Self-Confirming Transformer for Belief-Conditioned Adaptation in Offline Multi-Agent Reinforcement Learning

Anonymous Author(s)
Submission Id: 15

ABSTRACT

Offline reinforcement learning (RL) suffers from the distribution shift between the offline dataset and the online environment. In multi-agent RL (MARL), this distribution shift may arise from the nonstationary opponents in the online testing who display distinct behaviors from those recorded in the offline dataset. Hence, the key to the broader deployment of offline MARL is the online adaptation to nonstationary opponents. Recent advances in foundation models, e.g., large language models, have demonstrated the generalization ability of the transformer, an emerging neural network architecture, in sequence modeling, of which offline RL is a special case. One naturally wonders *whether offline-trained transformer-based RL policies adapt to nonstationary opponents online*. We propose a novel auto-regressive training to equip transformer agents with online adaptability based on the idea of self-augmented pre-conditioning. The transformer agent first learns offline to predict the opponent's action based on past observations. When deployed online, such a fictitious opponent play, referred to as the belief, is fed back to the transformer, together with other environmental feedback, to generate future actions conditional on the belief. Motivated by self-confirming equilibrium in game theory, the training loss consists of belief consistency loss, requiring the beliefs to match the opponent's actual actions and best response loss, mandating the agent to behave optimally under the belief. We evaluate the online adaptability of the proposed self-confirming transformer (SCT) in a structured environment, iterated prisoner's dilemma games, to demonstrate SCT's belief consistency and equilibrium behaviors as well as more involved multi-particle environments to showcase its superior performance against nonstationary opponents over prior transformers and offline MARL baselines.

KEYWORDS

Offline Reinforcement Learning, Multi-Agent Reinforcement Learning, Transformer, Online Adaptation, Self-Confirming Equilibrium

1 INTRODUCTION

Offline reinforcement learning (RL) has recently emerged as a promising alternative to online RL [22], which extracts policies purely from the previously collected dataset without any interaction with the environment. As such, offline RL avoids online explorations required by online RL algorithms, which can be expensive (e.g., end-to-end robotic control [21]), dangerous (e.g., self-driving [26]), and sometimes infeasible (e.g., healthcare [59]).

Yet, a fundamental challenge of offline RL is the distribution shift between the offline training dataset and the online testing environment [22]. In plain words, the offline RL agent needs to properly handle unseen state-action pairs in the dataset during testing [3]. When extending this offline RL to multi-agent RL (MARL) settings, the distribution shift may be caused by exogenous agents who are beyond the preview of the trained MARL policy. We refer to these exogenous agents as the opponents. When opponents display a behavior pattern different from those included in the offline dataset, the ego agents are unprepared for these unseen state-action pairs resulting from opponents' unexpected moves. We refer to such an opponent as nonstationary, as it employs a different and possibly time-varying policy in testing, as opposed to the stationary policy used to collect offline data. As shown in one motivating example in Figure 3, blindly applying offline MARL policy gives degrading performance when playing with a nonstationary opponent.

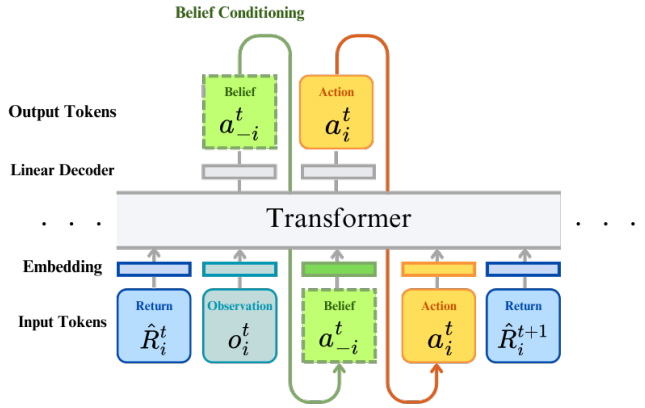


Figure 1: Self-augmented belief conditioning in the self-confirming transformer (SCT). SCT first generates a belief on the opponent's action a^t_{-i} (the green block), which is a fictitious token unobserved from the environment. Based on this belief, the transformer generates the action.

While the study of developing autonomous agents capable of reasoning and adapting to unknown opponent policies, referred to as *opponent modeling* (OM), is a long-standing and pivotal research topic in multi-agent systems and artificial intelligence [2], this works explores the transformer architecture [52] to tackle opponent modeling and offline policy extraction combined as an integrated sequence modeling problem. The question we ask is

whether the offline transformer policy can adapt to the nonstationary opponent online by modeling its behavior auto-regressively?

This work answers this question affirmatively by introducing the self-confirming transformer (SCT) that learns to predict the opponent's move from the partial observation, which is then fed to the transformer itself to generate the ego agent's action, as depicted in Figure 1. The SCT is inspired by the self-confirming equilibrium (SCE) [9], a weaker variant of the seminal Nash equilibrium (NE) [37], which does not assume every agent's compliance with NE policy. In contrast, SCE rests on subjective rationality, focusing on consistency between one's observations and subjective beliefs on the opponent's future move, which leads to the term

Proc. of the Adaptive and Learning Agents Workshop (ALA 2025), Avalos, Aydeniz, Müller, Mohammedalamen (eds.), May 19 – 20, 2025, Detroit, Michigan, USA, ala-workshop.github.io. 2025.

"self-confirming" (see Definition 4.1). From a sequence modeling perspective, the sheer difference between our SCT and previous transformers lies in that SCT creates a fictitious belief token as part of the input sequence, serving as the pre-conditioning for action generation, while prior works only use environment feedback (e.g., observations and rewards) to generate control actions. The additional belief generation in SCT, bearing a similar spirit to OM, prevents overfitting the offline data and helps the agent to adapt its action online. A summary of our contributions is as follows.

- We propose a novel auto-regressive self-confirming training, enabling the transformer to infer and adapt to the opponent's nonstationary policy without modifying its architecture.
- We conduct extensive experiments in benchmark MARL environments to demonstrate SCT's greater adaptability over offline MARL baselines and recent transformer models.
- We analyze SCT's equilibrium behavior in a structured iterated prisoner's dilemma, empirically certifying its self-confirming plays as instructed by self-confirming training.

2 RELATED WORKS

Offline Reinforcement Learning. Offline RL methods extract policies from pre-collected datasets without interacting with the environment. These methods can be categorized into constraint-based and sequential model-based approaches [36]. In constraint-based methods, off-policy algorithms treat offline datasets as a replay buffer to learn a policy with promising performance. However, the experiences in offline datasets and interactions with online environments have different distributions, i.e., there is a *distribution shift*, causing overestimation in policy and value-based methods. To address this issue, recent progress utilizes the conservatism idea [22] that compels the policy [10] or value function estimation [11, 20] to the data manifold to control the extrapolation error.

Transformer in RL. In addition to the constraint-based approach above, sequential model-based methods, which treat the offline policy training process as a sequence modeling problem, have also emerged as a powerful tool. Sequence modeling, motivated by the analysis of sequential data such as texts and time series, is concerned with learning the correlation among sequential data and forecasting or generating future data points, which has been the core research topic in natural language processing [50], speech recognition [7], and time-series prediction [23, 24]. Since RL trajectories also display temporal correlations, sequence modeling methods aim to predict future states, actions, and rewards using past observations.

Due to transformers' encouraging success in sequence modeling [52], recent efforts have been exploring the application of transformers in offline RL. One example of such is the decision transformer (DT) and its variants [4, 12]. DT learns the distribution of trajectories and predicts actions conditional on target rewards and previous observations. Instead of directly predicting the optimal actions, the trajectory transformer and its variants [18, 54] use the transformer to roll out future trajectories and search for optimal policies. As for multi-agent RL, the current multi-agent transformer research mainly employs the transformer architecture for representation learning [14, 31] under long horizons and partial observations, i.e., the transformer, learning the hidden representation of historical observations, becomes part of a bigger RL machinery for policy optimization [36, 56]. In contrast, our proposed transformer subscribes to DT and addresses the action generation conditional on the additional fictitious belief tokens. Another remark is that while those prior works on multi-agent transformers concentrate on cooperative tasks, a central transformer policy controls all agents; our approach alludes to independent learning [5, 30] where each agent handles the nonstationary opponents in noncooperative settings.

