Manipulating Hidden-Markov-Model Inferences by Corrupting Batch Data

William N. Caballero[a], Jose Manuel Camacho[b], Tahir Ekin[c], Roi Naveiro[d]

[a]*United States Air Force Academy, Colorado Springs, Colorado, USA.*
[b]*Institute of Mathematical Sciences (ICMAT), Madrid, Spain.*
[c]*McCoy College of Business, Texas State University, San Marcos, Texas, USA.*
[d]*CUNEF Universidad, Madrid, Spain.*

Abstract

Time-series models typically assume untainted and legitimate streams of data. However, a self-interested adversary may have incentive to corrupt this data, thereby altering a decision maker's inference. Within the broader field of adversarial machine learning, this research provides a novel, probabilistic perspective toward the manipulation of hidden Markov model inferences via corrupted data. In particular, we provision a suite of corruption problems for filtering, smoothing, and decoding inferences leveraging an adversarial risk analysis approach. Multiple stochastic programming models are set forth that incorporate realistic uncertainties and varied attacker objectives. Three general solution methods are developed by alternatively viewing the problem from frequentist and Bayesian perspectives. The efficacy of each method is illustrated via extensive, empirical testing. The developed methods are characterized by their solution quality and computational effort, resulting in a stratification of techniques across varying problem-instance architectures. This research highlights the weaknesses of hidden Markov models under adversarial activity, thereby motivating the need for robustification techniques to ensure their security.

Keywords: Adversarial Risk Analysis, Hidden Markov Models, Adversarial Machine Learning

1. Introduction

Hidden Markov models (HMMs) are stochastic processes over some time interval $\mathcal{T}$ which assume that an observed sequence of outputs, $\{x_t\}_{t\in\mathcal{T}}$, is a noisy observation from an underlying unobservable (hidden) sequence of states, $\{q_t\}_{t\in\mathcal{T}}$, that is, in turn, generated by a Markov chain, $\{Q_t\}_{t\in\mathcal{T}}$. In canonical HMMs, the state evolution model is a discrete-time, discrete-state, first-order Markov chain such that $\mathcal{T} = \{1, 2, \ldots, |\mathcal{T}|\}$ and $Q_t \in \mathcal{Q} = \{1, \ldots, |\mathcal{Q}|\}$. Transitions between Q_t and Q_{t+1} are governed by the state-transition probability matrix A having elements $a_{ij} = P(Q_{t+1} = j | Q_t = i), \forall (i, j) \in \mathcal{Q} \times \mathcal{Q}$, and initial-state probabilities are given by $\pi_i = P(Q_1 = i), \forall i \in \mathcal{Q}$. Furthermore, each observation $X_t \in \mathcal{X} = \{1, \ldots |\mathcal{X}|\}$ is determined by the observation probability matrix B having elements $b_{ik} = P(X_t = k | Q_t = i), \ \forall (i, k) \in \mathcal{Q} \times \mathcal{X}$. Figure 1 graphically depicts

this standard model; the nodes and arcs represent random variables and conditional dependencies, respectively.
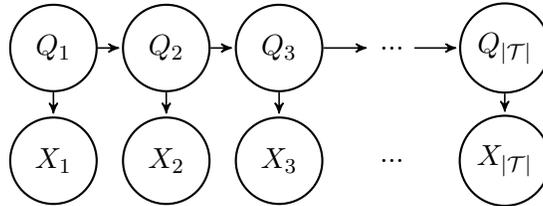

Figure 1: A basic hidden Markov model

Despite their relative simplicity, HMMs can be leveraged to perform a wide array of inferences and predictions. Rabiner (1989) discusses three commonly considered problems: (1) identifying the likelihood $P(\{X_t\}_{t \in \mathcal{T}})$ for a given HMM parameterization, (2) selecting the most likely sequence of latent states given an HMM model and a sequence of observations, i.e., decoding, and (3) learning to optimally parameterize, through maximum likelihood estimation, an HMM given a sequence of observations (e.g., estimating A and B). In addition to these standard problems, researchers and practitioners are also often interested in the probability of a specific hidden state at some time t. That is, given a sequence of observations, one may wish to infer $P(Q_{|\mathcal{T}|} = i | \{X_\tau\}_{\tau \in \mathcal{T}} = \{x_\tau\}_{\tau \in \mathcal{T}})$ and $P(Q_t = i | \{X_\tau\}_{\tau \in \mathcal{T}} = \{x_\tau\}_{\tau \in \mathcal{T}})$ for $t < |\mathcal{T}|$. These inferences are also known as filtering and smoothing, respectively. Bayesian approaches to HMMs (e.g., see Scott, 2002) also draw upon these building blocks, but vary mechanically based upon their alternative perspective. Moreover, whereas inference may be performed sequentially, HMMs often consider inference on batch data, i.e., using complete sequences of observations.

The utility of such predictions and inferences has been illustrated in myriad applications and diverse disciplines. From gesture recognition in computer science (Starner and Pentland, 1997) and chromatin state learning in computational biology (Ernst and Kellis, 2012) to stochastic thermodynamics in physics (Bechhoefer, 2015) and signal processing in electrical engineering (Crouse et al., 1998), HMMs have proven themselves to effectively characterize unobservable states based upon another related, observable process. Of such applications, speech recognition is, perhaps, the most successful and widely known historical use of HMMs (Gales et al., 2008) dating back to the canonical works summarized by Rabiner (1989). Security applications are also relevant including network intrusion detection (Scott, 2002) and spam detection (Gordillo and Conde, 2007).

Such successful applications drive increased incorporation of HMMs into commercial products and, as discussed by Biggio and Roli (2018), this increased utilization is accompanied by a wide array of security threats. More specifically, if an automated system utilizes a machine learning algorithm, a nefarious actor may attempt to subvert the system by manipulating its underlying statistical framework (e.g., by providing it corrupted data). The field of *adversarial machine learning* (AML) focuses on modeling these threats from both an offensive and a defensive perspective, i.e., modeling how the

attacks affect the algorithm and how to defend against the associated negative effects. Although AML is a relatively nascent field, much progress has been made in the last fifteen years since Dalvi et al. (2004) published the discipline's seminal work regarding adversarial classification. As a result of deep learning's recent widespread adoption in commercial products, neural networks have received substantial interest in the AML literature (Melis et al., 2017; Crecchi et al., 2020; Sotgiu et al., 2020), with support vector machines likely being a close second (Xiao et al., 2015; Indyk and Zabarankin, 2019; Alhajjar et al., 2021). Although classification and computer vision remain the predominant focus of AML research, other machine learning tasks have been emphasized in more recent investigations (e.g., see Jagielski et al., 2018; Caballero et al., 2021; Gallego et al., 2019). In particular, adversarial machine learning algorithms relating to temporal data and unsupervised learning are emerging areas of inquiry (e.g., see Alfeld et al., 2016; Chen and Zhu, 2020; Dang-Nhu et al., 2020; Naveiro, 2021; Hsu et al., 2021).

However, given the shear breadth and variety of available machine learning methods and variations, many techniques have been sparingly studied (if at all) from an AML perspective. HMMs are one such understudied technique of particular relevance (Caballero et al., 2020). Real-world application of the attacks abound, ranging from SMS spam misclassification (Xia and Chen, 2020) and target detection (Miller et al., 2015) to crisis prediction in international politics (O'Brien, 2010). Therefore, to address this gap in the literature, we develop herein multiple algorithmic methodologies that can be leveraged to corrupt HMM data such that the resulting inference is erroneous and benefits the interests of a nefarious actor. The development of such attacks is paramount to ensure the security of canonical HMM algorithms when the veracity of the underlying data is threatened; their weaknesses must first be identified before they can be strengthened.

This manuscript makes two primary contributions. Firstly, we provide a comprehensive collection of HMM corruption problems that encompass filtering, smoothing, and decoding inferences under realistic uncertainty conditions. Secondly, we develop and benchmark a set of attack frameworks that can be tailored to specific scenarios. Through extensive empirical testing, we demonstrate the effectiveness of our attacks and highlight the trade-off between solution quality and computational effort when corrupting larger-scale HMMs. In so doing, our research explores the limits of the developed frameworks, providing insight into their capabilities.

The organization for the remainder of this manuscript is as follows. We begin in Section 2 by providing requisite background on HMM inferences and our opponent modeling framework (i.e., adversarial risk analysis, Banks et al. (2015)). Subsequently, Section 3 formally defines a collection of HMM corruption problems for use against filtering, smoothing, and decoding inferences. Not only are the resultant mathematical programs non-linear and combinatorial in nature, but they are also characterized by uncertain parameters. Therefore, the considered models are stochastic programming problems for which most traditional solution techniques are unsuitable. Therefore, in Section 4, we set forth three approximation methods that enable the attacker to tractably identify high-quality solutions. These solution techniques are tested in Section 5 via a designed experiment over a subset of tunable, algorithmic parameters. The practical relevance of our methods is also illustrated via a case study whereby an HMM used for

part-of-speech tagging is attacked. Finally, we conclude this manuscript in Section 6 by providing closing remarks and presenting promising avenues of future research. The appendix provides implementation details of the computational algorithms.

2. Relevant, Contextual Background

This manuscript is multi-disciplinary in nature, integrating machine-learning and decision-analytic techniques within a competitive stochastic process. To ensure comprehension by a broad readership, this section briefly summarizes relevant background material regarding HMMs and adversarial risk analysis (ARA).

2.1. Inference and Prediction on Hidden Markov Models

HMMs, such as that provided in Figure 1, are probabilistic graphical models utilized in myriad applications. Akin to other graphical models (e.g., Bayesian networks), it is often of interest to update probabilities in the HMM by conditioning upon a subset of known variables. Given that Q_t is unobservable, such updates are performed in an HMM assuming some set of observed $\{x_t\}_{t \in \mathcal{T}}$. Provided this *batch* of observations, filtering, smoothing, and decoding are among the most frequently solved problems[1].

Algorithm 1 Forward Algorithm

---
    for $i \in \mathcal{Q}$ **do**
      $\alpha_{1,i} = \pi_i b_{i,x_1}$
    end for
    for $\tau = 2, \ldots, t$ and $i \in \mathcal{Q}$ **do**
      $\alpha_{\tau,i} = \sum_{j \in \mathcal{Q}} \alpha_{\tau-1,j} a_{j,i} b_{i,x_\tau}$
    end for

The filtering and smoothing problems can each be solved utilizing the Forward and Backward algorithms summarized in Algorithms 1 and 2. In the case of filtering, only Algorithm 1 must be utilized to identify the forward probabilities, i.e.

$$\alpha_{t,i} = P\left(Q_t = i, \{X_\tau\}_{\tau=1}^t = \{x_\tau\}_{\tau=1}^t\right).$$

Once identified, it can be shown that

$$P(Q_{|\mathcal{T}|} = i | \{X_\tau\}_{\tau \in \mathcal{T}} = \{x_\tau\}_{\tau \in \mathcal{T}}) = \frac{\alpha_{|\mathcal{T}|,i}}{\sum_{j \in \mathcal{Q}} \alpha_{|\mathcal{T}|,j}}.$$

Alternatively, for smoothing problems, the time of interest is some $t < |\mathcal{T}|$, implying that both the forward and backward algorithms must be leveraged. Once the backward probabilities, i.e.,

---

[1]The learning problem, i.e., fitting the most-likely HMM parameters given a set of observations, is another common HMM problem but not one that we address herein. We refer the interested reader to Rabiner (1989) for more information in this regard.

Algorithm 2 Backward Algorithm

1: **for** $i \in \mathcal{Q}$ **do**
2: $\quad \beta_{|\mathcal{T}|,i} = 1$
3: **end for**
4: **for** $\tau = |\mathcal{T}| - 1, \ldots, t$ and $i \in \mathcal{Q}$ **do**
5: $\quad \beta_{\tau,i} = \sum_{j \in \mathcal{Q}} \beta_{t+1,j} a_{i,j} b_{j,x_{\tau+1}}$
6: **end for**

$$\beta_{t,i} = P(\{X_\tau\}_{\tau=t+1}^{|\mathcal{T}|} = \{x_\tau\}_{\tau=t+1}^{|\mathcal{T}|} | Q_t = i).$$

are identified via Algorithm 2, it is known that

$$P\left(Q_t = i | \{X_\tau\}_{\tau \in \mathcal{T}} = \{x_\tau\}_{\tau \in \mathcal{T}}\right) = \frac{\alpha_{t,i} \beta_{t,i}}{\sum_{j \in \mathcal{Q}} \alpha_{t,j} \beta_{t,j}}. \tag{1}$$

Decoding problems, for their part, are typically addressed via application of the Viterbi algorithm summarized in Algorithm 3. Given some $\{x_t\}_{t \in \mathcal{T}}$, this algorithm functions by successively identifying the probability of the most likely latent-state path to $Q_t = i$, $\forall t \in \mathcal{T}, i \in \mathcal{Q}$. These are referred to as Viterbi probabilities and denoted by $\delta_{t,i}$. By storing the backtraces, i.e., $\psi_{t,i}$, the most likely sequences of states can be reconstructed by identifying $q^*_{|\mathcal{T}|} = \arg\max_{i \in \mathcal{Q}} \delta_{|\mathcal{T}|,i}$ and tracing the corresponding chain of $\psi_{t,i}$-values back to $t = 1$.

Algorithm 3 Viterbi Algorithm
---
for $i \in \mathcal{Q}$ **do**
$\quad$ Set $\delta_{1,i} = \pi_i b_{i,x_1}$ and $\psi_{1,i} = 0$
end for
for $t = 2, \ldots, |\mathcal{T}|$ and $i \in \mathcal{Q}$ **do**
$\quad \delta_{t,i} = \max_{j \in \mathcal{Q}} \delta_{t-1,j} a_{j,i} b_{i,x_t}$
$\quad \psi_{t,i} = \arg\max_{j \in \mathcal{Q}} \delta_{t-1,j} a_{j,i}$
end for
Set $q^*_{|\mathcal{T}|} = \arg\max_{i \in \mathcal{Q}} \delta_{|\mathcal{T}|,i}$
Set $q^*_t = \psi_{t+1,q^*_{t+1}}$ for $t = |\mathcal{T}| - 1, \ldots, 1$

2.2. Adversarial Risk Analysis

ARA is a Bayesian alternative to game-theoretic analysis of a competitive interaction. Whereas game theory solves every player's problem simultaneously in a given solution concept, ARA approaches the problem from a decision-theoretic perspective. That is, ARA extends canonical, decision-analytic methods to competitive interactions, by enabling a supported player to maximize their expected utility under aleatory, epistemic, and solution-concept uncertainties. Within this research, we adopt the ARA perspective toward opponent modeling to decompose the game into a decision problem for the

attacker. This is in juxtaposition to a game-theoretic approach that solves the game as a system to identify all possible equilibrium profiles.

