*Article*

# Safe reinforcement learning for multi-energy management systems with known constraint functions

**Glenn Ceusters [1,2,3,*], Luis Ramirez Camargo [2], Rüdiger Franke [1], Ann Nowé [3], Maarten Messagie [2]**

[1] ABB, Hoge Wei 27, 1930 Zaventem, Belgium; glenn.ceusters@be.abb.com; ruediger.franke@de.abb.com;
[2] Vrije Universiteit Brussel (VUB), ETEC-MOBI, Pleinlaan 2, 1050 Brussels, Belgium; glenn.leo.ceusters@vub.be; Luis.Ramirez.Camargo@vub.be; maarten.messagie@vub.be;
[3] Vrije Universiteit Brussel (VUB), AI-lab, Pleinlaan 2, 1050 Brussels, Belgium; gceusters@ai.vub.ac.be; ann.nowe@ai.vub.ac.be;
* **Correspondence:** glenn.ceusters@be.abb.com

Received: 08/07/2022

**Abstract:** Reinforcement learning (RL) is a promising optimal control technique for multi-energy management systems. It does not require a model *a priori* - reducing the upfront and ongoing project-specific engineering effort and is capable of learning better representations of the underlying system dynamics. However, *vanilla* RL does not provide constraint satisfaction guarantees - resulting in various unsafe interactions within its safety-critical environment. In this paper, we present two novel safe RL methods, namely SafeFallback and GiveSafe, where the safety constraint formulation is decoupled from the RL formulation and which provides hard-constraint satisfaction guarantees both during training (exploration) and exploitation of the (close-to) optimal policy. In a simulated multi-energy systems case study we have shown that both methods start with a significantly higher utility (i.e. useful policy) compared to a *vanilla* RL benchmark (94,6% and 82,8% compared to 35,5%) and that the proposed SafeFallback method even can outperform the *vanilla* RL benchmark (102,9% to 100%). We conclude that both methods are viably safety constraint handling techniques capable beyond RL, as demonstrated with random agents while still providing hard-constraint guarantees. Finally, we propose fundamental future work to i.a. improve the constraint functions itself as more data becomes available.

**Keywords:** reinforcement learning; constraints; multi-energy systems; energy management system

---

## Highlights

- A (near-to) optimal multi-energy management policy can be learned safely
- Any reinforcement learning algorithm can be used safely
- Constraint functions increase the initial utility of the policy
- Constraints can be formulated independently from the (optimal) control technique
- Better policies can be found starting with an initial safe fallback policy

## 1. Introduction

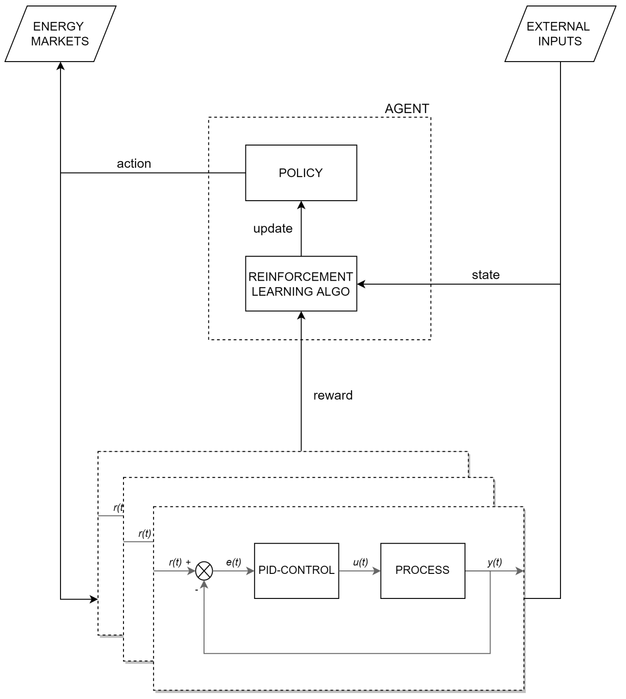
Energy systems continue to become increasingly interconnected with each other as the energy technologies that allow for this sector coupling are more mature and are being more widely implemented. This allows for an integrated control strategy that further can enhance the overall efficiency and performance of these so-called multi-energy, -carrier, -commodity or -utility systems. Furthermore, these multi-energy systems allow for the utilization of the flexibility (i.e. storage, controllable loads) within and across all energy carriers. This integrated control strategy then typically [1] has an economic or environmental oriented objective function or a combination thereof and therefor being multi-objective.

To ensure the optimum or Pareto optimum level of operation of such multi-energy systems, specific set-points are required to establish and maintain the desired objective (e.g. minimization of the energy costs or $CO_2$-equivalent emissions) while still fulfilling all system constraints [2]. As flexibility utilization exhibits dynamic behaviour and introduces a dependency between successive time steps, optimisation across (or considering) numerous time steps is necessary. Additionally, multiple uncertainties (i.e. variation in demands, pricing and weather) need to be managed so that the stability of the multi-energy system is preserved.

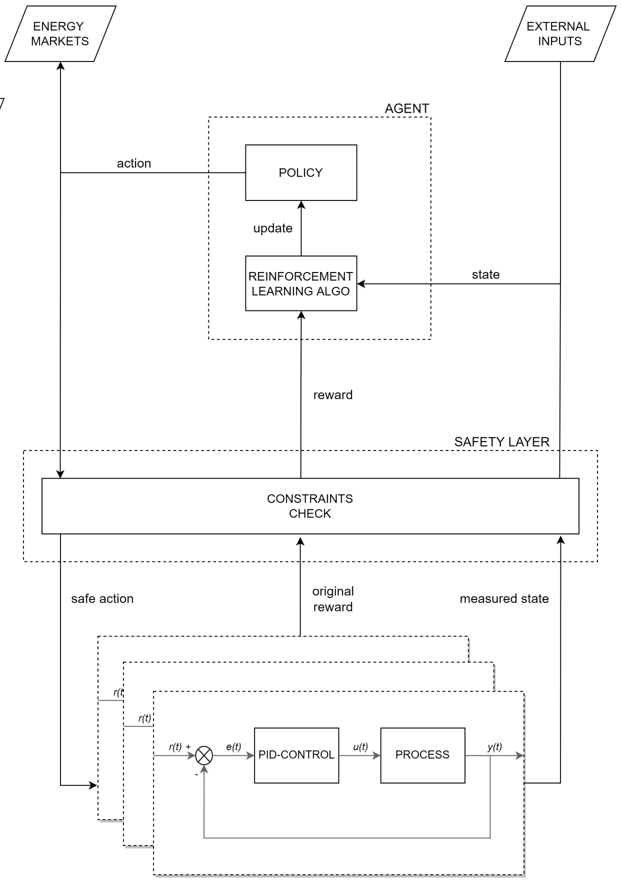
Model-predictive control (MPC) and reinforcement learning (RL) have recently been benchmarked within such a multi-energy management system context [3]. Ceusters *et al.* showed that RL-based energy management systems don't require a model *a priori* and that they can outperform linear MPC-based energy management systems after training. However, *vanilla* RL (see Figure 1a) performs a large number of unsafe interactions within its environment (e.g. neglecting the crucial energy balance constraint - especially problematic for non-grid-connected energy systems like thermal systems[1]) in order to learn an approximation of the optimal policy [3]. Therefore, we propose to always perform a constraint check before sending the actions to the real environment, guarantying hard-constraint satisfaction during exploration (i.e. training) and exploitation (i.e. greedy execution) - see Figure 1b. The constraint formulation is decoupled from the optimal control formulation (RL in this case), so that any optimization algorithm can be used (i.e. rather then formulating specific safe RL algorithms, so that future - presumably better - optimization algorithms can easily be used instead). Note that, as shown in Figure 1, we assume that the continuous unconstrained error handling (i.e. minimization of the difference between a desired set-point and a measured process variable) is performed by proportional-integral-derivative (PID) controllers.

In section 2 related work is discussed and our research question is formulated, section 3 introduces the proposed methodologies, while in section 4 the tool chain, the simulated multi-energy system environment, the safety layer, the RL agent and the evaluation procedure are presented. Furthermore, section 5 discusses the results as well as proposing future work while section 6 presents the conclusion. Finally, Appendix A shows time series visualizations of the

[1] unless connected to a district heating system, but grid-connected energy system then essentially rely on a "higher level" control systems to satisfy this particular constraint



**(a)** *vanilla* (unsafe) reinforcement learning



**(b)** shielded (safe) reinforcement learning

**Figure 1.** block diagrams comparison: the feasibility of the RL agent's actions, being in a given state, are always checked against the *a priori* constraint functions acting as a safety layer - shielding the environment from unsafe (control) actions different policies, Appendix B the pseudo-code of the specific RL agent (TD3) and Appendix C the run-time statistics.

## 2. Related work

In recent years, there have been numerous of works that proposed RL for various applications within energy systems as reviewed by e.g. [4], [5] and [6]. The majority of these applications can be classified under a broader energy management problem. RL based energy management systems have even been proposed and demonstrated within the more specific (and arguably more challenging) multi-energy systems context. For example, Rayati *et al.* [7] were one of the first to apply RL, specifically Q-learning [8], for the energy management of a simulated multi-energy residential building, which they later extended with demand-side management capabilities [9]. Posteriorly, Mbuwir *et al.* [10] successfully applied RL (Q-learning) for a battery energy management system within a simulated residential multi-energy system. They inlcuded a back-up policy to over-rule the actions of the RL agent in case of constraint violation. Furthermore, Wang *et al.* [11] used a path tracking interior point method and a RL algorithm (Q-learning) for a bi-level interactive decision-making model with multiple agents in a regional multi-energy system. A multi-agent RL (Q-learning) approach was also proposed by Ahrarinouri *et al.* [12] and this for the energy management of a simulated residential multi-energy system. Around the same time, Ye *et al.* [13] proposed the usage of a deep RL algorithm, specifically a deep deterministic policy gradient (DDPG) [14] with a prioritized experience replay strategy, again within a simulated residential multi-energy system.