Opponent Modeling. Our work is also closely related to transformer-based opponent modeling. The intuition is that modeling the opponent's behavior is also a sequence modeling problem, where the transformer learns to reconstruct opponent policies using offline data. Some early works utilize encode-decoder architecture (which is used by the transformer as well) to extract the temporal correlation between states and opponent actions [14, 46]. Most relevant to our work are [19, 53], where authors employ the transformer architecture to model the opponent's policy and forecast its actions for policy updates. These works treat the transformer as a purely predictive model for opponent action forecasting and call for additional mechanisms for policy learning. In contrast, our proposed SCT combines opponent action forecasting with policy generation using the vanilla decision transformer model [4]. We aim to investigate whether belief conditioning creates greater online adaptability when generating actions auto-regressively.

3 PRELIMINARY

Multi-Agent Reinforcement Learning. Consider learning in a multi-agent decision process described by a partially observable Markov game (POMG). A POMG with N agents indexed by $i \in \{1, 2, \ldots, N\} := [N]$ includes a global state space $\mathcal{S}$, each agent's action space $\mathcal{A}_i$, and a set of observations O_i for each individual. The typical elements of these spaces are denoted by the corresponding uncapitalized letters. The time step is denoted by $t \in \mathbb{N}_+$, appearing as the superscript in the sequel. Unaware of the global state s^t, each agent receives a local observation $o_i^t \in O_i$ and chooses an action a_i^t. Denote by the bold symbol $\boldsymbol{o}^t = \{o_1^t, o_2^t, \ldots, o_N^t\}$ the joint observation. Then, with the joint actions of all agents, denoted by the bold symbol $\boldsymbol{a}^t = (a_1^t, a_2^t, \ldots, a_N^t)$, the environment transits to the next state s^{t+1} according to the transition kernel $\mathcal{P} : \mathcal{S} \times \prod_{i\in[N]} \mathcal{A}_i \to \Delta(\mathcal{S})$, and the decision-making process repeat. We assume all involved sets in this work are Borel sets (either discrete or continuous), and $\Delta(\cdot)$ denotes the Borel probability measure, e.g., $\mathcal{P}(s^{t+1}|s^t, \boldsymbol{a}^t)$ give the distribution of the next state.

Agents' performance is evaluated through the reward function $r_i : \mathcal{S} \times \prod_{i\in[N]} \mathcal{A}_i \to \mathbb{R}$, and each agent aims to maximize its own discounted return $\sum_{t=1}^{T} \gamma^{t-1} r_i^t$, where $r_i^t = r_i(s^t, \boldsymbol{a}^t)$, and T denotes the horizon length. We consider the conventional decentralized information structure [30], consisting of local feedback: $\mathcal{I}_i^t = \{o_i^t, a_i^{t-1}, r_i^{t-1}\}$ in a non-cooperative multi-agent environment, including competitive and mixed cooperative-competitive scenarios [34], where agents may have distinct reward signals, i.e., $r_i \neq r_j$ for some $i, j \in [N]$. Each agent aims to find a policy $\pi_i \in \Pi_i$ that maps the past information to some action at each time step: $a_i^t \sim \pi_i(\cdot|\mathcal{I}_i^{1:t})$ to maximize the discounted return, where π_i is assumed to be a stochastic policy yielding a distribution on $\mathcal{A}_i$. We use π_i and its neural network parameterization θ_i interchangeably

to refer to the agent's policy. In general, agents' information structures are different, leading to the challenge of independent learning under asymmetric information and recursive reasoning [15, 25]. This work explores the use of the transformer in sequence modeling to address independent learning under asymmetric observability.

Transformer Architecture. Generally, a transformer consists of an encoder, an attention module, and a decoder, which can use either the encoder, the decoder, or both, depending on the applications. Decoder-only models are useful for generating sequence and forecasting tasks [48]. Encoder-only models are suitable for sequence understanding tasks [6]. We consider decoder-only architecture since the key of our SCT is to predict opponent actions. We now briefly review the attention module in the transformer model.

The raw inputs of the transformer (which we will call tokens) are initially embedded in vectors of dimension d_{model}. Each input embedding generates a query, key, and value vector of dimensions d_k, d_k, and d_v. Vectors of the same type are stacked column-wise to produce three matrices $Q \in \mathbb{R}^{l\times d_k}$, $K \in \mathbb{R}^{l\times d_k}$, and $V \in \mathbb{R}^{l\times d_v}$, with l the maximum context length (i.e., the length of the input sequence, defined as a hyperparameter). The attention score is then calculated with the formula

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^\intercal/\sqrt{d_k}\right)V.$$

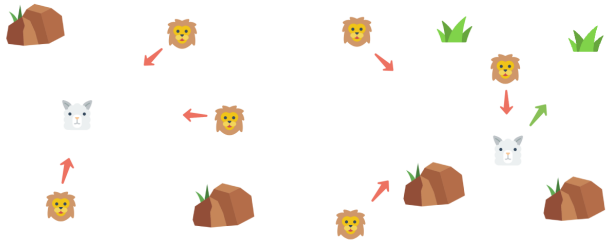
The matrix $QK^\intercal$ is divided by $\sqrt{d_k}$ to prevent the vanishing gradient problem when applying the softmax row-wise [52]. After the softmax computation, the upper off-diagonal triangle part of the resulting matrix $\text{softmax}(QK^\intercal/\sqrt{d_k})$ is masked with 0s. This causal mask prevents future tokens from influencing the prediction of the current target and is the defining feature of a causal transformer, which is applied in our experiments since the opponent action prediction can only leverage historical observations.

Offline MARL as Sequence Modeling. We take the multi-agent decision transformer (MADT) in [36] as an example to illustrate the sequence modeling. Consider a trajectory τ from the offline dataset given by $\tau = \{\boldsymbol{o}^1, \boldsymbol{a}^1, \boldsymbol{o}^2, \boldsymbol{a}^2, \ldots, \boldsymbol{o}^T, \boldsymbol{a}^T\}$. MADT, parameterized by a decoder-only transformer model q_θ, generates (predicts) sequential actions at each time step auto-regressively. Let $\hat{\tau}^t = \{\boldsymbol{o}^1, \hat{\boldsymbol{a}}^1, \ldots, \boldsymbol{o}^t, \hat{\boldsymbol{a}}^t\}$ be the truncated trajectory up to time t with previous action predictions. Then, MADT's sequential generation proceeds as follows: $\hat{\boldsymbol{a}}^t = \arg\max_{\boldsymbol{a}} q_\theta(\boldsymbol{a}|\hat{\tau}^{t-1}, \boldsymbol{o}^t)$. The learning objective of MADT is to minimize the distribution discrepancy between the generation q_θ and the offline data distribution q_{off}. Toward this end, one can consider the cross entropy (CE) loss to train MADT in discrete cases. Given predictions $\{\hat{\boldsymbol{a}}^t\}$, the CE loss is defined as $\mathcal{L}_{CE}(\theta) = 1/T \sum_{t=1}^{T} q_{\text{off}}(\hat{\boldsymbol{a}}^t) \log q_\theta(\hat{\boldsymbol{a}}^t|\hat{\tau}^{t-1}, \boldsymbol{o}^t)$. While for continuous control tasks, the mean-squared error $\|\hat{\boldsymbol{a}}^t - \boldsymbol{a}^t\|^2$ leads to decent transformer policies as observed in [4].

Note that RL is a sequential decision-making process where the current actions influence future states and rewards. To equip the agent with forward-looking ability, DT slightly modifies the trajectory representation and adds the reward-to-go (return) $\hat{R}_i^t = \sum_{k=t}^{T} r_i^k$. The resulting trajectory is $\tau = \{\hat{\boldsymbol{R}}^1, \boldsymbol{o}^1, \hat{\boldsymbol{a}}^1, \ldots, \hat{\boldsymbol{R}}^t, \boldsymbol{o}^t, \hat{\boldsymbol{a}}^t\}$, where the bold symbol $\hat{\boldsymbol{R}}^1$ denotes the agents' joint rewards, referred to as the return conditioning [4]. In plain words, such a return conditioning gives the transformer a sense of what to expect and directs its action generation to ensure that the cumulative rewards well approximate the return conditioning $\hat{\boldsymbol{R}}^1$. Such a practice is referred to as hindsight information matching (HIM), a popular technique in off-policy optimization [12].

4 SELF-CONFIRMING TRANSFORMER

Motivating Example. We consider the predator-prey task (a.k.a `simple-tag`) included in the multi-agent particle environment (MPE) [34], one of the benchmark environments in MARL. As shown in Figure 2a, the environment includes a prey who moves faster and aims to evade the three predators. The predators are slower and try to hit the prey while avoiding obstacles.



(a) `simple-tag`: the three slow-moving predators aim to catch the fast-moving prey while avoiding the obstacles.

