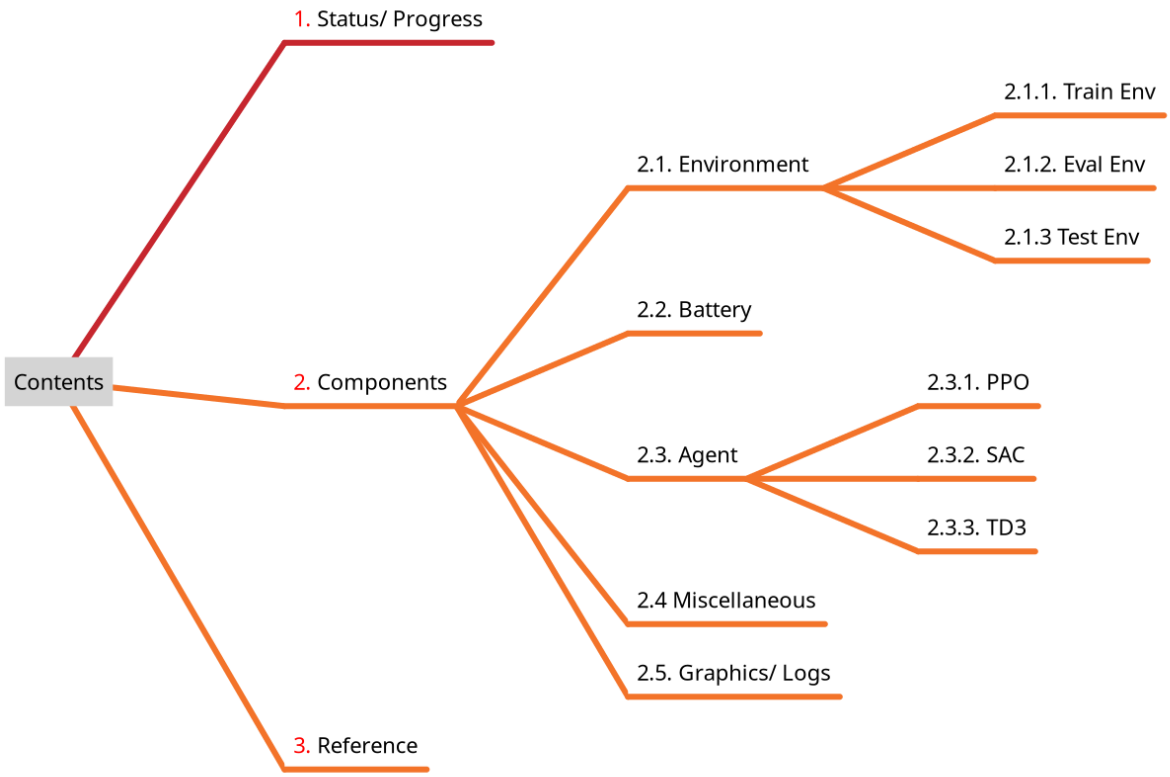
Training Prosumer Agents with Reinforcement Learning.

>>> **Biweekly Report 5.** ($7^{th}\,Apr - 24^{th}\,May : 2024$)



---

1. Status/ Progress

Current Iteration

- [x] vectorized environment for parallel learning
- [x] separate environment for train/eval/test added
- [x] parameter search script added with optuna library
- [x] improvement in structure & documentation

Next Iteration (Plan)

- [ ] explore common comparison metrics with rule based vs rl algorithms.
- [ ] run search for better hyperparameters tuning
- [ ] observe and report impact of different combination of input observation, parameters on rewards.

---

2. Components

Taking feedback from weekly catchups into account and updated understanding of the system following changes were made to different components.

2.1. Environment Development

2.1.0. **Rewards**

- Running train and evaluation side by side, the reward function shows some trends and plateaus to a celling value as shown in the graphics/logs section below.
- The simplified version of reward function is removed and instead energy cost based function is used.

2.1.1 **Train Env**

- A separate training environment was implemented that is used for policy learning. It could be run into different numbers (e.g. 1, 2, 4, ...) in parrallel, using forked processes. This is to learn variability with sample efficiency when training in shuffled(yet to setup) batches.
- It is also used for retraining a policy from a saved checkpoint.

2.1.2 **Evaluation Env**

- A separate single evaluation environment is implemented for calculating mean reward per evaluation steps.

