arXiv:2207.12355v2 [cs.CR] 5 Aug 2022

Developing Optimal Causal Cyber-Defence Agents via Cyber Security Simulation

Alex Andrew [*1] **Sam Spillard** [*2] **Joshua Collyer** [1] **Neil Dhir** [2]

Abstract

In this paper we explore cyber security defence, through the unification of a novel cyber security simulator with models for (causal) decision-making through optimisation. Particular attention is paid to a recently published approach: dynamic causal Bayesian optimisation (Aglietti et al., 2021, DCBO). We propose that DCBO can act as a blue agent when provided with a view of a simulated network and a causal model of how a red agent spreads within that network. To investigate how DCBO can perform optimal interventions on host nodes, in order to reduce the cost of intrusions caused by the red agent. Through this we demonstrate a complete cyber-simulation system, which we use to generate observational data for DCBO and provide numerical quantitative results which lay the foundations for future work in this space.

1. Introduction

In a recent paper Dhir et al. (2021) voice concern regarding cyber criminals developing new and complex malicious tools which defenders cannot rapidly adapt to, or rapidly counter with existing methods. They argue that the sophistication of these new tools warrants the entry of new advanced cyber-defence techniques to counter these threats. For the purposes of active cyber defence (ACD), they propose the rapid introduction of reinforcement learning (RL) and causal inference. We focus on the latter in this paper due to the much larger, existing, body of work that has hitherto focused on RL for cyber security applications see e.g. (Ridley, 2018) and (Sewak et al., 2021) for a review.

The actions that a blue agent (BA) takes when acting as a defender of a network node can be viewed as a solution to an optimal decision making problem. To have the best chance of survival at each time step, the BA acting as a defender must make the optimal intervention to protect itself. The BA is effectively trying to minimise the probability that it will be compromised, given the current state of the network. This becomes particularly complex in the computer network setting, as the causal relationships between network properties and the chance that a node will be compromised vary over time, as a red agent (RA) takes sequential malicious actions on the network.

To find the optimal interventions which the BA should take to counter the actions of the RA, we propose the use of DCBO which was introduced to identify optimal interventions in precisely this kind of setting. Furthermore, it was shown to converge faster than competing methods in a variety of similar dynamic scenarios (Aglietti et al., 2021). DCBO is able to provide the optimal sequence of interventions, accounting for causal temporal dynamics, in a dynamical system such as that of a network under attack. This optimisation is performed in relation to a specific cost function, such as minimising network down-time or a per-node appraisal. In order to investigate interventional dynamics, we apply DCBO to data simulated by Yawning Titan (YT) – a novel cyber security simulator developed by the UK Government to test new solutions to existing problems within the cyber security domain.

For our simulations, the BA has two available actions. They can either (1) restore a compromised node (thereby removing RA from that node) or (2) isolate the node (removing the connections from that node to other nodes, thereby making it impossible for the RA to reach it) for a number of time steps. In order to avoid trivial solutions in this setting, we impose a cost to both actions, as well as a cost to nodes in the network being compromised. The optimal set of interventions will then be those that minimise the total cost to the agent – keeping the network free from RA infection while minimising the use of expensive actions.

The paper is organised as follows; in §2 we review some necessary background material for the optimisation problem. This leads into related work on cyber simulators in

[*] Equal contribution [1] Defence Science and Technology Laboratory, Salisbury, UK [2] Defence & Security Group, The Alan Turing Institute, London, UK. Correspondence to: Sam Spillard <sam.s@turing.ac.uk>, Neil Dhir <ndhir@turing.ac.uk>.

Presented at the Workshop on Machine Learning for Cybersecurity (ML4Cyber) as part of the Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

§3, allowing us to introduce YT in §4. Having gone through YT, we demonstrate how we use it with models for (causal) decision-making through optimisation, with our methodology in §5. We present experiments, results and conclusion in §6, §7 and §8 respectively.

2. Preliminaries and problem setup

Notation. Random variables are denoted by upper-case letters and their values by lower-case letters. Sets of variables and their values are noted by bold upper-case and lower-case letters respectively. We make extensive use of the do-calculus, for details see (Pearl, 2009, §3.4). Samples (observational) drawn from a system or process unperturbed over time are contained in $\mathcal{D}^O$.

Structural causal model. Structural causal models (SCMs) (Pearl, 2009, ch. 7) are used as the semantics framework to represent an underlying environment. For the exact definition as used by Pearl see (Pearl, 2009, def. 7.1.1). An SCM is parametrised by the quadruple $\langle \mathbf{U}, \mathbf{V}, \mathbf{F}, p(\mathbf{U}) \rangle$. Here, $\mathbf{U}$ is a set of exogenous variables which follow a joint distribution $p(\mathbf{U})$ and $\mathbf{V}$ is a set of endogenous (observed) variables. Within $\mathbf{V}$ we distinguish between three types of variables: $\mathbf{X}$, manipulative (treatment); $\mathbf{Z}$, non-manipulative covariates and $\mathbf{Y}$ the target (output) variables. The sets of parents, children, ancestors, and descendants in a graph $\mathcal{G}$ will be denoted by PA, CH, AN and DE respectively.

