Multi-Agent Trajectory Prediction and Generation with Topological Invariants Enforced by Hamiltonian Dynamics

Christoforos I. Mavrogiannis[†] and Ross A. Knepper[‡]

[†]Sibley School of Mechanical & Aerospace Engineering, Cornell University
cm694@cornell.edu
[‡]Department of Computer Science, Cornell University
rak@cs.cornell.edu

Abstract. We present a planning framework for decentralized navigation in dynamic multi-agent environments where no explicit communication takes place among agents. Our framework is based on a novel technique for computationally efficient multi-agent trajectory generation from symbolic topological specifications. At planning time, this technique allows an agent to generate a diverse set of potential future scene evolutions in the form of Cartesian, multi-agent trajectory representations. The planning agent selects and executes the next action assigned to it from the evolution of minimum cost. The proposed strategy enables an agent to maintain a smooth and consistent navigation behavior that is robust to behaviors generated by heterogeneous agents and to unexpected events such as agents with changing intentions. Simulation results demonstrate the efficacy of our approach in responding to such situations and its potential for real-world applications in crowded environments.

1 Introduction

Several real-world, multi-agent navigation domains, such as pedestrian or street environments, prohibit the use of explicit communication among agents. Thus, rational agents need to employ mechanisms for predicting the behaviors of others to plan collision-free paths. Although predicting the behaviors of others in a detailed way is challenging, the assumption of rationality, combined with the constraints imposed to agents by the environment bounds results in the definition of a set of qualitatively distinct, global planning alternatives in the joint trajectory space of all agents, corresponding to different strategies of joint collision avoidance. Being cognizant of this structure, agents could anticipate different classes of unfolding multi-agent dynamics. This could enable them to execute actions of global outlook and consistency, which could allow for consistently expressive, smooth and efficient motion, even in the face of unexpected events, such as the appearance of agents with heterogeneous policies or agents with changing intentions. This ability is of particular importance for operation in human-populated domains, where agents continuously read and react to the rapidly changing environment. Important domains with these properties include automobile traffic in parking lots and pedestrian traffic in communal spaces like hallways.

In this paper, we approach this vision by contributing (1) a symbolic abstraction that allows us to classify and enumerate distinct multi-agent trajectory alternatives; (2) a multi-agent navigation planner that can drive a set of agents from a starting configuration to a goal configuration, while satisfying topological trajectory constraints; (3) an online, reactive navigation planning algorithm that makes use of the aforementioned multi-agent planner as a predictive mechanism; (4) simulation results, demonstrating the ability of the multi-agent planner to generate a broad range of topologically distinct, multi-agent trajectories; (5) simulation results, demonstrating the ability of the online algorithm to generate smooth, collision-free behaviors, even under challenging conditions such as the emergence of agents running different policies or agents with changing intentions.

2 Related Work

In the area of decentralized multi-agent navigation, a significant amount of research has been devoted to the design of planning algorithms, based on mechanisms for predicting the joint behavior of multiple agents. The majority of the literature typically assumes that navigating agents employ policies with shared architecture, with notable examples including the Social Force [6] and Reciprocal Velocity Obstacle [20] frameworks as well as more recent algorithms of similar basis (e.g. Moussaïd et al. [16], Karamouzas et al. [7]). These works do not make explicit predictions of other agents' trajectories but make decisions under strong assumptions on their behaviors. The problem of predicting the trajectories of multiple navigating agents in real time and for an adequate horizon to allow for motion planning is challenging, as suggested by the literature on tracking (e.g. [11]). This has motivated roboticists to look for more practical alternatives for multi-agent motion prediction. In particular, several approaches have leveraged the coupling of agents' decision making in multi-agent navigation as a way to guide the motion planning process. Some of them have employed learning techniques [5, 8, 10, 19] to develop models for prediction and generation of humanlike trajectories whereas others have employed heuristics to directly exploit the topological structure of the problem [9, 13] as a more tractable alternative for explicit trajectory prediction.

This paper is closer to the latter class of work. We leverage the topological properties of the spatiotemporal structure of the multi-agent navigation planning problem to generate diverse multi-agent trajectory predictions. In contrast to our past work [12–15], in which we used this topological structure at an abstract level to design an *action* selection mechanism, this paper introduces a framework for *trajectory* generation from symbolic, topological specifications which allows for smoother and more consistent behaviors. Our framework is based on the method of Berger [3], which allows for braiding multi-particle trajectories into desired topological patterns. Our prediction mechanism grows trajectories from agents' initial configurations to agents' predicted destinations in topologically distinct ways. This allows us to introduce desired global properties to the trajectory, in contrast to typical trajectory optimization methods (e.g. [21]), which act on a trajectory locally. Based on this mechanism, we present an online algorithm that generates a diverse set of distinctly entangled multi-agent trajectories and evaluates them with respect to their quality and likelihood, to select an action that

best adapts to other agents' impending behaviors and preferences. This allows for rapid adjustment to the changing environment and facilitates robustness to unexpected events such as the emergence of heterogeneous agents or agents with changing intentions.

3 Foundations

We consider the planning problem for an autonomous agent that navigates towards its destination in a known environment where other agents are also navigating towards their respective destinations. The agent aims at reaching its destination by following an efficient, smooth, collision-free trajectory and assumes that others share similar objectives, although it has no knowledge of their specific policies. The agent is not explicitly communicating with others and thus has no knowledge of their intended destinations or planned paths or policies but is able to perfectly observe their motion. Assuming that others share similar abilities and objectives, the agent may form a belief about how they are going to move in the future so that it can plan a safe and efficient path towards its destination.

In this paper, we present an approach inspired by the point vortex problem [1] from fluid dynamics. We design a planning framework built around the observation that the collision avoidance process for agents navigating on a plane resembles the dynamics of interacting point vortices in two dimensions. Treating agents as point masses subjected to vortex dynamics allows us to synthesize multi-agent trajectories with desired topological properties. At planning time, this technique enables a planning agent to construct several qualitatively distinct predictions about the future behavior of the system. This allows for an informed action selection towards facilitating a rapid and robust adjustment to the changing environment. Since the predictions are made with a global outlook, this strategy results in a consistently smooth and intent-expressive behavior, even in the face of unexpected events such as agents changing intentions or violating the assumption of rationality. Our approach is based on the method of Berger [3, 4] for generating braided trajectories of multi-particle systems from topological invariants [4]. In this section, we introduce some preliminaries about point vortex flows, review the method of Berger [3] and present the key components of our approach.

