussian distribution, while dilating the size of the agents by the uncertainty in their configuration. We believe that applying belief space planning to the MAPFU problem will allow for path planning that scales to large number of agents in cluttered environments.

We start by analyzing the MAPFU problem and comparing it to the MAPF problem, which does not consider uncertainty. The analysis shows that the MAPFU is broadly similar to the MAPF problem, but couples the dynamics of its constituent agents more tightly. As a result, MAPF algorithms can be adapted to the MAPFU problem but will lose guarantees of completeness and optimality with respect to path cost. We then present UM*, a variant of the M* MAPF algorithm (Wagner and Choset 2011) that uses belief space planning to solve the MAPFU problem and an extension of UM* that utilizes random restarts to enhance performance called *permuted UM** (PUM*). We test UM* and PUM* on simulated agents moving on a 4-connected grid to determine how their performance scales with increasing number of robots, and compare those results to those of two alternate approaches for solving the MAPFU. We then use UM* to generate plans for a mixed-reality system of real and simulated robots. The resulting plans can be safely executed by the robots, whereas following plans generated by a MAPF algorithm that ignores uncertainty led to collisions between agents.

Prior Work

Several works have successfully solved the MAPFU problem using *Partially Observable Markov Decision Processes* (POMDPs) (Ulusoy et al. 2012)(Miller, Harris, and Chong 2009), generating a control policy over all possible beliefs as to the configuration of the system. However, doing so for more than a small number of agents is computationally intractable (Papadimitriou and Tsitsiklis 1987). Therefore the community has developed a number of MAPFU algorithms that seek to approximate the full POMDP formulation while minimizing computational cost. Three basic approaches to the MAPFU problem are dynamic replanning (Ferguson, Kalra, and Stentz 2006)(Likhachev et al. 2005)(Stentz 1993), interaction regions (Dresner and Stone 2008)(Ferrari et al. 1998)(Hoenig et al. 2016)(Melo and Veloso 2009), and belief space planning (Bry and Roy 2011)(Gonzalez and Stentz 2005)(Patil, Van Den Berg, and Alterovitz 2012)(Prentice and Roy 2009)(Sun and Torres 2013).

Dynamic replanning approaches operate by computing

a deterministic plan, taking a step along the plan, then observing the result and recomputing the plan. These approaches work well when the effects of uncertainty are small over short periods of time. Examples include D* and other algorithms based on *Rapidly-Exploring Random Trees* (RRTs) (Ferguson, Kalra, and Stentz 2006)(Likhachev et al. 2005)(Stentz 1993) that reuse work done in previous planning steps to allow for rapid, global replanning for single agents in an unknown or partially known world. Bruce and Veloso (Bruce and Veloso 2005) applied a replanning approach to multi-agent systems, combining separate global path planning with one-step look-ahead velocity selection. Dynamic replanning approaches will fail if the uncertainty in single actions is significant or if continuous observation and replanning during plan execution is not feasible, and require significant inter-agent communication due to frequent replanning.

Reactive approaches, such as reciprocal velocity obstacles (Van Berg, Lin, and Manocha 2008)(Hennes, Claes, and Tuyls 2012), are similar to dynamic replanning, but use reactive controllers to mediate behavior instead of repeated planning. Using reactive controllers allows for higher update rates and requires less communication. Reactive approaches work well in open environments, where resolving a single conflict between a pair of agents is relatively easy. However, these approaches will fail or perform poorly in tight environments, such as those featuring long corridors wide enough for only a single agent, as preventing conflicts in such environments require sustained, deliberate cooperation.

An alternate approach specific to multi-agent systems with uncertainty is to identify a limited set of regions where agents may interact. In reservation based approaches, the planner reserves the interaction regions for specific agents, and requires other agents to wait safely outside the interaction region until the agent that has the region reserved has passed through (Dresner and Stone 2008)(Ferrari et al. 1998)(Hoenig et al. 2016). Reservation based approaches generally do not explicitly consider uncertainty in the paths of single agents, and may lead to agents waiting for prolonged periods of time if an agent with a reservation for an interaction region is delayed.

Melo and Veloso (Melo and Veloso 2009) exploited interaction regions to produce a simplified version of the full POMDP formulation, called a Decentralized Sparse-Interaction Markov Decision Process (DEC-SIMDP). An agent could only observe the state of other agents when both were in designated interaction regions, which were also the only regions where agent-agent interactions could lead to increased cost. As a result, the multi-agent POMDP was restricted to a small portion of the workspace, substantially reducing the computational cost of computing a policy. However, like reservation based systems, DEC-SIMDPs are only applicable to environments containing a low number of small, well defined interaction regions. This is most likely to be the case for structured environments like road networks, hallways, or warehouses, and unlikely to be true for cluttered environments

Belief space planning is commonly employed to solve the single agent path planning problem with uncertainty. A simple representation, such as a Gaussian, is chosen for the uncertainty in agent position and a conventional path planning algorithm such as A* or RRTs are used to compute a single trajectory through the resultant belief space (Bry and Roy 2011)(Gonzalez and Stentz 2005)(Patil, Van Den Berg, and Alterovitz 2012)(Prentice and Roy 2009)(Sun and Torres 2013). Work with non-Gaussian beliefs has also been done (Melchior and Simmons 2007)(Platt et al. 2012). Belief space planning is more efficient than solving the full POMDP because it computes a single nominal trajectory, rather than a control policy over all possible belief states. The only application of belief space planning to the MAPFU problem of which the authors are aware was done by van den Berg, Abbeel and Goldberg (van den Berg, Abbeel, and Goldberg 2011). They used a priority planner to perform belief space planning for a multi-agent system in an uncluttered environment with relatively small uncertainty.

