Fachhochschule Kiel

Hochschule für Angewandte Wissenschaften

MS. Information Engineering(MIE), IuE

Date: 05 August 2024

Training Prosumer Agents with Reinforcement Learning for Energy and Cost Optimization

In cooperation with

Forschungsstelle für Energiewirtschaft e. V.

Submitted by

Sudesh Acharya
Mat. Nr. : 932253

Under the Supervison of

Prof. Dr. Hauke Schramm
Mr. Vincenz Regener(FfE e.V.)

Submitted as Master Thesiswork for the partial fulfillment of M.S. in Information Engineering, IuE, University of Applied Science, Kiel

Abstract

Energy management within private household has seen significant progress in recent years due to the advancement in technology, increasing integration of solar energy generation and adoption of Electric Vehicles(EVs) operating bidirectionally to exchange power. This creates opportunities for a household to participate in energy market by utilizing local and mobile energy assets. This research and development undertaking explores a case of energy management within household by utilizing Reinforcement Learning(RL) methods to learn strategies to minimize net energy exchange cost to the grid.

With the use of several RL algorithms and rule based heuristic, this work compares performance of different learned strategies and demonstrates that the agent learns to optimize the objective of cost reduction by a significant margin compared to rule based strategies.

Acknowledgment

I would like to express my sincere gratitude to **Prof. Dr. Hauke Schramm**, for the supervision, guidance, and support.

I gratefully acknowledge the invaluable time, supervision, constant encouragement of **Mr. Vincenz Regener** and **Mr. Theodor Haug** throughout this undertaking.

Also, I greatly appreciate the opportunity from **Forschungsstelle für Energiewirtschaft e. V.** for letting me in on this exciting project.

Last, but not least, i would like to thank **University of Applied Science, Kiel** for great learning experience through this compulsory Thesiswork.

Contents

Introduction

1.1 Background

In recent years, due to increasing adoption of renewable energy sources and awareness of it's impact on mitigation of issues caused by climate change, industries and households are transforming towards climate neutrality. United efforts of countries around the world such as Paris Climate Agreement[1](PCA), signed by 195 countries, to deal with climate change issues, define collective and national goals in short and long terms. Main goals of the PCA agreement include reduction of average global temperature below $2°C$, transparency on progress, awareness programs, and regular assessment of the united goals. Climate Neutral Act [2] defines goal to achieve green house gas neutrality by 2050 for Germany, enforcing threshold on permissible emission of CO_2 on sectors such as energy, transport, and other industries. The energy industry is progressively transitioning from conventional to renewable energy sources in Germany as indicated by the positive trend in figure 1.1

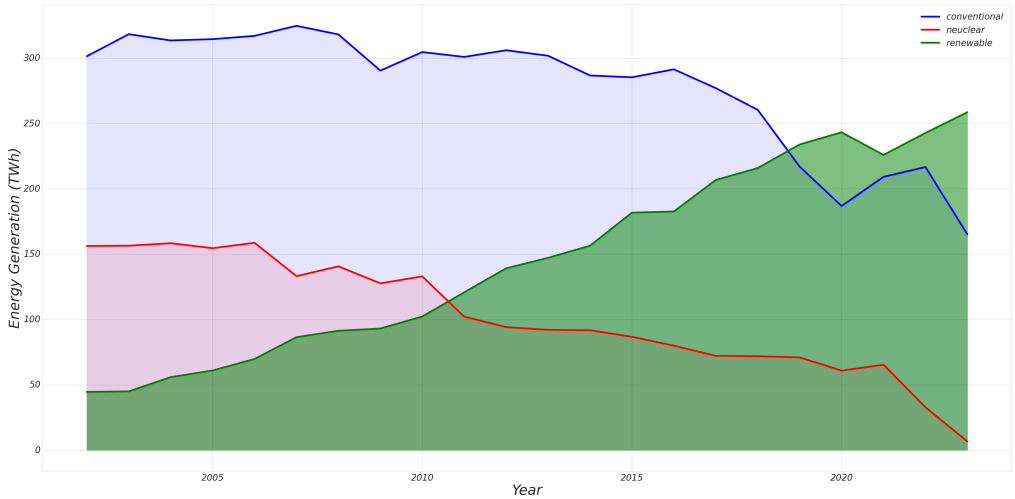

Figure 1.1: *Energy generation in Germany per year [2002-2023]* src:[3]

This trend is reflected in increasing adoption and integration of renewable sources of energy within households. Traditionally household demand were met solely from the grid, which depended mostly on the generation of energy from conventional

sources. With the integration Photo Voltaics(PV), households are not only able to meet partial demand by relying on self-generated energy, but are also able to participate in market by feeding excess back to the grid. These household with bidirectional power exchange to the grid are termed as Prosumers(consumer-producer).

The concept of Prosumer household introduced shift in some responsibility of energy management from centralized distribution hubs to the end user, resulting in Energy Management System(EMS) within Household. The term EMS was first pointed out during the rise of personal computing platform[4]. Home Energy Management System(HEMS) includes sensing, monitoring, communication and control of load consumption, generated PV, and Energy Storage Systems(ESS) in order to optimize user defined objectives such as minimization of net energy cost, reduced grid dependency, comfort, and emission minimization.

Electric Vehicles(EVs), in recent years, has been increasingly integrated within HEMS. [5] Proposes HEMS integrated with PV, EV and ESS with excess grid feed back. EVs with bidirectional power flow in Electric Vehicles Supply Equipment(EVSE) allows the EVs to be used as a distributed energy storage asset and brings forth opportunities and application for energy exchange to household, participation in grid demand response and energy consumption optimization.

Conventionally HEMS were used in the form of analytics and visualization, making the user aware of their consumption and thereby aiding conscious use of household appliances and perform manual scheduling. Rule based scheduling were adopted to include optimization based on pre-defined criteria and thresholds to partially automate control. With advancement in technology, optimization based techniques such as Linear programming(LP) and Mixed Integer Linear Programming(MILP) methods were used to find the best outcome in a scenario, such as optimization of energy consumption and production under constraints like battery capacity and grid limitations. [6] Utilizes MILP to reduce dissatisfaction and cost of consumers considering EES and Plug-In EVs(PEV). Recent approaches include Dynamic Programming(DP) for optimization objectives such as charging and scheduling strategies based on sequential decision-making. [7] Demonstrates a case study on optimization of rule-based strategies in Hybrid Elecric Vehicles(HEV) using DP. Model Predictive Control(MPC), use modeling, prediction of future dynamics, and suggest control strategies. Rule-based approaches often lack adaptability and may not capture complex dynamics. Optimization methods require accurate modeling of dynamics of the system and can be computationally expensive. While MPC approaches offer promising results, they often require large datasets and may not generalize well to unseen scenarios.

RL offers a unique advantage in directly learning optimal control strategies through trial and error, interaction with the environment, making it suitable for handling real-time decision making under uncertainty. [8] uses Q-Learning based scheduling and control of HEMS component with ESS charging and discharging to reduce consumption cost while maintaining user comfort. [9] Provides optimized control method using RL with distributed flexible assets within private households and provides a practitioner guide on implementing such a system.

1.2 Objective

Optimizing energy flow between the HEMS components while considering dynamic pricing, varying solar generation, distributed and local energy storage system remains a significant challenge. This undertaking focuses on an application domain of a household environment in Germany, integrated with Photo Voltaic(PV) energy generation and energy storage system, exchanging energy with the grid to meet household demand under variable exchange tarrifs(Eur/Kwh). A bidirectional power transfer to and from the energy storage as control action, with the objective of reducing the net exchange cost to a grid by learning useful control strategies using Deep Reinforcement Learning(DRL).

1.3 Research Questions

With the objective of learning useful strategies for reduction of net exchange energy cost within a household, several research question were explored. This includes the possibility of modeling of HEMS control system as a Prosumer RL Agent, impact of selection of algorithm on performance of the system and factors influencing the learned strategies by an Agent. Following list outlines the question more concretely.

- How can Prosumer Agents be modeled in a Reinforcement Learning environment?
- Can RL effectively minimize cost of energy under time-variable tarrifs ?
- How does the choice of RL algorithm impact performance of the control system ?
- What are the key factors influencing the optimal control strategies learned by the RL agent?

Within the scope of addressing these questions and optimizing the objective, remainder of the document is organized as follows. Section 2 describes methodology and underlying theoretical framework based on defined objectives. Section 3 describes different components involved in development of experimentation framework to model, test and collect observations. Section 4 explores the resulting observation, their interpretation and relevance on answering the research questions. Final Section 4 compares and contrasts on the objective and the achieved outcomes.

Methodology

This section describes the components and methodologies used to achieve the objectives defined in previous section. Since the chosen methodology is based on RL, which is a data-driven method that learns control action directly from the datasets using Deep Neural Networks(DNNs). It begins with descriptive analytics of dataset in use. The subsection following describes foundations and concepts of theoretical framework in use. The RL based framework requires trial and error learning, therefore there is need of implementation of custom environment simulating HEMS. The simulated environment and its different types, based on their involvement in steps of modeling are discussed. A reward function definition/design is crucial part of the setup to learn policy as the reward reinforcement dictates the behavior(action under given policy) of an agent. Finally appropriate algorithms are selected for the learning of the policy itself.

2.1 Data Description

Since the modeling of policy network or deep neural networks in general, is based direct mapping of data to resulting unit(action), it is important to select input observation/features to be representative of information needed by agent to make decisions. The dataset [10] provided is an open dataset on residential bidirectional EV charging. It includes roughly a year of time series dataset with 15 minutes resolution, collected in private customer household in southern germany. The dataset includes power monitoring at Grid Connected Point(GCP) with positive values, generated PV power with negative values and power through bidirectional Electric Vehicle Supply Equipment(EVSE) as positive values. Power drawn by a household (P_{HH}) is derived as $P_{HH} = P_{GCP} - P_{PV} - P_{EVSE}$. It further includes information on EV such as boolean connection status, current State of Charge(SoC) of the EV and target SoC(SoC level to attain with user pre-defined departure time). Although the dataset is useful for many scenarios such as EV power exchange to Household, Grid, other Energy consumption devices, for the use case at hand, only the information on Power consumed from grid to meet household demand (P_{HH}), power of generated PV(P_{PV}) is utilized. The unit of Power is measured in Watts. Additional time series dataset[3] on day ahead variable energy exchange price of hourly resolution is utilized.

