Adversarial Resilience Learning — Towards Systematic Vulnerability Analysis for Large and Complex Systems

Lars Fischer*†, Jan-Menno Memmen†, Eric MSP Veith†, Martin Tröschel†
*Universität Oldenburg, †OFFIS e.V.
Escherweg 2
26121 Oldenburg, Germany
Email: *l.fischer@uol.de, †{eric.veith,martin.troeschel}@offis.de

***Abstract*—Cyber-physical systems (CPSs) can be checked using numerous approaches, ranging from algorithmic model checking for a complete coverage of a finite-state system to extensive simulation, after which the system's state is compared with defined invariants. However, modern CPSs are confronted with an increased amount of stochastic inputs, from volatile energy sources in power grids to broad user participation stemming from markets. The search space for a complete cover of a CPS becomes too large, while contracts cannot be formulated anymore considering the potentially erratic behavior of a user, or even in the face of a cyber attack. At the same time, the goal of resilience critical infrastructure cannot be eschewed, but the integration of user behavior and even non-checkable artificial intelligence algorithms is mandated, even required to meet, e.g., the goal to satisfy 80% of the gross power consumption from renewable energy sources by 2050. The concept of Adversarial Resilience Learning (ARL) formulates a new approach to CPS checking and resilient operation. It defines two agent classes, attacker and defender agents. The goal of the attacker is to de-stabilize the CPS, whereas the defender works to maintain a stable operational state. The quintessence of ARL lies in the attacker training the defender on a model of the CPS; as such, it is not a zero-sum game, but the learning of a resilient operation strategy for a CPS. This paper introduces the concept and the nomenclature of ARL, and, based on it, the description of experimental setups and results of a preliminary implementation of ARL in simulated power systems.**

***Keywords*—agent systems; reinforcement learning; adversarial control; resilience; power grid**

I. Introduction

Current newspapers are full of horrific tales of "cyber-attackers" threatening our energy systems; the December 2015 Ukraine power grid cyberattack is a particularly notable one [1], [2], which has seen a continuation in 2017 [3]. And, if not for the notorious "evil state" actor, it is the ongoing digitization necessary to enable increasing renewable and volatile energy generation that threatens our energy supply and thus the stability of our society. While the main approach seems to be to patch-up the detected vulnerabilities of protocols, software and controller devices, our approach is to research and develop the means to systematically design and test systems that are structurally resilient against failures and attackers alike.

Security in cyber-systems mostly should be concerned with establishing asymetric control in favour of the operator of a system. In order to achieve this on a structural level at design time, reproducible benchmark tests are required. This is notoriously difficult for intelligent adversaries whose primary abilities are adaption and creativity. Thus, testing methods nowadays are either reproducible, but insufficiently model an attacker; or they involve unreproducible human elements. Reinforcement Learning (RL) may be useful to provide at least some adaptability of reproducible attacker models.

This work takes its motivation and first practical implementation from the power system domain, but the work can directly be applied to all highly complex, critical systems. Systems that may benefit from Adversarial Resilience Learning (ARL) are too complex to be sufficiently described using analytic methods, e.g., because the number of potential states is too large and the behaviour is too complex with too many non-trivial interdependencies. This also includes stochastic external factors, such as the behavior of market actors.

This work introduces ARL, which provides a method to analyze complex interdependent systems with respect to adversarial actors. The foremost motivation is to provide a method for analysis based only on an interface description of an agent's sensors and actuators in the cyber-physical system (CPS). We expect ARL to identify potentially unknown vulnerabilities. A key part of ARL is to identify the minimal chain of actions required to reproduce a vulnerability; this effectively entails both the ARL nomenclature introducted in the paper as well as careful Design of Experiments (DoE).

The main contribution of this paper is the introduction of a novel structure for training agents competing against each other on a model of a CPS without explicitly perceiving each other's actions. By setting up RL-based agents in a competitive situation, the learning-complexity is comprised not only of a highly complex system, but also of competing agents, whose changing state, manifested by modified behaviour of the system under consideration, has to be included in the trained model. We assume that this provides a very interesting new problem class for RL, as it introduces a cyclic learning competition.

The paper is structured as follows. First, a brief introduction into related techniques in machine learning and related work for complex system analysis is given in Section II. The paper then defines the concept of ARL in Section III, and introduces its application to adversary testing in power system control in Section IV. The paper is completed by a presentation of lessons learned and results from an early proof-of-concept demonstrator in Section IV-B. It concludes with a discussion and an outlook in Section V.

 ISBN: 978-1-61208-713-9

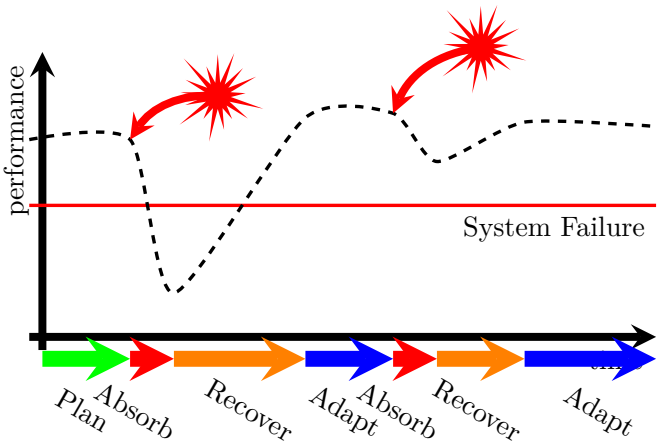


Figure 1. Resilience Process for system performance

II. Related Work

This work aims at exploring the feasibility of improving resilience of complex systems using machine learning to train adaptive agents. The term *resilience* is lacking a coherent and precise definition across fields. Generally, it denotes the ability of a system to withstand unforseen, rare and potentially catastrophic events, recover from the damage and adapt by improving itself in reaction to these events. Ideally, resilience is increasing monotonously throughout system improvement. A useful simplification is observation of the changing behaviour of system performance as an artefact resulting from resilience processes. Different formalization of resilience processes exist, but most distinguish subprocesses for planning, absorption of damage, recovery (or self-healing) and improvement (or adaption) [4].

