Carl von Ossietzky
Universität
Oldenburg

General, Learned, and Verified: What the Future of Learning Agents in Power Grids Could Be

Eric MSP Veith <eric.veith@uol.de> , 2025-03-10

Carl von Ossietzky Universität Oldenburg

% whoami

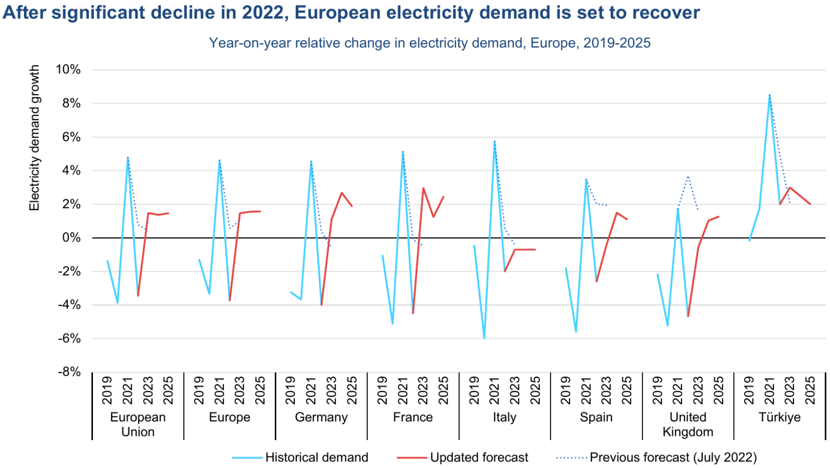
- Eric MSP Veith `<eric.veith@uol.de>`
- Currently head of a junior research group at University of Oldenburg, Germany
- Computer scientist by heart: First ICT, then distributed heuristics, then Multi-Agent Systems, now advanced Deep Reinforcement Learning
- PhD in 2017: “Universal Smart Grid Agent for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

Carl von Ossietzky Universität Oldenburg

% whereami

Carl von Ossietzky Universität Oldenburg

Electricity Demand Rising

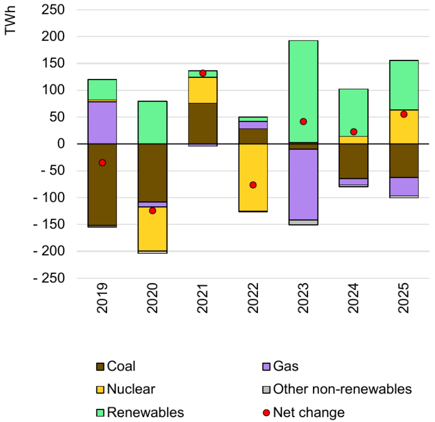


Carl von Ossietzky Universität Oldenburg

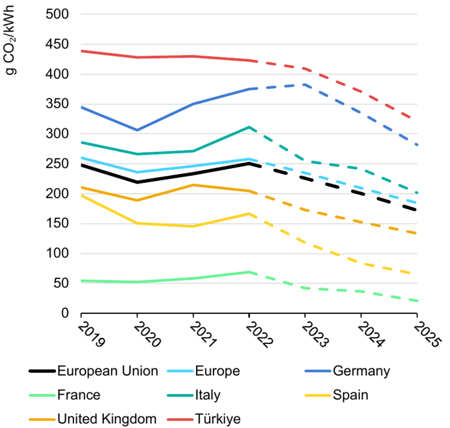
Renewables Are Replacing Fossil Fuels

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

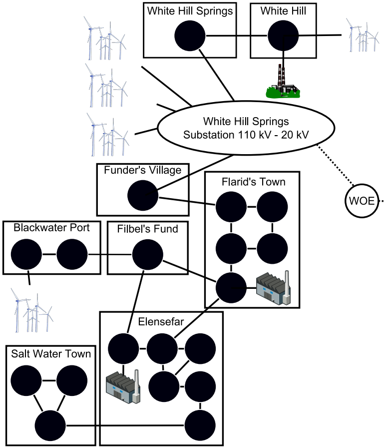


Development of average CO_2 intensity, Europe, 2019-2025



AI as Promise of an Alternative

- Multi-Agent Systems promise local, more more efficient grid operation
- Each node (subgrid, ...) an agent
- Nodes (agents) forecast local power generation/consumption
- On disequilibrium, match forecasts to achieve equilibrium
- An example, based on the literature [5, 3]

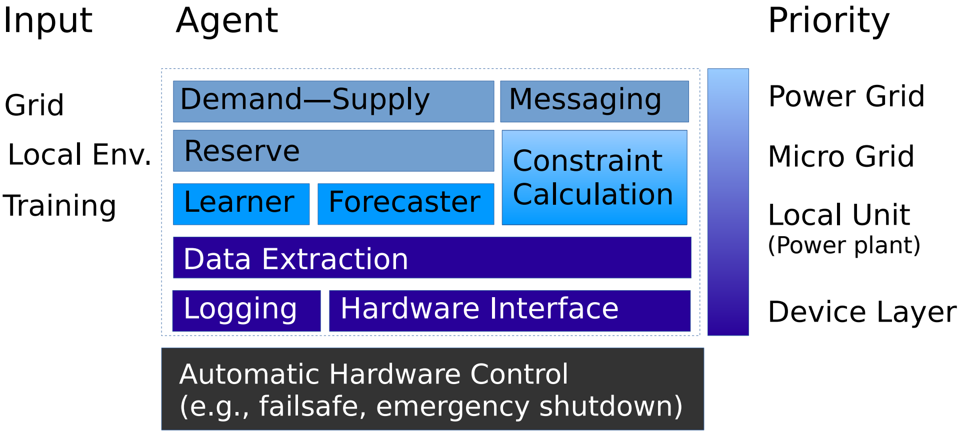


Pillars of CPES-MAS

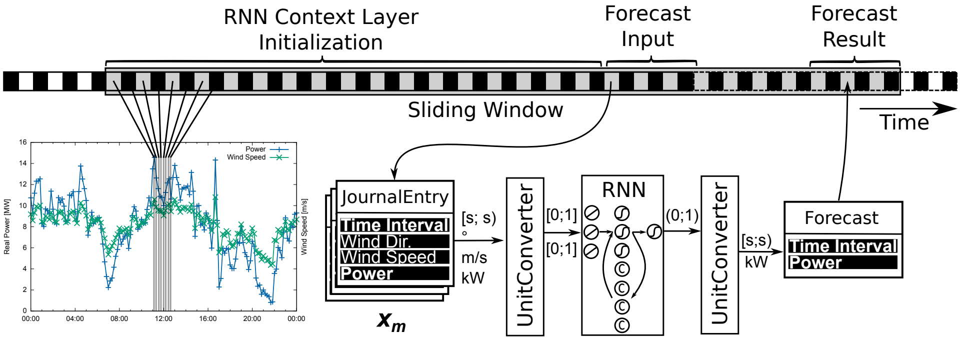
An approach for a Multi-Agent System (MAS) that manages high shares of volatile generators and consumers in an energy system is based on three pillars:

1. Forecasting of local generation or demand
2. Communicating demand and generation (distributed snapshotting with the power grid in mind)
3. Solving the combinatorical problem of demand and supply

Agent Design

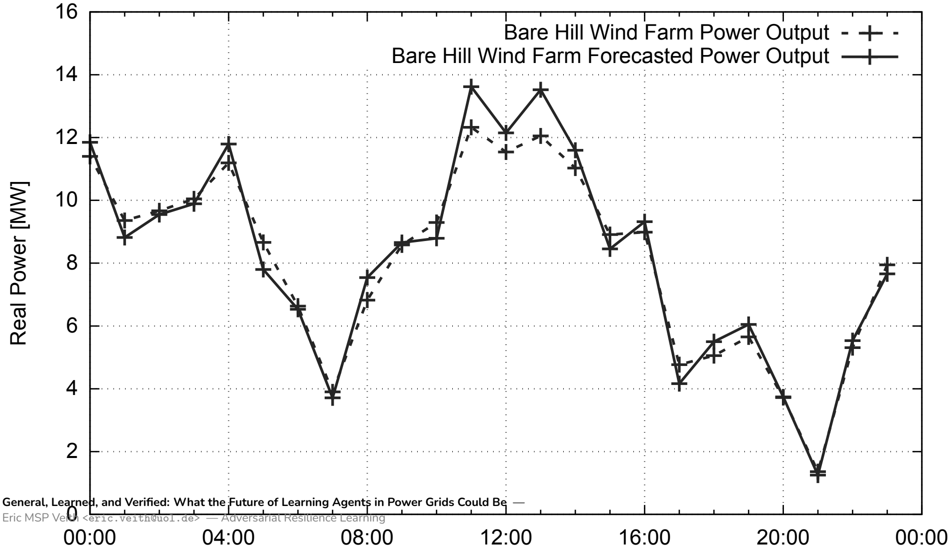


Forecasting



Forecasting is industry state of the art now.