Further description of data is projected as distribution of samples for each se-

lected features as shown in figure 2.1. The distribution of sample values in Power PV data is shown in the upper left section of the figure. As discussed in the data collection process[10], the data values of (P_{PV}) is always below 0. Around 64% of the sample values lie within $(-0.5, 0)$ with average value of -1.789. Similarly, in the upper right section of the figure represents distribution of values for Power Household with 71% of value within $(0, 0.5)$. These (P_{HH}) value have mean of 0.542 and always lie above 0. Auction price data is a near normal distribution with mean value of (0.192). These statistics could provide some insights on impact of set of these observation input values on resulting action of the Agent.

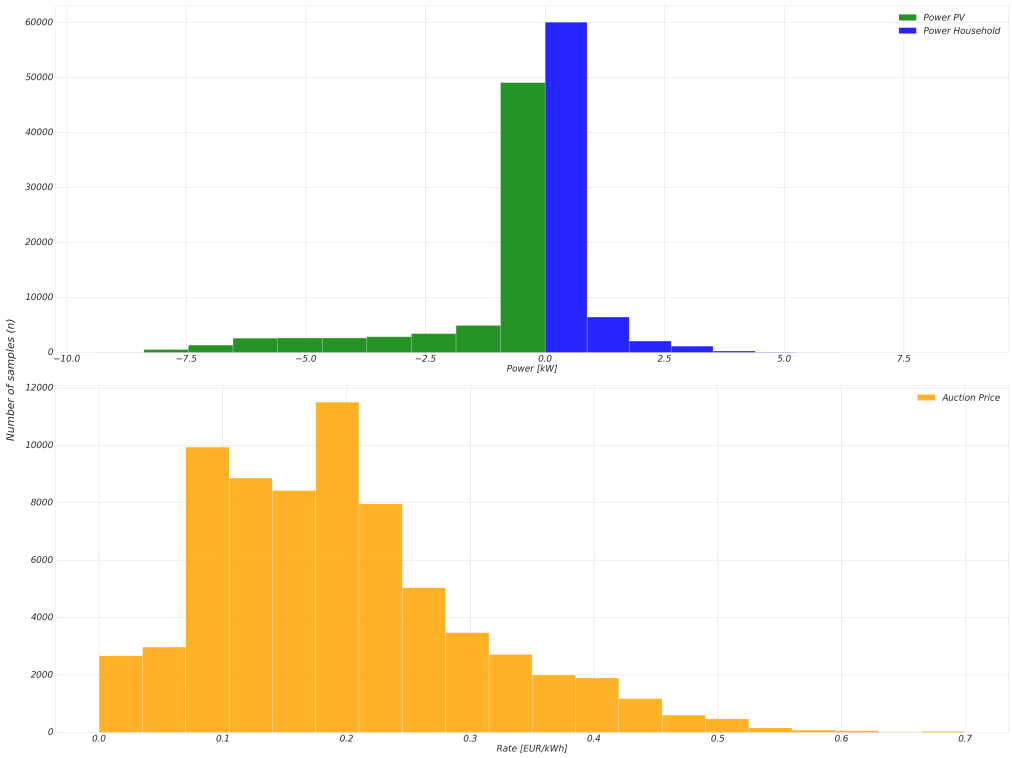

Figure 2.1: *Data distribution of observation inputs*

While distribution may provide information on individual samples, it fails to represent the temporal relationship among them. Upon additive decomposing, the data series can be visualized in terms of component such as Trend(T), Seasonality(S) and Residuals(i.e. Time Series $= T + S + R$). The Repetition of patterns of behaviors in household consumption and PV generation, reflected on different scale of time range are termed as seasonality. The trend component can be interpreted as margin of consecutive values over period of time or patterns without seasonal component. The residual part of the series, as the name suggests, is rest of the composition after reducing patterns of trend and seasonality.

Figures 2.2 and 2.3 show decomposition of input observation series for a day in a week and hour in a day respectively. There are several scales of patterns that

can be identified through the full range of dataset length such as season or months of the year with variation in PV generation, price of energy. Yearly and Monthly decomposition is limited by size of dataset. Selection of time frame window as day in a week allows us to see and compare daily pattern in consumption shown in blue graph in upper left section. It starts with Friday with heavy consumption pattern during morning and evening hours, higher during weekends and average for the rest of the weekdays. A trend in increase in auction price can be seen during weekdays and reduced in weekend likely reflection of less demand from industries.

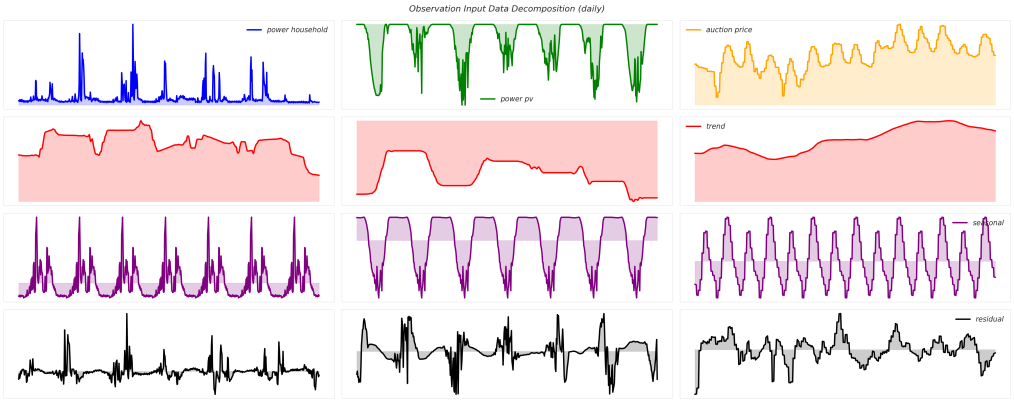

Figure 2.2: *Additive decomposition of observation inputs (daily)*

Since the PV patterns are independent of weekdays, an hourly decomposition in a day was selected as shown by the green curve. The PV peaks during mid-day when the sun is up, household demand is less during work hours.

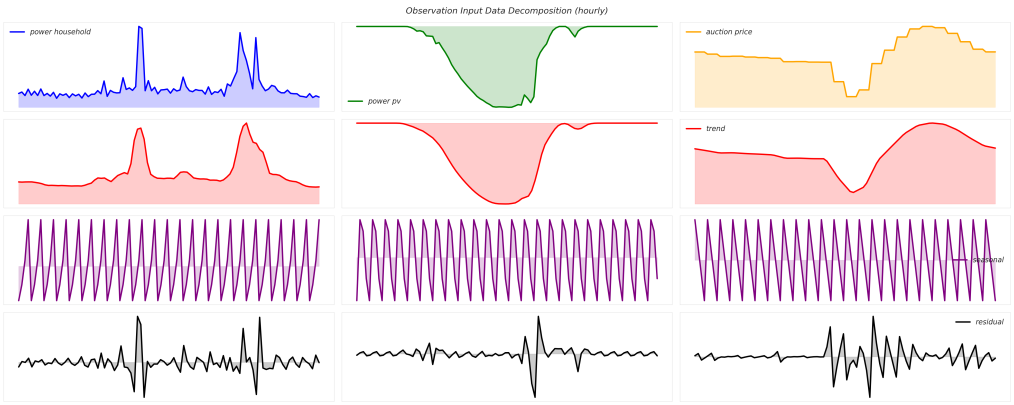

Figure 2.3: *Additive decomposition of observation inputs (hourly)*

The description of different properties of dataset such as sample distributions, seasonal and trend components decomposition could allow interpretation of the pattern in agents behavior later during inference, since these input series solely are used to learn policies. The residual part is distinguished from other component and

may represent important information whereas seasonality and trend might introduce fluctuations.

2.2 Theoretical Framework

Main concept of reinforcement learning (RL) revolves around an agent learning to make decisions in an environment through trial and error, with the goal of maximizing a reward. A typical agent environment interaction[11] is shown in the figure 2.4. The main components and terminologies of such interactions are described as follows.

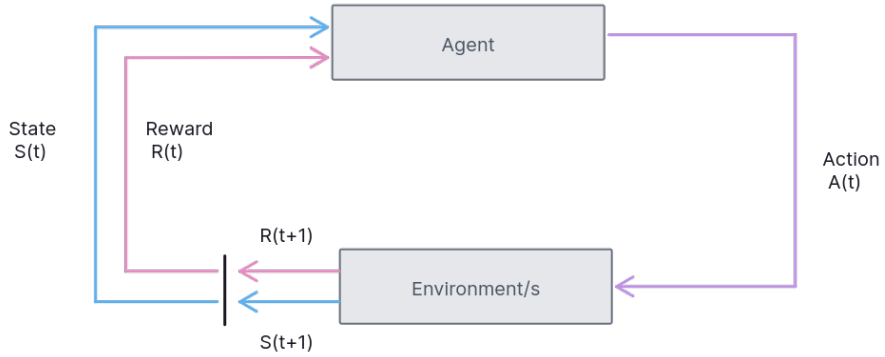

Figure 2.4: *Agent environment interaction*

- **Environment**: It is the world or system the agent operates in. It provides the agent with feedback in the form of rewards or penalties based on its actions.
- **State**: It represents the current situation in the environment relevant to the agent's decision-making. Special states are initial and termination states
- **Reward**: It is a numerical value assigned to the agent after it takes an action. Positive rewards indicate good choices, while negative rewards indicate bad choices. The reward function guides the agent's learning by shaping its understanding of what actions lead to intended behavior.
- **Policy**: It is the strategy the agent uses to map states to actions. Through trial and error, the agent learns and refines its policy to make better decisions in the future. The cycle of action, reward, and policy update continues as the agent learns through experience until some termination or truncation criteria is met.
- **Action**: It is what the agent chooses to do in a particular state under current policy/rule.

