MAPS-X: Explainable Multi-Robot Motion Planning via Segmentation

Justin Kottinger[1], Shaull Almagor[2], and Morteza Lahijanian[1]

***Abstract*— Traditional Multi-robot Motion Planning (MMP) focuses on planning trajectories for multiple robots acting in an environment, such that the robots do not collide when the trajectories are taken simultaneously. In *safety-critical* applications, a human supervisor may want to verify that the plan is indeed collision-free. In this work, we propose a notion of explanation for a plan of MMP based on visualization of the plan as a short sequence of images representing time segments, where in each time segment the trajectories of the agents are disjoint, clearly illustrating the safety of the plan. We show that standard notions of optimality (e.g., makespan) make conflict with short explanations.**

ML: do we discuss the above point in the paper?

Thus, we propose meta-algorithms, namely *multi-agent plan segmenting*-X (MAPS-X) and its lazy variant, that can be plugged on existing centralized sampling-based tree planners X to produce plans with good explanations with desirable number of images. We demonstrate the efficacy of this explanation planning scheme and extensively evaluate the performance of MAPS-X and its lazy variant in various environments and agent dynamics.

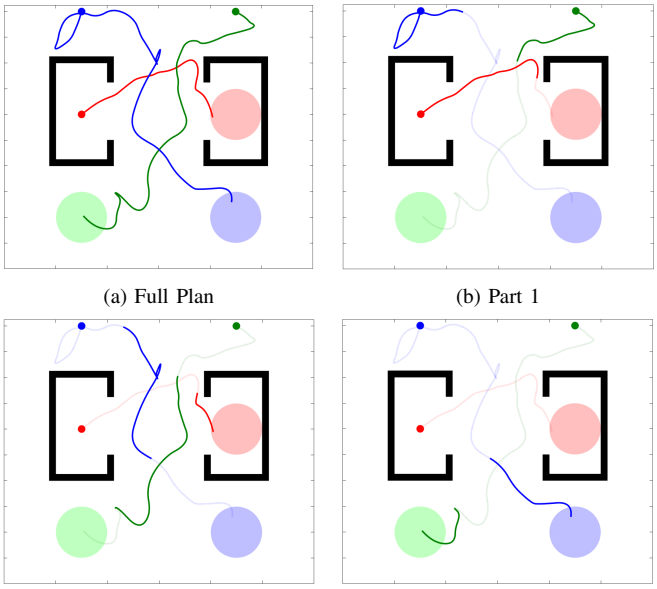
I. Introduction

Multi-robot motion planning (MMP) is a fundamental challenge in robotics and *artificial intelligence* (AI). The goal in MMP is to plan trajectories for multiple robots according to their dynamics to reach their respective goal regions such that, when the plans are executed simultaneously, every robot (agent) successfully completes its trajectory without colliding with other agents or obstacles. Applications of MMP can be found in many areas where several moving agents interact in a shared workspace. One limitation of various AI tools, including MMP, is their inability to explain their decisions and actions to human users [1]. In many *safety-critical* application domains, such as air-traffic control and hazardous material warehouses, MMP tools are rarely utilized if at all. Instead, the trajectories are either hand designed or, if a planner is used, the generated paths are given to a human-supervisor for safety verification before execution. Thus, these settings require the plan to be presented in a humanly-understandable manner. Specifically, the presentation should enable the supervisor to understand the path taken by individual agents and to verify that the agents do not collide. To this end, the goal of this work is to present a method of generating explainable motion plans for multi-agent systems.

Shaull Almagor has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 837327.

[1] Department of Aerospace Engineering Sciences, University of Colorado Boulder, USA {firstname.lastname}@colorado.edu

[2] The Henry and Marilyn Taub Faculty of Computer Science, Technion, Israel shaull@cs.technion.ac.il



(a) Full Plan (b) Part 1 (c) Part 2 (d) Part 3

Fig. 1: A plan for three (second-order) robotic agents in (a) is explained for visual verification via disjoint decomposition in (b)-(d). The small and large circles mark the initial and goal locations for the agents, respectively.

In general, MMP is **NP-Complete** even in the discrete (finite space) setting and naturally becomes intractable in the continuous (space) setting as the number of robots grows. A significant body of work is dedicated to overcoming this difficulty, both in the discrete domain (e.g., [2], [3]) and continuous domain (e.g., [4]–[7]). There are two general approaches to MMP: *centralized* methods, which work in the composed space of all the agents [6]–[8], and *de-centralized* approaches, which divide the problem into several subproblems and solve each separately [9]–[12]. The focus of all these works is to overcome the general difficulty of computing a plan in a short amount of time and optimizing the plan according to a measure such as makespan and path length. These works do not take into consideration the explainability of the plan. In fact, explainability often conflicts with other optimality measures, and requires separate treatment.