The ARA approach to a competitive interaction can be visualized via the bi-agent influence diagrams (BAID) in Figure 2a and the corresponding ARA reduction in Figure 2b. Squares represent decision nodes for the corresponding players (i.e, D or Z), circles represent uncertainty nodes, and hexagons are utility nodes. A white fill corresponds to Player-D nodes and a gray fill to Player-Z nodes; shared nodes are depicted with striped white-and-gray fill. Figure 2a represents an arbitrary, sequential game. The dashed informational arc between the decision nodes indicates that Player D takes an action after having observed Player Z's choice. Both player's utilities are, in turn, affected by the players' joint action profile and the outcome of the uncertainty X. For more information on BAIDs, we refer the interested reader to Koller and Milch (2003).

(a) Arbitrary, Sequential Game
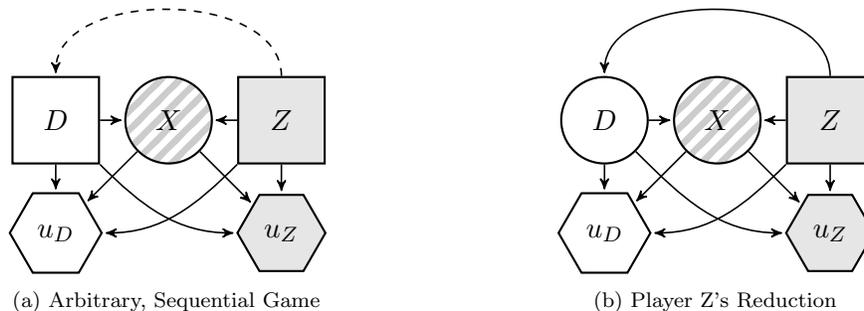
(b) Player Z's Reduction

Figure 2: A Visualization of the ARA Approach in a Sequential Game

Figure 2b illustrates how an ARA analysis decomposes this competitive interaction. Namely, assuming the ARA is supporting Player Z, Player D is treated as, simply put, another source of uncertainty. Whereas a multitude of opponent models may be leveraged within the ARA framework (Albrecht and Stone, 2018), most research adopts a recursive reasoning approach. This implies that a corresponding ARA reduction is constructed and solved from Player D's perspective; however, due to Player Z's uncertainty about their opponents beliefs, multiple instantiations of this reduction are solved to construct an empirical, probability distribution over Player D's actions. Once this distribution over Player D's actions has been identified, solving Player Z's problem can be accomplished via standard decision-analytic practices. Alternatively, Banks et al. (2011) illustrated how, under low information conditions, zeroth-order ARA allows for the direct elicitation of beliefs on the opponent's behavior.

Due to its generality, ARA can be applied to any competitive setting. A plurality of the ARA literature focuses on its application to physical security settings, e.g., counter-terrorism and military operations. However, recent research highlights its promise for adversarial machine learning as well (e.g., see Naveiro et al., 2019; Rios Insua et al., 2023; González-Ortega et al., 2021). Canonical AML techniques are rooted in game-theoretic analysis, and inherit the associated common knowledge assumption. An ARA approach allows one to loosen this assumption when it is inappropriate, e.g., for the security

settings considered herein.

3. Hidden Markov Model Corruption Problems

We consider herein an attacker, i.e., Player Z, attempting to thwart inference conducted on a HMM by a decision maker, i.e., Player D. Given that many HMM-inference techniques utilize batch data, i.e., a full observation sequence $\{x_t\}_{t \in \mathcal{T}}$, we assume the attacker seeks to modify this information so that, when utilized by the decision maker, the resulting inference somehow benefits Player Z. Whereas the attacker knows the decision maker is utilizing an HMM, they are uncertain of its exact parameterization (i.e., in the terms of Biggio and Roli (2018), we consider a grey-box setting) and describe their beliefs probabilistically in a Bayesian manner. Moreover, the attacker's attempts at data corruption are subject to error and are not guaranteed to be successful. Corruption attacks are also accompanied by an associated risk of discovery, implying that Player Z must balance the risks and rewards of their attacks. For this initial research, the decision maker is assumed to utilize the standard HMM inference procedures discussed in Section 2; that is, as with many HMM applications currently in use, Player D has not hardened their algorithms against potential attacks.

Figure 3a provides a BAID which graphically depicts this interaction[2]. Akin to the real-world scenarios discussed by Krasser (2023), the attacker is assumed to have either infiltrated the decision maker's information system or controls the input data. They desire to use this access, along with intelligence about the form of $\mathcal{Q}$ and $\mathcal{X}$, to manipulate inference while maintaining data plausibility. The series of true latent states, $\{Q_t\}_{t \in \mathcal{T}}$, is unknown to each player; however, it affects the observations, $\{x_t\}_{t \in \mathcal{T}}$. Player Z is able to first view this information and, in accordance with their own self interest, attempt to corrupt it by altering observations. That is, if it will maximize their expected utility, Player Z may attempt to change a set of x_t-variables but, because these attacks are subject to error, the attack may not be entirely successful. Via Player Z's interaction with $\{x_t\}_{t \in \mathcal{T}}$, a new, potentially perturbed series of observations, $\{y_t\}_{t \in \mathcal{T}}$ is utilized by the decision maker (i.e., Player D) as the basis of their inference. Such attacks are particularly relevant when the environment and the sensors determine the emissions and latent variables (e.g., quality control or fault identification on assembly lines).

Should Player Z have perfect knowledge about the HMM parameterization, then the resulting HMM corruption problem would be deterministic from the attacker's perspective. Herein, we adopt an alternative approach that we contend is more realistic. Namely, an attacker can quite readily ascertain via nefarious means the class of machine learning algorithm utilized in application by a decision maker, but discovering its exact parameterization is a more difficult task. According to such uncertainties, Figure 3b presents an ARA reduction of the previously described BAID from the attacker's perspective. Player Z's beliefs about the decision maker's parameterization (i.e., $a_{i,j}$, $b_{i,k}$, and π_i) are encapsulated by the uncertainty node on Player D's decision.

---

[2]In select problem variants (e.g., distribution disruption), a *functional arc* between $\{X_t\}$ and u_Z may need to be considered. We discuss such problems subsequently.

(a) BAID for the HMM Corruption Problem   (b) Attacker Reduction of the HMM Corruption Problem
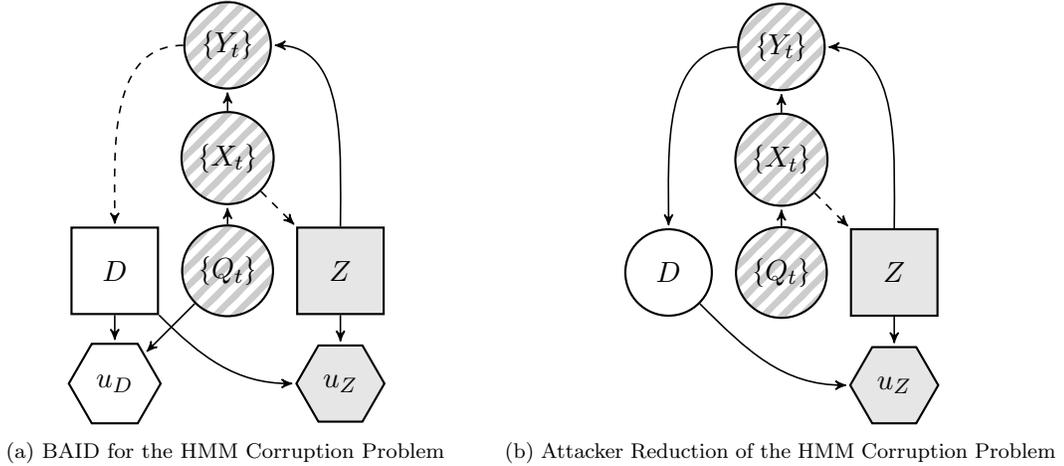
Figure 3: HMM Corruption Problem

Although Figure 3b provides an intuitive graphical means of depicting the attacker's problem, further notation is required to set forth an authoritative formulation. Table 1 details the additional notation leveraged within this section to do so. Therein, variable indices are represented at their finest level of granularity; however, hereafter, arrays of decision variables are denoted via the absence of a subscript (e.g., z_t contains $z_{t,k}$ over every $k \in \mathcal{X}$). The same convention is used for the probability matrices, e.g., row i of A is denoted as $a_i = (a_{i,1}, \ldots, a_{i,|\mathcal{Q}|})$.

Table 1: Summary of Additional Notation for HMM Corruption Problems

Notation	Definition
$u_Z(\cdot)$	Player Z's utility for a given set of decision variables and realized uncertainties
$z_{t,k}$	Binary decision variable equal to 1 if attacker inserts k at t, and 0 otherwise
w_1	Attacker's objective-function weight on decision maker's inference
w_2	Attacker's objective-function weight on data corruption costs
$\rho_{t,k}$	Random variable equaling 1 if insertion of k at t is successful, and 0 otherwise
y_t	Perturbed observation at t viewed by decision maker
$\alpha_{t,i}$	Standard forward probabilities calculated with $\{y_t\}_{t \in \mathcal{T}}$
$\beta_{t,i}$	Standard backward probabilities calculated with $\{y_t\}_{t \in \mathcal{T}}$
$\delta_{t,i}$	Standard Viterbi probabilities calculated with $\{y_t\}_{t \in \mathcal{T}}$
P_ρ	Joint probability mass function of ρ-variables having support $\mathcal{P}$
g_A	Joint density over entries in A having support $\mathcal{A}$
g_B	Joint density over entries in B having support $\mathcal{B}$
g_π	Dirichlet density over entries in π having support Π
g_ω	Joint density over all random variables having support Ω

3.1. Corrupting Filtering and Smoothing Inference

The attacker's problem described previously is simultaneously stochastic, combinatorial, and nonlinear. These characteristics are therefore present in Problem **P1** which represents an attacker with general, multi-objective utility over the corruption of a decision maker's filtering or smoothing distributions at time t'. When encountered with such a problem, the attacker should solve

$$\textbf{P1} : \max_{z} \quad u_Z(z, \alpha_{t'}, \beta_{t'}) = \mathbb{E}\left[w_1 f_1(\alpha_{t'}, \beta_{t'}) - w_2 f_2(z)\right]$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{X}} z_{t,k} = 1, \ \forall t \in \mathcal{T}, \tag{2a}$$

$$\alpha_{1,i} = \pi_i \left(\sum_{k \in \mathcal{X}} z_{1,k}(b_{i,k}\rho_{1,k} + b_{i,x_1}(1 - \rho_{1,k})) \right), \ \forall i \in \mathcal{Q}, \tag{2b}$$

$$\alpha_{t,i} = \sum_{j \in \mathcal{Q}} \alpha_{t-1,j} a_{j,i} \left(\sum_{k \in \mathcal{X}} z_{t,k}(b_{i,k}\rho_{t,k} + b_{i,x_t}(1 - \rho_{t,k})) \right), \ \forall t \in \mathcal{T}\backslash\{1\}, \ i \in \mathcal{Q}, \tag{2c}$$

$$\beta_{|\mathcal{T}|,i} = 1, \ \forall i \in \mathcal{Q}, \tag{2d}$$

$$\beta_{t-1,i} = \sum_{j \in \mathcal{Q}} \beta_{t,j} a_{i,j} \left(\sum_{k \in \mathcal{X}} z_{t,k}(b_{j,k}\rho_{t,k} + b_{j,x_t}(1 - \rho_{t,k})) \right), \ \forall t \in \mathcal{T}\backslash\{1\}, \ i \in \mathcal{Q}, \tag{2e}$$

such that

$$\rho_{t,k} \sim \mathscr{B}(\lambda_k), \forall t \in \mathcal{T}, \ k \in \mathcal{X},$$
$$a_i \sim \mathscr{D}(\xi_i), \forall i \in \mathcal{Q},$$
$$b_i \sim \mathscr{D}(\zeta_i), \forall i \in \mathcal{Q},$$
$$\pi \sim \mathscr{D}(\upsilon),$$

where $\mathscr{B}(\cdot)$ and $\mathscr{D}(\cdot)$ are shorthand for the Bernoulli and Dirichlet distributions characterized by the provided parameters. We note that attack success, i.e., $\rho_{t,k}$, is stationary across t but may vary in k. Similarly, uncertainty over the transition and emission probabilities may vary in i. For notational convenience, the joint probability mass function of all ρ-variables is denoted by P_ρ over support $\mathcal{P}$, the joint density of the transition matrix rows is g_A with support $\mathcal{A}$, the joint density of the emission matrix rows is g_B with support $\mathcal{B}$, and the Dirichlet distribution over π is denoted by g_π with support Π. The joint density of all random variables is denoted by g_ω with support $\Omega = \mathcal{P} \times \mathcal{A} \times \mathcal{B} \times \Pi$.

The objective function of Problem P1 is the expected value of a linear combination of $f_1(\alpha_{t'}, \beta_{t'})$ and $f_2(z)$ whereby the former captures the attacker's utility from the decision maker's inference and the latter represents the utility associated with their corruption decisions. That is, the weighted-sum method from multicriteria optimization is leveraged (Ehrgott, 2005). Constraint (2a) ensures that the attacker only selects a single $k \in \mathcal{X}$ to insert into the perturbed observation vector at each time. Note that, if $x_t = k$ and

$z_{t,k} = 1$, the attacker is not altering x_t and $y_t = k$ with certainty, i.e., this observation is not being changed. Alternatively, Constraints (2b)–(2e) ensure the correct calculation of both the forward and backward probability recursions. Although visually distinct, these constraints are conceptually similar to the equations provided in Section 2.1. The summand in (2b)–(2e) over $k \in \mathcal{X}$, ensures that the forward (backward) probabilities are calculated correctly with the realized $\{y_t\}_{t \in \mathcal{T}}$. Moreover, provided a realized set of attack-success outcomes (i.e., $\rho_{t,k}$), this sequence of perturbed observations can readily be calculated via

$$y_t = \sum_{k' \in \mathcal{X}} z_{t,k'} \left(\rho_{t,k'} k' + (1 - \rho_{t,k'}) x_t \right), \ \forall t \in \mathcal{T}.$$

From inspecting Problem P1, its difficulty becomes immediately apparent. By expanding the recursions in Constraint (2c) and (2e), it can be observed that the resulting functions are nonlinear for any non-trivial $\mathcal{T}$; both the forward and backward probabilities expand to, potentially, high-order polynomial functions. Adding further complexity is the fact that all of the parameters in Constraints (2b)–(2e) are unknown and subject to probabilistic uncertainty. Furthermore, whereas Constraint (2a) is deterministic and linear, it highlights the combinatorial nature of the attacker's problem. This complexity may be further exacerbated by the specific functional form of the attacker's multi-objective utility function.

Herein, we explore a subset of the most interesting utility functions. Multitudinous options exist to parameterize the component utility function $f_2(z)$. One straightforward and flexible option is

$$f_2(z) = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{X}: x_t \neq k} z_{t,k}. \tag{3}$$

This simple alternative penalizes any corruption conducted by the attacker, and ensures this risk is appropriately represented in the attacker's utility function by its product with w_2. Higher-order polynomials in the $z_{i,k}$-variables could also be leveraged if the Player Z's attack costs are not constant. A similar diversity of functional representation is associated with $f_1(\alpha_{t'}, \beta_{t'})$. Subsequent sections explore this dynamic in greater detail whereby we provide alternative structures according to varying attacker intentions.

