

Article

Receding Horizon Multi-Agent Deceptive Path Planner

Xubin Fang ¹, Brian M. Sadler ² and Rick S. Blum ^{1,*}

¹ Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015, USA; xuf220@lehigh.edu

² Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA; brian.sadler@ieee.org

* Correspondence: rblum@eecs.lehigh.edu

Abstract

Deceptive path planning enables autonomous agents to obscure their true goals from observers by deviating from an expected optimal path. Prior work largely solves full-horizon, end-to-end optimization for single agents, so planning complexity grows with the path length and size of the environment, and adaptation en route requires a new optimization. We propose a unified framework for deceptive path planning onboard the agent, computing over short-horizon candidate paths within a receding-horizon loop. The method processes an agent's position sensor information and develops a policy for the next trajectory plan. By parameterizing a user-defined cost that captures optimal planning and deception (and optionally includes constraints, trajectory smoothness, and coupling terms between agents), a Boltzmann framework yields stochastic policies that balance the tradeoff between optimal paths and deceptive deviation. Policies are updated locally and do not require learning or training. The level of deception and adherence to constraints can be dynamically tuned, enabling online adaptation to changes in goals and constraints. This step-by-step tuning opens the door to new forms of dynamic deception. In the multi-agent case, we develop a joint planner that is also tunable and can be applied as desired among all or subsets of agents. Single- and multi-agent simulation studies demonstrate the flexibility of our approach, maintaining deception while adapting as desired, avoiding the recomputation required by full-horizon methods, and supporting intuitive tuning via a small set of parameters.

Keywords: deceptive path planning; dynamic deception; multi-agent systems; boltzmann distribution; receding-horizon planning; adaptive planning; resource constraints



Academic Editor: Andrey V. Savkin

Received: 21 May 2026

Revised: 1 September 2026

Accepted: 11 September 2026

Published: 17 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Path planning is fundamental for multi-agent coordination and control [1]. Examples include full coverage path planning [2], planning on graphs for patrolling, routing, and surveillance [3], and search [4]. Time constraints can be incorporated with linear temporal logic [5], and planned rendezvous can be included for persistent networking and information transfer [6]. Local trajectory planning with nearest neighbor communications can provide adaptive collision avoidance [7].

In this paper, we consider deceptive path planning (DPP), where agents obscure their true goals from observers. Deceptive paths are characterized by deviation from an optimal (e.g., shortest) path to the intended goal, using exaggeration, ambiguity, or other deceptive measures aligned with false goal states. The adversary is modeled as a passive observer who assumes agents will follow an optimal path. DPP is important in adversarial scenarios

where an observer may attempt to predict the agent's destination goals. This applies to a variety of domains, such as resource allocation, human–robot interaction under adversarial observation, and autonomous delivery and logistics in competitive environments [8–10]. Much of the existing research on DPP focuses on single-agent cases and generating one deceptive path, assuming a known environment [8,10–16]. The complexity of end-to-end planners grows with the path length and environment size, and requires replanning when the environment, deception type, constraints, or other system parameters change [9,17,18].

We present a general receding horizon deceptive path planning framework that can be applied in single- and multi-agent settings and enables dynamic adaptation. The method relies on local sensing of position and obstacles and plans the local trajectory or action sequence. To model decision-making, we construct a probability mass function (PMF) based on a Boltzmann distribution over candidate trajectories, which is parameterized by a user-specified cost $C(\cdot)$. This PMF defines a stochastic policy from which steps are sampled and sequentially added to the path. A multi-agent formulation is obtained by lifting this distribution to the joint space of agents' trajectories, and costs can be coupled among groups of agents as desired. At each iteration, the planner performs a finite-step lookahead. The goals, false goals, constraints, and form of deception can all be updated at each planning iteration. The cost function $C(\cdot)$ and its parameters are updated (if desired), the policy is updated, and a new finite-step lookahead begins. This receding-horizon loop repeats until the agent reaches its goal. The approach is simple to implement, allows for easy modification of costs and tuning of the level of deception at each iteration, is robust to local environment dynamics, and generalizes to multiple agents.

The main contributions of this paper are:

- A DPP framework that uses local position information and enables adaptation to a dynamic environment, constraint changes, goal modifications, and moving goals, without end-to-end replanning. The proposed planning framework is simple and could be integrated into standard navigation pipelines.
- A flexible Boltzmann-based formulation for local random policy design that enables agents to adapt their deceptive behavior with time-varying intuitive parameters.
- A logical extension from single to multi-agent scenarios with coupled costs and/or incentives while allowing individual agent variation in goals and parameter tuning. Agents collaborate by sharing a limited parameter set.
- A generalizable approach that can handle various DPP scenarios and extends to other path planning problems by adjusting the cost functions. Cost terms may include incentives that reduce the total cost.
- A variety of simulation experiments illustrating the practical utility and adaptability of our framework, enabling new forms of dynamic deception.

Section 2 reviews related work in DPP. In Section 3, we present the single-agent case, providing the foundation for the multi-agent extension discussed in Section 4. Simulation results for various single- and multi-agent scenarios are presented in Section 5 to demonstrate the flexibility and effectiveness of our framework. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Related Work

Deceptive path planning (DPP) has roots in early studies of deceptive motion in human–robot interaction (HRI) [19] and in multi-agent reconnaissance with deception strategies [20]. It was formalized as a path planning problem by Masters and Sardina [21], who introduced the objective of minimizing early goal recognition and defined the notion of a last deceptive point. Building on a long-standing deception taxonomy of simulation (showing the false) and dissimulation (hiding the real), several works have converged on

two practical motion strategies: exaggeration and ambiguity [10,19]. Beyond generating deceptive trajectories, researchers also study (i) observer/goal-recognition models and prediction priors, including max-entropy and resource-allocation variants [8], (ii) algorithms ranging from trajectory optimization to learning-based and RL pipelines [9], and (iii) evaluation protocols and metrics, including human-subject studies [19,22].

Optimization-based DPP methods typically solve from a start position to a final goal and produce a single deceptive path plan, largely in single-agent and static environment settings [8,10–16]. Works that admit environmental changes typically re-solve end-to-end starting from the current state when conditions shift [9,17,18].

The foundational work of Savas et al. [10] developed an optimization framework resulting in a linear program (LP) formulation for balancing an optimal (e.g., shortest) path to the true goal with deceptive deviation with respect to false goals, and incorporating probabilistic constraint satisfaction. Agents' behavior is modeled as a Markov decision process, and entropy is used to model an observer's prediction of the path. Human studies validate the deception. Our work is inspired by the LP planner, providing a practical local planner using similar costs and enabling dynamic adaptation. We experimentally compare with the LP-based planner, showing that it is possible to employ our receding horizon method and achieve a globally optimal path.

Methods for multi-agent DPP are limited. An early heuristic search method uses constraints and lays out waypoints [20,23]. Extensions to [10] include a form of multi-agent deception cast in an MDP framework with the goal of achieving a desired final distribution of agents over goal states [8]. A two-phase solution is developed, switching from a deception phase with a false goal state distribution to a goal-directed phase ending in the true goal distribution. Other extensions to [10] include deceiving a supervisor by minimizing deviation from the expected nominal policy so that at least one agent will achieve a hidden alternate objective [24], and altering the path based on an adversarial intervention model [25].

Learning-based path planning methods face the challenge of generalization over environments, goal and false-goal placement, and forms of deception. In [9], a GNN-based inference engine was developed over a local graph of possible moves from the agent's current location. This used the LP solution as a training signal and was shown to generalize over different graph topologies and deception weights. The local graph can be obtained by spatial sampling around the agent, providing a link to continuous action and state space control.

We emphasize that our framework is not learning-based and does not incorporate training. Note that the Boltzmann distribution has been classically used in general reinforcement learning (RL) and bandit problems to balance exploration and exploitation. However, relative to classical RL, which manages exploration–exploitation via learned value/policy estimates and explicit exploration schemes (e.g., ϵ -greedy, UCB, and Thompson sampling) [26–29], our planner induces stochasticity directly at planning time without any training phase. Thus, while our method is similar to softmax action selection used in RL, we employ tunable deterministic costs that incorporate deception and constraints and map these to a random policy through normalization of the Boltzmann distribution.

Various comparison metrics in the above references are used to quantify the deviation from an expected policy; see also [22]. Our framework follows this arc but emphasizes a simple, on-the-fly stochastic policy, derived from the Boltzmann distribution over short-horizon action-sequence candidates.

3. Single-Agent Method

In this section, we develop a Boltzmann-based formulation for constructing single-agent DPP policies. The Boltzmann distribution originates from statistical mechanics, where the probability of a microscopic state x with energy $E(x)$ is proportional to $\exp\left(-\frac{E(x)}{\theta_{\text{temp}} C_B}\right)$. The temperature parameter θ_{temp} scales the exponent and thereby controls the spread of probability mass, and C_B denotes the Boltzmann constant [30]. By analogy, the same exponential form is used to map scalar costs over candidate action sequences to a probability distribution. Let A_t denote a finite candidate path or action sequence starting at time t , and let $C(A_t) \in \mathbb{R}$ denote its associated deterministic scalar cost. The resulting Boltzmann policy is based on the probability distribution over candidate action sequences given by

$$\Phi_t(A_t) = \frac{e^{-\lambda C(A_t)}}{\sum_{\tilde{A}_t \in \mathcal{A}_t} e^{-\lambda C(\tilde{A}_t)}}, \quad (1)$$

where scalar $\lambda > 0$ is the rationality parameter, playing a role analogous to the inverse of the temperature scaling factor $(\theta_{\text{temp}} C_B)^{-1}$. Note that λ tunes the level of randomness independently of the deterministic cost, and smaller λ yields more randomness. The denominator normalizes the ratio to always yield a valid probability, summing over all possible sequences in set $\mathcal{A}_t$ using dummy variable $\tilde{A}_t$.

We refer to $C(A_t)$ as cost, and under (1), candidate action sequences with smaller costs are assigned higher probabilities. We may also employ incentives that lower the cost, i.e., that move the cost to the left on the real axis, and subsequently yield a higher probability through (1).

This construction yields a stochastic policy that samples action sequences according to their relative costs, rather than deterministically selecting a single optimal action sequence. The exact form of $C(\cdot)$ is user- and scenario-dependent and will typically have two or more terms in a weighted sum to incorporate deception and constraints, as described below. The Boltzmann-based approach naturally extends to multi-agent path planning by including joint cost terms in $C(\cdot)$ and sampling from the resulting joint distribution; we develop this in Section 4.

Notation for the system variables is given at the top of Table 1, and tunable parameters for the single-agent case are listed in Table 2. To facilitate the algorithmic description, we consolidate the system variables into the set Γ_t defined as

$$\Gamma_t = \left\{ s_0, s_t, \hat{s}_{t+K_t}^j, A_t^j, a_t^{jm}, \mathcal{A}_t, A_t^*, a_t^{*m} \right\} \quad (2)$$

and the agent-tunable parameters into the set Θ_t , given by

$$\Theta_t = \left\{ G_t, F_t^1, \dots, F_t^z, \lambda_t, B_t, K_t, \right. \\ \left. M_t, w_t^1, \dots, w_t^4, \delta_t^3, \delta_t^4 \right\}. \quad (3)$$

The agent's initial state is denoted s_0 , its current state at time t by s_t , and its true goal by G_t (or simply G if not time-varying). We let τ denote the planned state trajectory induced by executing an action sequence A_t . The agent starts at s_0 , knows its current state s_t and true goal G , and moves one unit of distance per time step. The agent can adjust its parameters in Θ_t and stops upon reaching its goal state G .