Moreover, Xu *et al.* [15] demonstrated an industrial multi-energy scheduling framework using a RL (Q-learning) based differential evolution approach that adaptively determines the optimal mutation strategy and its associated parameters. While Zhu *et al.* [16] demonstrated a multi-agent deep RL energy management system, using multi-agent counterfactual soft actor-critic (mCSAC) [17], for a simulated multi-energy industrial park. Furthermore, Ceusters *et al.* [3] presented an on- and off-policy multi-objective model-free RL approach, using proximal policy optimisation (PPO [18]) and twin delayed deep deterministic policy gradient (TD3 [19]) and they did benchmark this against a linear MPC - both derived from the general optimal control problem. They showed, on two separate simulated multi-energy systems, that the RL agents offer the potential to match and outperform the MPC.

However, as RL inherently requires the interaction with its environment, adequate measures are required to avoid the violation of the environmental specific constraints both during online training as during pure policy execution (i.e. exploration and exploitation). All the works above have, knowingly (and thus reported as such) or non-knowingly, either neglected these specific environmental constraints or greatly simplified them - limiting the real-world use cases.

Concerning safe RL beyond the energy systems management field, García and Fernández [20], in a comprehensive review, identified and classified two broader approaches: (1) modifying the optimality criteria with a safety factor; (2) modifying the exploration process by incorporating external knowledge or the guidance of a risk metric. More recently, Dulac-Arnold *et al.* [21] identified nine challenges that must be addressed to implement RL in real-world problems, including safety constraint violation. While Brunke *et al.* [22] provided a broader safe learning review across both the control theory research space as well as the RL research space. More specifically, they showed (1) learning-based control approaches that start with an *a priori* model to safely improve the policy under the uncertain system dynamics, (2) safe RL approaches that does not need a model or even constraints in advance - but then also does not provide strict safety guarantees (yet encourages safety or robustness), and (3) approaches that provide safety certificates of any learned control policy (and then over-ruling unsafe actions).

The reviewed literature shows that RL is a promising and widely proposed approach for various applications in energy systems (and other domains, not discussed here), as well as for energy management systems specifically. It is also clear that the transition of RL towards real-world applications is not trivial and requires special attention concerning safety. Multiple safe (reinforcement) learning approaches exist, ranging in level of safety namely (from lower to higher level): (1) soft-constraint satisfaction, (2) chance-constraint satisfaction and (3) hard-constraint satisfaction. However, safe RL where the safety constraint formulation is decoupled from the RL formulation and which provides hard-constraint satisfaction guarantees both during training (exploration) and exploitation of the (close-to) optimal policy has - to the best of our knowledge - never been proposed and demonstrated for the energy management of multi-energy systems.

## 3. Proposed methodology

Following the standard RL formulation of the state-value function, yet extending this towards constraints subjection, the objective is to find a policy $\pi$, which is a mapping of states, $s \in S$, and actions, $a \in A(s)$, that maximizes an expected sum of discounted rewards and is subject to constraint sets $X$ and $U$:

$$\max_{\pi} \left( E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \right\} \right) \tag{1a}$$

$$\begin{aligned} s.t. \quad & s_t = s & t \in \mathbb{T}_0^{+\infty} = 0, \ldots, +\infty \quad & \text{(1b)} \\ & s_t \in X & t \in \mathbb{T}_1^{+\infty} = 1, \ldots, +\infty \quad & \text{(1c)} \\ & a_t \in U & t \in \mathbb{T}_0^{+\infty} = 0, \ldots, +\infty \quad & \text{(1d)} \end{aligned}$$

where $E_{\pi}$ is the expected value, following the policy $\pi$, of the rewards $R$ discounted with the discount factor $\gamma$ over an infinite sum at any time step $t$.

Note that Equation 1a is a discrete[2] time-invariant infinite-horizon stochastic optimal control problem, as also indicated by [3], yet differs from the standard formulation of RL due to the *a priori* constraint functions in the sets $X$ and $U$. We acknowledge that methods without *a priori* constraint functions exists, yet this can - under the state-of-the-art as reviewed by [22] - only reach safety level 2 at best (chance-constraint satisfaction). Rather than proposing a specific safe

[2] as the continuous error handling is performed by PID-controllers, see Figure 1

RL algorithm, we propose to decouple the *a priori* constraint function formulation from the (RL) agent so that any (new RL) algorithm can be used - while always guarantying the hard-constraint satisfaction.

### 3.1. SafeFallback method

The first method we propose relies on an *a priori* safe fallback policy $\pi^{safe}$, which typically can be derived through classic control theory in the form of a set of hard-coded rules (i.e. a simple rule-based policy, e.g. a priority-based energy management strategy - which is commonly available or easily constructable). As we furthermore assume that the constraint functions are given, we can simply check if the selected actions $a$ while in state $s$ satisfy the constraint conditions. When the constraints conditions are satisfied, the selected actions $a$ are *considered* to be safe actions $a^{safe}$ and are then executed in the environment so that the next state $s'$, the reward $r$ and done signal $d$ are observed - which is the regular experience tuple $(s, a^{safe}, r, s', d)$ for the RL agent. However, if the constraint conditions are violated, the selected action $a$ is overruled by the safe action $a^{safe}$ using the *a priori* safe fallback policy $\pi^{safe}$. Now not only the experience tuple $(s, a^{safe}, r, s', d)$ is formed, but also the experience tuple $(s, a, r - c, s', d)$ containing the infeasible action and additional negative reward (i.e. cost, $c$). The pseudo-code is given in algorithm 1.

**Algorithm 1:** SafeFallback

```
1 Input: initialize RL algorithm, initialize constraint functions in sets $X$ and $U$, initialize
    safe fallback policy $\pi^{safe}$
2 for $k = 0, 1, 2, \dots$ do
3     Observe state $s$ and select action $a$
4     if constraint check = True then
          keep selected action $a$ as safe action $a^{safe}$
      else
          get safe action $a^{safe}$ from safe fallback policy $\pi^{safe}$
      end
5     Execute $a^{safe}$ in the environment
6     Observe next state $s'$, reward $r$ and done signal $d$ to indicate whether $s'$ is terminal
7     Give experience tuple $(s, a^{safe}, r, s', d)$ and if $a^{safe} \neq a : (s, a, r - c, s', d)$ with cost $c$
8     If $s'$ is terminal, reset environment state
  end
```

### 3.2. GiveSafe method

Our second method does not require an *a priori* safe fallback policy, yet relies on the RL agent itself to pass safe actions $a^{safe}$ - which can again be checked by the given constraint conditions. If the selected actions $a$ while in state $s$ passes the constraint check, the safe actions $a^{safe}$ get executed in the environment and a regular experience tuple $(s, a^{safe}, r, s', d)$ is received. However, if the constraints get violated the RL agent receives the experience tuple $(s, a, r - c, s, d)$. Hence, the transition towards the next state is not observed (as the infeasible action is not executed) and a cost $c$ (i.e. negative reward) is given. The RL agent then selects a new action $a$ and a new constraint check is done. This is repeated until the constraint check gets passed and the selected action is considered to be a safe action $a^{safe}$. This safe action is then again executed in the environment and a regular experience tuple is received. The pseudo-code is given in algorithm 2 and a graphical representation, in the form of a Markov Chain, in Figure 2.

**Algorithm 2:** GiveSafe

```
1 Input: initialize RL algorithm, initialize constraint functions in sets $X$ and $U$
2 for $k = 0, 1, 2, \dots$ do
3     Observe state $s$ and select action $a$
4     if constraint check = True then
          keep selected action $a$ as safe action $a^{safe}$
      else
5         while constraint check = False do
              give experience tuple $(s, a, c, s, d)$ with cost $c$
              agent selects new action $a$
              check constraints
          end
          return safe action $a^{safe}$
      end
6     Execute $a^{safe}$ in the environment
7     Observe next state $s'$, reward $r$ and done signal $d$ to indicate whether $s'$ is terminal
8     Give experience tuple $(s, a^{safe}, r, s', d)$
9     If $s'$ is terminal, reset environment state
  end
```

## 4. Simulated case study

### 4.1. Toolchain

A multi-energy systems simulation model, that was developed in [3], was used in order to test the proposed methods without consequences (i.e. without the risk of violating real-life constraints and its associated harm) in an energy management context. It is a `Modelica` [23] model, as it allowed for the convenient construction of the real-life *presumed* system dynamics using multi-physical first-principle equations and due to the available highly specialized libraries, elementary components and its object-oriented nature.

This `Modelica` model is then exported as a co-simulation *functional mock-up unit* (FMU), similar to [24] , and wrapped into an `OpenAI` gym *environment* [25] in `Python`, similar to [3,26]. The architecture of the tool-chain is shown in Figure 3. Notice that the Differential Algebraic Equations solver is part of the co-simulation FMU and that the `do_step()` method in `PyFMI` [27] is used over `simulate()` - again as in [3] due to the significant run-time speed-up when initialized properly.