Given that Problem P1 contains unknown parameters described probabilistically, it is clearly a stochastic programming problem. The recursions within Constraint (2b) – (2e) are defined to ensure clarity of communication and highlight the problem's relationship to canonical HMM algorithms but are not strictly necessary. That is, the recursion may be written explicitly in the objective function.

3.1.1. State-Attraction and State-Repulsion Problems

In some settings, the attacker may only be interested in the decision maker's conditional distribution over a latent state at a particular time t'. For example, if the latent state represented the attacker's position and they wish to evade detection by the decision maker, it is in the attacker's best interest to minimize the decision maker's belief about

this true position. Similarly, it is conceivable that an attacker may wish to lure the decision maker into believing a specific latent state i' occurred at time t'. In either setting, these objectives can be modeled using

$$f_1(\alpha_{t'}, \beta_{t'}) = c\left(\frac{\alpha_{t',i'}\beta_{t',i'}}{\sum_{j \in \mathcal{Q}} \alpha_{t',j}\beta_{t',j}}\right)$$

where $c = 1$ in a state-attraction problem and $c = -1$ in a state-repulsion problem. Therefore, the attacker will maximize or minimize, respectively, the smoothing probability provided in Equation (1).

This component utility function is associated with an attacker who is concerned with a single latent state. While useful, this may not always properly characterize the attacker's intent. Some attackers may wish to alter the decision maker's inference over multiple latent states, e.g., potentially the entirety of $\mathcal{Q}$ as presented subsequently.

3.1.2. Distribution-Disruption Problems

The attacker may be interested in disrupting the decision maker's beliefs at time t' across $\mathcal{Q}$ to the maximum extent possible. More specifically, the attacker may wish to maximize the distance between the filtering (or smoothing) distributions under $\{x_t\}_{t \in \mathcal{T}}$ and $\{y_t\}_{t \in \mathcal{T}}$. Numerous options exist to characterize the distance between the uncorrupted and corrupted distributions, e.g., the Kullback-Leibler divergence and the Hellinger distance (Cha, 2007). Generally speaking, a distribution-disruption problem can be formulated by setting $f_1(\cdot)$ equal to the desired distance measure.

Let $\hat{\gamma}_{t,i}$ represent probability of state i at some time t given the uncorrupted data, $\{x_t\}_{t \in \mathcal{T}}$. Using this notation, the Kullback-Leibler divergence between the distributions induced by $\{x_t\}_{t \in \mathcal{T}}$ and $\{y_t\}_{t \in \mathcal{T}}$ is

$$f_1(\alpha_{t'}, \beta_{t'}) = \sum_{i \in \mathcal{Q}} \hat{\gamma}_{t',i}\left(\log(\hat{\gamma}_{t',i}) - \log\left(\frac{\alpha_{t',i}\beta_{t',i}}{\sum_{j \in \mathcal{Q}} \alpha_{t',j}\beta_{t',j}}\right)\right).$$

The above can be used to identify a maximally perturbed distribution akin to an ill-performing M-projection; however, the order of the distributions may be reversed in the Kullback-Leibler divergence to identify a modified I-projection (Koller and Friedman, 2009) as well. Alternatively, if the Hellinger distance is utilized, then one may set

$$f_1(\alpha_{t'}, \beta_{t'}) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i \in \mathcal{Q}}\left((\hat{\gamma}_{t',i})^{1/2} - \left(\frac{\alpha_{t',i}\beta_{t',i}}{\sum_{j \in \mathcal{Q}} \alpha_{t',j}\beta_{t',j}}\right)^{1/2}\right)^2}.$$

Other statistical distances calculated via an optimization step may also be used (e.g., the Wasserstein distance or the Kolmogorov–Smirnov statistic); however, when adopting such

11

an approach, Problem P1 may need to be augmented with the requisite, distance-specific constraints. Nevertheless, we henceforth utilize the Kullback-Leibler divergence in our exploration, but such a selection is merely a matter of attacker preference.

3.2. Corrupting Decoding Predictions

The probabilistic dynamics of the underlying HMM suggests that, when corrupting decoding predictions, the attacker is once more confronted with a non-linear, combinatorial, stochastic problem. Problem **P2** presents a mathematical programming model for an attacker trying to disrupt a decision maker inferring the most likely sequence of latent states given $\{y_t\}_{t \in \mathcal{T}}$.

$$\textbf{P2}: \max_z \quad u_Z(z, \delta) = \mathbb{E}\left[w_1 f_1(\delta) - w_2 f_2(z)\right] \tag{4a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{X}} z_{t,k} = 1, \ \forall t \in \mathcal{T}, \tag{4b}$$

$$\delta_{1,i} = \pi_i \left(\sum_{k \in \mathcal{X}} z_{1,k}(b_{i,k}\rho_{1,k} + b_{i,x_1}(1 - \rho_{1,k})) \right), \ \forall i \in \mathcal{Q}, \tag{4c}$$

$$\delta_{t,i} = \max_{j \in \mathcal{Q}} \delta_{t-1,j} a_{j,i} \left(\sum_{k \in \mathcal{X}} z_{t,k}(b_{i,k}\rho_{t,k} + b_{i,x_t}(1 - \rho_{t,k})) \right), \ \forall t \in \mathcal{T} \backslash \{1\}, \ i \in \mathcal{Q}. \tag{4d}$$

where

$$\rho_{t,k} \sim \mathscr{B}(\lambda_k), \forall t \in \mathcal{T}, \ k \in \mathcal{X},$$
$$a_i \sim \mathscr{D}(\xi_i), \forall i \in \mathcal{Q},$$
$$b_i \sim \mathscr{D}(\zeta_i), \forall i \in \mathcal{Q},$$
$$\pi \sim \mathscr{D}(\upsilon).$$

As with Problem P1, Problem P2's objective function is the expected value of a linear combination of two component utility functions. The cost of an attack vector, $f_2(z)$, may be of a similar form as those discussed for Problem P1. However, since the decision maker is assumed to be solving a decoding problem, the attacker's component utility on their inference, $f_1(\delta)$, differs from Problem P1. The storage of forward and backward probabilities is no longer strictly required, and the Viterbi probabilities, $\delta_{t,i}$, are computed instead. However, in so doing, the constraints of Problem P2 are further complicated via the inclusion of a maximum operator, i.e., Constraint (4d).

Given a realized set of uncertainties (i.e., ρ, A, B, π), the most-likely state sequence from the decision maker's perspective can be calculated via backtracking. However, for notational simplicity, the requisite back pointers that store the states maximizing Constraint (4d) are excluded from Problem P2; they are not necessary to identify the attacker's optimal solution for the following problems.

12

3.2.1. Path-Attraction and Path-Repulsion Problems

As with Problem P1, there exist multiple component utility functions that may be used for $f_1(\delta)$. However, we are primarily concerned herein with path-attraction and path-repulsion problems. In a path-attraction problem, the attacker wishes to encourage the decision maker to believe in a particular sequence of latent states. This is accomplished by setting

$$f_1(\delta) = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{Q}} c_{t,i} \delta_{t,i}$$

such that $c_{t,i} > 0$ for states the attacker would like to encourage at each $t \in \mathcal{T}$. Alternatively, a path-repulsion problem is characterized by $c_{t,i} < 0$. The attacker may identify these sequences of latent states exogenously or base them off their expectation of $\{q_t\}_{t \in \mathcal{T}}$ under the true data $\{x_t\}_{t \in \mathcal{T}}$. Likewise, it is straightforward to combine these two frameworks to simultaneously attract and repulse a subset of latent state sequences. By their nature, the $\delta_{t,i}$-variables decrease in t, implying that, unless offset by increasing $c_{t,i}$-values, latent states visited later in the sequence would become relatively less important. If such an effect is not desirable, an effective method is to set $c_{t,i} = (\sum_{j \in \mathcal{Q}} \delta_{t,j})^{-1}$ for any state i the attacker wishes to encourage and set $c_{t,i} = -(\sum_{j \in \mathcal{Q}} \delta_{t,j})^{-1}$ for any state i the attacker wishes to discourage. Notably, by working as such with the $\delta_{t,i}$-probabilities, the attacker does not necessarily induce the Viterbi algorithm to output the desired path; however, it is an approximation that is clearly associated with such induction.

4. Solution Methodologies

The HMM structure utilized by the decision maker presents multiple, simultaneous difficulties. In the filtering, smoothing, and decoding settings, the attacker is presented with a non-linear objective function. Moreover, this difficulty is compounded by a set of decision variables of combinatorial cardinality, as well as a lack of knowledge that induces a stochastic programming problem. Of the commercial optimization solvers available, only global solvers are capable of accepting such problems as input; however, as shown by Caballero et al. (2018), even these methods are not foolproof in complex situations such as the one described in this paper. Finally, as a stochastic programming problem, solving Problems P1 and P2 coincides with maximizing an expectation, but its non-linearity limits the usefulness of canonical methods (e.g., sample-average approximation).

It is therefore apparent that customized solution techniques are required for Problems P1 and P2. Given the success of standard HMM algorithms, their modification to our perturbed setting would be ideal. Unfortunately, the relative efficiencies of the forward, backward, and Viterbi algorithms draw from their dynamic-programming structure, and this structure is dependent upon having observed a sequence of observations (e.g., $\{x_t\}_{t \in \mathcal{T}}$) upon which to base calculations. While desirable, it is unclear how (or if), the dynamic-programming structure of the aforementioned algorithms can be adapted to simultaneously optimize the attacker's actions and calculate the HMM's joint probabilities.

The tractable identification of an optimal attack vector is, at this juncture, a dubious prospect; therefore, we set forth alternative means the attacker may leverage to maximize their expected utility based on their subjective beliefs.

Within this section, we provide customized algorithms for solving variants of Problems P1 and P2, with minor modifications. The first solution method is a heuristic inspired by the *ranking-and-selection* problem (Powell, 2019). The second is an augmented-probability-simulation technique rooted in inhomogeneous Markov chain Monte Carlo sampling. The third is a Monte-Carlo enumeration technique having complete and random-greedy variants. When allocated enough computational resources, the complete variant will converge to the optimal solution of Problems P1 or P2. Based on this property, this algorithm serves as the benchmark solution method. Notably, these attacks may be used for each of the problems from Section 3 with minor modifications. Likewise, they are applicable to alternative prior distributions to those assumed in the previous section; an analytic prior is not even necessary, as long as the attacker can sample from the associated distribution.

4.1. Ranking-and-Selection Heuristic

Once the attacker has probabilistically codified their beliefs about the decision maker's problem, Problems P1 and P2 can be recast via the universal canonical model set forth by Powell (2019). More specifically, the HMM corruption problem is conceptualized as a variant of the ranking-and-selection problem whereby the attacker is asked to estimate the optimal $z^* \in \mathcal{Z} = \{z : z_{t,j} \in \{0,1\}, \forall (t,j) \in \mathcal{T} \times \mathcal{X}; \sum_{j \in \mathcal{Q}} z_{t,j} = 1, \forall t \in \mathcal{T}\}$ after having run N experiments according to some policy η. These experiments are simulations of potential realities based upon the attacker's beliefs about attack success and the HMM parameterization. Traditional ranking-and-selection (R&S) problems consider a finite action space of relatively small cardinality, thereby enabling estimates of this expectation to be stored tabularly after each experiment n. In juxtaposition, the action-space cardinality may be exceedingly large in our HMM corruption problems. Therefore, in what follows, we utilize a function-approximation approach to estimate an action's expected value. This method is highly flexible, allowing for use of various combinations of function-approximation methods and combinatorial-optimization routines.

Powell's universal canonical model requires the following elements be defined. Sequences of experimental values and estimates are denoted by n superscripts to distinguish them from the temporal, HMM sequences denoted by t subscripts.

- *State Variables*: The system state prior to conducting experiment n is denoted by θ^n, and determines the attacker's beliefs about each action's expected value. These beliefs are codified via a functional approximation $\hat{\mu}(z^n|\theta^n)$, i.e., θ^n are the parameters of the attacker's functional approximation prior to experiment n. The exact structure of θ^n depends upon the approximation utilized; however, under any setting, the set of all possible state variables is denoted by Θ.

- *Decision Variables*: For each experiment $n = 1, ..., N$, the attacker selects $z^n \in \mathcal{Z}$.

- *Exogenous Information*: Given some attack vector z^n, the attacker's payoff is determined by the decision maker's inference on $\{y_t\}_{t\in\mathcal{T}}$ which is in turn affected by the realized exogenous information, i.e., the random ρ-, A-, B-, and π-variables. An instance of this exogenous information during experiment n is referred to as $\omega^n \in \Omega$.

- *Objective Function*: For Problems P1 and P2, the attacker's objective is to maximize their expected reward by selecting an experimental policy η that respectively solves

$$\max_{\eta} \mathbb{E}\left[u(Z^\eta(\theta^{N+1}), \alpha_{t'}, \beta_{t'})\right]$$

and

$$\max_{\eta} \mathbb{E}\left[u(Z^\eta(\theta^{N+1}), \delta)\right]$$

- *System Model*: Given that the system state, θ^n, denotes the parameters underpinning $\hat{\mu}(z^n|\theta^n)$, the system model, $S^M(\theta^n, z^n, \omega^n)$, corresponds to the sequential update of θ^n after each experiment, i.e., $S^M : \Theta \times \mathcal{Z} \times \Omega \to \Theta$. The system model reflects the sequential optimization of the θ^n parameters to θ^{n+1} after each experiment n. Example system models θ^n-updates include stochastic gradient descent and recursive least squares, among others.

Given this formulation, Algorithm 4 provides an overarching R&S heuristic that is applicable to either problem under any of the attacker utility functions presented in Section 3. The attacker must input their beliefs about the associated uncertainities (i.e., g_ω), their initial functional-approximation parameterization (i.e., θ^1), and the number of experiments they wish to conduct (i.e., N). The R&S heuristic concludes by outputting a recommended attack, $\hat{z}^*$. Note that Step 6 may be omitted for most problems; it is only required in select settings, e.g., distribution-disruption problems.