Significant effort has been dedicated to providing explanations for problems in AI and machine learning. For example, the work of [13] utilized visualization to explain the result of certain machine learning algorithms that often come up with complicated classifiers. In [14], explanations are given by analyzing alternative plans with some user-defined properties. In [15], a user proposes a plan, and explanations are

given as a minimal set of differences between the actual plan and the proposed plan. A broader approach was later given in [16], where multiple types of explanations are allowed. In that setting, the user can change the plan, motivating the planner to either explain why the original plan is better or to re-plan. None of these studies, however, focus on the MMP problem.

In [17], we propose an explanation scheme based on visualization for MMP in discrete domains. There, in order to convince a human supervisor that a suggested plan does not cause a collision between the agents, the plan is decomposed into time segments, such that within each segment the paths of the agents are *disjoint* (i.e., non-intersecting). Then, an explanation of the plan comprises a sequence of images representing each segment. An example of such explanations for three continuous agents is shown Fig. 1. It is important to note that, since identification of line intersections is made very early in the cognitive process (namely in the primary visual cortex) [18], [19], it is easy for a human to verify that in each segment, the paths do not intersect. Moreover, the sequence of images is potentially (and indeed, often in practice) much shorter than displaying, e.g., a slowed-down video of the agents taking their paths, and is hence easy to verify. In addition, [17] showed that finding optimal explanations for a given motion plan can be done in polynomial time, whereas generating plans for explainability, i.e., limiting the number of segments, is, at best, **NP-Complete**.

The algorithms proposed in [17] are based on a discrete (or discretized) environment, and thus overlook the challenges involved with motion planning in the continuous domain. In this work, we focus on MMP with explanations for realistic robotic systems in the continuous space with kinodynamical constraints. To this end, we treat explainability as an additional constraint on top of MMP, and incorporate it into existing sampling-based algorithms. As mentioned above and shown in [17], there is often a trade-off between planning for short explanations and short paths. Hence, explainability may conflict with state-of-the-art heuristics for MMP. In order to factor out precise heuristics, we devise generic meta-algorithms, that search for optimally-explainable plans using any centralized sampling-based algorithm. We demonstrate our meta algorithms by plugging them with several classical motion planners such as *rapidly-exploring random trees* (RRT) [20], *expansive space trees* (EST) [21] and the more recent *stable-sparse RRT* (SST) [22].

The main contribution of this work is an explanation scheme for MMP that is based on path segmentation in the continuous domain. This scheme introduces a new constraint (challenge) to the motion planning problem that is not previously studied, to the best of our knowledge. We present two meta algorithms called *multi-agent path segmenting*-X (MAPS-X), where X can be any existing centralized (kinodynamic) MMP planner. Another contribution is an extensive evaluation of these algorithms, highlighting their generality and differences in performance.

II. Problem Formulation

Consider $k \in \mathbb{N}$ robotic systems (agents), in a shared workspace $W \subseteq \mathbb{R}^2$ which includes a finite set of obstacles O, where each obstacle $o \in O$ is a closed subset of W, i.e., $o \subset W$. The motion of each agent $i \in \{1,2,\ldots,k\}$ is subject to the following dynamic constraint:

$$\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i), \qquad \mathbf{x}_i \in X_i \subseteq \mathbb{R}^{n_i}, \qquad \mathbf{u}_i = U_i \subseteq \mathbb{R}^{m_i}, \qquad (1)$$

where X_i and U_i are the agent i's state and input spaces, respectively, and $f_i : X_i \times U_i \to X_i$ is an integrable and possibly nonlinear function.

Given a time interval $[t_0, t_f]$, where $t_0, t_f \in \mathbb{R}_{\geq 0}$ and $t_0 < t_f$, a controller $\mathbf{u}_i : [t_0, t_f] \to U_i$, and initial state $x_{i,0} \in X_i$, function f_i can be integrated up to time $t_1 \leq t_f$ to form a *trajectory segment* $\mathbf{x}_i^{t_0:t_1}$ for agent i, where $\mathbf{x}_i^{t_0:t_1}(t_0) = x_{i,0}$. For $s \in \mathbb{N}$ consecutive time intervals

$$[t_0, t_1], [t_1, t_2], \ldots, [t_{s-1}, t_s], \qquad (2)$$

where $t_s = t_f$, we define a *trajectory*

$$T_i = \{\mathbf{x}_i^{t_0:t_1}, \mathbf{x}_i^{t_1:t_2}, \ldots, \mathbf{x}_i^{t_{s-1}:t_s}\},$$

where

$$\mathbf{x}_i^{t_{l-1}:t_l}(t_l) = \mathbf{x}_i^{t_l:t_{l+1}}(t_l) \quad \forall l \in \{1,2,\ldots,s-1\},$$

to be a set of s trajectory segments.

Let $X_i^G \subset X_i$ and $\mathbf{x}_{i,0} \in X_i$ denote the goal region (destination) and initial state of agent i, respectively. The goal of multi-agent motion planning (MMP) is to find a trajectory T_i with $\mathbf{x}_i(t_0) = \mathbf{x}_{i,0}$ for every agent $i \in \{1,2,\ldots,k\}$ such that no agent collides with any obstacles nor with other agents, and $\mathbf{x}_i(t_f) \in X_i^G$. In *explainable* MMP we require that in addition, the interval $[t_0, t_f]$ can be segmented to at most $r \in \mathbb{N}$ consecutive time intervals (for some user-defined upper bound r), such that within each time interval, the trajectories are disjoint when presented to the user.