Next, we describe the generation of candidate action sequences. Let $\mathcal{G}_{\mathcal{A}}(\cdot, \cdot)$ denote a generator of the feasible set $\mathcal{A}_t$, starting from state s_t for length K_t , so $\mathcal{A}_t = \mathcal{G}_{\mathcal{A}}(s_t, K_t)$ indicates generation of the entire set. The j -th candidate action sequence is $A_t^j = [a_t^{j1}, \dots, a_t^{jK_t}] \in \mathcal{A}_t$, and we denote its resulting end state as $\hat{s}_{t+K_t}^j$. The end state is obtained via $\hat{s}_{t+K_t}^j = \mathcal{T}_{K_t}(s_t, A_t^j)$, where $\mathcal{T}_{K_t}(\cdot, \cdot)$ generically denotes the K_t -step transition to the end state for the particular candidate action sequence A_t^j starting from s_t .

Table 1. System variables. Superscript i denotes the i -th agent. For the single-agent case, the superscript is omitted, e.g., s_t corresponds to $s^i(t)$. The index $m \in \{1, \dots, K_t\}$ denotes the step within a K_t -step sequence.

Symbol	Description
t	Current time step
s_0	Initial state (start location)
s_t	State at time t
a_t^{*m}	m -th action in the selected plan A_t^*
$\mathcal{A}_t$	Feasible set of K_t -step sequences at time t
A_t^*	Selected action sequence at time t
A_t^j	j^{th} candidate action sequence at time t of length K_t
a_t^{jm}	Action at step m within sequence A_t^j
$\hat{s}_{t+K_t}^j$	End state for sequence A_t^j
τ	Planned trajectory
Multi-Agent Case	
s_0^i	Initial state (start location) of agent i
N	Total number of agents
$s^i(t)$	State of agent i at time t
$S(t)$	Joint state of all agents at time t
$\hat{s}^{i,j}(t + K^i(t))$	End state of agent i under $A^{i,j}(t)$
$A^{i,j}(t)$	j^{th} candidate action sequence for agent i at time t
$\mathcal{A}^i(t)$	Set of feasible candidate action sequences for agent i
$\mathcal{A}(t)$	Joint set of feasible candidate sequences for all agents
$\mathbf{A}(t)$	Joint candidate action sequence at time t (one sequence per agent)
$\hat{S}(t)$	Collection of all end states under $\mathbf{A}(t)$
$\mathbf{C}_{\mathcal{A}}(t)$	Cost matrix
$\mathbf{A}^*(t)$	Selected multi-agent action plan

Table 2. Tunable single-agent parameters.

Parameter	Description
G_t	True goal at time t
F_t^z	False goals at time t , $z \in \{1, 2, \dots\}$
λ_t	Rationality parameter
K_t	Planning trajectory length
M_t	Number of steps before replanning, $M_t \leq K_t$
$w_t^1, \dots, w_t^4$	Cost term coefficients
B_t	Optional time budget constraint
δ_t^3, δ_t^4	Binary flags for optional cost terms

We define the overall cost function as a weighted sum of scalar terms, computed at time t based on the current state s_t and the trajectory with end state $\hat{s}_{t+K_t}^j$, given by

$$\begin{aligned}
 \mathcal{C}(\Theta_t, \Gamma_t) = & w_t^1 C_{\text{goal}}(G_t, s_t, \hat{s}_{t+K_t}^j) \\
 & + w_t^2 C_{\text{decep}}(G_t, \{F_t^z\}z, s_t, \hat{s}_{t+K_t}^j) \\
 & + \delta_t^3 w_t^3 C_{\text{time}}(t, B_t, s_t, \hat{s}_{t+K_t}^j) \\
 & + \delta_t^4 w_t^4 C_{\text{smooth}}(s_t, \hat{s}_{t+K_t}^j).
 \end{aligned} \tag{4}$$

Each cost term in (4) returns a scalar in $\mathbb{R}$. C_{goal} encourages reaching G_t , and C_{decep} encodes the deception objective relative to false goals $\{F_t^z\}z$. Optional costs include C_{time} , which enforces a time-to-goal constraint specified by B_t , and C_{smooth} , which penalizes

non-smooth motion. These will be described further in the examples presented in Section 5. This cost structure is flexible and can be varied based on user-defined objectives, incentives, and scenario-specific requirements.

By substituting $\mathcal{C}(\Theta_t, \Gamma_t)$ from (4) into (1), the probability of action sequence A_t is given by

$$\Phi_t(A_t | \Theta_t) = \frac{e^{-\lambda_t \mathcal{C}(\Theta_t, \Gamma_t)}}{\sum_{\tilde{A} \in \mathcal{A}_t} e^{-\lambda_t \mathcal{C}(\Theta_t, \tilde{\Gamma}_t)}}. \quad (5)$$

Here, the denominator sum is over all possible action sequences $\tilde{A} \in \mathcal{A}_t$, normalizing the ratio to provide a legitimate probability. The agent samples this distribution, represented as

$$A_t^* \sim \Phi_t(A_t | \Theta_t), \quad (6)$$

to obtain a K_t -step action sequence $A_t^* = (a_t^{*1}, \dots, a_t^{*K_t})$. The agent can then follow the selected short-distance trajectory τ that ends in state s_{t+1} . The agent's parameters and cost function are then updated if desired, and the planning process is repeated.

The single-agent deceptive path planner is summarized in Algorithm 1, with the inputs being parameter set Θ_t and starting state s_t , and the output being the planned trajectory τ . The algorithm statement loops over successive short-distance plans until the goal G_t is reached (the while-loop at line 2) and outputs a longer concatenated trajectory. The algorithm is easily modified to provide only one, or more, successive plans as desired. Line 3 generates the feasible candidate set of action sequences. Lines 4–7 compute the end state and cost for each candidate action sequence $A_t^j \in \mathcal{A}_t$. Line 8 generates the distribution of candidate sequences, which is sampled in line 9 to yield the selected sequence A_t^* . The loop in lines 10–17 allows for executing only the first $M_t \leq K_t$ steps if desired. The end state is recorded in line 11, and the trajectory is appended to a longer stored trajectory in line 12. The loop at line 13 terminates the algorithm if the goal G_t is reached. Finally, the trajectory τ is returned.

Algorithm 1 Single-Agent Deceptive Path Planner

Require: Θ_t, s_t

```

1:  $\tau \leftarrow [s_t], t \leftarrow 0$                                 ▷ Initialize trajectory
2: while  $s_t \neq G_t$  do                                    ▷ Loop until goal is reached
3:    $\mathcal{A}_t \leftarrow \mathcal{G}_{\mathcal{A}}(s_t, K_t)$                         ▷ Generate feasible set
4:   for each  $A_t^j \in \mathcal{A}_t$  do
5:      $\hat{s}_{t+K_t}^j \leftarrow \mathcal{T}_{K_t}(s_t, A_t^j)$                 ▷ Compute  $K_t$  step final state
6:      $c_j \leftarrow \mathcal{C}(\Theta_t, \Gamma_t^j)$                         ▷ Compute cost
7:   end for
8:    $\Phi_t \leftarrow \Phi_t(A_t | \Theta_t)$                         ▷ Construct policy per Equation (5)
9:    $A_t^* \sim \Phi_t$                                           ▷ Sample the policy
10:  for  $m \in \{1, \dots, M_t\}$  do                             ▷ Execute  $M_t \leq K_t$  steps
11:     $s_{t+1} \leftarrow T(s_t, a_t^{*m})$                        ▷ Advance to end state
12:     $\tau \leftarrow \tau \cup \{s_{t+1}\}$                          ▷ Append to full trajectory
13:    if  $s_{t+1} = G_t$  then                                   ▷ Check arrived at true goal
14:      break
15:    end if
16:     $t \leftarrow t + 1$                                        ▷ Increment global clock
17:  end for
18: end while
19: return  $\tau$                                               ▷ Return end-to-end planned trajectory
```

4. Multi-Agent Method

Consider N agents whose joint state at time t is $S(t) = (s^1(t), \dots, s^N(t))$, and let the collection of agent scalar rationality parameters be denoted by $\{\lambda^i(t)\}_{i=1}^N$. The true goal

of agent i at time t is denoted $G^i(t)$, or G^i when the goal is time-invariant. Similar to the single-agent case, the system variables for the multi-agent algorithm are listed in Table 1, and the agents' tunable parameters are listed in Table 3 (generalizing Table 2).

Table 3. Tunable i -th agent parameters (multi-agent case).

Symbol	Description
$G^i(t)$	True goal of agent i at time t
$\{F_z^i(t)\}_{z=1}^{n(i)}$	Set of $n(i)$ false goals for agent i at time t
$K^i(t)$	Number of planning-horizon steps for agent i
$M^i(t)$	Number of steps executed before replanning
$B^i(t)$	Max total allowed (time) steps for agent i
$\{w_\ell^i(t)\}_{\ell=1}^4$	Weights for i -th agent cost terms
$\{\lambda^i(t)\}_{i=1}^N$	Rationality parameters for each agent
$\lambda(t)$	Rationality vector

As we did in the single-agent case, for ease of notation we define sets of parameters as follows. The set of tunable parameters for all N agents at time t is given by

$$\Omega(t) = \left\{ \Omega^i(t) \right\}_{i=1}^N \quad (7)$$

where the subset of parameters for agent i is

$$\Omega^i(t) = \left\{ G^i(t), \{F_z^i(t)\}_{z=1}^{n(i)}, K^i(t), M^i(t), B^i(t), \{w_\ell^i(t)\}_{\ell=1}^4, \lambda^i(t) \right\}. \quad (8)$$

Similarly, the set of all system variables is

$$\mathbf{Y}(t) = \left\{ \left(s_0^i, s^i(t), A^{ij}(t), \hat{s}^{ij}(t + K^i(t)) \right)_{i=1}^N, \left(\mathcal{A}^i(t) \right)_{i=1}^N, N, S(t), \mathcal{A}(t), \hat{S}(t), \mathbf{A}(t), \mathbf{C}_A(t), \mathbf{A}^*(t) \right\}, \quad (9)$$

and we denote $\mathbf{Y}^i(t)$ as the set of system parameters for the i -th agent, drawn from $\mathbf{Y}(t)$.

We assume each agent knows the joint state $S(t)$, the goals and false goals for all agents, and shares collective knowledge of the variables and parameters $\Omega(t)$ and $\mathbf{Y}(t)$. These assumptions can be relaxed, for example, to form and disband subsets of coupled agents. To illustrate the planning in our grid-world simulations, each agent moves one unit distance per time step, and agents stop when their goal state is achieved.

Next, we describe the generation and cost assignment for candidate joint trajectory sequences, leading to the Boltzmann-based policy form in the multi-agent case. Let $A^{ij}(t) = [a_1^{ij}(t), \dots, a_{K_i(t)}^{ij}(t)]$ denote a feasible action sequence for the i -th agent over $K^i(t)$ steps. Beginning in the current state $s^i(t)$ and executing action sequence $A^{ij}(t)$, this j -th trajectory ends in state $\hat{s}^{ij}(t + K^i(t))$. Now, let $\mathcal{A}^i(t)$ be the collection of all feasible action sequences for the i -th agent beginning at state $s_i(t)$ at time t . Thus, $\mathcal{A}^i(t)$ is a set, and $A^{ij}(t) \in \mathcal{A}^i(t)$.

A joint candidate action is denoted

$$\mathbf{A}(t) = (A^{1j_1}(t), \dots, A^{Nj_N}(t)), \quad (10)$$

and contains one candidate action sequence for each of the N agents; j_i is the index of the candidate action sequence selected for agent i . Each $A^{ij_i}(t)$ has dimensions $1 \times K^i(t)$, with zero padding if some agent(s) have a shorter planning horizon than others. Therefore, $\mathbf{A}(t)$ consists of N rows, where the i -th row has dimension $1 \times \max_i K^i(t)$.

For a given joint action candidate $\mathbf{A}(t)$, we collect the ending state for each agent. The i -th agent begins in state $s_i(t)$ and follows $A^{ij_i}(t)$, arriving at end state $\hat{s}^{ij_i}(t + K^i(t))$. The collection of end states for the candidate joint action is denoted

$$\hat{S}(t) = \left\{ \hat{s}^{ij_i}(t + K^i(t)) \right\}_{i=1}^N. \quad (11)$$

Next, we evaluate the cost for each joint action candidate $\mathbf{A}(t)$. The total cost consists of two parts: (i) costs associated with the individual i -th agent and its candidate trajectory, and (ii) joint interaction costs that couple agents. Depending on the specific cost definitions, these may be evaluated step-by-step along a trajectory, or based only on the end states $\hat{\mathbf{S}}(t)$.