(b) `simple-world`: a variant of `simple-tag` with two added food particles. The prey is rewarded when hitting the food.

Figure 2: The predator-prey tasks in multi-agent particle environment.

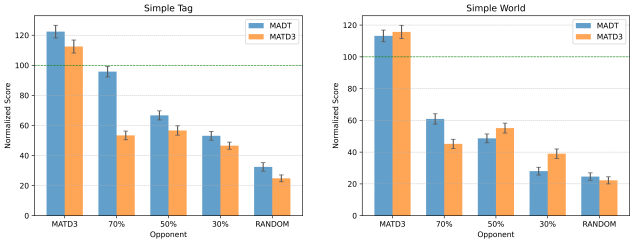


Figure 3: The normalized scores (the higher, the better) of playing MADT and MATD3 policy against the nonstationary opponent in `simple-tag` (left) and `simple-world` (right). The opponent employs a blend of MATD3 and the random policy, with the blending rate p shown on the x-axis. The green dashed line indicates the benchmark performance of the testing task.

The predators observe the relative positions and velocities of the prey, while the prey can only observe the relative positions of the other agents. All agents' actions are two-dimensional velocity vectors. Each time any one of the three predators collides with the prey, the former gets rewarded while the latter is penalized. The predator-prey is a mixed cooperative-competitive task, where the predators cooperate with each other to encircle the prey so that the rewards get tripled, while the game between the prey and predators is zero-sum like. Another environment we consider is `simple-world` shown in Figure 2b, a more complicated variant of `simple-tag` as it includes two food particles that prey is rewarded for being close to. More details are deferred to Section 5.

We here briefly touch upon the training and the testing procedure, while the detailed experiment setup is included in Section 5. We use MATD3 [1], one of the state-of-the-art MARL algorithms, to train the four agents (three predators and one prey) and collect expert-level trajectory data after MATD3 training stabilizes as the offline dataset. MADT is trained using offline trajectories of the

three predators. The prey employs the following baseline policies during the testing: 1) π^M, the same MATD3 policy used to collect the data; 2) π^R, the random policy; 3) π^B, a blend of the random and the MATD3 policy. The random policy takes a uniform distribution over the action set regardless of the observation input. The blending policy works like a bang-bang controller: at each time step, the prey flips a coin first; if heads up, then it chooses π^M, otherwise π^R. This blend can be written as $\pi_B = p \times \pi^M + (1-p) \times \pi^R$, where the parameter p is the mean value of the binomial distribution, capturing the opponent's nonstationarity.

The purpose of this example is to examine the MADT's online adaptability when facing a nonstationary opponent in testing. Since the opponent utilizes a policy distinct from that in training, the resulting trajectory deviates from the offline data. This adaptability concerns whether the MADT adjusts its action generation according to the changing trajectory distribution. The evaluation metric is the normalized score, a customary metric indicating the discounted returns [8]. Figure 3 reports the testing results, from which one can see that the MADT's performance gradually degrades as the opponent deviates from π^M.

Yet, one interesting phenomenon we observe is that the transformer-based policy does exhibit online adaptability compared with the pre-trained MARL policy, though to a limited extent. We equip the three predators with the MATD3 policies that are used in the data collection and let them play with the three baseline prey policies mentioned above. We denote the predators' MATD3 policies by π^M_{pred}. Note that π^M_{pred} includes three MATD3 policies and one for each predator. Figure 3 summarizes the testing results, from which one can see that π^M_{pred} gives even lower scores than the MADT does (orange bars). We believe this adaptability originates from the generalization ability of the transformer architecture, which is also observed in large language models [48, 57] and robotic transformers [49]. This motivating example prompts one to ask whether offline-trained transformers can adapt to unseen opponents online.

Self-Confirming Belief Conditioning. Reflecting on the return conditioning in MADT (see Section 3), one realizes that such conditioning does not provide the agent direct contextual information regarding the opponent since the rewards r_i^t, which affects subsequent $\hat{R}_i^t$, are jointly determined by all agents' actions. Using the language of HIM [12], the return conditioning fails to provide fine-grained *information statistics* regarding nonstationary opponents.

Taking inspiration from game-theoretic research on adaptive learning agents [29] and the self-confirming equilibrium [9], we propose to introduce an additional conditioning that represents the agent's subjective belief over the opponent's unobservable action. To articulate the intuition, we first digress from the transformer and briefly review the notion of self-confirming equilibrium (SCE). SCE was born from dissatisfaction with Nash equilibrium (NE) [37] since learning agents often display suboptimal and non-equilibrium behaviors [9, 44, 45]. As a relaxation to NE, SCE forgoes the *collective rationality* that requires every agent's compliance with the NE policy. Instead, SCE embodies *subjective rationality*, where the agent maximizes its rewards with respect to the beliefs over the opponent's policy, and the belief is consistent with observations.

Mathematically, denote by $\mathcal{I}_i^{1:t}$ the past observations up to time t. Similar to the agent's policy $\pi_i(\cdot|\mathcal{I}_i^{1:t}) \in \Delta(\mathcal{A}_i)$, the agent's belief is a distribution over the opponent's action set: $\mu_i(\cdot|\mathcal{I}_i^{1:t}) \in \Delta(\mathcal{A}_{-i})$. The definition of SCE in Markov games, adapted from its original version in extensive-form games [9], is as below.

Definition 4.1 (Self-Confirming Equilibrium). A strategy profile (π_i, π_{-i}) is a self-confirming equilibrium of the partially observable Markov game if, for each agent, there exists a belief μ_i such that

$$\pi_i \in \arg\max_{\hat{\pi}_i \in \Pi_i} \mathbb{E}_{\hat{\pi}_i, \mu_i, \mathcal{P}} \left[\sum_{t=1}^{T} \gamma^{t-1} r_i(s^t, a_i^t, a_{-i}^t) \right], \tag{1}$$

where the belief μ_i is consistent with the opponent's equilibrium policy π_{-i} with respect to all realizable information feedback in total variation, i.e., for all $\mathcal{I}_i^{1:t}$, $\mathcal{I}_{-i}^{1:t}$, $\mathbb{P}_{\pi_i, \pi_{-i}}[\mathcal{I}_i^{1:t}, \mathcal{I}_{-i}^{1:t}] > 0, t = 1, \ldots, T$,

$$\sup_{A \subset \mathcal{A}_{-i}} \left| \mu_i(A|\mathcal{I}_i^{1:t}) - \pi_{-i}(A|\mathcal{I}_{-i}^{1:t}) \right| = 0. \tag{2}$$

Some remarks are in order. First, NE is a special case of SCE if the belief consistency in (2) is imposed on every information structure [9], i.e., $\mu_i = \pi_{-i}$. Unlike NE, SCE does not mandate every agent to follow the optimal policy. It is likely that the opponent may employ arbitrary policies; as long as the ego agent can correctly identify these behavior patterns and best responds to the consistent belief, it still arrives at SCE. Second, the current practice of collecting offline datasets typically begins with centralized training of benchmark MARL algorithms, such as MADDPG [34] and MATD3 [1], to learn optimal policies for each agent. When the training stabilizes, recording the sample trajectories in the replay buffer produces the desired dataset, referred to as the expert-level dataset. Essentially, the optimal policies with benchmark performance generate the recorded trajectories, with which one trains the transformer in the auto-regressive manner presented in Section 3. Consequently, the transformer extracts the trajectory distribution under the NE policy and generates actions accordingly online, which could easily break down if the opponent does not follow the equilibrium policy [1].

Even though our proposed transformer follows the standard decoder-only transformer architecture and typical offline training dataset that records equilibrium trajectory distribution, we aim to make the transformer agent mindful of the opponent's nonstationary behavior through an extra sequence modeling on the opponent's action, which bears the same spirit of the belief generation in SCE and opponent modeling (OM). What distinguishes ours from existing works on transformer-based OM is that we train the transformer in a self-confirming way so that the policy π_i and belief μ_i are integrated into a single transformer, whereas prior works utilize transformers to represent μ_i and rely on additional policy search methods to configure π_i [19, 46].

