International Journal of Innovative Computing, Information and Control
Volume **18**, Number **6**, December **2022**

ICIC International ©2022 ISSN 1349-4198
pp. 1667–1688

LEARNING MULTI-OBJECTIVE DECEPTION IN A TWO-PLAYER DIFFERENTIAL GAME USING REINFORCEMENT LEARNING AND MULTI-OBJECTIVE GENETIC ALGORITHM

Amirhossein Asgharnia, Howard Schwartz and Mohamed Atia

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada
amirhosseinasgharnia@cmail.carleton.ca; schwartz@sce.carleton.ca; mohamed.atia@carleton.ca

Received April 2022; revised August 2022

Abstract. *In this paper, a framework is established to model a deceitful agent and train it in an adversarial two-player game. In the game, a player uses multi-objective deception to manipulate its opponent's belief about its true intention. In this regard, the player is trained to switch between different strategies based on its state in the game. There is a lower-level policy, which stores the policy to carry out a primitive task. Moreover, there is a higher-level policy, which changes the desired task in different states. The game of guarding territories is utilized to investigate the control mechanism. The lower-level policy is trained via the fuzzy actor-critic learning (FACL) algorithm, and the higher-level policy is extracted via the non-dominated sorting genetic algorithm II (NSGA-II). The results show that by implementing a two-level policy, the invader can increase its pay-off against a non-deceptive situation. In addition, a comparison is conducted between the higher-level policies based on their input information.*

Keywords: Reinforcement learning, Hierarchical reinforcement learning, Multi-agent systems, Deception, Guarding a territory

1. **Introduction.**

Deception has a long and deep history among humans and animals. They can take advantage of concealing the truth or acting dishonestly in nature. Generally, the deception concept can be present in any game with adversarial elements [25]. In other words, one may utilize deception in programming an agent to alter its performance in a multi-agent system. Deception is addressed in many fields such as robotics [31,34], active persistent threats (ATP) [5], and deception detection in the text by natural language processing [29].

Several definitions are proposed to describe the deception concept. Despite the similarity, there are a few differences between them [6,13,19,34]. However, the definition by Bond and Robinson is generally accepted. Bond and Robinson [6] define deception as any conscious or unconscious, intentional or unintentional belief manipulation. Most of the articles use a game theory approach to model deception [13,19].

In [28] deception detection is addressed in multi-agent systems. Although psychologists can detect fraud by nonverbal clues, this paper proposes a reasoning process to detect deceit. They followed a model to detect inconsistencies between the actions and the goal [18]. They implemented a Bayesian network (BN) to predict the agents' opinions based on their relationship with others. Their results show the better performance of using BN in deception detection rather than face-to-face evaluation.

DOI: 10.24507/ijicic.18.06.1667

Another application of deception in robotics is studied in [34]. In this study, the researchers investigate the algorithms to justify the usage of deception. They showed that there are many cases where deception usage does not significantly improve the agent's performance. In other words, using deception does not guarantee a successful result for the deceiver.

Robotic deception by using fuzzy inference system is studied in [19, 20]. In [19], the researchers addressed deception and trust in a multi-agent game simultaneously. In the paper, the researchers utilized game theory and fuzzy logic in a deceptive pursuit-evasion game. Robotic deception is also studied in [4, 12] in terms of deceiving humans. In [12], the authors conducted several experiments to study the effect of a robotic arm motion to deceive a human about a robot's target. An application of deception in an adversarial strategic game is studied in [24]. In this study, the agents are two unmanned aerial vehicles (UAVs) and decide their strategy. The results showed that by using deception, the deceiving agent could have better performance.

Deception in a grid world game is studied in [25]. The researchers provided a grid world game such that an agent is programmed to reach a particular cell. In the game, three cells are considered the targets, while only one of them is the invader's intended target. An adversary tries to guess the intended target.

Deception can be studied in the differential games, where there are more than an agent in the game. A deceitful agent can take advantage of using deception to increase its payoff. The game of guarding a territory is one of the examples [35], which can be used as a testbed for studying deception. In the game of guarding a territory, the invader has a target that it wants to invade. However, the desired purpose of the defender is to guard the goal by capturing the invader. The defender has to intercept the invader as far as possible from the target. There are many papers on studying the game of guarding a territory [1-3, 8, 22, 27, 35].

In all mentioned papers, deception is studied from a single-objective point of view. Often, the criterion to measure the deception performance is the number of the times an agent successfully deceives another agent in many games. However, the other aspects of the game are often neglected. For instance, if the deceitful agent uses deception a lot, the deceived agent may learn the deception model and strives to counter deceive. In addition, two different deceptive strategies may lead to a successful game for the deceitful agent. However, one of strategies may come in a higher operation cost. Or, one of the deceptive strategies may have more robustness in non-deterministic environments than other deceptive strategies.

Pursuit-evasion games are a class of games that have applications in surveillance and security missions. These games were initially studied by Isaacs in 1960s [17]. The initial method to solve pursuit-evasion games was based on geometrical approaches, such as the Apollonius circle. Research is abundant in solving pursuit-evasion games, using geometric approaches [14, 15, 21, 26, 35]. In addition, there are several studies on using analytical approaches, such as [14, 16]. However, in this study, we focus on the usage of machine learning approaches in pursuit-evasion games, reinforcement learning in particular.

Guarding a territory in a grid world is investigated in [22]. The minimax Q-learning algorithm and the win-or-learn-fast policy hill-climbing (WoLF-PHC) algorithm are compared in this paper. It is shown that in both algorithms, the players' policy has converged to the Nash equilibrium. It is shown that WoLF-PHC is able to find the optimal strategy in more games.

The game of guarding a territory in a continuous domain is simulated in [27]. The fuzzy Q-learning (FQL) algorithm is implemented to train the invader, while the defender uses an optimal policy based on the Apollonius circle.

Guarding a territory in the presence of one invader and two defenders is the subject of [1]. They investigate a game with one super invader in the presence of two defenders. The fuzzy actor-critic learning (FACL) algorithm is implemented in their study. The results show that both agents can learn their optimal policies simultaneously.

The game can be converted into a suitable platform to study deception in artificial intelligence. In this regard, several targets are added to the game. The invader has one target. However, it can change its intended goal during the game to deceive the defender. Such a game is studied in [3]. In [2], the game is played in a grid world. The Q-tables are derived with the minimax Q-learning algorithm and the single-objective Q-learning algorithm.

In this paper, the game of guarding several territories is simulated in a continuous domain. Two agents are involved in the game: one invader and one defender. There are also three targets in the game, and only one of them is the invader's real goal. However, all three targets are protected by the defender. We proposed a hierarchical strategy system to utilize multi-objective deception.

The motivations of this paper are twofold. This study investigates the possibility of learning deception using reinforcement learning and the genetic algorithm. Recent applications of the genetic algorithm are investigated in [36]. On the other hand, we investigate the multi-objective deception by studying a mobile robot, which strives to manipulate another robot by considering several objectives.

This paper's contributions are as follows.

1) We model multi-objective deception in a continuous domain game.
2) We implement a combination of machine learning and evolutionary optimization approaches to create a bi-level policy mechanism.
3) We test our deception model on the game of guarding several territories and create a deceitful invader.
4) We implement the non-dominated sorting genetic algorithm II (NSGA-II) to find the Pareto optimal strategy of the higher-level policy.

The rest of the paper is organized into five sections. In Section 2, the game of guarding several territories is briefly presented. In Section 3, the fuzzy actor-critic learning algorithm is described. Section 4 is dedicated to the proposed method. In Section 5, the learning process is done, and several tests are conducted. Finally, Section 6 concludes the paper.

2. **Guarding Several Territories.**

Guarding a territory is a standard benchmark in game theory and decision theory. Guarding a territory is the extended version of the pursuit-evasion game, introduced by Isaacs [17], where there is also a target in the game. The target may be stationary or moving [7]. In this game, normally, two agents exist other than the target(s). One of them is an invader, and its goal is to reach a predefined target. The other agent is a defender, which has to protect the goal by intercepting the invader. This game's terminal time is when either the invader reaches the target, or the defender captures the invader. Figure 1 depicts the game.

Two agents and three territories are shown in Figure 1. In the figure, θ_I and θ_D are the steering angles of the invader and the defender, while v is their speed. In Figure 1, G_2 is the true goal. Some of the game characteristics are as follows.

- The game field may be in any size or any shape. However, in this paper, we set a square-shaped game field.
- If the defender is faster than the invader, it is called a *superior defender*, while the invader is called an *inferior invader*. If the invader is faster than the defender, the

invader is called a *superior invader*. In addition, the defender is called an *inferior defender*. It is also possible to assume an equal speed for both agents.

- The agents are considered as cars, and their differential equations are as follows [30]:

$$\begin{cases} \dot{x} = v\cos(\beta) \\ \dot{y} = v\sin(\beta) \\ \dot{\beta} = \dfrac{v\theta}{L} \end{cases} \tag{1}$$

where (x, y) is the car location, β is the heading with respect to the x-axis, and θ is the steering angle. The term L is the distance between the two axles and is set to 0.3 units, which was suggested by [30].

- The action set of both agents is the steering angle of the robot, which is in the interval of $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$. The action set is continuous.
- Unlike [1, 22, 27], in this paper, there is more than one territory. The goal set can be presented as $G = \{G_1, G_2, \dots, G_n\}$. The defender is responsible for protecting all the goals, while only one of them is the invader's true goal.