We specify the costs as follows. First, similar to the single-agent case, we evaluate the cost of trajectory sequence $A^{ij}(t)$ for the i -th agent. This depends on the i -th agent's parameters $\Omega^i(t)$ and $\mathbf{Y}^i(t)$, the state transitions and final state for a given action sequence $A^{ij}(t)$, and the specific form of the individual cost terms that are summed. These per-agent scalar cost components are given by

$$\mathbf{c}^{ij}(t) = \begin{bmatrix} C_{\text{goal}}^i(t) \\ C_{\text{decep}}^i(t) \\ C_{\text{time}}^i(t) \\ C_{\text{smooth}}^i(t) \end{bmatrix}. \quad (12)$$

In the deception context of this paper, we always include $C_{\text{goal}}^i(t)$ and $C_{\text{decep}}^i(t)$. The costs in (12) reflect those used in the single-agent case, as in Equation (4), including the optional costs $C_{\text{time}}^i(t)$ and $C_{\text{smooth}}^i(t)$, which we use in some of our examples in Section 5.

Second, for the multi-agent case, we introduce L scalar interaction costs, denoted as

$$\boldsymbol{\phi}(t) = \begin{bmatrix} C_{(1)}(t) \\ \vdots \\ C_{(L)}(t) \end{bmatrix}. \quad (13)$$

These costs reflect coupling among agents; see the examples in Section 5.

Now, we combine the above costs in preparation for the joint Boltzmann formulation. For any joint candidate action $\mathbf{A}(t)$, we define the full cost vector as

$$\mathbf{C}_{\mathbf{A}}(t) = [C^{1,j_1}(t), \dots, C^{N,j_N}(t), C^{\text{int}}(t)]^{\top}. \quad (14)$$

The N leading terms in $\mathbf{C}_{\mathbf{A}}(t)$ are the costs for each individual agent, obtained from the weighted sum of costs in (12), similar to Equation (4). Recall that in (4), the combination weights are denoted ω_t , and the δ_t are binary in $\{0, 1\}$ to optionally include cost terms. The last term in (14) is the sum of interaction costs given by

$$C^{\text{int}}(t) = \sum_{l=1}^L C_{(l)}(t). \quad (15)$$

Generalizing the single-agent case, we apply the Boltzmann approach for N agents. The probability of selecting a specific joint action candidate $\mathbf{A}(t)$ is

$$\eta_{\mathbf{A}}(t) = \frac{e^{-\lambda(t)^{\top} \mathbf{C}_{\mathbf{A}}(t)}}{\sum_{\tilde{\mathbf{A}}(t) \in \mathcal{A}(t)} e^{-\lambda(t)^{\top} \mathbf{C}_{\tilde{\mathbf{A}}}(t)}}. \quad (16)$$

We introduced the rationality vector defined by

$$\lambda(t) = [\lambda^1(t), \dots, \lambda^N(t), \lambda^{\text{int}}]^{\top}, \quad (17)$$

which collects the rationality parameters of each agent and an additional parameter λ^{int} that weights $C^{\text{int}}(t)$, so that $\lambda(t) \in \mathbb{R}^{N+1}$ matches the dimension of $\mathbf{C}_{\mathbf{A}}(t)$. Here, $\tilde{\mathbf{A}}(t)$ denotes a joint action candidate from set $\mathcal{A}(t)$, and the normalizing sum in the denominator is over all joint action candidates to ensure $\eta_{\mathbf{A}}(t)$ is a valid probability.

Finally, the joint action $\mathbf{A}^*(t)$ is selected by sampling from $\mathcal{A}(t)$ according to the distribution $\eta_{\mathcal{A}}(t)$, represented by

$$\mathbf{A}^*(t) \sim \eta_{\mathcal{A}}(t), \quad (18)$$

so that $\mathbf{A}^*(t)$ contains the planned trajectories for each agent.

The multi-agent DPP method is summarized in Algorithm 2. The inputs are $\Omega(t)$ (multi-agent planning parameters) and the joint state $S(t)$; the output is the joint action sequence

$\mathbf{A}^*(t)$. Lines 2–4 generate feasible candidate action sequences for each agent. Lines 5–8 find the end state for each feasible agent sequence and compute the corresponding per-agent costs. The loop in lines 9–15 computes the probability for each feasible joint action sequence. Line 10 collects the end states. Lines 11–12 compute and aggregate the interaction costs. Line 13 constructs the cost vector, and line 14 computes the probability of the joint action sequence. Finally, line 16 samples the joint planning policy, and the joint plan is returned in line 17.

Algorithm 2 Multi-Agent Deceptive Path Planner

```

1: Input:  $\Omega(t), S(t) = (s^1(t), \dots, s^N(t))$ 
2: for all agents  $i$  do
3:    $\mathcal{A}^i(t) \leftarrow [s^i(t), K^i(t)]$  ▷ Generate feasible action sequences
4: end for
5: for all  $i$  and  $A^{i,j}(t) \in \mathcal{A}^i(t)$  do
6:    $\hat{s}^{i,j}(t + K^i(t)) \leftarrow [s^i(t), A^{i,j}(t)]$  ▷ Compute end state
7:    $\mathbf{c}^{i,j}(t) \leftarrow \begin{bmatrix} C_{\text{goal}}^i(t) \\ C_{\text{decep}}^i(t) \\ C_{\text{time}}^i(t) \\ C_{\text{smooth}}^i(t) \end{bmatrix}$  ▷ Compute per-agent costs
8: end for
9: for all  $\mathbf{A}(t)$  do
10:   $\hat{S}(t) \leftarrow \{\hat{s}^{i,j_i}(t + K^i(t))\}_{i=1}^N$  ▷ Collect end states
11:   $\phi(t) \leftarrow \begin{bmatrix} C^{(1)}(t) \\ \vdots \\ C^{(L)}(t) \end{bmatrix}$  ▷ Compute interaction costs
12:   $C_{\text{int}}(t) \leftarrow \sum_{\ell=1}^L C^{(\ell)}(t)$  ▷ Aggregate interaction costs
13:   $\mathbf{C}_{\mathbf{A}}(t) \leftarrow [C^{1,j_1}(t), \dots, C^{N,j_N}(t), C_{\text{int}}(t)]^\top$  ▷ Construct cost vector
14:   $\eta_{\mathbf{A}(t)} \leftarrow \frac{e^{-\lambda(t)^\top \mathbf{C}_{\mathbf{A}}(t)}}{\sum_{\tilde{\mathbf{A}}(t) \in \mathcal{A}(t)} e^{-\lambda(t)^\top \mathbf{C}_{\tilde{\mathbf{A}}}(t)}}$  ▷ Compute Boltzmann probability
15: end for
16:  $\mathbf{A}^*(t) \sim \eta_{\mathbf{A}(t)}$  ▷ Sample the joint policy
17: return  $\mathbf{A}^*(t)$  ▷ Return joint plan

```

Algorithm 2 may be executed repeatedly, beginning with the current joint state and any changes to the parameters in $\Omega(t)$. The planner ends when agents reach their goal states.

4.1. Information Sharing Among Agents

Multi-agent path planning is a classic problem with combinatorial complexity, e.g., see [31, Sect. 7.2]. In our multi-agent setting, only a small amount of information is assumed available for coordinated planning when repeatedly applying Algorithm 2. Once initialized, only changes to the tunable parameters in $\Omega(t)$ need to be shared. This includes parameters affecting individual agent costs, interaction costs, and constraints. As we show in the examples, a time schedule may be set to change some agents' parameters, where agents otherwise proceed independently or with dependent sub-groups. Algorithm 2 also requires the initial joint state, i.e., the end state for each agent after executing the most recent plan.

Our scheme yields joint rollouts from random sampling, as in Equation (18) and line 16 in Algorithm 2. If it is assumed the agents will reach their end states with high probability, then even this need not be communicated for every planning round. A further simplifi-

cation is to employ independent rollouts, where each agent evaluates its own candidate sequences $A^{i,j}(t)$ and shares only its final state $\hat{s}^{i,j}(t + K^i(t))$. Agents can also collaborate without evaluating all joint profiles by intermittently sharing summaries of their local policies, exchanging only the parameters that influence coupled costs, or interleaving local Boltzmann updates with occasional global synchronization. These quasi-independent strategies aim to preserve coordination while avoiding the communications and computational burden of full joint updates. This provides design flexibility for collaboration and rollouts, and the potential for decentralization.

The coupled planning problem is often cast in robotics, where moving obstacles and robot collisions must be avoided. Many heuristics have been studied with sophisticated implementations in complex environments, e.g., see [32]. We believe that our framework has sufficient generality to enable incorporation into such existing planning pipelines.

4.2. Complexity and Scaling

To study complexity and scaling, we consider the worst case where all feasible action sequences are enumerated. Let B denote the branching factor, i.e., the number of feasible actions from a state. For a single agent at each receding-horizon update, the planner enumerates all feasible action sequences of length K . Each planning step has a branching factor B , so the worst-case candidate set contains $\mathcal{O}(B^K)$ sequences. Each candidate requires a K -step rollout and constant-time cost evaluation, giving overall complexity $\mathcal{O}(B^K K)$.

Next, we consider the multi-agent case. As above, let each agent have a branching factor B and a K -step lookahead, producing a single-agent local candidate set of size $\mathcal{O}(B^K)$. If the team cost function decomposes (separable cost case), then each agent performs an independent update with cost $\mathcal{O}(B^K K)$, yielding a total complexity of $\mathcal{O}(NB^K K)$, i.e., the overall cost scales linearly with the number of agents N .

When the team cost is not separable, then evaluating a Boltzmann policy requires considering all feasible joint candidate action sequences whose set size is

$$|\mathcal{A}(t)| = \prod_{i=1}^N |\mathcal{A}^i(t)| \approx \mathcal{O}(B^{KN}), \quad (19)$$

where $|\cdot|$ denotes set cardinality. Each candidate joint action requires a K -step rollout and cost evaluation, giving an overall complexity of $\mathcal{O}(B^{KN} K)$, which grows exponentially with N under full coupling.

When K is relatively small, the update costs remain relatively lightweight. In practice, the complexity may be much smaller; the planner does not need to enumerate all B^K branches in the single-agent case, or explore the full B^{KN} sized space for the multi-agent case. Standard search techniques (e.g., pruning, heuristic rollouts, or beam-search-style truncation) can limit the number of explored sequences, ensuring that only a small subset of high-value branches is evaluated at each step. Thus, the empirical complexity per update can be substantially below the worst-case bound for receding horizon planning.

5. Experimental Results

In this section, we present a variety of experiments, showing how single and multiple agents can maintain deception while incorporating local and resource constraints. We demonstrate dynamic deception induced through time-varying parameters, and we also show that the algorithm can reproduce optimal end-to-end paths using updated local planning by using the LP method from [10] as a baseline. The experiments illustrate the flexibility and dynamic adaptation of the framework, revealing the potential for use in many different settings.

5.1. Single-Agent DPP: Exaggeration and Ambiguity

5.1.1. Exaggeration

An exaggerated path seemingly targets a false goal while eventually arriving at the true goal. This represents a tradeoff between an efficient path to the true goal and deviation towards a false goal, e.g., see the discussion in [10]. We instantiate the planner in Algorithm 1 on a 375×375 grid (four-neighbor moves), with a small margin from the boundary. We perform 500 Monte Carlo trials and accumulate visit-frequency heatmaps.