Problem Definition

The objective of the MAPFU problem is to find a minimal cost path from some initial distribution of agent states b_s to one of a set of goal distributions b_f sufficiently concentrated around a goal configuration, while satisfying a safety condition. Let the system consist of n agents $r^i, i \in I = \{1, \ldots, n\}$ where each agent r^i has a discrete configuration space Q^i and an action set $\mathcal{A}^i$. The configuration space Q^i of each agent is augmented with a special collision state $col^i \in Q^i$. If r^i collides with another agent, r^i is removed from the workspace and placed at col^i. The full system is described by a joint configuration space $Q = \prod_i Q^i$ and joint action set $\mathcal{A} = \prod_i \mathcal{A}^i$. Taking a joint action incurs a cost that is the sum of the costs of taking the individual agent actions.

The belief space $\mathcal{B}^i$ for a single agent is the space of belief states

$$\mathcal{B}^i = \left\{ b^i : Q^i \to \mathbb{R}^{\geq 0} \;\middle|\; \sum_{q^i \in Q^i} b^i(q^i) = 1 \right\} \qquad (1)$$

while the *joint belief space* $\mathcal{B}$ is the space of joint beliefs $b : Q \to \mathbb{R}^{\geq 0}$. The belief dynamics $\text{Dyn}^i : \mathcal{B}^i \times \mathcal{A}^i \to \mathcal{B}^i$ maps a belief for the state of r^i to the belief that results when r^i takes a given action. The belief dynamics of the full system $\text{Dyn} : \mathcal{B} \times \mathcal{A} \to \mathcal{B}$ are formed by applying the belief dynamics of the individual agents then checking for potential collisions.

The MAPFU problem can be formulated with multiple different safety conditions. Obvious choices include placing bounds on the probability that each robot is involved in a collision, the probability of any collision occurring, or on the total expected number of collisions. In this paper, we focus on the case where each agent is constrained to keep its probability of colliding with other agents below some threshold δ_{col}, i.e. $\forall r^i, b_f(col^i) < \delta_{\text{col}}$. A *constraint violation* occurs if the probability of a robot colliding ever exceeds δ_{col}. We chose this approach because it guarantees fairness, i.e. every robot will have at least some probability of completing its task. We also believe that plans where each robot has at

most some small probability of collision will be easier to 'repair' via later replanning than a plan where one or more robots are guaranteed to collide with other robots.

Structure of the MAPFU problem

This paper approaches the MAPFU problem by adapting algorithms developed to solve the MAPF problem to belief space planning. Efficient MAPF algorithms, particularly those that offer completeness guarantees and cost bounds (Sharon et al. 2012)(Wagner and Choset 2011), rely on a direct product structure possessed by MAPF problems that is not present in MAPFU problems. More specifically, the direct product structure of the MAPF problem means that both the configuration space and dynamics of the full system are the Cartesian product[1] of the single agent configuration spaces and dynamics. Furthermore, each agent is subject to a set of $n-1$ constraints requiring that it avoid collision with the other agents. Taken together, these facts imply that if a solution exists for a MAPF problem, then a solution exists for any subproblem created by selecting a subset of the agents. As a result, MAPF algorithms can efficiently solve large problems by combining solutions to constituent subproblems.

The MAPFU problem lacks the direct product structure, because the belief dynamics of the constituent agents are inherently coupled. Specifically, a small overlap in the belief state of two agents will result in the belief state of both robots being pruned, to remove the conflicting states, without necessarily causing a constraint violation. As a result, an agent can reach belief states as part of a system that it would be unable to reach on its own. In the formulation of the MAPFU problem described in the previous section, this means that it is possible to construct problems that have solutions, but which contain subproblems that do not have any solutions.

Consider the system of three agents in Fig. 1a where each agent has a single action available at each state. To simplify the example, we assume that actions are deterministic but collisions are stochastic, with a 20% chance of agents r^2 and r^3 colliding while at states e and h, and a 50% chance of a collision between r^1 and r^2 at states c and f respectively. As a result, the cumulative probability of r^2 colliding with another agent is 60%. Therefore, if the collision threshold is 59%, the problem has no solution.

Now consider adding a fourth agent r^4 (Fig. 1b) with two possible paths: a cheap path through k that entirely avoids all other agents and an expensive path through j that has a 20% chance of colliding with r^3. There is no solution if r^4 takes the cheap path. However, if r^4 takes the more expensive path then r^3 only has a 16% chance of colliding with r^2, as it may have collided with r^4 before reaching state h. As a result, the cumulative probability of r^2 colliding with another agent is only 58% which satisfies the collision threshold. The solution effectively trades slack in the safety constraint on r^4 to satisfy the constraint on r^2.

[1]If $f : A \to X$ and $g : B \to Y$, then $f \times g : A \times B \to X \times Y$ and $(f \times g)(a, b) \mapsto (f(a), g(b))$

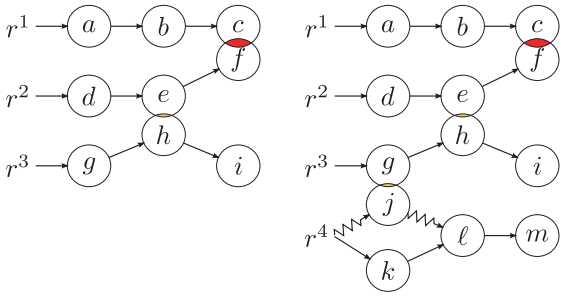


Figure 1: **(a)** A three agent problem with a collision threshold of $\delta_{\text{col}} = 0.59$. Each agent has a single action at each state. There is a 20% chance of a collision if states e and h are occupied simultaneously, and a 50% chance of collision if states c and f are occupied simultaneously. There is no solution, the probability of r^2 colliding is 0.6. In **(b)** a fourth agent is added with two paths: an expensive path passing through j and a cheap path passing through k. There is a 20% chance of collision if states g and j are occupied simultaneously. The only solution is when r^4 chooses the more expensive path, at which point the probability of r^2 colliding is 0.58 just below the threshold value.