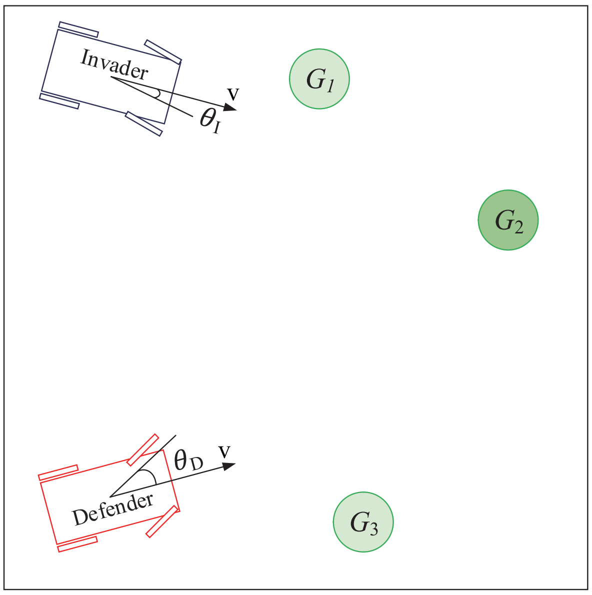

FIGURE 1. The game of guarding a territory with two agents and three goals

In the deceptive game, the invader tries to transmit a false signal to the defender by repeatedly choosing different fake goals. On the other hand, the defender has to guess the invader's true intention and tries to capture the invader.

3. The Fuzzy Actor-Critic Learning Algorithm.

Reinforcement learning (RL) is a branch of machine learning, where the agent tries to learn through direct interaction with an unknown environment. In RL, the agent receives a suitable reward if it achieves the desired outcome. The reward is a numerical value calculated by a reward function. Q-learning is one of the well-studied kinds of RL. The Q-learning algorithm is a repetitive process, which updates the state action values in each epoch. The state value of each action is called Q-value.

In the Q-learning algorithm, the Q-values can be stored in arrays, called Q-tables. However, a disadvantage of the Q-learning algorithm is the size of the Q-table, which becomes intractable as the number of states grows. A solution to this problem is utilizing a function estimator, such as a fuzzy inference system (FIS) or an artificial neural network

(ANN). The FIS has two advantages over the ANN [1]. Firstly, it allows the human designer to incorporate expert knowledge into design. Secondly, only the output parameters of the FIS are updated at each time step. Therefore, with a limited number of IF-THEN rules, the designer can estimate the state values with a reasonable amount of memory.

Two well-known algorithms for training in a continuous domain are the fuzzy Q-learning algorithm (FQL) [27], and the fuzzy actor-critic learning (FACL) algorithm [11]. The FQL algorithm is a modified form of the Q-learning algorithm. However, instead of arrays, the approximation of the Q-values are stored as a fuzzy inference system. In the FQL algorithm, discrete actions are evaluated in each state. Thus, the designer has to break the action interval into some discrete actions. The algorithm dedicates a Q-value to each action, and by using a temporal difference method, the output parameters of the FIS are updated. There are many disadvantages to the FQL algorithm. The first one is that the action set must be discrete. In addition, the action set interval is limited. Last but not least, it is shown that the FACL algorithm can perform better than FQL in terms of convergence speed and pay-off [1].

The fuzzy actor-critic learning algorithm is formed from two parts: the actor and the critic. The actor stores the actions in each state, while the critic gives an estimation of future performance. It is possible to implement different kinds of estimators as the actor or the critic in an actor-critic learning algorithm. It is shown that the FIS outperforms the ANN in terms of learning time and performance [11]. In this paper, a Takagi-Sugeno fuzzy logic controller (FLC) is utilized for the actor. The critic is a Takagi-Sugeno fuzzy inference system and estimates the state values.

The output of the actor is calculated as follows [32],

$$u_t = \sum_{l=1}^{M} \Phi^l \omega_t^l \tag{2}$$

where ω^l is the output parameters of the fuzzy controller. The term Φ^l is the firing strength of the rule l and is defined as follows,

$$\Phi^l = \frac{\partial u}{\partial \omega^l} = \frac{\prod_{i=1}^{n} \mu^{F_i^l(x_i)}}{\sum_{l=1}^{M}\left(\prod_{i=1}^{n} \mu^{F_i^l(x_i)}\right)} \tag{3}$$

where $\mu^{F_i^l(x_i)}$ is the membership degree of the fuzzy set $F_i^l(x_i)$. In each time step, ω^l is updated via

$$\omega_{t+1}^l = \omega_t^l + \beta_L \left\{\Delta\left(\frac{u_t' - u_t}{\sigma}\right)\right\} \frac{\partial u}{\partial \omega^l} \tag{4}$$

where $\beta_L \in (0,1)$, is the actor's learning rate, and $u_t' = u_t + \nu(0,\sigma)$. The term $\nu(0,\sigma)$ is a white noise and it is added to the control signal (u_t) to increase the exploration rate.

The critic's task is to estimate the state value in each time step. Similar to the Q-learning algorithm, we can define the state values as the expected sum of the discounted rewards over time, which is defined as

$$V_t = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \tag{5}$$

where V_t is the state value at time step t, $\gamma \in (0,1)$ is the discount factor, and r_{t+k+1} is the received reward at time step $t+k+1$. We can write (5) in a recursive form as

$$V_t = r_{t+1} + \gamma V_{t+1} \tag{6}$$

After taking each action, the critic gives an approximation on the quality of the new state. The agent can estimate the state value as

$$\hat{V}_t = \sum_{l=1}^{M} \Phi^l \zeta_t^l \tag{7}$$

where $\hat{V}_t$ is the state value approximation, and ζ_t^l is the output parameter of the critic FIS. The prediction error (Δ), also known as the temporal difference, is given as

$$\Delta = r_{t+1} + \gamma \hat{V}_{t+1} - \hat{V}_t \tag{8}$$

The critic consequence parameters are updated as

$$\zeta_{t+1}^l = \zeta_t^l + \alpha_L \Delta \frac{\partial \hat{V}}{\partial \zeta^l} \tag{9}$$

where $\frac{\partial \hat{V}}{\partial \zeta^l}$ is

$$\frac{\partial \hat{V}}{\partial \zeta^l} = \Phi^l = \frac{\prod_{i=1}^{n} \mu^{F_i^l(x_i)}}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu^{F_i^l(x_i)} \right)} \tag{10}$$

4. Proposed Method.

We establish a framework to model a multi-objective deceptive scheme in a two-player game. One of the players uses deception to find a trade-off between its desired pay-offs. The deceitful player implements deception by changing its strategy, based on its state in the game. Its opponent has to find the strategy, then selects a suitable policy to confront the player. The mentioned strategy can be implemented in different games and applications, where there are two groups of fully competitive agents, like chess or pursuit-evasion games. In this paper, the simulation platform is the game of guarding several territories (Section 2) with a deceptive invader. The defender must guard all the targets $G = \{G_1, G_2, \ldots, G_n\}$. However, only one of the members of G is the invader's true goal. The defender tries to guess the true goal by a predefined function, called the *belief function*. The invasion of the true goal is not always possible. Therefore, to increase the invader's pay-off, the invader may initially pretend to invade a false goal. We define the invader's strategy to pretend to invade a fake goal as the deceptive strategy. The proposed method is based on a two-level hierarchical policy system. The first level is called the lower-level policy (LLP), and the second level is called the higher-level policy (HLP).

4.1. Lower-level policy.

In a deceptive game, there are multiple lower-level policies. Each LLP stores the policy to achieve a particular goal or accomplish a task. In the example provided in this paper, the LLP of each agent returns the primitive action, which is the steering angle. The LLP tells the invader which action to take at each state to invade the currently selected goal. The LLP tells the defender which action to take at each state to defend the same goal.

In a discrete world such as the game represented in [22], the states can be defined the occupied tiles number. However, it is more efficient to define the states by the agents' location with respect to each other and the goal in a continuous domain. In [1, 27], the states are defined by three parameters. These parameters were the invader's and the defender's position with respect to each other and the orientation with respect to the goal. In other words, the distance of the agents to the goal was neglected. In those studies, the goal was stationary. In addition, only one goal was involved in the game. Thus, those parameters were sufficient to define the states. However, in this study, we are considering several goals. The LLP must be able to give the proper action to defend or invade each of them. Thus, more parameters are needed to define the game states. In this study, the states are represented by four parameters. These parameters are the actor (FLC) inputs and the critic (FIS) inputs. The invader's and the defender's inputs are as follows,

$$\begin{aligned} \text{Invader's LLP input} &= [d_{ID} \quad d_{IG} \quad \alpha_I \quad \beta_I] \\ \text{Defender's LLP input} &= [d_{ID} \quad d_{DG} \quad \alpha_D \quad \beta_D] \end{aligned} \tag{11}$$

In (11), d_{ID} is the invader's distance to the defender, d_{IG} is the invader's distance to the goal, d_{DG} is the defender's distance to the goal, α_I is the angle between the invader's heading and a straight line between the invader and the defender, while β_I is the angle between the invader's heading and a straight line between the invader and the goal. In addition, the term α_D is the angle between the defender's heading and a straight line between the defender and the invader, and β_D is the angle between the defender's heading and the straight line between the defender and the goal. Figure 2 depicts the parameters in the game. As shown in the figure, there is no global coordinate in the game setup and the agents use information that describes their relative position and orientation with respect to each other. Selecting respective coordinates in our game makes the fuzzy rule-base smaller. In addition, in the mentioned setup, the policies can be used without major changes if the environment changes.