3.1 Hamiltonian Motion for Multi-Particle Trajectory Braiding

A dynamical system whose evolution is described by Hamilton's equations is called a Hamiltonian system. Under the Hamiltonian formalism, the state of a system is completely described by a set of variables corresponding to the generalized coordinates of the system's degrees of freedom and their conjugate momenta. Hamilton's equations relate the evolution of an energy function, called the *Hamiltonian*, to the evolution of the coordinates and momenta for all degrees of freedom of the system. In particular, denoting by q_j and p_j, $j \in M = \{1, \dots, m\}$ the generalized coordinate and conjugate momentum of the ith degree of freedom of a Hamiltonian system respectively, its evolution is given by:

$$\dot{q}_j = \frac{\partial H}{\partial p_j}, \qquad \dot{p}_j = -\frac{\partial H}{\partial q_j}, \qquad \frac{dH}{dt} = 0, \qquad j = 1, \dots, m, \tag{1}$$

where the dot notation indicates time derivatives, H denotes the *Hamiltonian* of the system (which is preserved), defined as its total energy, i.e., the sum of the total kinetic and potential energy of all degrees of freedom of the system.

Let us now combine the coordinates and momenta for each degree of freedom into a complex coordinate $z_j = q_j + ip_j$, $j \in M$. We define an analytic function

$$F(z_1, \ldots, z_m) = \Psi(z_1, \ldots, z_m) + iH(z_1, \ldots, z_m), \tag{2}$$

where $\Psi : \mathbb{C}^m \to \mathbb{R}$ and $H : \mathbb{C}^m \to \mathbb{R}$. Berger [3] showed that the Hamiltonian flow (1) results in motion $\dot{z}_j$, $j \in N = \{1, \ldots, n\}$, that follows the Wirtinger derivative of Ψ with respect to z_j. Therefore, the collective Hamiltonian motion of all degrees of freedom follows the gradient of Ψ and points towards its direction of maximum increase. Berger [3] used this finding to generate braided trajectory patterns for systems of two and three particles. In particular, he replaced Ψ with *Topological Invariants* [4] towards forcing the system to evolve along the growth of the topological invariant.

3.2 Topological Invariants of Particle Trajectories

Consider a set of n particles, following trajectories $\xi_i : [0, T] \to \mathbb{R}^2$, $i \in N$, from time $t = 0$ to time $t = T$ and let us collect these trajectories into a system trajectory $\Xi = (\xi_1, \ldots, \xi_n)$. A *topological invariant* over Ξ, may be defined as a function $\Psi : \Xi \to \mathbb{R}$ that maps the system trajectory to a real number that characterizes the spatiotemporal topology of the system dynamics. For any distorted, topology-preserving trajectory $\tilde{\Xi} \neq \Xi$ with the same endpoints $\tilde{\Xi}(0) = \Xi(0)$, $\tilde{\Xi}(T) = \Xi(T)$, for which $\xi_i(t) \neq \xi_j(t)$, $\forall t \in (0, T)$ and $i \neq j \in N$, a topological invariant is preserved, i.e., $\Psi(\tilde{\Xi}) = \Psi(\Xi)$.

3.2.1 The Winding Number

The so called *Winding Number* is a topological invariant of particular interest for our problem. Consider a curve $\gamma : [0, t] \to \mathbb{C}\backslash\{0\}$. The complex winding number of the curve γ, from time 0 to time t is defined as:

$$\lambda(t) = \frac{1}{2\pi i} \oint_{\gamma} \frac{dz}{z}, \tag{3}$$

where $z \in \mathbb{C}$. Let us express γ in polar coordinates as $\gamma(t) = r(t)e^{i\theta(t)}$, where $r(t) = ||\gamma(t)||$ and $\theta(t) = \angle\gamma(t)$. Then, through the use of the Cauchy integral formula, (3) may be decomposed into:

$$\lambda(t) = \frac{1}{2\pi i} \int_0^t \frac{\dot{r}}{r} dt' + \frac{1}{2\pi} \int_0^t \dot{\theta} dt' \tag{4}$$

and computing the integrals yields:

$$\lambda(t) = \frac{1}{2\pi i} \log\left(\frac{r(t)}{r(0)}\right) + \frac{1}{2\pi}(\theta(t) - \theta(0)). \tag{5}$$

The real part of this integral, $w = Re(\lambda)$, is a topological invariant, counting the number of times the curve γ encircled the origin in the time interval $[0, t]$. In other words, fixing the endpoints of the curve, any topology-preserving deformations are mapped to the same value of the winding number. For closed curves, the imaginary part of the winding number is zero. In the following section, considering open curves (evolving trajectories), we describe how it can be used to enforce Hamiltonian motion to interacting particles.

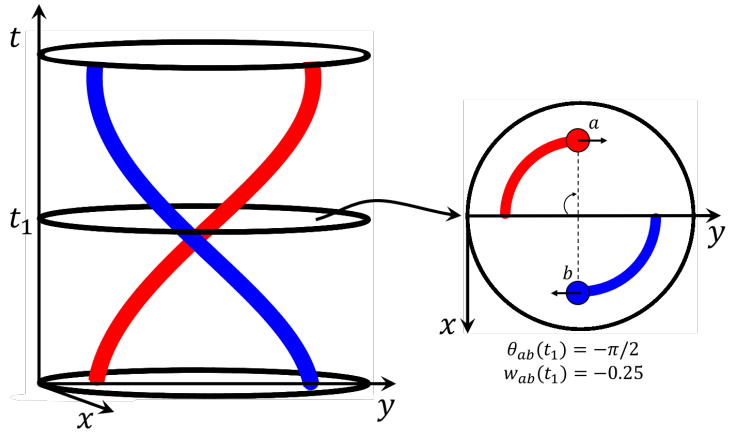


Fig. 1: Spacetime plot of the trajectories of two agents, navigating in a circular workspace (left) and projection of their trajectories until time t_1, onto the xy plane, along with the definition of their pairwise winding angle and winding number (right).

3.3 Two-Particle Vortex Motion

In this section we put the pieces together to demonstrate a motivating example from fluid dynamics that constitutes the computational basis of our approach. Consider a system of two particles, placed initially at positions $a = (a_x, a_y) \in \mathbb{R}^2$ and $b = (b_x, b_y) \in \mathbb{R}^2$ with respect to a fixed coordinate system and assume that a vortex[1] lies between them. Point vortex motion prescribes that the x and y coordinates are conjugate to each other (e.g. the conjugate momentum to a_x is a_y) [1]. Let us define the function γ_{ab} from the previous section to track the quantity $a - b$, i.e., let us set $\gamma_{ab}(t) = r_{ab}(t)e^{i\theta_{ab}(t)}$, where $r_{ab} = ||a - b||$ and $\theta_{ab}(t) = \angle\gamma_{ab}(t)$.