To formally define the explainable MMP problem, we start by formalizing the notion of disjoint segments. Let $\text{PROJ}_W^{X_i} : \mathbb{R}^{n_i} \to W$ be a function that projects a state $x_i \in X_i$ of agent i onto workspace W. Then, we call two trajectory segments $\mathbf{x}_i^{t_1:t_2}$ and $\mathbf{x}_j^{t_1:t_2}$ *disjoint* if

$$\text{PROJ}_W^{X_i}(\mathbf{x}_i^{t_1:t_2}(t)) \neq \text{PROJ}_W^{X_j}(\mathbf{x}_j^{t_1:t_2}(t')) \quad \forall t, t' \in [t_1, t_2],$$

where $\mathbf{x}_i^{t_1:t_2}(t) \in \mathbb{R}^{n_i}$ is the state on the trajectory segment at time t.

We extend the notion of disjoint from segments to trajectories as follows. Given s time intervals as in (2), we call a set of trajectories $T = \{T_1, T_2, \ldots, T_k\}$ *segment-disjoint* for the s time intervals in (2) if for every segment $l \in \{1,2,\ldots,s\}$ the induced trajectory-segments $\mathbf{x}_i^{t_{l-1}:t_l} \in T_i$ and $\mathbf{x}_j^{t_{l-1}:t_l} \in T_j$ are disjoint for all $i, j \in \{1,2,\ldots,k\}$ and $i \neq j$. We can now formulate the *explainable* MMP problem as follows.

Problem 1 (Explainable MMP*):* Given k robotic agents with dynamics described in (1), initial states $x_{1,0}, \ldots, x_{k,0}$, goal regions $X_1^G, \ldots, X_k^G$, and a bound $r \in \mathbb{N}$ on the number of segments, find a controller $\mathbf{u}_i : [t_0, t_f] \to U_i$ for each agent

$i \in \{1,\ldots,k\}$ and time points $t_1,\ldots,t_s$, where $s \leq r$ and $t_0 < t_1 < \ldots < t_{s-1} < t_s = t_f$, such that the obtained trajectory T_i takes agent i from $\mathbf{x}_i(t_0) = x_{i,0}$ to $\mathbf{x}_i(t_f) \in X_i^G$ while avoiding collisions with obstacles, and the set of trajectories $T = \{T_1,\ldots,T_k\}$ is segment-disjoint.

Note that the segment-disjoint requirement for the trajectories in Problem 1 implies that the trajectories do not cause collisions between the agents. Further, a solution to this problem consists of not only k valid trajectories but also a decomposition of these trajectories into s set of disjoint segments. The explanations are then s images, one image per set of segments projected onto the workspace W, e.g., Fig. 1b-1d.

We emphasize that our goal in this work is not to design an efficient algorithm that solves the general MMP problem. Rather, our goal is to design meta algorithms that turn an existing MMP planner into an explainable MMP that solves Problem 1. Specifically, we focus on centralized MMP, where planning takes place in the compound space of the agents. This approach has two advantages: first, centralized algorithms maintain most of the structure of the search space. This allows us to obtain optimal explanations (segments), which in turn sets a baseline by which to compare more involved algorithms for explainable MMP. Second, centralized planning uses well-understood algorithms, which enable us to reason about the explanations we obtain with respect to different algorithms. In contrast, de-centralized MMP algorithms may be too involved to separate their properties from the explanations they yield. We discuss incorporating explanations into more involved algorithms in Section V.

III. Algorithms

In this section, we present three methodologies to solve Problem 1. These methods are based on centralized approach to MMP, so we first present Planner X that is a generic centralized sampling-based tree planner. Then, we define a post-process procedure that can minimally decompose an existing MMP solution, e.g., returned by Planner X, into disjoint segments (explanations). Next, we outline two frameworks that can be incorporated into Planner X to solve explainable MMP queries.

A. Planner X: Centralized Sampling-based Tree MMP

Centralized tree-based planners grow a motion tree in the composed state space $\mathbb{X} = X_1 \times X_2 \times \ldots \times X_k$ of the agents according to the dynamics (1) through sampling and extension procedures. A generalized sampling-based tree planner X is outlined in Alg. 1. It takes the composed state and input spaces, $\mathbb{X}$ and $\mathbb{U} = U_1 \times \ldots \times U_k$, respectively, the goal set $\mathbb{G} = X_1^G \times \ldots \times X_k^G$, the set of obstacles O, an initial configuration for all agents $x_0 = (x_{1,0}^T,\ldots,x_{k,0}^T)^T \in \mathbb{X}$, a specified number of iterations $N \in \mathbb{N}$.