Forecasting

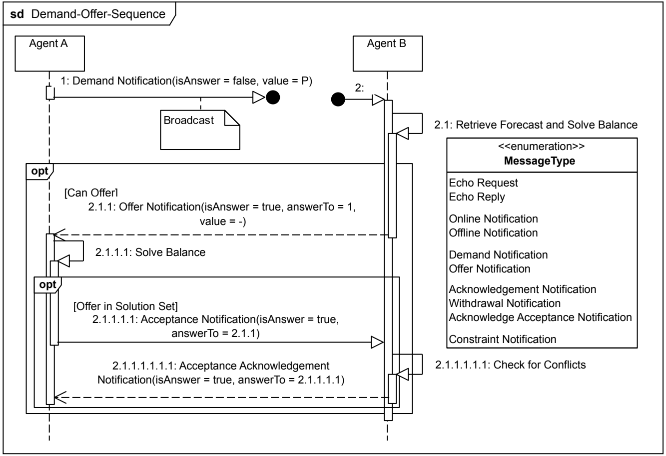


Communication

After forecasting works (i. e., imbalances are known in advance), each agent (= node) must ask its neighbors to help attain the equilibrium.

- Approach: Use overlay network – (virtual) communication lines between agents based on actual grid lines
- Requests (demand for power, or increase in feed-in that should be consumed) travel via selective broadcast
- Each node along the way must try to contribute!

Four-Way Handshake



LPEP Forwarding

- L_i: *Links* of the i-th agent
- $l_{i,k}$: k-th link of i-th agent
- distance($l_{i,k}$): Distance metric
- m_j: j-th message
- M_i: Message Journal of the i-th agent

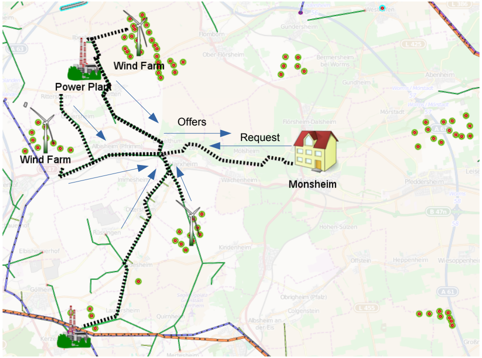
$$\begin{aligned} M_i = \{ & m_1 \mapsto \{(l_{i,1}, m_{1,\mathrm{distance}(l_{i,1})}), \ldots, (l_{i,n}, {m'}_{1,\mathrm{distance}(l_{i,n})})\}, \\ & \ldots, \\ & m_n \mapsto \{(l_{i,1}, m_{n,\mathrm{distance}(l_{i,1})}), \ldots, (l_{i,n}, {m'}_{n,\mathrm{distance}(l_{i,n})})\}\} \end{aligned}$$

$$l_{i,1}(t) \leq l_{i,2}(t) \quad \Leftrightarrow \quad l_{i,1,\mathrm{distance}}(t) \leq l_{i,2,\mathrm{distance}}(t)$$

Forwarding

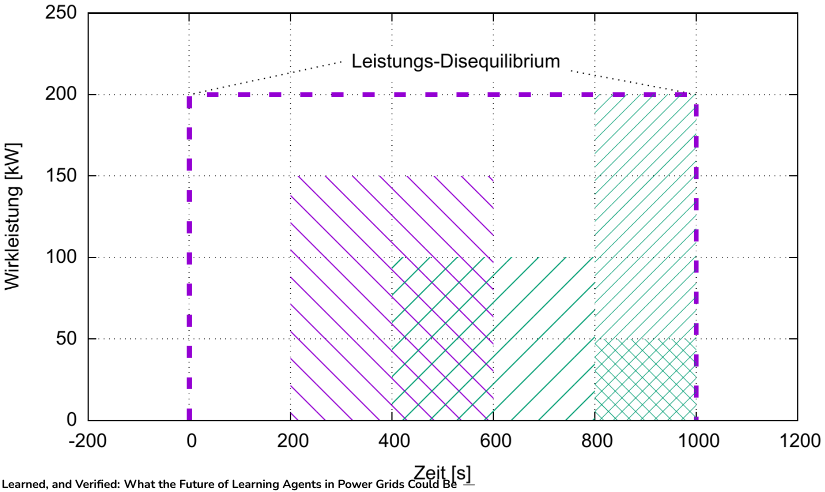
1. Respect *Constraint Notifications*:
   1.1 No answer if $\min\left(M(m)\right)$ a constraint notification to m, additionally
   1.2 send *Withdrawal Notification* iff already answered
2. $m_{isAnswer}$: forward on best connect ($\min\left(M(m_{answerTo})\right)$)
3. *Selective Broadcast* for requests:
   3.1 Replace request with *Constraint Notification*, if necessary
   3.2 $M(m) = \emptyset$: forward on $|L| - 1$ links
   3.3 $m' = \min\left(M(m')\right)$: Update by fowarding
   3.4 Otherwise: no forwarding

How to Decide…?



1. Local forecasting shows demand or oversupply of energy
2. Requests are sent
3. Other nodes make offers
4. Offers reach requestor
5. **Decision about offers?**

Power Balance Concept



Set of Mappings $[t_1; t_2) \mapsto P$.

Problem Statement

'Power Balance Algebra':

$$\{[t_1; t_3) \mapsto P_1\} \cup \{[t_2; t_4) \mapsto P_2\} = \{[t_1; t_2) \mapsto P_1, [t_2; t_3) \mapsto P_1 + P_2, [t_3; t_4) \mapsto P_2\}\,, \tag{1}$$

$$[t_1; t_2) \mapsto P_1 \ \subseteq\ [t_3; t_4) \mapsto P_2 \quad \Leftrightarrow \quad t_1 \geq t_3 \ \wedge\ t_2 \leq t_4 \ \wedge\ P_1 \leq P_2\,; \tag{2}$$

Distance Function:

$$\mathrm{d}(r_i) : r_i \mapsto \mathbb{R} \tag{3}$$

Problem Statement:

$$\sum_i b_i r_i \subseteq r_0\,,\ \ i \neq 0, b_i \in \{0, 1\}\,, \tag{4}$$

$$\text{Subject to: } \min \sum_i b_i \mathrm{d}(r_i),\ \ i \neq 0, b_i \in \{0, 1\}\,. \tag{5}$$

Atomization

$$\boldsymbol{P} = (|P_0|, |P_1|, \ldots, |P_i|, |P_C|) ,$$

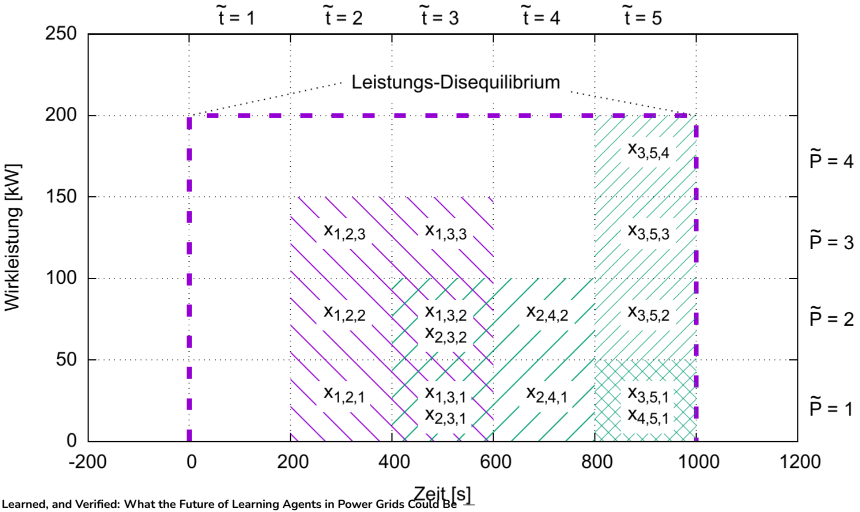
$$\boldsymbol{t} = (t_{2,0} - t_{1,0}, t_{2,1} - t_{1,1}, \ldots, t_{2,i} - t_{1,i}) ,$$

$$\Delta P = \mathrm{ggT}(\boldsymbol{P}) ,$$

$$\Delta t = \mathrm{ggT}(\boldsymbol{t}) ,$$

$$x_{i,\tilde{t},\tilde{P}} = \begin{cases} 1 & \text{if agent } i \text{ influences the grid in time-subinterval } \tilde{t} \text{ with} \\ & \text{power from the power-subinterval } \tilde{P}, \\ 0 & \text{else.} \end{cases}$$