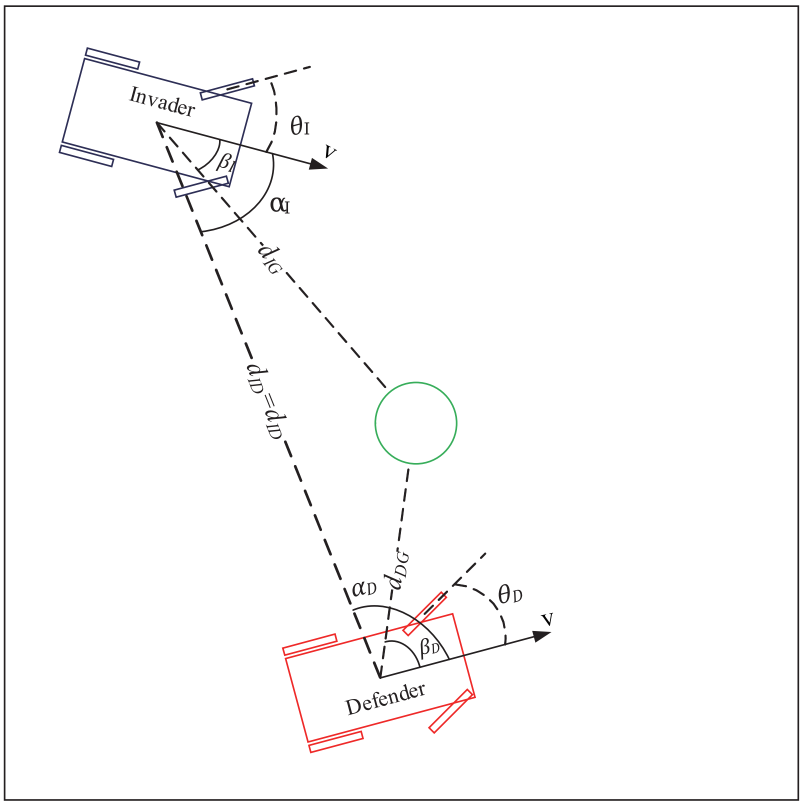

FIGURE 2. The inputs used in the game to evaluate the states

Different types of reward functions can be suggested for the game. Two primary reward functions are the terminal reward function [22], and the instantaneous reward function [27]. In RL problems, with the terminal reward function, the agent only receives the reward if it reaches a terminal state. Terminal reward functions can show satisfactory performance in problems with a small number of states, such as a grid world game, or a single-agent game [22]. However, in continuous domain games, using a terminal reward function can make the learning process slower or, in many cases, impossible [27]. In this paper, we implement an instantaneous reward function to train the LLP, similar to [27]. The reward function of the invader is defined as follows,

$$\begin{aligned} R_{inv} &= (1 - W_I)[d_{ID}(t+1) - d_{ID}(t)] + W_I[d_{IG}(t) - d_{IG}(t+1)] \\ R_{def} &= W_D[d_{ID}(t) - d_{ID}(t+1)] + k(1 - W_D)[d_{DG}(t) - d_{DG}(t+1)] \end{aligned} \tag{12}$$

In (12), R_{inv} and R_{def} are the invader's and defender's reward functions, respectively. The term W_I is a weight that amplifies the importance of reaching the goal in comparison to keeping a distance from the defender. The term W_D is a weight that amplifies the importance of chasing the invader rather than moving toward the goal. The term k is 1 if $d_{DG}(t) - d_{DG}(t+1) > 0$ and otherwise it is zero. Equation (12) shows that both reward functions consist of two conflicting objectives. Thus, the reward function could be written as vectors instead of scalers. Such a problem with a vector reward function is called a multi-objective Markov decision process (MOMDP). Although there are a few multi-objective RL algorithms to solve MOMDP, such as [23, 33], they are all in the discrete domain. Therefore, in this paper, we will find W_I and W_D with a different approach.

The learning process starts with initializing the agents' and target's locations randomly. In addition, the actor's and critic's output parameters $\left(\omega_0^l \text{ and } \zeta_0^l\right)$ are set to zero. In each state, the agents choose their actions by utilizing the actor component. The action, which is the steering angle of the car, is calculated via (2). White noise is added to the output to increase the exploration rate. After taking an action and receiving the reward, the critic value is calculated by (7). The prediction error is calculated via (8). The output parameters of the actor and the critic (ω_l and ζ_l) are updated by (4) and (9).

If a terminal state is met or the simulation time is reached, then the game will be restarted by reinitializing the agents' locations. However, the actor's and the critic's output parameters continue adapting.

4.2. **Higher-level policy (HLP).** The most essential element in a deceptive game is the HLP. The HLP is a controller, which can change the current LLP to another. Thus, the HLP changes the agent's task or goal. The quality of deception lies on the *state*, where the HLP returns another task or goal.

In the game of guarding several territories, the HLP returns a goal number at each time step. The goal number may be the real goal or a fake goal. Then, the invader calculates its goal-related inputs, such as d_{IG} and β_I for the selected goal. There are many states for which, the invader is not able to win the game against an optimal defender. In these states, the invader's HLP may return a fake goal. Thus, the invader pretends it is trying to invade another goal and conceals its real goal. The defender also has an HLP, which is called the *belief function.* The belief function guesses the intention of the invader. If the invader repeatedly changes its goal, it misleads the defender's belief function.

4.2.1. *Defender's higher-level policy.* The defender has a belief about the invader's true intention at each time step. In this paper, a hardcoded function is defined to model the defender's HLP. Equation (13) describes the belief function. The defender knows the exact location of the invader and the goals. Thus, by inspecting the invader's location at each time step, it guesses the invader's probable target. The belief function returns the goal number that the invader has gotten closer to in the past time step. If the invader is in its first time step, the belief function returns the closest goal to the invader. Equation (13) shows the defender's belief function.

For $t > 1$

$$D = \underset{j\in\{1,2,\ldots,n\}}{\arg\min} \left(\sqrt{\left(x_I^t - x_{G_j}\right)^2 + \left(y_I^t - y_{G_j}\right)^2} - \sqrt{\left(x_I^{t-1} - x_{G_j}\right)^2 + \left(y_I^{t-1} - y_{G_j}\right)^2} \right)$$

$$\text{For } t = 1 \qquad (13)$$
$$D = \underset{j\in\{1,\ldots,n\}}{\arg\min} \sqrt{\left(x_I^t - x_{G_j}\right)^2 + \left(y_I^t - y_{G_j}\right)^2}$$

where $\left(x_I^t, y_I^t\right)$ is the invader's location at time step t, $\left(x_{G_j}, y_{G_j}\right)$ is the location of goal j. In addition, D is the defender's belief. The term n is the total number of the goals. In each time step, the defender's goal-related inputs, such as d_{DG} and β_D, are calculated for goal D.

4.2.2. *Invader's higher-level policy.* The invader's HLP tells the invader the best goal to follow to maximize the defined pay-off. In other words, the invader can conceal its true goal and manipulates the defender's belief function by changing its goal repeatedly. Therefore, unlike the LLP, where the output is the steering angle of the agent, in the HLP the output is a goal number. The output of the HLP may be the real goal, or a fake goal.

To create the invader's HLP, a Takagi-Sugeno FIS structure is formed. The goal is to find the proper output parameters of the FIS. To find the best goal to chase, the invader may have different information about the game states. It seems trivial that if the invader has more information, it can perform a better deceptive behaviour. As such, we want to measure the deception performance for different sets of input parameters for the invader's HLP. Thus, we introduce three deceptive cases by choosing different sets of parameters as the inputs. Then, we examine the performance of those input sets. Different FIS input sets are presented.

The first deceptive case contains four parameters given as

$$\mathit{Input\ 1} = [X_{IG} \quad Y_{IG} \quad X_{ID} \quad Y_{ID}]$$

where (X_{IG}, Y_{IG}) is the Manhattan distance between the invader and the true goal. In addition, (X_{ID}, Y_{ID}) is the Manhattan distance between the invader and the defender.

The second deceptive case with four parameters is

$$\mathit{Input\ 2} = [X_{IG} \quad Y_{IG} \quad \beta_I \quad d_{ID}]$$

where β_I is the line of sight from the invader toward the defender, and d_{ID} is the distance between the invader and the defender.

The third case has four inputs given as

$$\mathit{Input\ 3} = [X_{IG} \quad Y_{IG} \quad d_{ID} \quad \theta_D]$$

where the term θ_D is the defender's heading. We can assume that the invader knows or can estimate the defender's heading with a Kalman filter.

To find the output parameters of the FIS, we implement the non-dominated sorting genetic algorithm II (NSGA-II). NSGA-II is an iterative and population-based optimization algorithm. Each population member (called the chromosome) represents the output parameters of the FIS. The chromosomes are formed of smaller components, called genes, which contain each rule's output parameter. The game is simulated for each chromosome as the fuzzy output parameters separately. The fitness functions are calculated based on the simulation result. We should notice that each game may have two conclusions: 1) the invader wins the game by a successful invasion, or 2) the defender captures the invader before it reaches the target. We are interested in successful invasions. So if the defender captures the invader, a penalty function is assumed, resulting in a poor fitness evaluation. Equation (14) shows the two fitness functions and the assigned penalties.

$$\begin{array}{l}\text{The invader successfully reaches the target:}\\ \begin{cases} Fitness_1 = \min \dfrac{1}{\sqrt{\left(x_D^{ter}-x_I^{ter}\right)^2+\left(y_D^{ter}-y_I^{ter}\right)^2}} \\ Fitness_2 = \min m \end{cases}\\ \text{The defender captures the invader:}\\ \begin{cases} Fitness_1 = 1{,}000 \\ Fitness_2 = 1{,}000 \end{cases}\end{array} \tag{14}$$

where $\left(x_D^{ter}, y_D^{ter}\right)$ is the defender's location in the terminal state and $\left(x_I^{ter}, y_I^{ter}\right)$ is the invader's location in the terminal state. The term m is the number of times that the invader changes its goal.

Equation (14) represents two objectives as fitness functions in the two cases. The first case is assigned if the invader can reach the goal. In this case the objectives are 1) to be far from the defender at the terminal state, and 2) to perform the game with a minimum number of goal changes. The second case is for the times that the invader fails to reach the target. In this case, penalty functions are assigned, which can be two large numbers. In this case, we assigned a penalty of 1,000. Thus, the chromosomes that lead the game to failure will be poorly evaluated, and they will be eliminated from the population.

To produce a suitable compromise between two objectives, we utilized the concept of Pareto dominance. For every pair of vectors $\vec{v}, \vec{v'} \in \mathbb{R}$, $\vec{v}$ dominates $\vec{v'}$ $\left(\vec{v} \succeq \vec{v'}\right)$ iff there exists a dimension i such that $v_i > v_i'$ and for no dimension j there is $v_j < v_j'$. If $\vec{v}$ does not dominate $\vec{v'}$ and $\vec{v'}$ does not dominate $\vec{v}$, we say $\vec{v}$ and $\vec{v'}$ are non-dominated. For a given set $V \in \mathbb{R}^n$, the set of non-dominated vectors (ND(V)) is called *Pareto front* and is defined as follows,

$$\mathrm{ND}(V) = \left\{\forall \vec{v} \in V \middle| \nexists \vec{v'} \in V, \vec{v'} \succ \vec{v}\right\} \tag{15}$$

We utilized the Pareto front to show the fitness functions in (14).