Assuming unit vorticity, the Hamiltonian for this system may be written as:

$$H = -\frac{1}{2\pi}\log r_{ab}. \tag{6}$$

Similarly to Sec. 3.2.1, we may define the complex winding number of γ_{ab} as:

$$\lambda_{ab}(t) = \frac{1}{2\pi i}\log(r_{ab}) + \frac{1}{2\pi}(\theta_{ab}(t) - \theta_{ab}(0)), \tag{7}$$

and let us set its real part to a dedicated variable

$$w_{ab}(t) = \frac{1}{2\pi}(\theta_{ab}(t) - \theta_{ab}(0)), \tag{8}$$

denoting the pairwise winding number of the two curves (see Fig. 1 for a graphic representation of the pairwise winding number). We may notice that $Im(\lambda_{ab}) = H$. Thus, according to Sec. 3.1, the Hamiltonian flow for this system maximizes the growth of the real part $Re(\lambda_{ab}) = w_{ab}$. This motion corresponds to the two points rotating about each other at a constant radius, in a counterclockwise direction. Hamilton's equations for this system may be derived as:

$$(\dot{a}_x, \dot{a}_y) = \left(\frac{\partial H}{\partial a_y}, -\frac{\partial H}{\partial a_x}\right) = \frac{1}{2\pi}\left(-\frac{a_y - b_y}{r_{ab}^2}, \frac{a_x - b_x}{r_{ab}^2}\right), \tag{9}$$

[1] A vortex is a region in a fluid in which the flow revolves around an axis line.

$$(\dot{b}_x, \dot{b}_y) = \left(\frac{\partial H}{\partial b_y}, -\frac{\partial H}{\partial b_x}\right) = \frac{1}{2\pi}\left(-\frac{b_y - a_y}{r_{ab}^2}, \frac{b_x - a_x}{r_{ab}^2}\right). \tag{10}$$

We may control the directionality of the rotation by switching the signs in the right hand side of eqs. (9) and (10).

3.4 Two-Agent Collision Avoidance as Vortex Motion

Treating agents as particles, we may use the outlined method of Berger [3] as a mechanism for generating two-agent, collision-avoidance maneuvers of desired topological specification. Given a winding number w_{ab}, by multiplying the right hand sides of eqs. (9) and (10) with $sign(w_{ab})$, we have a planning rule that allows us to grow trajectories for a and b that follow the direction indicated by w_{ab}, with $sign(w_{ab}) > 0$ and $sign(w_{ab}) < 0$ corresponding to right and left hand side collision avoidance respectively. In a two-agent scene, this may serve as a prediction of the emerging joint behavior. In a scene with high uncertainty, where no agent has committed to a passing side, this mechanism allows a planning agent to anticipate both outcomes. This is useful as it allows the agent to either enforce its own preference or adapt to the preference of the other agent. In the following section, we show how we use this method for synthesizing trajectories of complex topological specifications in environments with multiple agents.

3.5 Multi-Agent Trajectory Generation from Topological Specifications

Consider the problem of centralized trajectory planning for driving n agents from their initial positions $S = (s_1, \dots, s_n) \in \mathbb{R}^{2n}$ to their destinations $D = (d_1, \dots, d_n) \in \mathbb{R}^{2n}$ in a collision-free fashion and while following a global topological specification $\boldsymbol{w}$, prescribing passing sides to agents. In this paper, we model $\boldsymbol{w}$ as a tuple of pairwise winding numbers $\boldsymbol{w} = (w_{12}, w_{13}, \dots)$ from the set of such tuples $\mathcal{W}$. Assuming that each agent passes each other exactly once on its way to its destination (agents do not loop around others), the magnitude of w_{ij}, $i \neq j \in N$ is not important, so we will be using w_{ij} to refer to $sign(w_{ij})$. The cardinality of the set of possible specifications is $|\mathcal{W}| = 2^{\binom{n}{2}}$, corresponding to all possible combinations of passing sides for all agents. It should be noted that although all combinations in $\mathcal{W}$ are topologically possible, in practice, only a subset of them are meaningful and likely given agents' state history and under the assumption of rationality. Sec. 3.6 addresses the problem of evaluating the likelihood and the feasibility of a topological specification.

We now describe a policy $\pi : \mathbb{R}^{2n} \times \mathcal{W} \to \mathbb{R}^{2n}$ that can be sequentially iterated to produce a multi-agent trajectory that satisfies a topological specification $\boldsymbol{w}$. The policy (referred from now on as HTTG, standing for Hamiltonian Topological Trajectory Generation) prescribes an action $u_i \in \mathbb{R}^2$ to every agent $i \in N$, synthesized from a weighted consideration of all pairwise collision avoidance reactions between the agent and all others, towards meeting the pairwise specifications contained in $\boldsymbol{w}$. The policy is executed repeatedly until all agents reach their destinations. It may be formulated for agent i as follows:

$$u_i = \nu_i \cdot k \left(u_{att}^i + u_{rep}^i\right), \tag{11}$$

where $\nu_i \in \mathbb{R}$ is an agent's desired speed, u^i_{att}, u^i_{rep} are potentials attracting the agent towards its destination and repulsing it from others respectively and $k \in \mathbb{R}$ is a normalization constant. The potential

$$u^i_{att} = k_{att}(d_i - q_i) \tag{12}$$

attracts the agent from its current state q_i towards its destination d_i with k_{att} being an importance weight. The potential

$$u^i_{rep} = k_{rep} \sum_{j \neq i}^{N} c_{ij} w_{ij} v^i_j, \tag{13}$$

repulses agent i from each other agent $j \in N$, $j \neq i$, through the velocity v^i_j, derived from eqs. (9) and (10), with a degree of consideration equivalent to the criticality of their pairwise collision avoidance, expressed by $c_{ij} \in \mathbb{R}$ (the closer two agents are, the more critical their avoidance becomes) and along the direction indicated by w_{ij} whereas k_{rep} is an importance weight. The choice of the weighting factors k_{att}, k_{rep} expresses the relative significance between goal attraction and collision avoidance. The criticality term is designed to be a polynomial function of the distance between two agents, activated when the distance becomes lower than a threshold. By sequentially executing the outlined policy, in parallel for all agents, in equal time steps of length dt, the system of agents is forced to follow the specification $\boldsymbol{w}$. Note that this method does not guarantee the attainment of the desired topology. Depending on the number of agents, their initial configurations and intended destinations and the parameter tuning, the method has a varying control over the topological properties of agents' trajectories. Sec. 4 explores the performance of the method on scenarios with different numbers of agents.