Because slack in the constraints in one subproblem can be traded to satisfy constraints in a second subproblem, MAPF algorithms reliant on solving independent subproblems will loose theoretical guarantees on completeness and path quality when adapted to MAPFU problems. However, we note that incomplete methods have long been used successfully to solve interesting MAPF problems (Erdmann and Lozano-Perez 1986)(van den Berg, Abbeel, and Goldberg 2011)(Jansen and Sturtevant 2008). Combined with the fact that creating a solvable problem with unsolvable subproblems required carefully tuned values, we believe the incompleteness of MAPF algorithms adapted to MAPFU problems is unlikely to be problematic in practice.

The counterintuitive behavior displayed in Fig. 1, wherein additional conflicts can be added to produce a valid solution, is specific to the formulation of the MAPFU problem used in this paper and would not be present in alternate formulations that place a single, global constraint on the agents, such as a maximal number of expected collisions. However, a single, global constraint means that the full problem could not be cleanly split into subproblems for efficient solution by an adapted MAPF algorithm; some method of budgeting how many expected collisions each subproblem would be permitted would be necessary. Such work is out of the scope of the current paper, but we investigated an optimization approach inspired by (Stentz 2002) in (Wagner 2015) to solve the MAPFU problem subject to an expected number of collisions constraint.The resulting paths were similar to those found using the MAPFU formulation in this paper, but were much more computationally expensive to compute.

Uncertainty M*

In this section we introduce UM*, a variant of the M* MAPF algorithm (Wagner and Choset 2011) adapted to solve MAPFU problems by planning paths in the belief space of the system. UM* differs from M* primarily in exploring the joint belief space of the system, rather than the joint configuration space as does M*. UM* also replaces the structure that M* uses to track constraint violations, called the *collision set*, with the *conflict* and *coupled* sets to account for the shift from a hard no-collision constraint in M* to a constraint on likelihood that an agent collides with another agent in UM*.

UM* functions by alternating between running A* on a low-dimensional *search graph* and expanding the search graph to generate alternate paths around states found by A* that contain constraint violations. The search graph is initially constructed using the *individual policies* of the agents that describe the optimal path for each agent if no other agents were present. Every vertex v_k in the search graph maintains a conflict set C_k of agents that violate their collision probability constraints at some successor of v_k in the A* search tree and a coupled set Γ_k of agents that may be able to prevent the constraint violation by taking a different action v_k. The coupled set is then used to grow the search graph to provide alternate paths around constraint violations. We will first define the search graph before providing a detailed algorithmic description of UM*.

The search graph explored by UM* is a subgraph of a (possibly infinite) graph that represents the joint belief space called the *joint belief graph* $G = \{V, E\}$. Each vertex in the vertex set $v_k \in V$ represents a joint belief state $b_k \in \mathcal{B}$. Edges correspond to a subset of the feasible actions at b_k. As UM* finds constraint violations at successors of v_k, the set of actions considered at v_k grows, expanding the search space in a fashion that will be described in detail later in this section. An edge in the edge set $e_{k\ell} \in E$ connects v_k to the vertex v_ℓ that corresponds to the belief $\text{Dyn}(b_k, a_{k\ell})$.

With the search graph defined, we now will give a detailed description of UM*. UM* begins by computing a separate individual policy $\phi^i : \mathcal{B}^i \rightarrow \mathcal{A}^i$ for each agent r^i that describes the optimal action for the agent to take from any belief state. The individual policy computed by planning a path in the *belief graph* of r^i, which is analogous to the joint belief graph but for a single agent. In practice, the policy is computed in a lazy fashion.

Once the individual policies are initialized, UM* enters a standard A* search loop (Algorithm 1) to explore the search graph. The vertices in the open list are sorted by their *f-value*, $f = g + h$, where g is the current upper bound on the cost of the optimal path to a given vertex and h is a heuristic, computed by taking the sum of the cost-to-go of the individual policies.

When UM* expands a vertex v_k it only considers a subset of all possible actions $\mathcal{A}_k \subset \mathcal{A}$. If the conflict set C_k containing the agents that violate their collision probability constraints at a successor of v_k is empty then each agent takes the action determined by its individual policy. However if C_k is not empty, then to prevent the constraint violations UM* considers alternate actions for the agents in the coupled set

Algorithm 1 FindPath

```
1: function FINDPATH(b_s)
2:   open_list ← {VERTEXFROMBELIEF(b_s)}
3:   while open_list ≠ ∅ do
4:     v_k ← open_list.pop_best()
5:     if v_k.closed then
6:       continue
7:     v_k.closed ← True
8:     if ISGOAL(v_k) then
9:       return trace back_ptr to find optimal path to v_k
10:    for a ∈ A_k (Equation 2) do
11:      v_ℓ ← VERTEXFROMBELIEF(Dyn(b_k, a))
12:      BACKPROPAGATE(v_k, C_ℓ, Γ_ℓ open_list)
13:      v_ℓ.backpropagation_set.append(v_k)
14:      if v_ℓ.closed ∨ VIOLATESCONSTRAINT(v_ℓ) ∨ v_ℓ.g
     ≤ v_k.g + COST(a) then
15:        continue
16:      v_ℓ.g ← v_k.g + COST(a)
17:      v_ℓ.back_ptr = v_k
18:      open_list.insert(v_ℓ)
19:  return No Solution
```