Self-Confirming Loss. Recall that the transformer model for agent i is denoted by q_{θ_i}, which generates actions auto-regressively using past local trajectory. Let $\hat{\tau}_i^t = \{\hat{R}_i^1, o_i^1, \hat{a}_i^1, \ldots, \hat{R}_i^t, o_i^t, \hat{a}_i^t\}$ be agent i local trajectory (which only keeps individual information feedback

[1] Even though, except for a few value-based algorithms [13, 16, 28] with certified equilibrium convergence, most policy-gradient-based algorithms prove to be convergent to stationary points or approximate equilibrium [27, 33, 42, 43, 61], the key observation is that the resulting benchmark RL policies produce a fixed trajectory distribution corresponding to the stationary point in the offline dataset. Besides, the adapted SCE does not consider subgame perfectness under partial observability [32, 40] for simplicity.

from the trajectory τ), and the action generation in vanilla DT is given by $\hat{a}_i^t = \arg\max_{a \in \mathcal{A}_i} q_{\theta_i}(a|\hat{\tau}_i^{t-1}, o_i^t)$.

Inspired by Definition 4.1, we first let the transformer predict the opponent's action [see (3a)], based on which the transformer generates the agent's action in responding to the belief [see (3b)].

$$\hat{a}_{-i}^t = \arg\max_{a_{-i} \in \mathcal{A}_{-i}} q_{\theta_i}(a_{-i}|\hat{\tau}^{t-1}, o_i^t), \tag{3a}$$

$$\hat{a}_i^t = \arg\max_{a_i \in \mathcal{A}_i} q_{\theta_i}(a_i|\hat{\tau}^{t-1}, o_i^t, \hat{a}_{-i}^t). \tag{3b}$$

Note that the opponent action a_{-i}^t is unobservable to the ego agent during the implementation, and hence, the prediction $\hat{a}_{-i}^t$ is a fictitious token represents the agent's subjective inference extracted from the past observations. Such a token directs the transformer's future action generation, which we call belief conditioning. Compared with return conditioning, belief conditioning explicitly provides the information statistics of the opponent's actions. Of particular note is that such conditioning is created in a bootstrapping manner without intervention; that is, the transformer plays dual roles of both belief μ_i in (2) and policy π_i in (1).

After presenting its online implementation, we now shift the focus to offline training. To facilitate the discussion, we denote by π_i and π_{-i} (dropping the information feedback for simplicity) the policies used to collect the offline training data and assume π_i is the best response to π_{-i}: $\pi_i \in \arg\max_\pi \mathbb{E}_{\pi,\pi_{-i}}[\sum_{t=1}^T \gamma^{t-1} r_i^t]$. From Definition 4.1, one can see that the transformer needs to ensure that 1) the belief generation is consistent with the opponent's actual action [see (2)] and 2) the action generation is the best response to the belief [see (1)]. We propose the following cross-entropy loss: $\mathcal{L}_{\text{SCL}}(\theta_i) = \mathcal{L}_{\text{belief}}(\theta_i) + \mathcal{L}_{\text{policy}}(\theta_i)$, referred to as the self-confirming loss (SCL), to meet the two requirements simultaneously.

$$\mathcal{L}_{\text{belief}}(\theta_i) = 1/T \sum_{t=1}^T \pi_{-i}(\hat{a}_{-i}^t) q_{\theta_i}(\hat{a}_{-i}^t|\hat{\tau}_i^{t-1}, o_i^t). \tag{4a}$$

$$\mathcal{L}_{\text{policy}}(\theta_i) = 1/T \sum_{t=1}^T \pi_i(\hat{a}_i^t) q_{\theta_i}(\hat{a}_i^t|\hat{\tau}_i^{t-1}, o_i^t, \hat{a}_{-i}^t). \tag{4b}$$

Minimizing the belief consistency loss in (4a) is equivalent to minimizing the total variation distance between the SCT's belief generation and the opponent's actual policy since Pinsker's inequality tells that the square root of cross entropy upper bounds the total variation [47]. Ideally, the policy loss should correspond to the maximization problem in (1). Yet, since we assume the offline policy π_i is the best response, we can simply let the transformer imitate such policy by minimizing the cross entropy. The benefit is straightforward: two loss functions are cross entropy and fit the widely adopted auto-regressive training paradigm.

Some remarks regarding the training practice are in order. First, if access to the offline policies π_i, π_{-i} is not available, one can replace them with sample averages as in [4, 12]. Second, if the action is continuous, one can consider the mean-square loss [4]: $\mathcal{L}_{SCL}(\theta_i) = \|a_{-i}^t - \hat{a}_{-i}^t\|^2 + \|a_i^t - \hat{a}_i^t\|^2$, where a_i^t and a_{-i}^t denote the actions in the offline dataset, while $\hat{a}_i^t$ and $\hat{a}_{-i}^t$ are transformer's generated outputs. Third, the self-confirming loss does not include a weighing parameter to balance the two parts since the two are equally important. Finally, we remark that since the interdependency between the belief conditioning and the action generation, the SCL is actually a composite function, and its exact gradient computation $\nabla \mathcal{L}_{\text{SCL}}(\theta_i)$ is sophisticated. Encouraged by the recent success of first-order gradient approximation in stochastic composite optimization [27, 38], we ignore the interdependency and compute the first-order gradient as if the belief and action generation were independent.

5 EXPERIMENTS

This section seeks to empirically answer the question we raised at the beginning: can SCT adapt to nonstationary opponents online? Relatedly, one may wonder whether the offline-trained belief generation produces consistent beliefs online. If so, to what extent, does SCT's success depend on the belief (ablation)?

Environments. Following the benchmarking testbed in the literature, e.g., [41, 51], we consider `simple-tag` and `simple-world` discussed in the motivating example. We begin with `simple-tag` and present the setup of the partial observation, action, and reward of each agent. The partial observation of the prey contains its own velocity and position, and its relative positions to obstacles and other agents. The action variable of the prey is a two-dimensional vector, each entry of which ranges from -1 to 1. As the prey aims to escape from predators, it gets a positive reward proportional (the factor is 0.1) to the sum of its distance from each predator, while it is penalized for being caught by any of the predators (-10 reward).

The partial observation of one predator consists of its own velocity and position, its relative positions to the obstacles and other agents, and the prey's velocity. The predator's action space is the same as the prey's. The predator is rewarded +10 after hitting the prey, otherwise penalized by the relative distance to the prey. The obstacles are introduced to complicate the environment. Observable to all agents, obstacles are stationary once initialized within an episode. We set one prey, three predators, and two obstacles in this environment. Consequently, the observation space of prey is 14-dimensional: 2 for its velocity, 2 for its position, 4 for the relative position to obstacles, and 6 for other relative positions to agents. Similarly, the observation space of predators is 16-dimensional, and the additional two entries correspond to the prey's velocity.

`simple-world` is a more challenging task, where there are additional food particles that the prey is rewarded for being close to. the prey is rewarded +2 points for every time it hits a food particle. The environment includes only one obstacle. For the two environments, the episode length is $T = 25$. Yet, when training the transformer models, the context length is 20, i.e., the past 20 steps are used to calculate the loss. Table 1 summarizes the hyperparameters involved in the training of SCT and baseline methods.

Offline Datasets. The offline trajectories representing random, medium, and expert levels of play are divided into three datasets, where each dataset consists of 1 million transitions. The random dataset is obtained from unrolling episodes of a randomly initialized policy. The medium dataset is obtained by stopping the training phase of MATD3 once it reaches a medium level of play and then unrolling the episodes. The expert-level dataset is given by collecting transitions from the MATD3 once it is fully trained.

Baselines. We conduct a comparative study between SCT and existing works based on imitation learning, offline MARL, and sequence modeling. Specifically, we consider the following baselines.

Table 1: A summary of training hyperparameters.

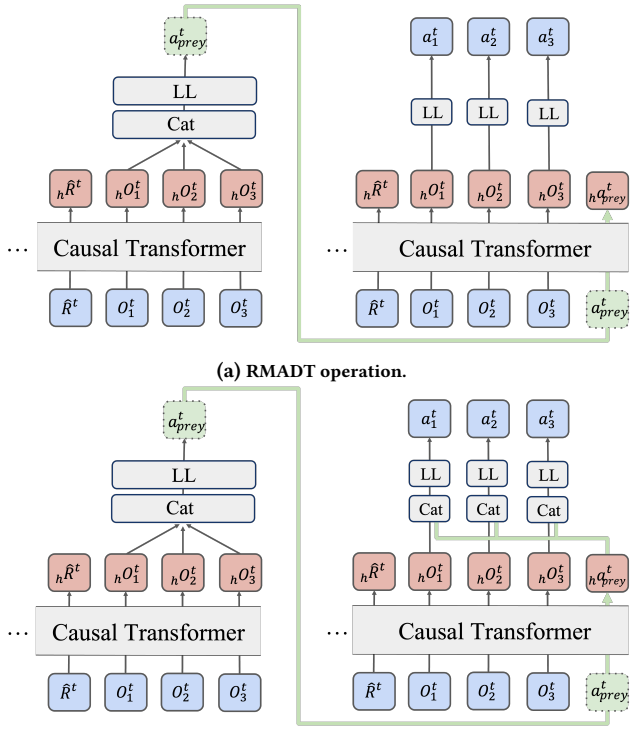
Hyperparameter	Value
Self-Confirming Transformer	
Maximum context Length	20
Batch Size	64
Hidden Dimensions	128
# of Layers	3
# of Attention Heads	1
Activation function	ReLU
# Steps per epoch	10000
# Epochs	1 for Medium and Expert 10 for Random
Learning Rate	1e-4
Weight Decay	1e-4
Behavior Cloning	
Maximum Context Length	20
Batch Size	64
Hidden Dimensions	128
# of Layers	3
Dropout	0.1
# Steps per Epoch	10000
# Epochs	15
Learning Rate	1e-4
Weight Decay	1e-4

1) Behavior Cloning (BC). Behavior Cloning is an imitation learning algorithm [17] where the three predators' behavior recorded in the dataset is replicated. The implementation consists of a Multilayer Perceptron with ReLu activation and dropout. The input consists of the three predator's observation histories concatenated and flattened. We utilize mean squared error loss during training, with the dataset's actions as ground truth. The hyperparameters are summarized in Table 1.