3.6 TANP: Topologically Adaptive Navigation Planning

In this section, we present an online, decentralized navigation planning algorithm that makes use of the described method for generating online a set of topologically distinct, multi-agent trajectory predictions. The algorithm comprises the following sequence of actions: (1) **predict** the destinations of other agents; (2) **generate** a set of candidate multi-agent trajectories that drive agents from their current locations to their predicted destinations; (3) **evaluate** candidates with respect to a cost function; (4) **execute** the next action assigned to the planning agent from the lowest-cost candidate. In the following subsections, we describe the main components of the algorithm and provide a detailed presentation of it in pseudocode format (see Alg. 1).

3.6.1 Destination Prediction

In sec. 3.5, it was assumed that the planning policy has access to the destinations of other agents. In the settings we are considering (no explicit communication among agents), this is not the case. Thus, the planning agent needs to make a prediction about the destinations of others in order to use the policy. However, in practice, an agent only interacts with others for as long as they lie within its sensing range, which for current robotic systems is quite limited. During this amount of time, other agents' observed behaviors may or may not be revealing

about their specific destination. And in fact, detailed predictions of agents' destinations may not be sufficiently informative regarding agents' future behaviors; in crowded environments, the collision avoidance process is a more significant influence over agents' behaviors. For this reason, we take a more practical approach, focusing on coarse predictions of agents' future locations.

In particular, we assume that an agent's sensing range has the shape of a disk of radius R, centered at the agent's position, q_i. Any agent lying outside of this disk is not perceived by the agent whereas any agents lying behind the robot are ignored at the planning stage. For each one of the perceived and actively considered agents, we approximate their intended direction of motion by fitting a line to their recent, observed trajectory and projecting their current velocity on it. We then propagate their current speed along this direction until it intersects the boundary of the sensing disk. For our planning algorithm, that point of intersection is considered to be that agent's destination (see Fig. 2). This prediction is expected to be a coarse approximation of where an agent is heading. However, since our algorithm runs in replanning cycles, this approximation provides a sufficient amount of detail for the trajectory prediction mechanism of sec. 3.5. This mechanism makes use of the assumption that agents act rationally, i.e., agents' behaviors are driven by an incentive of making progress towards their destinations. Finally, alternative methods of filtering could be employed to provide more accurate destination prediction; however, this is not our focus and as our will be shown in Sec. 4, this simplistic model may yield the desired performance.

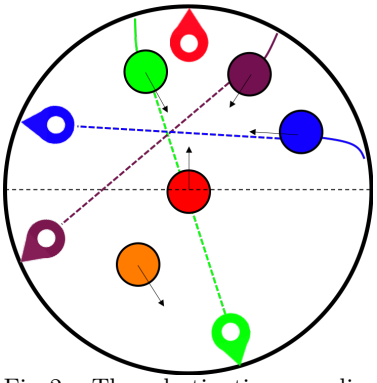


Fig. 2: The destination prediction mechanism. The red agent makes destination predictions for all agents, lying within its circular sensing disk and in front of it.

3.6.2 Outcome Evaluation

The set $\mathcal{W}$ contains symbolic representations of topologically distinct *outcomes* for the system of all considered agents. Naturally, a significant question that arises is: which outcome should the planning agent trust and follow? We approach this problem with the following sequence of computations: (1) we first evaluate an outcome with respect to its likelihood; (2) we then generate trajectory representations for the set of the K most likely outcomes $\mathcal{W}_K \subset \mathcal{W}$, using the policy presented in Sec. 3.5; (3) finally, we evaluate these K best outcomes with respect to the physical properties of their trajectory representations.

Probability of an outcome: An outcome is initially encoded symbolically as a tuple $\boldsymbol{w}$ that prescribes how agents avoid each other throughout the course of the scene. From a topological perspective, these symbols are independent of each other; any motion is allowed even if it is not efficient. However, from a real-world point of view, the collision-avoidance strategies that agents employ to avoid one-another are coupled and modeling the complex probabilistic relationships among them is a challenging problem. For our purposes in this paper, we are interested

in finding a way to bias our search towards the outcomes that are more likely to occur. We do so by using the following expression:

$$P(\boldsymbol{w}|\Xi_{past}) = P(w_{12}, w_{13}, \dots|\Xi_{past}) \propto \frac{1}{Z}\prod_{ij} P(w_{ij}|\Xi_{past}), \tag{14}$$

where Ξ_{past} denotes agents' past trajectories and Z is a normalization constant across all $\boldsymbol{w} \in \mathcal{W}$. This expression was derived by factorizing $P(w_{12}, \dots|\Xi_{past})$ using the product rule and then substituting each factor with its Bayes' rule expression. Similarly to our past work [13], we model $P(w_{ij}|\Xi_{past})$ by employing the physical quantity of angular momentum. For two agents i, j, navigating on a plane, their angular momentum L^{ij} lies along the z axis. Notice that the sign of the z-component of the momentum, L_z^{ij} is an indicator of agents' passing side and thus of the winding number of their trajectories w_{ij}, with $L_z^{ij} > 0$ indicating the emergence of $w_{ij} > 0$ (right hand side collision avoidance) and $L_z^{ij} < 0$ indicating the emergence of $w_{ij} < 0$ (left hand side collision avoidance). We incorporate the momentum as a heuristic in a sigmoid model as follows:

$$P(w_{ij}|\Xi_{past}) = \frac{1}{1 + \exp(-w_{ij}L_z^{ij})}. \tag{15}$$

The greater $|L_z^{ij}|$ is, the greater the mutual intention or preference of agents i and j over a collision avoidance along the direction of L_z^{ij} is.

Trajectory Evaluation: We evaluate a trajectory representation $\Xi_{\boldsymbol{w}}$ of an outcome $\boldsymbol{w}$ by computing its total energy $\mathcal{E} : \mathcal{Z}^n \to \mathbb{R}$, its required immediate acceleration $\mathcal{A} : \mathcal{Z}^n \to \mathbb{R}$ and its safety cost $\mathcal{S} : \mathcal{Z}^n \to \mathbb{R}$. The Energy measure (sum of squared speeds throughout the trajectory) gives an estimate of the efficiency of an outcome whereas the acceleration measure is indicative of the aggressiveness of the maneuvers required to comply with an outcome. We model the Safety cost as $\mathcal{S}(\Xi) = \exp(-d_{min})$, where $d_{min} \in \mathbb{R}$ is the minimum distance between any pair of agents in a trajectory Ξ. Note that other cost functions could be used to incorporate different considerations such as social comfort (see e.g. Sisbot et al. [18]).