See Figure 1 for an expression of a hypothetical system's performance suffering twice from damaging events. Resilience is modelled as a sequential process: *plan*, *absorb*, *recover*, and *adapt* [5]. As consequence of the first event, the performance of the system is pushed below a *failure threshold*, i.e., the system fails to provide its service. Improvement of the system is then achieved after recovery as the system is able to keep the performance above the failure threshold during the second event.

A. Analysis and Stochastic Modelling

The main distinction of our approach as compared to game theoretic modelling and stochastic analysis is the use of co-simulation and heuristic approaches instead of formal abstraction of complete systems. The underlying assumption is that a system-of-systems is too complex and malicious adversaries are too unpredictable to be sufficiently analyzed.

Traditional analysis of CPSs has either checked for liveness ("something good eventuall happens") or safety requirements ("nothing bad ever happens") through mathematical modelling and model checking, using temporal logic, decision trees, or similar devices, or by employing discrete simulations over k timesteps and checking against formulated invariants [6], through which stochastic effects can be introduced. Complexity has usually been abstracted away by contracts; combining contracts and simulation is still a topic of research [7].

Compared to Attacker-Defender Models, described by, e.g., Brown et al. [8], that aim at analyzing an equilibrium between attackers and defenders in dynamic systems, our work heuristically approaches an estimate of the asymmetry of attacker and defender in these systems. The approach of ARL is structurally similar to the concept of *Stackleberg Competitions* and related applications of stochastic analysis, e.g., pursuit-evasion in differential games [9]. These approaches seem to only be applicable to scenarios that can be restricted to few degrees of freedom. More realistic behaviours of opportunistically acting threat agents within complex system-of-systems leads to an explosion of states in analytic approaches.

Recent surveys seem to support this view. Referenced approaches on power systems by Do et al. [10] provide no details on the used game-theoretic model and use ambiguous terminology of the researched threat scenarios. Approaches in Machine Learning (ML) to tackle complex problems, on the other hand, have been very successful in providing practical solutions.

B. Machine Learning

Artificial Neural Networks (ANNs) are universal function approximators, meaning that they can be used as a statistical model of any Borel-measurable function $\mathbb{R}^n \mapsto \mathbb{R}^m$ with desired non-zero error [11]–[13]. Already the standard Recurrent Neural Network (RNN) has the capacity to approximate any non-linear dynamic system; Siegelmann and Sonntag have shown that RNNs are turing-complete [14]–[16].

In practice, a typical problem for which RNNs, especially structures containing Long-Short Term Memory (LSTM) cells [17] or Gated Recurrent Units (GRUs) [18], [19] are used, is time series prediction. Predicting a time series with an RNN constitutes the instantiation of a (non-linear) dynamic system [20]–[22], i.e., the prediction is the result of the system's behavior, which is, in turn, modeled and approximated by the RNN. Cessac has examined ANNs from the perspective of dynamical systems theory, characterizing also the collective dynamics of neural network models [23].

For ARL, we assume a common model that is used by two distinct agents: while one probes the model for weaknesses in order to find attack vectors, the other monitors the system and, unbeknowing of the presence of the attacker or its actions, works at keeping the system in its nominal state. Through this structure, the notion of ARL assumes that the model—i. e., each agent's environment—is not completely known to the respective agent. Therefore, the usage of RL readily suggests itself. In a setup such as ARL provides, RL is the natural choice for learning algorithms [24]–[26].

Even though in theory, the notion of RL is not tied to ANNs per se [27], the incremental training process makes them suitable for RL in contrast to other structures, such as decision trees, which usually need the full data set for effective training. For training ANNs and RNNs supervisedly, which is

 ISBN: 978-1-61208-713-9

a core task in RL, gradient-decent-based algorithms of the Backpropagation-of-Error family are leading by far [28]–[32], followed by evolutionary algorithms, such as CMA-ES [33]–[35] or REvol [36], [37]. In theory, RNNs have the capacity to simulate arbitrary procedures, given the proper set of parameters; in practice, this training task has proven to be complicated. Neural Turing Machines, such as the Differentiable Neural Computer (DNC) introduced by Graves et al. [38], [39], counter the complexity with a vastly increased addressable memory space and have shown to be able to simulate simple, but complete algorithms like sorting. In theory, DNCs at the core of ARL would make the concept itself transferable to similar CPSs once trained, as well as to allow a variable set of sensors and actuators over time.

However, all optimization methods adapt the ANN to minimize a cost function and not directly to create a model of a problem; this happens only indirectly. As a result, ANNs can still be "foiled," i.e., made to output widely wrong results in the face of only minor modifications to the input. This effect and how to counter it is the subject of Adversarial Learning (AL) research. Even though seemingly similar by name, ARL should not be confused with AL, as the core problem of ARL is not the quality of sensory inputs, but the unknown CPS being subject to ARL execution. The concept we propose in this paper is related to AL only insofar, as both concepts use two distinct ANNs with conflicting objectives [40].

A second concept that is potentially similar in the name only is that of Generative Adversarial Networks (GANs): With unsupervised learning, the ANN tries to detect patterns in the input data that diverge from the background noise. Unsupervised learning does not use the notion of expected output [41]. In GANs, a modern application of unsupervised learning has emerged. Here, one network, called the generator network, creates solution candidates—i.e., maps a vector of latent variables to the solution space—, which are then evaluated by a second network, the discriminator [42]. Ideally, the results of the training process are virtually indistinguishable from the actual solution space, which is the reason GANs are sometimes called "Turing learning." The research focus of ARL is not the generation of realistic solution candidates; this is only a potential extension of the attackers and defenders themselves. ARL, however, describes the general concept of two agents influencing a common model but with different sensors (inputs) and actuators (output) and without knowing of each others presence or actions.