2) Multi-Agent Batch-Constrained Q-Learning (MA-BCQ). Batch-constrained Q-learning [11] imposes constraints on the action space to compel the agent to align more closely with on-policy behavior regarding a subset of the provided data. We implement MA-BCQ (**MA-BCQ**) based on the BCQ implementation provided by [58]. Considering the fact that BCQ employs two Q networks for a single agent, MA-BCQ includes six Q networks, as each predator needs two critics. The QMixer network in MA-BCQ takes in six Q values and outputs one Q value to evaluate the joint actions of predators. We follow the hyperparameter setup in [11].

3) Offline MARL with Actor Rectification (OMAR). Assuming an actor-critic architecture, OMAR uses zeroth-order information to rectify the critic so as to update the actor conservatively. In addition to BCQ and OMAR, there exist many other competitive baselines, such as CQL [20] and ICQ [58]. However, it is reported in [41] that OMAR outperforms CQL and ICQ in `simple-tag` and `simple-world`. We follow the official implementation of OMAR offered by the authors [41].

4) Transformer Models. Finally, we consider transformer-based models for the ablation studies, which include the **MADT** discussed in the motivating example. To investigate the role of belief conditioning, we consider a middle point between MADT and SCT, which we call belief-regularized MADT (RMADT). Similar to SCT, RMADT also generates a belief using past observations, yet such a belief is not fed back to the transformer for action generation. It only appears in the belief loss as a regularizer to adjust the auto-regressive training. Figure 4 visualizes the RMADT and SCT operation. In SCT's multi-agent implementation, the transformer (attention module) first generates the hidden states ${}_h{o_1}^t, {}_h{o_2}^t, {}_h{o_3}^t$ of the three



(a) RMADT operation.

(b) SCT operation.

Figure 4: A comparison between RMADT and SCT operation. The belief generation in RMADT does not direct the action generation. Even though the two share the same loss function, RMADT only aims to accurately predict the opponent's action, and the resulting action is not self-confirming.

predators' partial observations, which are then concatenated (Cat) and passed through a linear layer (LL) to predict the opponent's action (the green block), which becomes part of the input stream. In contrast, RMADT merely generates the prey's action prediction and aims to close the gap between the belief and the actual opponent's action. Yet, the belief is not used for action generation.

Opponent in Testing. To evaluate the adaptability of the predators, the prey is controlled by five distinct policies for each task. These opponent policies include 1) the **MATD3** policy, the one used to collect the training data, 2) **MADDPG** policy, an actor-critic policy trained for each environment, 3) **Random** policy: a heuristic-based policy designed to randomly sample feasible actions, 4) **Still** policy: a simple policy that freezes the prey at the initialized location, 5) **Blend** policy, the blending policy introduced in the motivating example with 50% blending rate.

Quantitative Results. First, to complete the story in the motivating example, we add SCT's normalized scores to the bar plots in Figure 3, leading to Figure 5. The figure suggests that SCT adapts better to nonstationary opponents in testing than MADT.

We also report in Table 2 the mean and standard deviation of normalized scores (see [60, Sec. 5]) based on 100 runs using different random seeds in `simple-tag` and `simple-world`. We observe that SCT consistently outperforms both MA-BCQ and OMAR across all experiments, except for the random dataset case where MA-BCQ leads extensively. Furthermore, SCT's performance is on par with or superior to the BC approach. Notably, SCT exhibits greater adaptability compared to its basic counterpart, as indicated by higher

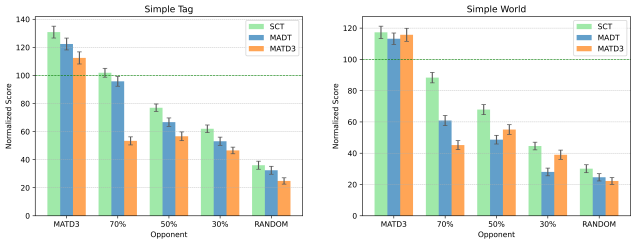


Figure 5: The normalized scores of SCT in `simple-tag` and `simple-world` environments. SCT outperforms MADT when facing nonstationary opponents.

Table 2: The normalized scores of SCT and baseline algorithms trained under the expert, medium, and random datasets in `simple-tag` and `simple-world`. SCT exhibits greater online adaptability than those baselines.

	Simple-Tag	MATD3	MADDPG	Still	Random	Blend
Expert	OMAR	103.19 ± 8.29	9.05 ± 1.62	38.80 ± 7.54	24.13 ± 4.08	58.02 ± 4.95
	BC	121.11 ± 7.81	11.22 ± 3.24	44.57 ± 6.8	35.7 ± 4.95	73.65 ± 5.04
	MA-BCQ	113.92 ± 8.16	10.52 ± 1.52	31.67 ± 6.83	31.26 ± 5.25	67.39 ± 5.97
	MADT	123.32 ± 8.04	8.16 ± 1.57	44.10 ± 7.9	32.83 ± 5.48	76.90 ± 5.23
	RMADT	122.94 ± 4.50	7.53 ± 1.80	32.52 ± 2.74	38.28 ± 4.29	70.04 ± 20.87
	SCT	**126.20 ± 7.48**	**11.94 ± 1.79**	**54.87 ± 7.54**	**38.98 ± 4.97**	**92.87 ± 5.92**
Medium	OMAR	72.61 ± 6.69	10.78 ± 1.21	42.49 ± 6.67	24.68 ± 3.73	44.14 ± 4.78
	BC	77.49 ± 6.93	11.60 ± 2.72	43.20 ± 6.39	31.41 ± 4.11	54.05 ± 4.35
	MA-BCQ	56.08 ± 6.05	10.36 ± 1.39	33.75 ± 6.27	25.65 ± 3.98	49.51 ± 4.6
	MADT	73.96 ± 5.76	11.98 ± 1.60	37.81 ± 5.96	27.58 ± 4.05	50.47 ± 4.42
	RMADT	74.67 ± 3.5	8.90 ± 1.31	29.21 ± 1.93	30.53 ± 2.2	44.15 ± 2.10
	SCT	**79.33 ± 5.80**	**12.22 ± 1.55**	**52.87 ± 6.02**	**34.78 ± 3.52**	**61.54 ± 5.05**
Random	OMAR	6.71 ± 3.03	−1.48 ± 0.43	1.53 ± 2.06	1.50 ± 1.11	3.53 ± 1.51
	BC	−0.31 ± 1.13	−3.27 ± 0.94	−1.48 ± 0.86	−1.48 ± 1.09	0.03 ± 1.08
	MA-BCQ	**32.86 ± 5.73**	**3.79 ± 1.16**	**28.1 ± 6.94**	**6.18 ± 2.94**	**12.9 ± 3.16**
	MADT	9.94 ± 2.56	−0.23 ± 0.63	4.31 ± 2.01	3.07 ± 1.49	4.97 ± 1.65
	RMADT	8.76 ± 1.36	−2.43 ± 0.20	3.44 ± 1.23	0.71 ± 4.25	4.93 ± 0.88
	SCT	24.75 ± 2.73	−0.03 ± 0.78	6.36 ± 1.73	5.03 ± 1.51	6.37 ± 1.57
	Simple-World	MATD3	MADDPG	Still	Random	Blend
Expert	OMAR	114.26 ± 3.52	−4.29 ± 8.33	34.52 ± 3.76	23.58 ± 2.21	48.33 ± 2.82
	BC	111.08 ± 3.12	−9.05 ± 6.10	37.73 ± 3.68	25.31 ± 2.16	53.14 ± 2.52
	MA-BCQ	106.59 ± 3.38	−2.75 ± 6.79	45.33 ± 2.96	21.62 ± 1.63	51.07 ± 2.81
	MADT	105.83 ± 3.18	−2.85 ± 6.76	31.69 ± 3.33	23.34 ± 21.3	48.61 ± 2.66
	RMADT	110.44 ± 3.42	−1.54 ± 7.70	41.76 ± 3.98	23.69 ± 2.13	75.90 ± 3.58
	SCT	**115.20 ± 3.38**	**−1.32 ± 7.15**	**53.31 ± 3.39**	**28.28 ± 2.29**	**92.87 ± 5.92**
Medium	OMAR	73.81 ± 4.46	−0.84 ± 7.23	**58.67 ± 3.73**	31.37 ± 1.73	42.23 ± 2.55
	BC	86.23 ± 2.39	−5.48 ± 10.47	41.11 ± 3.38	29.84 ± 2.06	43.45 ± 2.32
	MA-BCQ	76.99 ± 3.31	−3.00 ± 9.88	36.02 ± 3.71	30.12 ± 1.99	40.92 ± 2.38
	MADT	81.70 ± 2.84	0.54 ± 6.51	28.92 ± 2.82	26.07 ± 1.95	43.32 ± 2.22
	RMADT	86.87 ± 2.93	0.96 ± 8.11	30.25 ± 3.79	28.52 ± 2.13	55.78 ± 2.92
	SCT	**87.13 ± 2.97**	**1.30 ± 5.51**	33.41 ± 3.02	**31.87 ± 1.96**	**57.87 ± 2.66**
Random	OMAR	8.37 ± 1.16	−5.51 ± 0.87	4.54 ± 1.01	5.41 ± 0.77	6.39 ± 0.85
	BC	−0.62 ± 0.62	−13.41 ± 5.28	−0.75 ± 0.58	0.06 ± 0.69	0.15 ± 0.68
	MA-BCQ	6.52 ± 1.42	**−3.11 ± 7.98**	**6.59 ± 1.38**	3.40 ± 0.81	4.01 ± 0.85
	MADT	5.28 ± 1.10	−7.21 ± 5.87	1.27 ± 0.96	4.85 ± 0.89	4.28 ± 0.80
	RMADT	8.36 ± 1.27	−7.54 ± 5.96	2.93 ± 1.01	4.61 ± 0.82	**8.12 ± 0.93**
	SCT	**8.95 ± 1.42**	−4.47 ± 5.80	4.93 ± 0.81	**6.09 ± 0.91**	6.37 ± 1.57

