Carl von Ossietzky
Universität
Oldenburg

Play with me

Practical Autocurriculum Deep Reinforcement Learning for Resilient Operation of Cyber-Physical Energy Systems

Eric MSP Veith <eric.veith@uol.de> , 2023-03-10

Carl von Ossietzky
Universität
Oldenburg

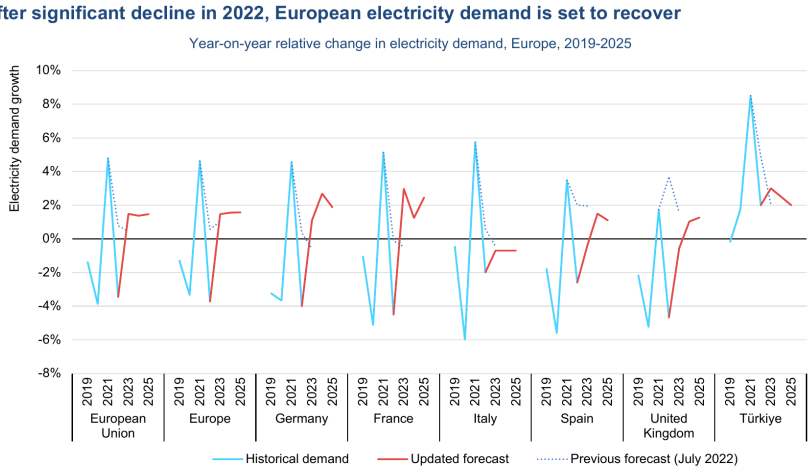
% whoami

- Eric MSP Veith <eric.veith@uol.de>
- Currently head of a junior research group at University of Oldenburg, Germany
- Computer scientist by heart: First ICT, then distributed heuristics, then Multi-Agent Systems, now advanced Deep Reinforcement Learning
- PhD in 2017: “Universal Smart Grid Agent for Distributed Power Generation Management.”
- Creator of the Adversarial Resilience Learning methodology (advanced DRL in CNIs)

Motivation

Carl von Ossietzky
Universität
Oldenburg

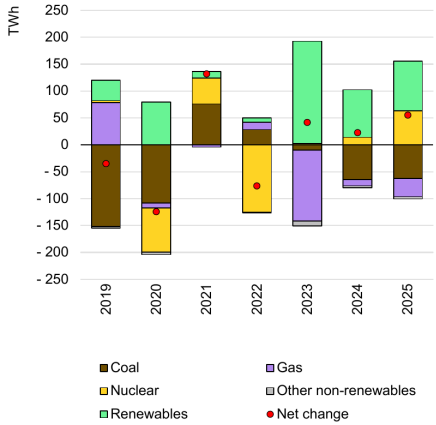
Electricity Demand Rising



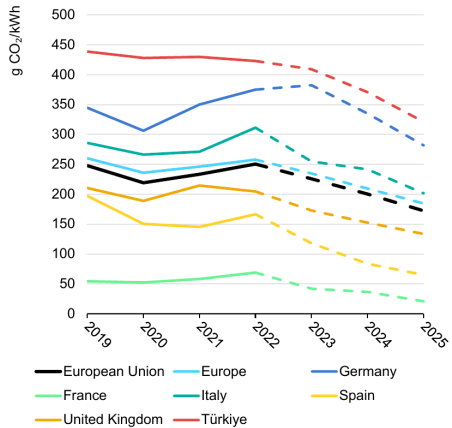
Renewables Are Replacing Fossil Fuels

Following two years of increases, CO_2 intensity starts to decline again from 2023 onward

Year-on-year change in electricity generation, European Union, 2019-2025



Development of average CO_2 intensity, Europe, 2019-2025



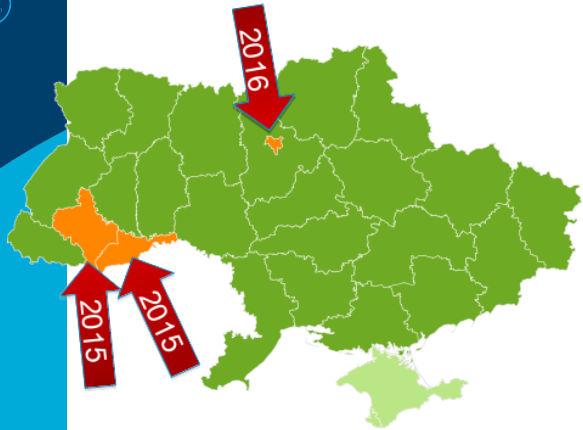
IEA. CC BY 4.0.

Reaching Climate Goals

- CO_2 reduction a necessity
- Options: Increase share of renewables (often wind, PV!), increase efficiency
- Approaches to both include, e. g., distributed heuristics and new energy market concepts
- Think: Multi-agent systems, ICT, direct-to-market of DERs, transactive energy, …