Further, endogenous variables are determined by a set of functions $\mathbf{F} \subset \mathcal{F}$. Let $\mathbf{F} \triangleq \{f_i\}_{V_i \in \mathbf{V}}$ (Lee & Bareinboim, 2020, §1) s.t. each f_i is a mapping from (the respective domains of) $U_i \cup \text{PA}_i$ to V_i – where $U_i \subseteq \mathbf{U}$ and $\text{PA}_i \subseteq \mathbf{V} \setminus V_i$. Graphically, each SCM is associated with a causal diagram (a directed acyclical graph, DAG for short) $\mathcal{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ where the edges are given by $\mathbf{E}$. Each vertex in the graph corresponds to a variable and the directed edges point from members of PA_i and U_i toward V_i (Pearl, 2009, ch. 7). A directed edge is s.t. $V_i \leftarrow V_j \in \mathbf{E}$ if $V_i \in \text{PA}_j$ (i.e. V_j is a child of V_i).

2.1. Problem statement

Formally we can pose our problem setting as version of the *control problem* as defined by Pearl & Robins (1995, §2). The setting consists of a DAG, associated vertex set $\mathbf{V}$ partitioned into four disjoint sets $\mathbf{V} = \{\mathbf{X}, \mathbf{Z}, \mathbf{U}, Y\}$ *for each* time-slice. There exists a temporal order on the disjoint sets so that $\mathbf{V} = \bigcup_{t=0}^{T} \mathbf{V}_t$ so that each slice is an ancestor of $\mathbb{1}_{t>0} \cdot \mathbf{V}_t$ in $\mathcal{G}$. Each slice contains *one* outcome variable with index t. Each slice is a sub-graph of $\mathcal{G}$ and each sub-graph is a directed rooted tree where the root node is the response variable Y_t. In the offline setting we are privy to observational samples contained in $\mathcal{D}^O$ and from that information we seek a *plan* which is an ordered sequence of interventions which when implemented, jointly maximise (minimise) the associated sequence of response variables $\{Y_0, Y_1, \ldots, Y_T\}$.

Using causal inference notation we seek the interventional plan: $\{\text{do}(\mathbf{X}_0^\star = \mathbf{x}_0^\star), \text{do}(\mathbf{X}_1^\star = \mathbf{x}_1^\star), \ldots, \text{do}(\mathbf{X}_T^\star = \mathbf{x}_T^\star)\}$ which is found by solving a *dynamic causal global optimisation* (Aglietti et al., 2021) problem of the form:

$$\mathbf{X}_t^\star, \mathbf{x}_t^\star = \underset{\mathbf{X}_t \in \mathcal{P}(\mathbf{X}_t), \mathbf{x}_t \in \text{dom}(\mathbf{X}_t)}{\arg\min} \\ \mathbb{E}[\mathbf{Y}_t \mid \text{do}(\mathbf{X}_t = \mathbf{x}_t), \mathbb{1}_{t>0} \cdot \boldsymbol{\mathcal{X}}_{t-1}] \qquad (1)$$

where $\boldsymbol{\mathcal{X}}_{t-1} \triangleq \bigcup_{i=0}^{t-1} \text{do}(\mathbf{X}_i = \mathbf{x}_i)$ and $\mathcal{P}(\cdot)$ is the powerset over the interventional variables at time t.

This setting makes a number of assumptions which are required to a sequence of optimal actions within a causal framework (the details of which can be found in (Aglietti et al., 2021, Assumptions 1)). Notwithstanding, we seek, at every time step, to construct models for different intervention sets (the expectation in eq. (1)) by integrating various sources of data while accounting for past interventions. With this construction we are able to query them to find an optimal intervention at each time-index given the history.

3. Related work

As noted in §1 much of the literature has focused on the use of RL in cyber security. We will instead consider relevant literature on cyber simulators (§3.1) and the few studies which have used concepts from causal inference for cyber security research (§3.2).

3.1. Cyber simulators

There are many cyber simulators that exist or are currently in development, so why the need for YT in this work? We consider our contemporaries to motivate the use of YT.

FARLAND (Molina-Markham et al., 2021) aims to support the development of RL agents within a hybrid simulation and emulation system. This increased fidelity provides a means of developing RL cyber defence agents capable of defending real systems but hinders the research of fundamental aspects such as algorithm development. YT is more abstract and flexible which enables us to easily test different and novel approaches without having to significantly re-engineer the underlying code base. CybORG (Standen et al., 2021), comparatively, shares many features with YT. Both are simulators and are controlled by YAML configuration files that define the scenario for a given environment. The fundamental difference however is the fidelity of node representation. As CybORG contains both a simulation and emulation component, the simulation models real features of hosts, such as the operating systems, hardware architectures, users and passwords. All of this information is intentionally abstracted away within YT. This level of abstraction could be viewed negatively but is the very reason new and novel decision making approaches can be rapidly integrated and experimented with whilst also providing a cyber relevant environment. Another recent development is Microsoft's Cyber Battle Simulator (Team., 2021, CBS). Similar to YT, CBS abstracts away from many of the details of real life however still incorporates host information such a specific host vulnerabilities and in-depth environment setup. When designing YT we wanted to focus on a very simple setup where the hosts only have a few shared attributes and it is easy to spin up new network models. This would allow someone to quickly set up an abstraction of a network topology, without having to worry about host-specific details, and easily integrate it into their own research.