Atomization Illustrated



Carl von Ossietzky Universität Oldenburg

Model of the Disequilibrium

A symmetric function for each time-subinterval:

$$\mathrm{S}_k^n(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}}=\boldsymbol{k},\tilde{\boldsymbol{P}}}) = \begin{cases} 1 & \text{if } n \text{ variables in } \boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}}=\boldsymbol{k},\tilde{\boldsymbol{P}}} \text{ equal } 1, \\ 0 & \text{else;} \end{cases}$$

Full Disequilibrium:

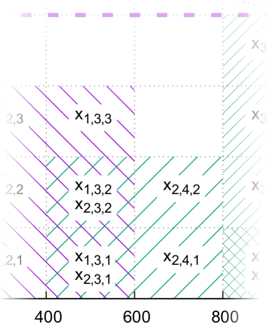
$$S = \bigcap_{k=1}^{m} \mathrm{S}_k^n(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}}=\boldsymbol{k},\tilde{\boldsymbol{P}}})$$

Modelling Responses

Acceptance Function:

$$r_i(\boldsymbol{x}_{i,\tilde{t},\tilde{P}}) = \begin{cases} 1 & \text{if } \boldsymbol{x}_{i,\tilde{t},\tilde{P}} \text{ describes a valid interval for accepting the re-} \\ & \text{sponse of } i, \\ 0 & \text{else.} \end{cases}$$

$$\begin{aligned} r_2(\boldsymbol{x}_{i,\tilde{t},\tilde{P}}) = &\ \bar{x}_{2,3,1} \land \bar{x}_{2,3,2} \land \bar{x}_{2,4,1} \land \bar{x}_{2,4,2} \\ &\lor x_{2,3,1} \land x_{2,3,2} \land \bar{x}_{2,4,1} \land \bar{x}_{2,4,2} \\ &\lor x_{2,3,1} \land x_{2,3,2} \land x_{2,4,1} \land x_{2,4,2} \end{aligned}$$



Equilibrium

$$S = \bigcap_{k=1}^{m} \mathrm{S}_k^n(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}}=\boldsymbol{k},\tilde{\boldsymbol{P}}})$$

$$R = \bigcap_{i \in I',\tilde{t},\tilde{P}} r_i(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}},\tilde{\boldsymbol{P}}}) \,,$$

$$C = S \cap R \,.$$

Equilibrium

$$
\begin{aligned}
S &= \bigcap_{k=1}^{m} \mathrm{S}_k^n(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}}=\boldsymbol{k},\tilde{\boldsymbol{P}}}) \\
R &= \bigcap_{i \in I', \tilde{t}, \tilde{P}} r_i(\boldsymbol{x}_{\boldsymbol{i},\tilde{\boldsymbol{t}},\tilde{\boldsymbol{P}}}) \ , \\
C &= S \cap R \ .
\end{aligned}
$$

- Best solution through ordering: $r_i \leq r_{i'} \quad \Leftrightarrow \quad \mathrm{d}(r_i) \leq \mathrm{d}(r_{i'})$
- Generating next vector in S through permutation
- Exploiting the commutative property of the intersection operator: $R_n \cap (\ldots \cap (R_2 \cap (R_1 \cap S)))$

Efficiency

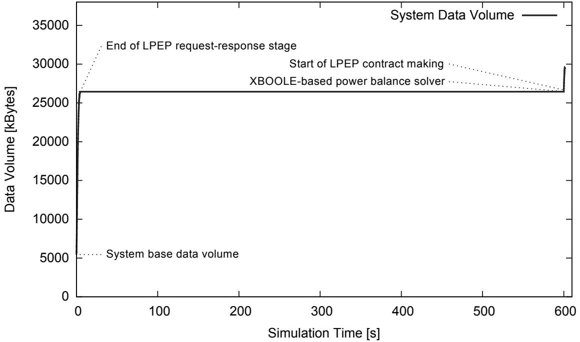
Data Effect	$\kappa = \frac{W}{D} \left[\frac{\mathrm{kWh}}{\mathrm{kB}}\right]$
Data Efficiency	$\xi = \frac{\Delta P}{D} \left[\frac{\mathrm{kW}}{\mathrm{kB}}\right]$

Comparison

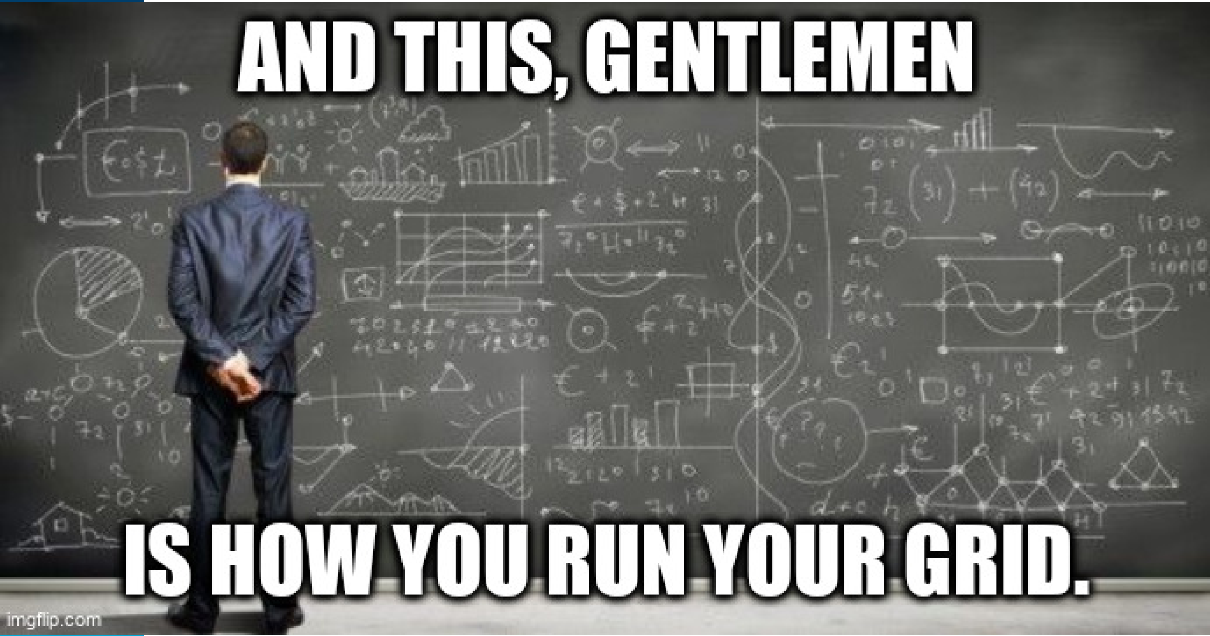
Comparison with BDD approach by Inoue *et al.* (2014):

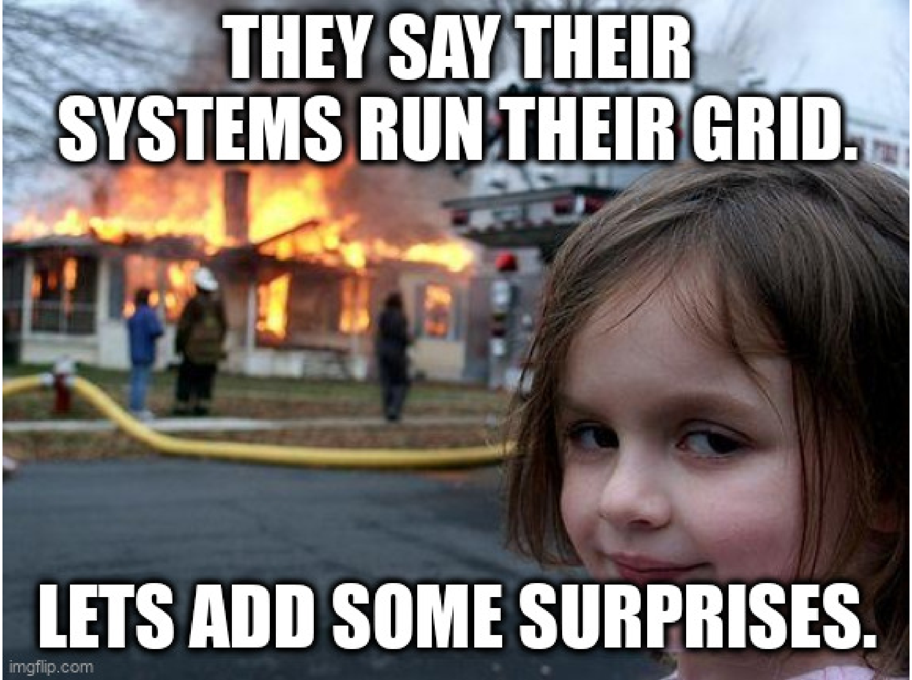
	BDD	Universal Agent
Loss Avoided (ΔP)	17 208 kW	17 208 kW
Runtime	> 16 min	< 11 min (simulated)
D	100 MB	28.9 MB
ξ	0.168 kW/kB	0.581 kW/kB