We use Euclidean distances to true and false goals given by

$$\Delta_{\text{true}}(s_t, s_{t+1}) = [d(s_t, G) - d(s_{t+1}, G)]_+ \quad (20)$$

$$\Delta_{\text{false}}(s_t, s_{t+1}) = [d(s_t, F) - d(s_{t+1}, F)]_+ \quad (21)$$

where $[x]_+ = \max\{x, 0\}$, G is the true goal, and F is the false goal. We adopt the cost in (4) with $K = 1$ and deactivate time and smoothness terms ($\delta_3 = \delta_4 = 0$). The per-step cost components are now

$$C_{\text{goal}}(G, s_t, s_{t+1}) \triangleq -\Delta_{\text{true}}(s_t, s_{t+1}) \quad (22)$$

$$C_{\text{decep}}(G, F, s_t, s_{t+1}) \triangleq -\Delta_{\text{false}}(s_t, s_{t+1}) \quad (23)$$

so moving closer to a target reduces cost. Deception dynamics are introduced by modulating exaggeration using a scalar time schedule $u(t) \in [0, 1]$ through the weights

$$w_1(t) = \kappa_0 (1 - u(t)), \quad w_2(t) = \alpha_0 u(t) \quad (24)$$

with constants $\alpha_0 > 0$ (deception gain) and $\kappa_0 > 0$ (true-goal gain). The single-step instantiation of (4) is now

$$\mathcal{C}_t \equiv \mathcal{C}(\Theta_t, \Gamma_t) \quad (25)$$

where

$$\begin{aligned} \mathcal{C}_t = & w_1(t) C_{\text{goal}}(G, s_t, s_{t+1}) \\ & + w_2(t) C_{\text{decep}}(G, F, s_t, s_{t+1}). \end{aligned} \quad (26)$$

We adopt a piecewise-linear scheduling profile for the deception weight. Let $S \geq 0$ be the switch-on time, $L > 0$ be the total scheduling window length, and let $r_{\text{up}} \geq 0$ and $r_{\text{down}} \geq 0$ denote, respectively, the ramp-up and ramp-down durations. The schedule is

$$u(t) = \begin{cases} 0, & t < S, \\ \frac{t - S}{r_{\text{up}}}, & S \leq t < S + r_{\text{up}}, \\ 1, & S + r_{\text{up}} \leq t < S + L - r_{\text{down}}, \\ \frac{S + L - t}{r_{\text{down}}}, & S + L - r_{\text{down}} \leq t < S + L, \\ 0, & t \geq S + L. \end{cases} \quad (27)$$

Additional parameters were set as follows; $\alpha_0 = 4$, $\kappa_0 = 10$, $\lambda = 0.8$, and lookahead depth $K_t = 3$. For the single-agent exaggeration experiment, we use a one-step cost formulation combined with a multi-step planning horizon. Along each trajectory, the cost is updated at each step based on the state, and it is accumulated over the trajectory. The temporal constants are $S = 40$ and $L = 300$. We consider three cases in (27) governed by the ramp parameters $(r_{\text{up}}, r_{\text{down}})$, set to $(0, 0)$ for Hard Switch, $(100, 0)$ for Ramp Up, and $(0, 100)$ for Ramp Down. The three cases are illustrated in Figure 1, plotting trajectory heat maps over 500 trials.

As illustrated in Figure 1, Algorithm 1 effectively generates exaggerated trajectories towards the false goal in the upper left. The differences between the three cases arise directly from the temporal weighting of deception defined in (24). The Ramp Up case eventually commits to a more exaggerated path, whereas Ramp Down has a strong early

bias to exaggeration. In comparison, the Hard Switch trajectories are focused on the false goal and then abruptly switch to the true goal.

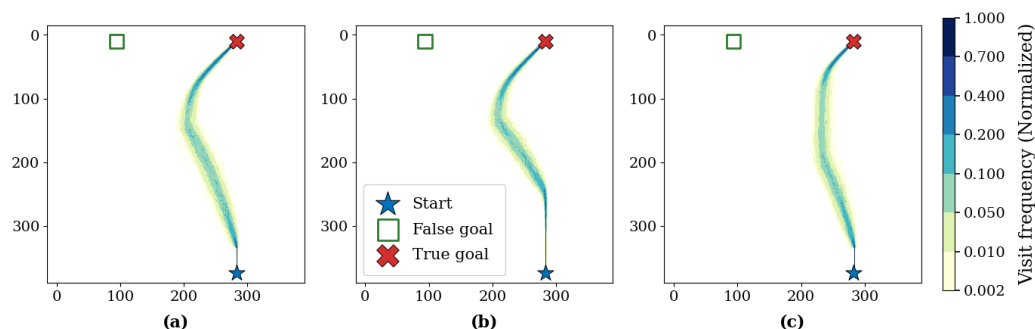


Figure 1. Exaggeration: Single-agent trajectory heatmaps over 500 trials. Three exaggeration deception scheduling cases are shown; (a) Hard Switch, (b) Ramp Up, and (c) Ramp Down. These illustrate dynamically tuning the deceptive trajectories from the start (star) towards the false goal F (square) before transitioning to the true goal G (red cross).

5.1.2. Ambiguity

Next, we consider deception through ambiguity. The agent deviates from the most efficient path to obscure the true goal by using a trajectory that is balanced between the true and false goal(s), e.g., as defined in [10]. The agent progresses more or less equally towards both possible goals, so that the observer cannot readily predict which is the true goal until the agent eventually closes on the true goal.

All settings (grid, action set, receding-horizon loop, and Boltzmann sampling) are identical to the exaggeration case above; only the cost differs. To maintain ambiguity, we penalize the absolute distance difference between the false and true goals so that the agent tends to approach and stay near a line of bisection. Along the ideal bisector line, the Euclidean distance to either goal remains the same. With Euclidean distance $d(\cdot, \cdot)$ and a one-step candidate s_{t+1} , we obtain

$$C_{\text{amb}}(G, F, s_{t+1}) \triangleq |d(s_{t+1}, F) - d(s_{t+1}, G)|. \quad (28)$$

Balancing ambiguity with progress toward the true goal (using the same goal-progress term $C_{\text{goal}}(G, s_t, s_{t+1})$ as before), the per-step total cost is now

$$C_t = \tilde{w}_1(t) C_{\text{amb}}(G, F, s_{t+1}) + \tilde{w}_2(t) C_{\text{goal}}(G, s_t, s_{t+1}). \quad (29)$$

We adopt the time scheduling approach as before, where now the gains for the ambiguity objective are set to $\alpha = 4$ and $\kappa = 4$, and the temporal constants are adjusted to $S = 40$ and $L = 250$ to accommodate the specific dynamics of intent concealment. The ramp parameters r_{up} and r_{down} are the same as those used above.

Ambiguity trajectory heatmaps over 500 trials are shown in Figure 2. Similar to Figure 1, we let $\tilde{w}_1(t)$ and $\tilde{w}_2(t)$ follow the same three-piecewise schedulers used previously. Now, the temporal schedule balances the ambiguity cost C_{amb} against direct progress to the true goal. Depending on the dynamic weighting schedule, the trajectories progress towards the line of bisection and maintain ambiguity, then transition towards the true goal. All three cases show the agent progressing to the line of bisection as the agent moves from bottom to top. Along this line, the agent's distance to both the true and false goals remains nearly constant. Depending on the schedule, the agent eventually turns toward the true goal.

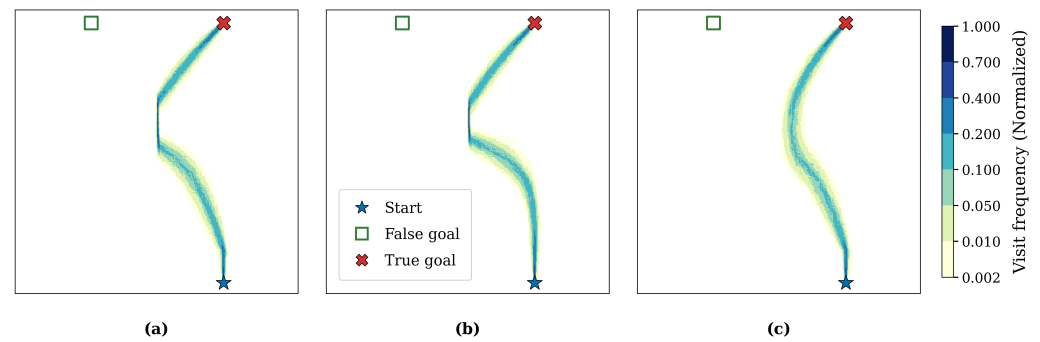


Figure 2. Ambiguity: Single-agent trajectory heatmaps over 500 trials. Three ambiguity deception scheduling cases are shown; (a) Hard Switch, (b) Ramp Up, and (c) Ramp Down. These illustrate dynamically tuning the ambiguity deception from the start (star) by approaching the bisection line where the true and false goals are equi-distant, before transitioning to the true goal G (red cross).

5.1.3. Obstacles and Trajectory Alignment

Our proposed Boltzmann-based Algorithm 1 is designed to maintain a consistent global intent while providing the necessary flexibility for local adaptation during execution. This allows the agent to, for example, navigate around unexpected obstacles without compromising its deceptive objective. The proposed approach can be integrated into standard navigation pipelines, heatmap-guided exploration, or safety-zone-constrained mission planning. Local environmental updates (e.g., temporary obstacles, risk zones, or dynamic no-go regions) can be incorporated without resorting to a new global planning process.

To implement the local constraints while retaining the full state space, we impose an infinite cost for transitions into obstacle locations, yielding

$$\begin{aligned} C_t = & w_1(t)C_{\text{goal}}(G, s_t, s_{t+1}) \\ & + w_2(t)C_{\text{decep}}(G, F, s_t, s_{t+1}) \\ & + \infty \cdot \mathbb{I}(s_{t+1}), \end{aligned} \quad (30)$$

where $\mathbb{I}(\cdot)$ is the indicator function given by

$$\mathbb{I}(s_{t+1}) = \begin{cases} 1 & \text{if transition into } s_{t+1} \text{ is infeasible} \\ 0 & \text{otherwise.} \end{cases} \quad (31)$$

Sample trajectories in Figure 3 demonstrate exaggeration while bypassing an obstacle. The same stochastic policy generates all paths shown.

To further characterize deception along a trajectory, such as those in Figure 3, we quantify deceptive steering using a continuous angular metric inspired by the trajectory-alignment perspective introduced in [22]. Let $\theta_v^{(a)}$ and $\theta_v^{(d)}$ denote the angular deviations between the agent's instantaneous motion direction and the directions to the true and decoy goals at time step v . We define the continuous alignment score (CAL) as

$$\text{CAL} = \frac{1}{N_{\text{CAL}}} \sum_{v=1}^{N_{\text{CAL}}} (\theta_v^{(a)} - \theta_v^{(d)}). \quad (32)$$

The angular deviations $\theta_v^{(a)}$ and $\theta_v^{(d)}$ are measured in radians. Under the sign convention in (32), negative CAL values indicate that the agent is more closely aligned with the decoy direction, whereas positive values indicate closer alignment with the true-goal direction. The sign and magnitude of CAL provide a directed measure of deceptive intent. Negative values indicate orientation toward the decoy, and the absolute value reflects the strength of this angular bias. Similarly, positive values indicate more emphasis towards the true goal direction. A persistently negative CAL suggests a deliberate and consistent behavioral bias that prioritizes misleading the observer over path efficiency. This metric is relatively insensitive to small random zig-zag angular variations over the trajectory, as they tend to average out.

