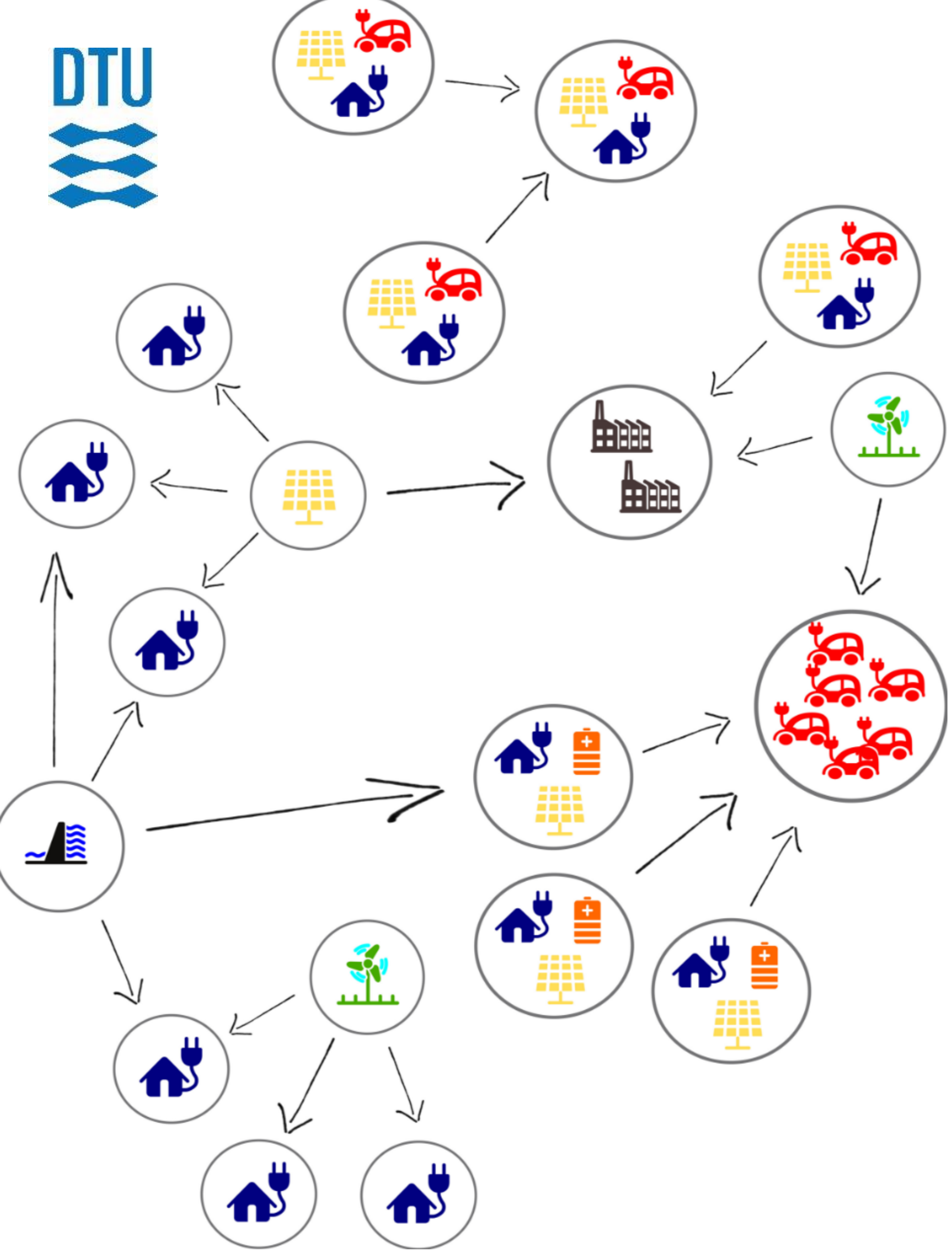
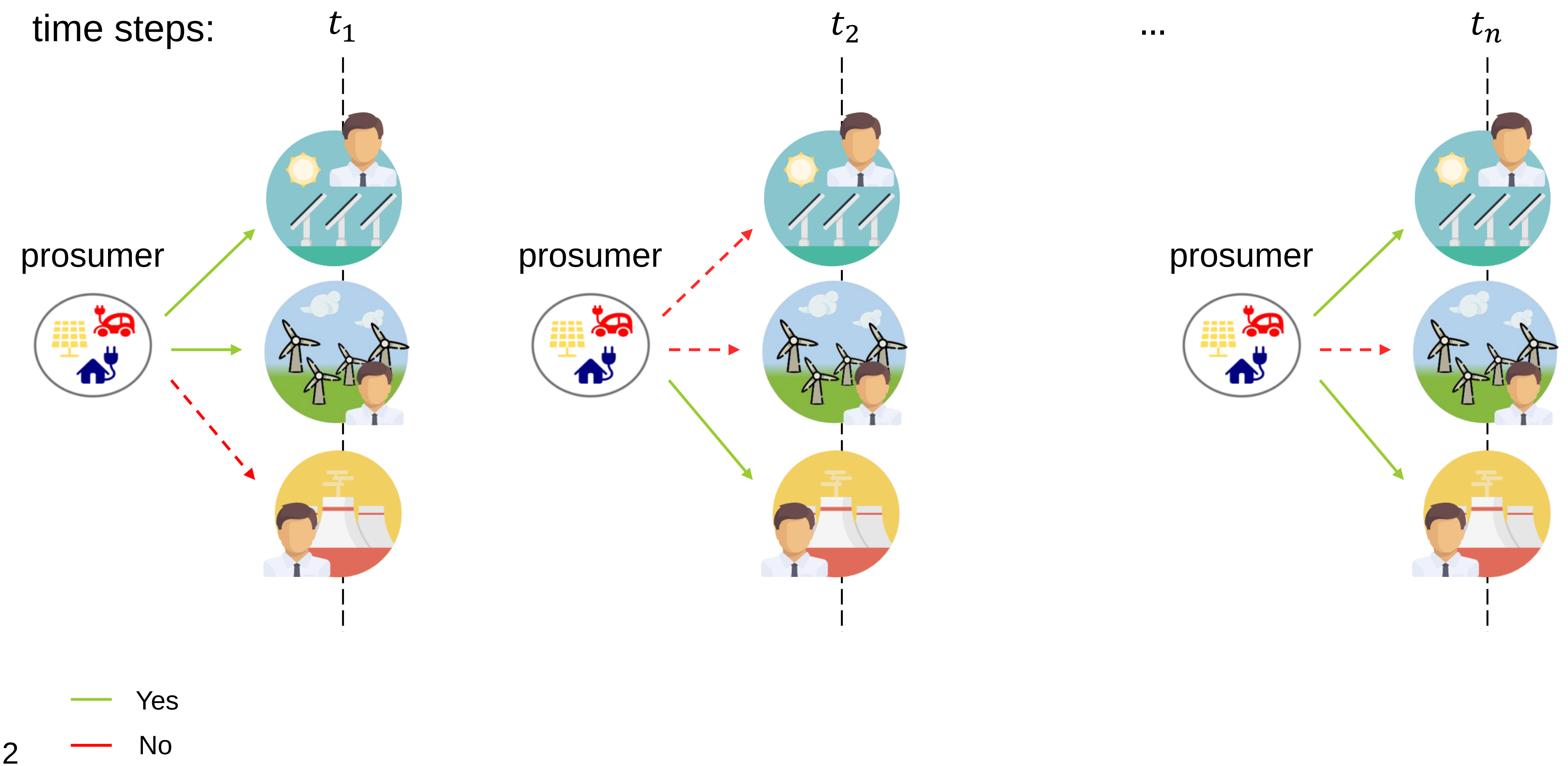