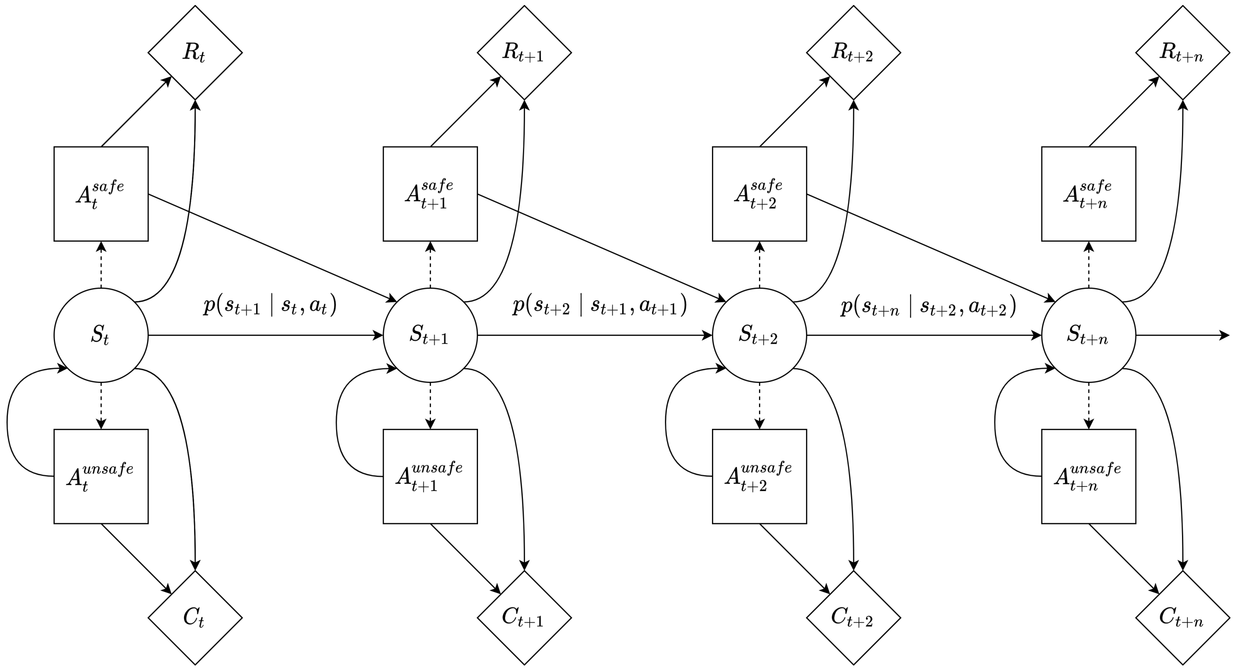


**Figure 2.** Markov Chain of algorithm 2. When the selected action is infeasible (does not satisfy to the constraints), that unsafe action $A_t^{unsafe}$ is not executed in the environment so no transition to the next state $S_{t+1}$ is observed and a cost $C_t$ is given. When the selected action is feasible (satisfies the constraints), that safe actions $A_t^{safe}$ is executed in the environment so a transition to the next state $S_{t+1}$ is observed with probability $p(s_{t+1} \mid s_t, a_t)$ and a reward $R_t$ is given.

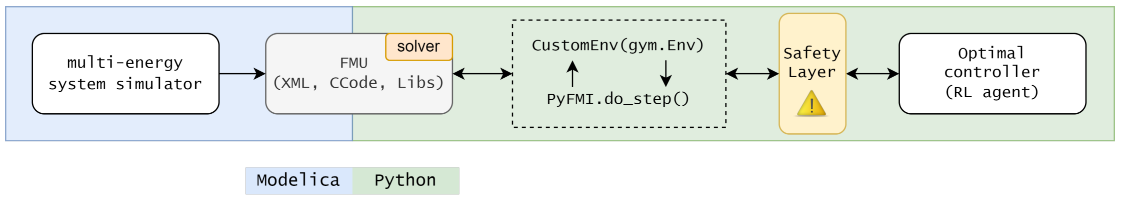


**Figure 3.** architecture of the tool-chain

## *4.2. Simulation model*

The considered multi-energy system is similar to the one from [3], yet without the gas turbine (back-up genset), and has the following structure:

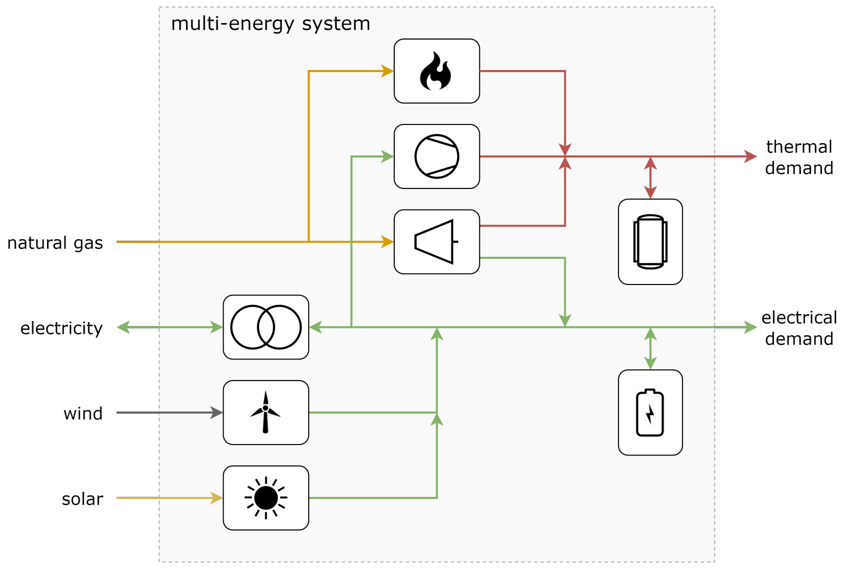


**Figure 4.** structure of the simulated multi-energy system

It includes (from left to right, from top to bottom): an electric transformer, a wind turbine, a photovoltaic (PV) installation, a natural gas boiler, a heat pump (HP), a combined heat and power (CHP) unit, a thermal energy storage system (TESS) and a battery energy storage system (BESS). The dimensions of the considered multi-energy system are also from [3] and are summarised in Table 1.

| **Energy asset** | **Input** | **Output** | $\mathbf{P_{nom}}$ | $\mathbf{P_{min}}$ | $\mathbf{E_{nom}}$ |
|---|---|---|---|---|---|
| transformer | elec | elec | $+\infty$ | $-\infty$ | |
| wind turbine | wind | elec | 0.8 $MW_e$ | 1.5 % | |
| solar PV | solar | elec | 1.0 $MW_e$ | 0 % | |
| boiler | $CH_4$ | heat | 2.0 $MW_{th}$ | 10 % | |
| heat pump | elec | heat | 1.0 $MW_{th}$ | 25 % | |
| CHP | $CH_4$ | heat | 1.0 $MW_{th}$ | 50 % | |
| | $CH_4$ | elec | 0.8 $MW_e$ | 50 % | |
| TESS | heat | heat | +0.5 $MW_{th}$ | -0.5 $MW_{th}$ | 3.5 MWh |
| BESS | elec | elec | +0.5 $MW_e$ | -0.5 $MW_e$ | 2.0 MWh |

**Table 1.** dimensions of the multi-energy system

*4.3. Safety layer*

The constraint functions, Equation 1c and Equation 1c, are in this case study specifically:

$$Q_{boil}^{min} \times \gamma_{boil}^{t} \leq Q_{boil}^{t} \leq Q_{boil}^{max} \times \gamma_{boil}^{t} \qquad \forall t,\ \ \gamma_{boil}^{t} \in \{0,1\} \tag{2a}$$

$$\begin{bmatrix} Q_{hp}^{min} \\ P_{hp}^{min} \end{bmatrix} \times \gamma_{hp}^{t} \leq \begin{bmatrix} Q_{hp}^{t} \\ P_{hp}^{t} \end{bmatrix} \leq \begin{bmatrix} Q_{hp}^{max} \\ P_{hp}^{max} \end{bmatrix} \times \gamma_{hp}^{t} \qquad \forall t,\ \ \gamma_{hp}^{t} \in \{0,1\} \tag{2b}$$

$$\begin{bmatrix} Q_{chp}^{min} \\ P_{chp}^{min} \end{bmatrix} \times \gamma_{chp}^{t} \leq \begin{bmatrix} Q_{chp}^{t} \\ P_{chp}^{t} \end{bmatrix} \leq \begin{bmatrix} Q_{chp}^{max} \\ P_{chp}^{max} \end{bmatrix} \times \gamma_{chp}^{t} \qquad \forall t,\ \ \gamma_{chp}^{t} \in \{0,1\} \tag{2c}$$

$$Q_{tess}^{min} \leq Q_{tess}^{t} \leq Q_{tess}^{max} \qquad \forall t \tag{2d}$$

$$P_{bess}^{min} \leq P_{bess}^{t} \leq P_{bess}^{max} \qquad \forall t \tag{2e}$$

$$Q_{production}^{t} = Q_{demand}^{t} \qquad \forall t \tag{2f}$$

where $Q_{boil}^{t}$, $Q_{hp}^{t}$, $Q_{chp}^{t}$ and $Q_{tess}^{t}$ are the thermal powers of the natural gas boiler, the heat pump, the combined heat and power unit (CHP) and the thermal energy storage system (TESS) respectively while $P_{hp}^{t}$, $P_{chp}^{t}$ and $P_{bess}^{t}$ represent the electrical powers of the heat pump, CHP and battery energy storage system (BESS), all constraint by its associated minimal and maximal power (see Table 1). Furthermore, $\gamma_{boil}^{t}$, $\gamma_{hp}^{t}$ and $\gamma_{chp}^{t}$ are binary variables that turn on/off the given asset (i.e. as the minimal powers are not zero). Yet these constraints, i.e. Equation 2a till Equation 2e, are easily handled by the dimensions of the (control) action space itself, i.e. Equation 4b till Equation 4f, and therefor do not require a specific constraint check.