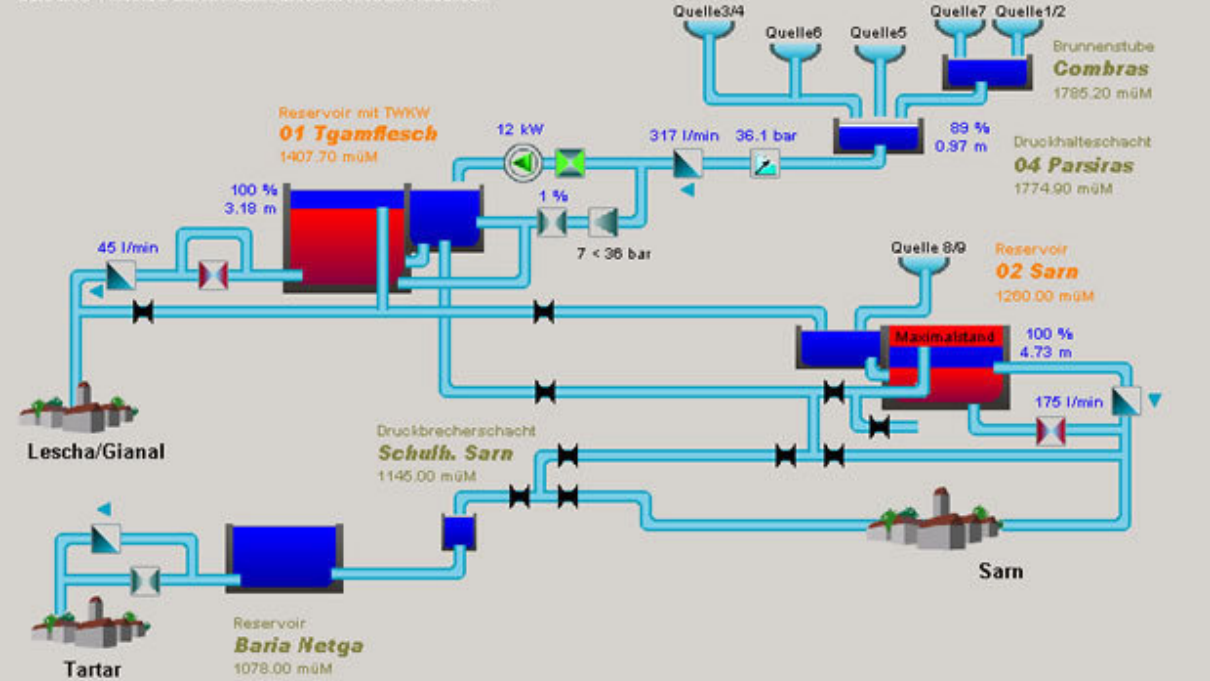

“There are only two types of companies:
those who have been hacked,
and those who don’t yet know
they have been hacked.”
— John T. Chambers

Energy Systems Fit The Bill Just As Well



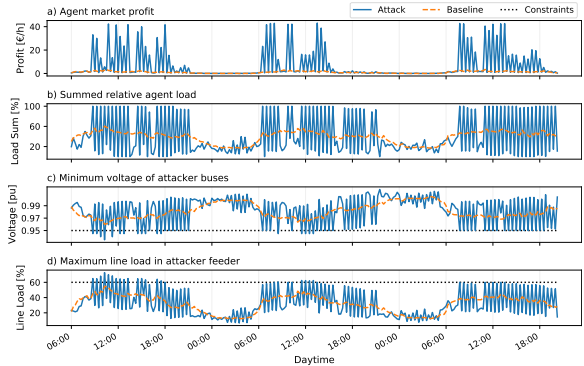
Dec 23rd, 2005

- **Cyber attack** causes blackout in the Ukraine
- **3 DSOs** targeted
- **High level of automation** helps attackers
- Operative intrusion in **OT**; disconnection of **several substations**
- Several months in preparation


Quelle3/4
Quelle6
Quelle5
Quelle7
Quelle1/2
Brunnenstube
Combras
1785.20 müM
Reservoir mit TWKW
01 Tgamflesch
1407.70 müM
12 kW
317 l/min
36.1 bar
89 %
0.97 m
Druckhalteschacht
04 Parsiras
1774.90 müM
100 %
3.18 m
1 %
7 < 36 bar
45 l/min
Quelle 8/9
Reservoir
02 Sarn
1260.00 müM
Maximalstand
100 %
4.73 m
175 l/min
Lescha/Gianal
Druckbrecherschacht
Schulh. Sarn
1145.00 müM
Sarn
Reservoir
Baria Netga
1078.00 müM
Tartar

Transactive Energy Can Be Gamed

- Economic and control techniques, based on market standard values
- There is no “sound” market design yet than cannot be gamed
- Worse yet: Agents can find weaknesses & gain market dominance without system knowledge

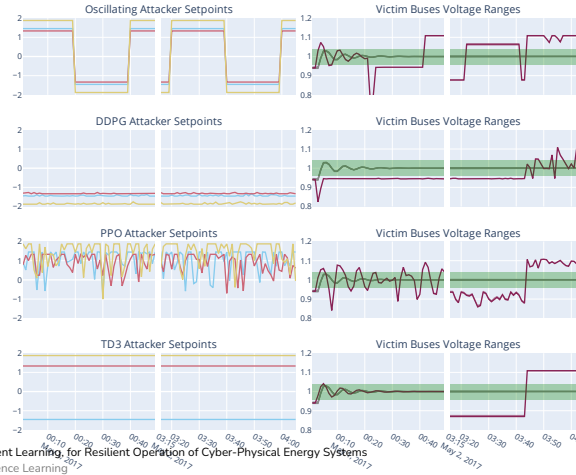


Agents learn to “game” local energy markets

Wolgast, Veith, and Nieße [2]

Attacks from Physical Nodes

- DRL-based attacker learns to violate voltage controllers
- Leverages control schemes to act against each other