Center for Electric Power and Energy
Department of Electrical Engineering

Reinforcement Learning applications for Peer-to-Peer energy markets

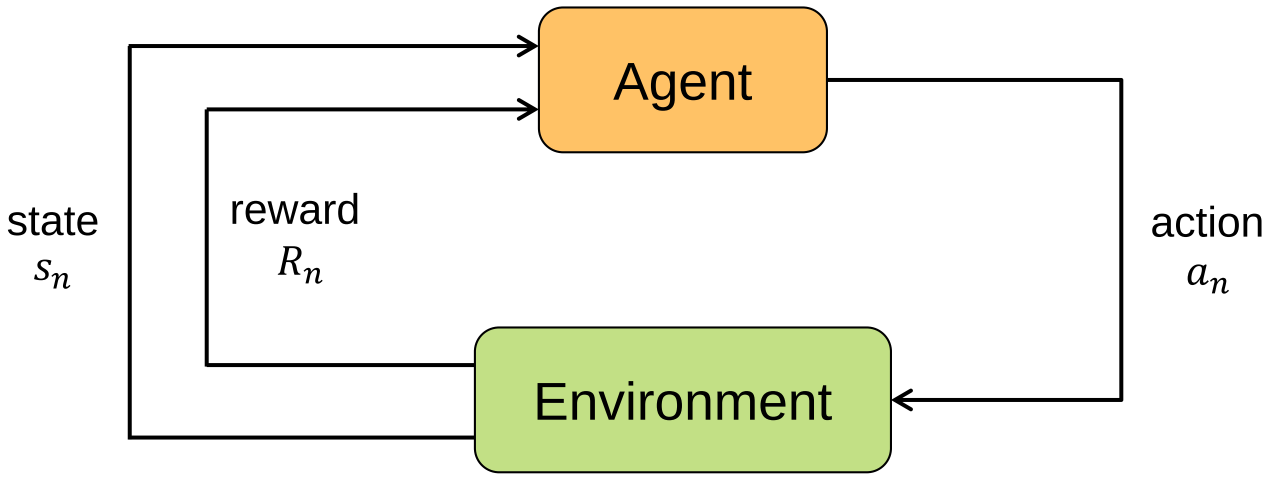
Tiago Sousa, Pierre Pinson

Motivation



Reinforcement learning

- We can design the P2P negotiation as Reinforcement Learning
  - Agent learns the optimal policy by interacting with the environment (P2P market)



Outline

- Reinforcement Learning approach
  - Multi-Armed Bandit
- Test case and results
- Conclusions and next steps

Multi-Armed Bandit

Agent

prosumer

Step n	***Arm 1***	***Arm 2***	***Arm 3***
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1

actions a_n

Step n ≠ time t

Multi-Armed Bandit

Agent

prosumer

Environment

Step n	***Arm 1***	***Arm 2***	***Arm 3***	***Reward***
1	1	0	0	1
2	0	1	0	0
3	1	0	0	0
4	0	0	1	1

