Measure Task-Level Inter-Agent Interaction Difficulty in Decentralized Scenarios, with Scenario Generalization Approximation

Gang Qiao[1*] Kaidong Hu[1] Seonghyeon Moon[1] Sejong Yoon[2] Mubbasir Kapadia[1] Vladimir Pavlovic[1]
[1]Rutgers University [2]The College of New Jersey
{gq19, sm2062, mk1353, vladimir}@cs.rutgers.edu kaidong.hu@gmail.com yoons@tcnj.edu

Abstract

Motivated by the need to understand the central complexity in problems like multi-agent path finding and generalizing models in new scenarios, we present an algorithm that measures task-level inter-agent interaction difficulties in decentralized crowd scenarios. The algorithm (i) views a crowd simulation as a transformation of the parameter of a steering model, (ii) captures the reduced amount of abstracted trajectory classes. We exploit the measurement to approximate the scenario generalizations of learning models. Experiments validate the efficacy of the measurement in characterizing interaction difficulty, and the potential to select domains before actually training and testing a model.

1 Motivation

In the context of decentralized crowd simulation [Qiao *et al.*, 2018], a scenario refers to the configuration of obstacles in an environment and the tasks of all agents in that environment. The task of an agent refers to the initial and destination positions of the agent, the starting time when the agent presents into the environment, the maximal number of simulation steps allowed, and the radius of the agent. The interaction among agents refers to the effort made by the agents to (i) follow their individual planned paths (nodes) and (ii) avoid collisions from each other. Given a decentralized scenario, some tasks will inevitably "encounter" more agents, hence potential inter-agent collisions, than other tasks despite the steering model been chosen. Such inherent task-level inter-agent interaction difficulty introduces the central complexity for problems including Multi-Agent Pathfinding, and scenario generalization (SG) of learning models [Qiao *et al.*, 2019]. To this end, an algorithm is presented. It takes the full information about a scenario as input and outputs a scalar for each agent (task) in the scenario, indicating the task-level interaction difficulty for that agent with all other agents, despite the steering model being chosen. We further use the measurement to approximate scenario generalization.

2 Approach to Measurement

An overview of the proposed method is illustrated in Figure 1.

2.1 Crowd Simulation

Given a scenario with obstacle configuration $\mathcal{E}$, suppose there are n agents in the scenario. Denote all agents as $X_{1\sim n} := (X_1, X_2, ..., X_n)$[1]. We aim to estimate the task-level interaction between an agent X_i and all other agents in the scenario, which are denoted as $X_{-i} := (X_1, ..., X_{i-1}, X_{i+1}, ..., X_n)$ [2], for i = 1, 2, ..., n.

Since our goal is to estimate task-level rather than model-level (how a particular model performs) interaction between agent X_i and agents X_{-i}, it is essential to avoid specifying a steering model with a fixed parameter for each agent. Instead, we model the uncertainty of the behavior of agent X_i by assuming the parameter of the model being a random variable, denoted as Θ_i, that obeys some distribution $p_i(\theta_i)$, i = 1, 2, ..., n. For notation simplicity, denote $\Theta_{1\sim n} := (\Theta_1, \Theta_2, ..., \Theta_n)$.

For a given scenario, at task-level crowd simulation, the input of the crowd system is $\Theta_{1\sim n}$ and the output is a tuple of interactive trajectories, denoted as $T_{1\sim n} := (T_1, T_2, ..., T_n)$, where T_i is the trajectory of agent X_i driven by $\Theta_{1\sim n}$. The relationship between the input and the output can be represented as $f(\Theta_{1\sim n}|\mathcal{E}, X_{1\sim n}, \mathcal{M}) = T_{1\sim n}$, conditioned on the obstacle configuration $\mathcal{E}$, the tasks of all agents $X_{1\sim n}$ from the given scenario, and the steering model family $\mathcal{M}$. The transformation function $f(\cdot)$ provides an alternative view of the input and output of a crowd simulation.

2.2 Trajectory Abstraction

To improve the representational efficiency for characterizing the relationships among trajectories of different agents, it is necessary to abstract the trajectory T_i of agent X_i to a finite number of classes. Each class contains realizations of T_i of the same modality, while realizations from different classes present different modalities, i=1, 2, ..., n.