The invader's HLP optimization flowchart is depicted in Figure 3. In the figure, two main blocks are depicted. The first block represents the NSGA-II algorithm. The second main block shows how a game is simulated to calculate the objective functions. The FIS

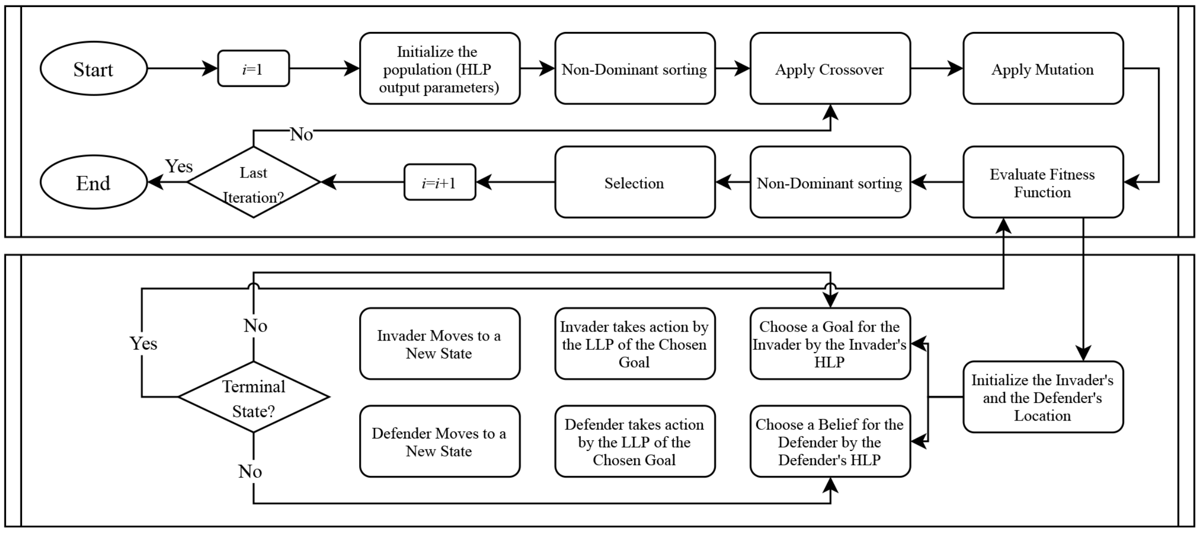
Start
i=1
Initialize the population (HLP output parameters)
Non-Dominant sorting
Apply Crossover
Apply Mutation
No
Yes
End
Last Iteration?
i=i+1
Selection
Non-Dominant sorting
Evaluate Fitness Function
Yes
No
Terminal State?
No
Invader Moves to a New State
Defender Moves to a New State
Invader takes action by the LLP of the Chosen Goal
Defender takes action by the LLP of the Chosen Goal
Choose a Goal for the Invader by the Invader's HLP
Choose a Belief for the Defender by the Defender's HLP
Initialize the Invader's and the Defender's Location

FIGURE 3. The HLP optimization flowchart

output parameters will be derived for the objective functions by repeating the NSGA-II operators (non-dominated sorting, crossover, and mutation) on the chromosomes.

4.3. **Optimal strategy.** The classical optimal strategy can be found using the perpendicular bisector method if both agents' speeds are equal [17]. In this approach, the optimal capture point is the closest point on the agents' bisector line to the goal. However, if the speeds are not the same, and the defender has a higher speed than the invader, the Apollonius circle may be implemented to evaluate the optimal strategy [1]. In the game of guarding a territory with two agents, foci represent the invader and the defender. If the invader and the goal are on the same side of the circle, the invasion will be guaranteed. If not, the closest distance between the goal and the Apollonius circle is the optimal capture point. The Apollonius circle always encircles the inferior agent, which is the invader. In games that the defender is superior, it can capture the invader at a point. However, in games that the invader's speed is higher, a positive radius is defined for the defender, noted as the capture radius. In the games that the invader's speed is higher, and a positive capture radius is assumed, implementing Cartesian ovals is proposed [14]. Cartesian ovals are described as follows,

$$\begin{aligned} x_{co} &= x_i + R_{co}\cos(\eta + \phi_{co}) \\ y_{co} &= y_i + R_{co}\sin(\eta + \phi_{co}) \end{aligned} \tag{16}$$

where

$$R_{co}(\phi_{co}) = \frac{\lambda\rho + d\cos(\phi_{co}) \pm \sqrt{\lambda\rho + d\cos(\phi_{co})^2 - (1-\lambda^2)(d^2-\rho^2)}}{1-\lambda^2} \tag{17}$$

for $\phi_{co} \in [-\phi_{co}^i, \phi_{co}^i]$, where

$$\phi_{co}^i = \arccos\left(\frac{\sqrt{(1-\lambda^2)(d^2-\rho^2)} - \lambda\rho}{d}\right) \tag{18}$$

In (16)-(18), d is the distance between the invader and the defender, ρ is the capture radius, and η is the line-of-sight from the defender to the invader. In this paper, we implemented the Cartesian ovals to find the reward weights in (12).

5. **Simulation and Results.**

In this section, the simulation results, the comparisons and the discussion are presented. The control structure to model deception, which was presented in Section 4 is trained using the FACL algorithm, which was presented in Section 3. The LLP is trained via the FACL, while the HLP is trained via NSGA-II. A comparison between the fuzzy Q-learning and the FACL is done in [1] and the results indicate that the FACL algorithm has a better performance.

5.1. **Preliminaries.** The game field is a 50 × 50 units square. The game has three stationary goals for testing the implementation of deception in the game. These goals are presented in Table 1. The goal radius r_G and the capture radius r_C are set to 1.00 unit. There are two agents in the game: one invader and one defender. The agents cannot leave the field, and if they hit the wall, they will remain at the boundary.

TABLE 1. The goals that are defined for the tests

Goals	x	y
G_1	10	40
G_2	25	25
G_3	40	10

Two speed scenarios are proposed to evaluate the effect of deception in the game. In the first scenario, both agents have an equal speed of 1.00 unit/second. In the second scenario, the invader is superior. The defender's speed is 1.00 unit/second, while the invader's speed is 1.10 units/second.

The FACL algorithm is implemented for training the LLP. In the training process, γ is set to 0.9 [1]. In addition, α_L and β_L, which are the learning rates in (4) and (9) are selected as 0.1 and 0.05, respectively [1]. These two coefficients are decayed over the training by $10^{\log\left(\frac{0.1}{MaxEpoch}\right)}$ in each epoch, where *MaxEpoch* is the total number of epochs and is equal to 20,000 games. Therefore, in the last epoch, α_L and β_L are 10% of their initial values. On the other hand, σ is assumed to be 1.0 radian. The maximum simulation time of each game is 200 seconds, with a time step of 0.1 seconds. Each input has 10 triangular membership functions, distributed uniformly within the interval. The inputs that are distances (d_{ID}, d_{IG}, d_{DG}) are in the interval of $\left[0, 50\sqrt{2}\,\right]$. The angular inputs ($\alpha_I$, α_D, β_I, β_D) are in the interval of $[-\pi, \pi]$. There are $5 \times 5 \times 5 \times 5 = 625$ rules for the LLP of each agent.

The NSGA-II algorithm is utilized for training the invader's HLP. We implement the non-dominated sorting algorithm from [10] with integer uniform crossover, and insertion mutation [9]. The total number of epochs is set to 400, while the population is set to 1,000. Each invader's HLP has L^M rules, where L is the number of membership functions, and M is the number inputs. There are seven triangular membership functions, distributed uniformly in the interval. Each gene contains the output parameter of one of the rules chosen from the set $\{1, 2, 3\}$, which represents a goal. Finally, the mutation probability is 20%, while the crossover probability is 80%.

A fuzzy controller such as (2) maps a real input vector to another real value. Thus, in order to discretize the HLP's output, we implemented a uniform clustering. If the fuzzy output is within the $[1, 1.666)$ it is mapped to 1. If the fuzzy output is within the $[1.667, 2.333)$ it is mapped to 2, and if the fuzzy output is within the $[2.333, 3]$, it is mapped to 3.

5.2. **Training the agent's LLP.** The invader and the defender learn their LLPs simultaneously. Ideally, their output parameters converge to their Nash equilibrium after multiple epochs. In the beginning, the FACL algorithm is initialized by setting ω_0^l and ζ_0^l to zeros. A uniform random number generator (URNG) sets random initial positions and headings for each agent. In this stage, we want to train the agent's LLP to handle any goal location in the game field. Thus, in each epoch, a random location is chosen for the goal by the URNG. With this approach, the agent's LLP will return the suitable action for any chosen goal, including the three goals mentioned in Section 5.1. The agents choose their actions by the actor component. At each time step, a reward is received, and the actor and critic output parameters are updated via (4) and (9), respectively. The process continues until the maximum simulation time is over or a terminal state is met. If the game finishes, the agents and goal locations are reinitialized in random locations. However, the output parameters of the actor and critic continue to adapt. The simulation of each game is considered as one epoch.

The process is done for each speed scenario separately. Thus, in the end, there are three sets of output parameters for the superior defender scenario, the equal speed scenario, and the inferior defender scenario.

The reward function weights (W_I and W_D) in (12) have a significant effect on the LLP performance. Different weights will cause different terminal states in the game. The reason for using the optimal policy given by Cartesian ovals is to evaluate W_I and W_D in the reward function (12). In this paper, we find the weights by trial and error that match the optimal policy. The training process is done for different W_Is and W_Ds, separately. When a training process is done, a test is conducted to find the terminal state. In the test, three goals are used in the evaluation of the weights, which are shown in Table 1. The weights that result in the LLP has the same terminal state as the policy given by the Cartesian ovals are selected. In the test, the invader is initialized at $(5, 5)$, and the defender is initialized at $(30, 30)$. With these initial locations, the defender can capture the invader for all the goals in Table 1. Thus, we can compare the performance by the location where capture has occurred. The optimal capture point is calculated for each of three speed scenarios and each of the three goals, separately. For each case, the LLP must have the same performance as the optimal policy. Figure 4 depicts the performance of the LLP and compares it to the optimal policy. The weights to obtain the results were $W_I = 0.65$ and $W_D = 0.50$. It is shown that by a proper set of weights, the FLC can have same performance as the policy given by the Apollonius circle.