Universal Agent Efficiency



- BDD approach in low-load situation: 100 kB
- *Universal Agent* concept especially useful in complex load situations


AND THIS, GENTLEMEN
IS HOW YOU RUN YOUR GRID.
imgflip.com


THEY SAY THEIR SYSTEMS RUN THEIR GRID.
LETS ADD SOME SURPRISES.
imgflip.com

Carl von Ossietzky Universität Oldenburg

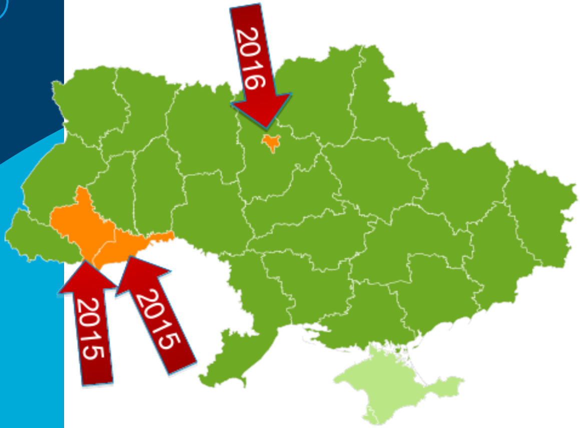
Knowns vs. Unknowns

- The well-defined way of traditional MAS can guarantee a (theoretical) optimal solution
- They are robust: Cases known at design time can be handled
- Unknown unknowns and even some known problems can not be handled.

… We need a system that can act universally.

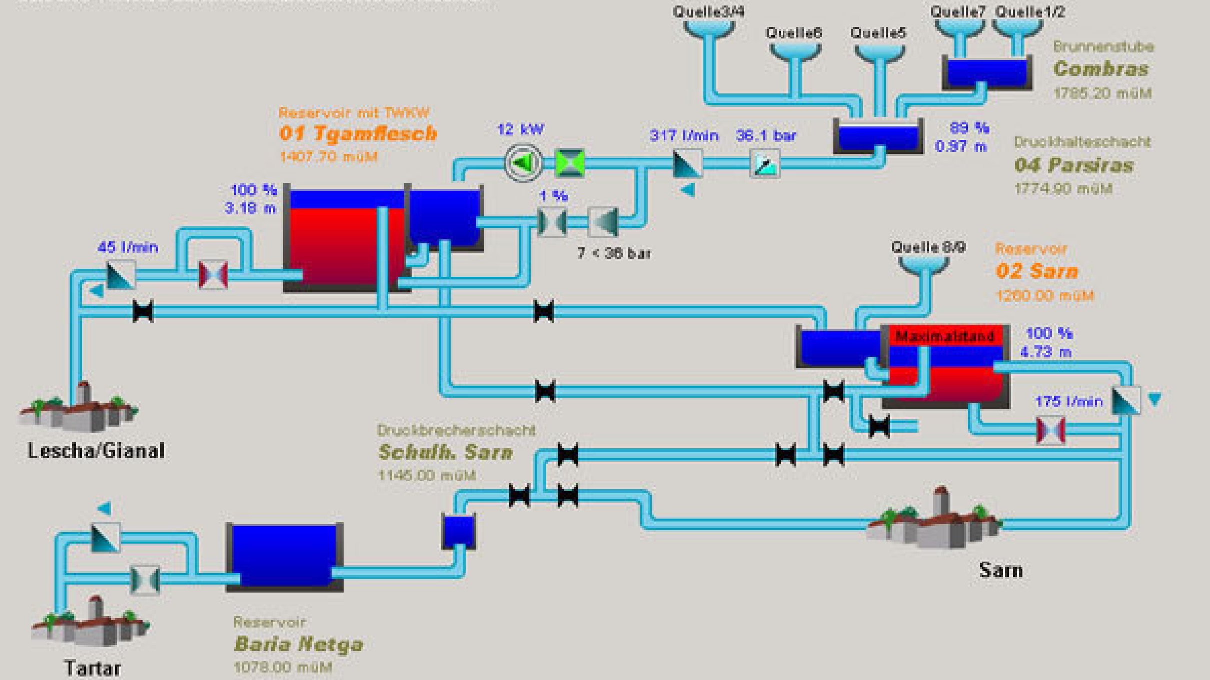

“There are only two types of companies:
those who have been hacked,
and those who don’t yet know
they have been hacked.”
— John T. Chambers

Energy Systems Fit The Bill Just As Well

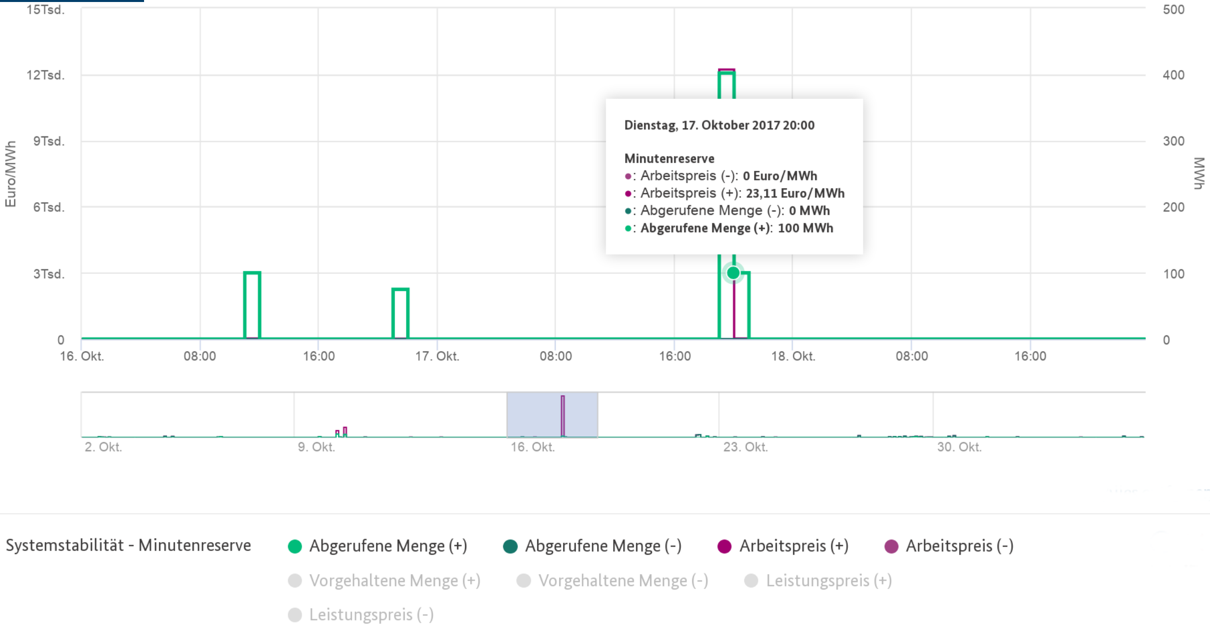


Dec 23rd, 2005

- **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

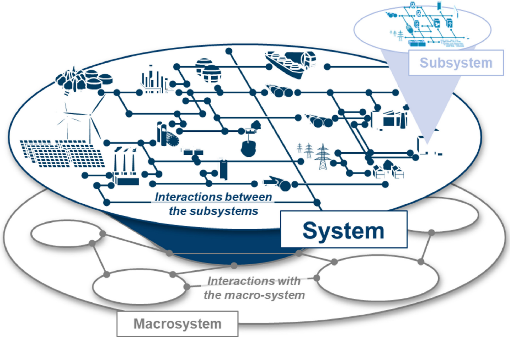


Quelle3/4
Quelle6
Quelle5
Quelle7
Quelle1/2
Brunnenstube
Combras
1785.20 müM
Reservoir mit TWKW
01 Tgamflesch
1407.70 müM
12 kW
317 l/min
36.1 bar
89 %
0.97 m
Druckhalteschacht
04 Parsiras
1774.90 müM
100 %
3.18 m
1 %
45 l/min
7 < 36 bar
Quelle 8/9
Reservoir
02 Sarn
1260.00 müM
Maximalstand
100 %
4.73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