Algorithm 4 Ranking-and-Selection Heuristic

1: **Input**: g_ω, θ^1, N
2: **Output**: $\hat{z}^*$
3: **for** $n = 1, \ldots, N$ **do**   ▷ May alternatively use clock-time limit
4:   Select $z^n = Z^\eta(\theta^n)$
5:   Sample $\omega^n \sim g_\omega$
6:   Solve decision maker's HMM problem with $\{x_t\}_{t\in\mathcal{T}}$ ▷ If required
7:   Use ω^n and z^n to determine $\{y_t\}_{t\in\mathcal{T}}$
8:   Solve decision maker's HMM problem with $\{y_t\}_{t\in\mathcal{T}}$ and ω^n, and compute $u_Z(\cdot)$
9:   Update $\theta^{n+1} = S^M(\theta^n, z^n, \omega^n)$
10: **end for**
11: Select $\hat{z}^* = Z^\eta(\theta^{N+1})$
12: **return** $\hat{z}^*$

The flexibility of the formulation allows for the construction of multitudinous R&S-heuristic variants. The policy η may adopt multiple structural forms (i.e., see Powell,

2019). Greedy or ε-greedy policies on $\hat{\mu}(z^n|\theta^n)$ are sensible alternatives prevalent in the R&S literature; however, since $|\mathcal{Z}|$ can inhibit the use of complete enumeration techniques, it may be necessary to identify the greedy policy via integer programming or analogous heuristic methods (e.g., genetic algorithms). Moreover, as with canonical reinforcement learning methods, the function approximation, $\hat{\mu}(z^n|\theta^n)$, may be based upon any statistical, regression model (e.g., regression trees, neural networks, etc.) with a correspondingly tailored system model. For example, if $\hat{\mu}(z^n|\theta^n)$ is a linear regression model then, akin to Powell (2007), it is sensible to utilize recursive least squares as a system model; however, if $\hat{\mu}(z^n|\theta^n)$ is a neural network then system model updates via stochastic gradient descent are more applicable[3]. Ultimately, the attacker is likely best served by tuning these qualitative hyperparameters via some sort of experimental design, e.g., see Jenkins et al. (2021).

4.2. Augmented Probability Simulation

The problems described in Section 3 can each be recast as indentifying a solution to

$$\max_z \sum_\rho \int_{\mathcal{A}} \int_{\mathcal{B}} \int_{\Pi} u_Z(z,\phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho) \, \mathrm{d}\pi \, \mathrm{d}B \, \mathrm{d}A \tag{5}$$

where $\phi = \Phi(z, A, B, \pi, \rho)$ is determined by z, A, B, π, and ρ, and represents the relevant set of parameters and decision variables for the associated problem (e.g., $\phi = (\alpha_{t'}, \beta_{t'})$ for Problem P1).

One possible means to solve this problem consists of leveraging simulation-based techniques, such as the augmented probability simulation (APS) methods of Bielza et al. (1999). This requires defining an augmented distribution on the product space of attacks and uncertainties of the form

$$\breve{g}(z, A, B, \pi, \rho) \propto u_Z(z, \phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho). \tag{6}$$

This distribution is well-defined provided that the utility is positive and bounded for all z and ϕ, which can be achieved by scaling the utility function appropriately.

It is straightforward to see that the global mode of the marginal of $\breve{g}(z, A, B, \pi, \rho)$ in z, coincides with the solution of Equation (5). Therefore, if we sample $(z, A, B, \pi, \rho) \sim \breve{g}(z, A, B, \pi, \rho)$, the sample mode of z approximates the optimal solution. However, as acknowledged by Müller et al. (2004), identifying the mode of the z samples may be challenging, especially when z is high dimensional. This difficulty is most apparent when the marginal augmented distribution is characterized by flat regions around its global mode. Both of these conditions are likely to emerge in our setting, suggesting an alternative to standard APS methods may be required.

---

[3]In Section 5, we illustrate two R&S variants based on neural networks, that differ in identification of the greedy action, i.e. Monte Carlo tree search and simulated annealing.

Following procedures set forth by Müller et al. (2004), mode identification can be facilitated by defining an alternative augmented distribution. More specifically, an augmented distribution must be defined such that its marginal in z is proportional to $\left[\sum_\rho \int_\mathcal{A} \int_\mathcal{B} \int_\Pi u_Z(z,\phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho) \, d\pi \, dB \, dA\right]^H$ where $H \geqslant 1$ is called the augmentation parameter. Since the associated, marginal-augmented distribution is more peaked, its mode is more easily identified. Therefore, we define an augmented distribution by creating H copies of our unknown parameters. This set of copies is denoted by $\{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}$ such that $\mathcal{H} = \{1, 2, ..., H\}$. The augmented distribution is given by

$$\breve{g}_H(z, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \prod_{h\in\mathcal{H}} u_Z(z, \phi^h) g_A(A^h) g_B(B^h) g_\pi(\pi^h) P_\rho(\rho^h). \qquad (7)$$

where $\phi^h = \Phi(z, A^h, B^h, \pi^h, \rho^h)$. It is straightforward to prove that the marginal of $\breve{g}_H(z, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}})$ in z is proportional to $\left[\sum_\rho \int_\mathcal{A} \int_\mathcal{B} \int_\Pi u_Z(z,\phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho) \, d\pi \, dB \, dA\right]^H$.

Direct sampling from the distribution in Equation (7) is not feasible, since the exact specification of $\breve{g}_H$ is costly. Instead, samples can be obtained using Markov-chain-Monte-Carlo (MCMC) methods (Tierney, 1994). In particular, we design a Metropolis-within-Gibbs sampling approach for use herein.

Implementing the Gibbs step requires generating samples from the full conditionals $\breve{g}_H(z_t | z_{-t} \{A, B, \pi, \rho\}_{h\in\mathcal{H}})$ where $z_{-t} = \{z_{t'}\}_{t'\in\mathcal{T}\setminus\{t\}}$, and to sample from the full conditionals for A^h, B^h, π^h and ρ^h for $h = 1, \ldots, H$. For communicative clarity, let us rewrite Equation (7) as

$$\breve{g}_H(z, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \exp\left\{ \sum_{h\in\mathcal{H}} \log[u_Z(z, \phi^h)] + \log[g_A(A^h)] + \log[g_B(B^h)] \right.$$
$$\left. + \log[g_\pi(\pi^h)] + \log[P_\rho(\rho^h)] \right\}.$$

In so doing, it can be readily observed that

$$\breve{g}_H(z_t | z_{-t}, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \exp\left(\sum_{h\in\mathcal{H}} \log\left[u_Z\left(z_t \cup z_{-t}, \phi^h\right) \right] \right).$$

This is simply the *softmax* distribution over $\sum_{h\in\mathcal{H}} \log[u_Z(z_t \cup z_{-t}, \phi^h)]$ for every possible value of z_t. Therefore, given z_{-t} and $\{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}$, sampling from the full conditional of z_t is straightforward.

Conversely, to sample from the full conditionals for A^h, B^h, π^h and ρ^h for $h = 1, \ldots, H$, we need to utilize the Metropolis algorithm. For example, noticing that the

full conditional for A^h is

$$\breve{g}_H \left(A^h | z, \{A^{h'}\}_{h' \in \mathcal{H} \setminus \{h\}}, \{B^{h'}, \pi^{h'}, \rho^{h'}\}_{h' \in \mathcal{H}} \right) \propto \exp \left(\log[g_A(A^{h'})] + \log \left[u_Z(z, \phi^h) \right] \right),$$

samples of $A^{h'}$ can be obtained via a Metropolis approach as follows. Assume the current state of the Markov chain is $z, A^h, B^h, \pi^h, \rho^h$ and $\phi^h = \Phi \left(z, A^h, B^h, \pi^h, \rho^h \right)$, then

1. Sample $\tilde{A}^h \sim g_A$.

2. Compute $\tilde{\phi}^h = \Phi \left(z, \tilde{A}^h, B^h, \pi^h, \rho^h \right)$

3. Accept $\tilde{A}^h$ with probability

$$\min \left\{ 1, \frac{u_Z(z, \tilde{\phi}^h)}{u_Z(z, \phi^h)} \right\}.$$

Samples from the conditionals in B^h, π^h, ρ^h can be obtained sequentially in a similar way. Moreover, samples from the full conditionals for A^h, B^h, π^h, ρ^h having different h-values can also be obtained in parallel. Sampling sequentially from these conditional distributions will asymptotically produce samples from $\breve{g}_H(z, \{A, B, \pi, \rho\}_{h \in \mathcal{H}})$.

However, as suggested by Müller et al. (2004) this sampling framework may result in the algorithm getting stuck in local modes, thereby inhibiting the identification of the global mode. This issue can be mitigated by combining this sampling scheme with an annealing schedule that iteratively increases H. This produces an inhomogeneous Markov chain whose limiting distribution is, under certain conditions (Müller et al., 2004), uniform over the set of optimal attacks. Algorithm 5 outlines our APS approximation method which utilizes the augmented distribution $\breve{g}_H$, the aforementioned Metropolis-within-Gibbs sampling approach, and an annealing schedule $\{\mathcal{H}_n\}_{n=1}^{\infty}$ of variable length; note that e_k denotes the standard basis vector in $\mathbb{R}^{|\mathcal{X}|}$ having a one in the position k and zero elsewhere. At each iteration n, a complete vector z^n is stored. The optimal attack vector $\hat{z}^*$ is estimated as the mode of these samples, excluding the burn-in period $n = 1, ..., n' - 1$. Since z is a discrete random variable, it is often useful to approximate this mode for large $\mathcal{T}$, e.g., by estimating the mode of each z_t independently, to avoid excessively large N-values.

To guarantee convergence to the optimal attack(s), the inhomogeneous MCMC simulation needs to be designed so that the stationary distribution of the Markov chain for a fixed H is precisely $\breve{g}_H$. Following well-established literature of MCMC (Tierney, 1994), it is straightforward to see that the limiting distribution of the Metropolis-within-Gibbs Markov chain defined in Algorithm 5 for each $h \in H_n$ is precisely $\breve{g}_h$. For further information regarding APS, we refer the interested reader to Müller et al. (2004) for theoretical proofs of the algorithm's convergence, and to Ekin et al. (2014) and Ekin et al. (2022) for augmented probability simulation applications in decision-theoretic settings.

Algorithm 5 APS Approximation

1: **Input**: g_A, g_B, g_π, P_ρ and $\{\mathcal{H}_n\}_{n=1}^\infty$, N
2: **Output**: $\hat{z}^*$
3: Initialize: z_t for $t = 1, \ldots, \mathcal{T}$
4: Set $n = 1$
5: **while** $n < N$ **do**   ▷ May alternatively use clock-time limit
6:   Sample A^h, B^h, π^h, ρ^h using the Metropolis step $\forall h \in \mathcal{H}_n$
7:   **for** $t = 1, 2, \ldots, \mathcal{T}$ **do**
8:     For $h \in H_n$ and $k \in \mathcal{X}$, determine $\phi^h := \Phi\left(e_k \cup z_{-t}\right), A^h, B^h, \pi^h, \rho^h\right)$
9:     Sample $\tilde{z}_t$ from *softmax* distribution described in the Gibbs step
10:    Update $z_t = \tilde{z}_t$.
11:   **end for**
12:   Set $z^n = \{z_t\}_{t \in \mathcal{T}}$
13:   Set $n = n + 1$
14: **end while**
15: Estimate $\hat{z}^*$ as the sample mode of $\{z^n : n \geqslant n'\}$
16: **return** $\hat{z}^*$

4.3. Monte-Carlo Enumeration

Monte-Carlo sampling techniques are popular solution approaches for standard ARA problems; they serve to numerically approximate otherwise intractable expectation functions. Akin to the proposed R&S heuristic, our complete Monte-Carlo enumeration (CME) method leverages samples from g_ω to inform better estimates of the utility of taking each $z \in \mathcal{Z}$; however, the sampling methods utilized are distinct. Both solution approaches randomly sample the attack effects and the decision maker's HMM parameterization (i.e., $\omega^n \sim g_\omega$), but they differ in their evaluation of corruption attacks. Whereas the R&S heuristic selects a single $z \in \mathcal{Z}$ to evaluate on ω^n according to η, the CME technique evaluates every $z \in \mathcal{Z}$ on each ω^n. In so doing, the latter method is able to garner more information from each ω^n, but at the expense of greater computational effort. Psuedocode for this solution method is provided in Algorithm 6. The algorithm is applicable to any of the problems presented in Section 3, but Step 5 is only strictly required in a distribution-disruption problem. This approach serves as a baseline for the other algorithms discussed herein.

To ensure termination on sufficiently large instances, Algorithm 6 can be modified to form a random-greedy variant. random-greedy Monte-Carlo enumeration (RME) approach selects an incumbent attack at random and calculates its expected utility. The expected utility of another random attack is then calculated and, if improved, this new attack becomes the incumbent solution. The algorithm terminates when no improvement is identified. While CME may be effective for solving small-size instances, RME is more scalable to larger instances.

Finally, we note that the CME approach is a simpler alternative to the APS scheme. However, several issues are anticipated to arise for larger-sized instances. The most significant concern arises when the maximization of the expected utility, approximated

---
Algorithm 6 Complete Monte-Carlo Enumeration

1: **Input**: g_ω, N
2: **Output**: $\hat{z}^*$
3: **for** $n = 1, \ldots, N$ **do**
4:     Sample $\omega^n \sim g_\omega$
5:     Solve decision maker's HMM problem with $\{x_t\}_{t \in \mathcal{T}}$ ▷ If required
6:     **for** $z \in \mathcal{Z}$ **do**
7:         Use ω^n and z to determine $\{y_t\}_{t \in \mathcal{T}}$
8:         Solve decision maker's HMM problem with $\{y_t\}_{t \in \mathcal{T}}$
9:         Set $\hat{u}^{n,z} = u_Z^n(\cdot)$
10:     **end for**
11: **end for**
12: Set $\bar{u}^z$ to the sample average of $\hat{u}^{n,z}$, $\forall z \in \mathcal{Z}$
13: Identify $\hat{z}^* = \arg\max_z \bar{u}^z$
---

through Monte Carlo sampling, becomes challenging due to the flatness of the approximation with respect to z. In such cases, numerical errors inherent in the Monte Carlo approximation of the expected utility may overshadow a relatively small difference in expected utility between the optimal and inferior attacks. In juxtaposition, the APS scheme transforms the simulation-optimization problem into a grand simulation problem using a series of augmented probability models that become more peaked around the optimal attack.

5. Testing, Results, and Analysis

This section analyzes the effects of HMM corruption on a decision maker's inference and examines the efficacy of the developed heuristics to conduct such attacks. This is accomplished via four blocks of experimentation. Our focus in Section 5.1 is to determine the degree to which data corruption can negatively affect a decision maker's inference. Therein, we explore the effect of varying objective-function weights on the attacker's behavior and illustrate the devastating effects that relatively low-perturbation data-corruption attacks may generate. Analysis within this section also reveals that, although the Monte Carlo enumeration algorithm can identify high-quality solutions, it is computationally infeasible for larger-sized HMMs. Therefore, in Sections 5.2 and 5.3 we provide detailed analyses of the use of our R&S heuristic and the APS approximation routine. Section 5.2 analyzes the effect of the HMM structure on each algorithm's performance, whereas Section 5.3 focuses on the impact of uncertainty on solution quality. Section 5.4 provides a case study illustrating the practical relevance of our attacks by attacking an HMM used for part-of-speech tagging.