3.2. Causality for cyber security

Causal inference methods have yet to find their place within the cyber security domain. There have been some previous attempts to apply causal decision making to historic cyber data by Abel et al. (2020) and Mueller et al. (2019). However, these attempts use causal models without a temporal dimension (the main feature of DCBO) and only focus on predicting and alerting rather than selecting optimal defensive actions. Since we are using a simulator to collect our data, we can easily test new scenarios and model performance differences that one would not be able to with a single historic data set.

In another strand of work, Shi et al. (2017) use transfer entropy to quantify the causal relationship between any two variables in their cyber-physical system. In other words they undertake causal-discovery of their complex network in order to not a priori have to rely on a model of the underlying dynamical system.

4. Yawning titan

We introduce Yawning Titan (YT). YT is an abstract, highly flexible, cyber security simulator that is capable of simulating a range of cyber security scenarios[1]. We provide a formal definition for completeness.

Definition 1. Cyber security simulator. Simulation in the cyber context is the process of modelling a real-world computer-system environment to predict outcomes of actions from a number of agents where the goals of said agents are to control either system or data from the system.

There are a number of these simulators (see §3 for related work), who broadly work on the same principle as described in the above definition. YT is built to train cyber defence agents to defend arbitrary network topologies in highly customisable configurations. It was developed with the following design principles in mind:

1. Simplicity over complexity
2. Minimal hardware requirements
3. Support for a wide range of algorithms
4. Enhanced agent/policy evaluation support
5. Flexible environment and game-rule setup

Design principles 1, 2 and 5 make YT an ideal environment to test and evaluate new and novel approaches to cyber security decision making and provides a means of transitioning approaches such as DCBO into a cyber defence context. Principally we will be modelling the actions of one blue agent (BA) and one red agent (RA).

Let $\mathcal{G}_{\text{net}} = \langle \mathbf{V}_{\text{net}}, \mathbf{E}_{\text{net}} \rangle$, with vertex set $\mathbf{V}_{\text{net}}$ and edge set $\mathbf{E}_{\text{net}}$, be the undirected graph which simulates a computer network in YT. Before going further, take care not to confuse the two types of graphs we employ in this work; $\mathcal{G}$ represents the directed causal diagram and $\mathcal{G}_{\text{net}}$ the undirected associated computer network – an example instance is shown in fig. 1. Each node $V_i \in \mathbf{V}_{\text{net}}$ represent the hosts on the network and arc $E_j \in \mathbf{E}_{\text{net}}$ represents possible connectivity between the two connected hosts (grey edges in fig. 1). This approach allows us to easily model a large variety of different types of network topologies in their simplest form without worrying about protocols or different types of hosts. This basic approach to the network allows us to abide by our first design principle.

YT allows for considerable customisation through the use of customised configurations. The configuration summary contains a number of different settings that the user can control in order to change how the game is played. From these settings you can change success conditions for the agents and how each of the different agents can act by toggling actions and action probabilities.

The machines in the network each have their own attributes that affect how they behave and how they are affected by the RA and the BA.

Vulnerability score Affects how easy it is for the RA to compromise the node. It is automatically generated at the start of each scenario based on settings in the configuration file and there are blue actions that can modify the vulnerability scores of nodes.

Isolation Status Indicates whether a node has been isolated. This means that all of the incoming and outgoing connections are disabled. This can be modified by blue actions.

[1] A Python implementation is available: https://github.com/dstl/YAWNING-TITAN.

True Compromised Status Indicates whether a node has been infected by the RA. If the RA has control of the node then it can use the node as a foothold to spread to other connected nodes.

Blue seen Compromised Status Represents if the node is compromised and the BA is aware of the intrusion. Depending on the scenarios configuration, the BA may only see an obscured view of the network and see this instead of the true value, effectively simulating the differences between perfect and imperfect detection.

Using the aforementioned configuration settings, the user is able to modify the subset of actions that are available to RA and BA. The BA has actions that allow it to:

- reduce the vulnerability of nodes,
- scan the network for red intrusions,
- remove RA from a node,
- reset a node back to its initial state,
- deploy deceptive nodes,
- isolate a node and
- reconnect a previously isolated node.