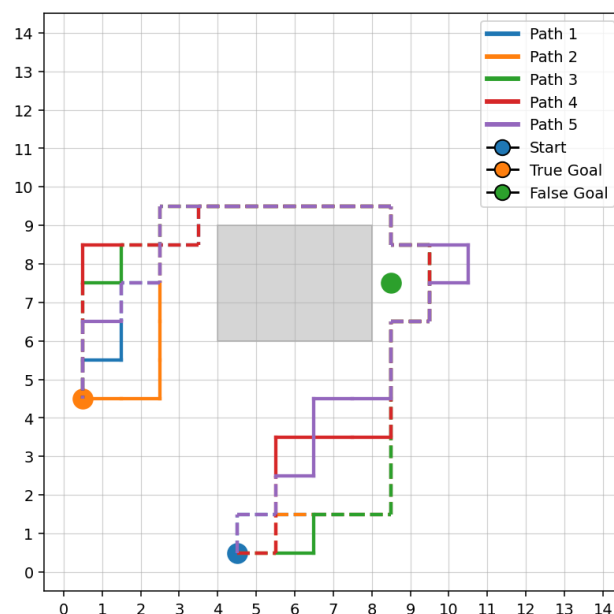


Figure 3. Examples of exaggeration paths. The stochastic policy maintains deception while avoiding the local obstacle region (in gray).

The use of CAL is illustrated in Figures 4 and 5. An example empirical distribution of CAL is shown in Figure 5, obtained from 500 trials of exaggeration deception; note the two clusters (modes). All trajectories yield negative CAL, as expected with exaggeration. These trajectories are drawn from the scenario shown in Figure 4. In this figure, 40 exaggeration trajectories are plotted, with blue and red colors each corresponding to 20 randomly chosen trajectories from within each CAL mode. Larger negative CAL values indicate a trajectory that more aggressively exaggerates towards the false goal for a longer duration while also avoiding the gray obstacle area and maintaining the deceptive trajectory.

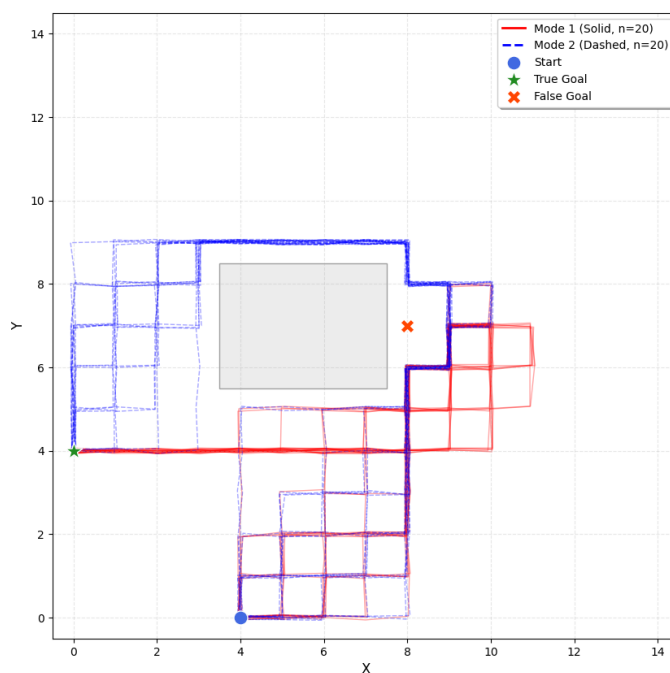


Figure 4. Forty exaggeration sample trajectories, with blue and red corresponding to 20 randomly selected within each CAL mode shown in Figure 5. The same random policy generates trajectories that are persistently deceptive while avoiding the obstacle.

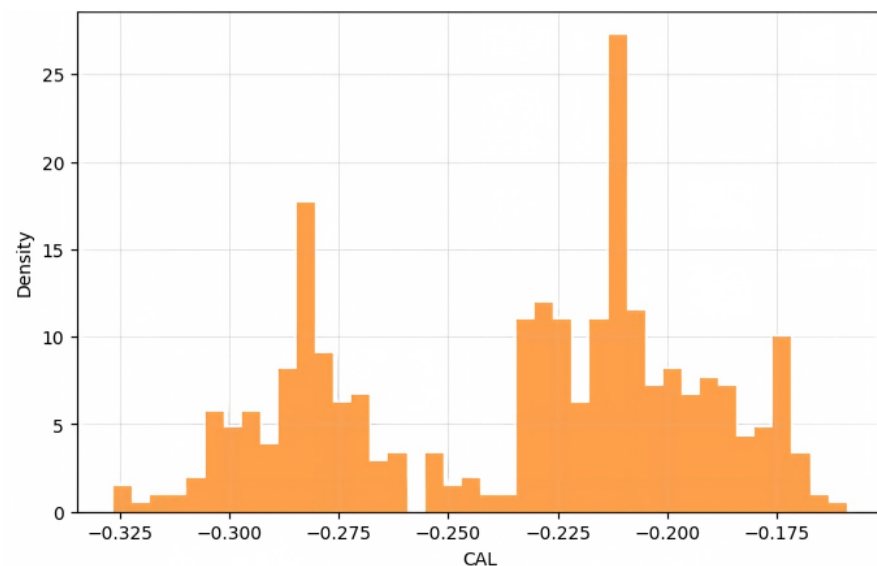


Figure 5. Example empirical distribution of the Continuous Alignment score (CAL) obtained from 500 exaggerated deception trajectory trials. The two CAL modes are illustrated in Figure 4.

5.1.4. Comparison with End-to-End DPP

Next, we compare our planner with an end-to-end reference trajectory generated by the LP-based method from [10] using exaggeration deception. Results are shown in Figure 6, where the grid setup corresponds to the example shown in Figure 2 of [10], with the true goal marked G1 and the false goal marked G2. We tune our method using parameters $\alpha_0 = 2$, $\kappa_0 = 8$, $K_t = 3$, $S = 0$, and $L = 4$. We show two cases with different choices for the parameter λ that induces randomness in our Boltzmann-based policy, compared to the same reference trajectory. With low randomness, shown in Figure 6a, our method tracks the end-to-end solution with little variation among realizations. With higher randomness, shown in Figure 6b, our method exhibits more variation with respect to the reference trajectory.

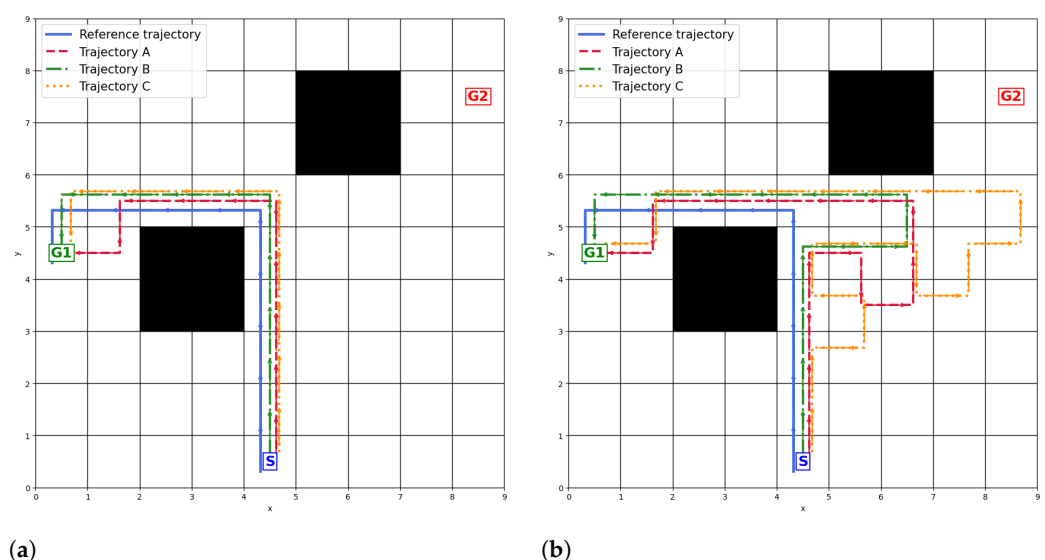


Figure 6. Exaggeration deception example, comparing an end-to-end plan using the LP optimization in [10] with random trajectories generated by the proposed approach. The setup corresponds to Figure 2 in [10], with true goal G1 and false goal G2. The LP method yields a single reference trajectory, shown in both panels. (a) $\lambda = 12$. With low randomness, realizations of our method track the LP-based plan. (b) $\lambda = 4$. Increased randomness yields trajectories that have more deviation from the LP-based reference trajectory.

Although our method does not explicitly optimize the LP objective in an end-to-end fashion, in this simple example, our method is shown to replicate the LP-based trajectory. While all trajectories exhibit the desired exaggeration behavior, initially exaggerating toward the false goal before converging to the true goal, tuning λ enables a narrower or wider spread of realizations.

5.2. Dynamic Deception: Moving False Goal

Our method enables new forms of dynamic deception by time-varying the agents' parameters. As an illustrative example, we let the true goal be static and move the false goal along a linear trajectory, while executing a deceptive trajectory.

Consider the scenario shown in Figure 7, where the agent starts at the bottom right with the true goal in the upper right, and the false goal moves linearly from upper left to bottom left. Two moving false goal cases are shown: slow (0.4) and fast (2.0) grid units per time step. Both the agent and the false goal begin motion simultaneously. We apply ambiguous deception in Figure 7a,b, and exaggeration deception in Figure 7c,d, plotting a heat map of 500 trials for each case. For ambiguity, we use the same agent parameters, including $u(t)$, as in Figure 2, and for exaggeration, we use the same parameters as in Figure 2.

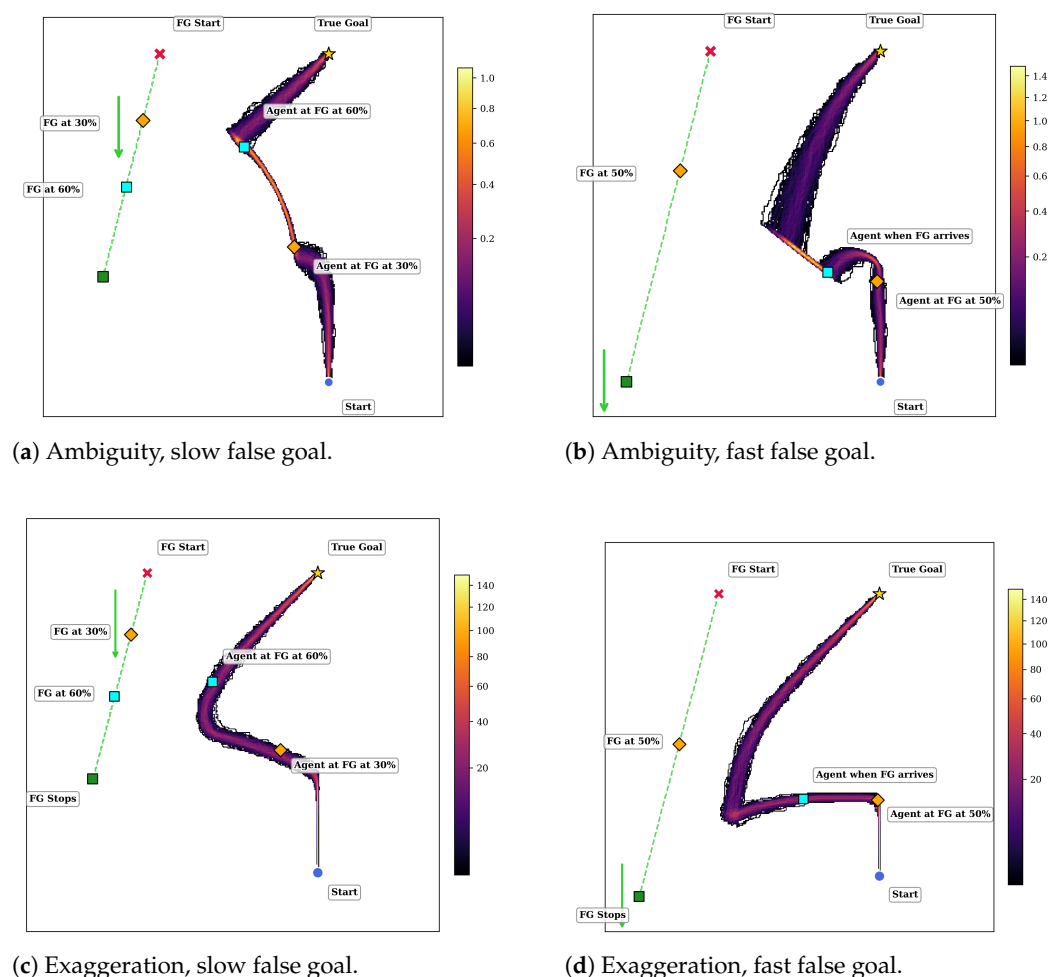


Figure 7. Deception with a moving false goal. Each panel shows 500 trials. Ambiguity, panels (a,b), and exaggeration, panels (c,d), with slow and fast false goals moving from top-to-bottom. Deceptive path planning dynamically responds to the moving false goal before eventually turning toward the true goal.