While the case study leverages a standard laptop for computation to demonstrate real world application feasibility, the remainder of testing was performed using relatively powerful machines. This enabled parallelization and multiple runs of each attack for improved statistical analysis. Such testing was performed with the LOVELACE High

Performance Computing (HPC) infrastructure housed at the Institute of Mathematical Sciences of the Spanish National Research Council (ICMAT-CSIC). In particular, 12 nodes were leveraged for experimentation, each equipped with 187GB of RAM and two 32-core, 2.30GHz Intel(R) Xeon(R) processors. For additional information regarding the HPC cluster layout and the computational resources available therein, we refer the interested reader to ICMAT-CSIC (2022).

5.1. Efficacy of HMM Corruption

The purpose of this section is to illustrate the adverse effects of data corruption on HMM inference and showcase how this relates to the attacker's objectives and knowledge. More specifically, we demonstrate that, even under substantial uncertainty about the decision maker's HMM, the methods developed herein can devastate inference quality. Moreover, we also explore the effectiveness of the CME attack and its limitations with respect to instance size. We examine each problem-and-objective-function combination with this attack by utilizing a modestly sized HMM for tractable illustration. Whereas in-depth testing of the R&S and APS solution approaches are performed in subsequent sections, this section also demonstrates the marginal effect of allocating additional computational resources for each attack. In so doing, we highlight that our attacks can expeditiously identify quality solutions.

Assume that Player D's true HMM is defined by

$$A_D = \begin{bmatrix} 0.85 & 0.05 & 0.1 \\ 0.05 & 0.9 & 0.05 \\ 0.5 & 0.25 & 0.25 \end{bmatrix},$$

$$B_D = \begin{bmatrix} 0.699 & 0.05 & 0.1 & 0.05 & 0.1 & 0.001 \\ 0.001 & 0.1 & 0.1 & 0.299 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.2 & 0.1 & 0.3 \end{bmatrix},$$

$$\pi_D = \begin{bmatrix} 0.5 & 0.3 & 0.2 \end{bmatrix},$$

such that there are three possible latent states and six possible emissions at each state. This precise parameterization is unknown to Player Z; however, we assume that the attacker's beliefs are correct in expectation (i.e., $\mathbb{E}[A] = A_D$). The attacker is also assumed to believe that the probability of a successful attack on an observation is constant. In this manner, we define two distinct uncertainty levels upon which to test our attacks. The lower uncertainty condition is characterized by a Dirichlet precision (i.e., κ)[4] of 10,000 and a constant attack-success probability of 0.95. Conversely, the high uncertainty condition corresponds to a Dirichlet precision of 100 and an attack-success probability of 0.75. Assuming $\mathcal{T} = \{1, ..., 5\}$, Player Z observes the true emissions $X = \{5, 4, 6, 4, 5\}$ and wishes to thwart the decision maker's inference in accordance with their own self-interest.

---

[4]Recall that the precision of a Dirichlet distribution equals the sum of its parameters. These values are varied under the constraint that, in expectation, Player Z's beliefs coincide with Player D's true parameterization.

We examine Player Z's behavior in the state-attraction, state-repulsion[5], distribution-disruption, and path-attraction problems under the two aforementioned uncertainty levels and varying ratios of w_1/w_2. For the state-attraction problem, Player Z desires Player D to believe $Q_3 = 1$, whereas in the state-repulsion problem, Player Z desires Player D not to believe $Q_3 = 2$. In the distribution-disruption problem, Player Z wants to maximally perturb Player D's smoothing distribution at $t' = 3$ while, in the path-attraction problem, Player Z wishes the decision maker to infer $Q = \{3, 1, 1, 1, 3\}$ as the most probable sequence of hidden states. The attacker aims to maximize their expected utility in these settings; however, to highlight the effect of HMM corruption for each problem, distinct performance measures are reported. More specifically, performance measures for the state-attraction, state-repulsion, distribution-disruption, and path-attraction problems are respectively set as Player D's subjective belief of $Q_3 = 1$, Player D's subjective belief of $Q_3 = 2$, the Kullback Leibler divergence between the true and corrupted smoothing distributions, and the normalized Hamming distance[6].

We approximate the effect of $\hat{z}^*$ on these metrics for each problem-and-uncertainty combination as a function of w_1/w_2. An N-value of 10,000 was utilized to estimate $f_1(\cdot)$ for each attack alternative and problem type. These estimates were subsequently used to estimate the expected utility of every attack for each w_1/w_2. The attack alternatives that maximize the expected utility were selected as $\hat{z}^*$-variables. Once identified, the attack $\hat{z}^*$ was simulated M times to generate an approximate distribution over the performance metrics. The value of M was selected to ensure an appropriate level of precision for the mean performance in each problem. This required $M = 5,000$ simulations for the state-attraction and distribution-disruption problems, but $M = 1,000$ sufficed for the state-repulsion and path-attraction problems. Within each problem, the associated M-value is kept constant across uncertainty levels to ensure proper comparisons.

Figure 4 summarizes the results of this testing for each problem-and-uncertainty pair. Within each plot, the mean performance measures over the M samples are represented by a blue and black line for the low- and high-uncertainty levels, respectively; $\pm 2 \left(s/\sqrt{M} \right)$ confidence regions are shaded grey. It can be observed that, in any problem when $w_1/w_2 \to 0$, the attacker determines the benefits of corrupting the data are outweighed by its costs. The attacker chooses to keep $\{x_t\}_{t \in \mathcal{T}}$ unchanged and, in so doing, concedes undesirable inference behavior by Player D. For example, in the state-repulsion problem, when $w_1/w_2 \to 0$, the probability of $Q_3 = 2$ is approximately 0.95, i.e., the probability inferred by the decision maker using the true observations. However, as w_1/w_2 increases and relative corruption costs decrease, this balance shifts and Player Z begins to corrupt the data. Notably, across all plots, it is interesting to note that step-function-like behavior is induced. This behavior can also be observed by inspecting $\hat{z}^*$ at each w_1/w_2 for every problem. That is, $\hat{z}^*$ tends to stay constant across an interval of ratios until the

---

[5]The state-attraction and -repulsion problems are, structurally speaking, the same problem. However, we solve both herein to explore how varying objective-function parameterizations may affect the attacker's solution. For brevity, we exclude such exploration of the remaining problems.

[6]The normalized Hamming distance between two strings with the same length is the number of positions at which the corresponding symbols are different, divided by the strings' length.

reward of attacking some x_t is worth the penalty, at which point a new optimal attack is determined. The result of this behavior is that the expected value of the performance measures stays constant over the domain wherein $\hat{z}^*$ is constant.
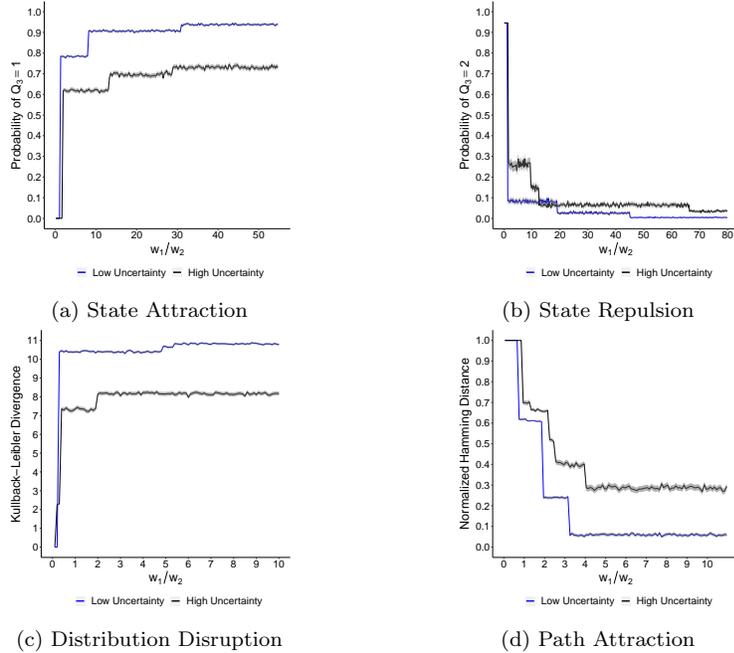

(a) State Attraction

(b) State Repulsion

(c) Distribution Disruption

(d) Path Attraction

Figure 4: Ratio Plots for each Problem-and-Uncertainty Pair

Inspection of Figure 4 reveals that distinct values of w_1/w_2 are necessary to induce modified attacker behavior. For example, in the state-attraction problems, Player Z is incentivized to maximally perturb the observation vector at $w_1/w_2 \approx 30$, but this behavior is not induced in the path-attraction problems until $w_1/w_2 > 3$. By inspecting each subfigure in Figure 4, once can discern how the uncertainty levels affect the attacker's behavior. Notably, within each problem, the high-uncertainty level alters the range of w_1/w_2-values in which a $\hat{z}^*$ is estimated to be optimal and, in expectation, tends to induce a less-preferable, maximally perturbed outcome. Likewise, tighter confidence regions about the attack's expected value can be derived under low uncertainty than high uncertainty.

The plots in Figure 4 are also interesting in that, once w_1/w_2 is sufficiently far from zero, a dramatic increase in the performance measure of each problem is observed. The degree of improvement is problem dependent (e.g., compare the behavior of Subfigures 4c and 4d near $w_1/w_2 = 0$), but the pattern is consistent. Nevertheless, this behavior naturally leads one to inquire about the nature of the requisite attacks, i.e., the degree to which they perturb $\{x_t\}_{t\in\mathcal{T}}$. Fortunately, the size of the HMM examined in this section allows us to readily examine this in greater detail.

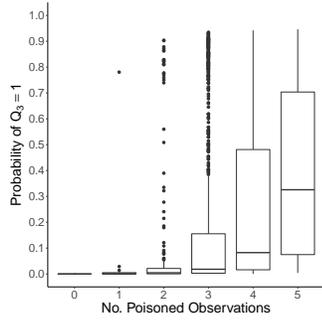We perform additional analysis on the attacker's problems by enumerating the effects

of every $z \in \mathcal{Z}$; M simulations of each attack z are performed to estimate its mean performance for the aforementioned problem types. The box plots in Figure 5 examine to what degree attacks of differing perturbation levels (i.e., the number of corrupted observations) affect the decision maker's inference. Examination of these plots yields noteworthy insights. The attacker could considerably disrupt Player D's inference by corrupting only a single observation in some cases, assuming they have the means to readily identify it. For example, in the case of low uncertainty, corrupting one observation could change Player D's subjective probability from ≈ 0 to 0.8 in the state-attraction problem while a similar change from ≈ 0.95 to 0.1 is observed in the state-repulsion problem. Such modest attacks are not quite as successful in the distribution-disruption or path-attraction problems, but high-quality results can still be achieved without corrupting the entirety of $\{x_t\}_{t \in \mathcal{T}}$. Moreover, it can also be observed in the variability of each subfigure that high-perturbation attacks are not necessarily effective; in fact, the variability in an attack's success increases with the number of corrupted observations. Although these results collectively highlight the disruption that HMM corruption can induce on a decision maker's inference, they also emphasize the difficulty of the attacker's problems. This is most apparent by considering the medians and inter-quartile ranges in Figure 5. Effective attacks that limit the number of corrupted observation are the exception, not the rule. Conversely, blindly corrupting a large number of observations has no guarantee of success; such attacks have better median performance but are highly variable.

Finally, perhaps the most noteworthy result in this section pertains to the performance of the CME algorithm. Despite the difficulty of the examined problems, this solution methodology was readily able to identify high-quality solutions. Furthermore, it is also worth emphasizing that, whereas the attacks are built under both aleatoric and epistemic uncertainty (i.e., the unknown attack success and HMM parameters, respectively), the evaluations provided herein are performed under aleatoric uncertainty only. This fact further emphasizes the utility of our methodology. Despite not knowing the outcome of an attack or the decision maker's true HMM at design time, our attacks were able to substantially thwart Player D's inferences.
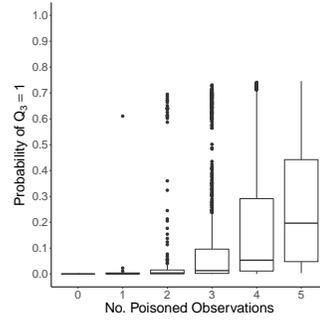
Such a result bodes well for the real-world applicability of such corruption attacks. The attacker is unlikely to know the true HMM, but the methods presented herein allow to identify high-quality attacks despite this limitation. Unfortunately, although the CME method is effective, additional experimentation found it too computationally burdensome for larger instances. Given some w_1/w_2, the problems examined within this section can generally be solved within 30-55 minutes using the aforementioned hardware but, for larger-sized HMMs having a $\mathcal{T}$ of greater cardinality, the required computational effort rapidly increases. This motivates the utilization of other methods, e.g., the R&S and APS methods discussed in the next two subsections.

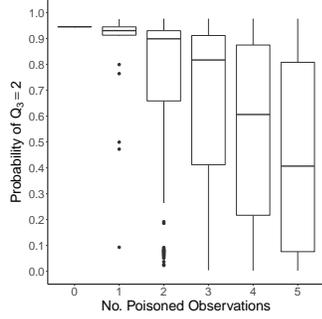*5.2. Effect of HMM Structure on Solution Method Performance*

This section focuses on the effect of the HMM structure on the R&S and APS methods. Following the best practices set forth by Coffin and Saltzman (2000), we consider the 2_{III}^{3-1} design presented in Table 2 upon which to test algorithm performance.
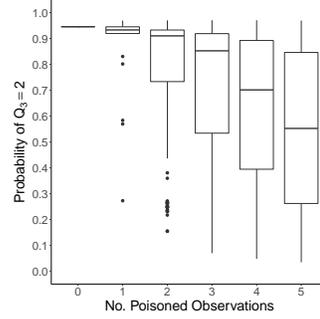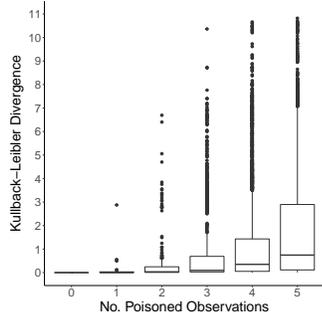
(a) State-Attraction, Low Uncertainty
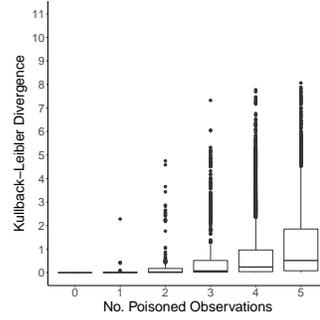
(b) State-Attraction, High Uncertainty

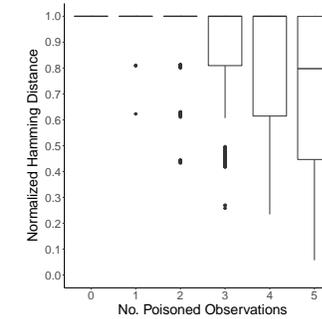(c) State-Repulsion, Low Uncertainty
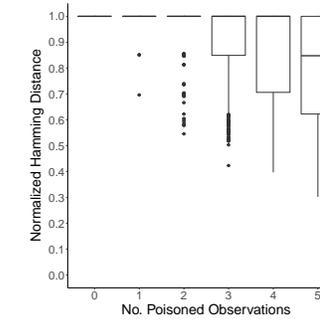
(d) State-Repulsion, High Uncertainty