Algorithm 2 Backpropagate constraint violations

```
1: function BACKPROPAGATE(v_k, C_ℓ, Γ_ℓ,open_list)
2:   temp ← Γ_k
3:   C_k ← C_k ∪ C_ℓ
4:   Γ_k ← Γ_k ∪ C_k ∪_{r^i ∈ C_ℓ} ζ(r^i, e_kℓ)
5:   if temp ≠ Γ_k then
6:     v_k.closed = False
7:     open_list.insert(v_k)
8:     for v_m ∈ v_k.backpropagation_set do
9:       BACKPROPAGATE(v_m, C_k, Γ_k, open_list)
```

Γ_k

$$\mathcal{A}_k = \prod_i \begin{cases} \mathcal{A}^i & r^i \in \Gamma_k \\ \phi^i\left(b_k^i\right) & r^i \notin \Gamma_k \end{cases} \tag{2}$$

where Γ_k is defined as the agents that are either in C_k or have a non-zero probability of colliding with an agent in C_k at a successor of v_k. Given the our formulation of the MAPFU problem, resolving the constraint violations given by C_k could potentially require any agent to change its action. However, Γ_k contains the agents which directly contribute to the known constraint violation and thus will have the strongest impact in preventing the violations. As described in the section on the structure of the MAPFU problem, the resulting algorithm will be incomplete, but should be able to solve most interesting problems. The computation of C_k and Γ_k is described in detail later.

For each action $a_{k\ell} \in \mathcal{A}_k$, UM* constructs the vertex v_ℓ corresponding to the belief $\text{Dyn}(b_k, a_{k\ell})$. UM* marks v_k as a predecessor of v_ℓ by adding v_k to the *backpropagation set* of v_ℓ. Then for each agent r^i, UM* finds the agents that may collide with r^i when the system traverses the edge $e_{k\ell}$ and stores the result in $\zeta\left(r^i, e_{k\ell}\right)$. If no agents violate their collision probability constraints at v_ℓ then the g-value of v_ℓ

is updated and v_ℓ is added to the open list, if appropriate.

If the conflict set of v_ℓ is non-empty UM* has found a new path to a constraint violation and needs to update coupled sets of v_k and all predecessor states (Algorithm 2). First UM* adds C_ℓ to C_k. Then C_k is added to Γ_k. Finally, for every $r^i \in C_\ell$, UM* adds $\zeta\left(r^i, e_{k\ell}\right)$ to Γ_k. If Γ_k changes then $\mathcal{A}_k$ is also changed, in which case v_k is added back to the open list so the new actions can be explored. Because agents are never removed from Γ_k, v_k will be returned to the open list at most n times, avoiding infinite loops. Then the process is repeated recursively to update the conflict and coupled sets of the predecessors of v_k, which are stored in the backpropagation set of v_k.

The search loop continues until UM* expands a vertex which corresponds to a goal belief, at which point UM* can reconstruct the path.

Algorithmic Improvements

In this paper we made use of several enhancements to the basic UM* algorithm. The first is that we replaced A* with a variant specialized for multi-agent path planning called Operator Decomposition (OD) (Standley 2010). In the second enhancement, UM* was modified to use the recursive framework developed for recursive M* (rM*) (Wagner and Choset 2011) that breaks the full problem into separate subproblems. The conflict set and coupled set are broken into elements that contain disjoint sets of mutually interacting agents. Thus if at some successors of v_k r^1 has some chance of colliding with r^2 which may in turn collide with r^3, while r^4 may collide with r^5, the coupled set in the recursive formulation of v_k would be $\{\{r^1, r^2, r^3\}, \{r^4, r^5\}\}$, containing two disjoint elements, rather than $\{r^1, r^2, r^3, r^4, r^5\}$. When UM* generates $\mathcal{A}_k$ it computes a single, optimal joint action for each coupled set element by calling UM* on just the agents in the coupled set element. Effectively UM* treats the coupled set element as a temporary meta-agent whose policy is computed with a recursive call to UM* If a single coupled set element contains all the agents in the subproblem then UM* considers all possible joint actions, forming the base case of the recursion.

We also inflated the heuristic h by $\epsilon > 1$. Doing so greatly reduces planning time, but increases path cost by up to a factor of ϵ.

Finally, we examined utilizing randomized restarts. UM* is a deterministic algorithm, but the order in which vertices with the same f-value are expanded, called tie-breaking, is arbitrary. We have found that the performance of M*, and by inference UM*, on any given problem is sensitive to small changes in tie-breaking. Furthermore, we have found that M* usually either finds a solution quickly or times out. As a result, we believe that the distribution of UM* planning time for a given problem over tie-breaking strategies has a peak at a relatively small duration followed by a long tail. Under these conditions, it can be more efficient to divide the available time into multiple instances with the planner using different strategies in each instance (Luby, Sinclair, and Zuckerman 1993) than to run a single strategy for all the available time. Cohen *et al* (Cohen et al. 2016) successfully applied randomized restarts to a deterministic MAPF algorithm by splitting the available planning time into equal instances and using a different permutation of agent order in each instance. Changing the agent order only changed the planner's internal representation, i.e. swapping the labels of r^1 and r^4, and did not change an agent's initial or goal location. Instances were solved sequentially until a solution was found. We applied the same strategy to UM* to produce *permuted UM** (PUM*).