15Tsd.
12Tsd.
9Tsd.
6Tsd.
3Tsd.
0
Euro/MWh
500
400
300
200
100
0
MWh
16. Okt.
08:00
16:00
17. Okt.
08:00
16:00
18. Okt.
08:00
16:00
Dienstag, 17. Oktober 2017 20:00
Minutenreserve
●: Arbeitspreis (-): 0 Euro/MWh
●: Arbeitspreis (+): 23,11 Euro/MWh
●: Abgerufene Menge (-): 0 MWh
●: Abgerufene Menge (+): 100 MWh
2. Okt.
9. Okt.
16. Okt.
23. Okt.
30. Okt.
Systemstabilität - Minutenreserve
Abgerufene Menge (+)
Abgerufene Menge (-)
Arbeitspreis (+)
Arbeitspreis (-)
Vorgehaltene Menge (+)
Vorgehaltene Menge (-)
Leistungspreis (+)
Leistungspreis (-)

Learning Resilient Control

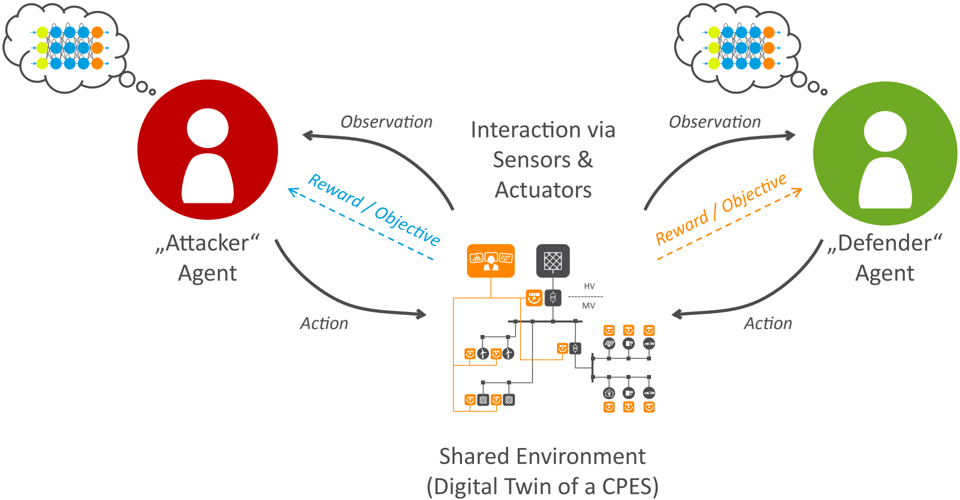
- **Interconnected CPS have always attack surface due to their inherent complexity**
- Low latency of ICT and OT
- High interdependence
- Complexity in breadth and depth
- Cricital Services as SPOF (DNS, BGP, SCADA, SDL)
- **Learning Stratgies for automatic issue mangement**
- “Adversarial Resilience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



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

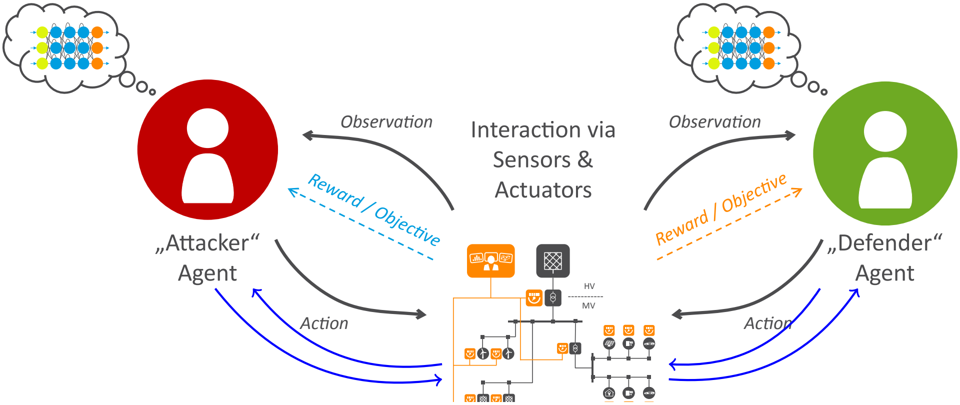
Carl von Ossietzky Universität Oldenburg

Adversarial Resilience Learning



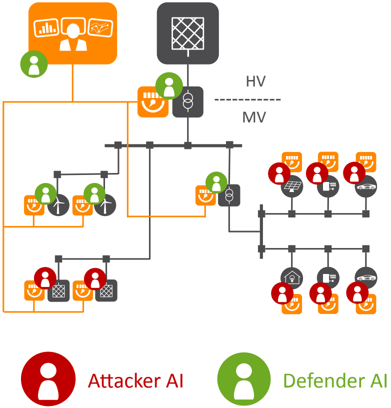
Carl von Ossietzky Universität Oldenburg

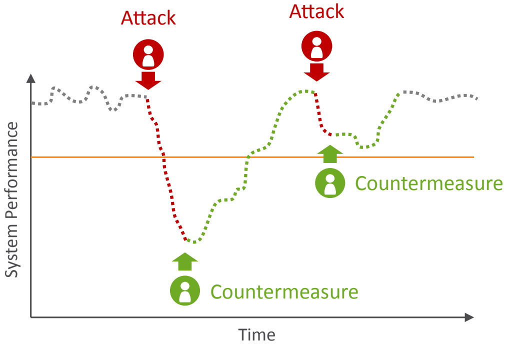
Adversarial Resilience Learning



Adversarial System-of-Systems Reinforcement Learning

ARL Agent Interaction






Defender Points
2656
Loads Connected
40
Generators Connected
24
Buses Connected
82
Transformers Connected...
10
Most Valuable Actions (Defender)
Info
Time
Points
Changed Scaling from Lehe Households - 4 to 0.6667
2018-01-01 02:00:10
-0.25
Changed Scaling from Leherheide Industrielast to 0.8889
2018-01-01 02:26:10
-0.07
Changed Tap_pos from trafo to 1.0000
2018-01-01 01:47:50
-0.07
Changed Scaling from PV Fischereihafen to 0.0000
2018-01-01 02:14:30
-0.07
Changed Tap_pos from trafo to 1.0000
2018-01-01 01:45:30
-0.06
Map
Wehden
SPECKENBÜTTEL
Elmlohe
Spaden
Bramel
Schiffdorf
BLEXEN
EINSWARDEN
GRÜNHÖFE
Bexhövede
0
0
1
5
1
36
9
7
1
L
L
G
G
T
S
Load
Generator
Transformer
Switch
Schhiffdorferdamm (1)
Manipulations over last 10 minutes
Attacker Actions
Loads: 1
Generators: 0
Defender Actions
Loads: 0
Generators: 0
Transformers: 0
Switches: 0
Leaflet | © OpenStreetMap
Time Left (Coins Left in %)
23%
Attacker Points
7344
Constraint Violations
ConstraintGeneratorVoltageChange
1026
ConstraintLoadVoltageChange
6017
ConstraintReactivePower
371
Malfunctions
20
15
10
5
0
01:00
01:30
02:00
02:30
03:00
Transformers
Loads
Generators
Most Valuable Actions (Attacker)
Info
Time
Points
Changed Scaling from Geestemünde Households - 0 to 0.5000
2018-01-01 01:00:00
0.96
Changed Scaling from Geestemünde Households - 0 to 0.5000
2018-01-01 01:00:00
0.93
Changed Scaling from Geestemünde Households - 0 to 0.5000
2018-01-01 01:00:00
0.91

Carl von Ossietzky Universität Oldenburg

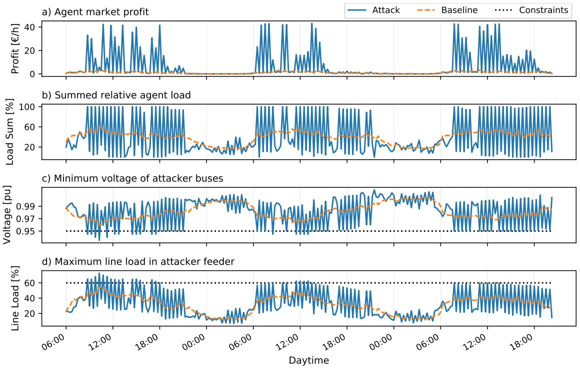
ARL Agent Can Discover Attacks

- Attack on voltage level
- Attacker controls Q feed-in
- Known attack: Oscillating behavior
- ARL agent indepently disovers attack, but also finds variant