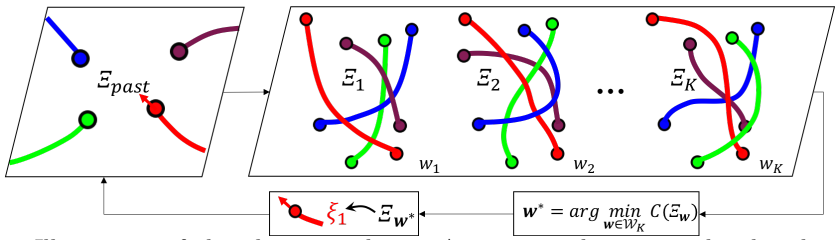


Fig. 3: Illustration of the planning scheme. At every replanning cycle, the planning agent generates a set of diverse (topologically distinct) predictions about the joint future behavior of all agents, evaluates them with respect to a cost function C and executes the action assigned to it from the prediction of lowest cost.

3.6.3 Decision Making

We first rank outcomes at a symbolic level through the use of the probability distribution, presented in Sec. 3.6.2 and determine the set of the K most likely outcomes $\mathcal{W}_K$. Then, we determine the outcome of lowest cost:

$$C(\Xi) = \alpha_e \mathcal{E} + \alpha_a \mathcal{A} + \alpha_s \mathcal{S}, \tag{16}$$

where α_e, α_a, α_s are importance weights and finally extract the optimal outcome through the following optimization scheme:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w} \in \mathcal{W}_K} C(\Xi_{\boldsymbol{w}}). \tag{17}$$

The planning agent executes the next action assigned to it from the trajectory of lowest cost $\Xi_{\boldsymbol{w}^*}$. Fig. 3 depicts a graphic representation of the planning scheme.

3.6.4 Pseudocode

Alg. 1 summarizes the described algorithm (Topologically Adaptive Navigation Planning – TANP) in pseudocode format. The algorithm runs in replanning cycles for as long as the boolean variable *AtGoal* is set to $False$, indicating that the agent has not reached its destination yet. At every cycle, the agent first determines a set of *reactive agents*, i.e., agents that lie within the robot's sensing disk and to the front of the robot's heading (function `Get_Reactive_Agents`). Then, function `Predict_Destinations` outputs predictions for the destinations of the reactive agents and `Get_Outcomes` returns a set of topological representations for outcomes that could emerge in the remainder of the execution. Function `Get_Outcome_Probability` returns the probability for each of the outcomes considered and function `Get_Best_Outcomes` returns the K best outcomes. Function `HTTG` executes the HTTG policy and generates trajectory representations for these outcomes and function `Score_Trajectory` evaluates them with respect to the cost function considered. Finally, function `Get_Best_Next_Action` returns the next action for the planning agent from the trajectory of lowest cost and function `Execute_Action` executes that action. The distance between the resulting agent state and its destination is compared to the predefined threshold ϵ and the flag *AtGoal* is updated to $True$ in case the agent is sufficiently close to its destination.

3.7 Complexity and Practical Considerations

The most computationally intense component of our algorithm is the estimation of the outcome probabilities. For n agents, this computation runs in time $O(2^{n^2})$ –the rest of the computations run in polynomial time. In practice, a replanning cycle of TANP on a scenario involving 4 agents and thus the evaluation of 64 topological classes with $K = 5$, runs at an average of $42ms$, with the worst case being $203ms$ in a non-optimized MatLab implementation on a MacBook Pro of 2015 with an Intel Core i7 processor of 2.5 GHz, running macOS High Sierra. Transfer to a faster language and optimization of parts of the code could help vastly improve performance.

Under the current design, scaling to large n is not possible. However, for a mobile robot application, we argue that it is also not practical. The sensing limitations would prohibit the emergence of a large number of agents. Even if more agents are sensed, pruning them to the subset of directly reactive agents is a motivated and human-inspired way of reducing the load. Future work involves the design of an online data-driven topology-classification mechanism that would enable agents to directly estimate the most likely candidates, without brute-forcing their evaluation.

Algorithm 1 TANP(q, d, Ξ)

Input: q − agent's current state; d − agent's intended destination; Ξ_{past} − state history of all agents; K − Number of outcomes to consider; ϵ − desired distance-to-goal threshold.

$AtGoal \leftarrow False$
while $\neg AtGoal$ **do**
  $\mathcal{R} \leftarrow$ `Get_Reactive_Agents`(Ξ_{past})
  $D \leftarrow$ `Predict_Destinations`$(\Xi_{past}, \mathcal{R})$
  $\mathcal{W} \leftarrow$ `Get_Outcomes`$(\mathcal{R})$
  $P \leftarrow$ `Get_Outcome_Probability`$(\mathcal{W}, \Xi_{past})$
  $\mathcal{W}_K \leftarrow$ `Get_BestOutcomes`$(P, \mathcal{W}, K)$
  $\mathcal{Z} \leftarrow \emptyset$
  for all $\boldsymbol{w} \in \mathcal{W}_K$ **do**
    $\Xi_{pred} \leftarrow$ `HTTG`$(\Xi_{past}, \boldsymbol{w}, D)$
    $\mathcal{Z} \leftarrow \{\mathcal{Z}, \Xi_{pred}\}$
  $C \leftarrow$ `Score_Trajectories`$(\mathcal{Z})$
  $u \leftarrow$ `Get_Best_NextAction`$(\mathcal{Z}, C)$
  $q \leftarrow$ `Execute_Action`(u)
  if $||q - d|| < \epsilon$ **then**
    $AtGoal \leftarrow True$
return None

4 Results

In this section, we present the performance of the offline planner in generating trajectories of desired topological properties and the behaviors generated by the online algorithm in different types of scenarios.

4.1 Offline Performance

We demonstrate the performance of the offline motion planner in generating topologically distinct, multi-agent navigation trajectories. We consider 4 different conditions, corresponding to different numbers of agents (2,3, 4 and 5 agents), navigating in a circular workspace of radius $2.5m$ (agents are represented as disks of radius $0.3m$). For each condition $n \in \{2, 3, 4, 5\}$, we randomly generate 100 distinct scenarios, by assigning agents initial and final locations that lead to challenging multi-agent encounters, requiring competent collision avoidance maneuvers. We execute each scenario, $2^{\binom{n}{2}}$ times, each with a distinct topological specification. We measure the success rate of the planner in generating the desired topology under all conditions considered and report it in Table 1 (a trial is considered successful if the planner was able to produce all of the distinct topologies). The planner parameters were kept constant across conditions and scenarios. It can be observed that the planner performance drops as the number of agents n increases. The method becomes more sensitive to parameter tuning, as the effects of the chaotic nature of the vortex problem [2] become more significant. In appendix A, Fig. A.1 and Fig. A.2 depict the trajectories generated by following all possible topological specifications on an example 3-agent and scenario and an example 4-agent scenario respectively. Finally, Fig. 5 shows

examples of how the outlined trajectory generation mechanism may be used for online prediction in scenarios involving two, three and four agents.