A RL system in context of HEMS with finite set of observation states is based on a decision process termed as Markov Decision Process(MDP) and is defined in terms

of tuple (S, A, P, R, γ). where, S is set of all possible permutation of observation values, A is set of all possible action taken by decision making agent, P is state transition probability matrix $P^a_{ss'}$ as state transition to future state S_{t+1}, given current state S_t for current action, i.e. $P_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s]$, R is estimated return $R^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$ given state and action pair, $\gamma \in [0, 1]$ is a discount factor that defines the relevance of future returns on computing the current reward i.e. if $\gamma = 1$, no discounting and if $\gamma = 0$, only considering the next consecutive reward R_{t+1}.

2.3 Custom HEMS Environment

In order for a learning system to work on different set of input observation and resulting action, a simulated household environment is to be implemented, which is safer and efficient than a real world scenario.

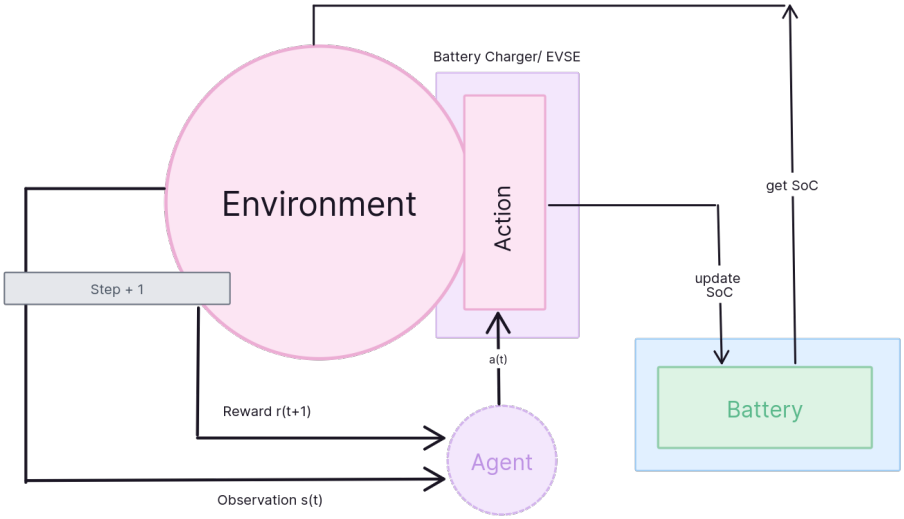

Figure 2.5: *Custom simulated HEMS environment*

A customized HEMS environment is setup using [17] as shown in the figure 2.5. It consists of an integration of a energy storage(EV or Home battery), with current state of it's content, used as part of input observation and updated with the subsequent actions depending on whether the action is to charge and discharge from the battery respectively. The action taken by an agent is modeling of charging and discharging power P (kW) of bi-directional battery charger/EVSE. A small update in battery energy content ΔSoC is formulated based on pre-defined range of action space and battery capacity as given by equation 1.

$$\boxed{\Delta\text{SoC} = \frac{\eta \times P \times dt}{C}} \tag{1}$$

where, η is Columbic efficiency accounting energy losses during charging or discharging, C is battery capacity (kWh), and dt is a small change in time (hours) for soc update. The energy content is therefore updated based on power flow direction $\pm P$ as shown in equation 2

$$\boxed{\text{Updated SoC} = \text{Current SoC} + \Delta\text{SoC}} \tag{2}$$

Based on the functionality, a custom HEMS environment is initialized with different set of parameters to train, evaluate and test given policy as shown in figure 2.6. The training environment is initiated with part of train set input observation and wrapper to facilitate vectorization depending on number of environments in use. A separate evaluation environment is to be initiated to evaluate current policy after i number of training iteration and parameter updates in policy network. The evaluation environment is wrapped for monitoring to log information during policy evaluation-improvement iteration. Finally, to assess the policy on unseen observation, a test environment is initialized with test set of observation and respective parameters.

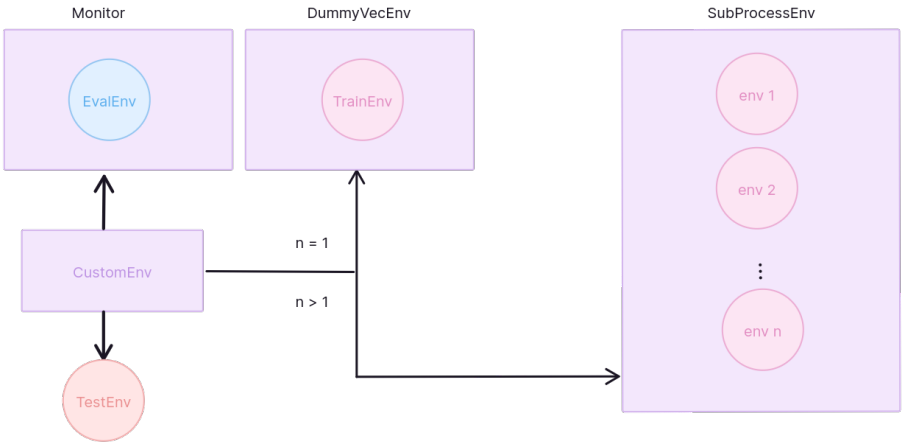

Figure 2.6: *Types of environments*

2.4 Reward Function Definition

Definition and design of reward function is at the core of modeling agents behavior. For each step in an environment, agent is encouraged for desirable action and vice-versa. For the use case at hand, the desirable actions are defined in terms of minimizing net household exchange cost to the grid, i.e. optimizing household total energy cost. An additional objective of retaining fraction of energy content could

be introduced, in case of emergency Household or EV usage in combination with minimizing energy cost. This combined formulation of rewards adds a layer of complexity but optimizes multiple objectives.

The reward function used to optimize net exchange cost of a household, computed as primary objective, is given by the equation 3. It serves as primary objective of this undertaking

$$\boxed{\textbf{cost reward} = (\text{power household} + \text{power pv} + \text{action}) \times \text{auction price}} \tag{3}$$

Combination of the primary reward with soc retain as secondary objective is defined in equation 4 to explore optimization of multiple/combined objectives.

$$\boxed{\textbf{combined reward} = \text{net exchange cost} \pm \text{soc retain reward}} \tag{4}$$

Where, soc retain reward $= +0.25$ for battery SoC to stay above defined threshold of 0.45 and -0.25 otherwise. The number 0.25 is chosen as a better value among values after several trial and improvement within range of $(0.1, 1)$. Since finding the right value in combined reward is a trade-off between constituent rewards, before this value there were no visible improvement in soc baseline. This trade-off is discussed in the supplementary observation section 4.

2.5 Algorithm Selection

Selecting an algorithm appropriate for modeling of the policy based on the custom environment, considering type of action and observation spaces, availability of information on dynamics of the environment and types of learning is key step to modeling Prosumer Agent. Figure 2.7 shows classification of algorithms based on these criteria.

As mentioned in custom environment section above, the action to be performed by an agent is the bidirectional charging and discharging power of a battery storage system on a continuous series of observation, an algorithm with continuous action space is selected. Based on policy learning type on whether to use the sample collection and simultaneous learning(i.e. on-policy) or to used sample collection and learn from them separately (i.e. off-policy), algorithms of both type were considered.

Based on state spaces exposed to the agent, the observable space is only partial. Since the full dynamics of environment are not known beforehand, model-free algorithms are considered for their simplicity and effectiveness. Considering above described criteria, policy gradient based algorithm Proximal Policy Optimization(PPO) [13], Soft Actor Critic(SAC)[14], and Twin Delayed Deep Deterministic

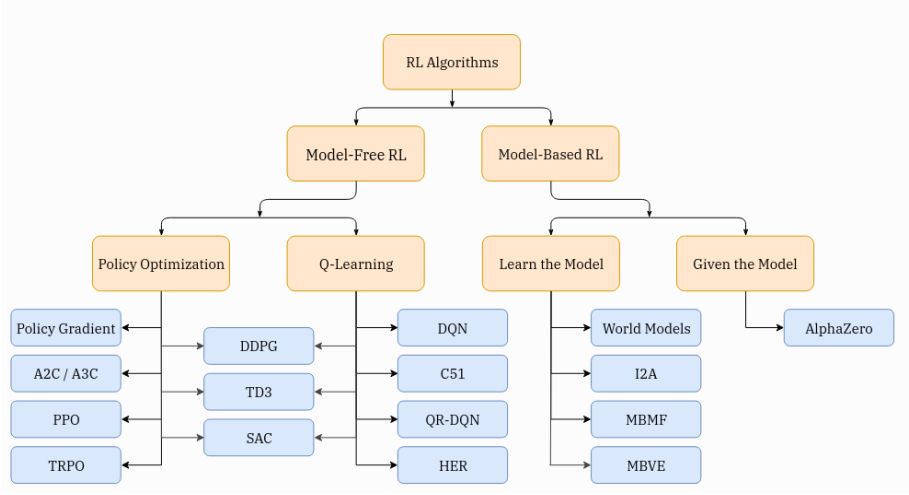

Figure 2.7: *RL algorithm classification* src: [12]

Policy Gradient(TD3)[15] were selected from [16] for the experiments.

A partial but crucial component of the algorithm is an actor network that predicts action given the set of observation. The actor network is part of in each all the selected algorithms. A typical architecture of an actor network is shown in figure 2.8. Given observation inputs of Power household, Power PV, Energy exchange price and current battery SoC, action is the resultant of an Fully connected Multi Layer Perceptron(FC-MLP), with appropriate activation Z(hyperbolic tangent due to it's compatibility with action space $A \in [-1, 1]$).