Belief Representation for MAPFU

In most of the work on single agent belief space planning, the belief state of the system is represented by Gaussian distributions (van den Berg, Abbeel, and Goldberg 2011)(Bry and Roy 2011)(Gonzalez and Stentz 2005)(Patil, Van Den Berg, and Alterovitz 2012)(Prentice and Roy 2009)(Sun and Torres 2013) to reflect imperfect localization. Gaussian distributions could be used with UM* but cannot represent low probability conflicts between robots with high fidelity. Furthermore, we are interested in problems where the individual agents are highly capable and the primary challenge arises from the lack of synchronization between robots, for which Gaussian distributions are inappropriate. Therefore, we model the belief state of each agent as a distribution over the position of the agent along the agent's intended trajectory (Fig. 2). Because the belief distribution conforms to the planned trajectory of the agent, planning can be done even when the support of the belief distribution is comparable in size to or larger than the scale of the features in the environment.

Allowing agents to run both ahead and behind schedule would case problems during planning, as the belief state for the agents running ahead of schedule would depend upon parts of the path that have not yet been computed. We therefore assume that the nominal behavior of the agent is to move as quickly as possible, with all uncertainty being modeled as delays. As a result, the belief distribution of the agent is fully defined throughout planning.

Ideally, the agents would have a continuous distribution in possible velocities, but this is impractical when using a graph-based representation of the agent configuration spaces. Instead, each agent is modeled as delaying at its current location instead of taking its planned action with probability P_{delay}. During execution, each agent counted the number of unplanned delay actions it took, and would subsequently skip an equal number of planned delay actions. Thus planned delay actions served as an indirect synchronization action.

The belief state b_k^i of r^i was represented by two sequences: $\text{pos}_k^i : \{1, \ldots, m\} \rightarrow V^i$ and $\text{prob}_k^i : \{1, \ldots, m\} \rightarrow [0, 1]$. pos_k^i was the planned trajectory for r^i consisting of a sequence of grid cells up to the current position $\text{pos}_k^i(1)$, and prob_k^i contained the probabilities that r^i occupied each state in pos_k^i. The probability that r^i had collided with another robot was given implicitly by $b_k^i(col^i) = 1 - \sum_{j=1}^{m} \text{prob}_k^i(j)$. Tracking the full joint probability would be computationally impractical, so the belief distributions for each agent were assumed to be independent.

As defined, the support of the belief of r^i would cover

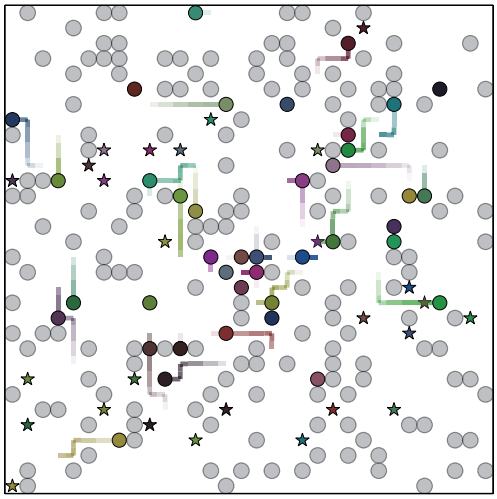


Figure 2: Typical step in a 40-agent plan computed by UM*. The gray circles are obstacles. Colored stars denote the goal configuration of agents. The colored bars represent the planning-time belief state as to where the agents would be. The color of more probable states are more saturated. The colored circles denote the actual position of the agents in one realization of the plan.

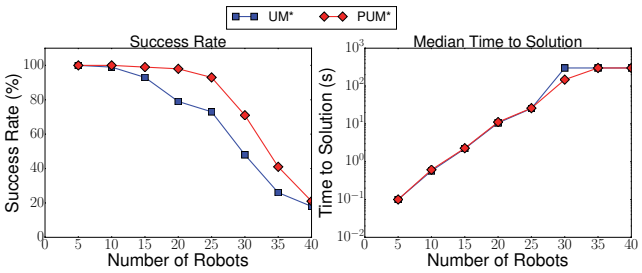


Figure 3: Comparison of performance of UM* and PUM* using $P_{\text{delay}} = 0.1$, $\delta_{\text{col}} = 0.1$, and $\epsilon = 3$. PUM* split the available time into six separate instances. The left figure gives the percentage of trials that were solved successfully in under five minutes and the right plot gives the median time to find solutions.

the entire path taken by r^i, even though the probability mass at many states would be infinitesimal. Therefore, states at the front and back of the distribution were removed if the probability mass at those states were less than some threshold value P_{prune}. After states are pruned, the remaining belief was re-normalized to preserve the total probability mass of the belief. For the simulated experiments, a value of $P_{\text{prune}} = 0.001$ was found to be appropriate.

Results

We ran two sets of experiments. The first was a simulation study in a discrete world with agents whose dynamics exactly matched those assumed in the previous section. We then conducted a set of experiments in a mixed-reality simulation which combined physical robots with simulated robots tuned to match the continuous time and space dynamics of the physical robots. Simulated robots were employed because the available lab space was insufficient to accommodate enough physical robots. The mixed-reality tests also served to validate that the belief representation described in the previous section reasonably models robots with continuous dynamics and velocity distributions.

Simulation Results

UM* was tested on randomly generated worlds, where each trial took place in a 32x32 four-connected grid of cells, with a 20% probability of a given cell being marked as an obstacle (Fig. 2). Unique initial and goal positions for between 5 and 40 agents were chosen randomly within the same connected component of the workspace. Any action by an individual agent, including waiting, incurred a cost of one, although an agent could wait at its assigned goal with zero cost. Agent actions where discrete; no kinematics were simulated. Unless otherwise stated, $P_{\text{delay}} = 0.1$, $\delta_{\text{col}} = 0.1$, and the heuristics used by M* and UM* were inflated by a factor of $\epsilon = 3$.