To achieve the trajectory abstraction, besides running the simulation involving all agents at the random parameter $\Theta_{1\sim n}$, we additionally run one simulation for agent X_i from its initial to its destination position, with obstacle configurations but no other agents, at an appropriately selected steering parameter θ_i^*, for i = 1, 2, ..., n. This results in a deterministic solo trajectory for agent X_i, denoted as s_i, which could be viewed as agent X_i's ideal trajectory in the sense that no extra effort is needed to avoid inter-agent collisions.

Therefore a generic function can be applied to compute the difference between the trajectory T_i of agent X_i and its solo trajectory s_i, which is a quantification of the effort that agent X_i makes for interaction with the rest agents X_{-i}. As a typical choice, Dynamic Time Warping (DTW) that accumulates

[1]At task level, X_i refers to both the i-th agent and the task of the i-th agent, and we use the two meanings of X_i interchangeably.

[2]This does not mean that during the crowd movement, there are no interactions among agents X_1, ..., X_{i-1}, X_{i+1}, ..., X_n.

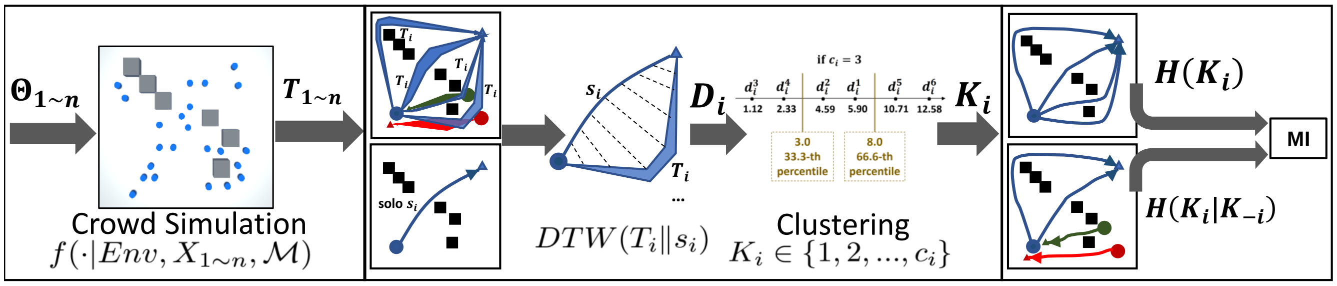

Figure 1: Diagram for measuring task-level inter-agent interaction (in this example, between the blue agent and other agents). It consists of a crowd simulation block, a trajectory abstraction block and a Mutual Information (MI) block. The crowd simulation block takes in a random parameter and generates random interactive trajectories of all agents. The trajectory abstraction block compares the trajectory of an agent (e.g., the blue agent) and its solo trajectory to yield DTW difference, which goes through a clustering process and outputs an abstracted trajectory index. The MI block measures the reduced amount of the abstracted trajectory classes of the agent resulting from the interaction with other agents. An option of an agent is shown with a curve, representing a class of similar trajectories for the agent. The destination of an agent is shown with a triangle of the same color. Black rectangles in the maps are obstacles.

distances over all pairs of aligned states [Salvador and Chan, 2007], yields the DTW difference: $D_i = DTW(T_i||s_i)$, which embodies the extra expenditure of agent X_i in the trajectory T_i due to the interaction with the rest agents X_{-i}.

D_i is further transformed to an abstracted trajectory index $K_i \in \{1, 2, \ldots, c_i\}$ by a clustering process, denoted by $K_i = g(D_i)$. We choose a simple method that defines the clusters as the proper percentiles on the cumulative distribution function (CDF) of D_i. The uniform partitions along the probability dimension of the CDF of D_i create the corresponding non-uniform partitions (bins) along the DTW dimension, and the index of the partition (bin) that D_i falls into on the DTW dimension is the K_i. Thus, K_i is an abstraction of the possible trajectory T_i of agent X_i. For instance, in the trajectory abstraction block of Figure 1, K_i abstracts T_i of the blue agent into four classes (an abstracted class of an agent represents a set of similar trajectories for the agent, in the sense of their DTW differences), and indexes them with the support $\{1, 2, 3, 4\}$.