With ambiguity deception shown in Figure 7a,b, as the false goal moves, so does the bisector between the false and true goal. The agent maintains ambiguity by tracking the moving bisector before eventually committing to the true goal. With exaggeration deception in Figure 7c,d, the agent initially steers toward the moving false goal, with the direction of deceptive movement tracking the false goal movement. Ultimately, the agent redirects toward the true goal.

These examples illustrate the dynamic flexibility of our approach, and many interesting behaviors may be induced through various parameter tunings, including adding and dropping goals and false goals, and altering the form of deception and its emphasis. Note also that our method can be generalized to target pursuit with deception, where the moving goal is the perceived target, and a virtual moving false goal is employed to confuse the target by not following an optimal pursuit path.

5.3. Multi-Agent Deception with Joint Budget Constraint

Next, we present a multi-agent example, where a joint constraint induces coupling among the agents. We consider a multi-agent team of N cooperative agents $\mathcal{I} = \{1, \dots, N\}$ moving on a graph $(\mathcal{V}, \mathcal{E})$ with neighborhood map $\mathcal{N} : \mathcal{V} \rightarrow 2^{\mathcal{V}}$. The team seeks to reach a true goal set $\mathcal{G}_T \subseteq \mathcal{V}$ while feigning a false goal set $\mathcal{G}_F \subseteq \mathcal{V}$ to deceive an observer. Agent $i \in \mathcal{I}$ follows a path $\tau_i = (s^i(0), \dots, s^i(T))$, where $s^i(t+1) \in \mathcal{N}(s^i(t))$. Let $\tau \triangleq (\tau_1, \dots, \tau_N)$ denote the joint path, and $\phi(\cdot)$ denotes the coupled cost term. Following (12) and (13), the time-indexed cost at each step t is

$$\begin{aligned} \mathcal{C}_t \equiv \mathcal{C}(\Theta_t, \Gamma_t) = & - \sum_{i \in \mathcal{I}} \kappa_i \Delta_i^{\text{prog}}(t) \\ & - \sum_{i \in \mathcal{I}} \beta_i \Delta_i^{\text{dec}}(t) + \phi(S(t)), \end{aligned} \quad (33)$$

where for each agent $i \in \mathcal{I}$ we denote κ_i and β_i as the weight coefficients for progress to the true goal and exaggeration to a false goal, respectively.

A standard measure of progress to the true goal $G^i(t)$ is the one-step distance drop,

$$\Delta_i^{\text{prog}}(t) = \max \{0, d(s^i(t), G^i(t)) - d(s^i(t+1), G^i(t))\} \quad (34)$$

and considering we apply exaggeration as the deception form, a convenient exaggeration term toward the false goal $F^i(t)$ is then

$$\Delta_i^{\text{dec}}(t) = \max \{0, d(s^i(t), F^i(t)) - d(s^i(t+1), F^i(t))\}. \quad (35)$$

Note that both (34) and (35) are non-negative. More progress per step is desired, so negative signs appear in (33) to incentivize larger steps by reducing these costs.

Let the team of agents share a total joint step budget $B_{\text{team}} > 0$, representing the total resources (a proxy for energy) available for motion over the fixed time horizon T . For each agent $i \in \mathcal{I}$, the per-step resource consumption is denoted by $b_i(s^i(t), s^i(t+1)) \geq 0$. The cumulative consumption across all agents is thus restricted by the shared team capacity

$$\sum_{t=0}^{T-1} \sum_{i \in \mathcal{I}} b_i(s^i(t), s^i(t+1)) \leq B_{\text{team}}. \quad (36)$$

Now, the multi-agent DPP is posed as minimizing the sum of local per-step costs while satisfying the total budget,

$$\begin{aligned} \min_{\tau} \quad & \sum_{t=0}^{T-1} \mathcal{C}_t \\ \text{s.t.} \quad & s^i(t+1) \in \mathcal{N}(s^i(t)), \quad \forall i \in \mathcal{I}, \forall t \\ & \sum_{t=0}^{T-1} \sum_{i \in \mathcal{I}} b_i(s^i(t), s^i(t+1)) \leq B_{\text{team}}. \end{aligned} \quad (37)$$

This explicitly incorporates the joint budget-limiting inequality constraint to enforce the team's resource capacity.

At each time step t , the next joint move $S(t+1)$ is selected from the set of feasible joint actions $\mathcal{A}(t)$ using the Boltzmann policy with the per-step cost $\mathcal{C}_t$. Feasible joint paths are restricted to those that maintain the shared budget constraint (36) while ensuring sufficient remaining capacity for agents to reach the true goals.

We instantiate a two-agent ($N = 2$) joint Boltzmann planner under the shared-capacity budget formulation (Figure 8). At each time t , a joint action is sampled with $\lambda = 0.6$ from the feasible set $\mathcal{A}(t)$, which is pruned to ensure the cumulative non-optimal expenditure D_{used} never exceeds the deception capacity D_{max} . For this setup, we set $B_{\text{team}} = 1500$ and $\gamma = 0.3$, yielding $D_{\text{max}} = 450$ as the “strategic slack” extracted from the initial energy reserve. The cost $\mathcal{C}_t$ in (33) is used with $\kappa_i = 1.0$ and $\beta_i(f) = w_F(f)$, where the intensity $w_F(f)$ follows a triangular ramp,

$$w_F(f) = w_{\min} + (w_{\max} - w_{\min}) \max(0, 1 - |2f - 1|), \quad (38)$$

with $w_{\min} = 0.5$ and $w_{\max} = 15.0$. The fraction $f = D_{\text{used}}/D_{\text{max}}$ measures the depletion of this finite resource. To regulate efficiency, the coupling term $\phi(S(t))$ is instantiated as a cost on the collective energy increment

$$\phi(S(t)) = \alpha \sum_{i \in \mathcal{I}} \Delta \varepsilon_i(t), \quad (39)$$

where $\alpha = 0.45$ and $\Delta \varepsilon_i(t) = \max\{0, e_i + \Delta d_G^i(t)\}$ represents the net energy cost relative to the goal. For this coupled energy cost, we favor joint paths that use less total incremental energy, and consequently, $\phi(S(t))$ is added in (33) without a sign change.

Four deceptive path cases are shown in Figure 8 with varying start positions and relative energy budget per-agent. The figure shows (a) asymmetric costs (1, 4) with identical starts, (b) symmetric costs (1, 1) with identical starts, (c) symmetric costs (1, 1) from distinct starts, and (d) asymmetric costs (1, 4) from the geometry of (c) to provide a consistent baseline for comparison. For each case, we plot several random realizations for each agent. In this setup, agents with identical marginal costs share the deceptive burden equally, resulting in similar trajectories. Conversely, an agent with a lower marginal cost possesses a higher effective budget that enables increased deception, leading to a more pronounced deceptive deviation from the efficient path.

5.4. Coupled Multi-Agent Transportation Example

We next consider a cooperative transportation scenario with two pairs of robots carrying rigid bars through assigned narrow doors. The setup is shown in Figure 9, with pairs A-B and C-D each transporting a bar. Robots A and B transport one rigid bar directly toward “Door A”, whereas robots C and D first exaggerate toward Door A before reaching their true goal, “Door B”. The trajectories are generated using Algorithm 2, where a random policy is constructed over the four-agent joint action space. The cost is defined as

$$\mathcal{C}_t = -\kappa_{AB} \Delta_{AB}^A(t) - u(t) \alpha_{CD} \Delta_{CD}^A(t) - (1 - u(t)) \kappa_{CD} \Delta_{CD}^B(t), \quad (40)$$

where

$$\Delta_{AB}^A(t) = [d(m_{AB}(t), D_A) - d(m_{AB}(t+1), D_A)]_+,$$

$$\Delta_{CD}^A(t) = [d(m_{CD}(t), D_A) - d(m_{CD}(t+1), D_A)]_+,$$

and

$$\Delta_{CD}^B(t) = [d(m_{CD}(t), D_B) - d(m_{CD}(t+1), D_B)]_+,$$

with $m_{AB}(t) = (s_A(t) + s_B(t))/2$ and $m_{CD}(t) = (s_C(t) + s_D(t))/2$. The rigid-bar constraint is enforced by requiring $\|s_A(t+1) - s_B(t+1)\| = L_b$ and $\|s_C(t+1) - s_D(t+1)\| = L_b$ for every feasible joint action. In Figure 9, robots A-B take a shortest (non-deceptive) path to Door A, whereas robots C-D show exaggeration towards the false goal (Door A), before passing through Door B.

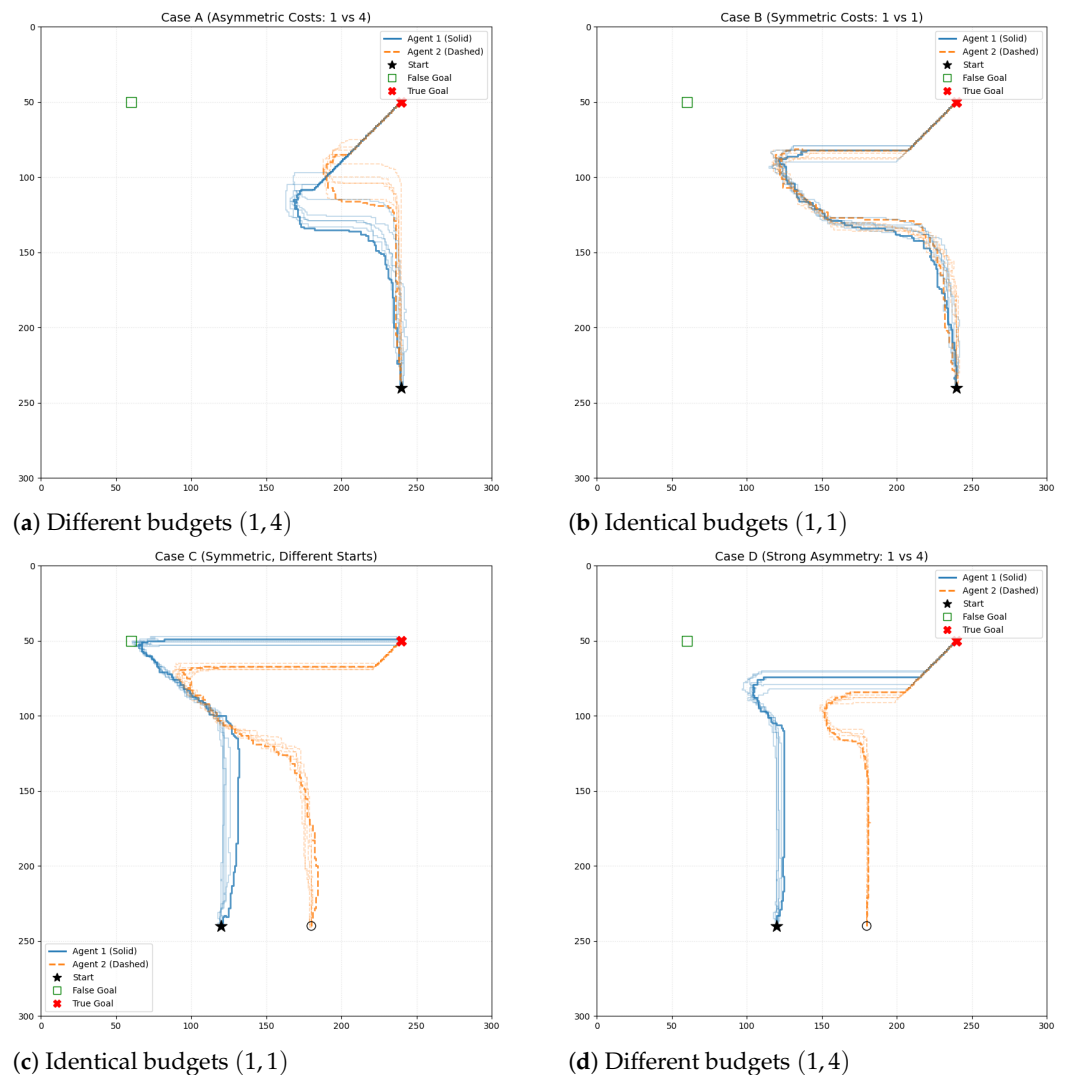


Figure 8. Two-agent deception example with total move joint budget constraints. (a) Different relative agent budgets (1, 4), same start location. (b) Same agent budgets (1, 1), same start. (c) Same agent budgets, different starts. (d) Different budgets (1, 4), different starts. Higher agent budgets enable more deceptive path variation with respect to the most efficient start-to-goal path.