(e) Distribution-Disruption, Low Uncertainty

(f) Distribution-Disruption, High Uncertainty

(g) Path-Attraction, Low Uncertainty

(h) Path-Attraction, High Uncertainty

Figure 5: Mean Attack Efficacy by Number of Perturbed Observations

The distributions defining the decision maker's true HMM parameters (e.g., A_D) are generated randomly via a Dirichlet distribution with all concentration parameters set to one. The true observations are generated randomly from a discrete uniform distribution with support $\{1, 2, ... |X|\}$, whereas $\lambda = 0.95$ and $\kappa = 10,000$ for each. Table 3 depicts the attacker's goals for each problem, and the objective function weight ratios, w_1/w_2, are selected to correspond with a conservative attacker who is unwilling to corrupt all observations. The complete parameterizations are available online. [7]

Table 2: 2_{III}^{3-1} Design For HMM Structure Testing

Design Point	$|Q|$	$|X|$	$|T|$
1	30	30	30
2	10	10	30
3	10	30	10
4	30	10	10

Table 3: Attacker's Objective Function Parameters by Problem and Design Point

Problem	w_1/w_2	Design Point			
		1	2	3	4
State-Attraction	20	$\{t' = 25, i' = 4\}$	$\{t' = 21, i' = 9\}$	$\{t' = 9, i' = 7\}$	$\{t' = 9, i' = 22\}$
State-Repulsion	15	$\{t' = 23, i' = 9\}$	$\{t' = 29, i' = 4\}$	$\{t' = 5, i' = 6\}$	$\{t' = 9, i' = 12\}$
Distribution-Disruption	3	$t' = 23$	$t' = 29$	$t' = 5$	$t' = 9$
Path-Attraction	2.55	$\{0\}_{\forall t \in \mathcal{T}}$			

Two variants of each of the R&S and APS methods are tested against the RME algorithms; the CME approach is ignored because empirical testing found it to be overly encumbered for instances of this size. The two R&S methods are henceforth referred to as R&S-A and R&S-B. Both utilize a multi-layer perceptron neural network as $\hat{\mu}(\cdot)$ to regress the attack's expected utility, but differ in their use of Monte-Carlo tree search and simulated annealing to identify a greedy action, respectively. Further details on the implementation of these methods, as well the hyperparameter tuning that informed them, is provided in Appendix A. Likewise, the APS techniques are referenced as APS-A and APS-B; the approaches have annealing schedule of $\{\mathcal{H}_n\}_{n=1}^{\infty}$ and $\{\mathcal{H}_n\}_{n=500}^{\infty}$, respectively. These schedules result in APS-A exploring the solution space more thoroughly than APS-B which instead favors exploitation.

We explore the efficacy of these attacks both in terms of solution quality and computational effort. This is accomplished by allowing each algorithm to operate for $\{15, 30, 45, ..., 1200\}$ seconds on every problem instance; the expected utility of the output attack at each time limit is calculated. Due to the stochastic nature of the attacks and the solution approaches, this procedure is replicated 10 times for each algorithm;

---

[7]Available at https://github.com/roinaveiro/corrupting_hmms

mean expected utilities across these repetitions plus/minus two standard deviations are reported.

Several trends emerged from this analysis across each problem type. This behavior is typified within the path-attraction instances as provided in Figure 6. Notably, APS-B and R&S-B converge toward attacks having comparable expected utilities across all instances and, although not a firm rule, APS-B tends to do so quicker than R&S-B. Alternatively, APS-A and R&S-A are more variable. For instances defined by smaller $|\mathcal{T}|$ (i.e., Design Points 3 and 4) they also converge toward high-quality solutions but, for larger $|\mathcal{T}|$-values, they do not identify valuable attacks. The computational time required to identify such solutions is variable as well. In juxtaposition, RME consistently identifies lower-quality attacks in every instance. Although not depicted herein, similar trends are also apparent across the other three problem types.
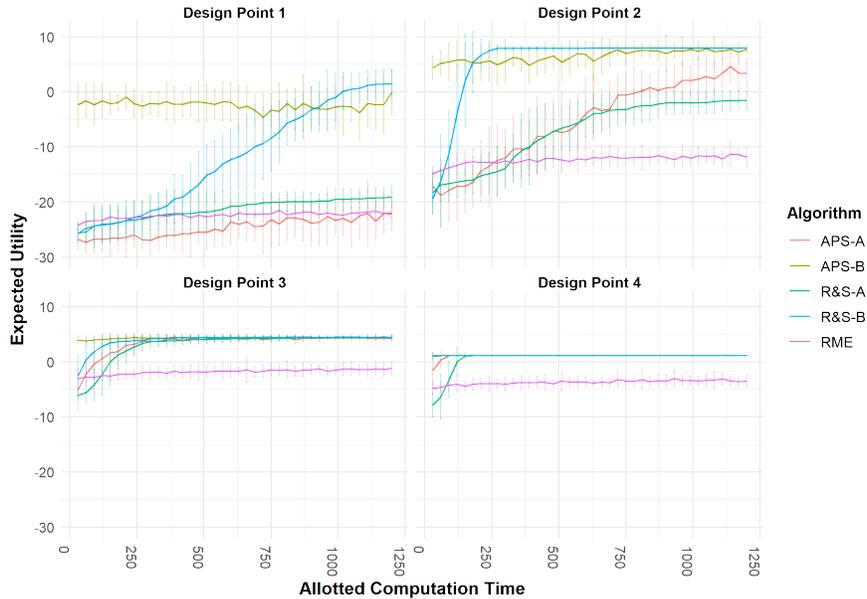

Figure 6: Evolution of Path-Attraction Expected Utilities over time by Algorithm

Despite their global similarities, it is interesting to note that the algorithms vary systematically in the type of attacks they identify. Table 4 compares and contrasts each algorithm with respect to the performance of their final attacks, i.e., those identified at 1200 seconds, across all problem-and-design-point pairs. Performance is based on the respective objective function of the problem of interest. To account for the stochasticity in attack success, solution quality was estimated by simulating their effects 500 times for the state-attraction and distribution-disruption instances, and 100 times for the state-repulsion and path-attraction instances; averages are reported across all repetitions. The impact measure is problem-specific and refers to damage caused to the decision maker's inference, whereas Δ refers to the total number of observations attacked. The greatest impact and the least $\Delta/|\mathcal{T}|$ achieved by the algorithms are bolded for each

problem-and-design-point pair.

Inspection of this table reveals additional patterns obfuscated by the expected-utility calculations. APS-B and R&S-B are better able to balance the attacker's multi-objective utility function; however, in so doing, they often find less damaging attacks to the decision maker's inference than their competitors. In particular, of the 16 problem-and-design-point pairs, the APS-B and R&S-B algorithms identified the most impactful attack two and four times, respectively. The expected impact varied substantially about problem types as well. R&S-B never identified the most impactful distribution-disruption attack; APS-B never did so for the state-attraction problem. However, the APS-B attacks appear to perturb relatively few perturbations in comparison to their impact. For specific problem-and-design-point pairs, the other three methods identified more impactful attacks. This success, however, was generally counterbalanced by higher rates of data perturbation. The APS-A approach is an exception to this rule in that it at times acts too conservatively by perturbing relatively few observations.

Table 4: Mean Impact and Proportion of Observations Attacked (Structure Testing)

Problem	Algorithm	Design Pt. 1		Design Pt. 2		Design Pt. 3		Design Pt. 4	
		Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$
State Att.[1]	APS-A	0.144	0.583	0.040	0.030	0.635	0.310	0.124	0.130
	APS-B	0.136	**0.043**	0.009	**0.007**	0.636	**0.300**	0.120	**0.120**
	R&S-A	0.169	0.773	0.038	0.517	**0.637**	0.420	**0.128**	0.280
	R&S-B	**0.179**	0.327	0.012	0.010	0.627	0.340	0.125	0.130
	RME	0.125	0.830	**0.074**	0.687	0.605	0.960	0.127	0.580
State Rep.[2]	APS-A	0.173	0.540	0.502	**0.033**	0.568	**0.100**	0.168	**0.100**
	APS-B	0.172	**0.037**	0.509	**0.033**	0.563	**0.100**	**0.170**	**0.100**
	R&S-A	0.174	0.740	0.500	0.490	0.566	**0.100**	0.166	0.180
	R&S-B	**0.177**	0.163	**0.515**	0.047	0.566	0.140	0.169	**0.100**
	RME	0.167	0.810	0.493	0.680	**0.577**	0.710	0.167	0.520
Dist. Disrupt.[3]	APS-A	2.221	0.653	2.202	**0.033**	**1.445**	0.210	**1.520**	**0.100**
	APS-B	2.140	**0.033**	2.213	**0.033**	1.435	**0.200**	1.518	**0.100**
	R&S-A	**2.283**	0.747	**2.297**	0.510	1.436	0.370	1.459	0.200
	R&S-B	2.206	0.453	2.279	0.327	1.426	0.290	1.511	**0.100**
	RME	2.203	0.877	2.192	0.687	1.270	0.830	1.517	0.550
Path Att.[4]	APS-A	0.980	0.817	**0.851**	0.190	**0.612**	0.340	**0.900**	**0.000**
	APS-B	0.940	0.080	0.860	**0.010**	0.707	**0.280**	**0.900**	**0.000**
	R&S-A	0.962	0.740	0.857	0.483	0.648	0.520	0.921	0.150
	R&S-B	**0.930**	**0.027**	0.867	0.017	0.620	0.360	**0.900**	**0.000**
	RME	0.972	0.813	0.897	0.690	0.718	0.700	0.923	0.460

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'

[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'

[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data

[4] Impact is the Normalized-Hamming distance of most-likely state sequences under perturbed data and attacker's goal

Collectively, these results illustrate that the APS and R&S algorithms can effectively thwart larger-scale HMM instances. The APS-B and R&S-B configurations were the most effective for the instances explored herein, but their performance varied. This variation was especially apparent across $|\mathcal{T}|$-values, a fact deriving from the exponential effect $|\mathcal{T}|$ has on the attack space's cardinality. Nevertheless, many of the results discussed herein

are specific to the parameterization utilized within this section. Alternative uncertainty structures about the decision maker's HMM may also play a significant role in the algorithm's efficacy. Exploring such dynamics is the focus of the next subsection.

5.3. Effect of Uncertainty on Solution Method Performance

The same algorithm variants studied in the previous section are utilized herein to explore the effect of uncertainty on their performances. The HMM size is fixed at $|\mathcal{Q}|$, $|\mathcal{X}|$, and $|\mathcal{T}| = 20$; however, the attacker's beliefs and the attack's success probability is varied via the 2^2 factorial design provided in Table 5. The decision maker's true HMM parameters are built in a similar manner to Section 5.2 sampling from Dirichlet distributions; the true observations are likewise built by sampling from discrete uniform distributions. The w_1/w_2-values for each problem are maintained constant from Section 5.2, as is the attacker's goal in the path-attraction problem. However, their goals in the state-attraction, state-repulsion and distribution-disruption instances correspond to $\{t' = 17, i' = 4\}$, $\{t' = 16, i' = 7\}$, and $t' = 16$, respectively. The complete parameterization of these instances is available online[8].

Table 5: 2^2 Full Factorial Design for Uncertainty Testing

Parameter	Design Points			
	1	2	3	4
λ	0.95	0.95	0.75	0.75
κ	10,000	100	10,000	100

Testing that is similar to the previous sub-section was conducted on each algorithm for every problem-and-design-point pair. Figure 7 provides the algorithms' performance in relation to the allocated computation time for the distribution-disruption problem. These results bear resemblance to those of the previous sub-section in the APS-B and R&S-B tend to find comparable solutions, but with APS-B doing so more expeditiously. The RME method again underperforms with respect to its peers. Nevertheless, the performance of APS-A is distinct in these instances from the previous section. It tends to converge toward comparable solutions to that found by APS-B and R&S-B; however, the variability about its expected utility is great, indicating highly variable performance. As with the previous sub-section, similar trends are also apparent across the other three problem types.

Table 6 depicts the mean performance of each algorithm for the final attacks across the four design points. These values are calculated by averaging the attacks' performances over quantities referenced in Section 5.2. Despite the varied uncertainty, APS-B continues to effectively balance perturbed observations with impact, and is able to identify high-quality but low-cardinality attacks. Likewise, as in the structural testing, the RME algorithm finds high-perturbation attacks of varied impact. In juxtaposition, the two R&S approaches only identify the most impactful attack once out of the sixteen combinations.

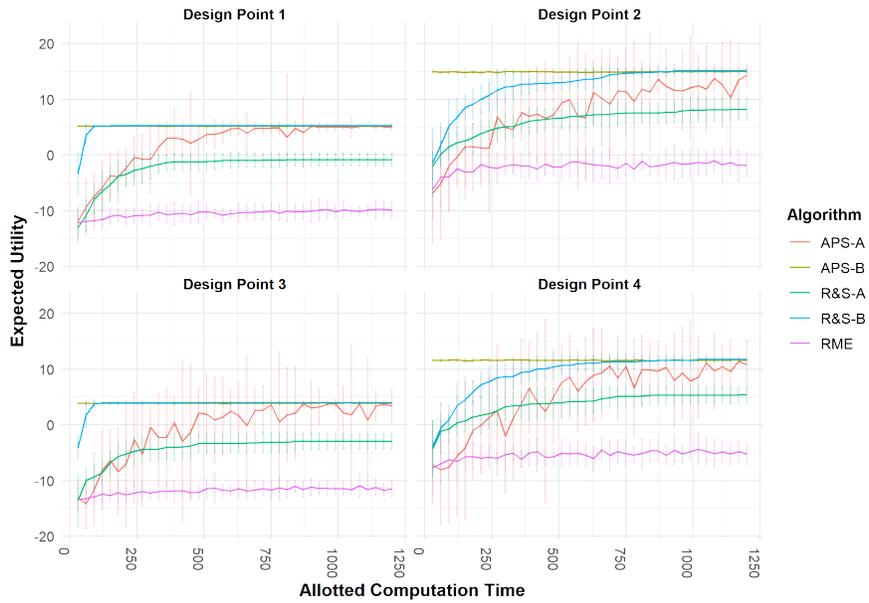[8]Available at https://github.com/roinaveiro/corrupting_hmms

Figure 7: Temporal Evolution of Distribution-Disruption Expected Utility by Algorithm

The APS-A algorithm performs much better across the uncertainty levels than the structural level; its impact is often among the top three of the examined attacks. Finally, no attack generates substantial impact on the path-attraction problem since the w_1/w_2-value examined makes the cost of the attack less than its reward.