The algorithm first initializes the graph with the initial state x_0. Next, an existing node x_{near} and a random state x_{rand} are picked through a sampling process in Line 3. After that, Line 4 samples control inputs $u \in \mathbb{U}$ and propagates the multi-agent system from x_{near} toward x_{rand} to generate a new state

Algorithm 1: Planner X ($\mathbb{X}$, $\mathbb{U}$, $\mathbb{G}$, O, x_0, N)

```
1  G = {V ← x_0, E ← ∅};
2  for N iterations do
3      (x_rand, x_near) ← sample(X, V);
4      x_new = multiAgentExtend(x_near, x_rand, U);
5      if isValid(x_near, x_new) then
6          V ← V ∪ {x_new}; E ← E ∪ {x_near, x_new};
7          if x_new ∈ G then
8              return x_0, x_new;
9  return G(V, E)
```

x_{new}. If the newly generated trajectory $\overrightarrow{x_{near},x_{new}}$ is valid, i.e., it does not result in a collision with obstacles nor with other agents, then Line 6 adds the vertex x_{new} and the edge $\overrightarrow{x_{near},x_{new}}$ to G. This process repeats a maximum of N times, exiting early if a solution is found. To speed up computation, it is common in the multi-agent setting to propagate only agents i that are *not* in their respective goal region X_i^G in the procedure *multiAgentExtend*. For the remainder of the paper, we refer to this planner as Planner-X.

B. Minimal Disjoint Segmentation

One method of explaining a multi-agent trajectory is to solve the explainability problem *after* the planning problem. This solution involves running a motion planner (e.g., Planner X) to generate a solution $\overrightarrow{x_0,x_{goal}}$, and then finding a minimal segmentation of the solution. This post-processing procedure, shown in Alg. 2, is called *SegmentSol*.

Algorithm 2: SegmentSol($T = \overrightarrow{x_0,x_{goal}}$)

```
1  d, s ← 1; v ← x_0;
2  while v ≠ x_goal do
3      intersection ← project2D(v, d);
4      if intersection then
5          for d − 1 times do
6              v.segmentNum ← s; v ← v.child;
7          d ← 1; s ← s + 1;
8      else
9          d ← d + 1;
10 return x_0, x_goal
```

Alg. 2 is implemented using a greedy approach – we traverse the nodes of the solution trajectory $\overrightarrow{x_0,x_{goal}}$, and add nodes to a segment as long as the projection of the agents' trajectories do not intersect (Line 3). Once an intersection occurs, we end the segment (collating the nodes before the intersection), and start a fresh segment (Lines 6 and 7).

Checking for intersections (procedure Project2D, line 3) can be implemented efficiently e.g. using linear interpolations of the projected paths in some Δt intervals. As we prove in [17], this greedy approach is guaranteed to obtain

a minimal segmentation among those whose end-points are multiples of Δt. We thus have the following.

Theorem 1 (Minimal disjoint segmentation): Given a set of trajectories $T = \{T_1, \ldots, T_k\}$ for k agents, Alg. 2 computes a segmentation of T such that the resulting trajectories are segment-disjoint with minimal number of segments[1].

Observe that Alg. 2 makes a single pass on the trajectory, but for each new node along the trajectory, intersections need to be checked against the segment collated so far. Thus, the algorithm has a quadratic running time, parametrized by the implementation of *project2D* (Line 3), which can be made efficient.

The post-processing procedure SegmentSol provides explanations for a solution of the MMP problem, i.e., enables users to validate that multi-agent trajectories are collision free. While this procedure guarantees a minimal segmentation for the given trajectory, it cannot guarantee that this number is below the given bound r (or indeed, the minimal segmentation of any trajectory). That is because the planning process and the segmentation process are completely separate. Thus, strictly relying on Alg. 2 could result in unsatisfiable explanations that require many segments. Next, we show how to solve the planning problem and the explanability problem simultaneously.

C. Planning with Segmentation

1) Lazy MAPS-X: The most intuitive solution for combining explanations into the planning procedure is to incorporate Alg. 2 into Planner X. The resulting algorithm is shown in Alg. 3. It runs Planner X until a solution is found. Then, rather than immediately exiting the loop with a solution, Line 8 calculates the number of disjoint segments of the solution. If it is satisfactory, i.e. the solution requires at most the user-defined upper bound r, it returns trajectory $\overrightarrow{x_0, x_{new}}$. Otherwise, Line 12 prunes the unsatisifiable portion of the solution and continues planning. The loop repeats until a satisfiable solution is found.