2.1.3 **Test Env**

- Test environment is setup for running trained policy, collect transitions with graphics/tensorboard logs and checking the actions, battery socs in comparison to the input observations.

2.2. Battery Module

- no change were made to the battery module.

2.3. Agent

- Algorithm in use:
  - Proximal Policy Optimization (PPO)
  - Soft Actor Critic (SAC)
  - Twin Delayed Deep Deterministic Policy Gradient (TD3)
- As per the used framework and referenced paper [1], the models above were setup for comparison.

2.4. Miscellaneous

- Improvement in running models: `python main.py --model=ppo/sac/td3/rbc --action=train/retrain/test`
- Addition of simple rule-based-control scenario for further comparison with rl-based strategies.
- Addition of search of relevant parameters using optuna [2] library
- Simplification of agents, graphics, documentation and configuration options.

2.5. Graphics/ Logs

parameters used

```
data_config["train_test_split_ratio"] = 0.04
policy_config: dict = {
    "policy_nw": "MlpPolicy",
    "reset_num_timesteps": False,
    "num_train_eval_cycles": 20,
    "num_retrain_eval_cycles": 20,
    "num_eval_episodes": 3,
    "num_test_episodes": 20,
    # ~ 14 days
    "train_timesteps": data_config["train_test_split_ratio"] * 35904,
"retrain_timesteps": 1500,
}
```

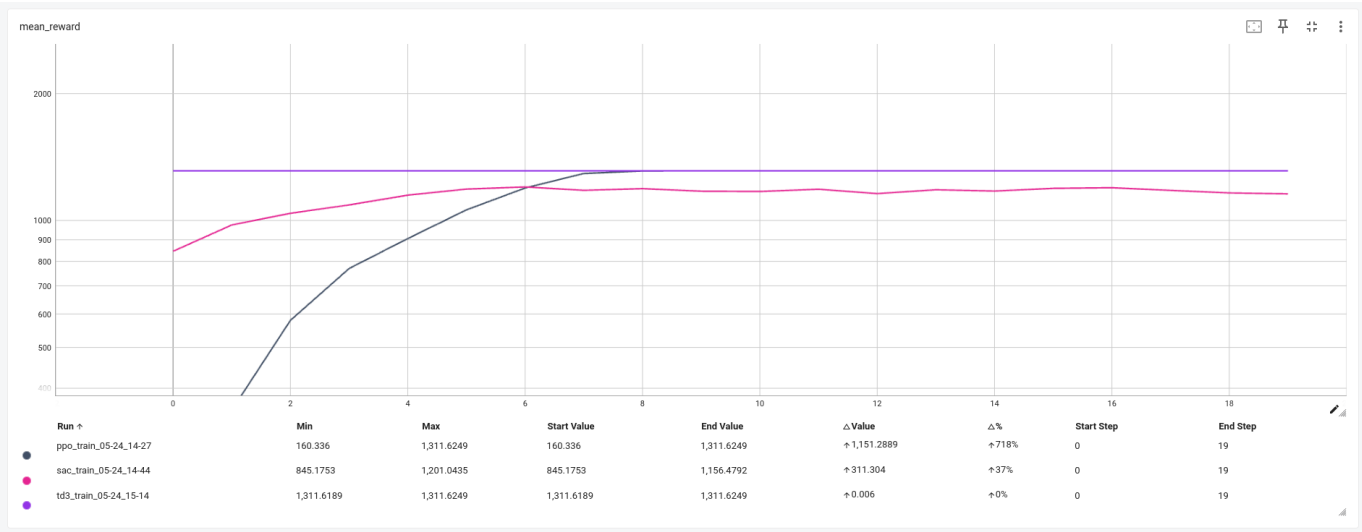
- The data used for observation is 1436 timesteps (~`14 days`).
- Reset of training timesteps is set to False, to continue train and evaluation on previously trained and saved model.
- Number of training and eval cycles is the number of iteration of training and evaluation, i.e. `num_train_eval_cycles` : 20 iterations of train with number of `train_timesteps` : 1436 with

policy evaluation(computing mean reward) of `num_eval_episodes` : 3 episodes for each iteration.

- Number of retraining and eval cycle is as the training iteration.
- Number of test episode is the number of episodes to sample the statistics from the trained policy.