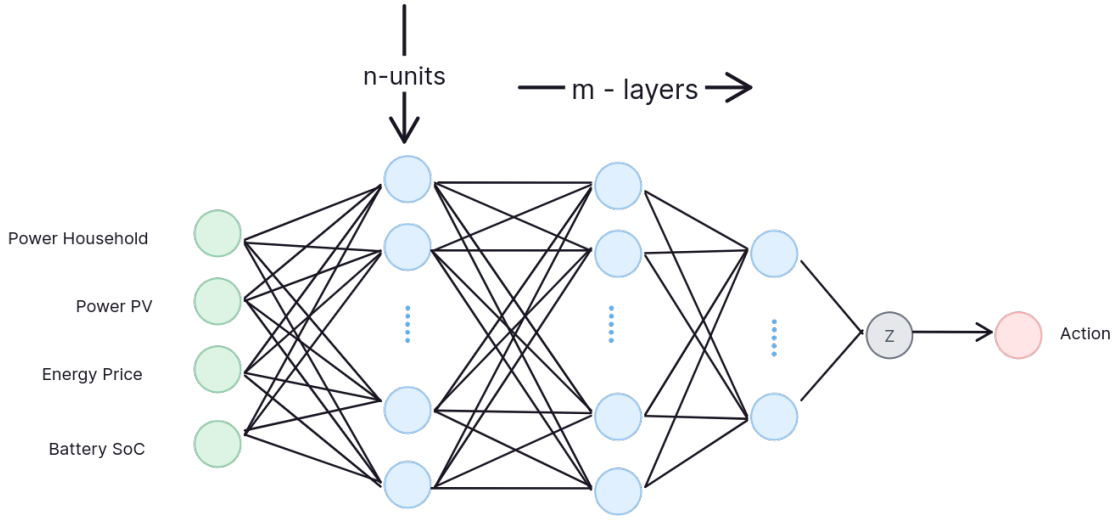

Figure 2.8: *Actor (pi) network*

Modeling a Prosumer Agent for selected algorithms effectively for the application under consideration, evaluating performance and identifying different factors

influencing the learning of better strategies is the main focus of this undertaking. The briefed methodologies above guide the development of experimental setup in the following section 3.

To bring the components together, steps of procedure is to be outlined for development. Following the definition of custom environment and their types, selected algorithms, relevant system states are defined. An action boundary, constraining agents action to operational range of a battery system is to be setup. A mapping of system states to resulting action is in accordance with the selected algorithms for learning policies. Starting from randomly initialized policy and criteria defining reward estimation, an appropriate behavior is modeled via the interaction of environment and policy learning algorithms. An agent is evaluated based on actions that optimizes the household net cost of exchange to a grid under variable tarrifs and improved based on the reward feedback. This iterative policy learning, evaluation and improvement process is expected to derive policy toward optimality.

Development

In order to experiment and collect observations on agents experience, implementation of different components defined in previous section, preparation of observation inputs, training of agents, iterative evaluation and improvement, taking feedback from different stages and re-training under set of parameter configuration, a pipeline is outlined based on flow of information as shown in figure 3.1. Following subsections describe each component of the pipeline in detail.

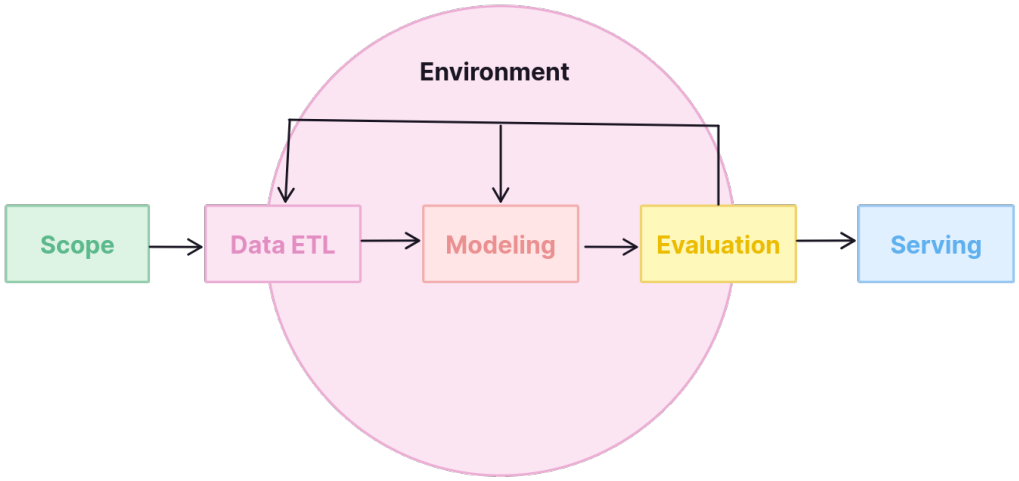

Figure 3.1: *Development pipeline*

Scoping defines development criteria for environment components, initialization parameters, metrics to track for the model performance and define a comparable baseline based on rule-based conditionals. Data processing step involves extraction, transformation, and merging dataset to prepare observation inputs for the environment. Train-evaluation involves steps required for modeling of a policy and saving intermediate/final results for inference. Testing involves sampling of action from the saved policy. Final step involves export of saved policy and sets up a prototypical scenario for a real world HEMS control.

3.1 Scoping

Based on the objectives identified and research question discussed in introduction section, the scope for environment development, selection of features, modeling of

policies and sample testing from a modeled policy is determined. The custom environment setup requires definition of observation and action space, step function for step transition in the environment, definition of reward objective, and resetting methods for every time environment step is initiated(i.e. resetting states to initial state). Following list outlines scoping of components

- **Observation space**: range of each observation input features is scoped within the maximum and minimum values of the series, defining upper and lower bounds respectively.
- **Action space**: similar to observation space, action space is defined, i.e. $a \in (-11, 11) \forall a \in A$. During implementation the range is set to $(-1, 1)$ and is later scaled during step transition in an environment to reduce computational complexity.
- **Optimization objective**: Main objective is to minimize cost using a power balance equation by computing the net exchange to the grid and applying exchange rate prices 3. Additional objectives are battery care or retain of soc by rewarding desired threshold and discouraging when out of range as defined in equation 4 of methodology section.
- **Metrics to track**: evaluation of performance of a model based on reward objectives during training are defined in terms of accumulation of mean episodic rewards representing the net exchange cost.
- **Rule based Baseline**: in order to measure the relative performance of a given policy, a rule based policy is set as a baseline. Although the actions in rule based scenario are discrete and pre-defined in contrast to action predicting learned policies, it provides a rough cost comparison among policies. The detail pseudo code is described in the following section.
- **RL based Algorithms**:

Rule-based Strategy

```
for length 'i' of test observation window:
  exchange = power_pv[i] + power_household[i]
  if exchange < 0:  # excess pv
    if battery.current_soc == 1:
      net_exchange = exchange + action # feed into grid
    else:
      action = max_charge_rate  # charge
  else:
    if battery.current_soc == 0:
      net_exchange = exchange + action # draw from grid
        else:
          action = max_discharge_rate # discharge
```

RL based Algorithms

PPO is an on-policy algorithm with improvement on Trust Region Policy Optimization(TRPO) [18] and mainly deals with large update in policy, so that it does not collapse. TRPO uses KL divergence to define this constraint. PPO uses clipping based objective to achieve this constraint. Figure 3.2 shows the steps involved in learning of the policy network. Initialized with random parameters θ of for policy network and value network parameters ϕ, it collects agents experience under policy π and uses Stochastic Gradient Accent(SGA) to maximize clip based objective and Stochastic Gradient Descent(SGD) to minimize the value error.

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

Figure 3.2: *Pseudo code for PPO*

TD3 is off-policy algorithm based on improvement of shortcomings in Deep Deterministic Policy Gradient(DDPG)[19], of overestimating Q values leading to policy degradation. TD3 introduces identical Q-value estimators as targets with delay and policy smoothing techniques. The learning of a policy network is done by computing target Q functions and updating the parameters of using gradient ascent and using the least of these two values mitigating overestimation as shown in figure 3.3

SAC is also an off-policy algorithm which uses double Q-functions as targets and policy smoothing techniques similar to TD3. It introduces entropy terms in the target networks for exploration and alternatively updates parameter using gradient descent for Q-function and gradient ascent for policy network parameters. A detailed pseudo-code for SAC is shown in figure 3.4

Algorithm 1 Twin Delayed DDPG

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  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 target actions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

      Update Q-functions by one step of gradient descent using

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

      if j mod `policy_delay` $= 0$ **then**
        Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

        Update target networks with

$$\begin{aligned} \phi_{\text{targ},i} &\leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2 \\ \theta_{\text{targ}} &\leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta \end{aligned}$$

      end if
    end for
  end if
until convergence

Figure 3.3: *Pseudo code for TD3*

Algorithm 1 Soft Actor-Critic

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:

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

      Update Q-functions by one step of gradient descent using

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

      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.
      Update target networks with

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

    end for
  end if
until convergence

Figure 3.4: *Pseudo code for SAC*

3.2 Data Processing: Extract, Transform and Load(ETL)

The dataset from two different Prosumers(alias p1 and p2), containing data within time frame($2021 - 2022$) were used. This dataset then go through series of transformation before it is initialized within an environment as observation inputs. These transformations include combination of household and auction price dataset by concatenation, aligning and transforming sample timesteps and frequency, unit conversion and handling of missing or outlying values as shown in figure 3.5. The resulting dataset is of length 70082 including p1 and p2 each containing 35041 entries with time index range of *2021-08-15 00:00* to *2022-08-15 00:00* UTC.