Total reward

$$R_{Total} = \mathbb{E}(R_n) = \frac{1}{N}\sum R_n$$

$$a_n^* = \text{argmax } R_{Total}$$

Multi-Armed Bandit

Agent

prosumer

Environment

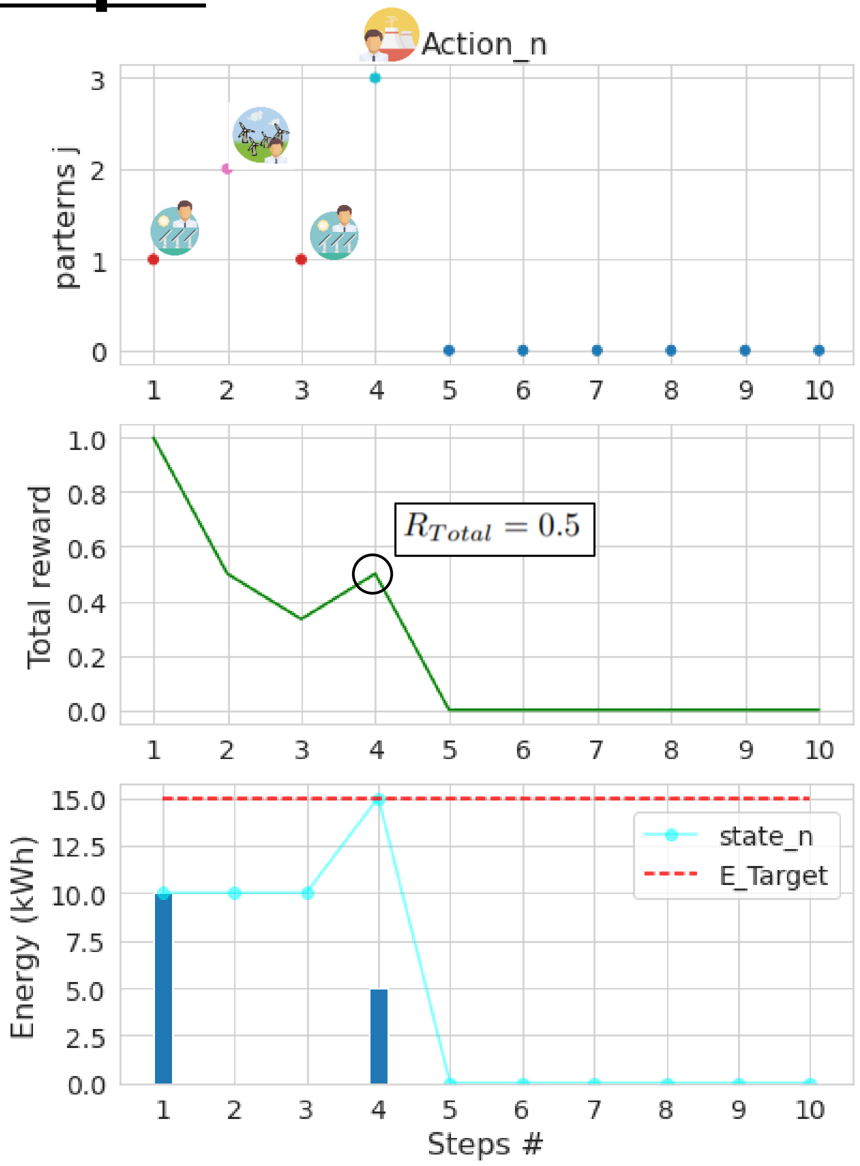
Step n	***Arm 1***	***Arm 2***	***Arm 3***	***Reward***	***Energy (kWh)***
1	1	0	0	1	10
2	0	1	0	0	0
3	1	0	0	0	0
4	0	0	1	1	5

State per step *n*

$$s_n = \sum E_n(j) R_n$$

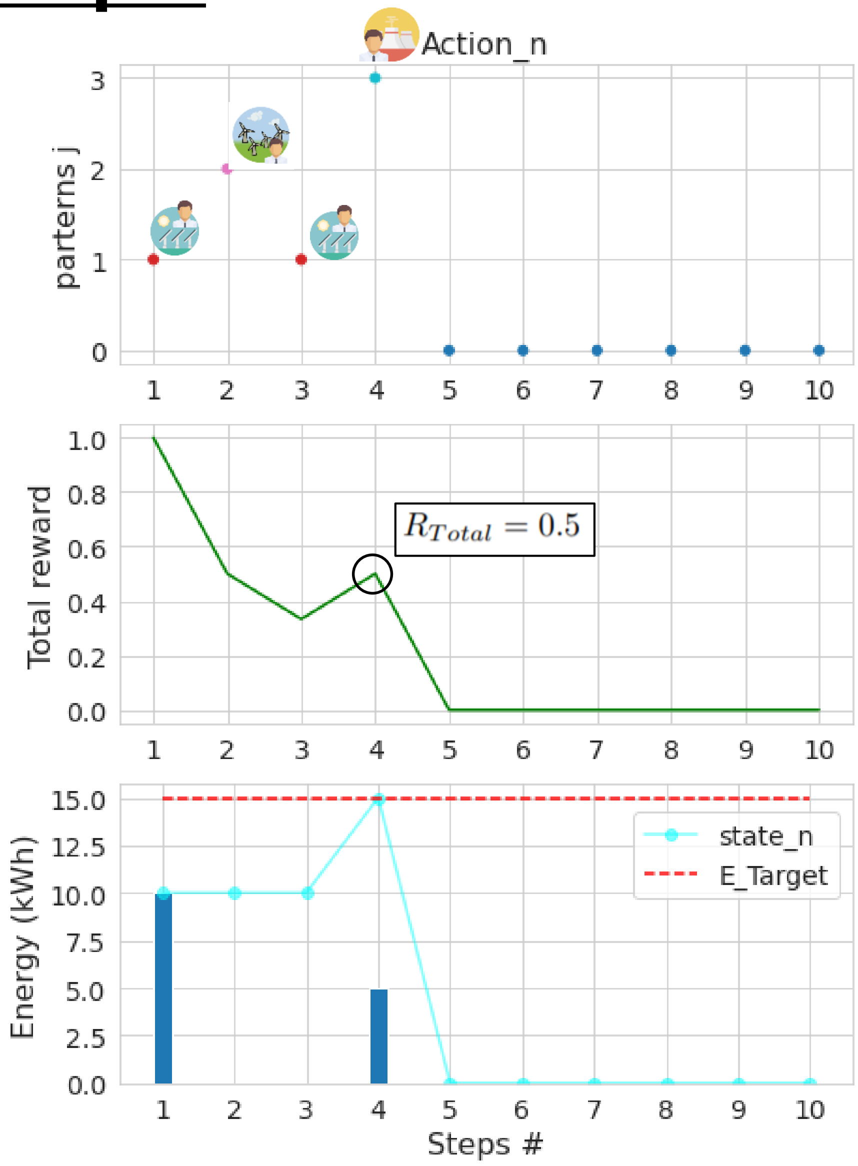
$$s_n \leq E_{Target}$$ Stopping condition

Example



Action_n
parterns j
3
2
1
0
1 2 3 4 5 6 7 8 9 10
Total reward
1.0
0.8
0.6
0.4
0.2
0.0
$R_{Total} = 0.5$
1 2 3 4 5 6 7 8 9 10
Energy (kWh)
15.0
12.5
10.0
7.5
5.0
2.5
0.0
state_n
E_Target
1 2 3 4 5 6 7 8 9 10
Steps #