5.5. Volleyball Game: Deceptive Team Attack Path Planning

Next, we consider a multi-agent deception scenario in volleyball, modeling the court as a rectangular grid $(\mathcal{V}, \mathcal{E})$. To disguise the setup and location of a spike shot close to the net, the team coordinates its movement before the shot using a “3–1 split” strategy across the left and right attack corridors. Figure 10 provides a coach’s schematic of this role-free 3–1 split strategy, in which three runners threaten the false corridor while one maintains a viable lane toward the true attack corridor. Three attackers move to threaten one corridor to draw the defensive blockers, while a fourth attacker maintains a viable lane in the opposite corridor. This tactical misdirection is designed to force the defense to account for multiple approach vectors to the net simultaneously.

The offense consists of a designated setter a_s and runners $\mathcal{R} = \{1, \dots, N\} \setminus \{a_s\}$. During the offensive sequence, the team aims to converge upon a true attack region $\mathcal{K}_T \subset \mathcal{V}$ while feigning an approach toward a false region $\mathcal{K}_F \neq \mathcal{K}_T$. To realize this team-level intent, we specialize the per-step cost $\mathcal{C}_t$ to include both individual target-tracking and tactical coupling terms, expressed as

$$\mathcal{C}_t = \sum_{i \in \mathcal{I}} d(s_i(t+1), s_i^{\text{ref}}) + \phi_{\text{split}}(t) + \phi_{\text{space}}(t) \quad (41)$$

where s_i^{ref} is a transient reference state assigned dynamically at each step based on the runner's proximity to the assigned tactical roles.

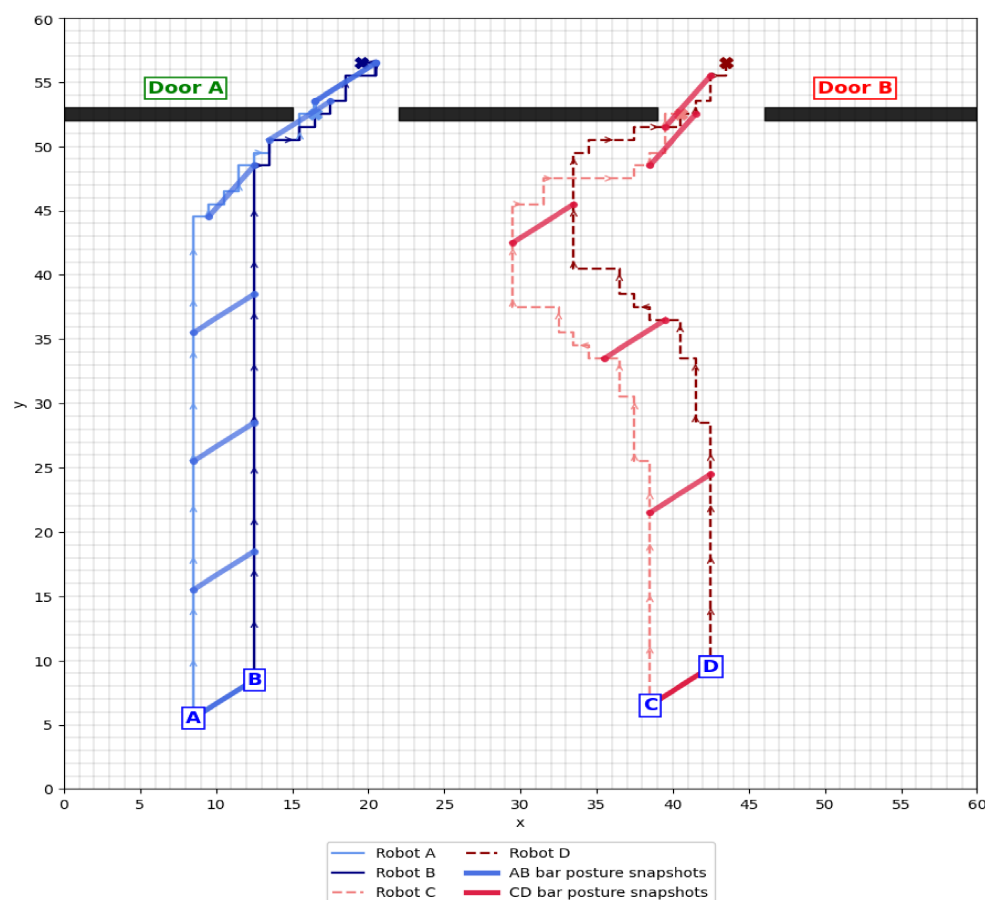


Figure 9. Cooperative transportation example with coupled robot motion. Four robots are organized into two teams, each transporting a rigid bar through an assigned narrow door. Team AB basically follows the shortest path to Door A, whereas Team CD first exaggerates toward Door A before redirecting to Door B under a joint Boltzmann policy. The parameters are $\lambda = 1.8$, $K_t = 3$, $L_b = 5$, $\alpha_{CD} = 5.5$, $\kappa_{AB} = 7.0$, $\kappa_{CD} = 7.0$, $S = 0$, $L = 55$, and $r_{\text{down}} = 16$.

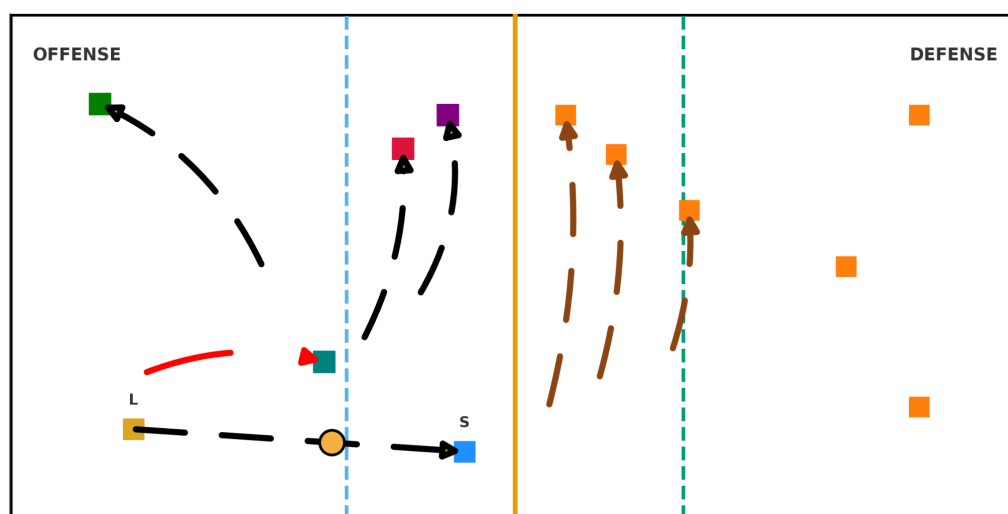


Figure 10. Volleyball coach schematic of the role-free 3-1 “left-right” split strategy, disguising the location of the set and spike attack. The runners aim to stretch the defensive formation toward a false corridor and create a viable lane for the true attack.

We divide the attacking side into two corridors $\sigma : \mathcal{V} \rightarrow \{-1, +1\}$, and let $\sigma_F, \sigma_T \in \{-1, +1\}$ denote the labels associated with the false and true attack corridors, respectively. The number of runners in each corridor is given by

$$\begin{aligned} S_F(t) &= \sum_{i \in \mathcal{R}} \frac{1}{2} [1 + \sigma(s_i(t)) \sigma_F], \\ S_T(t) &= \sum_{i \in \mathcal{R}} \frac{1}{2} [1 + \sigma(s_i(t)) \sigma_T]. \end{aligned} \quad (42)$$

The role-free coupling $\phi_{\text{split}}(t)$ then softly induces the 3–1 distribution by penalizing deviations from this target ratio,

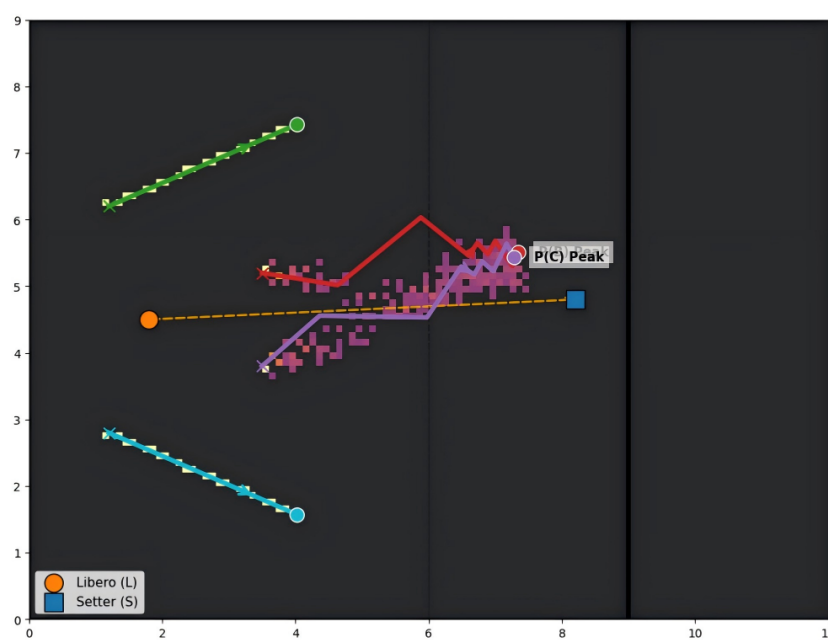
$$\phi_{\text{split}}(t) = \gamma_F (S_F(t) - 3)^2 + \gamma_T (S_T(t) - 1)^2 \quad (43)$$

where we set the coupling gains to $\gamma_F = \gamma_T = 12.0$. To avoid bunching and path crossings, we add a short-range repulsion term,

$$\phi_{\text{space}}(t) = \sum_{\{i,j\} \subseteq \mathcal{R}} \frac{\gamma_{\text{sep}}}{\epsilon + d(s_i(t), s_j(t))} \quad (44)$$

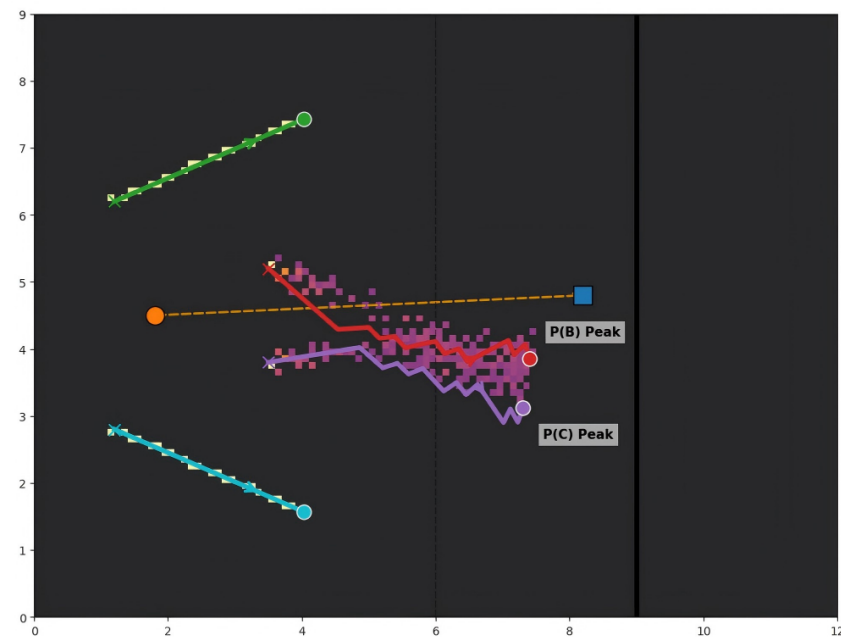
where $\gamma_{\text{sep}} = 0.5$ and $\epsilon = 0.1$ are fixed constants. Substituting the task-specific cost $\mathcal{C}_t$ from (41) into the joint-cost framework of (16) with $\lambda = 6.5$ yields the PMF for our stochastic policy. By sampling joint actions from the resulting PMF accordingly, the runners' trajectories emerge stochastically to satisfy the 3–1 split while adapting to real-time positions.