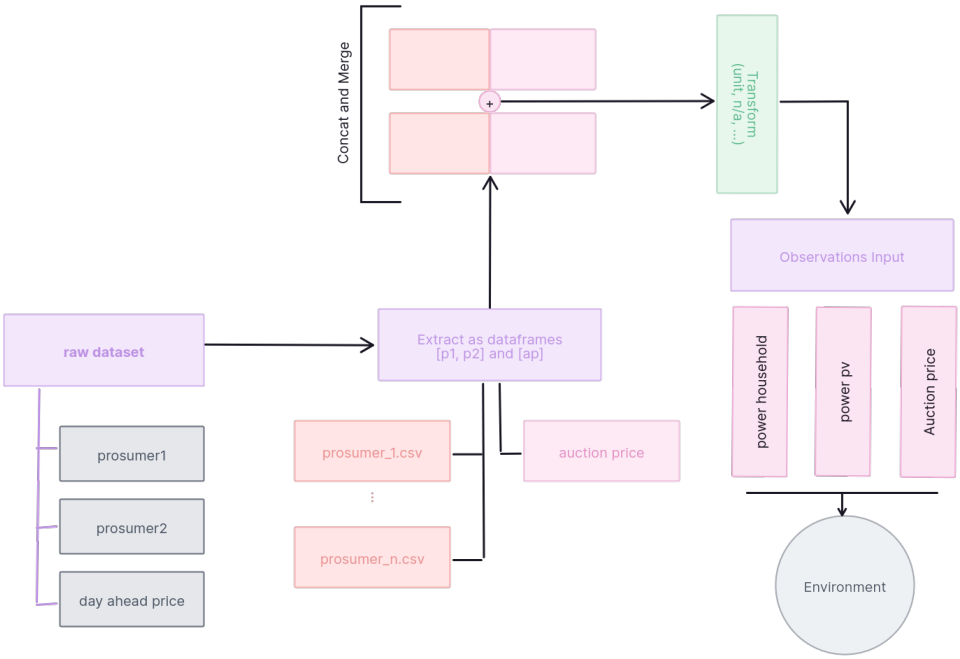

Figure 3.5: *Data Pre-processing*

3.3 Modeling: Policy Train-Evaluation Iteration

This step includes implementation of setup for learning parameters of a policy network, evaluation of performance and freezing of parameter saved as checkpoint for every step of policy evaluation-improvement iteration as shown with simplification in figure 3.6. Modeling in this context refers to the optimization of policy network parameters toward the ones resulting in near intended actions.

Training part of the iteration begins with a policy network with randomly initialized internal parameters. These parameters are updated for i number of iteration in a train environment and are evaluated in an evaluation environment. An evaluation step involves j number of episodes and taking an average to calculate the score. After evaluation the parameters are saved as checkpoint. The goal is to maximize

this score for each passing iteration using gradient assent to update parameters. The rate of update is defined by learning rate. The iteration of training continues untill the end of specified number of train cycle with continuous evaluation score returns. Retraining involves same steps except for starting from a randomly initialized policy, it must start with checkpoint of a policy that was previously trained. The rest of the step are identical, and one could either train or retrain at a time.

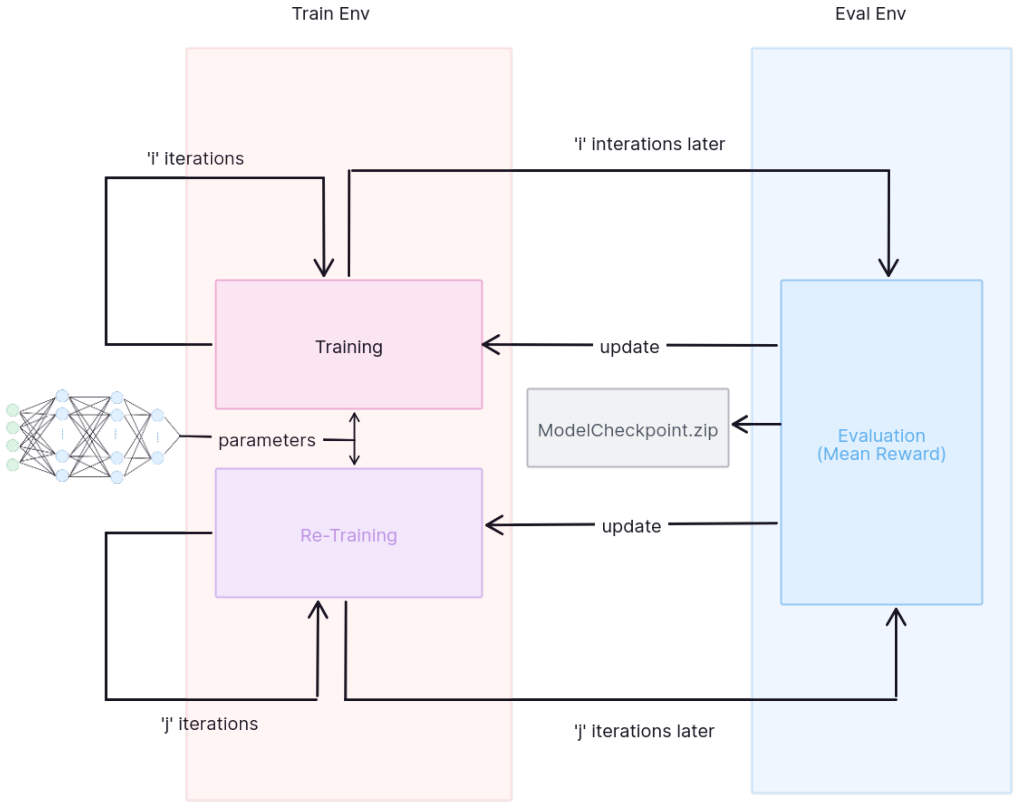

Figure 3.6: *Train-evaluation iteration*

3.4 Policy Testing

This steps is completely separate from training and evaluation as it uses input observation that are not used during training in a dedicated test environment. The policy network is loaded with the updated and saved parameters form the training-evaluation iteration. Test input observation are then passed through the actor network resulting in predicted action. This action is used to compute net exchange, update battery SoC and finally to calculate total cost of net exchange per episode as described in the section 2.

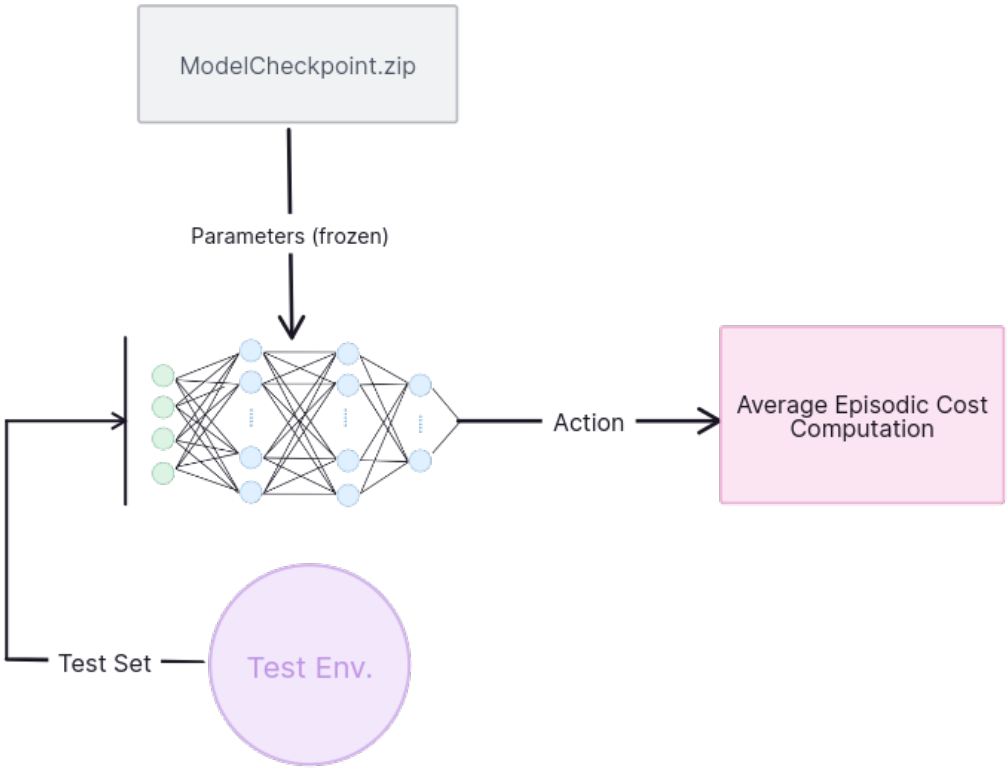

Figure 3.7: *Testing of Trained Policy*

3.5 Model Export

This step involves export of trained model to be used as a controller outside of the simulated HEMS environment. The models exported checkpoints saved as a compressed .zip format from modeling, are converted to pytorch format by extracting the Actor network, since it is responsible for action prediction, the critic network is discarded. The saved pytorch format checkpoint are further converted to open onnx format, which can be used for serving the model using various platforms. The model are then served through api endpoints to return the action given set of observations. The model server module, written in Flask, is containerized for further deployment.

The pipeline, following the methodology, starting from conceptual scoping to sample deployment, concludes the part of development. The checkpoints from the testing section above are used to sample actions for novel set of testing set for each algorithms and observations are visualized in the next section.