While the electrical energy balance is always fulfilled (given the assumption that the electrical grid connection is sufficiently large), the thermal energy balance (Equation 2f) does require special attention in order to achieve hard-constraint satisfaction. No additional constraints are considered in this case study (e.g. ramping rates, minimal run- and down-time) as energy balance equations are considered to be the most limiting constraints in energy management problems. Writing out the thermal energy balance in more detail and relaxing the equality constraint formulation then becomes:

$$\left| Q_{boil}^{t} + Q_{hp}^{t} + Q_{chp}^{t} + Q_{tess}^{t} - Q_{demand}^{t} \right| \leq Q_{tol} \qquad \forall t \tag{3a}$$

$$\begin{aligned} |A_{boil}^{t} \cdot \eta_{boil} \cdot Q_{boil}^{max} + A_{hp}^{t} \cdot \frac{COP_{hp}}{COP_{hp}^{max}} \cdot Q_{hp}^{max} + A_{chp}^{t} \cdot \eta_{chp}^{th} \cdot Q_{chp}^{max} & \\ + A_{tess}^{t} \cdot f(SOC_{tess}^{t}) - Q_{demand}^{t}| \leq Q_{tol} & \qquad \forall t \end{aligned} \tag{3b}$$

$$\begin{aligned} |A_{boil}^{t} \cdot f(T_{boil}^{t}) \cdot Q_{boil}^{max} + A_{hp}^{t} \cdot \frac{f(T_{evap}^{t}, T_{cond}^{t})}{COP_{hp}^{max}} \cdot Q_{hp}^{max} & \\ + A_{chp}^{t} \cdot f(P_{chp}^{t}, Q_{chp}^{t}, T_{env}^{t}) \cdot Q_{chp}^{max} + A_{tess}^{t} \cdot f(\overline{T_{tess}^{t}}) - Q_{demand}^{t}| \leq Q_{tol} & \qquad \forall t \end{aligned} \tag{3c}$$

where $A^{t}$ are the (control) actions, $\eta$ the energy efficiencies, $COP$ the coefficient of performance, $SOC$ the state of charge and $T$ various specific temperatures (i.e. $T_{boil}^{t}$ the return temperature to the boiler, $T_{evap}^{t}$ the evaporator temperature of the heat pump, $T_{cond}^{t}$ the condenser temperature of the heat pump, $T_{env}^t$ the environmental air temperature and $\overline{T_{tess}^t}$ the average temperature in the stratified hot water storage tank). Note that the different functions $f(\cdot)$ from Equation 3c are typically not trivial to *model* accurately. Therefor we assume the availability of a historical dataset to supervisory learn (using a feed-forward neural network) the function between the thermal power and the action directly (i.e. $Q_{asset}^t = f(A_{asset}^t, \chi_{asset}^t)$), with the possibility to include informative exogenous variables $\chi_{asset}^t$.

| Energy asset | R2-score | MAE | NMAE |
|---|---|---|---|
| boiler | 99.92% | 7.2 kW | 0.34% |
| heat pump | 99.74% | 6.1 kW | 0.62% |
| CHP | 99.86% | 4.3 kW | 0.38% |
| TESS | 96.22% | 12.9 kW | 1.37% |
| BESS | 99.43% | 4.6 kW | 0.46% |

**Table 2.** Safety layer model metrics with `test_size` of 0.25. Mean Absolute Error (MAE), Normalised Mean Absolute Error (NMAE) by range, i.e. NMAE = MAE / range(actual values)

## 4.4. Reinforcement learning agent

The fully observable discrete-time Markov decision process (MDP) is formulated as the tuple $\langle S, A, P_a, R_a \rangle$ so that:

$$S^t = (E_{th}^t,\ E_{el}^t,\ P_{wind}^t,\ P_{solar}^t,\ X_{el}^t,\ SOC_{tess}^t,\ SOC_{bess}^t,\ h^t,\ d^t) \qquad S^t \in S \tag{4a}$$

$$A_{boil}^t = (0,\ A_{boil}^{min} \rightarrow A_{boil}^{max}) \qquad A_{boil}^t \in A \tag{4b}$$

$$A_{hp}^t = (0,\ A_{hp}^{min} \rightarrow A_{hp}^{max}) \qquad A_{hp}^t \in A \tag{4c}$$

$$A_{chp}^t = (0,\ A_{chp}^{min} \rightarrow A_{chp}^{max}) \qquad A_{chp}^t \in A \tag{4d}$$

$$A_{tess}^t = (A_{tess}^{min} \rightarrow A_{tess}^{max}) \qquad A_{tess}^t \in A \tag{4e}$$

$$A_{bess}^t = (A_{bess}^{min} \rightarrow A_{bess}^{max}) \qquad A_{bess}^t \in A \tag{4f}$$

$$R_a = -(a \times L_{cost}^t + b \times L_{comfort}^t) - c \tag{4g}$$

where $E_{th}^t$ is the thermal demand, $E_{el}^t$ the electrical demand, $P_{wind}^t$ the electrical wind in-feed, $P_{solar}^t$ the electrical solar in-feed, $X_{el}^t$ the electrical price signal (i.e. day-ahead spot price), $SOC_{tess}^t$ the state-of-charge (SOC) of the TESS, $SOC_{bess}^t$ the SOC of the BESS, $h^t$ the hour of the day and $d^t$ the day of the week all at the $t$-th step, which consitute the state-space $S$. The action-space $A$ includes the control set-points from, $A_{boil}^t$ the natural gas boiler, $A_{hp}^t$ the heat pump, $A_{chp}^t$ the CHP unit, $A_{tess}^t$ the TESS and $A_{bess}^t$ the BESS all between a minimum and maximum power rate as shown in Table 1.

Furthermore, the reward function $R_a$ is the negative loss in energy costs $L_{cost}^t$ and loss in comfort $L_{comfort}^t$ both at the $t$-th time-step with multi-objective scaling constants $a$ and $b$ and with an additional cost $c$ when the constraint are *expected* to be violated[3]. The loss in comfort is defined as $|E^t_{th} - Q^t|$, where $Q^t$ is the thermal energy production. The electrical demand and natural gas consumption can always be fulfilled by (buying from) their respective *infinitely* large main grid connection, i.e. within the `Modelica` simulation model, it is assumed that the grid connections are sufficiently large. Note that the loss in comfort $L^t_{comfort}$ is bound by the tolerance of the constraints. This term, in the reward function, therefore serves as a fine-tuning mechanism to further minimize the loss in comfort within those bound and to mitigate the modelling error of the constraints itself (see Table 2 for the quality of the constraint functions).

The energy costs $L^t_{cost}$ is in EUR with scaling factor $a = 1/10$, the loss in comfort $L^t_{comfort}$ is in Watt with scaling factor $b = 1/5e5$ and cost $c = 1$ in algorithm 1 and $c = -50 + (10 \textbf{ if } A^t_{chp} > 0.5 \textbf{ else } 0)$ in algorithm 2. The state-space $S$ is normalized and all actions in the action-space $A$ are scaled between $[+1, -1]$.

Finally, we use a twin delayed deep deterministic policy gradient (TD3) agent, as it is considered one of the state-of-the-art RL algorithms, from the `stable baseline` [28] implementations and this with the following hyper-parameters (found after an hyper-parameter optimization study, similar to [3]). The pseudo-code of the TD3 algorithm is given in Appendix B.

| Hyper-parameters: TD3 | algorithm 1 | algorithm 2 | unsafe |
|---|---|---|---|
| gamma | 0.7 | 0.95 | 0.9 |
| learning_rate | 0.000583 | 0.000119 | 0.0003833 |
| batch_size | 16 | 16 | 100 |
| buffer_size | 1e6 | 1e5 | 1e5 |
| train_freq | 1e0 | 1e1 | 2e3 |
| gradient_steps | 1e0 | 1e1 | 2e3 |
| noise_type | normal | normal | normal |
| noise_std | 0.183 | 0.791 | 0.329 |

**Table 3.** Best found TD3 hyper-parameters

## *4.5. Evaluation*

The performance, in terms of energy cost minimization subject to the (thermal comfort) constraint fulfillment, of the proposed methods algorithm 1 and algorithm 2 is compared against an unconstrained (and therefore unsafe) RL agent and compared to safe and unsafe random agents - serving as minimal performance benchmarks - using a year-long training environment and a week-long evaluation environment while participating in a day-ahead electricity market. The model-predictive controller from [3] is here not considered, as constraints can be formulated directly in the method.

[3] as these unsafe actions are not executed in the environment - see algorithm 1 and algorithm 2

## 5. Results and discussion

The simulation results of the objective values (i.e. rewards) are shown, in Table 4, both in absolute values as relative to the unconstrained RL benchmark. The constraint tolerance is shown relative to the total demand (0% = all thermal demand fulfilled).

| Optimal controller | Objective value | | Constraint tolerance |
|---|---|---|---|
| Unsafe TD3 | -5.043 | 100,0% | 21,0% |
| Unsafe Random | -14.223 | 35,5% | 146,0% |
| SafeFallback TD3 | **-4.899** | **102,9%** | 10,1% |
| SafeFallback Random | -5.331 | 94,6% | **7,0%** |
| GiveSafe TD3 | -5.137 | 98,2% | 10,0% |
| GiveSafe Random | -6.089 | 82,8% | 15,1% |

**Table 4.** 5-run average policy performance with a training budget of 15-years worth of time steps per run (i.e. 525.150 time steps per run)

These results (Table 4) show that algorithm 1 (SafeFallback - 102,9%) outperforms algorithm 2 (GiveSafe - 98,2%) and the *vanilla* unsafe TD3 benchmark (100%). This as the *a prior* safe fallback policy has the highest utility before training (94,6% compared to 82,8% and 35,5%), indicating the additionally given expert knowledge. The additional expert knowledge (of the safe fallback policy of algorithm 1 itself) is reflected in the constraint tolerance as well (7,0%), yet is pushed to a limit of 10,1%. algorithm 2 has initially a higher constraint tolerance (the maximum tolerance as set in Equation 3c), yet is pushed to approximately the same limit (10,0%). Both methods are therefore, as intended, significantly safer than the *vanilla* TD3 benchmark - which has an initial constraint tolerance of 146,0% and reaching only 21,0%.