The RA has a variety of different attacks that it can use based on the settings in the configuration file. Unless using a guaranteed attack, eq. (2) will be used to determine if the RA (with a skill level $0 < RS \leq 1$) will compromise node $V_i \in \mathbf{V}_{\text{net}}$:

$$AS = \frac{100 \times RS^2}{RS + (1 - \text{vuln}(V_i))}. \tag{2}$$

The attack will succeed if:

$$AS \geq u \tag{3}$$

where $u \sim \mathcal{U}(0, 100)$ is sampled from a uniform distribution, AS is the attack score (a measure of how powerful an attack is) and $\text{vuln}(V_i)$ is the vulnerability of node V_i (a measure of how exposed a host is to being compromised). For example, a computer without a firewall or a user that has no security training, could be modelled as having high vulnerability. Since $0 \leq \text{vuln}(V_i) \leq 1$ and $0 \leq AS \leq 100$, the use of this formula ensures that the likelihood for the RA to compromise a node increases proportionately with the skill of the agent and decreases proportionately with the defence of the node $(1 - \text{vuln}(V_i))$.

At each time step, each agent is allowed one action, chosen from the subset of activated actions, to affect the environment. The order of YT execution is as follows:

1. The RA performs their action
2. The Environment checks if the RA has won
3. The BA performs their action
4. The Environment returns the reward of the BA's action
5. The Environment checks if the BA has won.

As YT is built upon the `OpenAI Gym` framework (Brockman et al., 2016), the code is simple to understand and highly decoupled. The agents can be any function that picks an action based on an observation matrix. Consequently, although YT was built to run with a `Stable-Baselines3` (Raffin et al., 2021) it is simple to model the actions of the BA using a causal sequential decision-making agent.

5. Methodology

In this section we characterise the various building-blocks required for the optimal intervention-recommendation part of this study. First and foremost, we consider the different decision-making methods under investigation – for continuous action spaces. All are based on Bayesian optimisation (Močkus, 1975; Garnett, 2022, BO) which is our weapon of choice to solve eq. (1).

BO is an optimisation method used for global optimisation of black-box functions (Garnett, 2022, §1) – i.e. those which do not assume a specific functional form (Mockus, 2012) and are only distinguished by their inputs and outputs. It finds use in numerous domains (Garnett, 2022), but is principally used for optimising expensive black-box functions (where the cost of an evaluation can be e.g. monetary, time-dependent or societal). BO has key components:

1. the objective function is modelled with a Gaussian process (GP) (Williams & Rasmussen, 2006) – the expectation in eq. (1);

2. each new evaluation of the objective function is incorporated via a Bayesian update procedure and

3. an acquisition function is used to determine the next, high utility, point of evaluation of the objective.

Numerous advancements and improvements to BO have been made over the years, but here we focus on those that extend the original framework to the explicitly causal setting. The first causal extension to BO is given by the causal Bayesian optimisation (CBO) model, introduced by Aglietti et al. (2020). CBO is used in settings where the response variable Y, is part of a SCM in which a sequence of interventions can be performed. CBO is designed for *static* environments. It does not account for the temporal evolution of the system, consequently breaking the temporal dependency structure which exists among variables – see fig. 2 for the causal diagram used in this study.

In order to handle problem spaces with explicit temporal dynamics, Aglietti et al. (2021) introduced dynamic causal Bayesian optimisation (DCBO). DCBO is useful in the kind

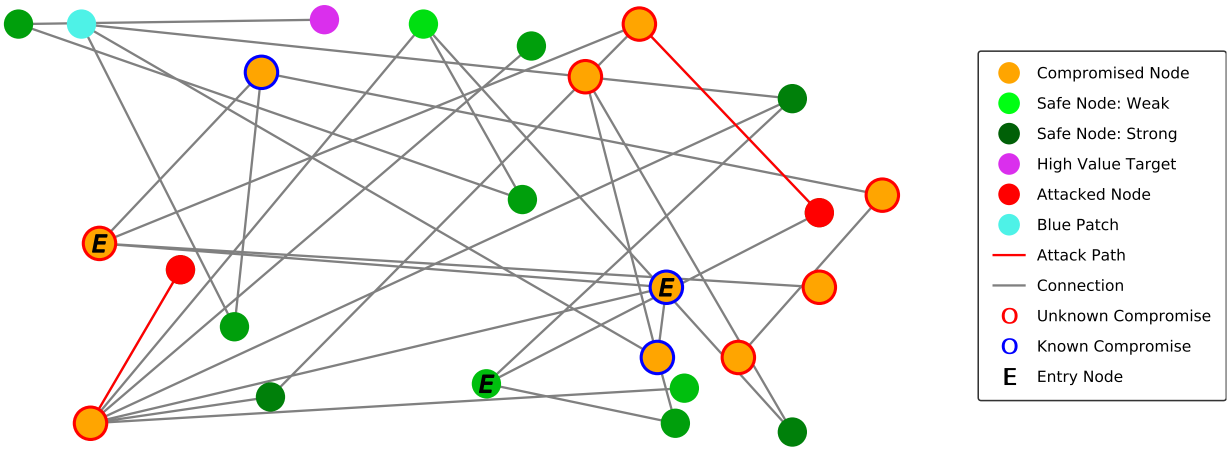

Figure 1: Example output from YT on a 25 node network $\mathcal{G}_{\text{net}}$. The BA is defending a high value target node (purple node). The RA controls a large portion of the network, indicated by yellow nodes with known and unknown compromise (indicated by the edge colour of the yellow nodes), unbeknownst to the BA. There are three entry nodes into the system, two of which the RA has made use of (marked by an 'E'). The BA is in the process of removing the RA from the node adjacent to the high value target node. The RA is attacking a node in the bottom left hand corner of the network. Best viewed in colour.