Energieversorgung in Deutschland

Stromhändler zocken fast bis zum Blackout

Der deutsche Strommarkt stand in den vergangenen Tagen mehrfach vor dem Zusammenbruch. Laut Bundesnetzagentur waren dafür aber nicht die Kälte oder der Atomausstieg verantwortlich, sondern Energiehändler - die offenbar ihre Profite maximieren wollten. Die Aufsichtsbehörde ist alarmiert.

dapd

Stromnetz in Deutschland: Starke Preisschwankungen durch hohe Nachfrage

Systemstabilität - Minutenreserve

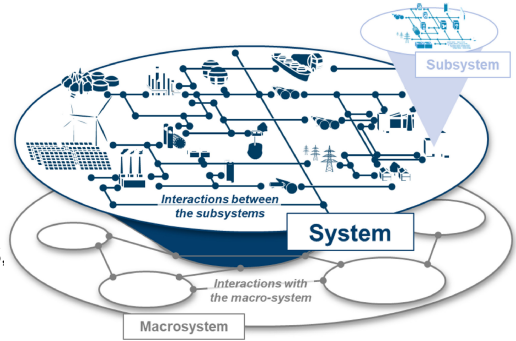
Abgerufene Menge (+)
Abgerufene Menge (-)
Arbeitspreis (+)
Arbeitspreis (-)
Vorgehaltene Menge (+)
Vorgehaltene Menge (-)
Leistungspreis (+)
Leistungspreis (-)

The Adversary

- Consumer behavior (prosumers), VPP, outages, weather effects: probabilistic modelling
- **DERs**
  - Prognosis deviations
  - VPP, direct marketing: highly non-deterministic
- **Terrorist**
  - Goal: Demolition
  - No route back needed in som cases
  - No sophisticated tactics necessary
- **Military**
  - Goal: destruction & takeover
  - Damage to CNIs is mostly collateral damage (or explicitly wanted)
  - Usually, CNIs are “don’t care,” but should be usable afterwards
- **Businnesspeople**
  - Goal: (short-term) profit maximization
  - Damage to CNIs unexpected (or simply “don’t care”)
  - Usues loopholes and grey areas in codices

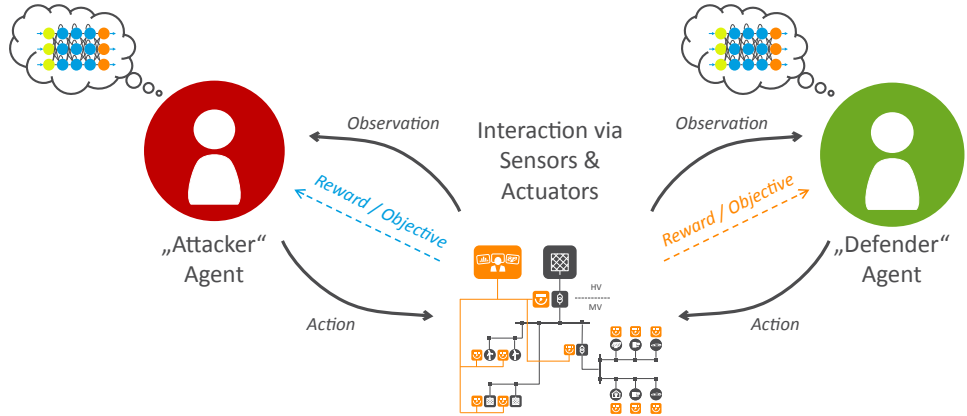
Learning Resilient Control

- **Interconnected CPS have always attack surface due to their inherent complexity**
- Low latency of ICT and OT
- High interdependence
- Complexity in breadth and depth
- Cricital Services as SPOF (DNS, BGP, SCADA, SDL)
- **Learning Stratgies for automatic issue mangement**
- “Adversarial Resilience Learning”

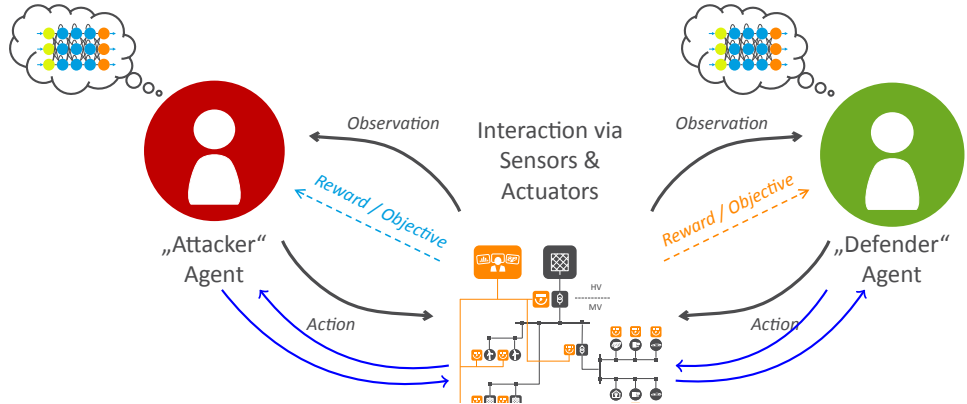


Kotzur, Leander, et al. “A modeler’s guide to handle complexity in energy systems optimization.” Advances in Applied Energy 4 (2021): 100063.