The trajectory abstraction block outputs the abstracted index tuple $K_{1\sim n} := (K_1, K_2, ..., K_n)$ at $\Theta_{1\sim n}$[3]. The rationale is that in a scenario, we represent the possible abstracted class of a task X_i with an index K_i, and a simulation as a co-occurrence of these abstracted indexes $K_{1\sim n}$.

2.3 Computing interaction

To exploit the co-occurrence of the abstracted indexes among agents in estimating the inter-agent interaction, we use the mutual information (MI):

$$\begin{aligned} I(K_i; K_{-i}) &= H(K_i) - H(K_i|K_{-i}) \\ &= \underset{(K_i, K_{-i})}{\mathbb{E}} \left[\log \frac{P(K_i, K_{-i})}{P(K_i)P(K_{-i})} \right] \end{aligned} \tag{1}$$

In Equ.(1), $H(K_i)$ measures the uncertainty in predicting agent X_i's abstracted trajectory class, while $H(K_i|K_{-i})$ measures the uncertainty in predicting agent X_i's abstracted trajectory class influenced by the classes of all the rest agents. Thus MI characterizes the influence of knowing agents X_{-i}'s abstracted trajectory index tuple in predicting agent X_i's abstracted trajectory class. This intuition is further pictured in the MI block of Figure 1, where the figure above illustrates the possible trajectory classes (indexes) for agent X_i (the blue agent) when the other agents' possible trajectory classes are unknown, while the figure below shows how agent X_i's possible trajectory classes are restricted by other agents' trajectory classes. The reduced amount of the abstracted trajectory classes of agent X_i reflects the interaction between agent X_i and agents X_{-i}, in the given scenario.

[3]Note that K_1=1 and K_2=1 are two different events. The abstracted index enumerates the event set of each individual agent.

3 Experiment Design for Evaluation

We design two sets of experiments to systematically verify the efficacy and utility of the algorithm. The first set of experiments verifies the efficacy of the measurement in characterizing the interaction difficulty of a scenario. This set of experiments consists of two parts. Part-I compares the measurement and a baseline on designated scenarios to demonstrate that the measurement presents advantages over the baseline on anticipated aspects. Part-II compares the measurement and the baseline in three data domains, both qualitatively and quantitatively. The second set of experiments aims to verify the utility of the measurement to approximate scenario generalization, and conducts a comparison with the baseline.

In the supplementary, we describe how to exploit the proposed measurement to estimate scenario generalization. We also provide part of the experiment results on Egocentric Representative (G) domain, by qualitatively comparing the measurement with the baseline [Berseth *et al.*, 2013].

4 Summary

We propose an algorithm to measure task-level inter-agent interaction in decentralized crowd scenarios, and exploit the measurement to estimate the scenario generalization of a learning model in crowd simulation. Experiment results validate the efficacy in characterizing interaction difficulty. In addition, the consistency with the ranking of true scenario generalizations on multiple candidate domains implies the potential of the measurement in helping to select suitable training and testing domains, before actually training and testing a model.

References

[Berseth *et al.*, 2013] Glen Berseth, Mubbasir Kapadia, and Petros Faloutsos. Steerplex: Estimating scenario complexity for simulated crowds. In *Proceedings of Motion on Games*, pages 67–76. 2013.

[Bhattacharya, 2010] Subhrajit Bhattacharya. Search-based path planning with homotopy class constraints. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[Qiao *et al.*, 2018] Gang Qiao, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. The role of data-driven priors in multi-agent crowd trajectory estimation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Qiao *et al.*, 2019] Gang Qiao, Honglu Zhou, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Scenario generalization of data-driven imitation models in crowd simulation. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG)*, 2019.