of scenarios considered in this paper; ones where all causal effects in the causal diagram are changing over time. At every time step DCBO identifies a local optimal intervention by integrating both observational and past interventional data collected from the system. Precisely, a DCBO agent seeks to minimise the objective function while accounting for the cost of intervention. It is important to note that interventions typically have a large financial, societal, ethical or other cost associated with them. This is true in the cyber setting as well where a potentially optimal intervention could be e.g. shutting down a node but which has huge financial cost due to the services running on that node. Hence an agent needs to optimise the objective function whilst taking *that* interventional cost into account.

Using observational data from YT and the three methods described above, we investigate optimal action policies which reduce and prevent intrusions into a network from a RA, by solving the DCGO problem in eq. (1). We now outline relevant causal components.

5.1. Causal diagram

The causal diagram induced by the SCM is shown in fig. 2 and is in accordance with causal sufficiency (no unobserved confounders) as per the assumptions made in (Aglietti et al., 2021, assumptions 1). Further, the graph topology is homogeneous within a time-slice and remains so across time. Simply, the DAG prescribes the causal relationships between endogenous variables within and across time. Variable descriptions are given in table 1.

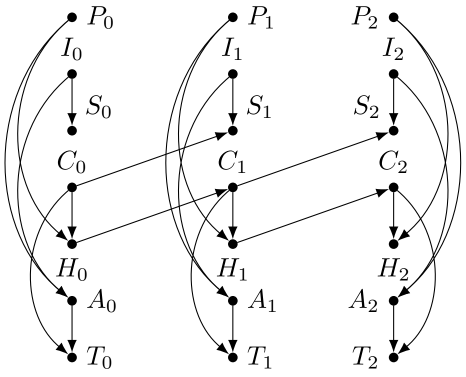

Figure 2: Causal diagram $\mathcal{G}$ shown for the first three time-steps. The target variable is T with manipulative variables given by $\mathbf{X}_t = \{P_t, I_t\}$. The rest are non-manipulative variables. The within slice topology repeats for 25 time-steps in our experimental setup and the variable connectivity repeats as shown for the same length of time. For variable descriptions see table 1.

This DAG is chosen as we believe it represents the simplest causal structure of how compromise spreads through a network, whilst accounting for the specific operating costs of taking actions to prevent further intrusion. It is important to factor in these costs to avoid trivial solutions, such as disconnecting every node from the network (despite that sometimes being the best option in a hostile cyber environment)! Additionally, in order to ensure the temporal func-

Table 1: SCM variable descriptions. For their causal relationship see fig. 2.

Variable	Description
P	Probability that a BA will restore (RES) a node, thereby removing the RA from that node.
I	Probability that a BA will isolate (ISO) a node, thereby making both inbound and outbound attacks from a RA impossible.
S	Available attack surface for the RA. Number of nodes that are connected to the network and not yet compromised.
C	Operating cost of a node being compromised. We assume that all nodes have equal operating cost of compromise.
H	Likelihood of further compromise. This is equal to the total vulnerability of all nodes that are directly connected to any compromised node.
A	Operating cost of taking a given action. We define different costs for both RES and ISO.
T	Total operating cost. Sum of C and A.

tional form of the RA is captured in our model, we choose to explicitly dynamically model S and H, the attack surface and likelihood of additional compromise. This allows the results of actions that the RA takes in between time steps, as well as the causal relationships, to be explicitly modelled. Due to our total control of the YT environment, we are able to make certain assumptions that would not hold in an equivalent real-world setting, such as; the only BA actions being taken are RES or ISO, and only one at a time; the RA can only spread through the network from nodes it already controls, and cannot create additional entry points; and the topology of the network remains consistent across time, apart from those cases where a node is disconnected by the BA. These are strong assumptions but critical to ensure our DAG accurately represents the data generating process.

5.2. Structural equation model

The DAG is chosen to represent the most important properties of the environment that could also be used in a real world scenario. The nodes represent the chance that a system can become compromised, how that compromise spreads, and costs associated with that compromise.

We setup YT as a data-generator in order to collect observational data, used to fit the causal relationships prescribed by the directed edges in $\mathcal{G}$ in fig. 2. For this, we initialise a BA with a two-dimensional action space of $\{\text{restore (RES)}, \text{isolate (ISO)}\}$ and random action probabilities – full description in table 1.