Adversarial Resilience Learning



Adversarial Resilience Learning



Adversarial
System-of-Systems Reinforcement Learning

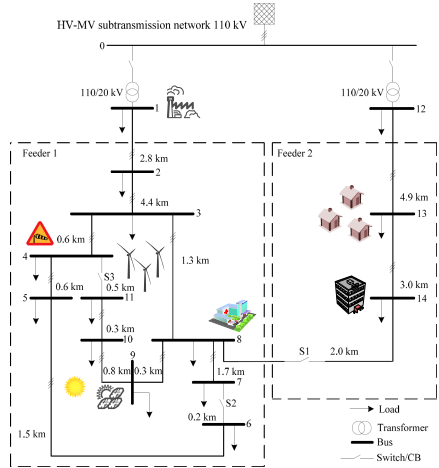
Deep Reinforcement and Complex Environments

MIDAS

Multi-Domain Test Scenario

Time Series:
- Wind
- Solar Irradiation

Asset/Unit Models:
- Wind Parks
- PV Power Plants
- Coal Power Plants
- Block & Heat PPs
- Batteries



Power Grid:
- Physical Model
- Power Flow
- Grid Code/Constraints

Load Profiles
- Households
- Commercial Buildings
- Hotels
- Hospitals

Software/Technology Decision

- Option A: Complex, monolithic or modular simulation software
  - Vendor availability? Vendor lock-in?
  - Feature complete?
  - Composability of scenarios?
  - Interface (e. g., for DRL given)?
- Option B: Co-simulation
  - PandaPower: Power grid model, load flow calcuation
  - MIDAS-powergrid: Constraints, grid codes, load profiles
  - PySimMods: Models of PV, windparks, power plants, etc.
  - MIDAS (core): Scenario definition, set up co-simulation

Getting Started

1. Install from PyPI:

```
% pip3 install midas-mosaik
% midasctl --help    # Should give the help
```

2. Set up MIDAS and download data:

```
% midasctl configure --autocfg
% midasctl download
```

3. Run a simple demo:

```
% midasctl run demo
% ls _outputs
midasmv_auto_script.py  midasmv_cfg.yml  midasmv.hdf5
% midasctl analyze _outputs/midasmv.hdf5
% ls _outputs/midasmv
midasmv-Powergrid_0_report.md  midasmv-Powergrid_0_report.pdf  Powergrid_0
```

(Simple) Analysis

Analysis of midasmv-Powergrid__0

Summary

- bus health: 100.00 %
- active energy sufficiency: 0.00 %

Demand and Supply

- total active energy demand: 58.60 MWh
- total active energy supply: 0.00 MWh or about 0.00 full load hours
- extg. active energy supply: 59.52 MWh
- total reactive energy demand: 28.22 MVArh
- total reactive energy supply: 0.00 MVArh
- extg. reactive energy supply: 22.74 MVArh
- total apparent energy demand: 65.04 MVAh
- total apparent energy supply: 0.00 MVAh
- extg. apparent energy supply: 63.71 MVAh

Bus Analysis

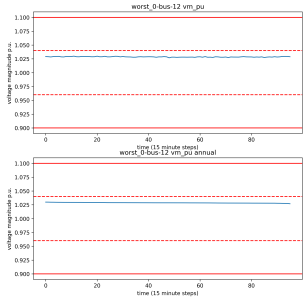


Figure 1: worst_0-bus-12_Bus_vm_pu

Carl von Ossietzky
Universität
Oldenburg

(Simple) Analysis

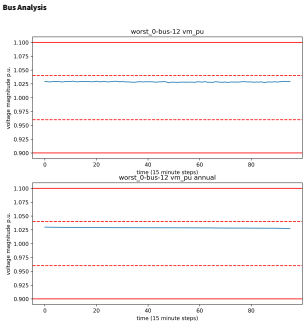


Figure 1: worst_0-bus-12_Bus_vm_pu

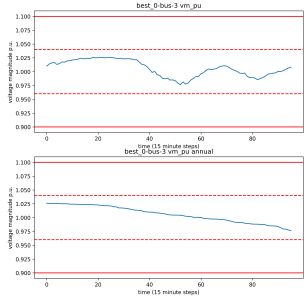


Figure 2: best_0-bus-3_Bus_vm_pu

(Simple) Analysis

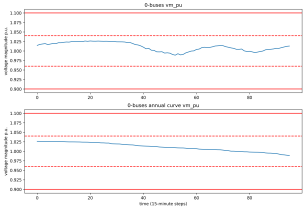


Figure 3: Total_Bus_vm_pu

ExtGrid Analysis

ExtGrid	Min (MW)	Max (MW)	Mean (MW)	Std (MW)	Sum (MW)	Energy (MWh)
0-ext_grid-0	0.699	4.627	2.480	1.093	238.064	59.516
total	0.699	4.627	2.480	1.093	238.064	59.516

ExtGrid	Min (MVAr)	Max (MVAr)	Mean (MVAr)	Std (MVAr)	Sum (MVAr)	Energy (MVArh)
0-ext_grid-0	0.021	2.142	0.947	0.583	90.951	22.738
total	0.021	2.142	0.947	0.583	90.951	22.738

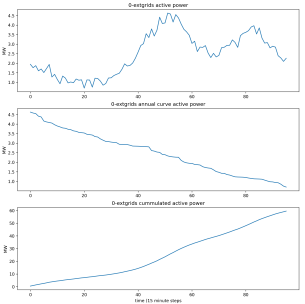


Figure 4: Total_ExtGrid_P

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Begin the scenario:

```
my_first_midas:
  modules: [store, powergrid, sndata]
  end: 1*24*60*60
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Add data store for analysis:

```
my_first_midas:
  modules: [store, powergrid, sndata]
  end: 1*24*60*60
  store_params:
    filename: mymidasdb.hdf5
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Select power grid:

```
my_first_midas:
  modules: [store, powergrid, sndata]
  end: 1*24*60*60
  store_params:
    filename: mymidasdb.hdf5
  powergrid_params:
    my_grid:
      gridfile: midasmv
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Add load profiles:

```
my_first_midas:
  modules: [store, powergrid, sndata]
  # ...
  sndata_params:
    my_grid:
      land_mapping:
        # Bus: [ [ Load Profile, Scaling ] ]
        1: [[0, 1.0], [2, 1.0], [3, 2.0], [6, 2.0], [7, 1.0]]
        3: [[2, 1.0], [3, 1.0], [6, 1.0], [7, 1.0]]
        4: [[0, 2.0], [3, 2.0], [7, 1.0]]
        5: [[3, 2.0], [7, 1.0]]
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Add more loads:

```yaml
my_first_midas:
  modules: [store, powergrid, sndata, comdata]
  # ...
  comdata_params:
    my_grid:
      interpolate: true
      randomize_data: true
      noise_factor: 0.2
      mapping:
        13: [[SuperMarket, 0.089]]
        14: [[SmallHotel, 0.022]]
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

Finally, add weather data…

```yaml
my_first_midas:
  modules: [store, powergrid, sndata, comdata, weather, der]
  start_date: 2017-01-01 00:00:00+0100
  # ...
  weather_params:
    my_weather_station:
      weather_mapping:
        WeatherCurrent: ["interpolate": true]
```

First Custom Scenario

- CIGRÉ MV grid
- Loads with time series
- DERs with weather data

… and DERs:

```
my_first_midas:
  # ...
  der_params:
    my_grid_pv:
      grid_name: my_grid
      mapping:
        3: [[PV, 3], [PV, 1]]
        7: [[PV, 1]]
        8: [[PV, 2]]
        14: [[PV, 2], [PV, 2]]
      weather_provider_mapping:
        PV: [my_weather_station, 0]
```

palaestrAI

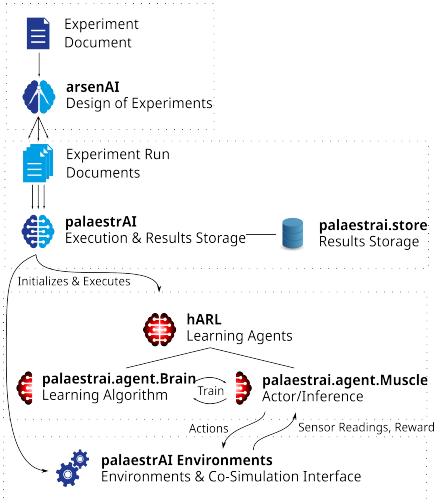
palaestrAI

> *A palaestra was any site of an ancient Greek wrestling school. Events requiring little space, such as boxing and wrestling, took place there. Palaestrae functioned both independently and as a part of public gymnasia; a palaestra could exist without a gymnasium, but no gymnasium existed without a palaestra.*

Wikipedia: Palaestra

The palaestra at Olympia, Greece, by Chris Kar

A Software Stack Approach



palaestrAI Setup

```
% pip3 install 'palaestrai[full]'
% palaestrai --version
% palaestrai runtime-config-show-effective
% palaestrai database-create
```

MIDAS-palaestrAI interface

- palaestrAI is “a training ground for autonomous agents.”
- Agents have *sensors* and *actuators*
  - Possible sensors: Voltage magnitude, line load, etc.
  - Possible actuators: Reactive power control, real power control, etc.
- MIDAS can offer sensors and actuators for palaestrAI:
  Add `with_arl: true` to a scenario (toplevel)
- Afterwards, MIDAS creates sensor/actuator descriptions with `midasctl run`
  E. g.,

```
{action_space: 'Box(low=-4.444444444444445, high=4.444444444444445,
                    shape=(), dtype=np.double)',
 actuator_id: PysimmodsPV-0.Photovoltaic-0.q_set_mvar},
{space: 'Box(low=0.0, high=1.2, shape=(), dtype=np.float32)',
 uid: Powergrid-0.0-bus-1.vm_pu}
```

Purpose of palaestrAI: Sound Experimentation

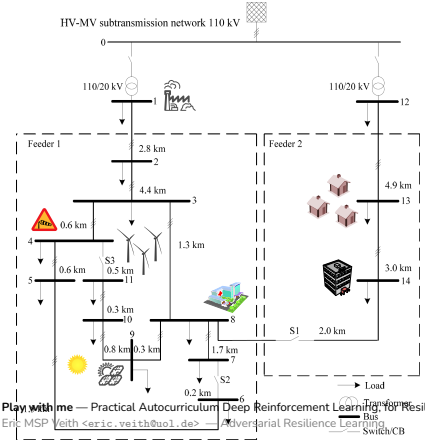
- Learning agents simulation in CPES not just a software logistics problem!
- Sound experimentation: Reproducibility, results storage, …
- palaestrAI driven by **experiment definitions**
  - Global configuraton
    - Name
    - Random seed
  - Definitions
    - Agents: Name, algorithm, hyperparameters, utility functions
    - Environments: software, parameters, reward functions
    - Sensor/actuator sets
    - Phase configurations: Training or test mode, number of repetitions
    - Simulation configuration: Execution strategy, termination conditions (invariants to check against)
  - Phases
    - Defining possible combinations of agents with sensors/actuators in environments, with particular configurations
    - Establishes DoE sample space

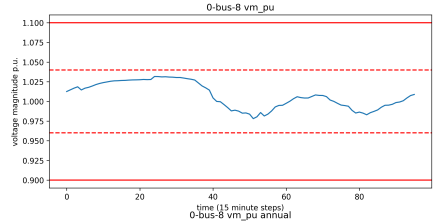
First Experiment: Simple Voltage Control

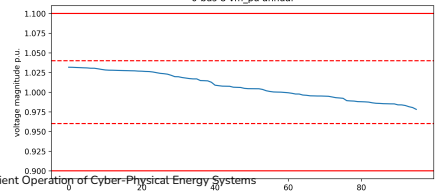
- Q Controller not a learning agent, but implementation of a simple formula:

$$\boldsymbol{q}(t+1) = [\boldsymbol{q}(t) - \boldsymbol{D}(\boldsymbol{V}(t) - 1)]^{+} \ ,$$

- Let's add one to bus no. 8







Find sensor/actuator assignments

- In MIDAS scenario configuration: PV on bus 8 has index 3