[Salvador and Chan, 2007] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

Supplemental Materials

S1 Scenario Generation and Approximation

S1.1 Scenario Generalization

[Qiao *et al.*, 2019] proposed the notion of scenario generalization (SG), which is different from regular generalization. In crowd simulation, the regular generalization involves an expert model and an imitation model. The input sample of the imitation model could be a local observation of an agent during the crowd movement, and the output of the imitation model could be the velocity in that observation. The goal is to make the *step-wise* predicted velocity of the imitation model as close as possible to the *step-wise* demonstrated velocity of the expert model in test scenarios (e.g., minimize L2 norm of the difference between the expert's velocity and the imitator's velocity on a test local observation). To this end, both training and testing samples (local observations) are collected along expert trajectories by running the expert model on the training and testing scenarios. During test, agents with the trained imitation model move along the expert trajectories and make velocity predictions based on their local observations. When the imitation model makes a velocity prediction, the corresponding agent does not move according to that velocity, but *moves along the expert trajectory*. In a word, in order to compare the velocity outputs of the expert model and the imitation model based on the same local observation, the imitation model does not make sequential decisions during test phase.

In contrast, scenario generalization is a macroscopic view of the performance of a model. Under the macroscopic view, a model here involves not only an imitation model trained from an expert model, but could also cover a traditional reinforcement learning model with a pre-defined reward function. Although a model still takes in the local observation of each agent at each step and makes a velocity prediction for that agent at that step, the whole crowd simulation (movement) could be viewed as providing a scenario to the model, running the model, and obtaining a tuple of trajectories for agents in that scenario. Thus, the macroscopic input sample of a model is a scenario, and the macroscopic output of the model is a tuple of trajectories. The goal is to make application-oriented metrics such as the number of agent-agent collisions, the number of agent-obstacle collisions, the total travel distances of the model trajectories, to be as good (low) as possible in test scenarios. Fundamentally, the trained model makes sequential decisions on a test scenario and yield its own complete tuple of trajectories.

From an application point of view, scenario generalization focuses on a model's performance on a test domain, regardless of the model's performance on its training domain. Thus scenario generalization can be directly characterized by:

$$\epsilon_T(h) = \int_{\chi} p_T(x) \cdot \text{cost}(h(x))\, dx \approx \frac{1}{m}\sum_{i=1}^{m} \text{cost}(h(x_i)) \tag{S1}$$

where T denotes a target domain, with the distribution of scenarios being $p_T(x)$. x denotes a scenario in the scenario space χ. h is a model already trained on some source domain. $\{x_i\}_{i=1}^m$ are m scenarios drawn from the target domain. For each sampled scenario x_i, the model h is applied to make sequential decisions for agents within x_i in a decentralized manner, yielding a tuple of trajectories, on which true cost can be directly evaluated. For instance, one may choose the amount of agent-agent collisions within the tuple of model trajectories as the cost.

S1.2 Approximating Scenario Generalization

As shown in Equ.(S1), the direct evaluation of scenario generalization of a model h involves training the model on some source domain S, and then testing the trained model on the target domain T. However, training a model using either reinforcement learning paradigm or imitation learning paradigm requires repeated trials with hyper-parameter turnings, which might be difficult and time-consuming.

In addition, one may have more than one source-target domain pair. If one has a single target domain and multiple candidate source domains, usually the goal is to select the best source domain to train a model such that the model has the highest scenario generalization (performance on the target domain), compared with models trained on other source domains. If one has a single source domain and multiple candidate target domains, a common goal is to select the best target domain on which the trained model has the highest performance, compared with other target domains.

The issues motivate us to approximate scenario generalization *before* actually training and testing a model, by applying the measurement on the source and target domains. Specifically, the true cost term in Equ.(S1) is decomposed into two independent terms. One term contains the interaction difficulty of the testing scenario itself, the other encodes the influence of the source domain on the test performance:

$$\begin{aligned} \epsilon_T(h) &= \frac{1}{m}\sum_{i=1}^{m} \text{cost}(h(x_i)) \\ &\approx \frac{1}{m}\sum_{i=1}^{m} [MI(x_i) + \lambda Q_i(S)] \\ &= [\frac{1}{m}\sum_{i=1}^{m} MI(x_i)] + \lambda Q(S) \end{aligned} \tag{S2}$$