The abstract notion of a model can see multiple instantiations; one such instantiation of ARL would be using a power grid as the model considered by both agents. Ernst et al. employ RL for stability control in power grids [43]. In their paper, they design a dynamic brake controller to damp large oscillations; however, since the reward function is easily well-defined, there is no need for using an ANN for function approximation.

III. Adversarial Resilience Learning

ARL is distinguished from AL by the recurrent structure in which adversary and defender are interacting. While GAN directly connect a generating adversary with a detecting defender, ARL adversary and defender interact only through the system they are using for input and output. In this interaction adversaries are identified as agents inserting disturbances into the system, while defenders provide resilience control.

Definition 1 (Adversarial Resilience Learning (informal))**.** ARL is an experimental structure comprised of two disjoint groups of agents and a system or simulated system. The agents are distinguished as attacker and defender by adhering to conflicting optimization objectives. Both groups of agents receive their input from a, potentially overlapping, set of measurements from the system. They influence the system through two disjunct sets of outputs connected to controls in the simulated system.

A. Fundamental Notation and Model

The basic abstract scenario using ARL consists of two competing agents and a system model. Each of the three elements resembles a state transition. In order to establish a sound formal base, a definition of notation and processes of ARL is provided here. A summary of notations used is given in Table I.

ARL consists of a set of agents, where each agent has a model, denoted by $\mathcal{A}$, and a model of a system, $\mathcal{M}$. The agent model $\mathcal{A}$ serves as a "blue-print" for the actual behavior of a running system; similarly, $\mathcal{M}$ denotes a static model of a world. An index identifies a particular agent model, e. g., $\mathcal{A}_A$ denotes the category of attacker models, $\mathcal{A}_\Omega$ serves to denote the category of defender models. At run-time, the models are instantiated. We denote instances of a model with lower-case letters a, where the superscript denotes a particular state of the model, such as $a^{(t)}$, with t commonly referring a point in simulation time. In the same vein, $m^{(t)}$ denotes an instance of a world model at t.

Each agent tries to maximize its rewards by approximating the agent-specific performance function,

$$p_a\left(m^{(t)}\right) \ . \tag{1}$$

For an agent, the performance function $p_a(\cdot)$ is equal to its reward function in RL terminology. However, the notion of the *performance* function lets us decouple agent behavior from the desired/intended or undesired performance of the world, denoted by

$$p\left(m^{(t)}\right) \ , \tag{2}$$

as the difference between the world's current performance to its nominal performance, p^*.

Agents are categorized through their performance function, an agent model is identified as attacker model $\mathcal{A}_A$ if his reward function p_a behaves inverse to the systems performance. The opposite is true for agents from $\mathcal{A}_\Omega$. Thus, we can define:

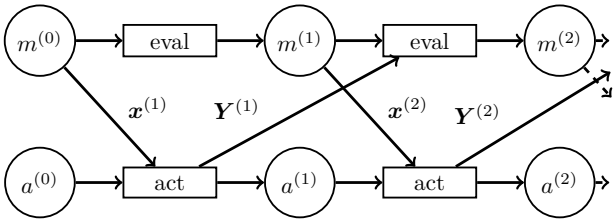


Figure 2. ARL sequence of execution

Definition 2 (Attacker and Defender Classes)**.** For all times t and model instances $m \in \mathcal{M}$, the following provides a classification rule for attackers and defenders:

$$\begin{array}{lll} a \in \mathcal{A}_A & \Rightarrow & p_a\left(m^{(t)}\right) \quad \nsim p\left(m^{(t)}\right) \ , \\ a \in \mathcal{A}_\Omega & \Rightarrow & p_a\left(m^{(t)}\right) \quad \sim p\left(m^{(t)}\right) \ . \end{array} \tag{3}$$

The performance of an agent is tightly coupled to an agent's view of its environment, which can change over time as an agent gains control over more sensors (or looses it). Thus, each $p_a(\cdot)$ can only be defined in terms of the agent's sensory inputs. The portion of the state of a system instance an agent a can observe is denoted by

$$\boldsymbol{x}_a^{(t)} = \psi_a\left(m^{(t)}\right) \ . \tag{4}$$

The agent can act by approximating its reward function $p_a(\cdot)$. This approximation is the agent's activation of its internal dynamic system approximator $\mathrm{act}(\cdot)$; implemented through, e.g., an RNN or DNC, expressed in a mapping such that

$$\mathrm{act}_a : \left(a_t, \boldsymbol{x}_a^{(t)}\right) \mapsto \left(a^{(t+1)}, \boldsymbol{Y}_a^{(t)}\right) \ , \tag{5}$$

where we assume that an agent can choose not to act, just as in a classical RL approach, and where $\boldsymbol{y}$ denotes the probabilities of an agent's action policy, i.e., $\forall y_1, \ldots, y_i, \ldots, y_n, y_i \in [0; 1]$ denotes the probability that the agent uses its ith actuator. In ARL, $\boldsymbol{Y}_a^{(t)}$ denotes a matrix, in which the aforementioned $\boldsymbol{y}$ constitutes the first column vector, and all other elements are set points of the agent's actuators. Each agent defines an action policy for controlling its actuators.

However, this direct mapping of each y_i to an actuator constitutes only the simplest case. In general, an action policy takes on a form that is suitable for the whole action search space, such as a policy network steering a monte carlo tree search as has been shown in [44]. Thus, an agent is acting through the evaluation and application of its system approximator. This happens for each agent individually. In brief, the systems behavior is heavily influenced by the set of all actuators that can be controlled by the respective agents. Thus, an agent does not simply perceive a model (or a part thereof), but the state of the model as the result of all agents acting upon it. Thus, an agent does not simply create an internal representation of a dynamic system, but of a dynamical system-of-systems.

TABLE I. ARL NOTATION.