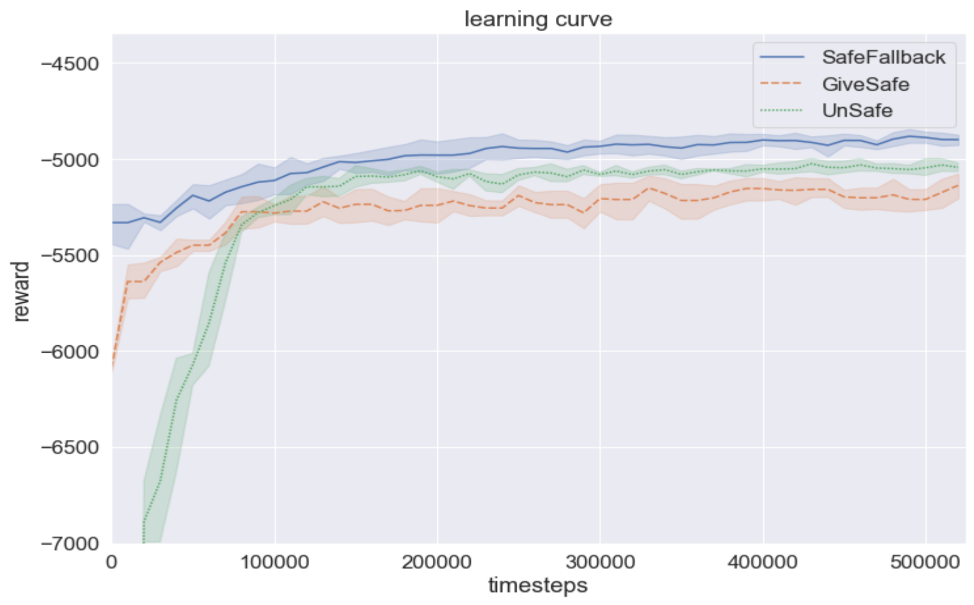


**Figure 5.** 5-run average learning curves with a training budget of 15-years worth of time steps per run (i.e. 525.150 time steps per run)

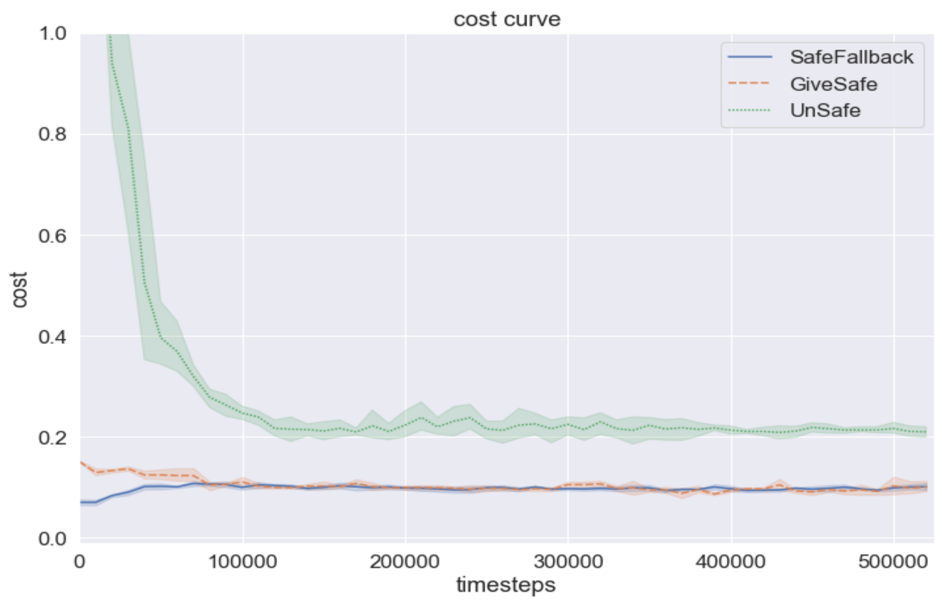


**Figure 6.** 5-run average cost curve (i.e. constraint tolerance) with a training budget of 15-years worth of time steps per run (i.e. 525.150 time steps per run)

The learning curves of the TD3 agents are presented in Figure 5, where the initial (at time step 0) and final (at time step 525.150) results are the figures reported in Table 4. We observe a steep initial learning rate, low variance, and a stable (slightly increasing) performance with increasing number of interactions with its environment. We also observe that algorithm 1 (SafeFallback) and algorithm 2 (GiveSafe) have a significantly higher initial performance (before any training has occurred, i.e. at time step 0) compared to its *vanilla* unsafe TD3 counterpart. This, again, due to the *a priori* expert knowledge in the form of a known safe fallback policy and in the form of safety constraint equations. The unsafe RL agent only reaches the *initial* performance (-6.089) of algorithm 2 after $\sim$ 50.000 time steps (1 year and 5 months) and of algorithm 1 (-5.331) after $\sim$ 85.000 time steps (2 years and 5 months).

The cost curves (constraint tolerance) of the TD3 agents are presented in Figure 6. We observe a steep initial decrease of the constraint tolerance of the *vanilla* TD3 agent, yet never converging to the safety threshold as defined by Equation 3c - while algorithm 1 and algorithm 2 never exceed this threshold (i.e. proving the hard-constraint satisfaction during training). The constraint tolerance convergence of both methods is less steep and reaches a stable performance ($\sim$ 10%) after approximately 3 years ($\sim 1e^5$ time steps).

However, the performance in terms of both the utility (objective value) and cost (constraint tolerance) of algorithm 1 (SafeFallback) and algorithm 2 (GiveSafe) relies on an accurate formulation of the actual constraints, i.e. the accurate formulation of Equation 3c in this case study. As presented in subsection 4.3, this is not always trivial - especially for the TESS and the HP. When we replace the pre-trained TESS and HP functions from the safety layer, by simpler (less accurate) white-box equations we observe a reduction in performance. For example, for algorithm 1 by its initial objective value of -5.331 to -5.509 and its initial constraint tolerance of 7,0% to 10,1%. This problem can be mitigated however, by artificially lowering $Q_{tol}$ from Equation 3c.

Nevertheless, training a RL agent on a real safety-critical environment would only be possible with a sufficiently accurate safety layer (i.e. constraints) and using adequate *a priori* unknown hyper-parameters. Therefore, we propose the following future work:

- Providing chance-constraint satisfaction guarantees for when no constraint functions are available *a priori* (but only limited, known to be safe, operational data) and to safely improve the *a prior* constraint functions as more data becomes available. Exploration and exploitation will then never leave the safe region with high probability, by updating any statistical model (e.g. a Gaussian Process model).
- Further reducing the training budget (i.e. improving sample efficiency) and rolling out a fixed sequence of (robust[4]) control actions (e.g. using model-based RL agents) for day-ahead market *planning*.
- Robustness of the RL-based energy management systems under faulty and noisy measurements / observations and utilizing *online* hyper-parameter optimization methods (i.e. that the hyper-parameters are tuned during online training).

[4] The transition probability matrix can then also be used to generate a *robust* planning rather than a pure *most-likelihood* planning

## 6. Conclusion

This paper presented two novel safe RL methods, where the safety constraint formulation is decoupled from the RL (MDP) formulation and which provided strict constraint satisfaction guarantees both during training (exploration) and exploitation of the near-optimal policy. These methods are demonstrated in a multi-energy management systems context, where detailed simulation results are provided.

Both methods are viable safety constraint handling techniques capable beyond RL, as demonstrated by random agents while still providing strict safety guarantees. Preferably, however, algorithm 1 (SafeFallback) is used as it showed significantly higher overall performance, and as the availability of a simple safe fallback policy is common or relatively easily constructable (i.e. in the form of a simple rule-based policy, e.g. a priority-based control strategy).

## 7. Acknowledgement

This work has been supported in part by ABB n.v. and Flemish Agency for Innovation and Entrepreneurship (VLAIO) grant HBC.2019.2613.

### CRediT authorship contribution statement

**Glenn Ceusters**: Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Data curation, Writing - original draft, Visualization, Funding acquisition; **Luis Ramirez Camargo**: Conceptualization, Writing - review and editing, Supervision; **Rüdiger Franke**: Supervision; **Ann Nowé**: Writing - review and editing, Supervision; **Maarten Messagie**: Supervision.

## Appendix A  Simulations visualization

In this section, we show a time series visualisation sample (a week) of the found control policies. The first observation that can be made (in Figure A1) is the violently unsafe behavior (146,0% of constraint tolerance, Table 4) of the TD3 agent before training, which at this stage acts as a random agent. Specifically, a large thermal overproduction is the cause of the thermal discomfort and thus the constraint violation. The overproduction is avoided after training the TD3 agent (Figure A2). The policy itself has a high utility, yet now a significant thermal underproduction is observed (21,0% of constraint tolerance, Table 4). In practice, the natural gas boiler could be forced on to satisfy the thermal underproduction (yet this by itself would be an *a prior* "fallback" policy).