In the second line of Equ.(S2), the true cost of the model h on the test scenario x_i is approximated by the linear combination of the interaction difficulty of the test scenario x_i itself, and the influence of the source domain S on the current test, denoted as $Q_i(S)$, with λ adjusting the relative importance between the two terms. The influences of the source domain S on all the tests are averaged into $Q(S)$. $Q(S)$ may encode several intrinsic aspects of the source domain S that are significant in the performance of a trained model. In our study, we measure the diversity of scenarios drawn from the source domain, based on the intuition that a model exposed to a higher variation of scenarios during the training phase would be more adaptive to unseen varying scenarios. The computation of the diversity is introduced in subsection S1.3.

Though scenario generalization from a source domain S to a target domain T is approximated in Equ.(S2), usually the practical goal is to support decision making in selecting a domain among multiple candidate domains by comparing their estimated scenario generalizations. In the case of a single target domain and multiple source domains, with the goal to select the best source domain to train a model, according to Equ.(S2), we only need to select the source domain with the smallest $Q(S)$ term, since the MI term of the target domain keeps the same for all source domains. Similarly, in the case of a single source domain and multiple target domains, to select the best target domain for a trained model, we only need to select the target domain with the smallest MI term.

S1.3 Diversity of Scenarios within Domain

We suggest to exploit entropy to characterize the variation in the scenarios sampled from a source domain. First, the obstacle configuration, denoted as Env, may vary within the samples. Assume that among m sampled scenarios from a source domain, there are L different types of obstacle configurations. One may estimate the probability (frequency) of the occurrence of each type of obstacle configuration, and use the entropy to measure the diversity of Env:

$$H(Env) = -\sum_{l=1}^{L} p(env_l)\log p(env_l) \tag{S3}$$

Second, the initial position of an agent, denoted as I, may vary within the samples. One may divide the spatial space of the movement environment into, say, N cells and estimate the probability (frequency) of each cell that the initial position of an agent falls into, in view of all agents from the source domain as independent and identically distributed. The same method applies to the destination position of an agent, denoted as D. One may further estimate the diversity of the paired initial position I and destination position D of an agent with joint entropy:

$$H(I, D) = -\sum_{i=1}^{N}\sum_{j=1}^{N} p(i_i, d_j)\log p(i_i, d_j) \tag{S4}$$

Third, the diversity of initial position I, destination position D and obstacle configuration Env can be jointed:

$$\begin{aligned} H(I, D, Env) &= H(Env) + H(I, D \mid Env) \\ &= H(Env) + \sum_{l=1}^{L} p(env_l) H(I, D \mid Env = env_l) \end{aligned} \tag{S5}$$

Lastly, the higher the variation in the scenarios of a source domain, the lower the true test cost would be:

$$Q(S) = -H(I, D, Env) \tag{S6}$$

Note that scaling the spatial space of environments of different source domains may be necessary in order to compare them. Also, the start time when an agent presents into the environment could be incorporated in the formulation, depending on the amount of scenario samples.

S2 Experiment Results on G domain

We present qualitative results and analysis on Egocentric Representative (G) domain, from which scenarios are sampled to embody small-scale while general inter-agent interactions. The positions of obstacles and the initial/destination positions of agents obey a uniform distribution in the 2D spatial space. A scenario contains about 25 agents.

The left figure of Figure S1 visualizes the per-agent results of the measurement in a scenario. The marks on the agents reveal that (i) the measurement identifies tasks that are spatially isolated from other tasks. These tasks, indicated by a textbox with “a” inside, are either short, or spatially far away from other tasks; (ii) the measurement captures temporally isolated tasks, indicated by a textbox with “b” inside. Even though these tasks may seemingly have intersections with other trajectories in the 2D spatial space, in the 3D spatial-temporal space, such tasks allow the agents with regular speed to temporally avoid other agents, by reaching the intersections earlier or later; (iii) the measurement reflects the interaction of other tasks, except for the noise introduced by an agent stuck by obstacles, indicated by a textbox with “c” inside, with its destination circled by an eclipse. Overall, the measurement can distinguish tasks in the same scenario.