Example

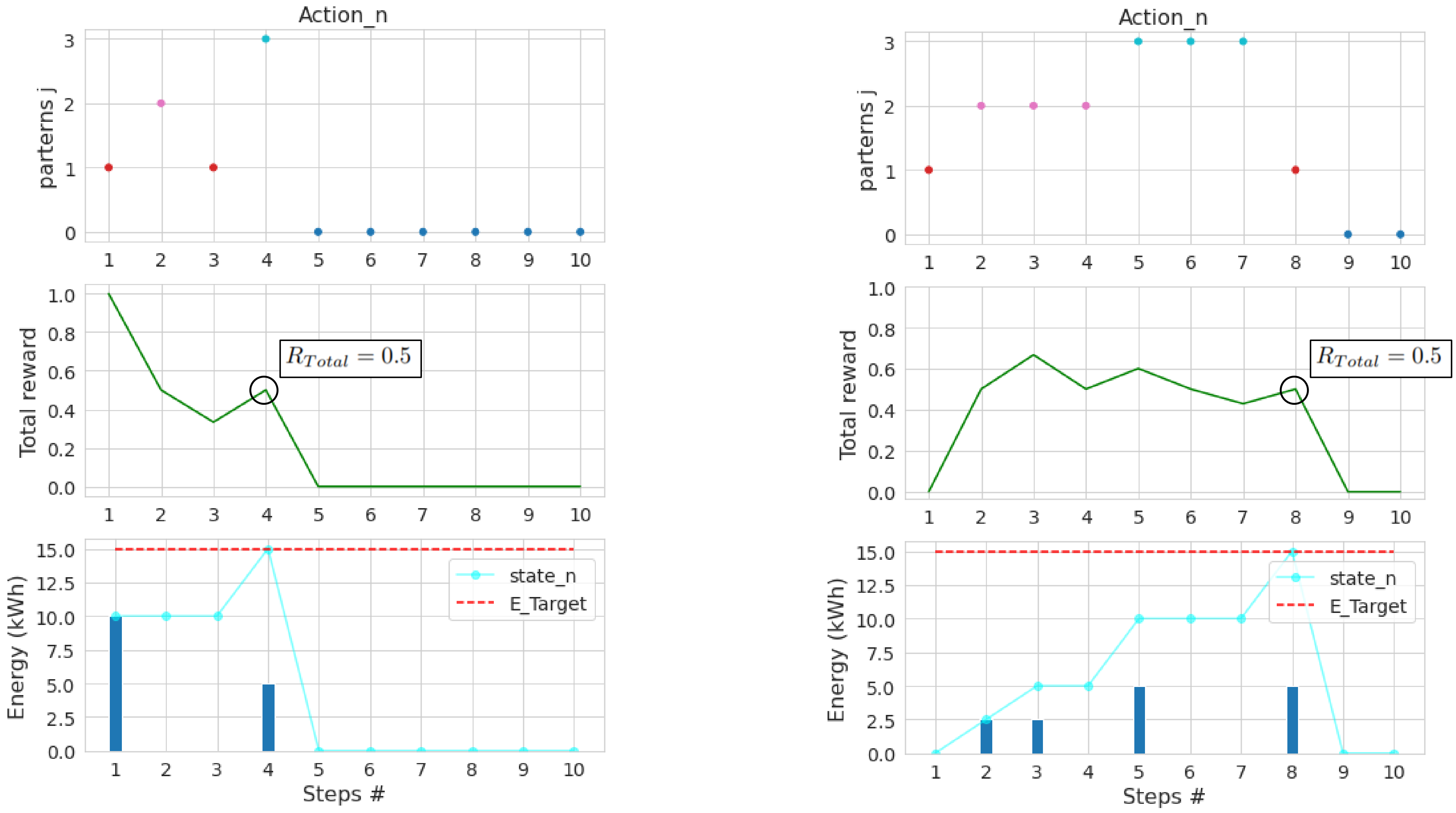


- This iterative process is an episode

- We terminate when
  - $s_n = E_{Target}$

episode = time *t*

How to differentiate between episodes?



Multi-Armed Bandit

- Reward is a random variable

Bernoulli distribution

$$R_n(j) \sim \mathrm{B}(1, p_j)$$

For a large number of steps *n*:

$$R_n(j) \approx p_j$$

Step n	***Arm 1***
1	1
2	0
3	1
4	0

Environment

Reward
1
-
0
-

Multi-Armed Bandit

- Reward is a random variable

Bernoulli distribution

$$R_n(j) \sim \mathrm{B}(1, p_j)$$

For a large number of steps *n*:

$$R_n(j) \approx p_j$$

Step n	***Arm 1***
1	1
2	0
3	1
4	0

Environment

Reward
1
-
0
-

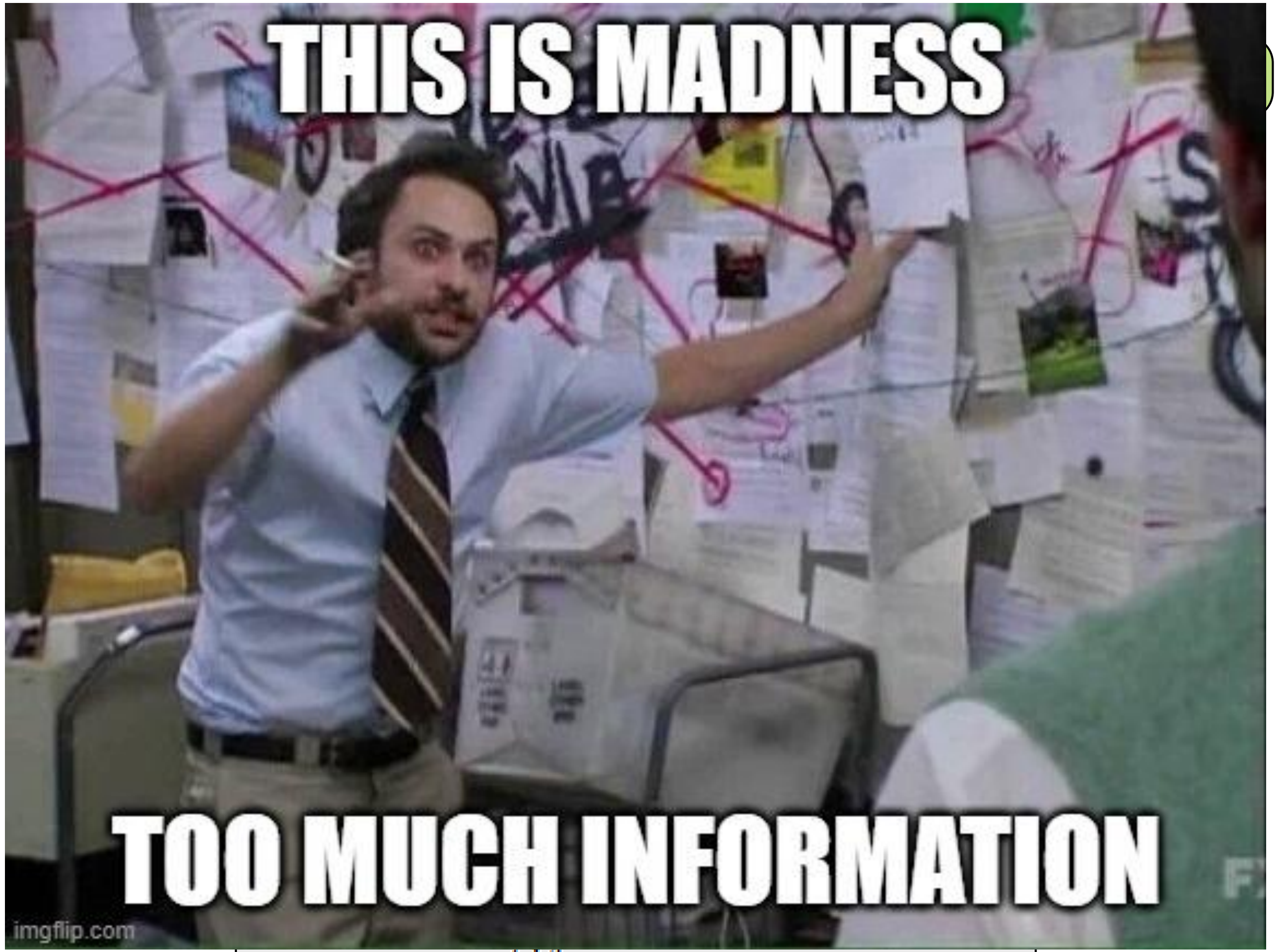
Action-Value function

Estimator of p_j for every arm j

$$Q_n(j) = \frac{1}{N_j} \sum R_n(j) = \hat{p}_n(j)$$

Multi-Armed Bandit

- Learning algorithms estimate the Action-Value function $Q_n(j)$
  - Random
  - ϵ-greedy
  - Thompson Sampler
  - Upper Confidence Bound