The functional relationship between variables, the true SEM, are provided in eq. (4) to eq. (10). Here K_t is the subset of all nodes that are compromised at time t and ϕ_t the subset of all nodes that are isolated at time t. Where ϕ^c denotes the complement of subset ϕ, $N^+(V_i)$ denotes all nodes that can be reached via a single edge from node V_i (see fig. 1 for an example), $\Gamma_{\{c,\text{ RES, ISO}\}}$ represents the cost of compromise, restore and isolate respectively, $\mathcal{A}_t$ represents the action the BA took at time t, $\text{vuln}(V_i)$ represents the vulnerability of node V_i and $[\cdot]$ represents the Iverson bracket. The exponent in eq. (7) is implemented in order to improve convergence of an RL agent trained on a similar environment, so we include it here for consistency.

$$P_t = p_t(\text{RES}) \tag{4}$$

$$I_t = p_t(\text{ISO}) \tag{5}$$

$$S_t = |K_t^c \cap \phi_t^c| \tag{6}$$

$$C_t = \left(\sum_{n=1}^{n=N} \Gamma_c[n \in K_t]\right)^{1.5} \tag{7}$$

$$H_t = \sum_{n \in K_t} \sum_{v \in N^+(n)} (\text{vuln}(v)[v \notin \phi_t]) \tag{8}$$

$$A_t = \begin{cases} \Gamma_{\text{RES}} & \mathcal{A}_t = \text{RES} \\ \Gamma_{\text{ISO}} & \mathcal{A}_t = \text{ISO} \end{cases} \tag{9}$$

$$T_t = C_t + A_t. \tag{10}$$

These structural equation models allow us to transfer observable information from YT by manipulating properties of the environment so that we receive an expression for each variable in the DAG. The time dependence of the two observable variables that have transition edges, namely S_t and C_t, is implicit in the above equations. As C_{t-1} increases (due to an increased number of compromised nodes), there is an implicit change in K_t and therefore in S_t. Similarly, as H_{t-1} increases due to more vulnerability in the network, there is an implicit increase in C_t as additional nodes get compromised between times $t-1$ and t.

As noted, having access to observational data means we are able to estimate the SEM (since we do not have access to the true SEM):

$$
\begin{aligned}
P_t &= f_P(t) + \epsilon_P \\
I_t &= f_I(t) + \epsilon_I \\
S_t &= f_S(C_{t-1}, I_t) + \epsilon_S \\
C_t &= f_C(H_{t-1}) + \epsilon_C \\
H_t &= f_H(P_t, C_t) + \epsilon_H \\
A_t &= f_A(P_t, I_t) + \epsilon_A \\
T_t &= f_T(C_t, A_t) + \epsilon_T
\end{aligned}
$$

by placing Gaussian process estimators on all functions $f_i(\cdot), i \in \{P, I, S, C, H, A, T\}$. This SEM is then used to find the ground truth optimal intervention for the system at each time step, $\{y_t^\star \mid t = 0, \ldots, 24\}$, given it has full access to the data from YT and the causal graph in fig. 2.

6. Experiments

To fit our initial SEM, we use YT to generate observational data. This involves setting up a number of YT environments each with a different dummy agent that has a probabilistic chance to take each action. We then step through the environments to see how well the chosen probabilities perform[2].

We configure a simple blue agent (BA) that can take two actions, restore (RES) or isolate (ISO), with probabilities $p(\text{RES})$ and $p(\text{ISO})$. Additionally, if a node is isolated, it will automatically be reconnected to the network after five time steps.

We also configure a simple red agent (RA) that can only perform a single action, a basic attack, that uses eq. (2) and eq. (3) with a skill level of 25%. A 25% skill level is rather low and means that on average the weakest nodes will take four turns to compromise. The RA chooses its target by picking a random node V_i from the set of all nodes $V_i \in \mathbf{V}_{\text{net}}$ such that V_i is an entry node or V_i is connected to a compromised node. In fig. 1 the RA can target any node with a link to an orange node or any node marked with an 'E'. In fig. 1, the RA has chosen a node in the bottom left.

We initialise ten identical network environments with identical RA's. We pick a simple network topology with ten network nodes. Combined with the basic action spaces of both the BA and RA, this simple setup allows the relatively simple DAG in fig. 2 to be constructed. Integrating DCBO with more complex topologies and agents would be easy to do, but would require re-drawing the DAG such that all causal relationships are accounted for. In the setup we randomly assign a node in the network to be the entry node into the network for the RA. We then randomly select a single node to be a high value target (HVT) from the set of nodes that are furthest away from the entry node – the purple node in fig. 1. The configuration selected for this experiment was chosen to create a non-trivial environment that may not necessarily reflect any real world systems but still contains features and challenges that agents would have to face in these systems. For each environment, we draw a single random value for $p(\text{RES})$ and $p(\text{ISO})$ from a Gaussian distribution centred around 0.5, and instantiate a BA using these values. We then run the simulation for 25 time steps, or until the HVT is compromised, allowing both RA and BA to sequentially take actions according to their configurations. At each time step, we calculate the values for the nodes in fig. 2 according to the SEM in eq. (4) to eq. (10).