Corresponding to the visualization of the measurement, the right figure of Figure S1 illustrates the per-agent baseline results on the same scenario, where a detected interaction is denoted with a small circle comprised of two rings belonging to two interacted trajectories of the corresponding colors. We can see that even though the baseline avoids the noise encountered by the measurement when an agent is stuck and unable to reach the destination, illustrated with the big green circle, detailed marks are not as good as the measurement in distinguishing tasks within a scenario. For instance, the big black circle with baseline value 2 inside designates the baseline results that are noticeably lower than intuition, compared with the planned green path on the upper part of the figure.

On the other hand, the big yellow circles designate the baseline results that are higher than intuition. For instance, one blue planned path highlighted with a big yellow circle is marked with a higher-than-intuition value of 4, which might result from the discritization error when planning diagonally. For the other three planned paths highlighted with a big yellow circle, the cause might be that if there are multiple detected interactions, the consequence of preceding interactions is not able to be reflected in the subsequent interactions. This can be seen that if the blue planned path highlighted with a big yellow circle and marked with the value of 3 could detour a little to avoid the pink agent, consequently this blue agent may also avoid the later spatial-temporal encounter with the green path marked also with the value of 3 and highlighted with a big yellow circle. Besides, the red path highlighted with a big yellow circle and marked with the value of 4 could have chosen a path belonging to a different homotopy [Bhattacharya, 2010] under the influence of other agents. A detailed example about homotopy is illustrated in Figure S2. The right figure of Figure S2 shows that the baseline marks a task higher than intuition due to the homotopy inflexibility: baseline plans a path for a task in advance with A-star and detects interactions on these fixed planned nodes. While for the measurement, even though it also plans a path in advance with A-star, during simulation the agent does not strictly follow the planned nodes owing to the repulsive forces from obstacles and other agents, and may yield a trajectory belonging to a different homotopy class, shown in the left figure of Figure S2. The homotopy diversification during simulation in the measurement could better reflect the inherent interaction difficulty of a task.

Overall, the measurement distinguishes different tasks in scenarios of G domain better than the baseline.