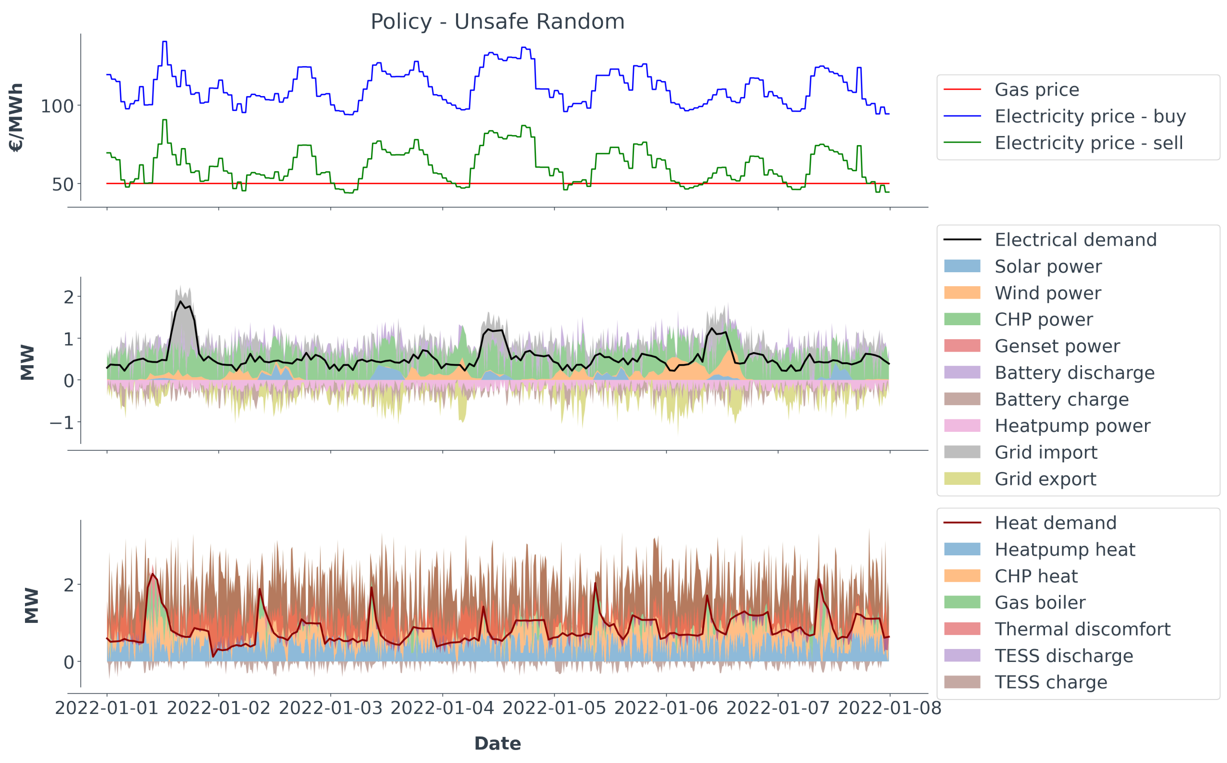


**Figure A1.** Policy visualization: unsafe random (or TD3 before training)

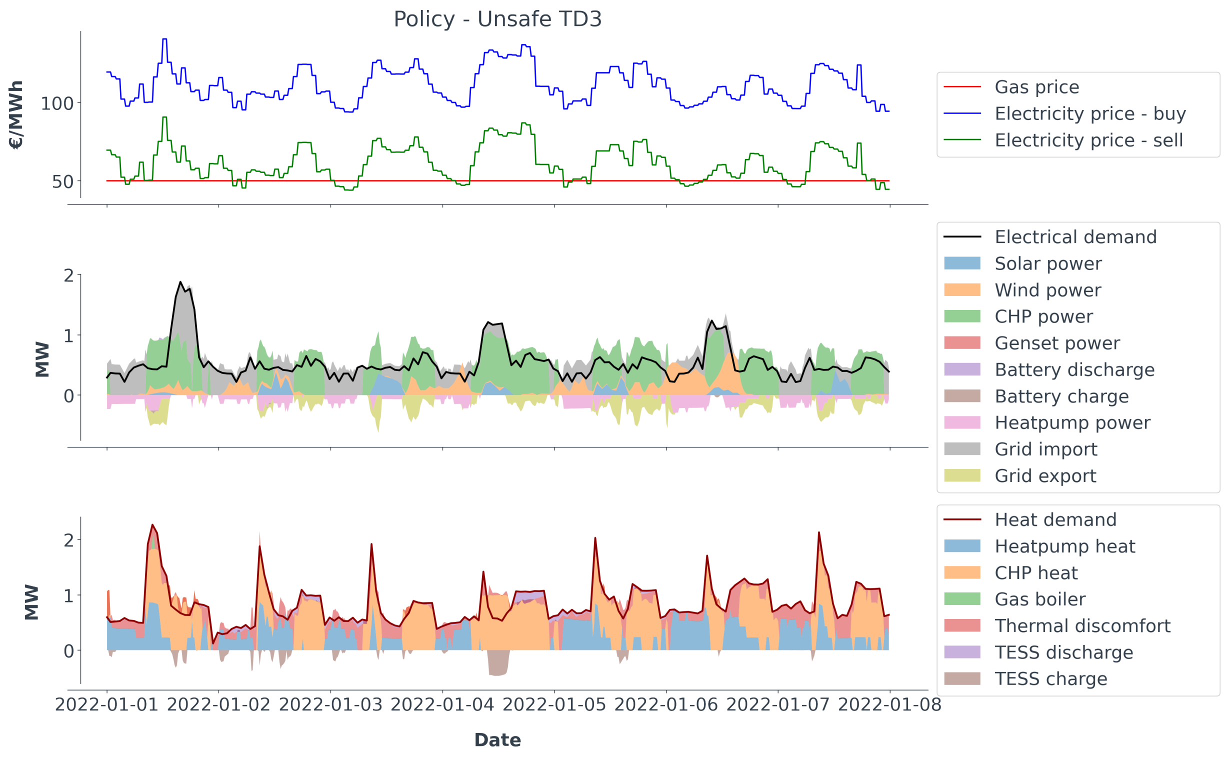


**Figure A2.** Policy visualization: unsafe TD3 (after unsafe training)

When analysing the SafeFallback (algorithm 1) policies, and comparing them against the *vanilla* unsafe TD3 policies, we observe safe behavior. Before training, the constraint check mostly fails - using the safe fallback policy. Initially (Figure A3), when the constraint check passes, safe random actions are observed (e.g. thermal "overproduction" is properly stored in the TESS). After *safely* training the TD3 agent, a policy with a high utility and a low constraint tolerance is observed (Figure A4). Thermal underproduction is still present, yet within the set bound $Q_{tol}$ from Equation 3c.