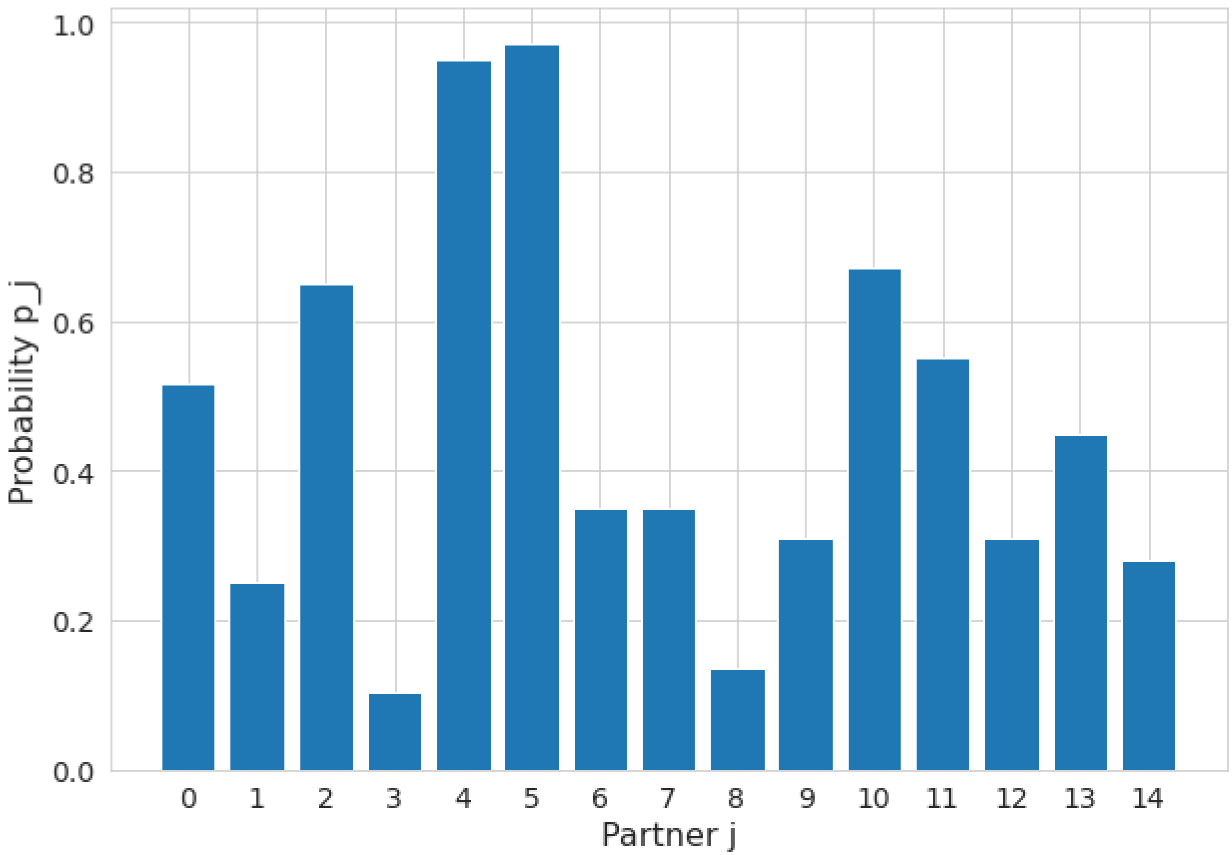

Enough is Enough
sky
sports
KAPWING
We want RESULTS

Test case

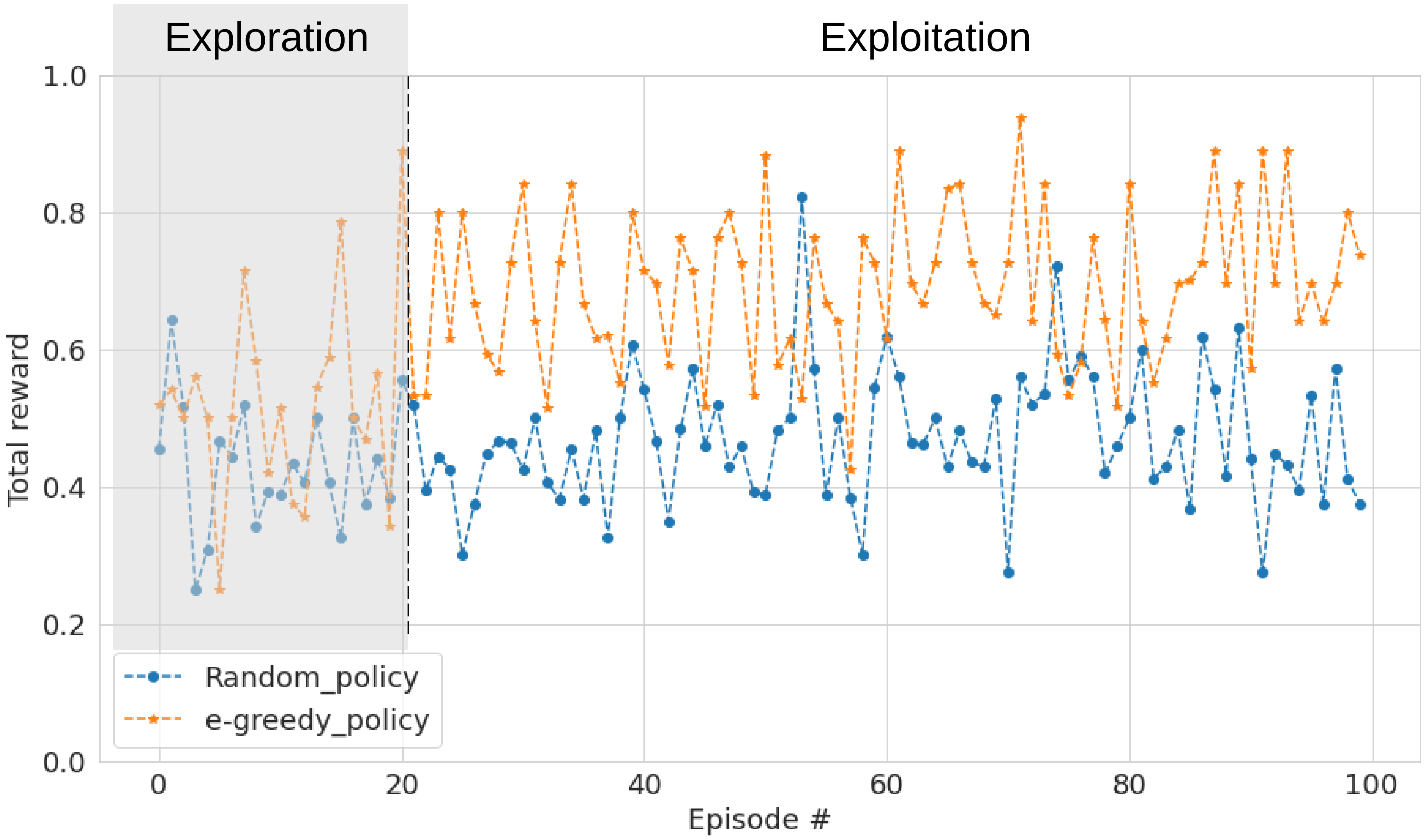
- Case with 15 parterns j

- Test for 100 episodes
  - $E_{Target} = 15 \text{ kWh}$

- Learning algorithms:
  - Random
  - ϵ-greedy

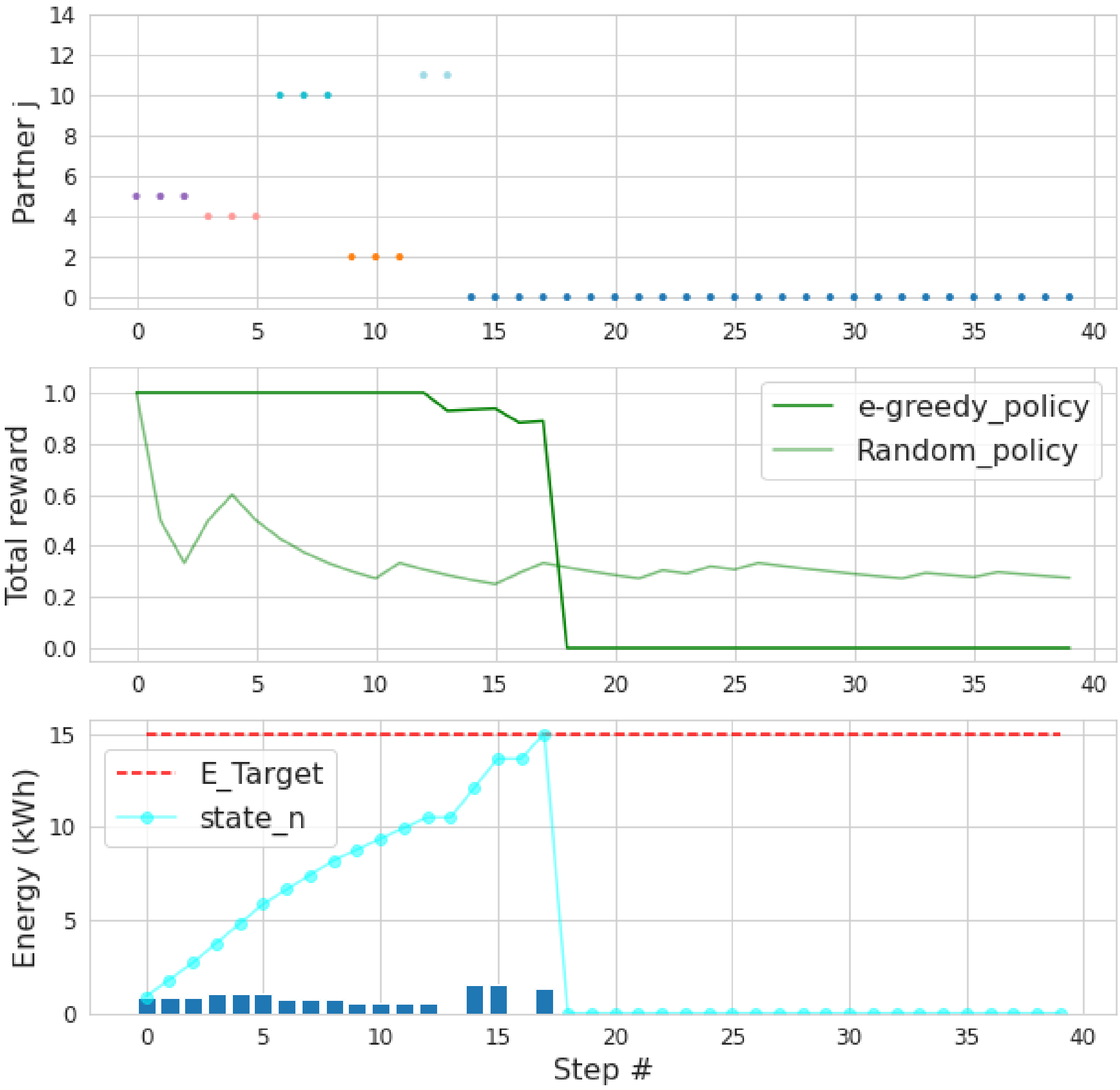


Test case



Exploration
Exploitation
1.0
0.8
0.6
0.4
0.2
0.0
Total reward
0
20
40
60
80
100
Episode #
Random_policy
e-greedy_policy