We run this implementation for 500 trials. Each trial rollout consists of $T = 22$ steps, with the resulting trajectories and visit-frequency heatmap shown in Figure 11. The split cost in (44) favors joint configurations with three runners in the false corridor and one runner in the true corridor, without prescribing which runner occupies each role. Consequently, the runner maintaining the true-side attack lane can vary according to the current team configuration. Figure 11 illustrates this behavior for both choices of the false corridor; when the false corridor is switched between the upper and lower sides, the concentration of runner trajectories shifts accordingly while a viable attack lane is maintained on the opposite side. These results demonstrate that the role-free coupling induces the desired 3–1 deceptive split under the joint stochastic policy.



(a) Upper false corridor tactic ($G_F = \text{UP}$).

Figure 11. Cont.



(b) Lower false corridor tactic ($G_F = \text{DOWN}$).

Figure 11. Volleyball example of the deceptive split over 300 rollouts. The heatmaps demonstrate the emergent 3–1 coordinated misdirection: (a) three runners bias toward the upper false side; (b) three runners bias toward the lower side, while one preserves a viable true-side attack lane.

6. Conclusions and Future Work

This paper introduced a unified deceptive path planning framework that uses a Boltzmann distribution over short-horizon trajectory candidates within a receding-horizon loop. This approach relies on local position and constraint sensing with deterministic cost and produces stochastic trajectories that preserve deception without training. Our method introduces and enables new forms of dynamic deception planning and extends naturally from single- to multi-agent settings via parameter sharing through coupled cost terms. Step-by-step tuning enables new forms of dynamic deception and time-varying cost, adapting to the local environment and incorporating constraints such as an energy budget. This permits, for example, moving goals and false goals, goal reassignment, role swapping among agents, changing the form of deception, and tuning the desired level of deception. Similar to other deceptive path planners, the work so far has only modeled a passive adversarial observer. Nevertheless, our framework opens the door to incorporating adversarial agents. For example, the use of moving true and false goals can be applied to deceptive pursuit-evasion.

It is of interest to further develop decentralized and partially observed variants that limit communications while preserving coordinated team deception (for example, using the common model of restricting to nearest neighbor communications). It is also of interest to introduce perception, networking, and new forms of dynamic deception. For example, adaptive deception may be linked to perception of adversarial agents. Learning cost weights and deception features from data (e.g., human demonstrations or adversarial self-play) may broaden applicability. Finally, we believe the method can be more generally applied in other dynamic deceptive resource allocation settings.

Author Contributions: Conceptualization, X.F., B.M.S. and R.S.B.; methodology, X.F., B.M.S. and R.S.B.; software, X.F.; validation, X.F., B.M.S. and R.S.B.; formal analysis, X.F., B.M.S. and R.S.B.; investigation, X.F., B.M.S. and R.S.B.; writing—original draft preparation, X.F., B.M.S. and R.S.B.; writing—review and editing, X.F., B.M.S. and R.S.B.; visualization, X.F.; supervision, B.M.S. and

R.S.B.; project administration, R.S.B.; funding acquisition, R.S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Office of Naval Research (ONR) under grant number N00014-22-1-2626.

Data Availability Statement: Data used for simulations may be generated as described in the examples.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fregene, K.; Kennedy, D.C.; Wang, D.W.L. Toward a systems- and control-oriented agent framework. *IEEE Trans. Syst. Man. Cybern. Part B* **2005**, *35*, 999–1012. [\[CrossRef\]](#)
2. Fu, J.; Li, Y.; Liao, Y.; Zhang, K.; Zhu, H.; Xu, S. Mission-Driven Trajectory Homotopy to Explore Dynamic Coverage of USV-UAV Systems. *IEEE Trans. Syst. Man. Cybern. Syst.* **2025**, *55*, 8877–8888. [\[CrossRef\]](#)
3. Wang, T.; Li, Y.; Huang, P. A Universal Reactive Approach for Graph-Based Persistent Path Planning Problems With Temporal Logic Constraints. *IEEE Trans. Syst. Man. Cybern. Syst.* **2025**, *55*, 6696–6709. [\[CrossRef\]](#)
4. Papaioannou, S.; Kolios, P.; Theocharides, T.; Panayiotou, C.G.; Polycarpou, M.M. Distributed Search Planning in 3-D Environments With a Dynamically Varying Number of Agents. *IEEE Trans. Syst. Man. Cybern. Syst.* **2023**, *53*, 4117–4130. [\[CrossRef\]](#)
5. Tian, D.; Fang, H.; Yang, Q.; Wei, Y. Decentralized Motion Planning for Multiagent Collaboration Under Coupled LTL Task Specifications. *IEEE Trans. Syst. Man. Cybern. Syst.* **2022**, *52*, 3602–3611. [\[CrossRef\]](#)
6. Yu, X.; Hsieh, M.A. Synthesis of a Time-Varying Communication Network by Robot Teams With Information Propagation Guarantees. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1413–1420. [\[CrossRef\]](#)
7. Zhou, Y.; Hu, H.; Liu, Y.; Lin, S.W.; Ding, Z. A Real-Time and Fully Distributed Approach to Motion Planning for Multirobot Systems. *IEEE Trans. Syst. Man. Cybern. Syst.* **2019**, *49*, 2636–2650. [\[CrossRef\]](#)
8. Chen, S.; Savas, Y.; Karabag, M.O.; Sadler, B.M.; Topcu, U. Deceptive Planning for Resource Allocation. In Proceedings of the 2024 American Control Conference (ACC), Toronto, ON, Canada, 8–12 July 2024; pp. 4188–4195.
9. Fatemi, M.Y.; Suttle, W.A.; Sadler, B.M. Deceptive Path Planning via Reinforcement Learning with Graph Neural Networks. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Auckland, New Zealand, 6–10 May 2024; pp. 2258–2260.
10. Savas, Y.; Verginis, C.K.; Topcu, U. Deceptive Decision-Making Under Uncertainty. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Vol. 36, pp. 5332–5340. [\[CrossRef\]](#)
11. Xu, K.; Zeng, Y.; Qin, L.; Yin, Q. Single Real Goal, Magnitude-Based Deceptive Path-Planning. *Entropy* **2020**, *22*, 88. [\[CrossRef\]](#)
12. Fu, J. On Almost-Sure Intention Deception Planning that Exploits Imperfect Observers. In Decision and Game Theory for Security (GameSec 2022). *Lect. Notes Comput. Sci.* **2022**, *13727*, 58–78.
13. Lv, P.; Li, S.; Yin, X. Optimal Deceptive Strategy Synthesis for Autonomous Systems Under Asymmetric Information. *IEEE Trans. Intell. Veh.* **2024**, *9*, 6108–6119. [\[CrossRef\]](#)
14. Price, A.; Pereira, R.F.; Masters, P.; Vered, M. Domain-Independent Deceptive Planning. In Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems, London, UK, 29 May–2 June 2023; pp. 95–103.
15. Lenhard, C.; Leão, T.R.S.; Bordini, R.H.; Silva, L.A.L. The CADETOPATH Framework: Case-Based Deceptive Topographic Path Planning for Computing, Recording, and Reusing Deceptive Path Plans in Agent-Based Simulation Systems. *SSRN* **2025**. [\[CrossRef\]](#)
16. Xu, K.; Hu, Y.; Zeng, Y.; Yin, Q.; Yang, M. Improving the Scalability of the Magnitude-Based Deceptive Path-Planning Using Subgoal Graphs. *Entropy* **2020**, *22*, 162. [\[CrossRef\]](#)
17. Xue, Y.; Chen, W. Efficient Deceptive Path Planning for UAVs via Attention-Based Reinforcement Learning. *IEEE Trans. Netw. Sci. Eng.* **2026**, *13*, 539–551. [\[CrossRef\]](#)
18. Chen, D.; Zeng, Y.; Zhang, Y.; Li, S.; Xu, K.; Yin, Q. Deceptive Path Planning via Count-Based Reinforcement Learning under Specific Time Constraint. *Mathematics* **2024**, *12*, 1799. [\[CrossRef\]](#)
19. Dragan, A.; Holladay, R.; Srinivasa, S. Deceptive Robot Motion: Synthesis, Analysis and Experiments. *Auton. Robots* **2015**, *39*, 331–345. [\[CrossRef\]](#)
20. Root, P.J.; De Mot, J.; Feron, E. Randomized Path Planning with Deceptive Strategies. In Proceedings of the 2005 American Control Conference (ACC), Portland, OR, USA, 8–10 June 2005; pp. 1551–1556.
21. Masters, P.; Sardina, S. Deceptive Path-Planning. In Proceedings of the International Joint Conference on Artificial Intelligence Proceedings, Melbourne, VIC, Australia, 19–25 August 2017; pp. 4368–4375.

22. Gutierrez, N.B.; Sadler, B.M.; Beksi, W.J. Agent Deception via Polynomial Path Planning. *Eng. Appl. Artif. Intell.* **2025**, *159*, 111205. [[CrossRef](#)]
23. Root, P.J. Collaborative UAV Path Planning with Deceptive Strategies. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2005.
24. Probine, C.; Karabag, M.O.; Topcu, U. A Decentralized Shotgun Approach for Team Deception. In *Decision and Game Theory for Security*; Sinha, A., Fu, J., Zhu, Q., Zhang, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2024; Volume 14908.
25. Suttle, W.A.; Milzman, J.; Karabag, M.O.; Sadler, B.M.; Topcu, U. Value of Information-based Deceptive Path Planning Under Adversarial Interventions. In Proceedings of the 2025 IEEE 64th Conference on Decision and Control (CDC), Rio de Janeiro, Brazil, 10–12 December 2025; pp. 3728–3735.
26. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
27. Auer, P.; Cesa-Bianchi, N.; Fischer, P. Finite-Time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* **2002**, *47*, 235–256. [[CrossRef](#)]
28. Bubeck, S.; Cesa-Bianchi, N. Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit Problems. *Found. Trends Mach. Learn.* **2012**, *5*, 1–122. [[CrossRef](#)]
29. Thompson, W.R. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika* **1933**, *25*, 285–294. [[CrossRef](#)]
30. Boltzmann, L. Weitere Studien über das Wärmegleichgewicht unter Gasmolekülen. *Sitzungsberichte Kais. Akad. Wiss.* **1872**, *66*, 275–370.
31. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
32. Okumura, K.; Yang, G.; Gao, Z.; Woo, H.; Prorok, A. Concrete multi-agent path planning enabling kinodynamically aggressive maneuvers. *npj Robot.* **2026**, *4*, 20. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.