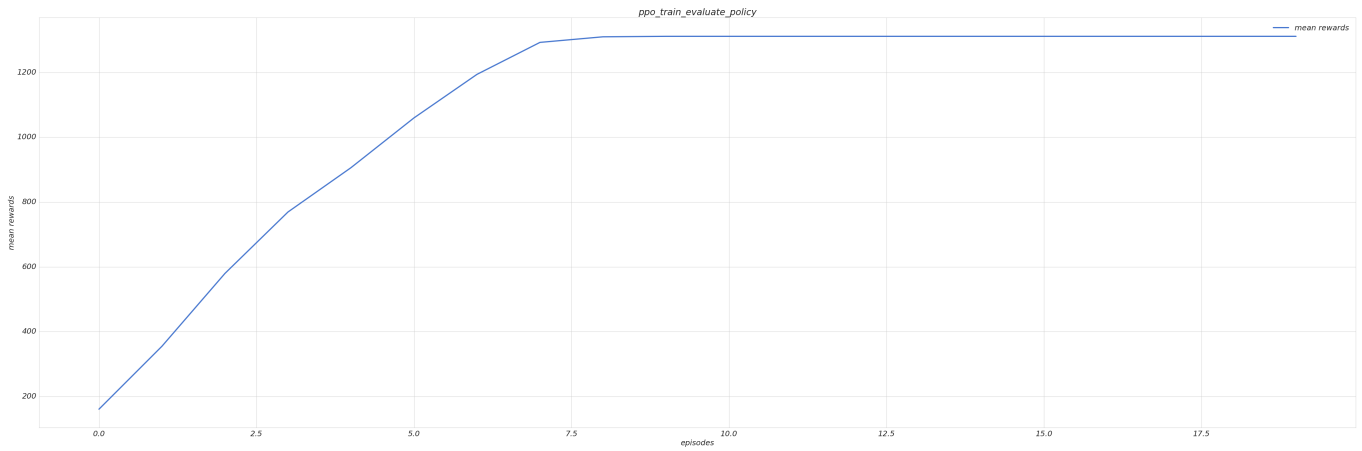
During Learning/ Evaluation

Evaluation of Mean Rewards per iteration : 20, with each model.

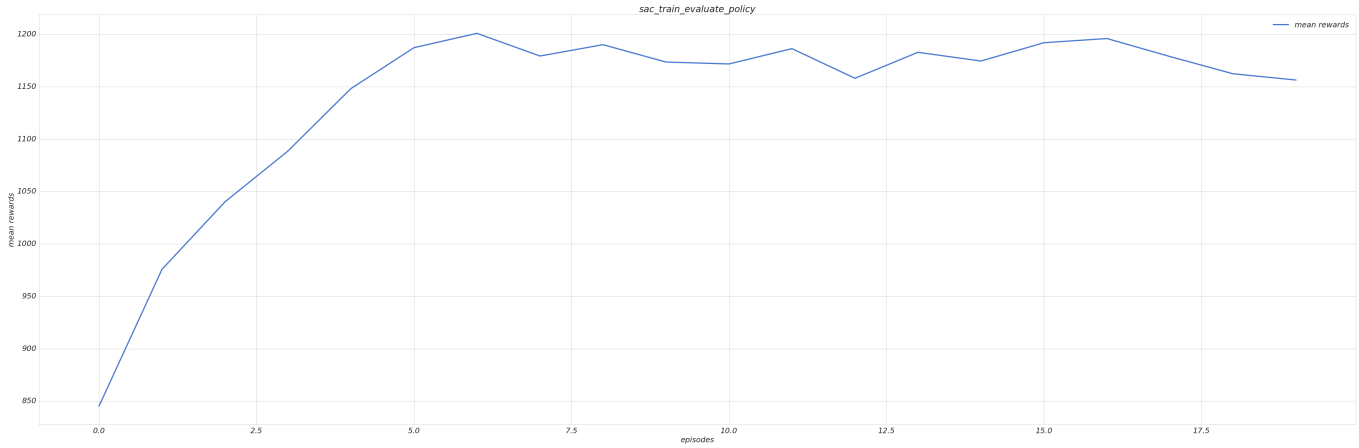


- The agents learns to maximize the reward funciton from a few number of iteration(ppo, and sac) to even single iteration(in case of td3), given short length of input observations (i.e. 1436 data timesteps) , then plateaus to a max celling cost reward. The evaluation is shown in detailed in the following figures for each models.