Observations

This section reports the observed results, during training, testing and derived net grid exchange cost from action sampled given trained policy. The dataset was split into training and test set with the ratio of 90% and 10% respectively. The custom environment was setup in such a way that it takes window of observation input of 2 weeks length and shuffles the train set uniformly randomly to get different set of input observation for next iteration. Discount factor for the model were kept 0.99 for all models

4.1 Parameters

There exists plethora of combination of hyper-parameters that can be searched and tested further to improve current policy. List 4.1 shows a set of model hyper-parameter settings under which the observations are collected during training and testing of the policy.

```
env_observation_window_train=int(24 * 4 * 7 * 2)
policy_nw="MlpPolicy"
num_train_eval_cycles=200
num_eval_episodes_per_cycle=2
num_test_episodes=5
train_timesteps=env_observation_window_train * 2
ppo_learning_rate=0.0003
ppo_clip_range=0.20
ppo_nws={"pi": [64, 64, 16], "vf": [64, 64, 16]}
sac_learning_rate=0.0003
sac_tau=0.005
sac_nws={"pi": [256, 256, 16], "qf": [256, 256, 16]}
td3_learning_rate=0.001
td3_tau=0.005
td3_nws={"pi": [128, 128, 16], "qf": [128, 128, 16]}
```

4.2 Observation during Train/Evaluation Cycle

Observation were recorded for optimization of perceived cost per episode for constant and variable auction price as described in the section below. A combined objective was also recorded with cost based reward and reward for retaining fraction of SoC in a battery. Observations for the combined objective are included as part of supplementary observation 4

With constant exchange rate

The performance of a policy during training is measured using the cost optimized by each algorithm. The networks were each trained for 200 cycle of training and for each training, 2 iteration of evaluation was performed and average was taken as a performance score for that iteration. The metrics recorded and visualized for optimization under constant tariff for grid feed and draw rate of 0.08 and 0.33 Eur/kWh respectively as shown in figure 4.1.

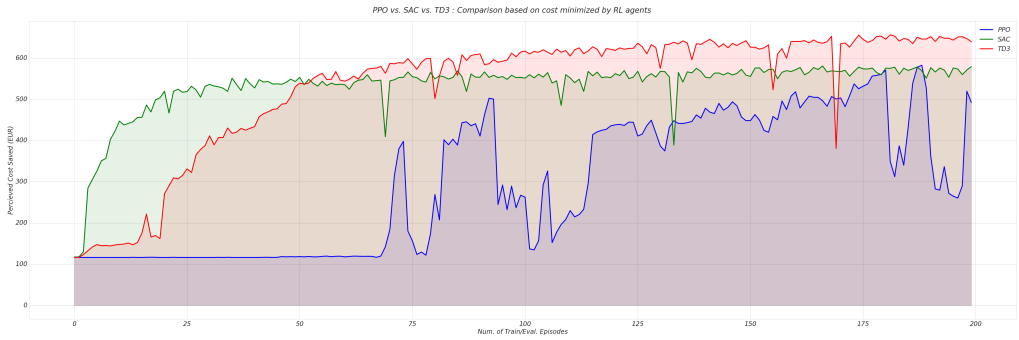

Figure 4.1: *Cost reward evaluation per episode with constant auction price*

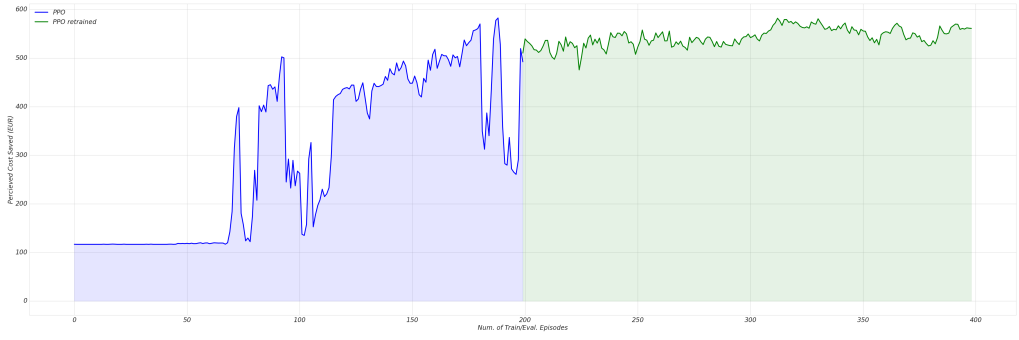

Figure 4.2: *Cost reward evaluation per episode with retrained PPO*

With perceived saved cost in relation to number of train-evaluation cycle, the comparison of cost is higher for TD3, SAC has more consistency over all with early positive trend. The PPO however is fluctuating and does not seem to plateau until

the end of 200^{th} cycle. Upon further retraining for 200 more cycles, it is improved, showing consistent value as shown in the figure 4.2. One of the reason for PPO to learn relatively slowly is due to the slow update of the network parameters based on sample collected under current policy, since it is on-policy algorithm. In contrast, TD3 and SAC which are off-policy therefore more sample efficient.

With variable exchange rate

The same cost based metrics were tracked for comparison with variable auction price dataset. Unlike under constant exchange rate price the feed in and draw from the grid is a same rate. The evaluated perceived cost saving per episode for variable auction price is shown in figure 4.3

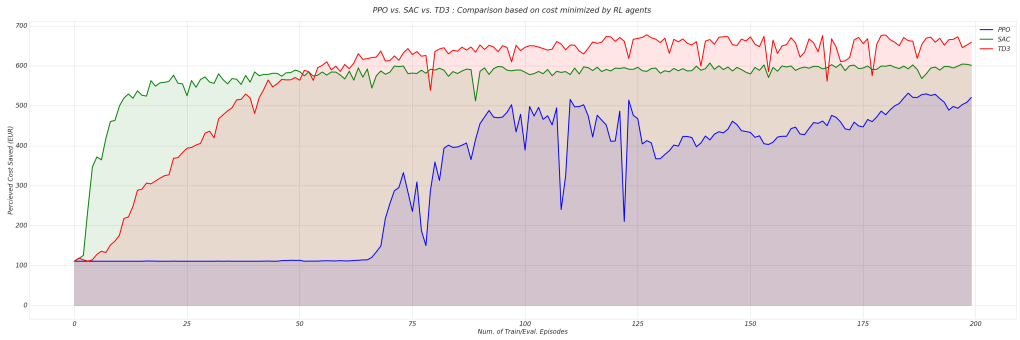

Figure 4.3: *Cost reward evaluation per episode with variable auction price*

The PPO is retrained also in the case of model with variable exchange cost and shows improvements after 200^{th} iteration. Although there seems to be a small trend at the end of the 200 cycle of retraining and evaluation, no further training was done. The retrained ppo with evaluation score is shown in figure4.4

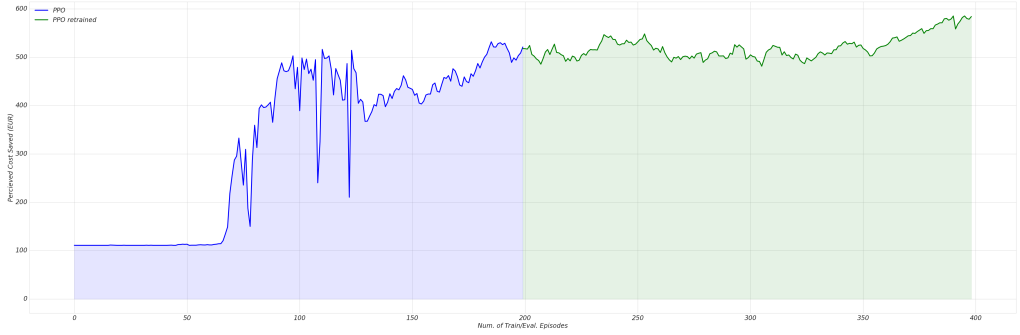

Figure 4.4: *Cost reward evaluation per episode with retrained PPO*

4.3 Observation during Testing

This section shows an agents experience on untrained dataset. In correspondence to the scenario of constant and variable auction price discussed in observation during training, the performance of a model was calculated for observation input window of A week(24*4*7 time steps). In addition, for the same window cost is computed for rule based strategy for comparison.

with Constant Exchange rate

The experience of an agent under constant auction price for rule based strategy can be seen in figure 4.5. The effect of variable PV generation directly corresponds to the action(i.e. charging during pv generation) and consequent change in battery SoC.

Figure 4.5: *Rule based policy with constant auction price*

RL based agent with TD3 under the same scenario of constant auction price is shown in figure 4.6. The reflection of variable PV generation corresponds to small change in action and the SoC however, the average cost saving in higher. Looking into the grid exchange component in the subplot, the agent has learned to save cost by selling all the energy back to the grid, as indicated by the negative grid exchange .

Comparison of cost based score for RL based and Rule based methods is shown is the table 4.1. TD3 model has learned to optimize the cost more effectively than the other RL based algorithm and slightly better than the rule based strategy.

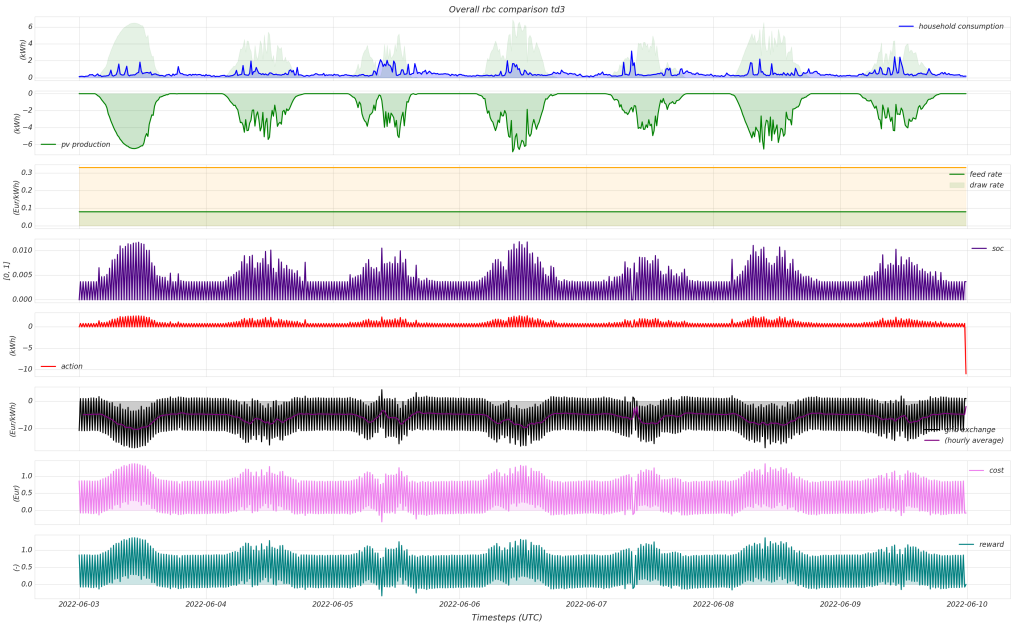

Figure 4.6: *TD3 with constant auction price*

Model Config	PPO	SAC	TD3	RBP
Untrained	54.405	54.417	54.405	0.0
Constant AP	242.696	255.239	**314.556**	312.382
Constant AP with PPO retrained	272.971	-	-	-

Table 4.1: Comparison per policy with constant exchange rate

with Variable Exchange rate

With variable auction price as part of observation input, rule based strategy has similar results as shown in figure 4.7. Since the exchange rate has changed and there is no complex modeling of variability, the change is apparent only in the net cost of exchange computation relative to rbc with constant price.