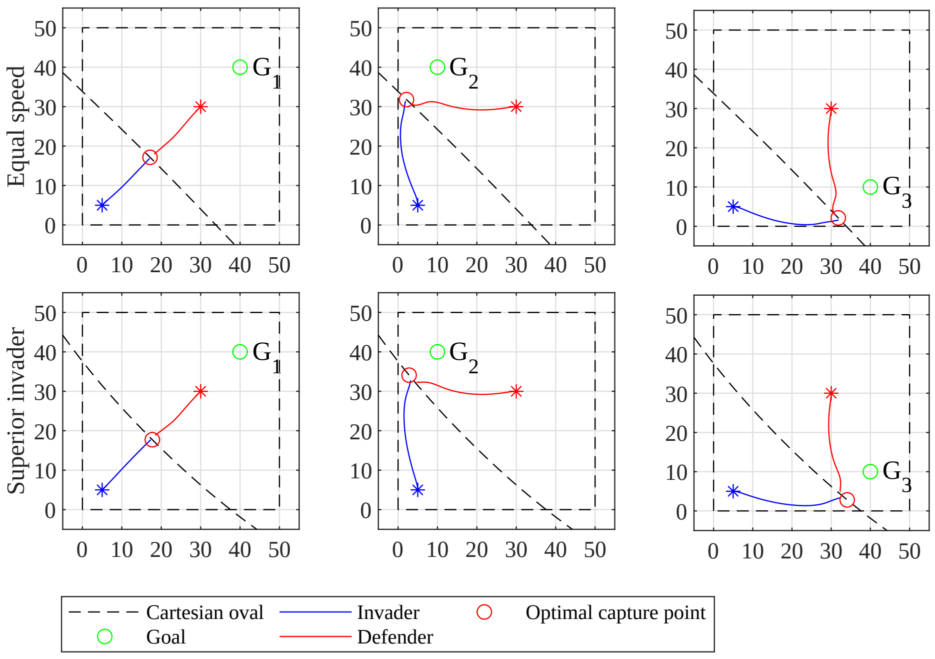

FIGURE 4. Optimal terminal states in the game and the LLP performance ($W_I = 0.65$ and $W_D = 0.50$)

Figure 4 shows that in all cases, the capture point of playing the game by using the derived LLP is close to the capture point given by optimal policy. This means that the FACL is able to learn the optimal policy, which validates the results in [1].

5.3. **Optimization of invader's HLP.** The invader pretends to choose different goals during the simulation. Therefore, it can manipulate the defender's belief function in (13). An FIS acts as the invader's HLP to select the goal. However, the defender uses a hard-coded function as its HLP or its belief function.

In order to find the output parameters of the invader's HLP, an integer NSGA-II algorithm is used. In the beginning, a set of chromosomes, which represents the output parameters of the FIS rule-base, is created randomly. The genetic algorithm tries to minimize the two objective functions in (14) by optimizing the output parameters. The invader's initial location is $(5, 5)$, and the initial orientation is $\pi/4$ radians. The defender's initial location is $(30, 30)$, and the initial orientation is $-3\pi/4$ radians.

A unique invader's HLP is needed to invade each goal under different speed scenarios. For instance, the invader's HLP to invade G_1 under the superior defender scenario differs from the invader's HLP to invade G_3 under the inferior defender scenario. In this paper, we are comparing three speed scenarios and three goals. Thus, we have a total number of nine different invader's HLPs, which must be obtained by using the NSGA-II algorithm. The results and tests are presented in the following section.

5.4. **Results and discussion.** We describe a non-deceptive situation. In a non-deceptive game, the invader knows its true goal. It chooses the true goal as its primary goal and does not pretend to change it over the time. For instance, if the true goal is G_2, the term β_I in the invader's LLP is only calculated for the second goal. On the other hand, the defender does not know the true goal, and it has to guess it via (13). The performance of the non-deceptive cases is depicted in Figures 5 and 6. It is shown that in all cases, the invader is not able to reach the target. In the following of this section, first we present the optimization results. And then we present the performance results.

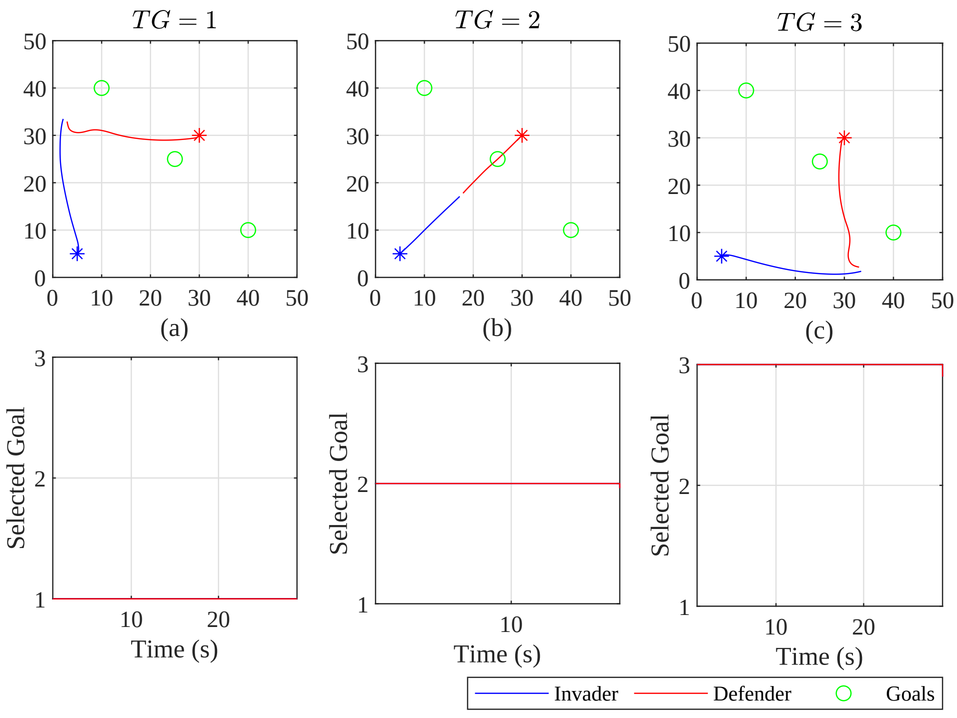

FIGURE 5. Non-deceptive case for the equal speed case

Figure 5 shows the performance of the non-deceptive case, where the agents' speed is equal to 1.00 unit/sec. It is observed that for all three goals the invader has been captured by the defender. Although in Figure 6 the invader is superior, it is captured by the defender.

5.4.1. *The invader's HLP optimization result.* In the deceptive case, the invader pretends to change its intention by using its HLP until it finds a proper state to capture the goal. However, there might be cases where winning a game is still impossible [34]. The term $Fitness_1$ in (14) is defined to minimize the number of times the invader pretends to change its goal, while $Fitness_2$ is defined to increase the defender's terminal distance to the invader (if the invader reaches the goal). If the defender captures the invader, then penalty functions are returned. This will have the effect of eliminating that chromosome from the Pareto front and the chromosome will not be selected in the next generations.