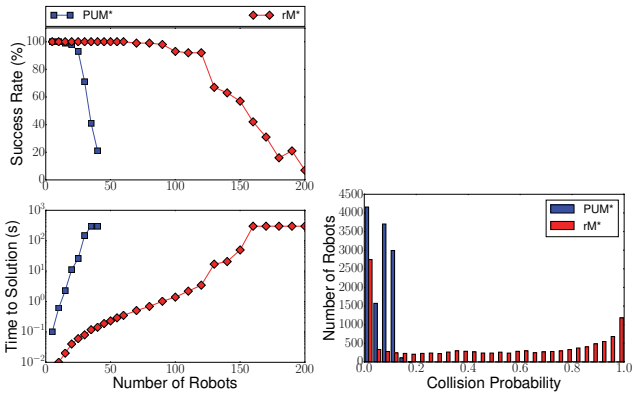
Each trial was given 5 minutes to find a solution. 100 random environments were tested for a given number of agents. We present the percentage of trials that were successful within 5 minutes as well as the median time required to find solutions. Run time is plotted on a logarithmic scale.

We start by comparing UM* to PUM*, where PUM* splits the five available minutes for planning into six instance each of which use a different permutation of agent order (Fig. 3). PUM* shows noticeably improved performance compared to UM* for between 15 and 35 agents. We thus use PUM* for the rest of the results on the grid.

We then compared PUM* to two alternative approaches to solving the MAPFU problem: running rM* without accounting for uncertainty and belief space planning based on priority planners (Erdmann and Lozano-Perez 1986).

PUM* was first compared to rM* which did not account for unplanned delays (Fig. 4). Not accounting for uncertainty allowed rM* to solve much larger problems than PUM*, achieving similar success rates on problems of 180 agents as PUM* did on problems with 40 agents (Fig. 4a). However, the plans computed by rM* were not safe, with a significant number of agents colliding with other agents with probability near one (Fig. 4b). In contrast, the probability of a robot colliding while following a PUM* paths is bounded by the collision threshold to within random noise from using a limited number of simulations. Thus accounting for uncertainty is expensive, but necessary to maintain safety.

We believe that there are two major factors that increased the difficulty of planning with uncertainty compared to ignoring uncertainty. Tracking a belief distribution for each robot effectively causes each robot to occupy a larger portion of the work space, increasing the effective robot density. This will in turn increase the number potential conflicts between agents the planner has to resolve. The second issue is that the belief distribution trails behind the nominal position of each agent. As a result, constraint violations can become inevitable a number of time steps before they actually occur.



(a) Planning performance (b) Collision probability during execution

Figure 4: Comparison between PUM* and rM*, where rM* ignores the probabilistic dynamics. Each agent has a 10% chance of delaying rather than taking its planned action. The heuristics of PUM* and rM* are inflated by 3. **(a)** The percentage of trials that were solved successfully in under five minutes are plotted on the left and the median time to find solutions are plotted on the right. **(b)** Every trial solved by both PUM* and rM* was executed 100 times to compute the collision probability of each agent, and are plotted in a histogram on a per-agent basis.

This results in deep local minima that are computationally expensive to escape.

A priority planner is a MAPF algorithm (Erdmann and Lozano-Perez 1986) wherein each agent is assigned a priority. Paths are then computed for the agents one at a time in declining order of priority with low priority agents treating high priority agents as moving obstacles. Priority planners can generally find solutions to the MAPF problem quickly, but are not guaranteed to find any path, much less the optimal path. van den Berg, Abbeel and Goldberg (van den Berg, Abbeel, and Goldberg 2011) have previously applied priority planners to the MAPFU problem using belief space planning. Plans for individual agents were computed using RRT in an environment devoid of static obstacles. Individual

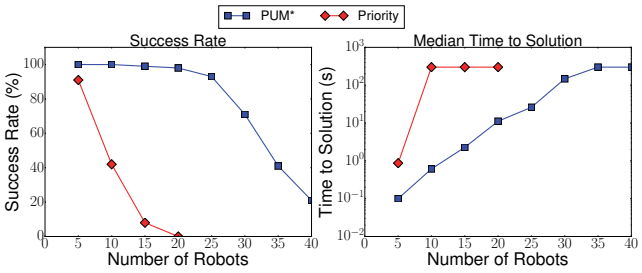
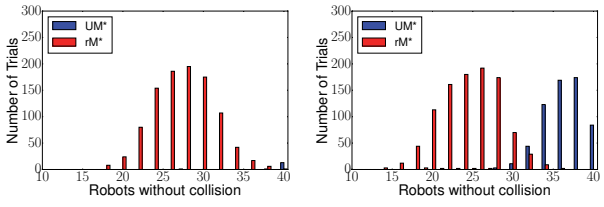


Figure 5: Comparison of performance of PUM* and belief space priority planning. All trials were run with $P_{\text{delay}} = 0.1$, $\delta_{\text{col}} = 0.1$, and $\epsilon = 3$. The left figure gives the percentage of trials that were solved successfully in under five minutes and the right plot gives the median time to find solutions.



(a) Planning horizon: 4 steps (b) Planning horizon: 10 steps

Figure 6: Number of robots that reach their goals without collision with receding horizon planning. Trials that timed out are not counted.