mean rewards in most experiments (orange entries in Table 2). We also observe that most of the outliers (colored in red) in Table 2 pertain to MA-BCQ pre-trained over the random dataset, for which we speculate that the batch constraint better controls the extrapolation error than OMAR. Since this work focuses on transformer models, we leave the speculation for future investigation.

Ablation. We compare SCT with MADT and RMADT to see to what extent the belief generation contributes to the SCT's success. Recall that MADT is trained to generate predators' actions using the offline trajectories without any beliefs about the prey. Even though RMADT also generates the belief and uses the same loss function as SCT, i.e., $\mathcal{L}(\theta_i) = \|\hat{a}^t_{prey} - a^t_{prey}\|^2 + \|\hat{a}^t_{pred} - a^t_{pred}\|^2$. However, the belief $\hat{a}^t_{prey}$ and the action $\hat{a}^t_{pred}$ simultaneously based on past observations, whereas SCT first generates the conjecture that later serves as the input to the action generation, see Figure 4 for visualization of SCT and RMADT. We record the opponent's action prediction accuracy of SCT and RMADT in `simple-tag` and `simple-world`. The accuracy metric is defined as follows.

Table 3: A comparison of the prediction accuracy of SCT and RMADT in `simple-tag` and `simple-world`.

	simple-tag	MATD3	MADDPG	Still	Random	Blend
Exp	SCT	1.000	0.690	0.978	0.310	0.803
	RMADT	1.000	0.611	0.910	0.293	0.690
Med	SCT	1.000	0.665	0.996	0.300	0.888
	RMADT	1.000	0.644	0.988	0.324	0.872
Rand	SCT	1.000	0.823	1.000	0.301	0.865
	RMADT	1.000	0.843	1.000	0.300	0.630
	simple-world	MATD3	MADDPG	Still	Random	Blend
Exp	SCT	1.000	0.082	1.000	0.285	0.713
	RMADT	1.000	0.088	1.000	0.282	0.661
Med	SCT	1.000	0.102	1.000	0.293	0.650
	RMADT	1.000	0.088	1.000	0.229	0.577
Rand	SCT	1.000	0.092	1.000	0.285	0.707
	RMADT	1.000	0.054	1.000	0.287	0.627

$$\textbf{Accuracy} = \frac{\#\text{steps with accurate predictions}}{\#\text{total steps}}.$$

We consider an opponent's action prediction $\hat{a}^t_{-i}$ accurate if its relative error falls within an ϵ-neighborhood of the ground truth: $\hat{a}^t_{-i} \in \{a : \|a - a^t_{-i}\|/\|a^t_{-i}\| < \epsilon\}$. The prediction accuracy of an episode indicates the number of steps at which the prediction is accurate. We set the radius to be $\epsilon = 10\%$. As shown in Table 3, SCT and RMADT return comparable results, suggesting that the two acquire similar forecasting abilities. Hence, the superiority of SCT, as indicated in Table 2, shows that belief conditioning in SCT plays a bigger part than regularization in RMADT.

Self-Confirming Play in Iterated Prisoner's Dilemma. We further apply the proposed SCT to the iterated prisoner's dilemma (IPD) [35]. Compared with more sophisticated MARL environments above, IPD presents a repeated matrix game environment, enabling a close inspection of the equilibrium behaviors of the proposed SCT. The primary question we ask is: does SCT play the SCE? Similar to our previous discussions, we need to investigate 1) whether the SCT can form accurate beliefs on the opponent's play and 2) whether the SCT can optimize its own action sequence based on its beliefs. More importantly, the SCE in IPD admits a simple strategy representation, which is referred to *win-stay-lose-shift* (WSLS) [39] or *pavlov* [55], to be introduced later with the game matrix. We aim to inspect if SCT's action generation coincides with SCE.

Iterated Prisoner's Dilemma. We begin by introducing the prisoner's dilemma game. Suppose two criminals (the row and column players) are arrested and imprisoned, each of whom can cooperate (denoted by C) for mutual benefit (a lesser charge for both, denoted by R) or betray their partner ("defect", denoted by D) for individual freedom (denoted by T) while leaving the other with the charge (denoted by S). If they both choose to defect, then the two face the same charge (denoted by P). Table 4 presents the payoffs of players under different action profiles. A customary setup requires that $T > R > P > S$ and T + S < 2R, and the classical chosen values are presented in the table [35]. In IPD, two players play the matrix game in Table 4 repeatedly with full observations of the other's past plays. The iterated plays enable players to adjust their strategies to the

Table 4: The payoff matrix of the prisoner dilemma.

		Player II	
		(C)ooperate	(D)efect
Player I	(C)ooperate	R=3, R=3	S=0, T=5
	(D)efect	T=5, S=0	P=1, P=1

opponent's behavior pattern. One of the self-confirming equilibria in IPD is given by the **pavlov** strategy: cooperates on the first move and then, starting from the second round, cooperates if and only if both players opt for the same action in the previous move.

Offline Dataset. We curate a training dataset by collecting action sequences under diverse strategies provided in [35]. The selected strategies include **all_d**, **all_c**, **tit_for_tat**, **spiteful**, **soft_majo**, **hard_majo**, **per_ccd**, **per_ddc**, **mistrust**, **per_cd**, **tf2t**, **hard_tft**, **slow_tft**, **gradual**, **prober**, and **mem2**. The detailed descriptions of these strategies are in [35, Sec. 3.3]. Each strategy participates in a round-robin tournament against all other strategies, including itself. Each game consisted of 100 rounds, and the results were recorded in the final dataset. SCT is trained on this data to predict the opponent's action and use this prediction to generate its own. The loss function follows (4).