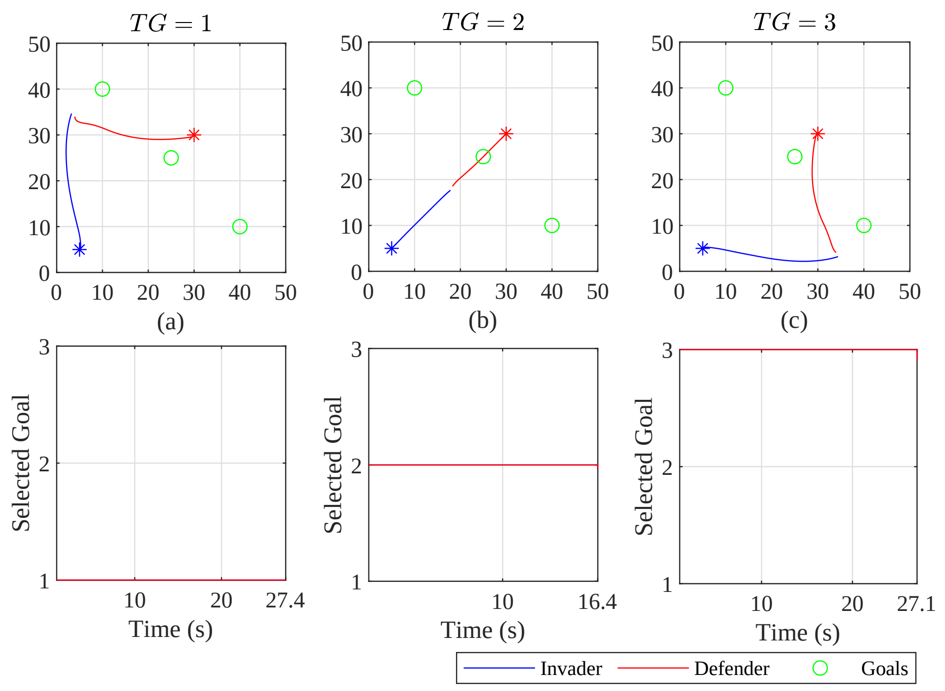

FIGURE 6. Non-deceptive case for the superior invader case

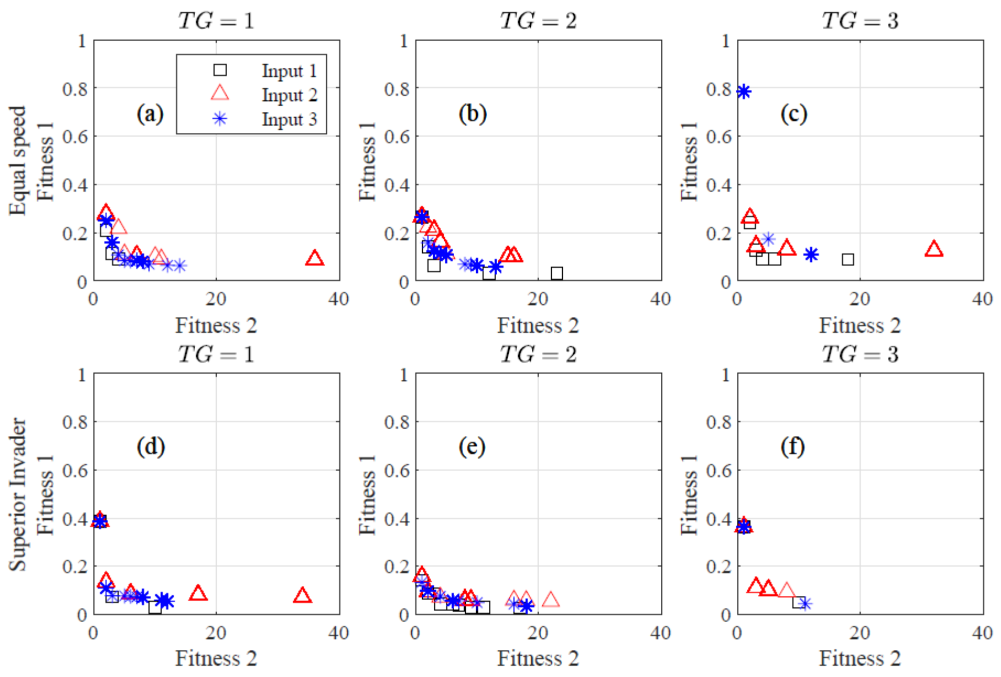

FIGURE 7. Pareto fronts for $Fitness_1$ and $Fitness_2$ in (14) in bi-objective optimization. The Pareto fronts are for the three invaders' HLP under two speed scenarios for the invasion of each goal.

Figure 7 depicts the optimization results in the form of Pareto fronts for $Fitness_1$ and $Fitness_2$. The figure shows the Pareto front at the final epoch.

Remark 5.1. *In order to keep the description tractable, we refer the phrase "the invader's HLP with Input n" as "Input n". For example, instead of writing "the invader's HLP with Input 3", we write "Input 3".*

The Pareto fronts for $Fitness_1$ and $Fitness_2$ in (14) are shown in Figure 7. The fitness function values in the figure are calculated in the final epoch of the invader's HLP optimization. Each row is dedicated to a speed scenario, while each column shows the results for the invasion of each goal. In all cases (a)-(f), the solutions for Input 1 dominate all other solutions. In addition, in all cases except (c), the solutions for Input 3 dominate the solutions for Input 2. However, to compare the result with numerical values, we brought the hypervolume parameter of each case in Table 2. Hypervolume indicates the quality of the Pareto front. The reference point for calculating the hypervolume is $(40, 1)$. Table 2 shows that in the equal case, Input 1 has almost the highest hypervolume. In other words, the Input 1 gives the best quality in the results. Input 3 returns the second best results in most of the cases and Input 2 stands in the last. Probably, Input 3 has a better quality in the results, since it has some information from the defender. However, Input 1 gives the best information about the agents location in the game. In the superior invader case, the same result is repeated. Input 1 has the best result, while Input 2 has the worst result.

Table 2. The hypervolume of the Pareto front of each input and goal

	Equal speed			Superior invader		
	Input 1	Input 2	Input 3	Input 1	Input 2	Input 3
G_1	0.88	0.856	0.894	0.922	0.891	0.909
G_2	0.935	0.858	0.911	0.937	0.904	0.929
G_3	0.857	0.833	0.766	0.87	0.887	0.842

5.4.2. *The invader's HLP testing result.* Each member on the Pareto fronts of Figure 7 represents a set of output parameters. These members are non-dominated. Thus, all of the solutions on the Pareto fronts can be selected. However, a designer can select one of the non-dominated solutions based on other aspects of the design. We prefer to choose the solutions that have relatively smaller $Fitness_2$. This selection helps the invader to successfully invade the target with less changes in the HLP. In addition, to have a fair comparison between different types of HLPs, we choose the Pareto front members that have roughly the same $Fitness_2$. Table 3 shows the fitness functions of the selected solutions, from the Pareto fronts in Figure 7. As shown in Table 3, in each speed scenario and goal case (the columns), the selected output parameters have almost the same $Fitness_2$, which are among the smallest ones. The output parameters corresponding to these fitnesses will be used in further simulations.

For the results in Table 3, we freeze the value of fitness 2. We tried to select solutions, close to the trade-off point. It is shown that for equal numbers of fitness 2, Input 1 has the best result. Input 3 stands on the second stage, while Input 2 is the worst.

Table 3. Selected results from the Pareto front

		G_1		G_2		G_3	
		Fitness 1	Fitness 2	Fitness 1	Fitness 2	Fitness 1	Fitness 2
Equal speed	Input 1	4	0.091	3	0.062	3	0.127
	Input 2	4	0.217	4	0.163	3	0.144
	Input 3	4	0.108	3	0.124	5	0.173
Superior invader	Input 1	3	0.073	4	0.043	1	0.363
	Input 2	6	0.086	4	0.072	3	0.112
	Input 3	3	0.077	4	0.754	1	0.363

Figure 8 shows the game, where the invader's HLP uses Input 1 and the agents' speeds are equal. We describe case (b). The invader chooses G_3 in the beginning and keeps it for 23.9 seconds. The defender changes its belief to G_1 at $t = 25.2$ seconds. The defender changes its belief at $t = 28.7$ seconds to G_2. However, after less than two seconds, the invader chooses G_1. The defender chooses G_1 after less than a second. The invader changes its goal to G_2 at $t = 49.3$ seconds and keeps it until the end. The defender mistakenly chooses G_1 at $t = 45.4$ seconds. The defender corrects its belief at $t = 62.8$ seconds by changing its goal to G_2.

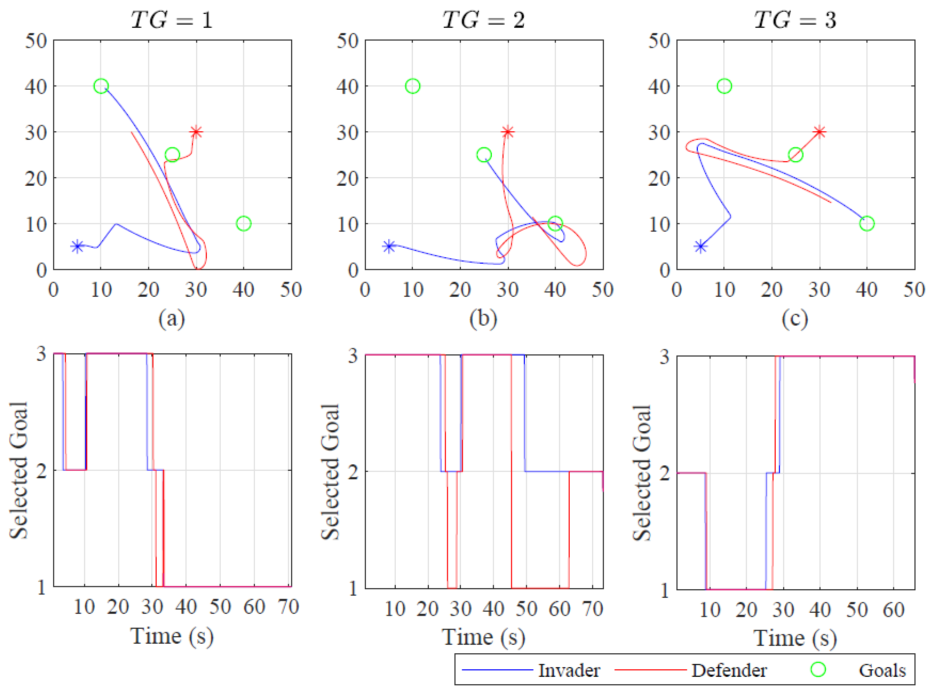

FIGURE 8. Deceptive case, where the speeds are equal and the invader's HLP uses Input 1

Figure 9 shows the game, where the invader's HLP uses Input 2 and the agents' speeds are equal. We describe case (a). The invader and the defender choose G_2 in the beginning. The invader changes its goal to G_1 at $t = 8.3$ seconds. The defender finds out the invader's goal after less than a second. The invader chooses G_2 at $t = 24.4$ seconds and keeps it for three seconds. The defender chooses G_2 at $t = 26$ seconds and G_3 at $t = 26.5$ seconds. After 1 second, the defender chooses G_1. The game finishes with a successful invasion at $t = 40.1$ seconds.

Figure 10 shows the game, where the invader's HLP uses Input 3 and the agents' speeds are equal. We describe case (c). The invader and the defender choose G_2 in the beginning. The invader chooses G_1 at $t = 11.7$ seconds. The defender chooses the G_1 at $t = 12$ seconds. The invader chooses G_2 at $t = 23$ seconds. In this time, the defender starts to switch between 2 and 3. At $t = 31.5$ seconds, both agents take G_3 as the chosen goal. However, it is too late for the defender to capture the invader.

Figure 11 shows the game, where the invader's HLP uses Input 1 and the invader is superior. We describe case (c). The agents take G_2 in the beginning. At $t = 15.4$ seconds, the invader chooses G_3. In 0.4 of a second, the defender finds out the goal change. However, it is too late for capturing the invader. The game finishes at $t = 36.7$ seconds with a successful invasion.