Symbol	Description
m of $\mathcal{M}$	An instance of a system model
a of $\mathcal{A}$	An instance of an agent model
$\mathcal{A}_A, \mathcal{A}_\Omega$	Attacker model, defender model (Definition 2)
$p(\cdot) \in \mathbb{R}_+$	Performance function
p^*, p^f	Reference performance of normal operation, of failure threshold
$p\left(m^{(t)}\right)$	Overall performance of a system instance m at time t, (2)
$p_a\left(m^{(t)}\right)$	Performance with respect to the objectives of agent instance a at t given the system instance m, (1)
$\psi_a\left(m^{(t)}\right)$	Observation function mapping a system model to the inputs available to agent a, (4)
$\boldsymbol{x}_a^{(t)}$	Inputs to agent a at t, (4)
$\boldsymbol{Y}_a^{(t)}$	Actions y of a at t, (5)

Finally, the simulator evaluates the actions of all agents applied to the world model at t, $m^{(t)}$. This is represented by the evaluation mapping,

$$\mathrm{eval} : \left(\boldsymbol{Y}^{(t)}, m^{(t)}\right) \mapsto m^{(t+1)} \ . \tag{6}$$

Note that if the activation vectors of the participating agents consider a disjoint set of controllers, i.e., the actions application is commutative, the transition of the world state from $m^{(t)}$ to $m^{(t+1)}$ is the result of an aggregation of all agents' actions $\boldsymbol{Y}^{(t)}$. Non-commutative application of actions is out of scope of this work.

B. Formal Definition

Using the notation introduced here and summarized for reference in Table I, we define the concept of ARL as model and connection setup with transition process in the following way.

A setup in ARL is comprised of agents $a_1, a_2, \ldots, a_n$ instantiated from a models $\mathcal{A} \in \{\boldsymbol{A} \cup \boldsymbol{\Omega}\}$ with $|\boldsymbol{A}| > 0$ and $|\boldsymbol{\Omega}| > 0$. Each agent is related to a set of inputs $\boldsymbol{X}_a$ and a set of outputs $\boldsymbol{Y}_a$. Further, the setup requires a world model $\mathcal{M}$ that provides a set of sensors $\boldsymbol{X}_m$ and controls $\boldsymbol{Y}_m$.

The central process of ARL is the dynamic system-of-systems view of a set of agents $a_0, a_1, \ldots, a_n$ acting upon a shared instance of a world model. Activation functions $act_a(\cdot)$ of agents and application $eval(\cdot)$ of agent agents to a world model form a cyclic sequence of activation and application that transforms the states of model and agents into a sequence of states as shown in Figure 2.

An experiment of ARL is the execution of this sequence. The resulting data of an experiment is the sequence of states and outputs as well as the initial setup $m^{(0)}, \boldsymbol{a}^{(0)}$. The vector of evaluations, i.e., states and outputs, $[\mathrm{eval}_a^{(1)}, \ldots, \mathrm{eval}_a^{(t)}, \ldots, \mathrm{eval}_a^{(n)}] \ \ \forall a$, contains the minimal chain of actions necessary to exploit a CPS, iff this is the final result of an ARL execution. Thus, we can finally strive to formalize the idea by collecting all components in a single scenario:

 ISBN: 978-1-61208-713-9

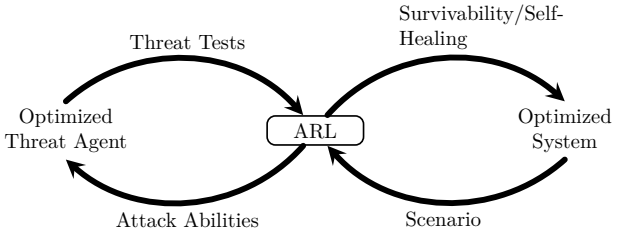


Figure 3. Optimization Objectives

Definition 3 (Adversarial Resilience Learning Scenario)**.** Any experimental setup is comprised of agent instances a of $\mathcal{A}$ of two opposing classes, A and Ω, and a system model $\mathcal{M}$, as well as, for each agent instance a, a reward function $p_a\left(m^{(t)}\right)$, a mapping of observable states $\boldsymbol{x}_a^{(t)}$ and action matrices $\boldsymbol{Y}_a^{(t)}$.

Thus, ARL is the application of RL, as introduced in Section II-B, to iteratively improve the internal decision structure that determines the behaviour of an agent's $act_a(\cdot)$. The output of ARL then is, depending on the exerimenters objectives, an observation of the performance of the system model $\mathcal{M}$ or a set of agents trained towards the defined objectives.

C. Optimization Problem Statement

This section describes possible optimization problems that provide the motivation for ARL.

ARL resembles a closed-loop control situation with (at least) two conflicting controls. Herein are distinguished two different optimization objectives that provide different uses of ARL. The different uses, as depicted in Figure 3, improve different elements to achieve either an improved threat test, or a more resilient system. The primary distinction is between evolving parameters of ANN in order to optimize individual agents or step-wise advancing the structure of the system model. Our concept itself is oblivious to the algorithms used for optimization.

1) System Optimization: The primary objective is to find the inherent control asymmetry of a given control system to finally recommend system designs that favor the defender over the attacker. In control theory this could be expressed as a system, where, for all possible sequences of actions by the attacker for a given system model $\mathcal{M}$, there is at least one corresponding sequence of actions for the defender, and the resulting performance of the system will never drop below a given failure threshold. This requirement can be relaxed by defining a finite measure of failure that may be acceptable, for example during an initiation phase.

The objectives of defender and attacker in control scenarios are focused on system states measured by a model performance function (2), as formally given in Definition 2. In general, we call an agent *defender* if its objective is to keep the performance at least above the failure threshold. We denote an agent as *attacker* if it aims at pushing the performance below a expression for a failure threshold, as seen in Figure 1.