Evaluation. We test SCT's adaptability to different opponents' strategies by playing the transformer against each strategy recorded in the training dataset. We compare the normalized cumulative payoffs and prediction accuracy (the higher, the better) and present the results in Table 5. Note that when generating beliefs and actions, we configure the transformer to pick the one with the maximum likelihood, and hence, there is no stochasticity in testing (unlike the previous experiments). The final cumulative reward was 4840 (unnormalized for ranking purposes), placing SCT as the best strategy overall (refer to [35] for the complete ranking). Moreover, we measure its prediction accuracy against each opponent and obtain a mean accuracy of 97.82%.

In addition, we inspect SCT's adaptation ability to unseen strategies by testing the transformer against a new family of strategies. We consider a class of finite-memory-based strategies defined in [35], denoted by $\textsc{memory}(X, Y)$, which determines the future actions using the ego agent's last X plays and opponent's Y plays. Following the definition in [35], $\textsc{memory}(X, Y)$ begins with the $\max(X, Y)$ first moves, and the agent observes the $\max(X, Y)$ rounds of prisoner dilemma game. Then, based on its last X actions and its opponent's last Y actions, the agent determines the future plays using a deterministic mapping from $\{\mathrm{C}, \mathrm{D}\}^{X+Y}$ to $\{\mathrm{C}, \mathrm{D}\}$. We pick ten best-performing $\textsc{memory}(1, 2)$ and $\textsc{memory}(2, 1)$ strategies in the tournament ranking [35, Sec. 5.10] and refer to these strategies, in order, as $\textsc{mem}_s$, for $1 \le s \le 10$.

We report the reward and belief accuracy in Table 6 when testing SCT against these unseen memory-based strategies. The total reward is 2752, with a mean accuracy of 90.9% when predicting these strategies. This places the SCT as the best strategy in the tournament with the unseen strategies, surpassing the second by a margin of 200.

SCT Equilibrium Behavior. We now discuss some interesting behavior patterns displayed by SCT when testing against three representative strategies: **all_d** (always defect), **all_c** (always cooperate), and the SCE strategy **pavlov**. The first observation is that

Table 5: Normalized rewards and prediction accuracy of SCT against training strategies.

	all_d	tit_for_tat	spiteful	soft_majo	hard_majo	per_ddc	per_ccd	mistrust
reward	87.93	110.25	142.34	107.59	144.81	118.60	116.57	143.36
accuracy	0.99	0.99	0.99	0.99	0.98	0.96	0.98	0.98
all_c	**per_cd**	**pavlov**	**tf2t**	**hard_tft**	**slow_tft**	**gradual**	**prober**	**mem2**
122.40	128.44	107.82	103.47	129.90	109.23	114.66	153.66	134.71
0.99	0.94	0.99	0.99	0.99	0.99	0.99	0.90	0.99

Table 6: Normalized rewards and prediction accuracy of SCT against unseen strategies.

	$\textsc{mem}_1$	$\textsc{mem}_2$	$\textsc{mem}_3$	$\textsc{mem}_4$	$\textsc{mem}_5$	$\textsc{mem}_6$	$\textsc{mem}_7$	$\textsc{mem}_8$	$\textsc{mem}_9$	$\textsc{mem}_{10}$
reward	101.65	106.96	236.87	135.07	135.07	135.88	175.60	175.60	130.31	437.77
accuracy	0.96	0.96	0.96	0.96	0.56	0.84	0.96	0.95	0.95	0.99

Table 7: SCT v.s. Pavlov. After first a few rounds of cooperation, SCT starts to exploit Pavlov. Yet, Pavlov will penalize such exploitation, forcing SCT to cooperate and leading to NE plays.

pavlov	···	C	C	C	D	C	C	···
SCT	···	C	C	D	D	C	C	···

when playing against **all_d**, after the misbelief in the first step, SCT quickly realizes the opponent's **all_d** strategy (i.e., 0.99 accuracy) and plays **all_d** as well, reaching a NE equilibrium (D, D) in IPD (which is also a SCE). When playing against **pavlov**, SCT first chooses to cooperate for a few rounds. Since **pavlov** also opts for cooperation in these rounds, SCT starts to play defection to exploit the opponent for one round, misbelieving the unconditional cooperation from the opponent. Yet, such exploitation is penalized by the **pavlov** opponent, who also switches to defection one round later. Learning its lesson, SCT falls back to cooperation thereafter. Finally, the players reach another NE (SCE) (C, C), receiving higher rewards than in the first scenario. Another interesting observation is that when facing **all_c** opponent, SCT starts to exploit the opponent by playing defection. Yet, it also plays cooperation occasionally, for which we speculate that SCT memorizes some periodic plays in the offline dataset. In summary, belief conditioning and self-confirming loss equip the vanilla transformer with greater adaptability to sophisticated opponents.

6 CONCLUSION

Inspired by the self-confirming equilibrium (SCE), this work has developed a novel auto-regressive training paradigm for decision transformers in offline MARL tasks. The key operation of the proposed self-confirming transformer (SCT) is belief conditioning: the transformer first generates a fictitious token representing its inference about the opponent's action, which is then fed back to itself to generate its own action. The SCE-motivated loss consists of belief consistency loss and best response loss, mandating that the agent behave optimally under the correct belief. Experimental results in multi-particle environments demonstrate SCT's superior performance against nonstationary opponents unseen in the training. Moreover, when deployed in the iterated prisoner's dilemma, SCT indeed displays equilibrium behaviors as instructed by the self-confirming loss.

One of the most pressing future works is to investigate the interplay between return and belief conditioning. Our experiments employ the grid search to find the optimal return conditioning for all transformer models. Since the two conditionings are essentially the HIM technique [12], it would be helpful if the transformer could also self-adapt return conditioning, together with belief.

REFERENCES

[1] Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465* (2019).

[2] Stefano V. Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95. https://doi.org/10.1016/j.artint.2018.01.002 arXiv:1709.08071

[3] James Bannon, Brad Windsor, Wenbo Song, and Tao Li. 2020. Causality and batch reinforcement learning: Complementary approaches to planning in unknown domains. *arXiv preprint arXiv: 2006.02579* (2020). https://doi.org/10.48550/arxiv.2006.02579 arXiv:2006.02579

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[5] Caroline Claus and Craig Boutilier. [n.d.]. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI '98/IAAI '98)*. American Association for Artificial Intelligence, USA, 746–752.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[7] Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5884–5888. https://doi.org/10.1109/ICASSP.2018.8462506

[8] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219* (2020).

[9] Drew Fudenberg and David K Levine. 1993. Self-Confirming Equilibrium. *Econometrica* 61, 3 (1993), 523. https://doi.org/10.2307/2951716

[10] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.

[11] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[12] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. 2021. Generalized Decision Transformer for Offline Hindsight Information Matching. In *International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arxiv.2111.10364

[13] Amy Greenwald and Keith Hall. 2003. Correlated-Q Learning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03, Vol. 20)*. 242.

[14] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. 2018. Learning Policy Representations in Multiagent Systems. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1802–1811. https://proceedings.mlr.press/v80/grover18a.html

[15] Kim Hammar, Tao Li, Rolf Stadler, and Quanyan Zhu. 2024. Automated Security Response through Online Learning with Adaptive Conjectures. *arXiv* (2024). https://doi.org/10.48550/arxiv.2402.12499 arXiv:2402.12499

[16] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4, Nov (2003), 1039—1069.

[17] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation Learning: A Survey of Learning Methods. *ACM Comput. Surv.* 50, 2, Article 21 (April 2017), 35 pages. https://doi.org/10.1145/3054912

[18] Michael Janner, Qiyang Li, and Sergey Levine. 2022. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 1273–1286. https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf

[19] Yuheng Jing, Bingyun Liu, Kai Li, Yifan Zang, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. 2024. Opponent Modeling with In-context Search. In *Advances in Neural Information Processing Systems*, Vol. 37. 61549–61591. https://proceedings.neurips.cc/paper_files/paper/2024/file/710445227fa8c1b6a9ceada902dd4741-Paper-Conference.pdf

[20] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[21] Artúr István Károly, Péter Galambos, József Kuti, and Imre J. Rudas. 2021. Deep Learning in Robotics: Survey on Model Structures and Training Strategies. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 1 (2021), 266–279. https://doi.org/10.1109/TSMC.2020.3018325

[22] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv* (2020). arXiv:2005.01643

[23] Tao Li, Zilin Bian, Haozhe Lei, Fan Zuo, Ya-Ting Yang, Quanyan Zhu, Zhenning Li, Zhibin Chen, and Kaan Ozbay. 2024. Digital Twin-based Driver Risk-Aware Intelligent Mobility Analytics for Urban Transportation Management. *arXiv* (2024). https://doi.org/10.48550/arXiv.2407.15025 arXiv:2407.15025