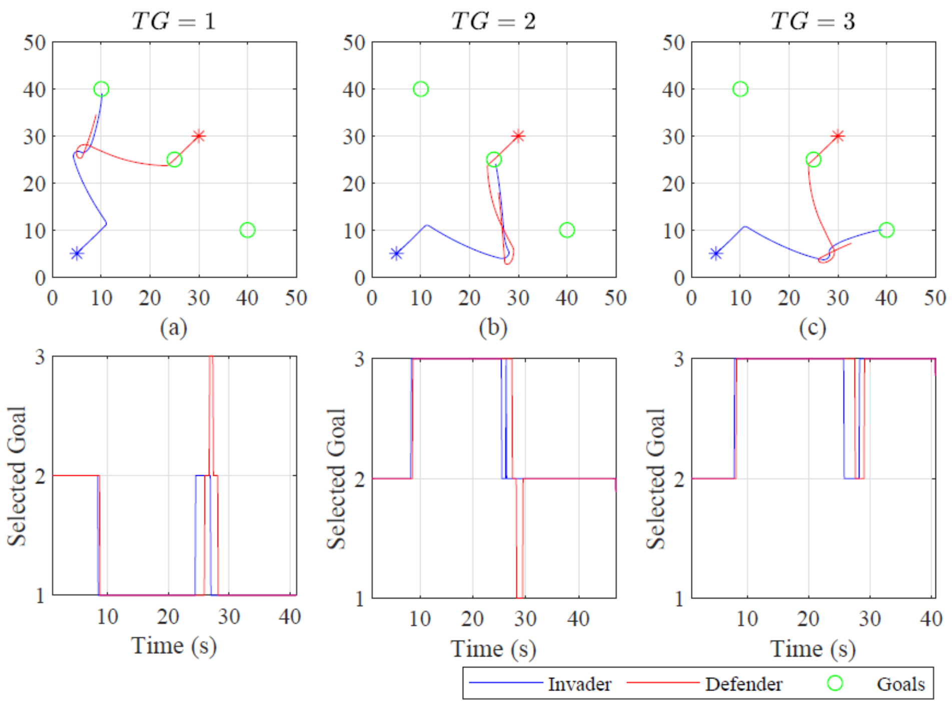

FIGURE 9. Deceptive case, where the speeds are equal and the invader's HLP uses Input 2

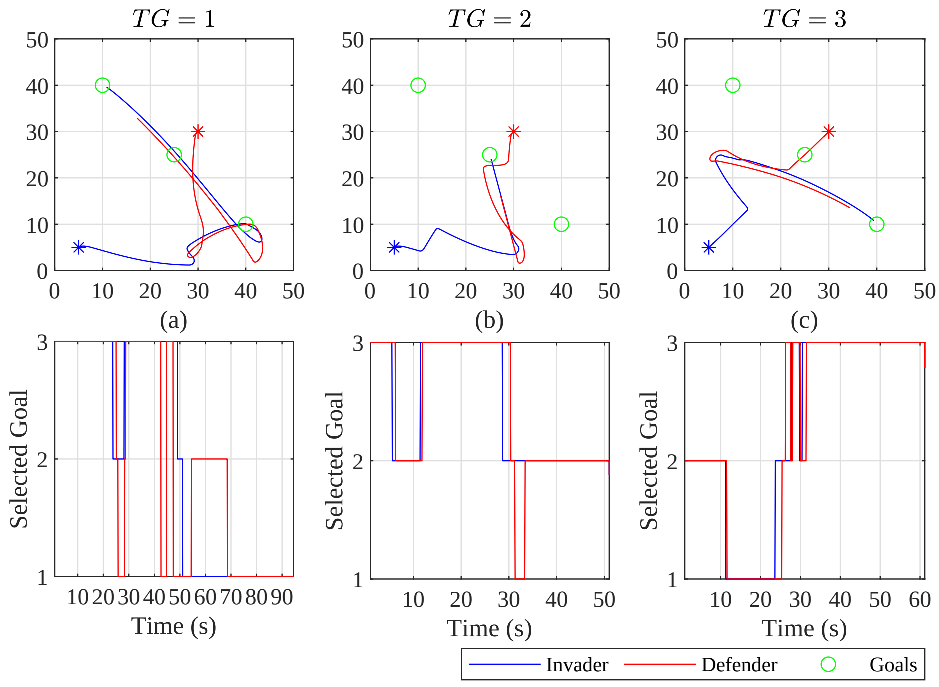

FIGURE 10. Deceptive case, where the speeds are equal and the invader's HLP uses Input 3

Figure 12 shows the game, where the invader's HLP uses Input 2 and the invader is superior. We describe case (b). The invader and the defender choose G_2 in the beginning. At $t = 6.7$ seconds the invader chooses G_3. In less than a second, the defender finds out the invader's decision. At $t = 25.9$ seconds, the invader chooses G_2, and after 2 seconds, the invader chooses G_1. Finally, the invader chooses G_2 at $t = 35.4$ seconds. The defender

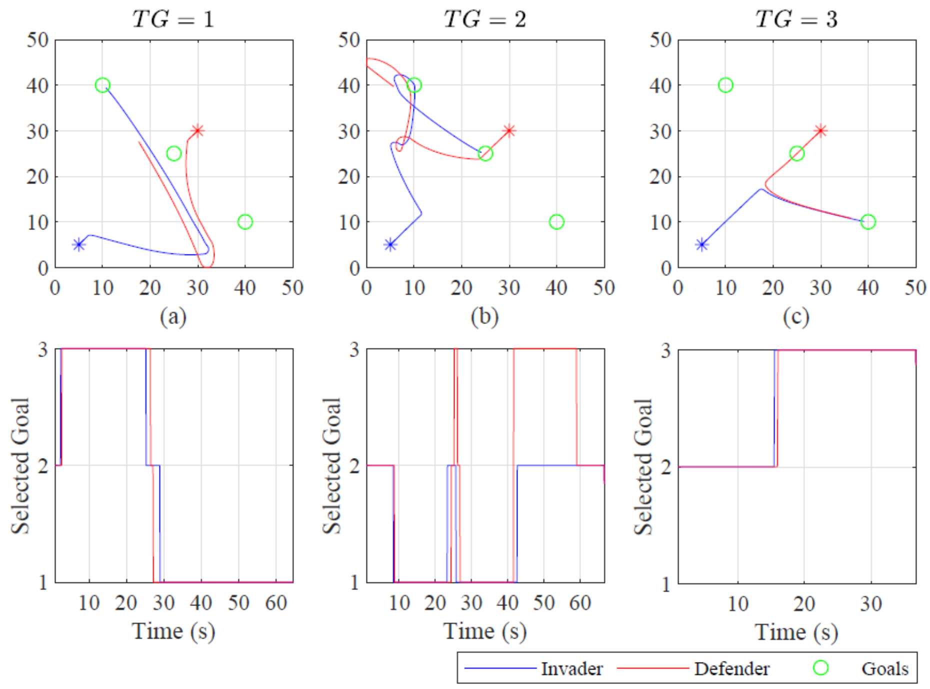

FIGURE 11. Deceptive case, where the invader is superior and the invader's HLP uses Input 1

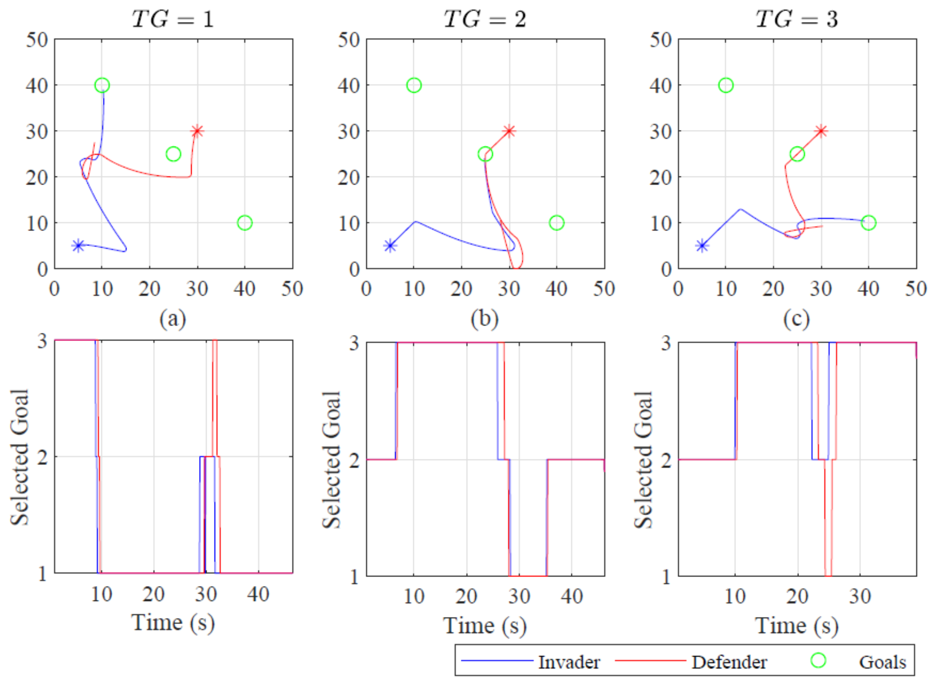

FIGURE 12. Deceptive case, where the invader is superior and the invader's HLP uses Input 2

chooses G_1 at $t = 28$ seconds. The defender's HLP returns G_2 at 35.6 seconds. The game finishes at 46.1 seconds with a successful invasion.