This provides an array of observational data for each node of size 10×25. As in (Aglietti et al., 2021, 4.2 Real experiments), CBO and DCBO use this data to fit a non-parametric simulation of the relationships in fig. 2 and to compute a causal prior.

We provide both CBO and DCBO this observational data and run them, along with BO, for 50 trials each, allowing them to intervene on the manipulative variables P and I in the domain $[0, 1]$, to minimise T at time steps $t = 22, 23, 24$. The convergence of the models on the optimal outcome value is shown in fig. 3, against the cost of intervention. Note that CBO and DCBO converge much more consistently and more efficiently than BO.

7. Results and discussion

Figure 3 shows that DCBO and CBO provide a way of efficiently evaluating the optimal decisions to make in a complex cyber environment, with a time-dependent causal structure. They are able to take historic data, along with a causal model of the data generating process, and find an optimal intervention set, choosing actions with the least associated cost from both the action, and the effect of the RA.

A large part of both CBO and DCBO's efficiency, and an additional hurdle to implementing these causal methods in a real-world cyber setting, is the creation of the underlying DAG on which these methods rely for inference. For a controlled setting, such as those within simulators such as YT, the creation of this DAG and corresponding observational data is relatively trivial. On the contrary, in a real-world setting, the DAG creation must be knowledge-driven and can often grow to unmanageable scales when considering all the factors at play in a potentially hostile cyber environment. However, with sufficient domain expertise and an understanding of the data-generating process, we have shown that a causal view of activities within a complex cyber simulator is possible and indeed more powerful than a naïve data-centric view. We leave it as an exercise to the reader to think of DAG's for real-world networks that they

[2] Python code required to reproduce this experiment: https://github.com/alan-turing-institute/causal-cyber-defence

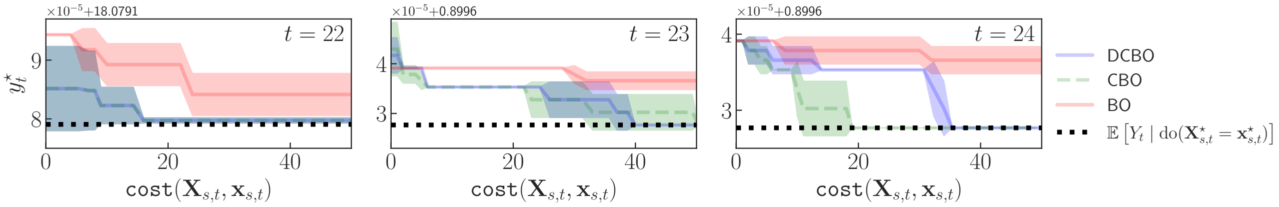

Figure 3: Experimental results of different optimisation methods applied to YT data, showing convergence of DCBO and competing methods (CBO and BO) across five replicates. The black dotted line shows the optimal response value $\{y_t^\star \mid t = 22, 23, 24\}$. Shaded areas are ± one standard deviation. The x-axis shows the total *cumulative cost* of the intervention set over 50 trials - effectively how many times the optimisation had to probe the underlying function in order to build up an approximation of the causal relationship.

are familiar with, as we believe it can shed light on relationships that might otherwise go un-noticed.

The results show that CBO generally converges to the optimal decision the quickest with DCBO varying in its relative efficiency. This implies that in this specific network topology, the temporal relationships do little to aid in the convergence on the optimal solution. However, the causal relationships *within* time slices, exploited by both CBO and DCBO, do significantly improve the convergence over traditional BO.

One of the main advantages of CBO and DCBO is speed and their handling of data-sparsity. In addition, provided that the training environment remains broadly stationary, DCBO and similar approaches are able to be rapidly re-trained and re-calibrate defensive policies as the threats and risk environment changes. Compare this to RL where a major disadvantage of those methods is their sample inefficiency and training time. As a means of comparison, a Proximal Policy Optimisation (PPO) agent (Schulman et al., 2017) was trained using the stable-baselines3 (Raffin et al., 2021) RL library within the same environment used to gather data for our comparison models. It took significantly longer to converge to its optimal interventions (see fig. 4) and its optimal solution was far from the true optimal. This shows our possible need for another method to choose optimal decisions and DCBO and CBO have been shown to quickly hone in on the true solution. These results are not conclusive. We will need to perform further analysis with RL to fully understand the pros and cons.

8. Conclusion and future work

We have demonstrated the utility of using causal inference in a cyber-simulation framework where the goal is to provide a blue agent with optimal action recommendations. In that process we have made a number of operative assumptions, discussed throughout the paper. Due to the preliminary nature of the work, there are many avenues going forward.