[24] Tao Li, Zilin Bian, Haozhe Lei, Fan Zuo, Ya-Ting Yang, Quanyan Zhu, Zhenning Li, and Kaan Ozbay. 2024. Multi-level traffic-responsive tilt camera surveillance through predictive correlated online learning. *Transportation Research Part C: Emerging Technologies* 167 (2024), 104804. https://doi.org/10.1016/j.trc.2024.104804

[25] Tao Li, Kim Hammar, Rolf Stadler, and Quanyan Zhu. 2023. Conjectural online learning with first-order beliefs in asymmetric information stochastic games. In *2024 63rd IEEE Conference on Decision and Control (CDC)*. https://doi.org/10.48550/arxiv.2402.18781

[26] Tao Li, Haozhe Lei, and Quanyan Zhu. 2023. Self-Adaptive Driving in Nonstationary Environments through Conjectural Online Lookahead Adaptation. *2023 IEEE International Conference on Robotics and Automation (ICRA)* 00 (2023), 7205–7211. https://doi.org/10.1109/icra48891.2023.10161368 arXiv:2210.03209

[27] Tao Li, Henger Li, Yunian Pan, Tianyi Xu, Zizhan Zheng, and Quanyan Zhu. 2024. Meta stackelberg game: Robust federated learning against adaptive and mixed poisoning attacks. *arXiv preprint arXiv:2410.17431* (2024). https://doi.org/10.48550/arXiv.2410.17431

[28] Tao Li, Guanze Peng, and Quanyan Zhu. 2021. Blackwell online learning for Markov decision processes. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*. 1–6. https://doi.org/10.1109/CISS50987.2021.9400319

[29] Tao Li, Guanze Peng, Quanyan Zhu, and Tamer Baar. 2022. The Confluence of Networks, Games, and Learning a Game-Theoretic Framework for Multiagent Decision Making Over Networks. *IEEE Control Systems* 42, 4 (2022), 35–67. https://doi.org/10.1109/mcs.2022.3171478

[30] Tao Li, Yuhan Zhao, and Quanyan Zhu. 2022. The role of information structures in game-theoretic multi-agent learning. *Annual Reviews in Control* 53 (2022), 296–314. https://doi.org/10.1016/j.arcontrol.2022.03.003

[31] Tao Li and Quanyan Zhu. 2019. On Convergence Rate of Adaptive Multiscale Value Function Approximation for Reinforcement Learning. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6. https://doi.org/10.1109/MLSP.2019.8918816

[32] Tao Li and Quanyan Zhu. 2023. On the Price of Transparency: A Comparison Between Overt Persuasion and Covert Signaling. In *2023 62nd IEEE Conference on Decision and Control (CDC)*. 4267–4272. https://doi.org/10.1109/CDC49753.2023.10383897

[33] Shutian Liu, Tao Li, and Quanyan Zhu. 2023. Game-Theoretic Distributed Empirical Risk Minimization With Strategic Network Design. *IEEE Transactions on Signal and Information Processing over Networks* 9 (2023), 542–556. https://doi.org/10.1109/tsipn.2023.3306106

[34] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30 (Advances in Neural Information Processing Systems)*. Curran Associates, Inc., 6379–6390. http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf

[35] Philippe Mathieu and Jean-Paul Delahaye. 2017. New Winning Strategies for the Iterated Prisoner's Dilemma. *Journal of Artificial Societies and Social Simulation* 20, 4 (2017), 12. https://doi.org/10.18564/jasss.3517

[36] Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, and Bo Xu. 2023. Offline Pre-trained Multi-agent Decision Transformer. *Machine Intelligence Research* 20, 2 (2023), 233–248. https://doi.org/10.1007/s11633-022-1383-7

[37] John Nash. 1951. Non-Cooperative Games. *The Annals of Mathematics* 54, 2 (1951), 286–295. https://doi.org/10.2307/1969529

[38] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. *arXiv* (2018). arXiv:1803.02999

[39] Martin Nowak and Karl Sigmund. 1993. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature* 364, 6432 (1993), 56–58. https://doi.org/10.1038/364056a0

[40] Yi Ouyang, Hamidreza Tavafoghi, and Demosthenis Teneketzis. 2016. Dynamic Games With Asymmetric Information: Common Information Based Perfect Bayesian Equilibria and Sequential Decomposition. *IEEE Trans. Automat. Control* 62, 1 (2016), 222–237. https://doi.org/10.1109/tac.2016.2544936

[41] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. 2022. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International Conference on Machine Learning*. PMLR, 17221–17237.

[42] Yunian Pan, Tao Li, Henger Li, Tianyi Xu, Zizhan Zheng, and Quanyan Zhu. 2023. A First Order Meta Stackelberg Method for Robust Federated Learning. In *Adversarial Machine Learning Frontiers Workshop at 40th International Conference on Machine Learning*. https://doi.org/10.48550/arxiv.2306.13800

[43] Yunian Pan, Tao Li, and Quanyan Zhu. 2023. Is Stochastic Mirror Descent Vulnerable to Adversarial Delay Attacks? A Traffic Assignment Resilience Study. In *2023 62nd IEEE Conference on Decision and Control (CDC)*. 8328–8333. https://doi.org/10.1109/CDC49753.2023.10384003

[44] Yunian Pan, Tao Li, and Quanyan Zhu. 2023. On the Resilience of Traffic Networks under Non-Equilibrium Learning. In *2023 American Control Conference (ACC)*. 3484–3489. https://doi.org/10.23919/ACC55779.2023.10156139

[45] Yunian Pan, Tao Li, and Quanyan Zhu. 2024. On the Variational Interpretation of Mirror Play in Monotone Games. *arXiv preprint arXiv:2403.15636* (2024). https://doi.org/10.48550/arXiv.2403.15636

[46] Georgios Papoudakis, Filippos Christianos, and Stefano Albrecht. 2021. Agent Modelling under Partial Observability for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 34. 19210–19222. https://proceedings.neurips.cc/paper_files/paper/2021/file/a03caec56cd82478bf197475b48c05f9-Paper.pdf

[47] Mark S. Pinsker. 1964. *Information and Information Stability of Random Variables and Processes*. Holden-Day. Transalted by Amiel Feinstein.

[48] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. [Online] Available at https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[49] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A Generalist Agent. *Transactions on Machine Learning Research* (2022).

[50] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[51] Wei-Cheng Tseng, Tsun-Hsuan Johnson Wang, Yen-Chen Lin, and Phillip Isola. 2022. Offline Multi-Agent Reinforcement Learning with Knowledge Distillation. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 226–237. https://proceedings.neurips.cc/paper_files/paper/2022/file/01d78b294d80491fecddea897cf03642-Paper-Conference.pdf

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeuriPS, Vol. 30)*. https://doi.org/10.48550/arxiv.1706.03762

[53] Conor Wallace, Umer Siddique, and Yongcan Cao. 2024. Opponent Transformer: Modeling Opponent Policies as a Sequence Problem. In *Coordination and Cooperation in Multi-Agent Reinforcement Learning Workshop (RLC)*.

[54] Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. 2022. Bootstrapped Transformer for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 34748–34761. https://proceedings.neurips.cc/paper_files/paper/2022/file/e0ccda3cb17b084a6f43c62cfac4784b-Paper-Conference.pdf

[55] C Wedekind and M Milinski. 1996. Human cooperation in the simultaneous and the alternating Prisoner's Dilemma: Pavlov versus Generous Tit-for-Tat. *Proceedings of the National Academy of Sciences* 93, 7 (1996), 2686–2689. https://doi.org/10.1073/pnas.93.7.2686

[56] Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 16509–16521. https://proceedings.neurips.cc/paper_files/paper/2022/file/69413f87e5a34897cd010ca698097d0a-Paper-Conference.pdf

[57] Xinhong Xie, Tao Li, and Quanyan Zhu. 2024. Learning from Response not Preference: A Stackelberg Approach for LLM Detoxification using Non-parallel Data. *arXiv preprint arXiv:2410.20298* (2024). https://doi.org/10.48550/arXiv.2410.20298

[58] Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. 2021. Believe What You See: Implicit Constraint Approach for Offline Multi-Agent Reinforcement Learning. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:235358250

[59] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. 2021. Reinforcement Learning in Healthcare: A Survey. *ACM Comput. Surv.* 55, 1, Article 5 (nov 2021), 36 pages. https://doi.org/10.1145/3477600

[60] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms. https://openreview.net/forum?id=t5lNr0Lw84H

[61] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control* (2021), 321–384.