Veith, Wellßow, and Uslar [4]

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



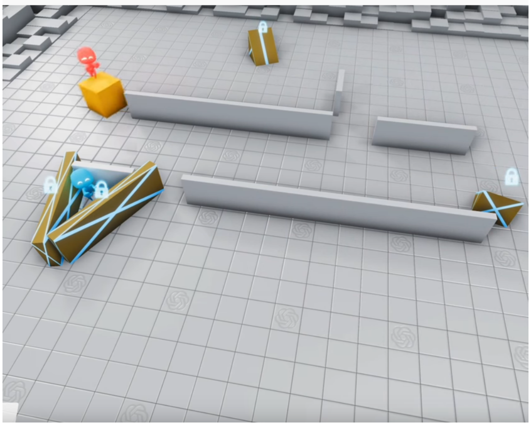
Agents learn to “game” local energy markets

Wolgast, Veith, and Nieße [6]

Carl von Ossietzky
Universität
Oldenburg

Multi-Agent Autocurricula

- ARL is an autocurriculum setup
- Independently known & verified to work
- Example Setup: Two groups of agents play hide and seek
- No domain information; agents learn strategies and tool use independently
- Result: Agents learn to exploit bugs in the underlying game engine
  - Holes in walls
  - Sliding boxes
  - Edge/corner jumps



Autocurricula Helpful in Theory

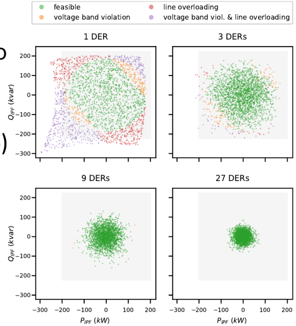
- DRL agents collect initial samples from random actions
- However, random actions over correlated actuators lead to convolution problem, i. e., if $X, \; Y \sim \mathcal{U}$, then

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x) \, dx \, , \tag{6}$$

  which is a triangle distribution
- Equally, consider SAC's entropy maximization,

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha \underbrace{H(\pi(\cdot | s_t))}_{\text{Entropy term}} \right) \right] \tag{7}$$

- … obviously, a "push" is required



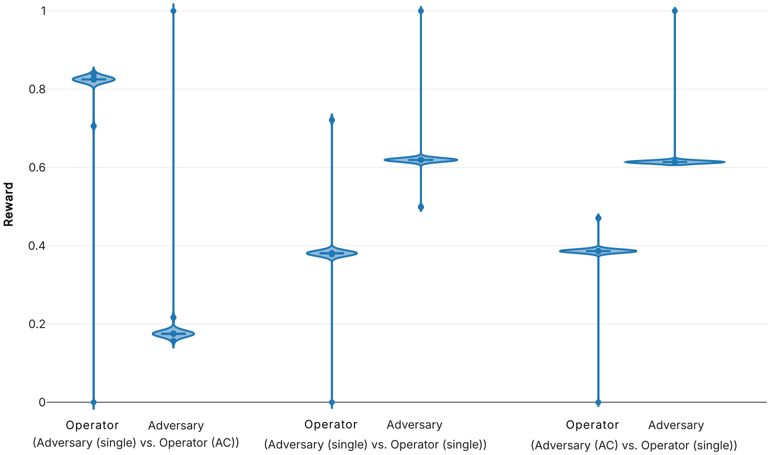
Autocurricula Helpful in Theory II

1. Formally, DRL approximates the unknown environment distribution p with q, i. e.,

$$\begin{array}{ll} \text{minimize} & KL(p, q) \\ \text{subject to} & q \end{array} \tag{8}$$

2. Learn a policy to exploit q, π_Ω
3. (Single agent: get stuck in local optimum because p is mostly unknown because of missing sample data)
4. Adversary agent: Observe p as influenced by π_Ω
5. $R_A(\boldsymbol{s}_t \sim p) = -R_\Omega(\boldsymbol{s}_t \sim p)$, therefore $\pi_A \widehat{=} -\pi_\Omega$
6. Result: agents observe adversarial sampels from the “other end” of p’s spectrum
7. Agents try to counter adverse effects: efficent state/action space exploration

… and in Practice



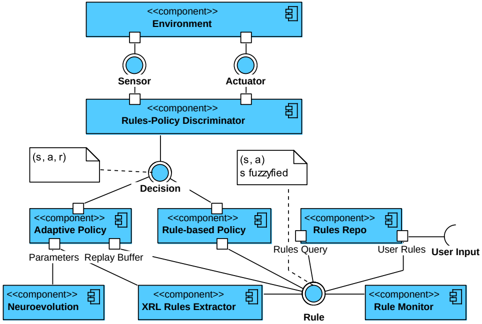
ARL Works

To summarize. . .

- ARL works for finding attack vectors (“easy”)
- ARL defender learn resilient control (“not quite so easy, but still. . . ”)
- ARL agents learn faster & more robust strategies through the autocurriculum setup (“proove me, I’m only circumstantial evidence!”)
- ARL defender agents can control modern power grids (“ha-ha, as if that would be acceptable. . . ”)
- **There is still a lot missing:**
  - Behavior guarantees
  - Adhere to constraints (rulesets)
  - Learn from existing domain knowledge
  - Adapt during production use (not just retraining)
  - . . .

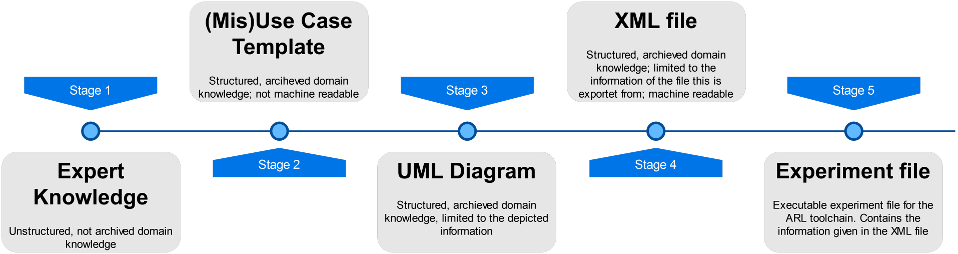
ARL Agent Architecture

- Learn from sensor inputs (policy: DRL)
- Deploy & forget, don't design policy networks: Neuroevolution
- Explainability
- Learn from domain knowledge
- Follow rules, if given



Learning from Domain Knowledge

Example: Misuse Cases



Trajectories from (Mis-) Use Cases

- Annotate UML diagrams to allow sampling; construct:
  - Experiment file
  - State machine from transitions

$$\begin{aligned}
M_{tg} &= (Q, \Sigma, \delta, q0, F) \\
&\text{with} \\
&(q, (\{c_q\} \in \mathit{ActuatorSetpoints}, \\
&\{i_q\} \in \mathit{TimeStepIntervals})) \in Q \\
&(i \in Q, n \in Q, \{sc\} \in \mathit{StepConstraints}) \in \delta
\end{aligned} \tag{9}$$

Relevant properties:

- Non-determinism
- State/actuator constraints c_q (think Gymnasium spaces)
- Time step intervals (sync to simulation semantics)
- Constrained steps (e. g., grid codes)

Carl von Ossietzky Universität Oldenburg

Combined AWAC and State Machine Sampling

Initialize Simulation S

State Machine $M_{tg} = (Q, \Sigma, \delta, s_0, F)$

$maximum\_steps \leftarrow x$

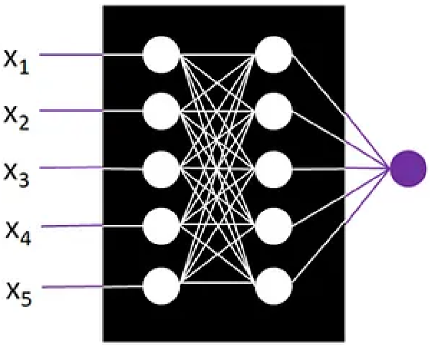
for $j \leq x$ **do**

- $s \leftarrow S.state$
- $a \leftarrow \{c\} \in (M_{tg}.state, c_{M_{tg}.\mathbf{state}}) \in Q_{M_{tg}}$
- $r \leftarrow R(a)$
- $s' \leftarrow S.step(a)$
- $db \leftarrow db \cup \{(s, a, s', r)\}$
- $advance(M_{tg})$