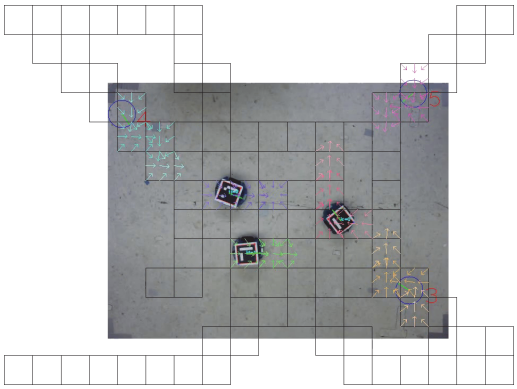
agents were able to localize using static beacons, resulting in belief distributions which were small compared to those encountered in our test cases (Fig. 2).

We used a different version of priority planning in the belief space dictated by our discrete representation of the world. Individual agent plans were computed using A* with a heuristic inflated by a factor of 3. The resulting plans had to satisfy the collision probability constraints for the agent being planned for as well as all higher priority agents. Agents were assigned priority in decreasing order of the cost of their paths in the absence of other agents. If the planner determined that the priority order prevented a solution from being found, which occurred in only 3 trials, the agent that was unable to find a solution was set as the highest priority agent and planning was restarted. If doing so caused a previously considered priority ordering to be revisited then the priority order was randomly permuted.

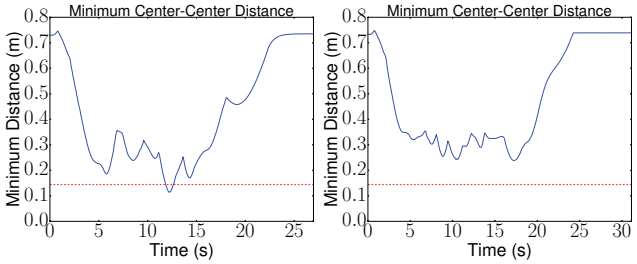
The priority planner was dramatically outperformed by PUM* (Fig. 5). We believe there are several reasons for the poor performance of the priority planner. First of all, the belief space is very large, which means that the priority planner is unlikely to determine whether a given priority order excludes a solution. Secondly, the priority planner is prone to finding constraint violations far from where action must be taken to resolve said constraint. Let r^1 and r^2 be high priority agents that have a low probability of colliding late in their path. If r^3 encounters r^2 early in its path then the increased probability of colliding may cause r^2 to violate its constraint when it encounters r^1 much later. Because the priority planner can only alter the path of r^3 it may have to back track a long way and explore a large fraction of the belief space of r^3 to find a path for r^3 that avoids r^2. Depending upon the inflation factor used, PUM* would have the option of altering the path of r^2 to avoid interacting with r^1 when it finds the constraint violation, which would require less backtracking and thus be much faster.

Finally, we explored receding horizon approaches that replan regularly out to a limited depth, thus allowing knowledge of robot location during the run to be utilized. In the implementation described here, the agents executed half of their computed plan before replanning. Therefore, if planning was conducted to a depth of 4 actions, then the agents would execute two actions before replanning.

Receding horizon planning was tested on 100 randomly generated problems containing 40 robots, with each trial run



(a) Snapshot of experiment



(b) rM* (c) UM*

Figure 7: **(a)** Snapshot of UM* experiment execution. The boxes are the grid cells over which control policies in the plan for at least one robot were defined. Circles denote simulated robots. The colored arrows depict the control policies in the current planned belief for each robot; more saturated policies are more probable. We compare the minimum distance between the centers of the robots while executing plans computed using rM*, neglecting uncertainty **(b)**, and UM* **(c)**. The red dashed lines mark 1 robot diameter; distances below this value indicate collision.

10 times, using both rM* and UM* as planners. Each trial was given 30 seconds total of planning time. If all robots did not reach their goals before the budgeted planning time was exhausted the trial was counted as a failure. UM* was run with the collision threshold set to $\delta_{\text{col}} = 0.075$ for each planning iteration. In all other respects the environments were the same as in other trials.

For short planning horizons (Fig. 6a) UM* was prone to live locks that caused many time outs. When using longer planning horizons, UM* produced much safer paths than rM* (Fig. 6b), confirming the benefits of accounting for uncertainty in planning.

Experimental Results

We then validated UM* in a mixed-reality simulation featuring three physical Scribbler robots and three simulated robots with kinematics tuned to match the Scribblers. Localization was provided by an overhead camera. The workspace was divided into a grid of rectangular cells. Five control policies were defined over each grid cell; one to wait in place and one continuous piecewise-linear control policy to move the robot out of the cell through each edge of the cell (Fig. 7a). Calibration runs indicated that the robots could execute the non-wait motion primitives in a minimum of 0.6 seconds and an average of 0.9 seconds. A delay probability of 0.3 was chosen to match the distribution. Execution of plans was fully decentralized. Each robot was given a path in the form of a sequence of beliefs over control policies. A robot tracked its progress through the plan by selecting either the current belief in the plan or its successor depending upon which had greater probability density at the robot's current position. The robot then executed the most probable control policy from that belief. Control policy selection and the control policies themselves ran at approximately 10 Hz.

We tested plans computed using rM* (neglecting uncertainty) and UM*. The minimal center-center distance between robots over the course of a run is plotted in Fig. 7b for rM* and Fig 7c for UM*. The distance dropped below one robot diameter (the red dashed line) when following the rM* plan as a result of a simulated robot running over a physical robot. In contrast, the robots were able to safely execute the UM* plan, demonstrating its benefit for a physical system.