Epsilon-Greedy algorithm



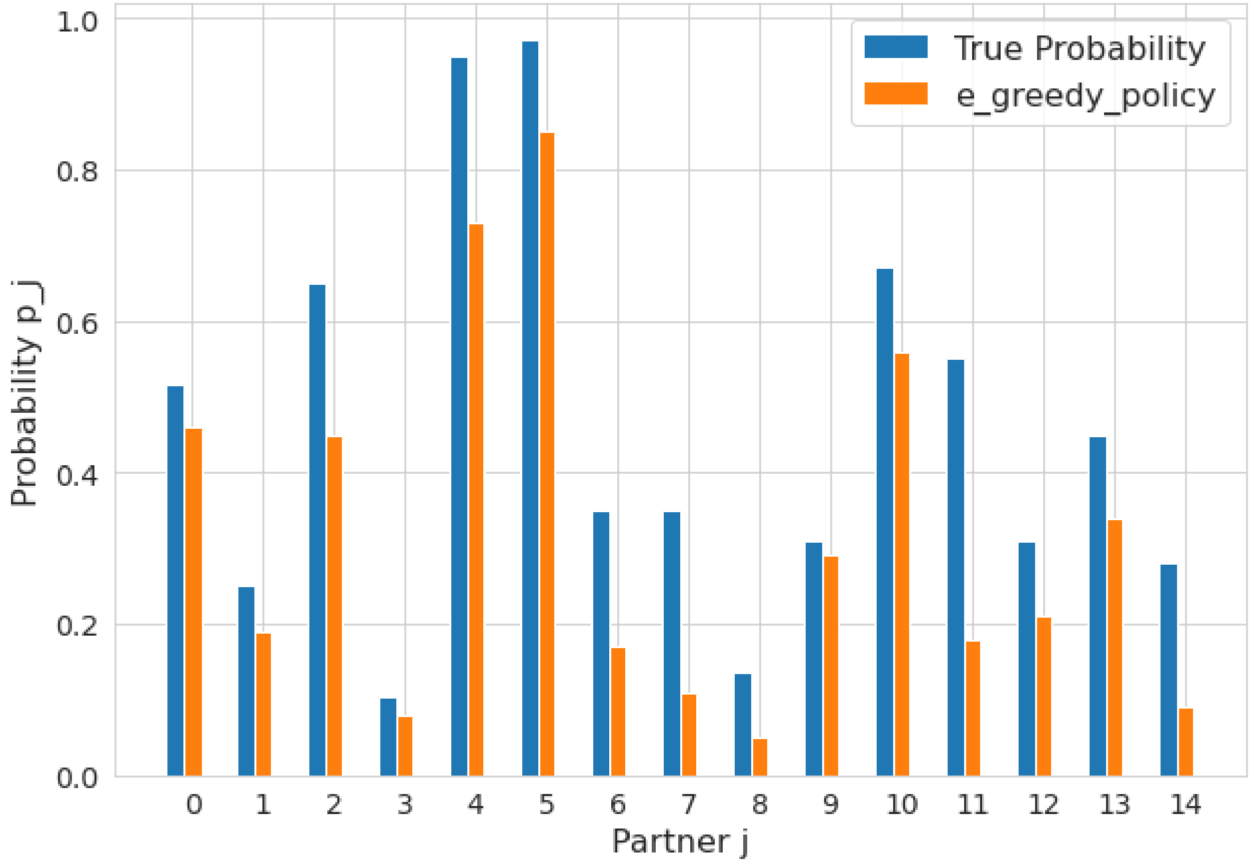
- Solution found on episode 91
  - $R_{Total} = 0.89$

- However, there is no guarantee to reach always this reward value:

$$\overline{R}_{Total} = 0.67 \qquad epi \in [21, 100]$$

Optimal solution

- We can compute the estimator Q_j^*
  - $Q_j^* \approx p_j$
  - Calculate the mean Q_j^* for the last 10 episodes



Conclusions and next steps

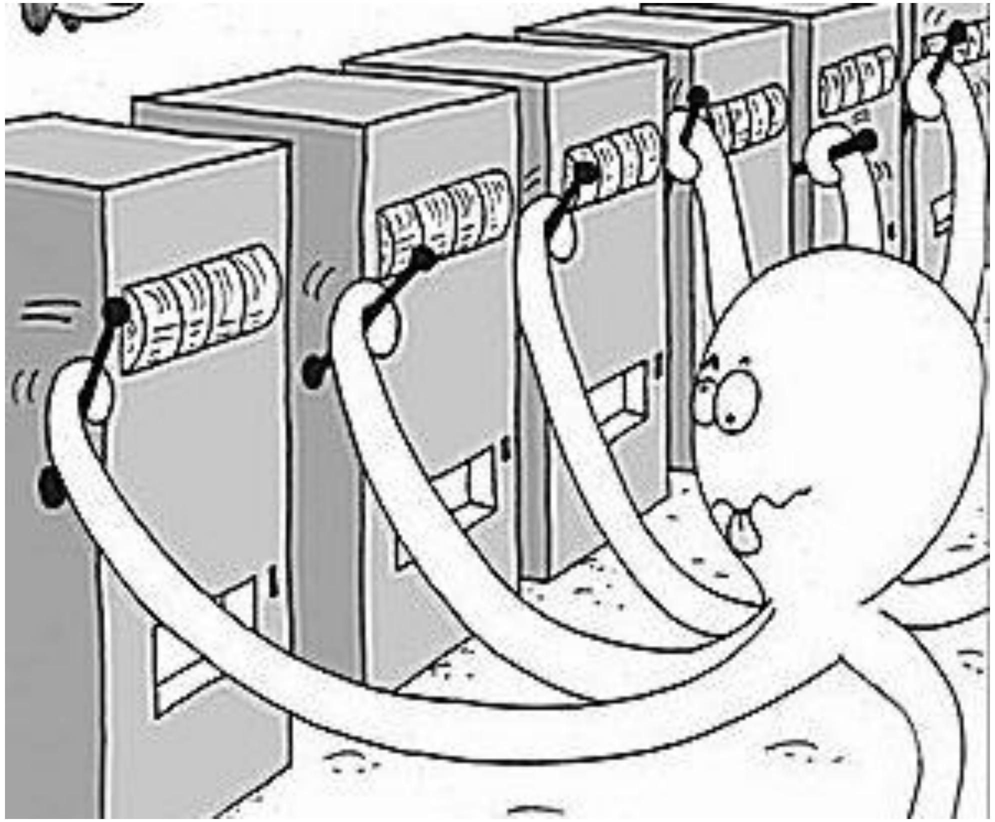
- We are able to learn the partners with high success probability
- We can use the learning algorithm for pre-selection of partners in ADMM optimization
- Code uploaded in this github link