Condition	2 agents	3 agents	4 agents	5 agents
Number of Outcomes	2	6	64	1024
Success (%)	1	99.75	89.70	65.48

Table 1: Success rate of HTTG in generating the desired, topologically distinct executions for each of the 100 scenarios consider per condition.

4.2 Comparison with Trajectory Optimization

To the best of our knowledge, this is the first work that addresses the problem of generating trajectories for multiple agents in distinct topologies, specified a priori. Conceptually similar, the work of Rösmann et al. [17] considered the problem of generating multiple, topologically distinct trajectories for a single agent in a static environment with obstacles. However, optimizing multiple trajectories together and accounting for topological constraints while ensuring trajectory smoothness is a challenging problem. Typical gradient-based methods (e.g. [21]) act on the trajectory locally, with costs comprising several objectives; thus the gradient action could lose sight of the global, topological specification in favor of a different, local cost improvement. Furthermore, a differentiable cost function that would quantify the progress towards a desired topological outcome is hard to hand-design and we were not able to find any functions of that form in the literature. Our method constitutes a principled alternative to trajectory optimization for this problem. Instead of locally reshaping a set of trajectories according to the gradients on the trajectory waypoints to attain local optima, our method grows the trajectories from initial conditions with a policy that has global knowledge of the desired trajectory topology. Similar to gradient-based optimization techniques, our method cannot guarantee the attainment of global optima. However, the physical encoding of the topological specification into the planning mechanism results in satisfactory performance for a class of problems.

Planner	Success (%)	No. of iterations	Time (s)
CHOMP	78.80	80.3325	0.1291
HTTG	98.40	86.8862	0.0048

Table 2: Success rates and computation times of HTTG and CHOMP over 500 randomly generated 2-agent scenarios.

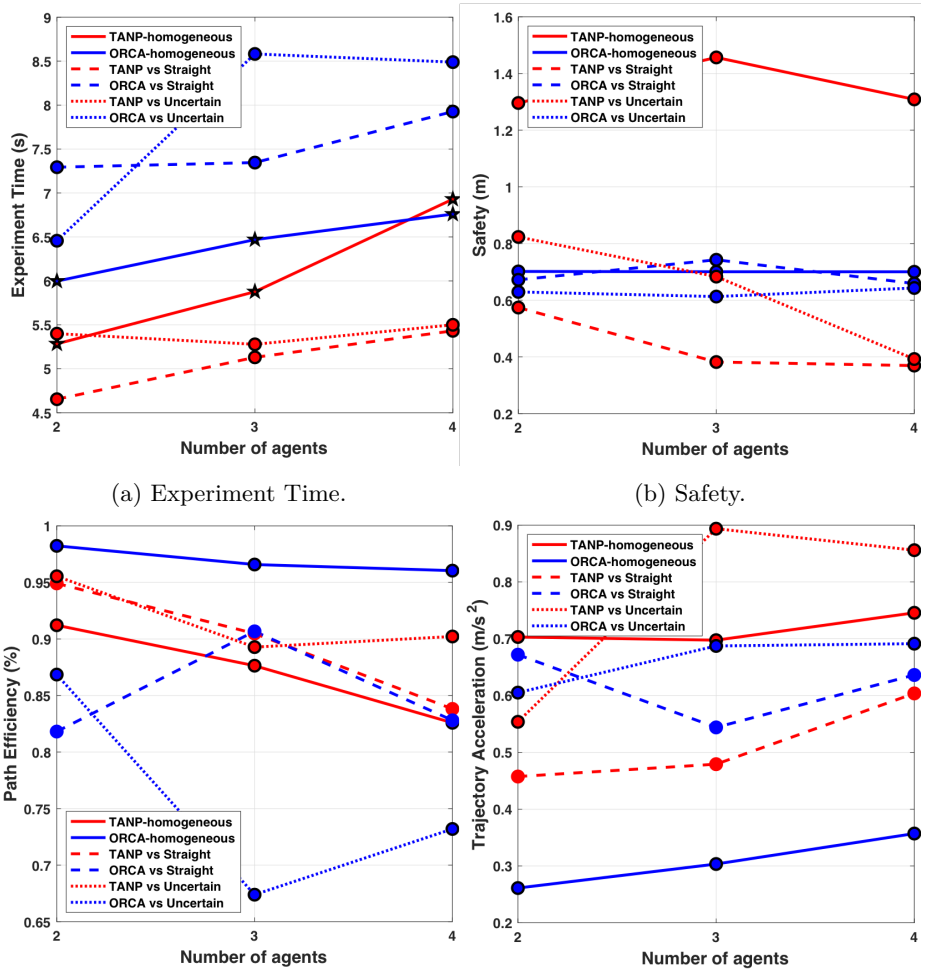
To illustrate the difficulty of automatically synthesizing multi-agent trajectories of desired topological specifications through trajectory optimization techniques, we consider a simple case study, in which we compare the performance of HTTG with the performance of CHOMP [21]. We randomly generate 500 different scenarios involving 2 agents navigating towards opposing sides of a circular workspace (workspace has 5m diameter, starting positions are uniformly distributed along the circumference, speed normally distributed between $0.3m/s$ and $1.5m/s$ for each agent). For each scenario, we randomly sample a passing side that agents should pass one another. To encode the objective of respecting a passing side to CHOMP, we construct a cost functional $\mathcal{F}_{top} = \frac{1}{2}(w_{ab} - w_{des})^2$ which approaches zero as the winding number of agents' trajectories w_{ab} ap-

proaches the winding number corresponding to the desired passing side, w_{des}. Table 2 illustrates the performance of the two approaches, which is measured with respect to success rate and computation time (non-optimized MatLab implementation on a MacBook Pro of 2015 with an Intel Core i7 processor of 2.5 GHz, running macOS High Sierra). For CHOMP, a trial is considered successful if it generates trajectories of the desired topology within 500 iterations whereas for HTTG, a trial is considered successful if the desired topology is achieved once the agents reach their destinations. It can be observed that HTTG dominates with a success rate of 98.40% (corresponding to 492/500 successful trials). The computation time is comparable in terms of iteractions but HTTG requires almost two orders of magnitude less time in seconds. The benefits provided by HTTG in terms of success rate and computation time make the consideration of multiple trajectory topologies at planning time a more practical strategy.