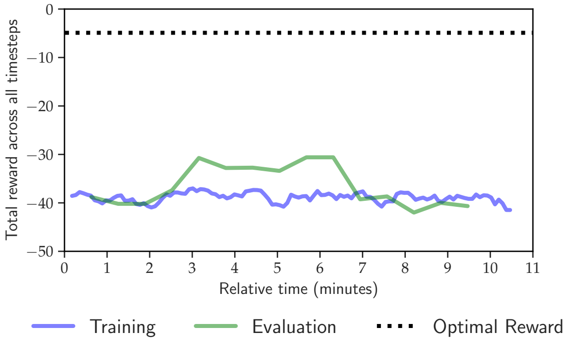

Figure 4: PPO agent trained in the same environment as the other decision making algorithms. The reward on the horizontal axis is the cumulative reward over 25 time steps. The reward (R) at each timestep is $R = -T$ where T is the total cost calculated in eq. (10). The PPO manages to reach its optimal solution within a couple of minutes but it should be noted that we encountered a considerable spread in performance when testing PPO in this context.

On the causal inference side, we have assumed *causal sufficiency* i.e. the absence of any unmeasured confounders. This is a strong and oftentimes unrealistic assumption. There is a host of literature which focuses on causal decision-making in the RL domain, which we aim to bring across to the BO sphere, such as the work by Lee & Bareinboim (2018). Moreover, it is equally true that we have only focused on continuous interventions in this paper but cyber-security data is often discrete, hence future work will incorporate options for handling discrete data.

On the YT side, we hope to further explore causal decision making algorithms and their integration into YT, increasing the complexity of the environments and number of actions available. We also hope to use these models in an on-line setting, where causal decision making is able to provide real-time recommendations of interventions, which are then fed directly into YT's simulation.

With the on-line setting in place, we then plan to gather more results on how algorithms such as DCBO fare against traditional reinforcement learning algorithms, such as PPO, in these different and more complex environments. What is more, DCBO is *always* active but this may not be necessary. It would be better if DCBO activated only *if* a threat has been detected upon which an optimal causal decision-making agent is deployed. Early work on one such (non-causal) approach was recently published by Hammar & Stadler (2022) using optimal stopping (Bertsekas, 2012). Finally, we hope that a systematic comparison of agents trained using a variety of different algorithms will show the value that a causal understanding of the cyber environment can bring to optimal cyber decision making.

References

Abel, S., Tang, Y., Singh, J., and Paek, E. Applications of causal modeling in cybersecurity: An exploratory approach. *Advances in Science, Technology and Engineering Systems Journal*, 5(3):380–387, 2020.

Aglietti, V., Lu, X., Paleyes, A., and González, J. Causal Bayesian Optimization. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3155–3164. PMLR, 26–28 Aug 2020.

Aglietti, V., Dhir, N., González, J., and Damoulas, T. Dynamic Causal Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 35, 2021.

Bertsekas, D. *Dynamic programming and optimal control: Volume I*, volume 1. Athena Scientific, 2012.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Dhir, N., Hoeltgebaum, H., Adams, N., Briers, M., Burke, A., and Jones, P. Prospective artificial intelligence approaches for active cyber defence. *arXiv preprint arXiv:2104.09981*, 2021.

Garnett, R. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

Hammar, K. and Stadler, R. Intrusion prevention through optimal stopping. *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022. doi: 10.1109/TNSM.2022.3176781.

Lee, S. and Bareinboim, E. Structural causal bandits: where to intervene? *Advances in Neural Information Processing Systems*, 31, 2018.

Lee, S. and Bareinboim, E. Characterizing optimal mixed policies: Where to intervene and what to observe. *Advances in neural information processing systems*, 33, 2020.

Močkus, J. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pp. 400–404. Springer, 1975.

Mockus, J. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

Molina-Markham, A., Miniter, C., Powell, B., and Ridley, A. Network environment design for autonomous cyberdefense. *arXiv preprint arXiv:2103.07583*, 2021.

Mueller, W. G., Memory, A., and Bartrem, K. Causal discovery of cyber attack phases. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1348–1352, 2019. doi: 10.1109/ICMLA.2019.00219.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009.

Pearl, J. and Robins, J. M. Probabilistic evaluation of sequential plans from causal models with hidden variables. In *UAI*, volume 95, pp. 444–453. Citeseer, 1995.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Ridley, A. Machine learning for autonomous cyber defense. *The Next Wave*, 22(1):7–14, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sewak, M., Sahay, S. K., and Rathore, H. Deep reinforcement learning for cybersecurity threat detection and protection: A review. In *International Conference On Secure Knowledge Management In Artificial Intelligence Era*, pp. 51–72. Springer, 2021.

Shi, D., Guo, Z., Johansson, K. H., and Shi, L. Causality countermeasures for anomaly detection in cyber-physical systems. *IEEE Transactions on Automatic Control*, 63(2):386–401, 2017.

Standen, M., Lucas, M., Bowman, D., Richer, T. J., Kim, J., and Marriott, D. Cyborg: A gym for the development of autonomous cyber agents. *arXiv preprint arXiv:2108.09118*, 2021.

Team., M. D. R. Cyberbattlesim. https://github.com/microsoft/cyberbattlesim, 2021. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.