As shown in the figure 4.8, SAC has learned to optimize it behavior based on variable auction price as well with the spike in charging action, when the energy cost is 0 and discharges immediately when the exchange price is higher. In rest of the cases it chooses to sell all the energy for cost profit, since there is no constraint, neither incentive on keeping the energy.

Table 4.2 shows the comparison score with significant improvement of RL based algorithms, specially with SAC, taking advantage of variable exchange price, selling all possible energy to maximize the profit. Rule based strategies however has slightly lower score than with the case of constant auction price.

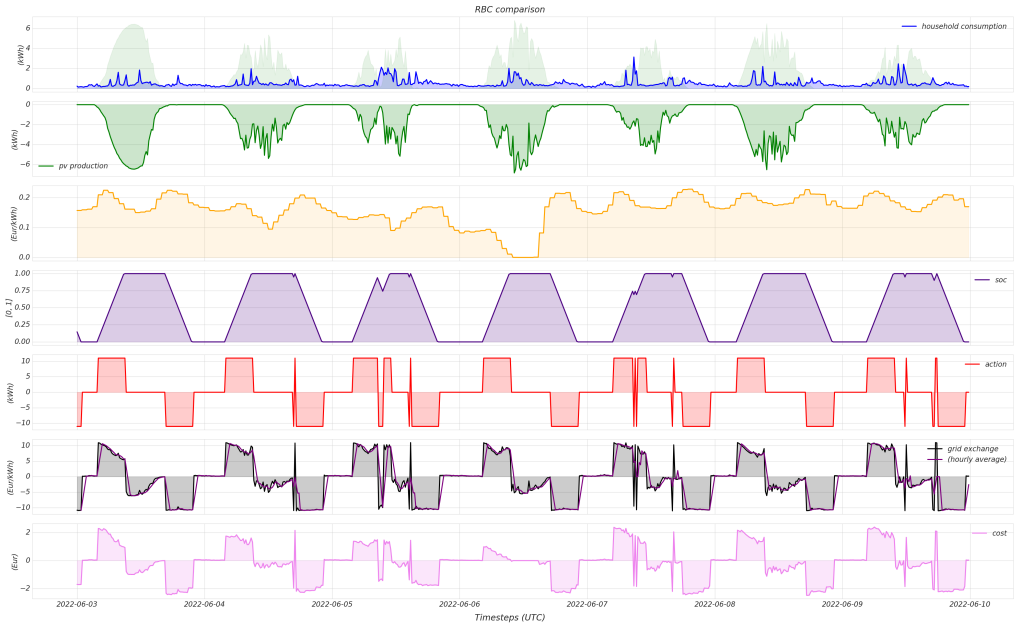

Figure 4.7: *Rule based policy with variable auction price*

Model Config	PPO	SAC	TD3	RBP
Untrained	91.371	91.371	91.371	0.0
Variable AP	459.578	**601.228**	297.622	140.857
Variable AP wit PPO retrained	453.212	-	-	-

Table 4.2: Comparison of performance per policy with variable exchange rate

With simple rule-based strategy as a baseline and score of randomly initialized untrained policy as a comparison criteria, improvement can be seen in each case whether under constant or variable exchange price.

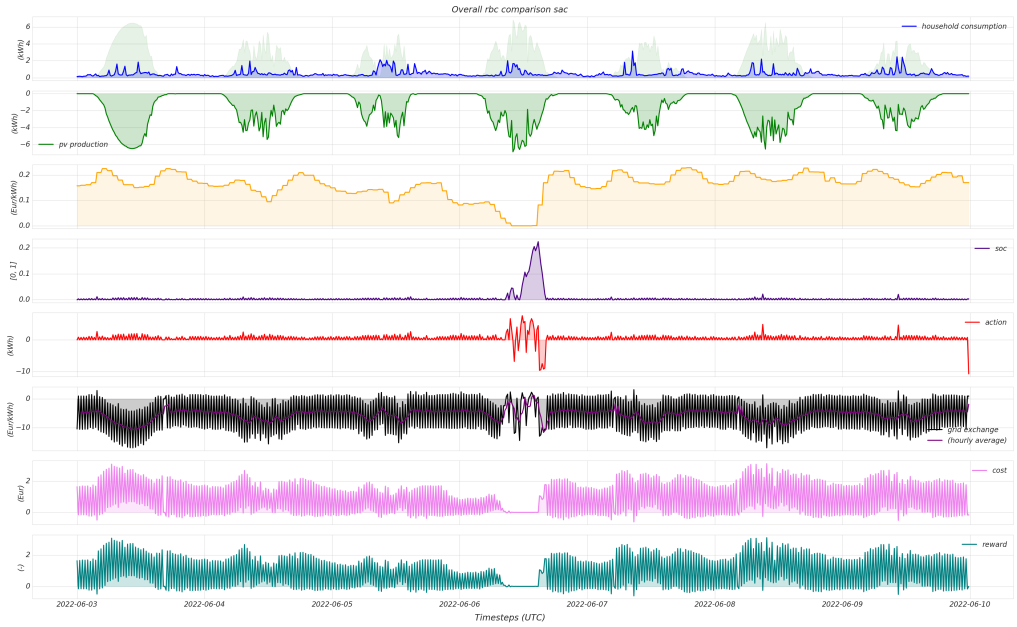

Figure 4.8: *SAC with variable auction price*

4.4 Discussion

There were some improvement that were challenging to address in the implementation setup used to collect observation, mainly design of reward function. It was several time updated back and forth to include and improve the objectives. The combination, as in the case with retaining SoC were resulting in poor observations. The improvement achieved in this combined part is included as supplementary observation 4. The results are better than random policy but in combination with cost optimization objective simultaneously not as much. Multi-objective frameworks are likely suitable to mitigate this issue.

In addition, there were several experimentation and efforts that were made to stabilize and improve the modeling and observation of the Policy. Searching of hyper-parameters were conducted with the use of optimization libraries however, the search trial were based on running the experiment for couple of episode to calculate score. In practice there were runs that even after 100 of episodes suddenly drop their performance to close to initial policy. This led to manual trial and error starting with default values and changing and observing the results.

In general most of the investment were made for iterative implementation, change and experimentation of different components.

Conclusion and Further Work

This work primarily focused on exploration and usage of RL based methods to find control strategies within a Prosumer household in an effort to minimize cost of energy usage.

To address the first research question of how a Prosumer agent is modeled using RL, an experimental platform was implemented using customized simulated HEMS environment and through iterative evaluation and improvement, policy for an Agents actions were modeled. The second question of effective minimization of cost of energy under time-variable tariffs was demonstrated by the test of the agents policy on unseen set of inputs as shown in 4.2. To test out the choice of algorithm and it's impact on performance and efficiency of the control system was also demonstrated with exploration algorithm that are on and off-policy. The off-policy algorithms TD3 and SAC are perfoming better in each of the three scenarion of cost optimization under constant energy exchange price, under variable exchange price and combined objective under variable price with fractional soc retain. Finally, the key factors influencing the optimal control strategies were identified as the selection of RL algorithm, usage of suitable parameters such as size of observation window, number of hidden layers and number of hidden layer in each unit of the Actor, Value and Q network, and contribution of other model specific parameters such as reward discounting, constraint on policy update. Since small change in value of one parameter causes drastic change in behavior of RL based agent due to the sensitivity of these algorithms, it is often the case of trial and improvement as in the case of most of deep neural network based algorithm.

In conclusion, this work demonstrates progressive improvement in performance of RL algorithm, starting from randomly initialized network, learning in different set of observation input setting and performing with significant score on the defined objective.

Further improvement can be made to include multi objective reward optimization, searching for more optimal parameters and inclusion of more input features and separate action space for EVSE and Household battery charger.

Supplementary Observations

This is a supplementary section including optimization of multiple objectives of minimizing the net cost and retain 45% of the battery soc. The observation of performance of policy during training and testing is described in this section. To retain defined amount of SoC during policy learning was implemented by encouraging the agent with the value of 0.25, if it keeps the SoC value above 45% and -0.25 otherwise. This value is added to the cost reward to obtain the final reward.

Combined Reward Optimization (cost with soc retain)

The performance of a policy of all the model based on cost reward and combined reward is shown in figure 4.9 and 4.10 respectively. It is clear from these figures, the agent is maximizing the combined reward with higher emphasis because it achieves maximum value of around 800. On the other hand the cost based reward only achieves maximum value around 115. Since there is no rule based comparison for multiple rewards, the rl based models are compared among themselves by referencing the score during initialization.