Algorithm 3: Lazy MAPS-X ($\mathbb{X}$, $\mathbb{U}$, $\mathbb{G}$, O, x_0, N, r)

1 $G = \{V \leftarrow x_0, E \leftarrow \emptyset\}$;
2 **for** N *iterations* **do**
3 $\quad (x_{rand}, x_{near}) \leftarrow sample(\mathbb{X}, V)$;
4 $\quad x_{new} = multiAgentExtend(x_{near}, x_{rand}, \mathbb{U})$;
5 $\quad$ **if** $isValid(\overrightarrow{x_{near}, x_{new}})$ **then**
6 $\quad\quad V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{\overrightarrow{x_{near}, x_{new}}\}$;
7 $\quad\quad$ **if** $\mathbf{x}_{new} \in \mathbb{G}$ **then**
8 $\quad\quad\quad s \leftarrow segmentSol(\overrightarrow{x_0, x_{new}})$;
9 $\quad\quad\quad$ **if** $s \leq r$ **then**
10 $\quad\quad\quad\quad$ **return** $\overrightarrow{x_0, x_{new}}$
11 $\quad\quad\quad$ **else**
12 $\quad\quad\quad\quad pruneSol(\overrightarrow{x_0, x_{new}})$;
13 **return** $G(V, E)$

[1]Minimal among segmentations that whose end-points multiples of Δt.

We call this framework *lazy multi-agent path segmentation* X (Lazy MAPS-X) because it 'lazily' turns Planner X into an explainable planner. The number of segments required to explain a trajectory segment from the root node x_0 to a node x_{new} is *only* considered if it is part of a possible solution. All other nodes are ignored. In Section IV, we show that Lazy MAPS-X works well when the optimal explanation scheme matches intuitive trajectories, but its benefits are hindered when the system behavior must be changed to match optimal explanation schemes. In such a situation, a more complete framework is desired.

2) MAPS-X:

ML: first, give a high-level overview of this planner and how it is different form lazy maps-x.

Alg. 4 outlines the complete framework for generating satisfiable explanations schemes to the MMP problem. Here, we define a *cost* for each node x_{new} in the graph G equivalent to the number of segments required to explain the trajectory from the root node x_0 to x_{new}. After verifying that x_{new} is valid, MAPS-X calculates the cost of the node in Line 6. If it is satisfiable ($x_{new}.cost \leq r$), then the node is added to the tree; otherwise, it is rejected. The resulting behavior is a graph G that tracks the number of segments required to explain each of its branches as the tree grows. Because only satisfiable nodes are added to G, MAPS-X guarantees that number of segments (explanations) of a solution is less than r Hence, MAPS-X solves Problem 1. Furthermore, since all nodes individually track their segment count (cost), once a solution is found, the segmentation information is already embedded in the solution, and no further computation is needed.

Algorithm 4: MAPS-X ($\mathbb{X}$, $\mathbb{U}$, $\mathbb{G}$, x_0, N, ΔT, r)

1 $G = \{V \leftarrow x_0, E \leftarrow \emptyset\}$;
2 **for** N *iterations* **do**
3 $\quad (x_{rand}, x_{near}) \leftarrow sample(\mathbb{X}, V)$;
4 $\quad x_{new} = multiAgentExtend(x_{near}, x_{rand}, \mathbb{U})$;
5 $\quad$ **if** $isValid(\overrightarrow{x_{near}, x_{new}})$ **then**
6 $\quad\quad findTotalCost(x_{new})$;
7 $\quad\quad$ **if** $x_{new}.cost \leq r$ **then**
8 $\quad\quad\quad V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{\overrightarrow{x_{near}, x_{new}}\}$;
9 $\quad\quad\quad$ **if** $x_{new} \in \mathbb{G}$ **then**
10 $\quad\quad\quad\quad$ **return** $\overrightarrow{x_0, x_{new}}$;
11 **return** $G(V, E)$

Alg. 5 presents the procedure for calculating the cost of each node. It checks the current segment for an intersection in Line 1 through procedure *checkDisjoint* (Alg. 6). If it exists, then it iteratively shortens the trajectory segment until it is disjoint. It does so by adding a node at the location of intersection x_{int} (Line 4) and checking the new trajectory segment $\overrightarrow{x_{int}, x_{new}}$ for an intersection (Line 5).

Alg. 6 outlines the procedure *checkDisjoint*, which checks if $\text{PROJ}_W^{\mathbb{X}}(\mathbf{x}^{t_1:t_2})$ is disjoint.

Algorithm 5: findTotalCost(x_{new})

```
1 intersection, x_int, steps ← checkDisjoint(x_new);
2 if intersection then
3     while intersection do
4         addIntermediateState(x_int, x_new, steps);
5         intersection, x_int, steps ← checkDisjoint(x_new);
6 else
7     x_new.cost ← x_new.parent.cost;
```

ML: PROJ is an operator for points, not trajectories. Need to say with an abuse of notation, we also use PROJ for projection of trajectories on to W.

Firstly, line **??** interpolates between x_{new} and its parent to generate a discretized trajectory segment between the two nodes for all agents.

ML: how is it different from $project2D$? Also, do we need to interpolate, interpolation is one way of doing this?

Next, it checks the segment for each agent against all other discretized segments of the other agents. If all trajectory segments are disjoint, the procedure returns *null*. Otherwise, Lines 5- 7 calculates the relevant information and provides it to Alg. 5 which utilizes it to add an intermediate state at the location of intersection x_{int}, create a new segment at $\overrightarrow{x_{int}, x_{new}}$, and checks the newly created trajectory segment for an intersection. The cost calculation process of the trajectory segment from $x_{new}.parent$ to x_{new} repeats until the trajectory segment is fully disjoint. There is no requirement on the type of interpolation that is performed, however, our experiments show linear interpolation works well.