We denote the objective of asymmetry—favouring defence of a system—given a candidate system model instance m and defender agent a_Ω as:

$$p^f < p\left(m^{(t)}\right) \text{ for all } t > t^{(0)} \ . \tag{7}$$

Hence, given any attacker, there exists an (optimal) defender a_Ω^* that ensures that the system performance never falls below a failure threshold p^f. To account for a learning period, we allow for a finite initialization time until $t^{(0)}$. Note that this potentially also excludes black starts. For fully initialized agents competing in a black-start scenario, (7) must hold for all t.

Improvement is achieved by evolutionary changes to the system model $\mathcal{M}$, improved defensive agent models $\mathcal{A}$ or training of defensive agents a_Ω, as discussed in the following section.

2) Agent Training: Training of threat agents aims at improving attack abilities, including the identification of previously unknown attack vectors, in order to provide testing capabilities. Improved threat tests allow to define test requirements for system designs that improve systems resilience against security threats. One objective is to train threat agents that can be used as benchmarks for future system designs.

An agent's objective is implemented through a *reward function* that is used within a RL process that successively improves the agent's behaviour towards that objective.

One particularly surprising success of RL algorithms has been the identification of solutions unthought-of by experts, especially if applied to zero-information initial states. A two-agent, conflicting-objectives game only one potential learning structure usable with ARL. But the concept allows potentially for all combinations of one-or-many zero-information RL agents and static or even human-controlled competition.

IV. Application to Power Systems

Applied to power systems, the performance function is expressed as a diversion from a specified range of acceptable state values. Such state values include voltage, but can also be frequency response in dynamic simulation. The attacker's objective is to force the system to a state where one or more values are outside allowed ranges; its success is measured by the amount and duration of the deviation. The defender has lost the competition if the attacker is able to divert any of the system's parameters beyond the acceptable range. Specific objectives for attackers can vary widely as there are many different parts of a power system that can be affected in order to disrupt service and reduce system performance. Attackers may aim at the demolition of connected machines or components of the transmission and control system. Thus, to strive for a more general specification of objectives, we better consider the objectives of defenders and specify a deviation from these objectives as success for the attackers.

Different specific requirements apply for different parts of the power system, also depending on whether steady-state or dynamic simulation is required. Common parameters

 ISBN: 978-1-61208-713-9

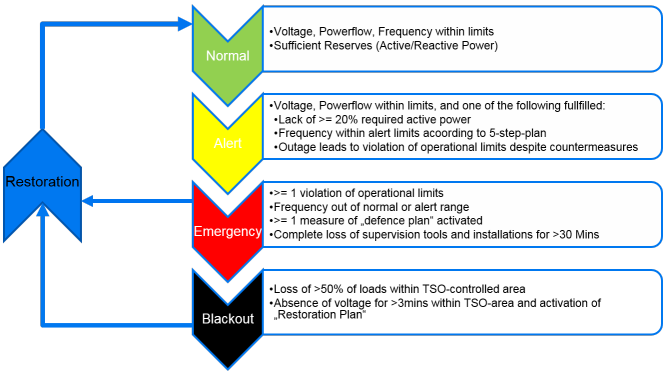


Figure 4. ENTSO-E Operational Phases

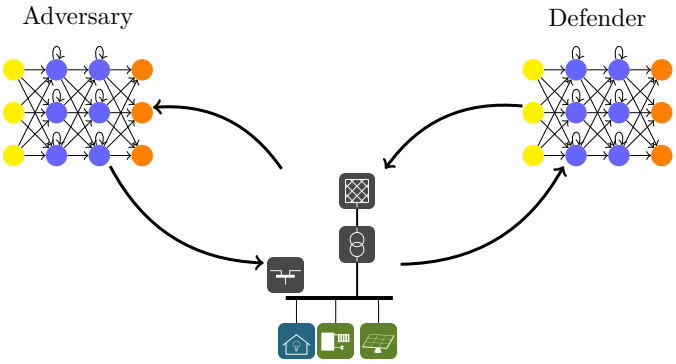


Figure 5. ARL ANN structure

to consider are voltage, frequency and frequency response, and real and reactive power. In general, phase synchronicity is more important for high-voltage transmission grids, as asynchronicity leads to harmonics in the power system, with potentially disastrous large power flows between large segments of the grid. For the European transmission grid, the operation guidelines define conditions for four phases: *normal*, *alert*, *emergency*, and *blackout*, as shown in Figure 4.

Similarly operational parameters exist for medium- and low-voltage grids, power generation and connected loads. DIN EN 50160 specifies parameters for the operation of distribution grids: Acceptable voltages range from $0.9\,\mathrm{pu}$ to $1.0\,\mathrm{pu}$. It is acceptable, by definition in EN 50160, that voltage drops down to at least $0.85\,\mathrm{pu}$ for at most 5% of a week. Frequency must only deviate from the nominal $50\,\mathrm{Hz}$ by at most 4% above or 6% for not more than 0.5% of the year, i.e., less than 2 days overall. Normal operation must deviate no more than $\pm 1\,\%$ [45]. Accordingly, an attacker is successful if any of these values exceeds the defined limits.

Figure 5 shows the refinement of the generic ARL-structure as described in Section III. Both agents interact only through sensors and actuators that influence different controls in the power grid.

In the remainder of this section, we introduce a proof-of-concept implementation of ARL using pandapower [46] for stationary grid simulation and the Keras-RL library [47] for RL algorithms, specifically Deep Q-Learning. First, a brief description of the control scenario is provided, followed by a discussion of the preliminary results.

A. Static Control Scenario

The objective of this proof-of-concept is to show the general feasibility of using (multiple) ANN-heuristics and train them by RL to modify controls in a stationary power system simulation towards their objectives.

The simulation uses a simple medium voltage power grid as model from the grid simulation software pandapower [46]. The grid contains four generators, six loads, and six transformers. We chose to only use voltage as state-indicator and input to the reward of the attacker. The initial configuration of the grid comprises of a stable, healthy state of the grid that would be held up constantly if no control actions would be initiated. Actuators in this scenario are: tap positions, reactive power control, and loads and generation levels as represented by the commonly deployed and future automated controls in power systems.