```yaml
peak_mapping:
    3:
        - [PV, 4]  # 0
        - [PV, 1]  # 1
    7:
        - [PV, 2]  # 2
    8:
        - [PV, 4]  # 3 <--
    14:
        - [PV, 3]
        - [PV, 2]
```

- Corresponding actuator is `PysimmodsPV-0.Photovoltaic-3.q_set_mvar`
- Corresponding sensor is `Powergrid-0.0-bus-8.vm_pu`

Find sensor/actuator assignments

- In MIDAS scenario configuration: PV on bus 8 has index 3
- Corresponding actuator is `PysimmodsPV-0.Photovoltaic-3.q_set_mvar`

```
{action_space: 'Box(low=-4.444444444444445, high=4.444444444444445,
                    shape=(), dtype=np.double)',
 actuator_id: PysimmodsPV-0.Photovoltaic-3.q_set_mvar}
```

- Corresponding sensor is `Powergrid-0.0-bus-8.vm_pu`

```
{space: 'Box(low=0.0, high=1.2, shape=(), dtype=np.float32)',
 uid: Powergrid-0.0-bus-8.vm_pu}
```

Experiment Definition I

```yaml
%YAML 1.2

---
uid: Voltage Experiment
seed: 20240310
version: 3.5.2
output: palaestrai-runfiles
repetitions: 1
max_runs: 1
definitions:
  environments:
    midasmv:
      environment:
        name: palaestrai_mosaik:MosaikEnvironment
        uid: midas_powergrid
```

Experiment Definition II

```yaml
params:
  module: midas.tools.palaestrai:Descriptor
  description_func: describe
  instance_func: get_world
  arl_sync_freq: &step_size 60
  end: &end 24*60*60
  silence_missing_input_connections_warning: True
  params:
    name: palaestrai_ts
    config: midas-scenarios/tutorial.yml
    end: *end
    step_size: *step_size
    start_date: random
    mosaik_params: {addr: [127.0.0.1, 5678]}
    store_params:
      buffer_size: 1000
      keep_old_files: true
```

Experiment Definition III

```
            filename: voltage-experiment.hdf5
      reward:
        name: midas.tools.palaestrai.rewards:ExtendedGridHealthReward
  agents:
    q_controller:
      name: Reactive Power Controller
      brain:
        name: palaestrai.agent.none_brain:NoneBrain
        params: {}
      muscle:
        name: midas.tools.palaestrai:ReactivePowerMuscle
        params:
          sensor_actuator_mapping:
            0-bus-8.vm_pu: Photovoltaic-3.q_set_mvar
      objective:
        name: midas.tools.palaestrai:ArlDefenderObjective
        params: {}
```

Experiment Definition IV

```yaml
sensors:
  bus8:
    midas_powergrid:
      - midas_powergrid.Powergrid-0.0-bus-8.vm_pu
      - Powergrid-0.0-bus-8.in_service
actuators:
  bus8:
    midas_powergrid:
      - PysimmodsPV-0.Photovoltaic-3.q_set_mvar
simulation:
  ttsc:
    name: palaestrai.simulation:TakingTurns
    conditions:
      - name: palaestrai.experiment:EnvironmentTerminationCondition
        params: {}
phase_config:
  training:
```

Experiment Definition V

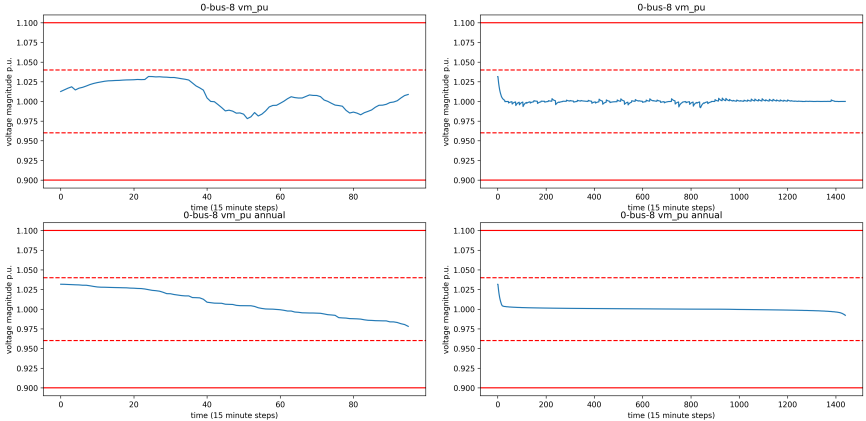
```
      mode: train
      worker: 1
      episodes: 1
    test:
      mode: test
      worker: 1
      episodes: 1
  run_config:
    condition:
      name: palaestrai.experiment:VanillaRunGovernorTerminationCondition
      params: {}

schedule:
  - reactive_power_control:
      environments: [[midasmv]]  # one factor, one level
      agents: [[q_controller]] # one factor, one level
      simulation: [ttsc] # one factor, one level
```

Experiment Definition VI