Table 6: Mean Impact and Proportion of Observations Attacked (Uncertainty Testing)

Problem	Algorithm	Design Pt. 1		Design Pt. 2		Design Pt. 3		Design Pt. 4	
		Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$	Impact	$\Delta/|\mathcal{T}|$
State Att.[1]	APS-A	0.255	**0.145**	0.254	0.140	0.121	**0.085**	0.121	0.080
	APS-B	**0.266**	0.150	0.247	0.135	**0.133**	0.090	0.127	**0.085**
	R&S-A	0.171	0.625	0.162	0.550	0.108	0.570	0.116	0.625
	R&S-B	0.216	0.275	**0.259**	0.145	0.122	0.345	**0.133**	0.090
	RME	0.167	0.830	0.156	0.810	0.079	0.755	0.077	0.765
State Rep.[2]	APS-A	0.328	**0.050**	0.319	**0.050**	0.261	**0.050**	0.252	**0.050**
	APS-B	**0.329**	**0.050**	**0.330**	**0.050**	**0.266**	**0.050**	0.259	**0.050**
	R&S-A	0.318	0.470	0.322	0.570	0.252	0.505	0.249	0.560
	R&S-B	0.324	0.260	0.327	0.220	0.258	0.265	**0.266**	0.230
	RME	0.313	0.755	0.308	0.755	0.246	0.725	0.235	0.750
Dist Disrupt.[3]	APS-A	**2.171**	0.075	**1.394**	0.090	1.629	0.075	**1.156**	0.070
	APS-B	2.057	**0.050**	1.364	**0.050**	1.603	**0.050**	1.082	**0.050**
	R&S-A	2.153	0.525	1.364	0.595	1.622	0.595	1.071	0.615
	R&S-B	2.104	0.225	1.352	0.185	**1.677**	0.295	1.089	0.185
	RME	2.154	0.820	1.387	0.900	1.478	0.795	1.104	0.900
Path Att.[4]	APS-A	**0.941**	0.045	**0.940**	0.105	**0.942**	0.065	**0.945**	0.060
	APS-B	0.950	**0.000**	0.950	**0.000**	0.950	**0.000**	0.950	**0.000**
	R&S-A	0.942	0.365	0.954	0.325	0.955	0.090	0.952	0.490
	R&S-B	0.951	0.190	0.964	0.295	0.954	0.045	0.955	0.145
	RME	**0.941**	0.750	0.953	0.750	0.947	0.770	**0.945**	0.750

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'
[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'
[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data
[4] Impact is the Normalized-Hamming distance of most-likely state sequences perturbed data and attacker's goal

The overall performance of the attacks appears to be influenced most by λ, i.e., the probability of an attack changing the true observation, in the state-attraction and -repulsion problems. However, this pattern does not necessarily hold in the other problems. Within the distribution-disruption instances, κ has the most influence, whereas both λ and κ have a marginal to nil effect on the path-attraction instances as a consequence of the high w_1/w_2-values. The APS-A, APS-B and R&S-B algorithms identify a high-quality attack in the state-attraction, state-repulsion, and distribution-disruption problems. However, the identified attacks balance impact and data perturbation differently between algorithms. The totality of these results illustrate that, in general, the trends identified in the previous sub-section with regard to algorithmic performance are valid with varied uncertainty levels while the improved performance of APS-A is an exception.

5.4. Case Study: Attacking an HMM for Part-of-Speech Tagging

To highlight the practical relevance of our methods, we consider attacks against a larger-scale HMM used for natural-language processing (NLP). In particular, we are motivated by part-of-speech (POS) tagging, a widely used approach in the lexical analysis of text data. POS tagging improves accuracy of text analysis by reducing the computational effort required for data processing and revealing the syntactic structure of sentences. Herein, hidden states correspond to a POS, and the observations are words.

Emission probabilities refer to the probability of a word given a POS, whereas transition probabilities capture POS sequencing. Attacks in this section are accomplished using an Apple M2 workstation with 8 GB of RAM and 8 CPU cores in order to demonstrate algorithm efficacy even by using a standard laptop.

Concretely, we train an HMM on the Named-Entity-Recognition dataset (Kaggle, 2023). The top-300 words in the database are considered, along with all 29 POSs, in the examination of 30-word text strings. The trained HMM is taken as the decision-maker's model having $|\mathcal{Q}| = 29$, $|\mathcal{X}| = 300$, $|\mathcal{T}| = 30$. As in previous sections, the attacker's prior beliefs are centered about the decision maker's true model. We work in a low uncertainty scenario. We implement R&S-B method for this case-study due to its better scalability with respect to $|\mathcal{X}|$.

Let the following pre-processed phrase from the NER data be the uncorrupted text:

> mr. said among countries six party talks (among china, south korea, japan, russia, united states north korea) north korea's nuclear program

Figure 8 examines the utility of attacking this phrase via the R&S-B method across the state-attraction, state-repulsion, distribution-disruption and path-attraction problems. Specific details regarding the associated POS information for the state-attraction and -repulsion problems, as well as the desired POS sequences for the path-attraction problem are available online in our code repository. Attack success probabilities are similar to the previous sections and are also detailed in the code repository. For each of the problems, we again vary the objective weights. We alternatively set (w_1, w_2) to $(1, 5)$ and $(2, 1)$ to explore its effect. These weights are selected to ensure problem difficulty; namely, we do not want the attacker to be incentivized to make drastic changes to the uncorrupted text. For each of the problem-and-weight combinations, the attack was allowed to search the solution space for 1500, 3000, 6000 and 9000 seconds. Each experiment was performed 10 times to enable estimation of the mean expected utility as well as its standard deviation.

Examination of Figure 8 reveals several noteworthy insights. For each of the ten R&S-B attacks performed on each problem-and-weight pair, a different $\hat{z}^*$ may be identified based on the stochasticity inherent in the simulated-annealing optimization. Inspection of Figure 8 confirms intuition that lower computation times result in attacks of lesser quality. As additional computation time is allocated, it is apparent that not only does the solution quality improve, but the reduction in variability suggest that the algorithms converge toward an attack across the ten runs. Alternatively, variability in the expected utility measurements is reduced when $(w_1, w_2) = (2, 1)$, implying that, even when allocated less computation time, the runs converge toward comparable solutions.

Additional information regarding the performance of the superlative attacks generated using 9000 seconds of computational effort is detailed in Table 7. The same performance metrics are tallied as in Table 6. Notably, whenever $w_2 > w_1$, it can be observed that the attack is minimally perturbing the true data with limited impact due to the higher weighted cost. More aggressive and impactful attacks are identified when $w_1 > w_2$. The totality of these results highlight that a conventional machine can effectively corrupt HMMs in a grey-box setting, even when the optimal attack minimally perturbs the uncorrupted data.
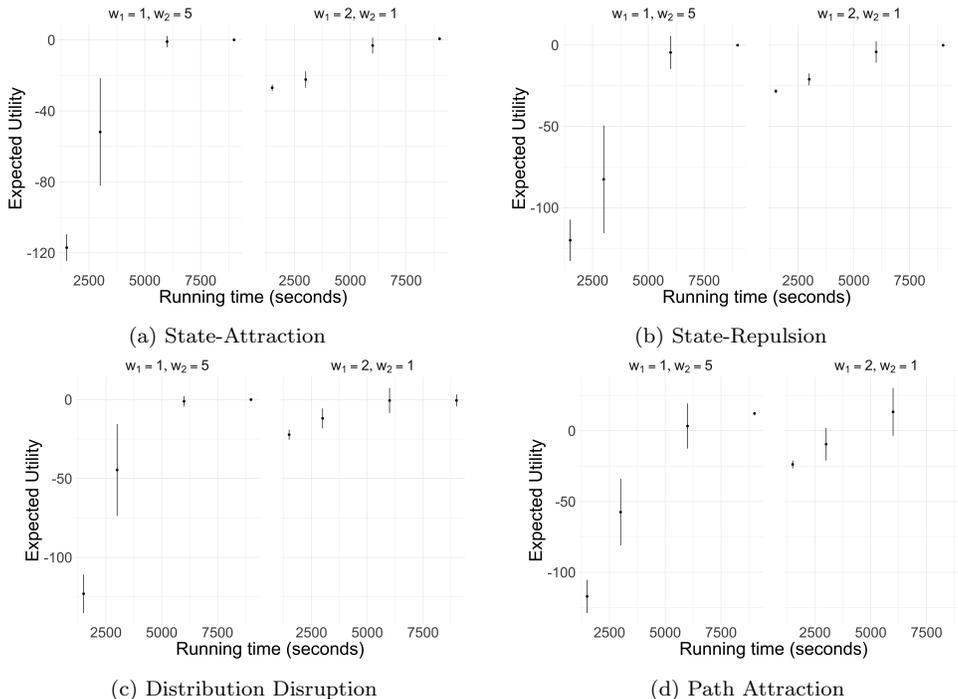
Figure 8: Expected Utility of Attacks Plus/Minus One Standard Deviation in the POS Case Study

Table 7: Mean Impact, Proportion of Observations Attacked and Expected Utility (NER Experiment)

Problem	w_1	w_2	Impact	$\Delta/|T|$	Expected Utility
State Attraction[1]	1	5	0.00	0.00	0.07
	2	1	0.77	0.13	0.68
State Repulsion[2]	1	5	0.00	0.00	-0.07
	2	1	0.00	1.0	-0.14
Distribution Disruption[3]	1	5	1.47	0.01	-0.03
	2	1	11.21	0.77	-0.59
Path Attraction[4]	1	5	0.53	0.00	12.17
	2	1	0.71	0.74	10.08

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'

[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'

[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data

[4] Impact is the Normalized-Hamming distance of most-likely state sequences perturbed data and attacker's goal

This feature is exemplified in the state-attraction case having $(w_1, w_2) = (2, 1)$ wherein the attacker wishes to encourage the classification of the fifth word as a singular proper noun. The superlative attack identified across all ten runs is listed below:

mr. said among countries u.s. party talks (among china, south korea, japan, russia, united states north korea) north korea's nuclear program

Notably, this text string is quite similar to the uncorrupted data. Their variations could easily be attributed to a typographical error or a bug in the text's pre-processing routine. However, this small change greatly increases the probability that the attacker achieves his goal; the attack increases the posterior probability of a singular-proper-noun by 0.77. The ability of this attack to thwart the decision maker's HMM with minimal changes to the true data is reminiscent of adversarial examples in other applications, e.g., computer vision (Goodfellow et al., 2014).

6. Conclusion

Probabilistic graphical models are fundamental to modern technology (Koller and Friedman, 2009). Diverse applications, from natural language processing to autonomous navigation, have all leveraged such models to achieve state-of-the-art results. Within this manuscript, we have illustrated that, like other machine learning methodologies, these models are susceptible to attack.

Focusing on HMMs, dynamic Bayesian networks with a specific structural form, we have formulated a suite of corruption problems for filtering, smoothing and decoding inferences. Leveraging an ARA perspective, a collection of general solution methods was also developed by alternatively viewing the problem from frequentist and Bayesian perspectives. Extensive empirical testing on these algorithms illustrated the devastating impacts of even minor data perturbations on HMM inferences. Moreover, this testing also examined the effects of HMM structure and uncertainty on the developed algorithms highlighting that, even for more complex instances, one may identify high-quality attacks in a reasonable amount of time. Our case study highlighted the real-world applicability of our attacks, implying that corrupted HMM models can create adversarial HMM examples akin to those that thwart computer vision algorithms (Goodfellow et al., 2014).

However, the development of these attacks begets numerous avenues of future inquiry. For example, given the vulnerability of HMMs to data perturbation, there is an obvious need for more robust HMM inference algorithms. The efficacy of the developed attacks on a trained HMM also suggests that traditional approaches to HMM learning may be vulnerable. Future research should therefore focus on the corruption of the Baum-Welch algorithm as well as on modifications to robustify it. The same variety of questions can also be formulated with respect to other probabilistic graphical models (e.g., Latent Dirichlet Allocation models). Notably, the problems and attacks provided herein can be directly extended to inference over alternative dynamic Bayesian networks (e.g., Kalman filters, autoregressive HMMs). Nevertheless, AML research upon standard HMMs is by no means exhausted either; varied assumptions developed herein can be relaxed to distinct, real-world settings (e.g., real-time inference or uncertain $\mathcal{Q}$ and $\mathcal{X}$). Moreover, given the generality of the HMM corruption problems, the variety of prior distributions available to the attacker, and the flexible hyperparameterization of the R&S heuristic, additional experimentation on our algorithms is a worthwhile endeavor, as is the development of alternative attacks. Therefore, while this research has incrementally developed AML techniques for probabilistic graphical models, avenues of future inquiry abound.

Acknowledgments

Appendix A. Implementation Details of the R&S Heuristic

This appendix provides implementation details of the R&S heuristics utilized in Sections 5.2 and 5.3. We discuss the functional approximation $\hat{\mu}(z^n|\theta^n)$, the associated state variables (i.e., θ^n), the system model (i.e., S^M), and the optimization routines used to select z^n given some parameterization of $\hat{\mu}(z^n|\theta^n)$. Associated code implementing the procedures discussed herein is available at https://github.com/roinaveiro/corrupting_hmms.

As a functional approximation of an attack's value, we use a fully connected neural network. Different architectures were tested and the superlative was selected, i.e., see Appendix A.3. The system state at each iteration n captures the current parameterization of this neural network, e.g., the arc weights. The updated state variables (i.e., θ^{n+1}) are identified using a stochastic-gradient-descent algorithm as the system model. In particular, the Adam optimizer is utilized. The performance of different learning rates is tested as well; see Appendix A.3 for additional details.

To iteratively select z^n, the R&S heuristic uses an ε-greedy policy to encourage exploration. Within each iteration of R&S, the greedy action that maximizes the objective function under our current belief state is of foremost interest. This requires solving a non-linear, integer optimization problem. Unfortunately, given the high cardinality of the attack space (i.e., $\mathcal{Z}$), solving this optimization via complete ennumeration is infeasible; other canonical branch-and-bound techniques are likely to struggle as well. Therefore, we compare the performance of two meta-heuristics to approximate this solution: Monte Carlo Tree Search (MCTS) and Simulated Annealing. Implementation details for both algorithms are provided subsequently in Appendix A.1 and Appendix A.2. Hyperparameter tuning is discussed in Appendix A.3.

Appendix A.1. Monte Carlo Tree Search

To find the greedy action using MCTS, we first reformulate this optimization problem as an undiscounted, sequential-decision problem wherein the greedy attack is constructed

iteratively by time period. With slight recycling of notation, this problem is formally defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{E}, \mathcal{R})$ wherein $\mathcal{S}$ is the set of all partially and fully filled attack vectors, $\mathcal{A}$ is the set of feasible actions at each $s \in \mathcal{S}$, $\mathcal{E}$ is a map from $\mathcal{S} \to \mathcal{S}$, and $\mathcal{R}$ is the set of instantaneous rewards at each $s \in \mathcal{S}$. At the beginning of each episode, we start with an empty attack vector (i.e., s_0). An episode concludes after the attack vector is fully specified. This is accomplished by selecting an emission for each time t from $\mathcal{A}(s_t)$ whereby s_t captures all emissions inserted into the attack up to this time. The transition function $\mathcal{E}$ deterministically maps a state-action pair (s_t, a_t) into the next state s_{t+1} by inserting the selected emission into the attack vector. The set of state rewards, $\mathcal{R}$, varies depending upon whether the state is complete or incomplete, i.e., whether emissions have or have not been selected for every $t \in \mathcal{T}$, respectively. The reward associated with any incomplete state is zero. However, the reward for complete states (i.e., fully specified attack vectors) corresponds to the evaluation of $\hat{\mu}(s|\theta^n)$.