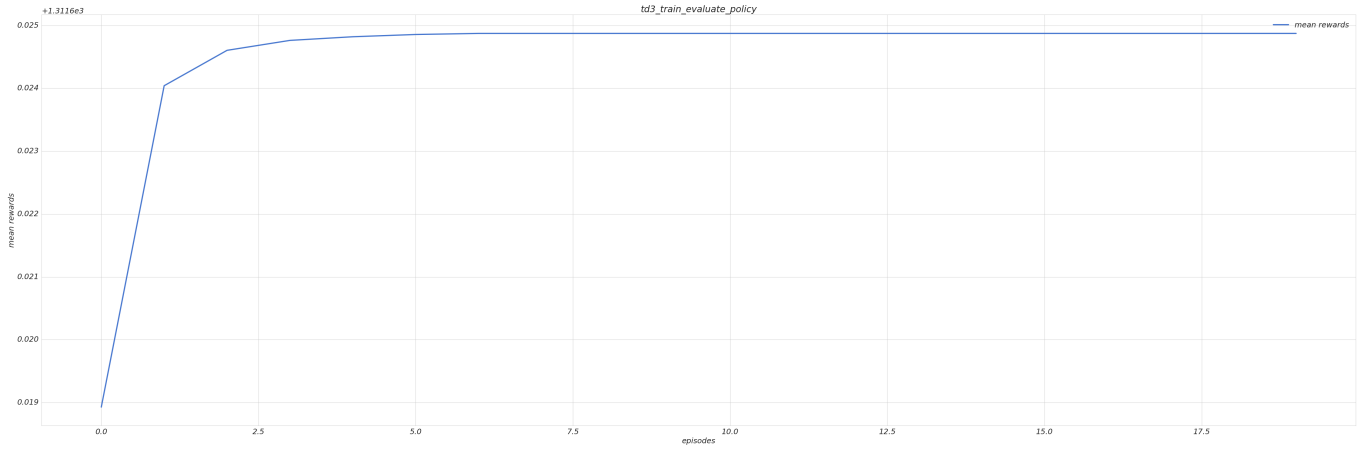
PPO



SAC

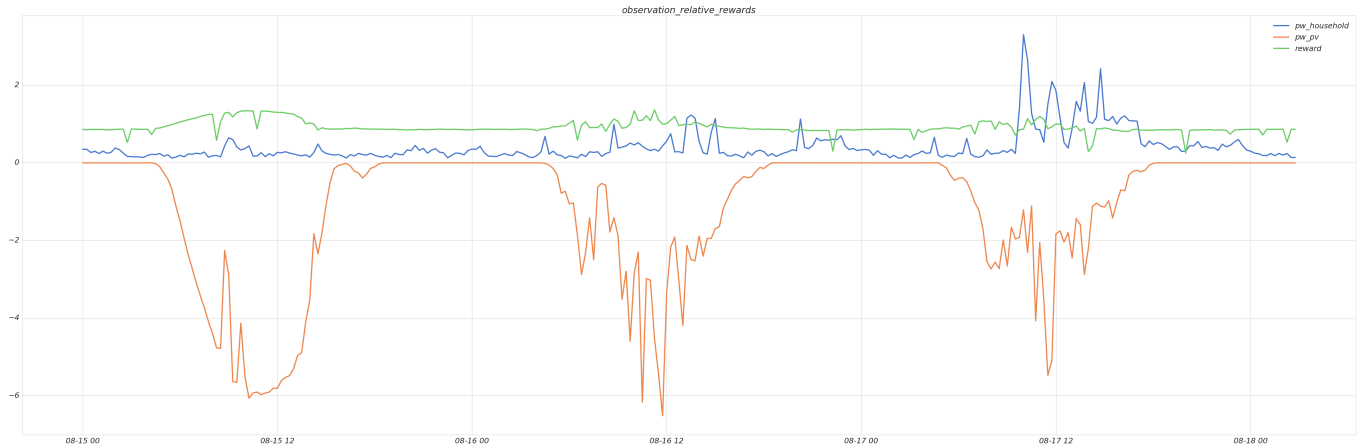


TD3

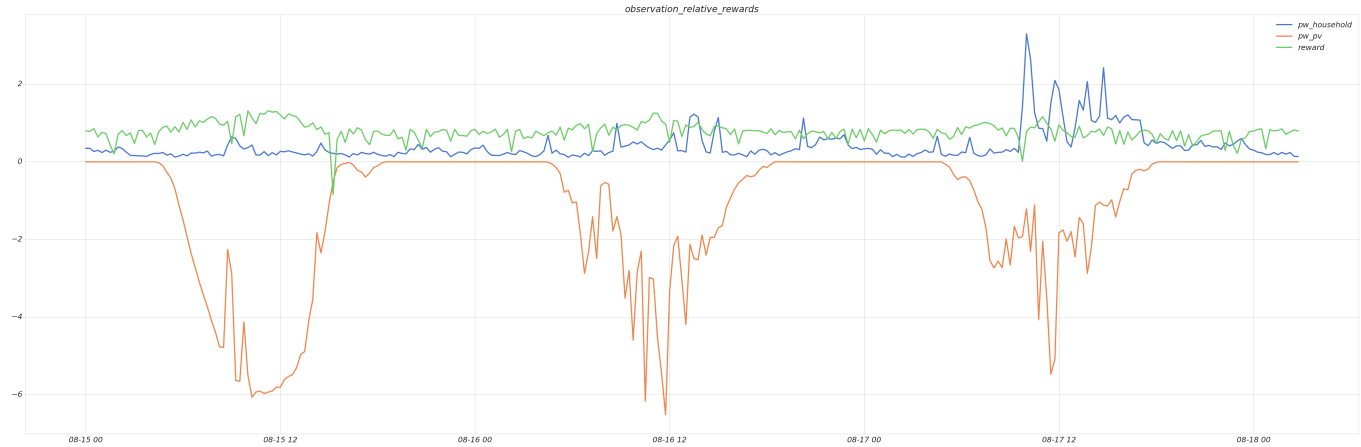


During Sampling Actions from trained policy (non deterministic)

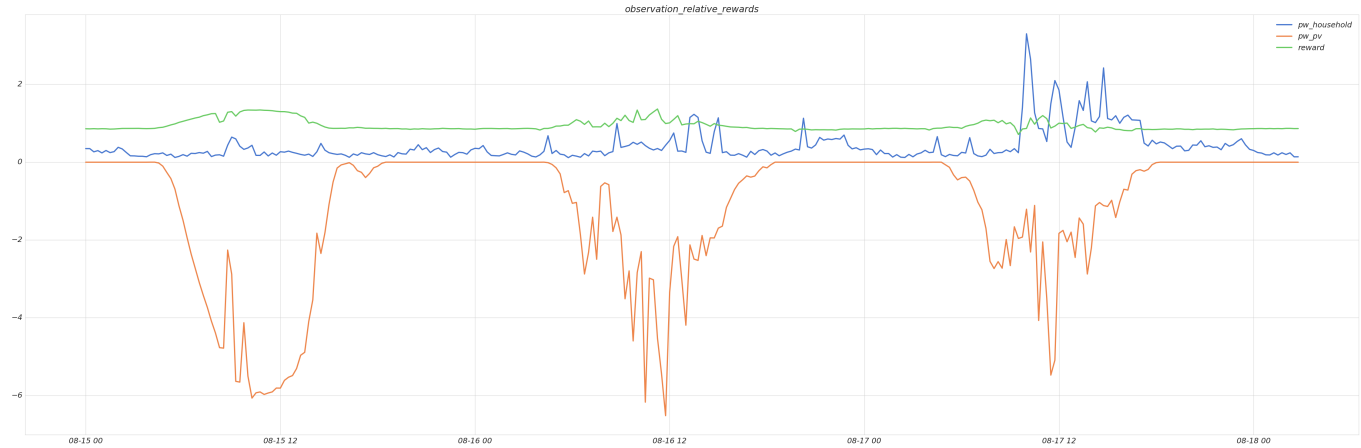
PPO



SAC


observation_relative_rewards
pw_household
pw_pv
reward
2
0
−2
−4
−6
08-15 00
08-15 12
08-16 00
08-16 12
08-17 00
08-17 12
08-18 00

TD3


observation_relative_rewards
pw_household
pw_pv
reward
2
0
−2
−4
−6
08-15 00
08-15 12
08-16 00
08-16 12
08-17 00
08-17 12
08-18 00

RBC

5. References

- [1.] Deep reinforcement learning for the optimized operation of large amounts of distributed renewable energy assets
- [2.] Optuna: A hyperparameter optimization framework