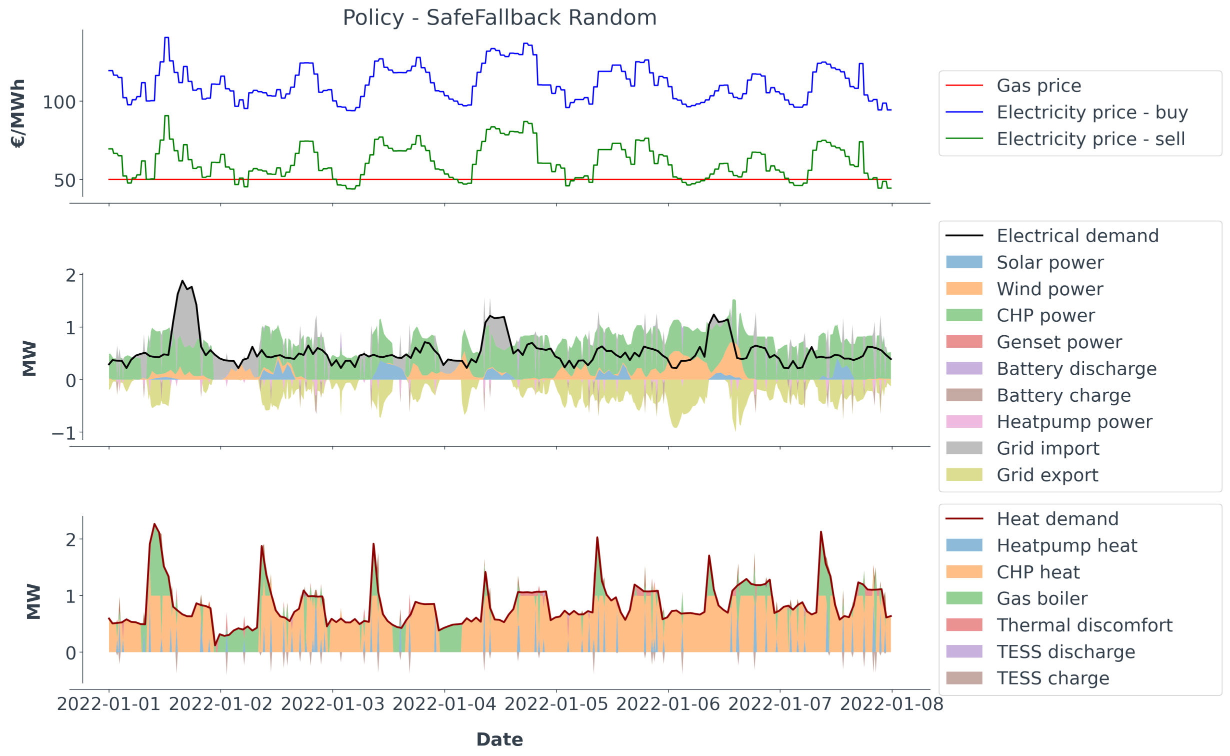


**Figure A3.** Policy visualization: SafeFallback random (or TD3 before training)

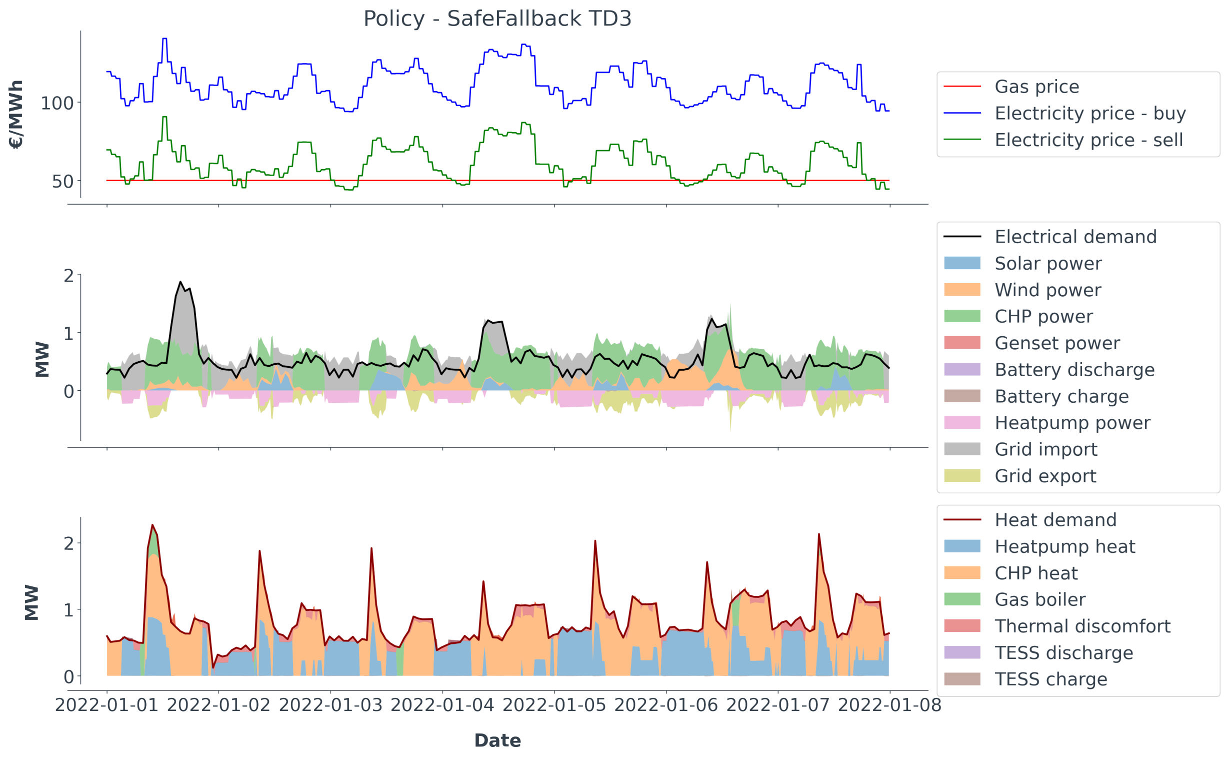


**Figure A4.** Policy visualization: SafeFallback TD3 (after safe training)

When analysing the GiveSafe (algorithm 2) policies, and comparing them against the *vanilla* unsafe TD3 policies, we again observe safe behavior. Before training, *all* actions are random but safe (e.g. thermal demand is matched by the thermal production or any thermal "overproduction" is stored in the TESS) - resulting in both a lower initial utility and higher constraint tolerance as Figure A3. After *safely* training the TD3 agent, a policy with a high utility and a low constraint tolerance is observed (Figure A6) - yet with a lower utility as Figure A4. Thermal underproduction is again still present, yet within the set bound $Q_{tol}$ from Equation 3c.

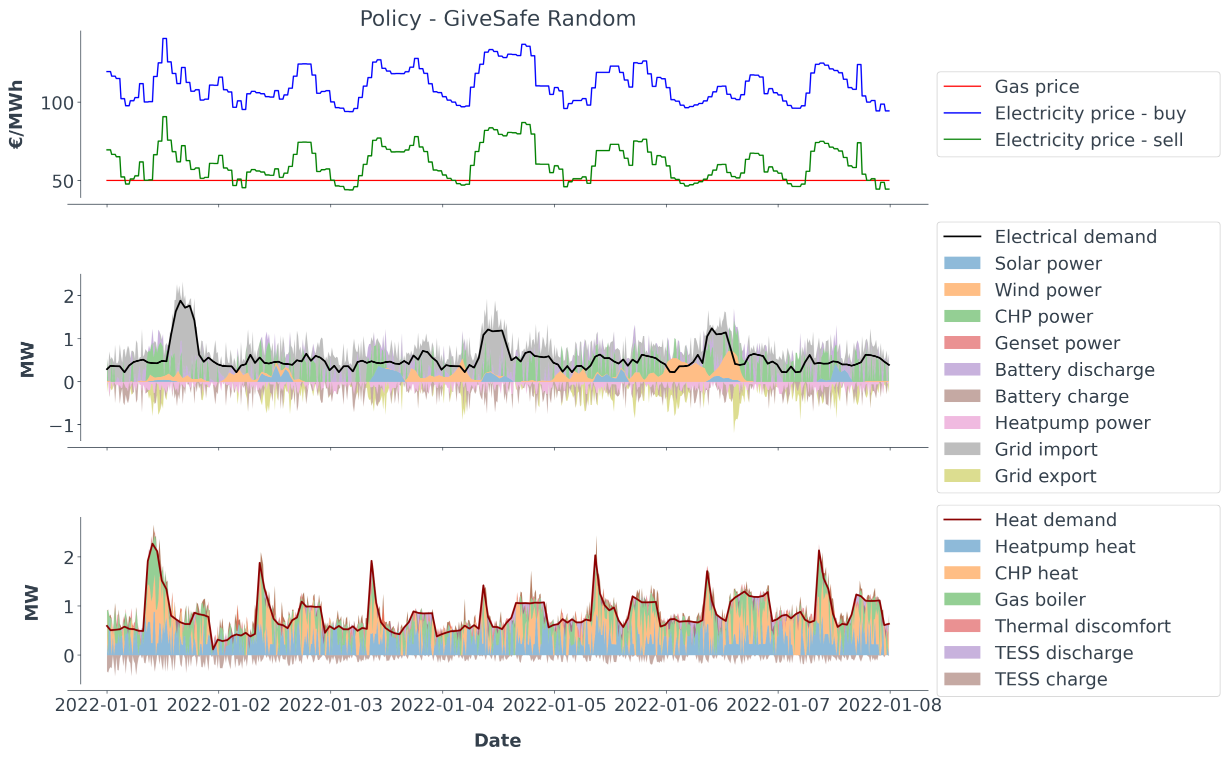


**Figure A5.** Policy visualization: GiveSafe random (or TD3 before training)

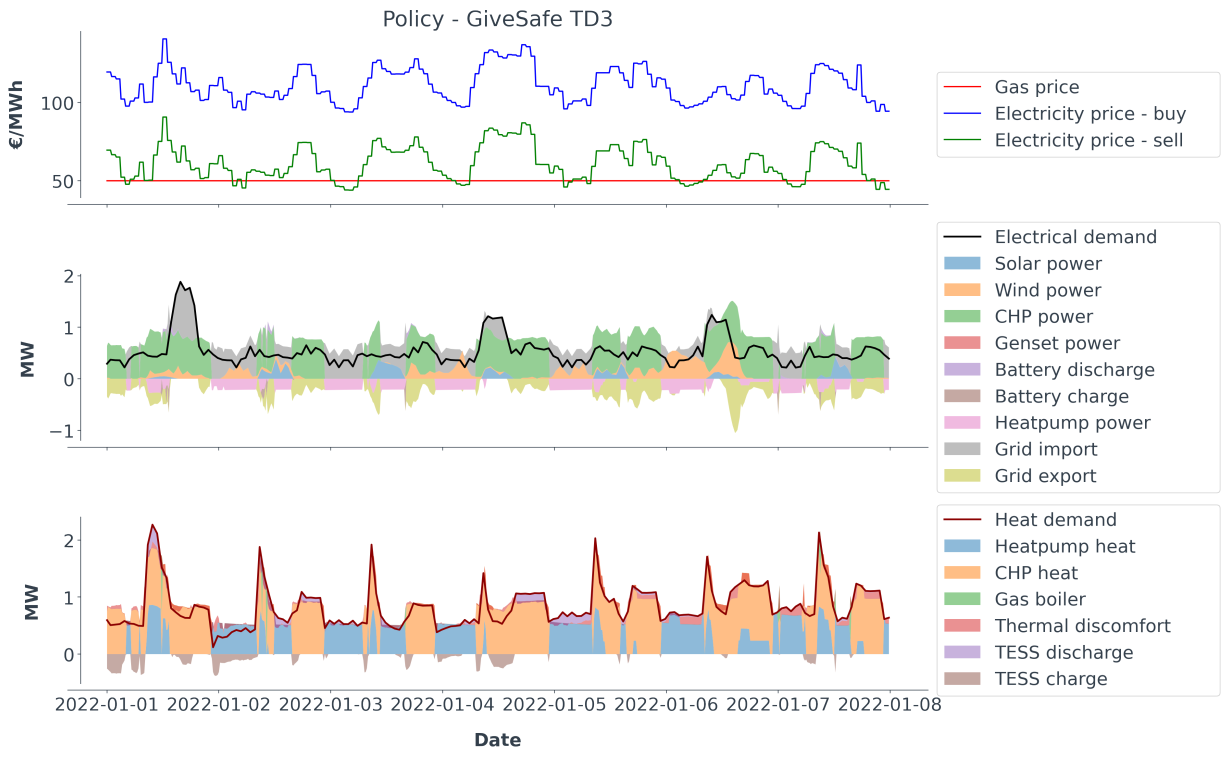


**Figure A6.** Policy visualization: GiveSafe TD3 (after safe training)

## Appendix B  Pseudo-code of TD3

**Algorithm 3:** Twin Delayed DDPG (TD3) [29]

```
1  Input: initial policy parameters $\theta$, Q-function parameters $\phi_1$, $\phi_2$,
   empty replay buffer $\mathcal{D}$
2  Set target parameters equal to main parameters $\theta_{targ} \leftarrow \theta$, $\theta_{targ,1} \leftarrow \theta_1$, $\theta_{targ,2} \leftarrow \theta_2$
3  repeat
4    Observe state $s$ and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
5    Execute $a$ in the environment
6    Observe next state $s'$, reward $r$ and done signal $d$ to indicate whether $s'$ is terminal
7    Store $(s, a, r, s', d)$ in replay buffer $\mathcal{D}$
8    If $s'$ is terminal, reset environment state
9    if it's time to update then
10     for $j$ in range(however many updates) do
11       Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
12       Compute target actions
           $$a'(s') = \text{clip}\big(\mu_{\theta_{targ}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\big), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$
13       Compute targets
           $$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{targ,i}}(s', a'(s'))$$
14       Update Q-function by one step of gradient descent using
           $$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s,a) - y(r,s',d))^2 \qquad \text{for } i = 1, 2$$
15       if $j$ mod policy_delay $= 0$ then
16         Update policy by one step of gradient ascent using
             $$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$
17         Update target networks with
             $$\phi_{targ,i} \leftarrow \rho\phi_{targ,i} + (1-\rho)\phi_i \qquad \text{for } i = 1, 2$$
             $$\theta_{targ} \leftarrow \rho\theta_{targ} + (1-\rho)\theta$$
18       end if
19     end for
20   end if
21 until convergence
```