The reward function for the attacker is shown in Figure 7a. Initial trials pointed towards the inverse of a Poisson Density Function centered on the nominal voltage unit. The reward function thus resembles the objective for an attacker, providing only positive rewards if the mean voltage deviates more than $5\,\%$ from the nominal voltage. The single agent in this demonstration had been assigned direct control of every transformer, generator and load in this scenario.

In terms of optimization from Section III-C, the scenario instantiates m with a single agent $a \in \mathcal{A}_A$ with a parametrized normal distribution,

$$p_a\left(m^{(t)}\right) = -1^{[a\in\mathcal{A}_A]} \exp\left[-\frac{\left(\overline{\psi_a\left(m^{(t)}\right)} - \mu\right)^2}{2\sigma^2}\right] - c\,, \quad (8)$$

where c, μ and σ parametrize the reward curve, $-1^{[a\in\mathcal{A}_A]}$ negates the reward if a is an attacker [48], and $\overline{\psi_a(\cdot)}$ is the arithmetic mean of all inputs. Note that this reward function does not include any information specific to the energy domain. E.g., it treats the difference between $1.0\,\mathrm{pu}$ and $0.8\,\mathrm{pu}$ similar to a reduction to $0.5\,\mathrm{pu}$, even though this would mean a tremendous success to the attacker compared to a reduction to $0.8\,\mathrm{pu}$. This simplification was done deliberately to verify the general feasibility of the ARL concept without explicitly tying it to the energy domain, but to remain useful to any CPS.

B. Demonstrator

In order to show the genearal feasibility of the concept, we implemented a demonstrator for RL in power control scenarios. The current implementation uses static simulation in pandapower [46], supporting free configurability of controlled sensors and actuators of multiple agents, selection of ANN-algorithms and -parameters, as well as different logging and output formats.

 ISBN: 978-1-61208-713-9

In order to support documentation and reproducibility, each experiment is specified within a single configuration file. A experimental configuration defines three major simulation components: a grid model, one or more agents, and a collection of result logs that collect results. At the time of writing, the whole demonstrator is refactored to use the mosaik co-simulation framework [49], [50].

The interconnection between agents and grid simulation, i.e., $\boldsymbol{x}_a$ and $\boldsymbol{Y}_a$ respectively, are separately defined for each agent.

The execution of the simulation is round-based. The rounds are advanced in steps according to a defined evaluation order of agents. Agents are sequentially executed, a defined number of steps each. The grid state is evaluated between each consecutive pair of agent evaluation steps. After each step, RL takes place for each agent individually according to its configuration. Current result monitors output the grid states at every node of the grid into a grid-state log. The results are graphically evaluated as is discussed in Section IV-C below.

C. Results

To show the usability of our demonstrator, we pitched two simple agents with inverse reward functions (Figure 7a and Figure 7d) against each other, using the example grid shown in Figure 6a as an arena. Both agents were assigned all voltage sensors as input. The attacker was assigned control of all tap changers, representing a scenario where a vulnerability in one type of controller was exploited. The defender would be granted access to all generators and loads in this scenario. This was a deliberate choice in order to force the defender to develop a strategy that involved all generators and loads; in a reverse scenario, control of the tap changers would allow the defender to act easily against a series of attacker actions and would require a more sophisticated experiment setup involving, e.g., a digital twin in the attacker code for decoupled RL training for a devastating one-shot attack.

Figure 6 shows a late state of the simulation. Seemingly, the attacker gained the upper hand and has been able to increase voltage levels beyond $1.05\,\mathrm{pu}$. The grid representation in Figure 6a shows that especially two central sensors (numbered 4 and 3) are stuck with very high voltage levels, represented by the length of the bars rooted at the nodes, most likely sufficient for the connected loads to shut down or be damaged. The mean voltage level of the system, depicted for steps 1900 to 2000 in Figure 6b, shows that even the lower voltages of other nodes are not sufficient to lower the mean voltage to acceptable levels. Thus, in this example, the attacker has been able to destabilize the grid, despite the efforts of the defender.

Evaluating the two agents in Figure 7 provides no immediately conclusive cause for the loss of the defender. The cumulative number of positive rewards in Figure 7b for the attacker and Figure 7e, show only small differences. These asymmetries might be explained by the order of execution, where the defender always acts in response to the attacker. The current reward for the depicted step in the simulation, depicted in Figure 7a and Figure 7d, shows that the defender is evaluating a different mean voltage than the attacker. As rewards are calculated after the actions of an agent, thus these graphs show the results of two actions that both improved the performance towards their own objectives.

The effect of the ARL structure of competing agents that is beneficiary for RL algorithms becomes apparent in positive learning curves for both agents (Figure 7b and Figure 7e). In preliminary tests with a lone attacker, the learning process first went through a lengthy phase where only little positive rewards were achieved.

V. Conclusion and Future Work

This work introduced Adversarial Resilience Learning (ARL), a novel approach to analyze cyber-physical systems (CPSs) through competitive situations in highly-complex systems using self-improving agents. This work is motivated by the need to find better methods to evaluate the behaviour of CPSs under threat of maliciously acting, intelligent threat agents. The main idea is that groups of agents struggle to enforce their objectives against agents with conflicting goals.

Pitching two—or more—Reinforcement Learning (RL) agents with conflicting reward functions against each other may allow to define more realistic tests for adversarial or competitive situations. It harbours the promise of finding novel strategies for both attack and defense, which both can be used to strengthen the resilience of systems during the design and testing phase of a power system or individual components. ARL-based analysis should contribute to building grid structures that are more resilient to attacks and train both artificial and human operators in better handling of security incidents.