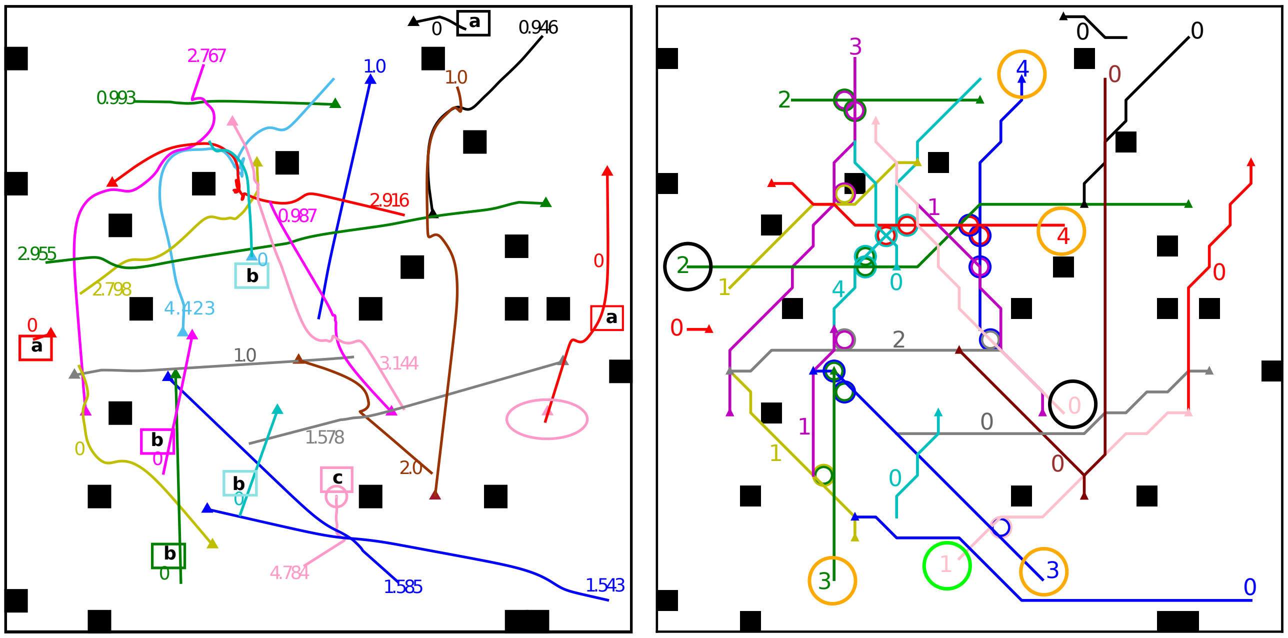

Figure S1: Visualizations of the per-agent measurement (left figure) and the per-agent baseline (right figure) in a scenario of G domain. In both figures, black rectangles represent obstacles. A numeric number labeled near a task of the same color denotes the inter-agent interactivity for that task. In the left figure, the trajectories are simulated at θ^* and ended with triangles. In the right figure, smaller circles denote the detected interactions by the baseline.

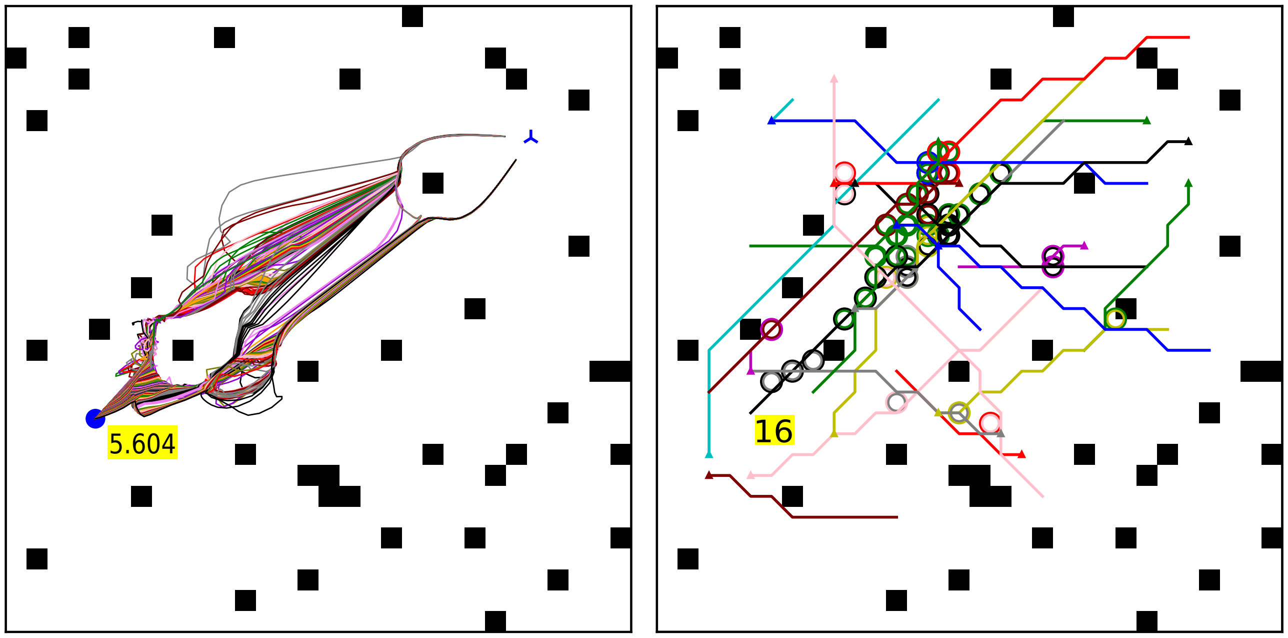

Figure S2: Comparison of a single task with homotopy in a scenario of G domain. The task marked by the measurement (left figure) is 5.604, which is about 2.6 times of the average interactivity (2.160) over all tasks in that scenario. The same task marked by the baseline (right figure) is 16, about 3.67 times of the average detected interactions (4.364) over all tasks in the scenario. This implies that the baseline marks this task with a higher value, due to homotopy inflexibility.