4.3 Online Performance

In order to demonstrate the benefits of our online algorithm 1, we perform a simulation study comprising a series of experiments on the circular workspace considered in the previous sections (diameter 5m). We consider 9 different experiment configurations, each corresponding to a different group of navigating agents. In particular, we consider groups of 2, 3 and 4 agents, navigating under three different conditions: (a) a homogeneous condition –all agents run the same planner; (b) a heterogeneous condition in which one agent runs our planner and others are moving straight to their goals without avoiding collision; (c) a heterogeneous condition in which one agent runs our planner and others are changing intentions over a destination twice, without avoiding collisions. Note that the two latter cases are are particularly challenging for decentralized planners, as a typical assumption they rely heavily on is homogeneity. For reference, we perform the same experiments with the Optimal Reciprocal Collision Avoidance (ORCA) [20] (clearance and speed parameters tuned similarly to ensure a fair comparison). We quantify the performance of the planners with respect to four aspects of trajectory quality: (1) Experiment time, measured as the amount of time that the last agent to reach its destination took; (2) Safety, measured as the minimum distance between any two agents for the homogeneous condition and as the minimum distance between a TANP/ORCA agent and any other agent for the heterogeneous conditions; (3) Path Efficiency, measured as the ratio between the length of the optimal path to goal and the length of the path a TANP/ORCA agent followed (averaged over agents in the homogeneous case); (4) Trajectory Acceleration, measured as the average acceleration per time step per TANP/ORCA agent throughout the experiment.

Fig. 4 depicts the performance of TANP and ORCA under each of the configurations considered. For each configuration, each planner executed the same set of 200, randomly generated scenarios. Overall, TANP exhibits the best time-efficiency for almost all configurations (Fig. 4a). When executed under homogeneous settings, TANP establishes a consistently high clearance from others, which results in a drop in terms of path efficiency (Fig. 4c) and a high acceleration per time step (Fig. 4d). The increased time efficiency of TANP could be attributed to the implicit consensus that is reached through the considera-



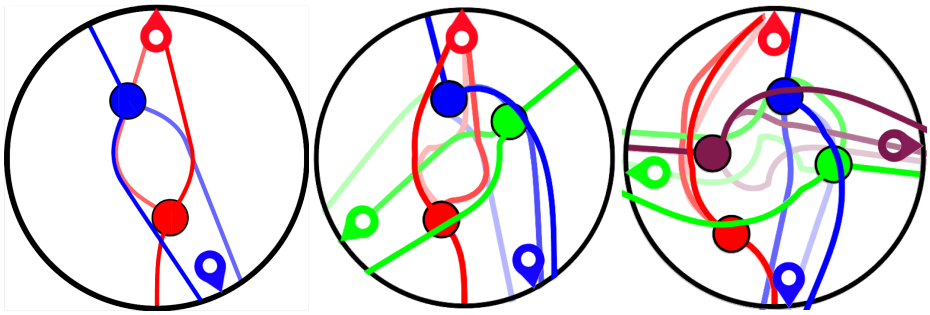
(a) Experiment Time.

(b) Safety.

(c) Path Efficiency.

(d) Trajectory Acceleration.

Fig. 4: Trajectory Quality for all experiment configurations considered. For group size, the same 200 randomly generated scenarios are executed under each of the conditions considered with both planners. For each condition and measure, we perform a paired Student's t-test to compare the populations yielded by TANP and ORCA. Points with black circular boundaries indicate rejection of the null hypothesis with p-value < 0.001 whereas points with star boundaries indicate rejection of the null hypothesis with p-value < 0.05.



(a) Two agents. (b) Three agents. (c) Four agents.

Fig. 5: Overlaid predictions made by a TANP agent (red color) as it navigates towards the red landmark in environments with 2, 3 and 4 agents.

tion of joint strategies of collision avoidance. The price the TANP agents pay is increased accelerations and generally lower path efficiency. On the other hand, ORCA is consistently slower but stably safe across all conditions. Under the homogeneous condition, it achieves the highest path efficiency and lowest acceleration, which was expected by its optimality-driven design. This efficiency advantage fades under the heterogeneous conditions, in contrast to TANP, which demonstrates a more balanced behavior.

5 Discussion & Future Work

We presented an online planning framework for the generation of adaptive robot motion in dynamic environments where multiple other agents navigate by executing generally different policies. Our framework is based on an offline planner that generates a diverse set of multi-agent trajectory predictions. Each prediction prescribes a different, collision-avoiding behavior to the planning agent. The agent selects the prediction of lowest cost and executes the first action from it. This architecture enables an agent to make local decisions with a global outlook that allows for anticipation of any upcoming agent interactions and rapid adjustment to them. Simulated examples demonstrate the performance of the offline and online parts of our framework. Future work involves (a) evaluating our algorithm in environments of more complex geometry, (b) reducing its computational load by designing a mechanism that efficiently reuses past trajectories and (c) performing real-world experiments on a robot platform navigating in human environments.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants IIS-1526035 and IIS-1563705. We are grateful for this support.

Bibliography

[1] H. Aref. Point vortex dynamics: A classical mathematics playground. *Journal of Mathematical Physics*, 48(6):065401, 2007.

[2] H. Aref, S. Jones, S. Mofina, and I. Zawadzki. Vortices, kinematics and chaos. *Physica D: Nonlinear Phenomena*, 37(1):423 – 440, 1989.

[3] M. A. Berger. Hamiltonian dynamics generated by Vassiliev invariants. *Journal of Physics A: Mathematical and General*, 34(7):1363, 2001.

[4] M. A. Berger. Topological invariants in braid theory. *Letters in Mathematical Physics*, 55(3):181–192, 2001.
[5] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1343–1350, 2017.
[6] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51:4282–4286, May 1995.
[7] I. Karamouzas, B. Skinner, and S. J. Guy. Universal power law governing pedestrian interactions. *Physical Review Letters*, 113:238701, 2014.
[8] B. Kim and J. Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8 (1):51–66, 2016.
[9] R. A. Knepper and D. Rus. Pedestrian-inspired sampling-based multi-robot collision avoidance. In *Proceedings of the 2012 IEEE International Symposium on Robot and Human Interactive Communication*, pages 94–100, 2012.
[10] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.
[11] W. C. Ma, D. A. Huang, N. Lee, and K. M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4636–4644, 2017.
[12] C. I. Mavrogiannis and R. A. Knepper. Decentralized multi-agent navigation planning with braids. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2016.
[13] C. I. Mavrogiannis and R. A. Knepper. Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research*, 2018.
[14] C. I. Mavrogiannis, V. Blukis, and R. A. Knepper. Socially competent navigation planning by deep learning of multi-agent path topologies. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6817–6824, 2017.
[15] C. I. Mavrogiannis, W. B. Thomason, and R. A. Knepper. Social momentum: A framework for legible navigation in dynamic multi-agent environments. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 361–369, 2018.
[16] M. Moussaïd, D. Helbing, and G. Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011.
[17] C. Rösmann, F. Hoffmann, and T. Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88: 142 – 153, 2017. ISSN 0921-8890.
[18] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Siméon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.
[19] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation. *International Journal of Robotics Research*, 34(3):335–356, 2015.
[20] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Proceedings of the International Symposium on Robotics Research*, pages 3–19, 2009.
[21] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. D. Bagnell, and S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research*, 2013.