Using this foundation, our MCTS algorithm constructs a tree comprised of all possible attack vectors. Its nodes correspond to some $\mathcal{S}' \subseteq \mathcal{S}$. At each iteration of the algorithm, the tree is traversed by selecting actions via a tree policy until a node with a child outside of the partially constructed tree is reached. If such a node is reached, actions are henceforth selected at random until a complete state is achieved. After reaching a complete state, its reward is computed and used to update the value of nodes that have been traversed in the tree such that better nodes are more likely to be selected in the future.

More specifically, given a partially explored tree $\mathcal{G}$, each MCTS iteration consists of four distinct steps:

1. *Selection.* Starting from the root node, (i.e., the empty attack vector, s_0), actions are selected using the tree policy until a node with some child outside the tree is reached. In our implementation, we use the UCT selection criterion as the tree policy's basis. That is, at state s_t, action a_t (leading to node $\mathcal{E}(s_t, a_t)$) is identified using

$$\pi^{\mathcal{G}}(s_t) = \arg\max_{a \in \mathcal{A}(s_t)} Q(s_t, a) + c \cdot \sqrt{\frac{2 \log N(s_t)}{N(s_t, a)}}$$

   where $Q(s_t, a)$ is a Monte Carlo estimate of the state-action value, $N(s_t)$ is the number of visits to the parent node, $N(s_t, a)$ is the number of times action a has been taken at node s_t and c is an exploration parameter. In our implementation, we fix $c = 0.5$.

2. *Expansion.* If a leaf node is reached in the selection step, it is appended to the tree and the simulation step begins.

3. *Simulation.* Actions outside the tree are selected uniformly at random from the available actions until a complete state has been reached.

4. *Backpropagation.* Upon reaching a complete state, its reward R is evaluated by invoking $\hat{\mu}(s|\theta^n)$. This is used to update the value of the leaf node reached in the

selection step as well as each of its parents. For every (s', a) along the traversed path in the tree, the following updates are performed:

$$
\begin{aligned}
N(s', a) &\leftarrow N(s', a) + 1 \\
Q(s', a) &\leftarrow Q(s', a) + \frac{R - Q(s', a)}{N(s', a)}
\end{aligned}
$$

In our implementation of the R&S heuristic, MCTS is used to approximate the greedy policy in each iteration of R&S (i.e., each time Step 4 is reached in Algorithm 4). In particular, we perform $1,000$ MCTS iterations and record the best complete state reached as the approximate greedy action.

Appendix A.2. Simulated Annealing

The second meta-heuristic used to find the action maximizing $\hat{\mu}(z^n | \theta^n)$ is simulated annealing (SA) (Kirkpatrick et al., 1983). SA generates a Markov chain in the space of possible attacks, whose stationary distribution is proportional to $\exp(\hat{\mu}(z^n | \theta^n)/T_e)$, where T_e is gradually decreased to 0 according to some pre-specified annealing schedule. We use Gibbs sampling to generate such a Markov chain, sequentially sampling from full conditional of the stationary distribution for each z_t^n conditioned on z_{-t}^n. T_e is reduced according to the exponential decay annealing schedule suggested by Spears (1993). Therefore, at the jth iteration of SA, we have $T_e = \exp(-l \cdot j/\mathcal{T})$, where l is set to 5.

Appendix A.3. Hyperparameter Tuning

This appendix explains how hyperparameter tuning was performed for the two R&S variants. Specifically, we varied the following hyperparameters: the architecture of the multilayer perceptron (MLP) neural network (i.e., the number of neurons in the two layers), the number of SA or MCTS iterations used to find the greedy action, the learning rate, and ε. Tables A.8 and A.9 lists the tested hyperparameter combinations.

Table A.8: Hyperparameter Combinations for the R&S-A Algorithm.

Combination	MLP Architecture	Iterations	Learning rate	ε
1	{16; 8}	100	0.005	0.05
2	{32; 16}	10	0.005	0.05
3	{16; 8}	100	0.1	0.005
4	{32; 16}	10	0.1	0.005
5	{64; 64}	100	0.005	0.05
6	{64; 64}	100	0.1	0.05

Table A.9: Hyperparameter Combinations for the R&S-B Algorithm.

Combination	MLP Architecture	Iterations	Learning rate	ε
1	{16; 8}	50	0.005	0.05
2	{32; 16}	10	0.005	0.05
3	{16; 8}	50	0.1	0.005
4	{32; 16}	10	0.1	0.005
5	{64; 64}	100	0.005	0.05
6	{64; 64}	100	0.1	0.05

An extensive grid search on the hyperparameters was not performed; instead, a simple search procedure was conducted to (1) examine how the algorithms behave under different hyperparameters and (2) identify settings that perform well on the examined instances. To do so, we replicated the testing procedure described in Sections 5.2 and 5.3 for each combination. Each algorithm-and-hyperparameter combination was tested 10 times for run times of $\{15, 30, 45, ..., 1200\}$ seconds on each problem instance. Mean expected utility values plus/minus two standard deviations were recorded for each run time across all 10 repetitions. Algorithm performance was judged by their convergence time, solution quality, and relative balance of these characteristics. A qualitative assessment of these measures was utilized for model selection.

Tables A.10 and A.11 indicate the superlative hyperparameter combinations for each problem-and-design-point pair. The indicated hyperparameter setting is leveraged within Sections 5.2 and 5.3, respectively. Notably, no single combination is dominant across all instances, but some combinations were always dominated by another. For example, combination four and three of the R&S-A and -B algorithms, respectively, are dominated in the Section 5.2 instances. Likewise, combination six is dominated across all problem-and-design-point pairs. Analogous patterns are apparent in the Section 5.3 testing as well.

Table A.10: Superlative Hyperparameter Combination for Section Structural Testing

Problem	Algorithm	Design Point			
		1	2	3	4
State-Attraction	R&S-A	2	3	2	2
	R&S-B	2	1	2	2
State-Repulsion	R&S-A	2	3	2	2
	R&S-B	2	1	2	5
Distribution-Disruption	R&S-A	2	5	2	1
	R&S-B	2	5	2	5
Path-Attraction	R&S-A	2	1	1	1
	R&S-B	4	1	2	2

Table A.11: Superlative Hyperparameter Combination for HMM Uncertainty Testing

Problem	Algorithm	Design Point			
		1	2	3	4
State-Attraction	R&S-A	3	3	5	5
	R&S-B	5	1	2	1
State-Repulsion	R&S-A	5	5	5	5
	R&S-B	2	2	2	2
Distribution-Disruption	R&S-A	5	2	5	2
	R&S-B	2	2	2	2
Path-Attraction	R&S-A	2	2	1	1
	R&S-B	4	4	4	4

References

Albrecht, S.V., Stone, P., 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence 258, 66–95.

Alfeld, S., Zhu, X., Barford, P., 2016. Data poisoning attacks against autoregressive models, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1452–1458.

Alhajjar, E., Maxwell, P., Bastian, N., 2021. Adversarial machine learning in network intrusion detection systems. Expert Systems with Applications 186, 115782.

Banks, D., Petralia, F., Wang, S., 2011. Adversarial risk analysis: Borel games. Applied Stochastic Models in Business and Industry 27, 72–86.

Banks, D.L., Aliaga, J.R., Insua, D.R., 2015. Adversarial Risk Analysis. CRC Press.

Bechhoefer, J., 2015. Hidden Markov models for stochastic thermodynamics. New Journal of Physics 17, 075003.

Bielza, C., Müller, P., Insua, D.R., 1999. Decision analysis by augmented probability simulation. Management Science 45, 995–1007.

Biggio, B., Roli, F., 2018. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition 84, 317–331.

Caballero, W.N., Jenkins, P.R., Keith, A.J., 2021. Poisoning finite-horizon Markov decision processes at design time. Computers & Operations Research 129, 105185.

Caballero, W.N., Kline, A.G., Lunday, B.J., 2018. Challenges and solutions with exponentiation constraints using decision variables via the BARON commercial solver, in: 2018 IISE Annual Conference Proceedings, pp. 1331–1336.

Caballero, W.N., Lunday, B.J., Uber, R.P., 2020. Identifying Behaviorally Robust Strategies for Normal Form Games under Varying Forms of Uncertainty. European Journal of Operational Research In press. doi:`https://doi.org/10.1016/j.ejor.2020.06.022`.

Cha, S., 2007. Comprehensive survey on distance similarity. Int. J. Math. Model. Methods Appl. Sci. 1.

Chen, Y., Zhu, X., 2020. Optimal attack against autoregressive models by manipulating the environment, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3545–3552.

Coffin, M., Saltzman, M.J., 2000. Statistical analysis of computational tests of algorithms and heuristics. INFORMS Journal on Computing 12, 24–44.

Crecchi, F., Melis, M., Sotgiu, A., Bacciu, D., Biggio, B., 2020. Fader: Fast adversarial example rejection. arXiv preprint arXiv:2010.09119 .

Crouse, M.S., Nowak, R.D., Baraniuk, R.G., 1998. Wavelet-based statistical signal processing using hidden Markov models. IEEE Transactions on signal processing 46, 886–902.

Dalvi, N., Domingos, P., Sanghai, S., Verma, D., 2004. Adversarial classification, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 99–108.

Dang-Nhu, R., Singh, G., Bielik, P., Vechev, M., 2020. Adversarial attacks on probabilistic autoregressive forecasting models, in: International Conference on Machine Learning, pp. 2356–2365.

Ehrgott, M., 2005. Multicriteria optimization. volume 491. Springer Science & Business Media.

Ekin, T., Naveiro, R., Insua, D.R., Torres-Barrán, A., 2022. Augmented probability simulation methods for sequential games. European Journal of Operational Research doi:`10.1016/j.ejor.2022.06.042`.

Ekin, T., Polson, N.G., Soyer, R., 2014. Augmented Markov chain Monte Carlo simulation for two-stage stochastic programs with recourse. Decision Analysis 11, 250–264.

Ernst, J., Kellis, M., 2012. ChromHMM: Automating chromatin-state discovery and characterization. Nature methods 9, 215–216.

Gales, M., Young, S., et al., 2008. The application of hidden Markov models in speech recognition. Foundations and Trends® in Signal Processing 1, 195–304.

Gallego, V., Naveiro, R., Insua, D.R., 2019. Reinforcement learning under threats, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9939–9940.

González-Ortega, J., Ríos Insua, D., Ruggeri, F., Soyer, R., 2021. Hypothesis testing in presence of adversaries. The American Statistician 75, 31–40.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 .

Gordillo, J., Conde, E., 2007. An HMM for detecting spam mail. Expert Systems with Applications 33, 667–682.

Hsu, C.Y., Chen, P.Y., Lu, S., Liu, S., Yu, C.M., 2021. Adversarial examples for unsupervised machine learning models. arXiv preprint arXiv:2103.01895 .

ICMAT-CSIC, 2022. ICMAT-CSIC: Equipment and IT Infrastructure. Available at `https://www.icmat.es/facilities/computation/`.

Indyk, I., Zabarankin, M., 2019. Adversarial and counter-adversarial support vector machines. Neurocomputing 356, 1–8.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B., 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 19–35.

Jenkins, P.R., Robbins, M.J., Lunday, B.J., 2021. Approximate dynamic programming for military medical evacuation dispatching policies. INFORMS Journal on Computing 33, 2–26.

Kaggle, 2023. Named entity recognition dataset. `https://www.kaggle.com/datasets/debasisdotcom/name-entity-recognition-ner-dataset`. Accessed: 2023-09-18.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Koller, D., Friedman, N., 2009. Probabilistic graphical models: principles and techniques. MIT press.

Koller, D., Milch, B., 2003. Multi-agent influence diagrams for representing and solving games. Games and economic behavior 45, 181–221.

Krasser, S., 2023. #infosecurityeurope: Preparing for adversarial machine learning attacks.

Melis, M., Demontis, A., Biggio, B., Brown, G., Fumera, G., Roli, F., 2017. Is deep learning safe for robot vision? Adversarial examples against the icub humanoid, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 751–759.

Miller, N., Thomas, M.A., Eichel, J.A., Mishra, A., 2015. A hidden markov model for vehicle detection and counting, in: 2015 12th Conference on Computer and Robot Vision, IEEE. pp. 269–276.

Müller, P., Sansó, B., De Iorio, M., 2004. Optimal Bayesian design by inhomogeneous Markov chain simulation. Journal of the American Statistical Association 99, 788–798.

Naveiro, R., 2021. Adversarial attacks against Bayesian forecasting dynamic models, in: 22nd European Young Statisticians Meeting, p. 66.

Naveiro, R., Redondo, A., Ríos Insua, D., Ruggeri, F., 2019. Adversarial classification: An adversarial risk analysis approach. International Journal of Approximate Reasoning 113, 133–148.

O'Brien, S.P., 2010. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. International studies review 12, 87–104.

Powell, W.B., 2007. Approximate Dynamic Programming: Solving the curses of dimensionality. volume 703. John Wiley & Sons.

Powell, W.B., 2019. A unified framework for stochastic optimization. European Journal of Operational Research 275, 795–821.

Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77, 257–286.

Rios Insua, D., Naveiro, R., Gallego, V., Poulos, J., 2023. Adversarial machine learning: Bayesian perspectives. Journal of the American Statistical Association , 1–12.

Scott, S., 2002. Bayesian methods for hidden Markov models. Journal American Statistical Association 97, 337–351.

Sotgiu, A., Demontis, A., Melis, M., Biggio, B., Fumera, G., Feng, X., Roli, F., 2020. Deep neural rejection against adversarial examples. EURASIP Journal on Information Security 2020, 1–10.

Spears, W.M., 1993. Simulated annealing for hard satisfiability problems. Cliques, Coloring, and Satisfiability 26, 533–558.

Starner, T., Pentland, A., 1997. Real-time American sign language recognition from video using hidden Markov models, in: Motion-based recognition. Springer, pp. 227–243.

Tierney, L., 1994. Markov chains for exploring posterior distributions. The Annals of Statistics , 1701–1728.

Xia, T., Chen, X., 2020. A discrete hidden markov model for sms spam detection. Applied Sciences 10, 5011.

Xiao, H., Biggio, B., Nelson, B., Xiao, H., Eckert, C., Roli, F., 2015. Support vector machines under adversarial label contamination. Neurocomputing 160, 53–62.