ML: still need to polish the paragraph above. Too specific to our implementation.

Algorithm 6: $checkDisjoint(x_{new})$

```
1 v → x_0;
2 while current segment do
3     in progress;
4     intersect → project2D(v) if intersect then
5         intersection ← True;
6         x_int ← locOfIntersect;
7         steps ← findSteps(x_new.parent, x_int);
8         return intersection, x_int, steps
9 return intersection, x_int, steps
```

Completeness and Optimality: It is important to note that Alg. 3 and 4 inherent the completeness property of the underlying Planner X. For example, it is common for many centralized sampling-based tree MMP planners to be probabilistically complete (PC), i.e., as number of planning iterations N approaches infinity, the probability of finding a solution, if one exists, approaches 1. Then, we can make the following statement for Lazy MAPS-X and MAPS-X that solve Problem 1.

Theorem 2 (Completeness): Planners Lazy MAPX-X and MAPS-X are probabilistically complete if and only if Planner X is probabilistically complete.

The proof for MAPS-X follows from Theorem 1 (the computed cost of each node is the minimal number of disjoint segments from root to that node) and the fact that MAPS-X mimics the behavior of X, while only being more restrictive in the validity of nodes in the graph G with respect to bound r, i.e., it adds a node to the tree only if the number of disjoint segments to the node from the root is less than r.

ML: Also, state a reason for PC of lazy MAPS-X

Therefore, if a solution to Problem 1 exists in a particular query, MAPS-X will find it with probability approaching 1 as N approaches infinity.

JK: (1) does this work? (2) PC would also work for Lazy, right??

ML: yes and yes.

ML: add a discussion on optimality: by iteratively lowering cost, can asymptotically compute the optimal number of segments.

IV. Experiments and Benchmarks

In this section, we demonstrate the efficacy of our explanation scheme and evaluate the performance of the proposed meta-algorithms lazy MAPS-X and MAPS-X. We implemented these algorithms with two classical motion planners RRT [20] and EST [21], and evaluated their performance in several environments, numbers of agents, and dynamics. The benchmarking results are shown in Table I. See the full version of the paper [23] for a more extensive analysis and evaluation.

SA: arXiv will unlikely publish in time. Shall we upload it to a personal website in the meantime?

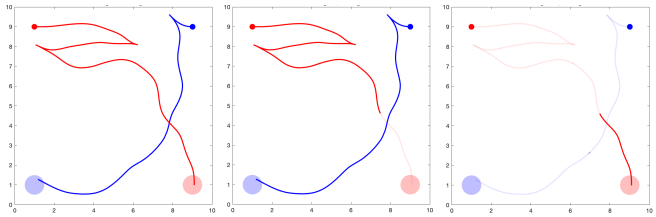
All of our planners were developed using the *Open Motion Planning Library* (OMPL) [24] and are available here [25]. The benchmarks were performed on a machine with AMD Ryzen 7 3.9 GHz CUP and 64 GB of RAM.

A. MAPS Planning

We begin by showcasing our explainability scheme in several settings, and gaining some insight into its properties.

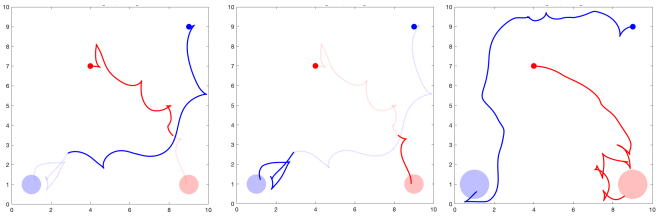
Fig. 2a shows an example solution of the continuous MMP problem produced by RRT. The example shows two agents, each with 2nd order car dynamics, that cross each others' paths to get to their goals (indeed, the paths must cross in this environment). While it is difficult for a human to validate that this plan is collision free, this becomes easy with MAPS-RRT, using two images (Figs. 2b-2c). Observe that in this example, the plan found by RRT already admits a minimal decomposition, therefore MAPS-RRT merely finds it.

In contrast, the environment in Fig. 3 can easily admit non-intersecting paths as the blue agent can go around the red agent (Fig. 3c). However, with RRT, we often get intersecting paths as shown in Fig. 3a-3b. By using MAPS-RRT and



(a) Full Solutions (b) $\Delta T_1 = [0, 38.5]$ (c) $\Delta T_2 = [38.5, 45.5]$

Fig. 2: Open Space 1: solutions via RRT and MAPS-RRT.



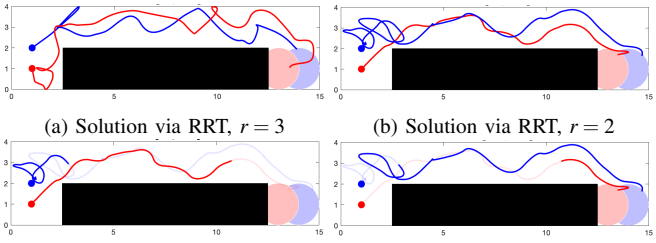
(a) $\Delta T_1 = [0, 22.1]$ (b) $\Delta T_2 = [22.1, 48.9]$ (c) MAPS-X $r = 1$.