A Trajectories Generated with HTTG

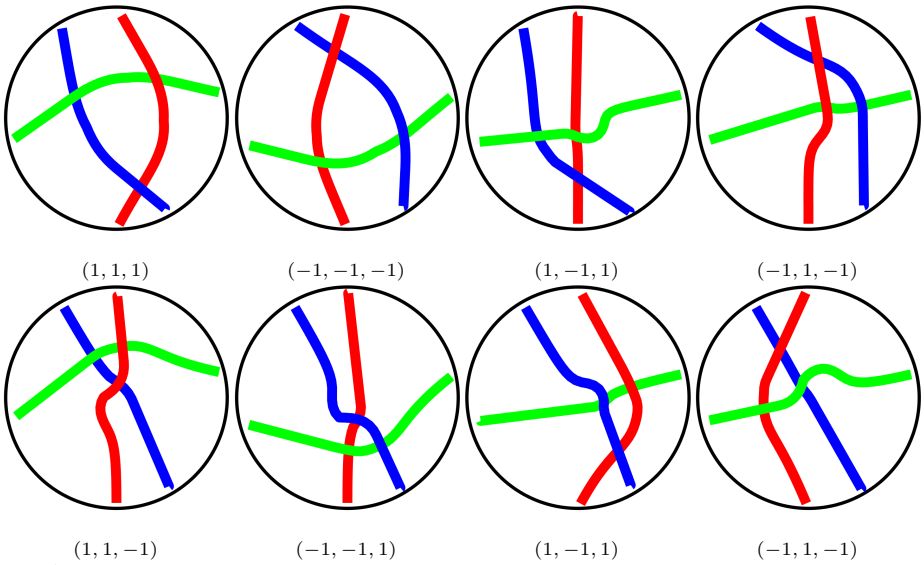


Fig. A.1: Top view of trajectories generated by executing the same 3-agent scenario with all possible topological specifications. The subcaptions denote the topology tuple that was used as a specification for each execution.

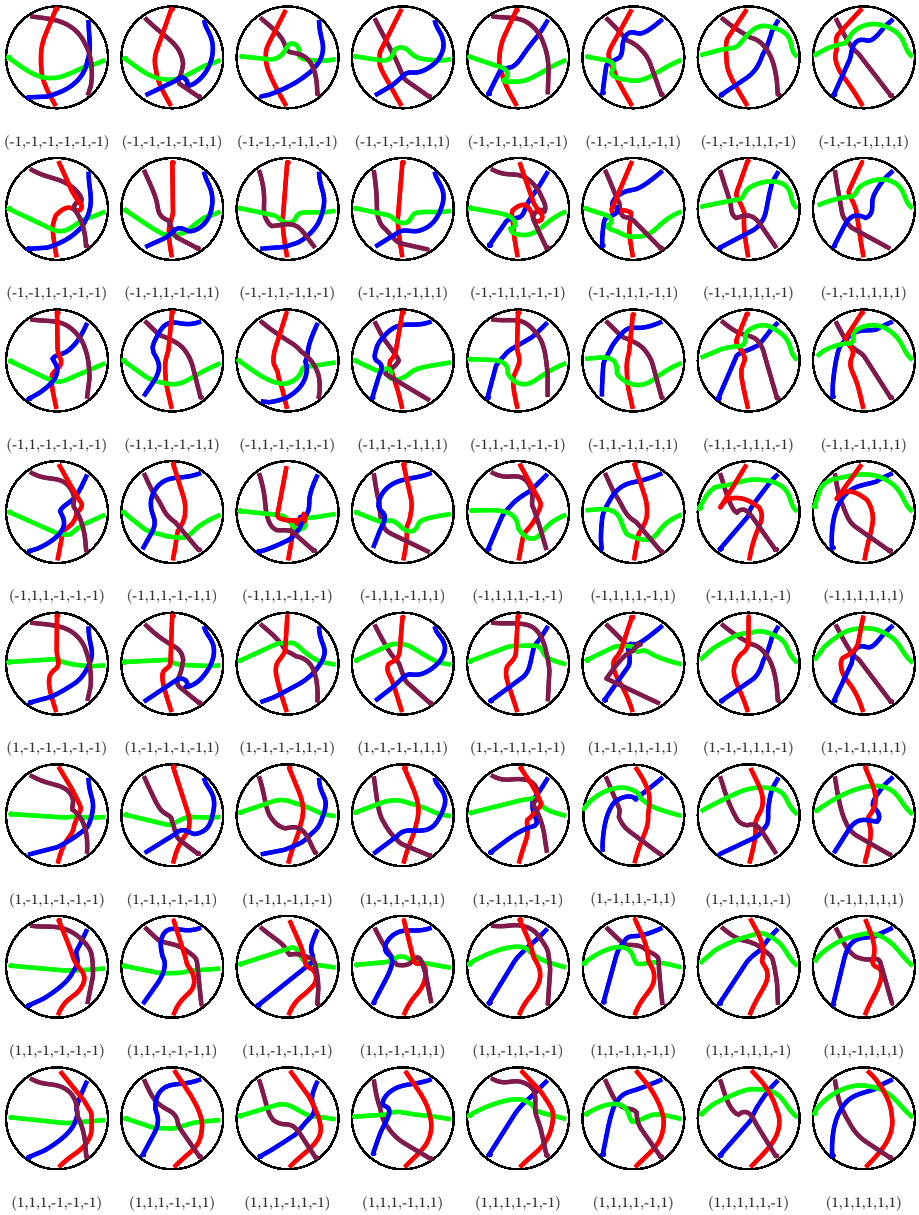


Fig. A.2: Top view of trajectories generated by executing the same 4-agent scenario with all possible topological specifications. The subcaptions denote the topology tuple that was used as a specification for each execution.