```yaml
phase_config: [test] # one factor, one level
sensors:
  q_controller: [bus8]
actuators:
  q_controller: [bus8]
```

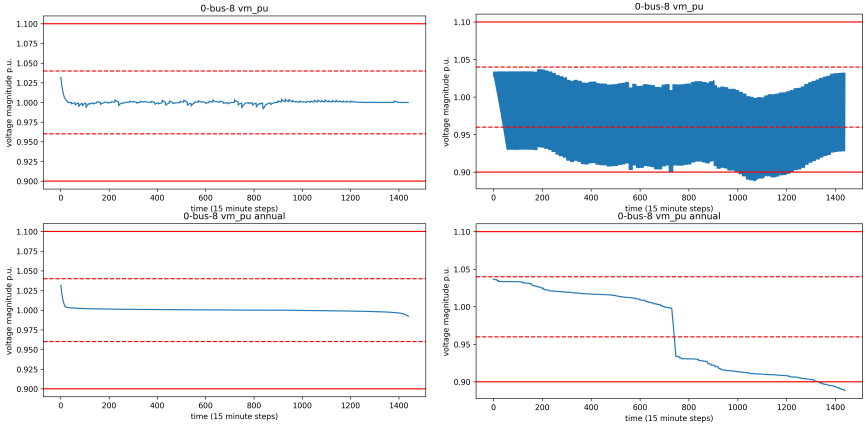
Results


0-bus-8 vm_pu
voltage magnitude p.u.
time (15 minute steps)
0-bus-8 vm_pu annual
voltage magnitude p.u.
time (15 minute steps)

Now Add an Evil Attacker

```yaml
evil_attacker:
  name: Sauron
  brain:
    name: harl:SACBrain
    params:
      fc_dims: [4, 4]
      update_after: 120
      update_every: 10
      batch_size: 64
  muscle:
    name: harl:SACMuscle
    params:
      start_steps: 120
  objective:
    name: midas.tools.palaestrai:ArlAttackerObjective
    params: {}
```

From “All is Well” to “System Under Attack” in 20 Lines of YAML



Adversarial Resilience Learning

Value of an Action & Entropy Bonus I

- Entropie and the Bellman Equation:

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{s'\sim P, a'\sim\pi}\Bigg[\underbrace{R(s,a,s')}_{\text{Next States}} + \gamma\Big(\underbrace{Q^{\pi}(s',a')}_{\text{Next Actions}} + \alpha H\left(\pi(\cdot|s')\right)\Big)\Bigg] \tag{1}$$

$$= \mathop{\mathbb{E}}_{s'\sim P, a'\sim\pi}\left[R(s,a,s') + \gamma\left(Q^{\pi}(s',a') - \alpha\log\pi(a'|s')\right)\right] \tag{2}$$

  - Next states during training from the *Replay Buffer*
  - Next actions during training from the *current Policy*
- Expectation $\mathbb{E}[\cdot]$ approximated via samples:

$$Q^{\pi}(s,a) \approx r + \gamma\left(Q^{\pi}(s',\tilde{a}') - \alpha\log\pi(\tilde{a}'|s')\right),\ \tilde{a}' \sim \pi(\cdot|s'). \tag{3}$$

Value of an Action & Entropy Bonus II

- Loss function:

$$\mathcal{L}(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathbb{E}} \left[\left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \right] \tag{4}$$

- Target:

$$y(r,s',d) = r + \gamma(1-d) \left(\min_{j=1,2} Q_{\phi_{\text{target},j}}(s',\tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right)$$
$$\tilde{a}' \sim \pi_\theta(\cdot|s') \tag{5}$$

Lerning the Policy: Maximize $V^{\pi}(s)$ I

- Learn policy: maximize $V^{\pi}(s)$:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a\sim\pi}\left[Q^{\pi}(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right) \tag{6}$$

$$= \mathop{\mathbb{E}}_{a\sim\pi}\left[Q^{\pi}(s,a) - \alpha \log \pi(a|s)\right] \tag{7}$$

- **Reparameterization Trick:** Sample a value from $\pi_\theta(\cdot \mid s)$ by calculating a deterministic function of state, policy and independent noise:

$$\tilde{a}_\theta(s,\xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right),\ \xi \sim \mathcal{N}(0, I). \tag{8}$$

  (tanh acts as limiter)

- … thanks to this, we can eschew the policy parameters:

$$\mathop{\mathbb{E}}_{a\sim\pi_\theta}\left[Q^{\pi_\theta}(s,a) - \alpha \log \pi_\theta(a|s)\right] \tag{9}$$

$$= \mathop{\mathbb{E}}_{\xi\sim\mathcal{N}}\left[Q^{\pi_\theta}(s,\tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right] \tag{10}$$

Lerning the Policy: Maximize $V^{\pi}(s)$ II

– Substitute Q^{π_θ} with an approximator: via min:

$$\max_{\theta} \mathop{\mathbb{E}}_{s\sim\mathcal{D},\xi\sim\mathcal{N}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s) \right] \tag{11}$$

SAC Pseudocode I

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
    Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
    Execute a in the environment
    Observe next state s', reward r, and done signal d to indicate whether s' is terminal
    Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
    If s' is terminal, reset environment state.
    if it's time to update **then**
        for j in range(however many updates) **do**
            Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
            Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right)$$

$$\tilde{a}' \sim \pi_\theta(\cdot|s')$$

SAC Pseudocode II

13: Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2 \qquad \text{for } i = 1,2$$

14: Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} \Big(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta \left(\tilde{a}_\theta(s) | \, s \right) \Big),$$

where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.

15: Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1-\rho)\phi_i \qquad \text{for } i = 1,2$$

16: **until** convergence

SAC in a Gist

- SAC is off-policy
  - Key idea: Entropy regularization
  - Entropy bonus term: $H(\pi(\cdot|s))$
  - Entropy: Measure for randomness
  - **During training, SAC rewards the agents for more randomness in its actions**
  - Solves exploration vs. exploitation dilemma
  - Helps against premature convergence to bad local minimum
  - Trade-off between exploration and exploitation via factor α: Annealing
- However: Entropy regularization does not guarantee a robust strategy
- Remedy: Autocurriculum (Agent A vs. Agent B)

Adding the Autocurriculum Variant I