Fig. 3: Solution via RRT and MAPS-RRT with $r = 1$.

setting $r = 1$, we quickly get the easily-explainable solution in Fig. 3c with only one segment.

Explainability can come at a cost of plan length, as we show in Fig. 4: when planning with a bound of $r = 3$ segments, we obtain the short plan in Fig. 4a, where the agents follow one another closely in the corridor. This example creates a zig-zag behavior between agents that follow 2^{nd}-order linear dynamics, making it difficult for human validation (and indeed requires a 3-segment explanation).

However, a better-explainable plan, obtained by setting $r = 2$, is given in 4b. There, the blue agent waits for the red agent to get far ahead, before entering the corridor, as is explained in Figs. 4c, 4d



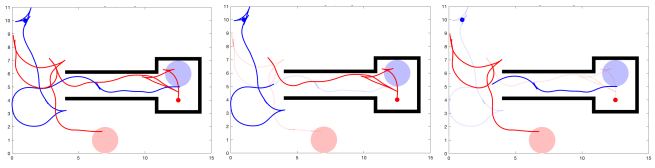
(a) Solution via RRT, $r = 3$ (b) Solution via RRT, $r = 2$

(c) $\Delta T_1 = [0, 14.5]$s (d) $\Delta T_2 = [14.5, 42.3]$s

Fig. 4: Congested space: solutions via RRT and MAPS-RRT.

When agents go in opposite directions through a corridor, as demonstrated in the environment of Fig. 5c, a controller may want to assure that the planner does not get both robots in the corridor at the same time, as that would be risky. The explanation given in Figs. 5b, 5c shows that indeed, only one robot is at the corridor at any given time.

Validating trajectories without explanations gets more difficult as the number of agents increases. For example, consider the solution shown in Fig. 6a where the agents have 2^{nd} order linear dynamics. A human user could have a difficult time checking that the entire trajectory is collision



(a) Full solution (b) $[0, 24.5]$s (c) $[24.5, 41.2]$s

Fig. 5: Solution via MAPS-EST in a narrow alley.

free. However, planning with the MAPS framework presents an easily verified solution, as shown in Fig. 6. Moreover, by decreasing bound r, we improve the explainability and as a result get a "cleaner" plan, e.g., the solution in Fig 1 was obtained with $r = 3$.

B. Performance Evaluation

We now turn to study the performance of our algorithms, our results are summarized in Table I.

Recall that MAPS-X is paramterized by an algorithm X, and inherits the planning properties from it. As this is an orthogonal concern to planning (for the scope of this paper), we focus on RRT as the planner. Further results can be found in the full version [23].

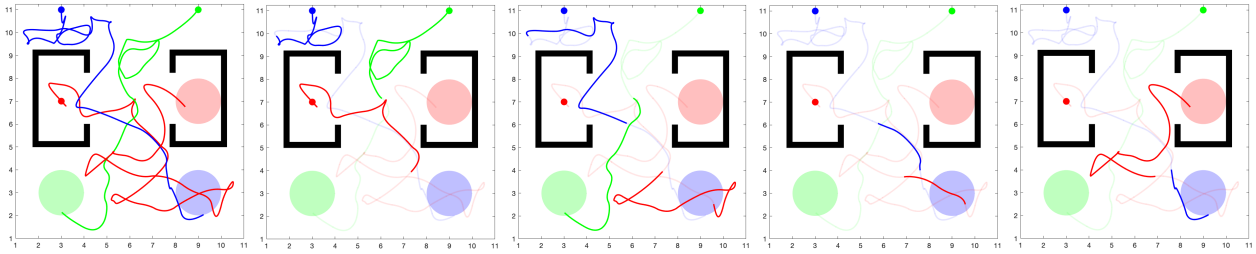
Our results confirm expected phenomena of explanations:

- For $r = \infty$, i.e., when planning without a bound, MAPS-RRT and Lazy MAPS-RRT perform the same search, but MAPS-RRT has additional computational cost of tracking segmentation. Thus, in e.g., lines 1,2 both algorithms obtain the same average number of segments (2 ± 1 segments), but MAPS-RRT takes longer to compute the segmentations.
- As the bound r decreases (lines 5,6,7 and lines 8,9), MAPS-RRT takes longer to find solutions, but gives a lower number of segments.
- For low values of r, Lazy MAPS-RRT is required to prune very often. Since unline MAPS-RRT, it does not store segmentation information in the nodes, these prunings are expensive, which reflects in the higher runtimes and fewer solutions found, as can be see in lines 3,4.

V. Conclusion and Discussion

This work outlines a new aspect of MMP by considering explanation schemes for generated plans. The MAPS-X framework can be readily plugged to existing planners in order to provide explainable plans. As we showed, using MAPS-X to find short explanations sometimes generates longer plans, but can also make plans neater (at the cost of computational time).