## Appendix C Run-time statistics

The simulations are conducted on a local machine with a Intel® Core™ i5-8365U CPU @1.6GHz, 16 GB of Ram and an SSD. Over a yearly simulation, the following run-time statistics per simulated time-step (with a control horizon of 15 min) are observed.

| Optimal controller | min | mean | std | max | total |
|---|---|---|---|---|---|
| Unsafe TD3 | 0.027 s | 0.041 s | 0.006 s | 0.100 s | 1424 s |
| Unsafe Random | 0.027 s | 0.040 s | 0.011 s | 0.120 s | 1394 s |
| SafeFallback TD3 | 0.042 s | 0.058 s | 0.008 s | 0.250 s | 2047 s |
| SafeFallback Random | 0.037 s | 0.044 s | 0.005 s | 0.142 s | 1545 s |
| GiveSafe TD3 | 0.041 s | 0.060 s | 0.040 s | 2.661 s | 2090 s |
| GiveSafe Random | 0.041 s | 2.272 s | 3.760 s | 52.390 s | 79615 s |

**Table A1.** Run-time statistics

The maximum run-time per time-step never exceeds the control horizon of 15 minutes, as this otherwise would be considered impractical with the given hardware. We observe that the unsafe agents have the fastest run-time, as they don't have the constraint check to compute. Yet, we have argued that using unsafe agents is not realistic in safety-critical environments. Furthermore we observe that the SafeFallback method itself (demonstrated by using random agents) is significantly faster then the GiveSafe method, as the GiveSafe method can require multiple additional "offline" time-steps for every "online" (i.e. real) time-step. After the TD3 agents are trained though, the run-time is approximately the same - as the amount of unsafe actions proposed by the TD3 agent (and thus the need for additional "offline" training steps) is greatly reduced.

Notice that this are the run-time statistics *after* training (i.e., pure policy execution, in the case of the TD3 agents). Including the online training run-time statistics (i.e. fitting the function approximation algorithm - which is a multi-layer perceptron in our case), the mean run-time would result in 0.058 s for the unsafe TD3 agent, 0.076 s for the SafeFallback TD3 agent and 2.229 s for the GiveSafe TD3 agent. These online training run-time statistics are still magnitudes faster then the control horizon of 15 minutes.

## References

1. Fabrizio, E.; Filippi, M.; Virgone, J. Trade-off between environmental and economic objectives in the optimization of multi-energy systems. *Building Simulation 2009 2:1* **2009**, *2*, 29–40. doi:10.1007/S12273-009-9202-4.
2. Engell, S. Feedback control for optimal process operation. *Journal of Process Control* **2007**, *17*, 203–219. doi:10.1016/J.JPROCONT.2006.10.011.
3. Ceusters, G.; Rodríguez, R.C.; García, A.B.; Franke, R.; Deconinck, G.; Helsen, L.; Nowé, A.; Messagie, M.; Camargo, L.R. Model-predictive control and reinforcement learning in multi-energy system case studies. *Applied Energy* **2021**, *303*, 117634. doi:10.1016/j.apenergy.2021.117634.
4. Cao, D.; Hu, W.; Zhao, J.; Zhang, G.; Zhang, B.; Liu, Z.; Chen, Z.; Blaabjerg, F. Reinforcement Learning and Its Applications in Modern Power and Energy Systems: A Review. *Journal of Modern Power Systems and Clean Energy* **2020**, *8*, 1029–1042. doi:10.35833/MPCE.2020.000552.

5. Yang, T.; Zhao, L.; Li, W.; Zomaya, A.Y. Reinforcement learning in sustainable energy and electric systems: a survey. *Annual Reviews in Control* **2020**, *49*, 145–163. doi:10.1016/J.ARCONTROL.2020.03.001.
6. Perera, A.T.; Kamalaruban, P. Applications of reinforcement learning in energy systems. *Renewable and Sustainable Energy Reviews* **2021**, *137*, 110618. doi:10.1016/J.RSER.2020.110618.
7. Rayati, M.; Sheikhi, A.; Ranjbar, A.M. Applying reinforcement learning method to optimize an Energy Hub operation in the smart grid. *2015 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2015* **2015**. doi:10.1109/ISGT.2015.7131906.
8. Watkins, C.J.C.H. Learning from delayed rewards. PhD thesis, King's College, Cambridge United Kingdom, 1989.
9. Sheikhi, A.; Rayati, M.; Ranjbar, A.M. Demand side management for a residential customer in multi-energy systems. *Sustainable Cities and Society* **2016**, *22*, 63–77. doi:10.1016/j.scs.2016.01.010.
10. Mbuwir, B.V.; Kaffash, M.; Deconinck, G. Battery Scheduling in a Residential Multi-Carrier Energy System Using Reinforcement Learning. 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018. Institute of Electrical and Electronics Engineers Inc., 2018. doi:10.1109/SmartGridComm.2018.8587412.
11. Wang, X.; Chen, H.; Wu, J.; DIng, Y.; Lou, Q.; Liu, S. Bi-level Multi-agents Interactive Decision-making Model in Regional Integrated Energy System. 2019 3rd IEEE Conference on Energy Internet and Energy System Integration: Ubiquitous Energy Network Connecting Everything, EI2 2019. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 2103–2108. doi:10.1109/EI247390.2019.9061889.
12. Ahrarinouri, M.; Rastegar, M.; Seifi, A.R. Multi-Agent Reinforcement Learning for Energy Management in Residential Buildings. *IEEE Transactions on Industrial Informatics* **2020**, p. 1. doi:10.1109/tii.2020.2977104.
13. Ye, Y.; Ye, Y.; Qiu, D.; Wu, X.; Strbac, G.; Ward, J. Model-Free Real-Time Autonomous Control for a Residential Multi-Energy System Using Deep Reinforcement Learning. *IEEE Transactions on Smart Grid* **2020**, *11*, 3068–3082. doi:10.1109/TSG.2020.2976771.
14. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* **2015**.
15. Xu, Z.; Han, G.; Liu, L.; Martinez-Garcia, M.; Wang, Z. Multi-energy scheduling of an industrial integrated energy system by reinforcement learning-based differential evolution. *IEEE Transactions on Green Communications and Networking* **2021**, *5*, 1077–1090. doi:10.1109/TGCN.2021.3061789.
16. Zhu, D.; Yang, B.; Liu, Y.; Wang, Z.; Ma, K.; Guan, X. Energy Management Based on Multi-Agent Deep Reinforcement Learning for A Multi-Energy Industrial Park. *Applied Energy* **2022**, *311*, 118636. doi:10.1016/j.apenergy.2022.118636.
17. Pu, Y.; Wang, S.; Yang, R.; Yao, X.; Li, B. Decomposed Soft Actor-Critic Method for Cooperative Multi-Agent Reinforcement Learning. *arXiv* **2021**.
18. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arxiv* **2017**.
19. Fujimoto, S.; Van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning, ICML 2018* **2018**, *4*, 2587–2601.
20. García, J.; Fernández, F. A comprehensive survey on safe reinforcement learning, 2015.
21. Dulac-Arnold, G.; Levine, N.; Mankowitz, D.J.; Li, J.; Paduraru, C.; Gowal, S.; Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* **2021**, *110*, 2419–2468. doi:10.1007/S10994-021-05961-4/TABLES/11.
22. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. *Annual Review of Control, Robotics, and Autonomous Systems* **2021**, *5*. doi:10.1146/annurev-control-042920-020211.
23. Mattsson, S.E.; Elmqvist, H.; Otter, M. Physical system modeling with Modelica. Control Engineering Practice. Pergamon, 1998, Vol. 6, pp. 501–510. doi:10.1016/S0967-0661(98)00047-1.

24. Gräber, M.; Fritzsche, J.; Tegethoff, W. From system model to optimal control - A tool chain for the efficient solution of optimal control problems. Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017. Linköping University Electronic Press, 2017, Vol. 132, pp. 249–254. doi:10.3384/ecp17132249.
25. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arxiv* **2016**.
26. Lukianykhin, O.; Bogodorova, T. ModelicaGym: Applying reinforcement learning to Modelica models. *ACM International Conference Proceeding Series* **2019**, pp. 27–36. doi:10.1145/3365984.3365985.
27. Andersson, C.; Akesson, J.; Fuhrer, C. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface. Technical Report 2, Lund University, 2016.
28. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* **2021**, *22*, 1–8.
29. OpenAI. Twin Delayed DDPG — Spinning Up documentation, 2020.