Combined AWAC and State Machine Sampling II

Dataset $D = \{(\text{s, a, s', } r)_j\} \sim db$
Initialize buffer $\beta = D$
Initialize π_θ, Q_ϕ
for iteration $i = 1, 2, \ldots, n$ **do**
  Sample batch $(s, a, s', r) \sim \beta$
  $y = R(s, a) + \gamma \mathbb{E}_{s', a'}[Q_{\phi_{k-1}}(s', a')]$
  $\phi \leftarrow \arg\min_\phi \mathbb{E}_D[Q_\phi(s, a) - y^2]$
  $\theta \leftarrow \arg\max_\theta \mathbb{E}_{s, a \sim \beta}[\log \pi_\theta(a|s) \exp(\frac{1}{\lambda} A^{\pi_k}(s, a))]$
  if $i > \mathit{num\_offline\_steps}$ **then**
    $\tau_1, \ldots, \tau_K \sim p_{\pi_\theta}(\tau)$
    $\beta \leftarrow \beta \bigcup \{\tau_1, \ldots, \tau_K\}$

"The Black Box"

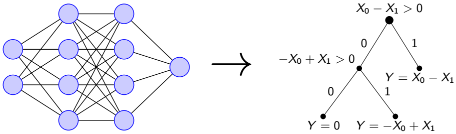


Explanation goals

- Motivation: No trust without explanation of learned strategies of agents
- Idea: Use Decision Trees (DTs) with extraction of rulesets for explanation
  - DTs are transparent and somewhat interpretable
  - They can be trained directly (no need for black-box Deep Neural Network (DNN) models)
  - But DNNs are better regularized, which increases trainability [2]
- Conflicting goals:
  - Construction of powerful (Deep Reinforcment Learning (RL) (RL)) learning system
  - (Post-hoc) Explainability with comprehensible model (e. g. DTs)

Learned Policy Explanation

- Equivalent transformation of efficient-learnable Feed-Forward DNNs (DNNs) into compressed DTs



- NN2EQCDT algorithm heavily relies on equivalence description of DNNs and DTs [1], but still addressed research gaps to better use it for explainability:
  - Transformation algorithm and actual implementation proposed for PyTorch models
  - Exponential growth is addressed by lossless pruning
  - Dynamic compression reduces computation time significantly and may reduce inference time
  - Option to directly include global constraints for further pruning

NN2EQCDT algorithm

$\hat{\boldsymbol{W}} = \boldsymbol{W}_0$
$\hat{\boldsymbol{B}} = \boldsymbol{B}_0^\top$
$rules = \text{calc\_rule\_terms}(\hat{\boldsymbol{W}}, \hat{\boldsymbol{B}})$
$T, new\_SAT\_leaves = \text{create\_initial\_subtree}(rules)$
$\text{set\_hat\_on\_SAT\_nodes}(T, new\_SAT\_leaves, \hat{\boldsymbol{W}}, \hat{\boldsymbol{B}})$
for $i = 1, \ldots, n-1$ **do**
  $SAT\_paths = \text{get\_SAT\_paths}(T)$
  for SAT_path in SAT_paths **do**
    $\boldsymbol{a} = \text{compute\_a\_along}(\text{SAT\_path})$
    $SAT\_leave = \text{SAT\_path}[-1]$
    $\hat{\boldsymbol{W}}, \hat{\boldsymbol{B}} = \text{get\_last\_hat\_of\_leave}(T, SAT\_leave)$
    $\hat{\boldsymbol{W}} = (\boldsymbol{W}_i \odot [(\boldsymbol{a}^\top)_{\times k}])\hat{\boldsymbol{W}}$
    $\hat{\boldsymbol{B}} = (\boldsymbol{W}_i \odot [(\boldsymbol{a}^\top)_{\times k}])\hat{\boldsymbol{B}} + \boldsymbol{B}_i^\top$
    $rules = \text{calc\_rule\_terms}(\hat{\boldsymbol{W}}, \hat{\boldsymbol{B}})$
    $new\_SAT\_leaves =$ $\text{add\_subtree}(T, SAT\_leave, rules, invariants)$
    $\text{set\_hat\_on\_SAT\_nodes}(T, new\_SAT\_leaves,$ $\hat{\boldsymbol{W}}, \hat{\boldsymbol{B}})$

Finally:

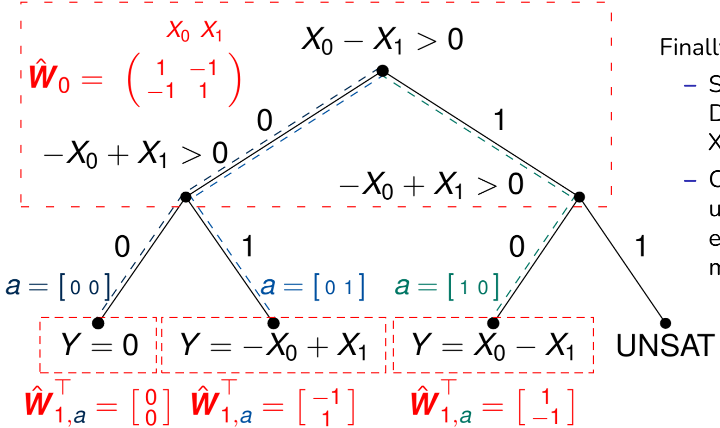
- Converting final rules to expressions
- Pruning the (temporary) UNSAT nodes

Effective weight matrix calculation

$\hat{\boldsymbol{W}} = \boldsymbol{W}_0$
$\hat{\boldsymbol{B}} = \boldsymbol{B}_0^\top$
for $i = 0, \ldots, n-2$ **do**
    $\boldsymbol{a} = [\ ]$
    for $j = 0, \ldots, m_i - 1$ **do**
        if $(\hat{\boldsymbol{W}}_j \boldsymbol{x}_0^\top + \boldsymbol{B}_j^\top)^\top > 0$ **then**
            $\boldsymbol{a}.\mathrm{append}(1)$
        else
            $\boldsymbol{a}.\mathrm{append}(0)$
    $\boldsymbol{W}_{i+1} \in \mathbb{R}^{m_i \times k}, \boldsymbol{a} \in \mathbb{Z}_2^{m_i}$
    $\hat{\boldsymbol{W}} = (\boldsymbol{W}_{i+1} \odot [(\boldsymbol{a}^\top)_{\times k}])\hat{\boldsymbol{W}}$
    $\hat{\boldsymbol{B}} = (\boldsymbol{W}_{i+1} \odot [(\boldsymbol{a}^\top)_{\times k}])\hat{\boldsymbol{B}} + \boldsymbol{B}_{i+1}^\top$
return $(\hat{\boldsymbol{W}} \boldsymbol{x}_0^\top + \hat{\boldsymbol{B}})^\top$

- Using right-handed linear transformation with bias
- Tailored to ReLU(-like) activation functions (e.g. ReLU, PReLU, LeakyReLU)

XOR model: DT Construction



Finally:

- Simple example of an DT representing an XOR function
- Construction of DT using calculated effective weight matrices

XOR model: DT Pruning

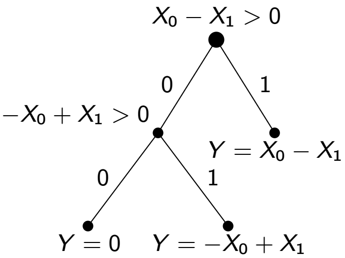


Figure: Simple pruning example

Pruning UNSAT node by

- remove parent and
- connecting sibling subgraph to parent of parent

Carl von Ossietzky Universität Oldenburg

Comparison of construction methods

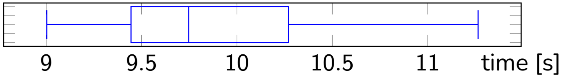


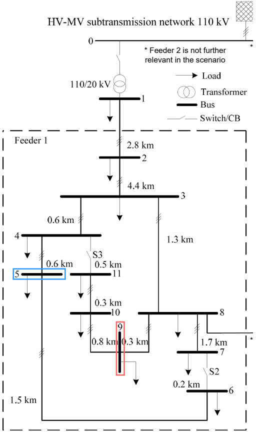
Figure: Boxplot ($n = 30$) for the computation time of the NN2EQCDT algorithm for the simple model