In future work, we plan to add explanation generation to state-of-the-art techniques, and in particular to decentralized approached, in order to improve scalability. Decentralized algorithms pose a challenge to explainability, as they often utilize techniques such as follow-the-leader [27], or highways [28], which tend to elicit poor explanations. Combining



(a) Full solution (b) $\Delta T_1 = [0, 17.1]$s (c) $\Delta T_2 = [17.1, 34.6]$s (d) $\Delta T_3 = [34.6, 38.1]$s (e) $\Delta T_4 = [38.1, 58.2]$s

Fig. 6: Solution via RRT and MAPS-RRT with 3 agents

	Space	Planner	Dyn.	# of Agents	r	% of Solutions Found	Computation Time	Computation Time Success	Average Cost	Segmentation Time	States per Second
1	Open	MAPS-RRT	1C	2	∞	100	4 ± 3	4 ± 3	2 ± 1	0.8 ± 0.5	1300 ± 75
2	Open	Lazy MAPS-RRT	1C	2	∞	100	3 ± 2	3 ± 2	2 ± 1	0.12 ± 0.01	1600 ± 94
3	Open	MAPS-RRT	1C	2	1	90.5	11 ± 9	9 ± 6	1 ± 0	4 ± 3	800 ± 99
4	Open	Lazy MAPS-RRT	1C	2	1	7.2	20 ± 11	12 ± 9	1 ± 0	2 ± 1	1200 ± 77
5	Congested	MAPS-RRT	2C	2	2	51	400 ± 200	200 ± 140	2 ± 0	47 ± 10	220 ± 52
6	Congested	MAPS-RRT	2C	2	7	95.6	35 ± 30	52 ± 20	6 ± 1	10 ± 2	570 ± 56
7	Congested	MAPS-RRT	2C	2	∞	100	40 ± 41	40 ± 41	7 ± 2	6 ± 1	600 ± 60
8	Corridor	MAPS-RRT	1U	2	6	84.3	30 ± 37	20 ± 23	4 ± 1	8 ± 9	600 ± 188
9	Corridor	MAPS-RRT	1U	2	∞	88.6	30 ± 33	20 ± 20	5 ± 2	5 ± 5	600 ± 172

TABLE I: Benchmark Results

these methods requires a careful examination of the behaviors of these algorithms, and possibly an adaptation of the explanation schemes for them.

REFERENCES

[1] M. Turek, "Explainable artificial intelligence." [Online]. Available: https://www.darpa.mil/program/explainable-artificial-intelligence

[2] R. Stern, N. R. Sturtevant, D. Atzmon, T. Walker, J. Li, L. Cohen,

H. Ma, T. K. S. Kumar, A. Felner, and S. Koenig, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Symposium on Combinatorial Search (SoCS)*, pp. 151–158, 2019.

[3] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[4] F. Gravot and R. Alami, "A method for handling multiple roadmaps and its use for complex manipulation planning," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3. IEEE, 2003, pp. 2914–2919.

[5] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2471–2476.

[6] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[7] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[8] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 591–607.

[9] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs," *arXiv preprint arXiv:1405.5848*, 2014.

[10] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 2112–2119.

[11] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 430–435.

[12] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *Robotics Research*. Springer, 2018, pp. 599–616.

[13] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, no. 1, Mar 2019. [Online]. Available: http://dx.doi.org/10.1038/s41467-019-08987-4

[14] R. Eifler, M. Cashmore, H. Jorg, D. Magazzeni, and M. Steinmetz, "Explaining the space of plans through plan-property dependencies," *Proceedings of the 2nd Workshop on Explainable Planning (XAIP 2019)*.

[15] S. Kambhampati, "Synthesizing explainable behavior for human-ai collaboration," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 1–2.

[16] M. Fox, D. Long, and D. Magazzeni, "Explainable planning," 2017.

[17] S. Almagor and M. Lahijanian, "Explainable multi agent path finding," in *To appear in Int'l Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2020.

[18] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, 1959.

[19] S. Tang, T. S. Lee, M. Li, Y. Zhang, Y. Xu, F. Liu, B. Teo, and H. Jiang, "Complex pattern selectivity in macaque primary visual cortex revealed by large-scale two-photon imaging."

[20] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[21] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3. IEEE, 1997, pp. 2719–2726.

[22] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[23] J. K. amd Shaull Almagor and M. Lahijanian1, "MAPS-X: Explainable multi-robot motion planning via segmentation." [Online]. Available: ????

[24] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.
[25] [Online]. Available: https://github.com/JustinKottinger/MAPS-X
[26] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1431–1443, 2016.
[27] H. Choset and W. Henning, "A follow-the-leader approach to serpentine robot motion planning," *Journal of Aerospace Engineering*, vol. 12, no. 2, pp. 65–73, 1999.
[28] L. Cohen and S. Koenig, "Bounded suboptimal multi-agent path finding using highways." in *IJCAI*, 2016, pp. 3978–3979.