```yaml
definitions:
  # ...
  agents:
    # ...
    sac_defender:
      name: Gandarlf
      brain:
        name: harl:SACBrain
        params:
          fc_dims: [4, 4]
          update_after: 120
          update_every: 10
          batch_size: 64
      muscle:
        name: harl:SACMuscle
        params:
```

Adding the Autocurriculum Variant II

```
        start_steps: 120
    objective:
      name: midas.tools.palaestrai:ArlDefenderObjective
      params: {}
# ...
sensors:
  bus8:
    midas_powergrid:
      - midas_powergrid.Powergrid-0.0-bus-8.vm_pu
      - Powergrid-0.0-bus-8.in_service
  attacker:
    midas_powergrid:
      - midas_powergrid.Powergrid-0.0-bus-8.vm_pu
      - Powergrid-0.0-bus-8.in_service
      - midas_powergrid.Powergrid-0.0-bus-3.vm_pu
      - Powergrid-0.0-bus-3.in_service
actuators:
```

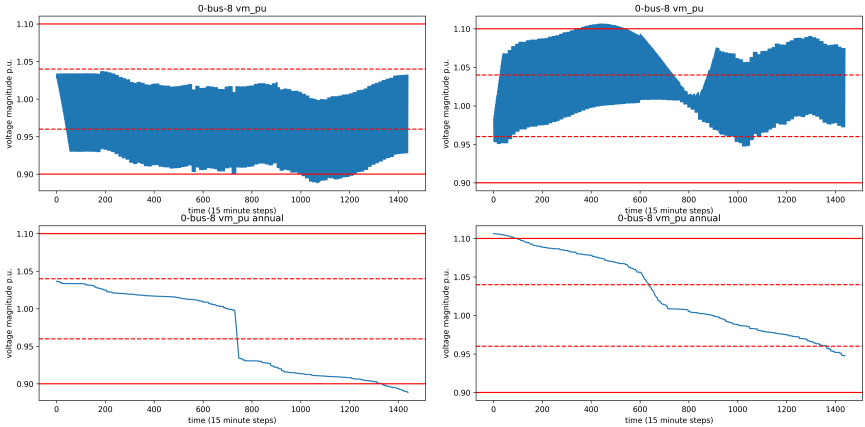
Adding the Autocurriculum Variant III

```
    bus8:
      midas_powergrid:
        - PysimmodsPV-0.Photovoltaic-3.q_set_mvar
    attacker:
      midas_powergrid:
        - PysimmodsPV-0.Photovoltaic-0.q_set_mvar
# ...
schedule:
  - reactive_power_control_training:
      environments: [[midasmv]]  # one factor, one level
      agents: [[q_controller, evil_attacker], [sac_defender, evil_attacker]] # o
      simulation: [ttsc] # one factor, one level
      phase_config: [training] # one factor, one level
      sensors:
        q_controller: [bus8]
        sac_defender: [bus8]
        evil_attacker: [attacker]
```

Adding the Autocurriculum Variant IV

```
      actuators:
        q_controller: [bus8]
        sac_defender: [bus8]
        evil_attacker: [attacker]
  - reactive_power_control_test:
      environments: [[midasmv]]  # one factor, one level
      agents: [[q_controller, evil_attacker], [sac_defender, evil_attacker]] # o
      simulation: [ttsc] # one factor, one level
      phase_config: [test] # one factor, one level
      sensors:
        q_controller: [bus8]
        sac_defender: [bus8]
        evil_attacker: [attacker]
      actuators:
        q_controller: [bus8]
        sac_defender: [bus8]
        evil_attacker: [attacker]
```

Gandarlf Saves the World



Conclusion

Conclusion

- MIDAS: Multi-Domain Test Scenario – CPES Co-Simulation
- palaestrAI: A Training Ground for Autonomous Agents
- Adversarial Resilience Learning Methodology: Autocurriculum DRL for resilient operation of critical infrastructures
- Here: from reference scenario to Q controller to attacker to autocurriculum in one experiment

`http://gitlab.com/midas-mosaik`

`http://palaestr.ai`

Eric MSP Veith `<eric.veith@uol.de>`