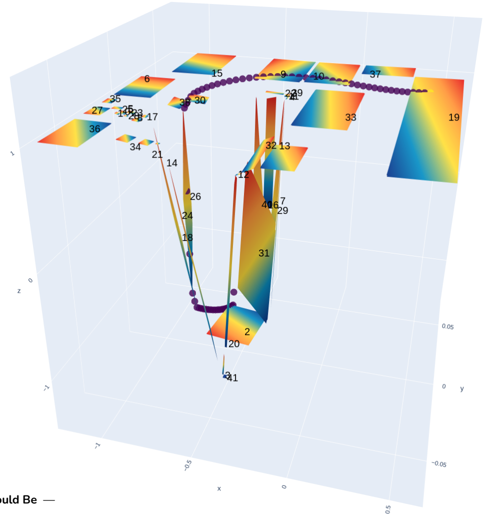
Table: Comparison of results or calculations for the construction of a DT from the simple model without and with compression of the NN2EQCDT algorithm

Pruning	#nodes	Computation time
☐	262143	> 1.5h
☑	83	9.75s

– Pruning ratio (amount of nodes) of 99.97%

Applications in Practice



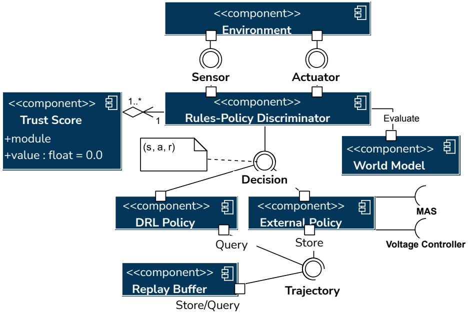


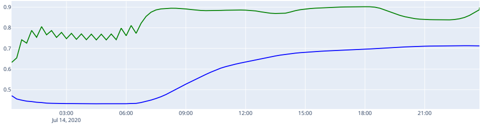
Co-Existence of MAS and DRL

- Hybrid systems out of focus, mostly either DRL, or MARL, or MAS.
- However, any agent isn't alone in its environment!
  - Game-theoretical models focus on a form of interaction (cooperation, competition, conflict, . . .), but not on co-existence
  - Underrated in literature: controller conflicts
- Many possible hybrid architectures, e. g.,
  - Hierarchies
  - Imitation Learning
  - Safeguarding (research gap!)

“Cover me:” A Practical Example for Safeguarding MAS

1. Observe MAS, **imitation learn** nominal behavior
2. For every *t*, internally propose actions
3. Check: MAS action proposal, ARL agent proposal against world model, note projected future states & rewards
4. Update trust by averaging reward over an LTI function
5. Apply actions from proposal with highest trust value
6. Observe state, learn from all three transitions





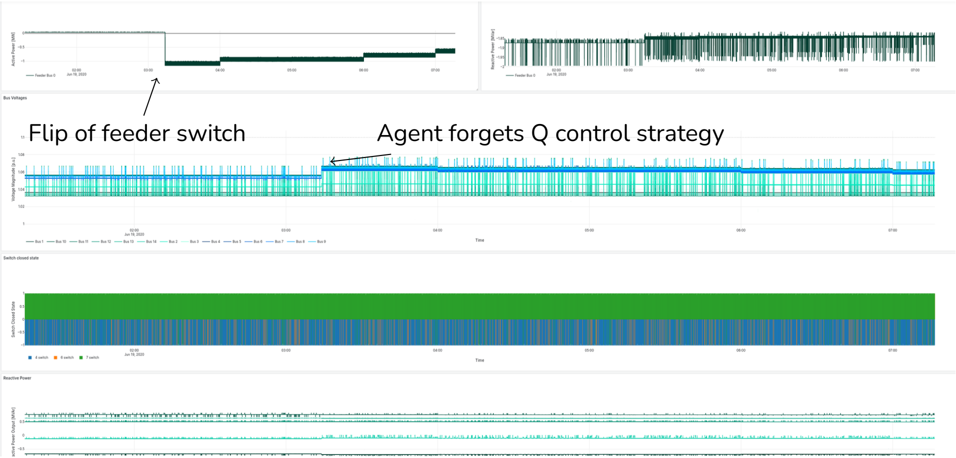
Deep Reinforcement Learning is not the Only Answer

The state of the art has many nice features:

- Offline learning (learning from domain knowledge)
- Imitation learning (learn existing control strategies by example)
- Model-based and model-free DRL
- eXplainable Reinforcement Learning to explain each action with low computational overhead

... however, this agent is still far from being safe.

“Good” Agents Fail to Apply Learned Strategies



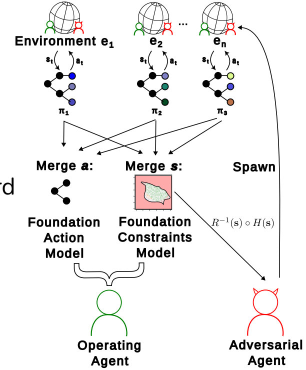
Catastrophic Forgetting on Topology Changes

A simple topology change screws the agent completely. Countermeasures:

1. Train the agent on as many scenarios as possible.
2. Verify the DRL agent.
3. Create a *Foundation Model* for actions
4. … Combine all of the above!

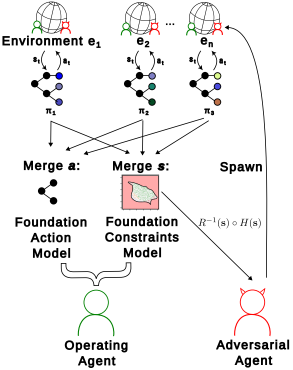
Training Strategy

- Take the autocurriculum approach to spawning environments
- Two adversaries: One “spawner” and n “workers”
- Operator agent trains on all of them (traditional multi-worker)
- Adversary spawns environments based on inverted reward and entropy $R^{-1}(\boldsymbol{s}) \circ H(\boldsymbol{s})$

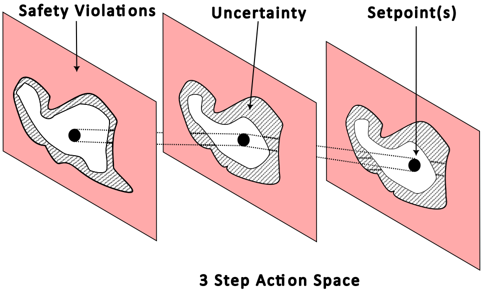


Latent Spaces

- Core idea: Use Graph Neural Networks to learn representations of underlying grids
- Graph space is our feature space: $\mathcal{X} = G(\boldsymbol{A}, \boldsymbol{K})$
- Train encoder for latent space representation of all G_i, where i is an environment instance we encountered: $\gamma : \mathcal{X} \mapsto \mathcal{L}$
- Use transformer to work directly on latent space
- Result: A foundation model for actions



And Verification. . . ?



- Alongside the Foundation Action Model, train a foundation model for contraints: *Foundation Constraints Model*
- Use the Foundation Constraints Model for N-step verification of trajectories to provide safety guarantees

A Lookout

- The journey towards highly automated grid operation & extension has just begun.
- AI can help testing future grids, be part of certification processes
- AI itself needs safeguards: Rulesets, explainability, and eventually certification, too. (Insurance…?)
- We will see sophisticated agent architectures in the near future.
- If you want to see interesting code, head over to `http://palaestr.ai` or shout out to `eric.veith@uol.de`!

Carl von Ossietzky Universität Oldenburg

Bibliography I

[1] Çaglar Aytekin. “Neural Networks are Decision Trees”. In: *CoRR* abs/2210.05189 (2022). [retrieved: 05, 2023], pp. 1–8. arXiv: 2210.05189. URL: https://arxiv.org/abs/2210.05189.

[2] Jimmy Ba and Rich Caruana. “Do deep nets really need to be deep?” In: *Advances in Neural Information Processing Systems* 27 (2014), pp. 2654–2662.

[3] Emilie Frost, Eric Veith, and Lars Fischer. “Robust and Deterministic Scheduling of Power Grid Actors”. In: *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. Vol. 1. 2020, pp. 100–105. DOI: 10.1109/CoDIT49905.2020.9263948.

[4] Eric Veith, Arlena Wellßow, and Mathias Uslar. “Learning new attack vectors from misuse cases with deep reinforcement learning”. In: *Frontiers in Energy Research* (2023).

[5] Eric MSP Veith. *Universal Smart Grid Agent for Distributed Power Generation Management*. Logos Verlag Berlin GmbH, 2017.

Bibliography II

[6] Thomas Wolgast, Eric MSP Veith, and Astrid Nieße. "Towards Reinforcement Learning for Vulnerability Analysis in Power-Economic Systems". In: *DACH+ Energy Informatics 2021: The 10th DACH+ Conference on Energy Informatics*. Freiburg, Germany, Sept. 2021, pp. 1–20.