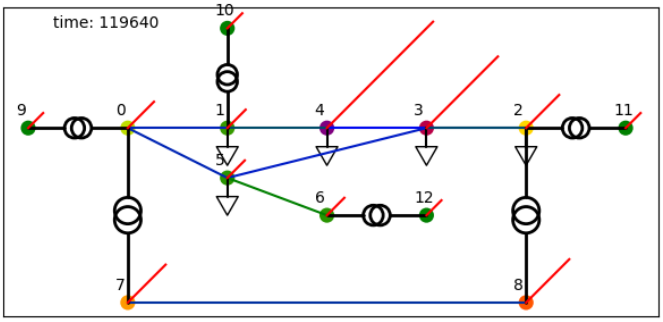
Generally, the concept may allow to estimate threat-related indices, for example the maximum amount of control that an adversary may be allowed to gain over a system, which leads to improved and more effective recommendations for security directives and risk mitigations.

The concept of ARL and its ongoing implementation in the ARL-Demonstrator only marks the starting point for in-depth research on structural asymmetries of complex systems and protection against learning threat agents. The demonstrator provides the abilities to further research in a number of interesting directions.

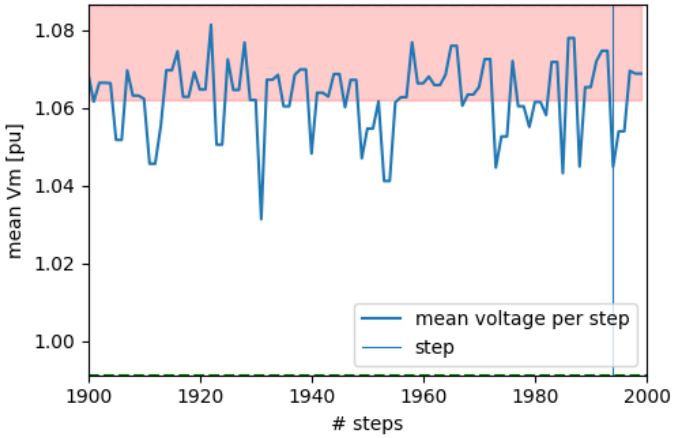
Foremost, this is the analysis of structural resilience of complex systems, especially finding minimum control sets of critical components that provide the most defensive capabilities, or estimates of the structural strength of a system. The integration into our co-simulation framework mosaik opens up the possibility of extending the single system into a whole composition into an interdependent system-of-systems. In the energy domain, the introduction of communications infrastructure (SCADA, CDMA450, etc.) is necessary.

Deeper extensions of the demonstrator itself will involve capabilities of the defender to affect structural changes to the system. This would allow to use RL to identify novel and more resilient structures. The dual ability for threat agents would be the extension of control, i.e., simulation of further compromise

 ISBN: 978-1-61208-713-9

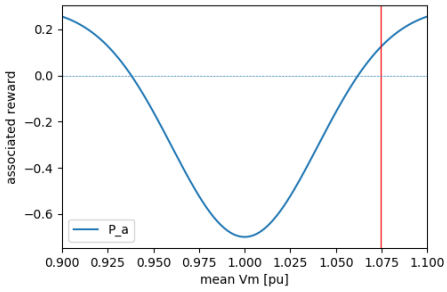


(a) Grid Simulation

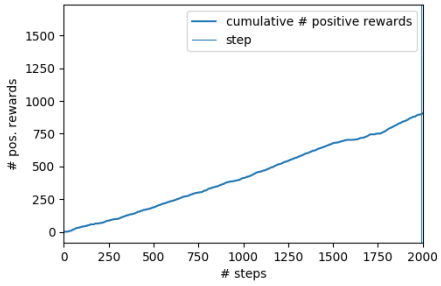


(b) Grid performance (mean voltage) over time/evaluation steps

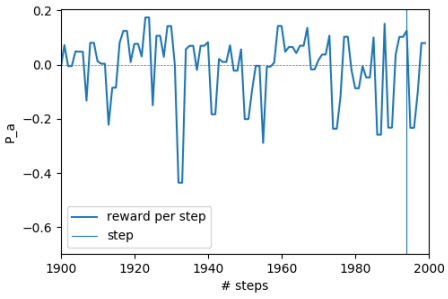
Figure 6. Proof-of-concept ARL grid results



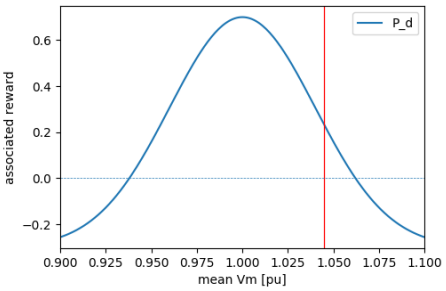
(a) Attacker reward function, based on mean voltage



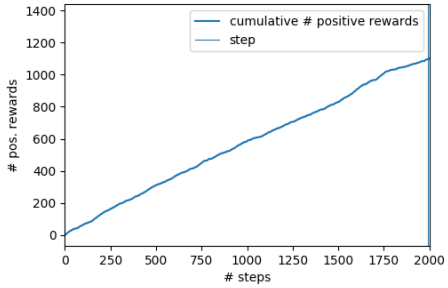
(b) Cumulative number of positive rewards over training steps



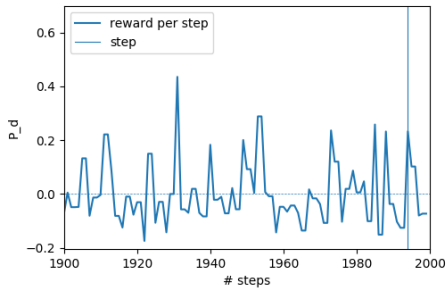
(c) Attacker performance $p_a(m^{(t)})$ over time (inverse mean voltage)



(d) Defender reward function, based on mean voltage



(e) Cumulative number of positive rewards over training steps



(f) Defender performance $p_a(t)$ over time (mean voltage)

Figure 7. Proof-of-concept ARL agent results

from within a system. Both activities require the introduction of a measure of cost to the demonstrator.