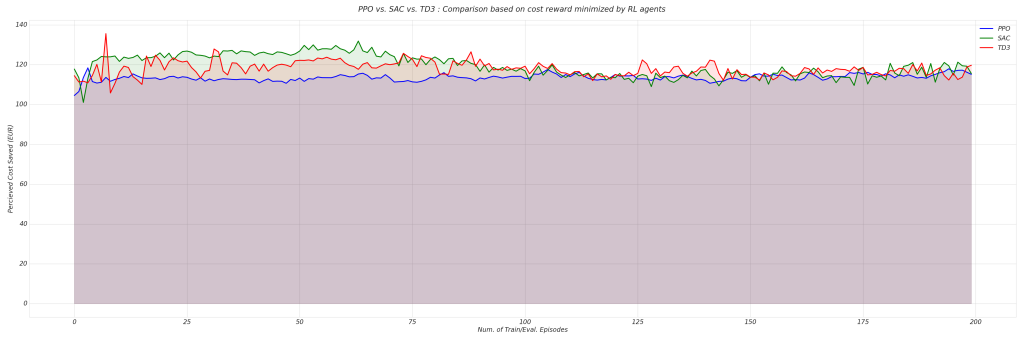

Figure 4.9: *Cost reward evaluation per episode with variable auction price and soc retain*

The combined reward value has no significance by itself like cost based reward which is calculated with the energy balance equation described by 3 and actually represents net exchange cost. The combined reward dominates the cost reward with

the soc retain reward value of ±0.25. Decreasing this value retains less to no soc and increasing the value only increases the combined reward at the expense of cost reward.

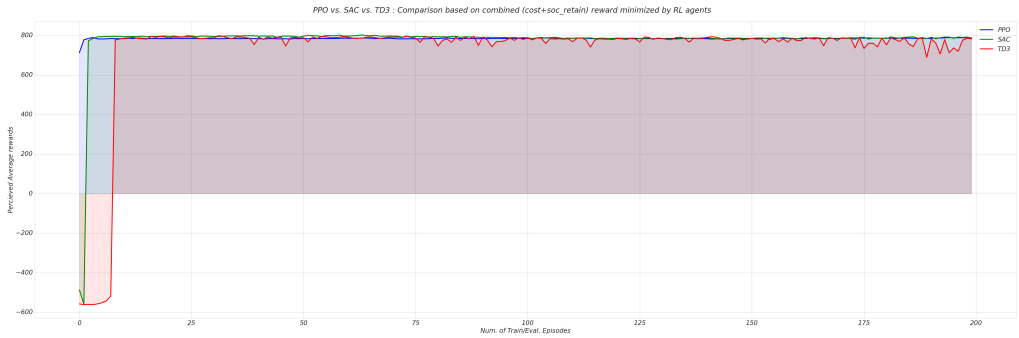

Figure 4.10: *Combined reward evaluation per episode with variable auction price and soc retain*

While testing of the learned policy reinforced by this combined reward, each model retains the desired amount but the cost optimization is hindered as shown in the table 4.3

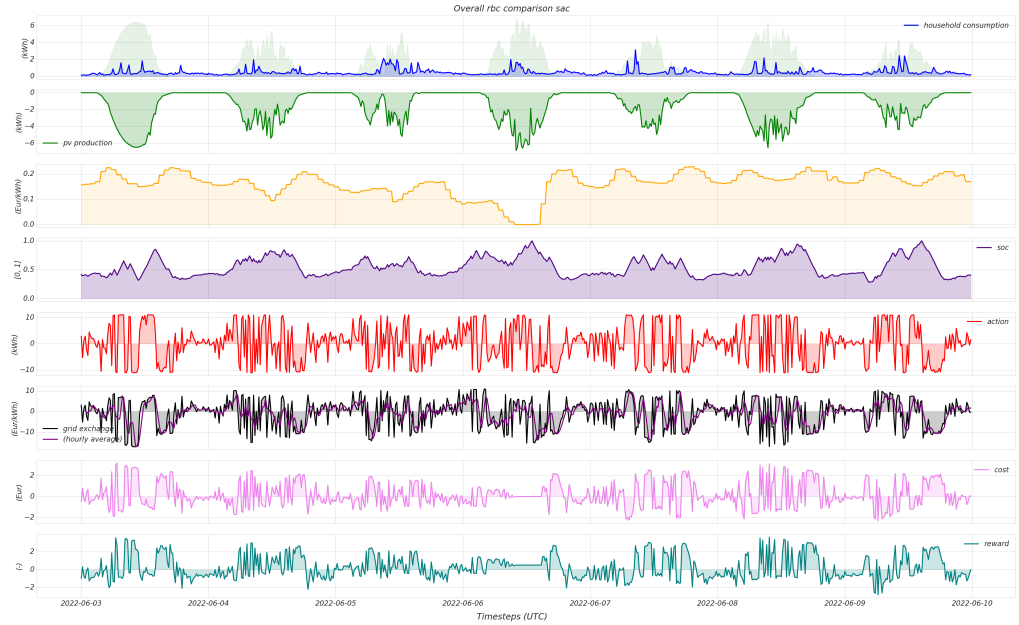

Figure 4.11: *Agents sample action from SAC policy with variable auction price and soc retain*

The figures 4.11, 4.12 show that, regardless of hindrance in average cost minimization, both policies show some pattern of reflected variability in the observation

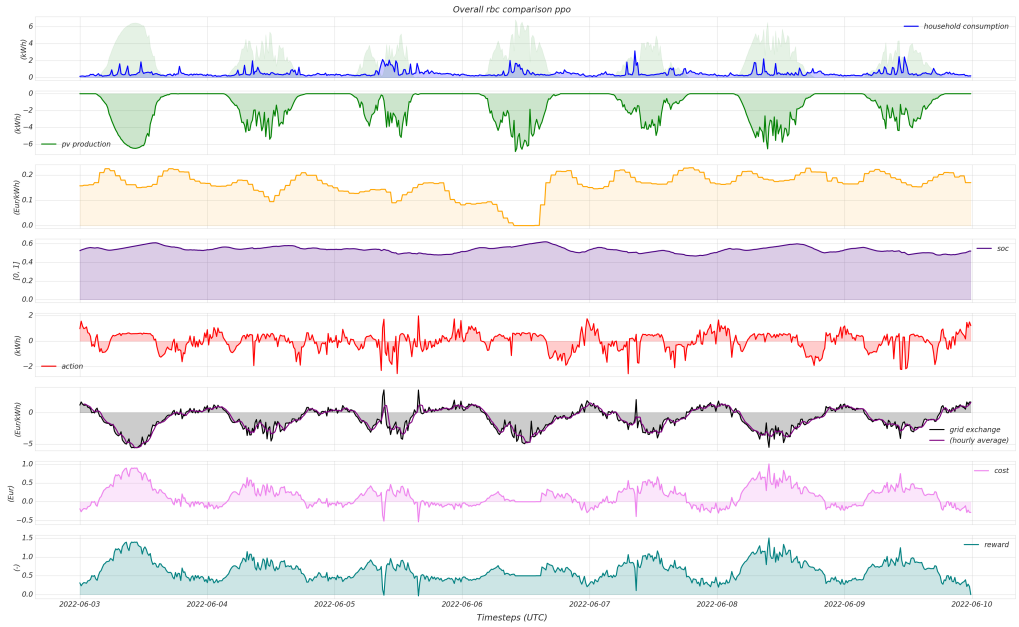

Figure 4.12: *Agents sample action from PPO policy with variable auction price and soc retain*

Model Config	PPO	SAC	TD3
Untrained	91.371	91.371	91.371
Variable AP and SoC retain	98.325	**109.296**	93.987

Table 4.3: comparison per policy with variable exchange rate

input for SAC and PPO respectively. The patterns are most visible in SAC, which resonates with the relatively higher value than other RL based strategies. The TD3 has barely improved on from the random state of initialization.

Overall, these observations show that the objective of retaining SoC is met, but the average cost of exchange is not minimized, indicating that the improvement in cost contribution could be made with searching for better definition of reward function and balanced composition of combined rewards constituents.

List of Figures and Tables

Figures

Tables

Bibliography

[1] The Paris Agreement. (2015). *UN Climate Change Conference*

[2] Climate Change Act, Intergenerational contract for the climate. (2021). *Die bundesregierung, Energie und Klimaschutz.*

[3] Energy charts info. (2024). *www.energycharts.info.*

[4] Moen, Roger. Solar energy management system. (1979). *IEEE,* 18^{th} *Conference on Decision and Control.*

[5] Mohammad et al. (2021). Integration of Electric Vehicles and Energy Storage System in Home Energy Management System with Home to Grid Capability. *Energies 2021, 14, 8557*

[6] Hou et al. (2019). Smart Home Energy Management Optimization Method Considering ESS and PEV. *ICIEA*

[7] ZHU et al. Optimization of rule-based energy management strategies for hybrid vehicles using dynamic programming. (2021). *Combustion Engines.*

[8] Lee et al. (2019). Reinforcement Learning-Based Energy Management of Smart Home with Rooftop Solar Photovoltaic System, Energy Storage System, and Home Appliances. *ensors (Basel). 2019 Sep 12;19(18):3937*

[9] Specht et al. (2023) Deep reinforcement learning for the optimized operation of large amounts of distributed renewable energy assets. *Energy and AI 11 (2023) 100215*

[10] Ostermann et al, (2023). *Bidirectional Electric Vehicles Field Trial Data Set*, NEIS 2023.

[11] Sutton et al. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

[12] OpenAI: Spinning Up in Deep Reinforcement Learning. (2018).

[13] Schulman et al. (2017). *Proximal Policy Optimization Algorithms.* arXiv:1707.06347v2.

[14] Tuomas et al. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.* arXiv:1801.01290v2.

[15] Fujimoto et al. (2018). *Addressing Function Approximation Error in Actor-Critic Methods*. arXiv:1802.09477v3.

[16] Raffin et al. (2021). *Stable-Baselines3: Reliable Reinforcement Learning Implementations*. Journal of Machine Learning Research.

[17] Towers et al. (2023). *Gymnasium, An API standard for single-agent reinforcement learning environments*. Zenodo.

[18] Schulman et al. (2017). Trust Region Policy Optimization *arXiv:1502.05477v5*.

[19] Lillicrap et al. (2019). Continuous control with deep reinforcement learning. *arXiv:1509.02971v6*.