Figure 13 shows the game, where the invader's HLP uses Input 3 and the invader is superior. We describe case (c). The agents take G_2 in the beginning. At $t = 15.4$ seconds, the invader chooses G_3. In 0.2 of a second, the defender finds out the goal change. However,

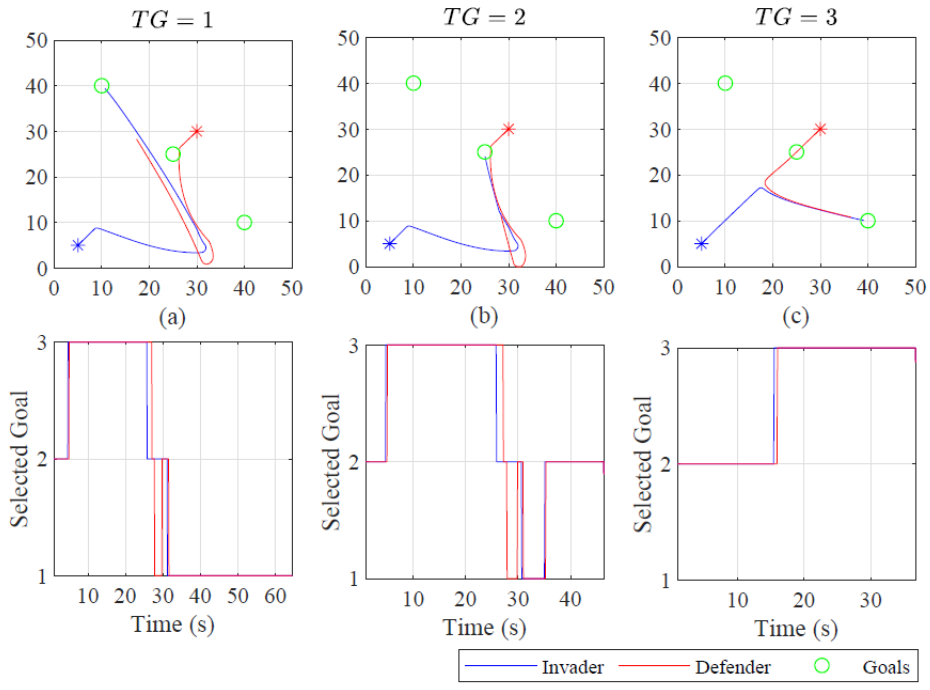

FIGURE 13. Deceptive case, where the invader is superior and the invader's HLP uses Input 3

it is too late for capturing the invader. The game finishes at $t = 36.7$ seconds with a successful invasion. This solution is very similar to the one with Input 1.

6. **Conclusion.** In this paper, an application of deception in a multi-agent game is presented. A two-level policy strategy is defined to model deceptive behaviour in a multi-agent two-player competitive game. A modified version of the game of guarding a territory is used as the simulation testbed. In the modified version, there are several targets instead of one, which gives a suitable background for studying deception. In the game, the invader acts deceptively by changing its goal consistently. Thus, the invader can conceal its true intention and deceive the defender. The results show a significant improvement in the invader's pay-off when it uses deception in comparison to the non-deceptive mode. While in the non-deceptive mode, the invader fails to reach the targets in all nine cases. In most deceptive cases, the invader wins the game by a successful invasion. Although the proposed deceptive strategy was not initially designed to address the curse of dimensionality, it has a significant effect on the size of the rule base. A system with four LLP inputs and four HLP inputs deals with two sets of 625 rules. At the same time, modelling deception within a single-level eight input policy needs 390,625 rules.

REFERENCES

[1] C. V. Analikwu and H. M. Schwartz, Multi-agent learning in the game of guarding a territory, *International Journal of Innovative Computing, Information and Control*, vol.13, no.6, pp.1855-1872, 2017.

[2] A. Asgharnia, H. M. Schwartz and M. Atia, Deception in the game of guarding multiple territories: A machine learning approach, *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp.381-388, 2020.

[3] A. Asgharnia, H. M. Schwartz and M. Atia, Deception in a multi-agent adversarial game: The game of guarding several territories, *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp.1321-1327, 2020.

[4] A. Ayub, A. Morales and A. Banerjee, An adaptive Markov process for robot deception, *arXiv Computer Science*, pp.1-10, 2019.

[5] R. P. Baksi and S. J. Upadhyaya, Decepticon: A hidden Markov model approach to counter advanced persistent threats, *Communications in Computer and Information Science*, vol.1186, pp.38-54, 2020.

[6] C. F. Bond and M. Robinson, The evolution of deception, *Journal of Nonverbal Behavior*, vol.12, no.4, pp.295-307, 1988.

[7] R. L. Boyell, Defending a moving target against missile or torpedo attack, *IEEE Transactions on Aerospace and Electronic Systems*, vol.AES-12, no.4, pp.522-526, 1976.

[8] M. Chen, Z. Zhou and C. J. Tomlin, Multiplayer reach-avoid games via pairwise outcomes, *IEEE Transactions on Automatic Control*, vol.62, no.3, pp.1451-1457, 2016.

[9] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[10] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, vol.6, no.2, pp.182-197, 2002.

[11] S. F. Desouky and H. M. Schwartz, Self-learning fuzzy logic controllers for pursuit-evasion differential games, *Robotics and Autonomous Systems*, vol.59, no.1, pp.22-33, 2011.

[12] A. Dragan, R. Holladay and S. Srinivasa, Deceptive robot motion: Synthesis, analysis and experiments, *Autonomous Robots*, vol.39, no.3, pp.331-345, 2015.

[13] D. Ettinger and P. Jehiel, A theory of deception, *American Economic Journal: Microeconomics*, vol.2, no.1, pp.1-20, 2010.

[14] E. Garcia, Cooperative target protection from a superior attacker, *Automatica*, vol.131, https://doi.org/10.1016/j.automatica.2021.109696, 2021.

[15] E. Garcia, D. W. Casbeer and M. Pachter, Active target defence differential game: Fast defender case, *IET Control Theory and Applications*, vol.11, no.17, pp.2985-2993, 2017.

[16] E. Garcia, D. W. Casbeer and M. Pachter, The complete differential game of active target defense, *arXiv*, https://doi.org/10.1007/s10957-021-01816-z, 2020.

[17] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, Courier Corporation, 1999.

[18] P. E. Johnson, S. Grazioli, K. Jamal and R. G. Berryman, Detecting deception: Adversarial problem solving in a low base-rate world, *Cognitive Science*, vol.25, no.3, pp.355-392, 2001.

[19] M. Kouzehgar and M. A. Badamchizadeh, Fuzzy signaling game of deception between ant-inspired deceptive robots with interactive learning, *Applied Soft Computing Journal*, vol.75, pp.373-387, 2019.

[20] M. Kouzehgar, M. Badamchizadeh and M. R. Feizi-Derakhshi, Ant-inspired fuzzily deceptive robots, *IEEE Transactions on Fuzzy Systems*, vol.24, no.2, pp.374-387, 2016.

[21] L. Liang, F. Deng, Z. Peng, X. Li and W. Zha, A differential game for cooperative target defense, *Automatica*, vol.102, pp.58-71, 2019.

[22] X. Lu and H. M. Schwartz, An investigation of guarding a territory problem in a grid world, *Proc. of the 2010 American Control Conference (ACC2010)*, pp.3204-3210, 2010.

[23] S. Mannor and N. Shimkin, The steering approach for multi-criteria reinforcement learning, *Advances in Neural Information Processing Systems*, 2002.

[24] W. McEnenaey and R. Singh, Deception in autonomous vehicle decision making in an adversarial environment, *Collection of Technical Papers – AIAA Guidance, Navigation, and Control Conference*, vol.4, pp.3032-3043, 2005.

[25] M. Ornik and U. Topcu, Deception in optimal control, *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton2018)*, pp.821-828, 2019.

[26] M. V. Ramana and M. Kothari, Pursuit-evasion games of high speed evader, *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol.85, no.2, pp.293-306, http://dx.doi.org/10.1007/s10846-016-0379-3, 2017.

[27] H. Raslan and H. Schwartz, A learning invader for the "guarding a territory" game – A reinforcement learning problem, *Journal of Intelligent & Robotic Systems*, pp.55-70, 2016.

[28] E. Santos and D. Li, On deception detection in multiagent systems, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol.40, no.2, pp.224-235, 2010.

[29] E. Saquete, D. Tomás, P. Moreda, P. Martínez-Barco and M. Palomar, Fighting post-truth using natural language processing: A review and open challenges, *Expert Systems with Applications*, vol.141, 2020.

[30] H. M. Schwartz, *Multi-Agent Machine Learning: A Reinforcement Approach*, John Wiley and Sons, 2014.

[31] J. Shim and R. C. Arkin, A taxonomy of robot deception and its benefits in HRI, *Proc. of 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC2013)*, pp.2328-2335, 2013.

[32] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Transactions on Systems, Man and Cybernetics*, vol.SMC-15, no.1, pp.116-132, 1985.
[33] K. Van Moffaert and A. Nowé, Multi-objective reinforcement learning using sets of Pareto dominating policies, *The Journal of Machine Learning Research*, vol.15, no.1, pp.3483-3512, 2014.
[34] A. R. Wagner and R. C. Arkin, Robot deception: Recognizing when a robot should deceive, *Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp.46-54, 2009.
[35] J. Wang and W. Li, Motion patterns and phase-transition of a defender-intruder problem and optimal interception strategy of the defender, *Communications in Nonlinear Science and Numerical Simulation*, vol.27, nos.1-3, pp.294-301, 2015.
[36] A. Wilson, H. Armanto, Gunawan and J. Tanwijaya, Focused web crawler using genetic algorithms and symbiotic organism search, *ICIC Express Letters*, vol.15, no.12, pp.1345-1354, 2021.

Author Biography

Amirhossein Asgharnia joined the Department of Systems and Computer Engineering at Carleton University as a Ph.D. student in Fall 2019. He received his B.Sc. and M.Sc. degrees in Mechanical Engineering from the University of Guilan, Iran in 2015 and 2018, respectively. His research focuses on reinforcement learning and multiagent systems.

Howard Schwartz received the B.Eng. degree in Civil Engineering from McGill University, Montreal, QC, Canada in 1981, and the M.S. in Aeronautics and Astronautics in 1982 and the Ph.D. degree in Mechanical Engineering in 1987 from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His most recent research is in multi-agent learning with applications to teams of mobile robots.

Mohamed Atia is an Assistant Professor in the Department of Systems and Computer Engineering of Carleton University. He received his B.Sc. and M.Sc. degrees in Computer Systems from Ain Shams University in 2000 and 2006 and a Ph.D. in Electrical and Computer Engineering from Queen's University at Kingston in 2013. His research interests include real-time embedded systems, sensor-fusion, signal processing, estimation, machine learning, artificial intelligence, multi-sensors integrated navigation, guidance, and control, collaborative navigation, wireless positioning, global navigation satellite systems (GNSS), Inertial Sensors (INS), simultaneous localization and mapping, attitude and heading reference systems (AHRS), vision/radar/liDAR-aided navigation, localization and mapping.