Further, this demonstrator allows to analyze simulated systems from the point of view of threat agents, by pitching the agent against novel security measures, for example simulation of distributed coordinated attacks. Combining this view with multi-domain scenarios would enable analysis of sophisticated, multi-level attack techniques that involve, for example information hiding or emission of misleading information by attacker or defender. That means finding novel ways of attack using a combination of illegal and legal operations and interdependencies between different systems. Consequentially, all these approaches would lead to the development of improved designs and testing methods for highly complex systems.

We can only assume that this finally leads to more resilient designs and defensive adaptable strategies—and, in the end, to improvements for the security of supply, but at this stage of the work, the first results are very satisfying.

VI. Acknowledgements

The authors would like to thank Sebastian Lehnhoff for his counsel, strategic advice and for providing extensive compute power for the first experiments coducted with this initial concept.

 ISBN: 978-1-61208-713-9

REFERENCES

[1] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[2] N. Beach-Westmoreland, J. Styczynski, and S. Stables, "When the lights went out: Ukraine cybersecurity threat briefing," *Booz Allen Hamilton*, vol. 12, p. 20, 2016.

[3] A. Prentice, "Ukrainian banks, electricity firm hit by fresh cyber attack," *Reuters*, June, 2017.

[4] R. Arghandeh, A. Von Meier, L. Mehrmanesh, and L. Mili, "On the definition of cyber-physical resilience in power systems," *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 1060–1069, May, 2016.

[5] I. Linkov and A. Kott, *Fundamental Concepts of Cyber Resilience: Introduction and Overview*. Cham: Springer International Publishing, 2019, pp. 1–25.

[6] R. Alur, *Principles of cyber-physical systems*. MIT Press, 2015.

[7] E. Böde *et al.*, "Design paradigms for multi-layer time coherency in adas and automated driving (multic)," in *FAT-Schriftenreihe 302*, 302nd ed., ser. FAT-Schriftenreihe. Forschungsvereinigung Automobiltechnik e.V. (FAT), October, 2017.

[8] G. Brown, M. Carlyle, J. Salmerón, and K. Wood, "Defending critical infrastructure." *Interfaces*, vol. 36, no. 6, pp. 530–544, 2006.

[9] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. John Wiley and Sons, 1965.

[10] C. T. Do *et al.*, "Game theory for cyber security and privacy," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 30:1–30:37, May, 2017.

[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[14] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[15] D. R. Seidl and R. D. Lorenz, "A structure by which a recurrent neural network can approximate a nonlinear dynamic system," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. ii, Jul 1991, pp. 709–714 vol.2.

[16] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of computer and system sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[20] H. Tong, *Non-linear time series: a dynamical system approach*. Oxford University Press, 1990.

[21] A. Basharat and M. Shah, "Time series prediction by chaotic modeling of nonlinear dynamical systems," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 1941–1948.

[22] W. Yu, G. Chen, J. Cao, J. Lü, and U. Parlitz, "Parameter identification of dynamical systems from time series," *Physical Review E*, vol. 75, no. 6, p. 067201, 2007.

[23] B. Cessac, "A view of neural networks as dynamical systems," *International Journal of Bifurcation and Chaos*, vol. 20, no. 06, pp. 1585–1629, 2010.

[24] L. C. Baird, "Reinforcement learning in continuous time: Advance updating," in *IEEE World Congress on Computational Intelligence, 1994 IEEE International Conference on Neural Networks*, 1994.

[25] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[26] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[27] M. Minsky, "Neural nets and the brain-model problem," *Unpublished doctoral dissertation, Princeton University, NJ*, 1954.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] T. Dozat, "Incorporating nesterov momentum into adam," *Phys. Rev.*, 2016.

[31] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[32] M. P. Cuéllar, M. Delgado, and M. Pegalajar, "An application of non-linear programming to train recurrent neural networks in time series prediction problems," in *Enterprise Information Systems VII*. Springer, 2007, pp. 95–102.

[33] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[34] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 296–305.

[35] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[36] M. Ruppert, E. M. Veith, and B. Steinbach, "An evolutionary training algorithm for artificial neural networks with dynamic offspring spread and implicit gradient information," in *Proceedings of the Sixth International Conference on Emerging Network Intelligence (EMERGING 2014)*. International Academy, Research, and Industry Association (IARIA), 2014.

[37] E. M. Veith, *Universal Smart Grid Agent for Distributed Power Generation Management*. Logos Verlag Berlin GmbH, 2017, ch. Forecasting Power Demand and Supply, pp. 100–108.

[38] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[39] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[40] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *CoRR*, 2015.

[41] Z. Ghahramani, "Unsupervised learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 72–112.

[42] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[43] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: reinforcement learning framework," *IEEE Transactions on Power Systems*, vol. 19, no. 1, pp. 427–435, feb 2004.

[44] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, p. 354, 2017.

[45] "DIN EN 50160:2000-03: Merkmale der Spannung in öffentlichen Elektrizitätsversorgungsnetzen," 2008.

[46] L. Thurner *et al.*, "pandapower - an open source python tool for convenient modeling, analysis and optimization of electric power systems," *IEEE Transactions on Power Systems*, 2018.

[47] M. Plappert, "keras-rl," https://github.com/keras-rl/keras-rl, 2016, [Retrieved: 2019-04-23].

[48] K. E. Iverson, "A programming language," in *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, ser. AIEE-IRE '62 (Spring). New York, NY, USA: ACM, 1962, pp. 345–351.

[49] S. Schütte, "Simulation model composition for the large-scale analysis of smart grid control mechanisms," PhD, Carl von Ossietzky University of Oldenburg, 2013.

[50] S. Lehnhoff *et al.*, "Exchangeability of power flow simulators in smart grid co-simulations with mosaik," in *2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES*, 2015.

 ISBN: 978-1-61208-713-9