Conclusions

In this paper, we contributed to the literature on MAPFU problems, leveraging our previous MAPF work. The MAPF problem has a direct product structure on which many efficient MAPF algorithms rely, particularly those that have optimality and completeness guarantees. We started by showing that the MAPFU problem does not have the direct product structure. As a result, while MAPF algorithms that rely on the direct product structure can be adapted to the MAPFU problem, they will lose any guarantees of completeness or optimality. We then presented UM*, a variant of rM* for the MAPFU problem and an extension, PUM* that leverages randomized restarts to improve performance. We then introduced a non-Gaussian belief space representation that is appropriate for multi-agent systems when individual agents can localize themselves well, but have little ability to synchronize their actions with other agents. We then showed that PUM* outperforms alternate approaches to the MAPFU problem in simulation and mixed-reality tests. We also showed that UM* was beneficial in a receding horizon framework for longer planning horizons.

The improvement shown by PUM* over UM* was impressive for such a small change. However, the randomization in tie-breaking by permuting the agents is indirect and coarse. In particular, all instances used the same individual policies. We will explore whether applying vertex-level randomized tie-breaking at both the level of UM* and the individual policies would further enhance the performance of UM*.

Acknowledgments This work was supported by ONR Subcontract to the Applied Physics Lab entitled "Autonomous Unmanned Vehicles Applied Research Program" Prime Contract Number N00024-13-D-6400

References

Bruce, J. R., and Veloso, M. 2005. Real-time multi-robot motion planning with safe dynamics. In *Multi-Robot Systems. From Swarms to Intelligent Automata*, volume III. 1–12.

Bry, A., and Roy, N. 2011. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, 723–730.

Cohen, L.; Uras, T.; Kumar, T. K. S.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding. In *International Joint Conference on Artificial Intelligence*.

Dresner, K., and Stone, P. 2008. A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research* 31:591–653.

Erdmann, M., and Lozano-Perez, T. 1986. On multiple moving objects. In *IEEE International Conference on Robotics and Automation*, volume 3, 1419–1424. Institute of Electrical and Electronics Engineers.

Ferguson, D.; Kalra, N.; and Stentz, A. 2006. Replanning with rrts. In *IEEE International Conference on Robotics and Automation*, 1243–1248. IEEE.

Ferrari, C.; Pagello, E.; Ota, J.; and Arai, T. 1998. Multirobot motion coordination in space and time. *Robotics and Autonomous Systems* 25(3-4):219–229.

Gonzalez, J. P., and Stentz, A. 2005. Planning with Uncertainty in Position: an Optimal Planner. In *IEEE International Conference on Intelligent Robots and Systems*.

Hennes, D.; Claes, D.; and Tuyls, K. 2012. Multi-robot collision avoidance with localization uncertainty. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 4–8.

Hoenig, W.; Kumar, T. K. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-Agent Path Finding with Kinematic Constraints. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling*.

Jansen, M., and Sturtevant, N. R. 2008. Direction maps for cooperative pathfinding. In *AAAI Conference on Artificial Intelligence and Interactive Digitial Entertainment poster*, 185–190.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2005. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *International Conference on Automated Planning and Scheduling*.

Luby, M.; Sinclair, A.; and Zuckerman, D. 1993. Optimal speedup of Las Vegas algorithms. *Information Processing Letters* (June).

Melchior, N. A., and Simmons, R. 2007. Particle RRT for path planning with uncertainty. In *IEEE Conference on Robotics and Automation*, number April, 10–14.

Melo, F. S., and Veloso, M. 2009. Learning of Coordination: Exploiting Sparse Interactions in Multiagent Systems. In *International Conference on Autonomous Agents and Multiagent Systems*.

Miller, S. A.; Harris, Z. A.; and Chong, E. K. P. 2009. Coordinated guidance of autonomous UAVs via nominal belief-state optimization. In *Proceedings of the American Control Conference*, 2811–2818.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research* 12(3):441–.

Patil, S.; Van Den Berg, J.; and Alterovitz, R. 2012. Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In *IEEE International Conference on Robotics and Automation*, 3238–3244.

Platt, R.; Kaelbling, L.; Lozano-Perez, T.; and Tedrake, R. 2012. Non-Gaussian belief space planning: Correctness and complexity. In *IEEE International Conference on Robotics and Automation*, 4711–4717.

Prentice, S., and Roy, N. 2009. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research* 28(11-12):1448–1465.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Meta-Agent Conflict-Based Search For Optimal Multi-Agent Path Finding. *Symposium on Combinatorial Search* 97–104.

Standley, T. 2010. Finding Optimal Solutions to Cooperative Pathfinding Problems. In *AAAI Conference on Artificial Intelligence*.

Stentz, A. 1993. Optimal and Efficient Path Planning for Unknown and Dynamic Environments. *International Journal of Robotics and Automation* 10:89–100.

Stentz, A. 2002. CD*: A Real-Time Resolution Optimal Re-Planner for Globally Constrained Problems. In *AAAI Conference on Artificial Intelligence*, 605–612.

Sun, W., and Torres, L. 2013. Safe Motion Planning for Imprecise Robotic Manipulators by Minimizing Probability of Collision. In *International Symposium on Robotics Research*, 1–16.

Ulusoy, A.; Smith, S. L.; Ding, X. C.; and Belta, C. 2012. Robust multi-robot optimal path planning with temporal logic constraints. In *IEEE International Conference on Robotics and Automation*, 4693–4698. Saint Paul, Minnesota, USA: Ieee.

Van Berg, J. D.; Lin, M.; and Manocha, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proceedings - IEEE International Conference on Robotics and Automation*, 1928–1935. Ieee.

van den Berg, J.; Abbeel, P.; and Goldberg, K. 2011. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research* 30(7):895–913.

Wagner, G., and Choset, H. 2011. M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. In *IEEE International Conference on Intelligent Robots and Systems*.

Wagner, G. 2015. *Subdimensional Expansion: A Framework for Computationally Tractable Multirobot Path Planning*. Phd, Carnegie Mellon University.