- **Next steps:**
  - However, there is still room for improvements…but I'll not be here ☹

Thanks for your attention!

Multi-Armed Bandit

- Agent learns which arm returns the highest payoff



Real-world applications:

- Clinical trials
- Online Advertising
- Network routing

Applied RL agent

Algorithm 1: Applied RL agent

Initialize episodes $e \in \mathbb{E}$, steps $n \in \mathbb{N}$, actions $a_n \in \mathbb{A}$, states $s_n \in \mathbb{S}$, trading prosumers $j \in \omega_i$;

for *each episode* e **do**

  $E_i^e \leftarrow$ random sample between $[\underline{E}_i, \overline{E}_i]$;

  Initialize step $n \leftarrow 1$;

  while $E_i^e \geq s_n$ **do**

    Take action $a_n = j \in \omega_i$ using policy strategy;

    Observe $R_n(a_n)$, s_n;

    if $R_n(a_n) = 1$ **then**

      $E_{ij}^n \leftarrow$ energy offer by selected prosumer j;

    else

      $E_{ij}^n \leftarrow 0$;

    end

    $s_n \leftarrow \sum E_{ij}^n$, $R^{total} \leftarrow \sum R_n(a_n)$;

    Update the cumulative probability $\theta(a_n)$;

    $n \leftarrow n + 1$;

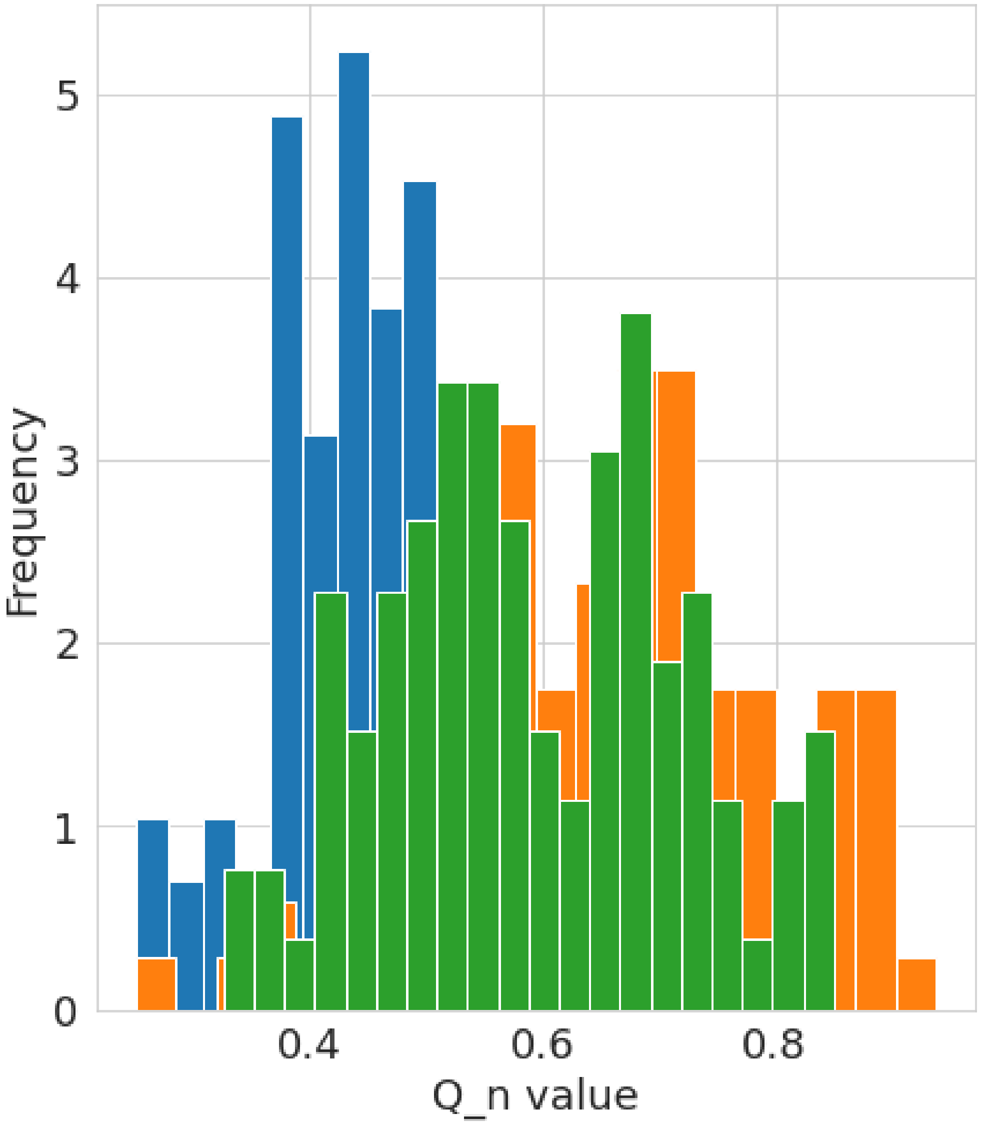
  end

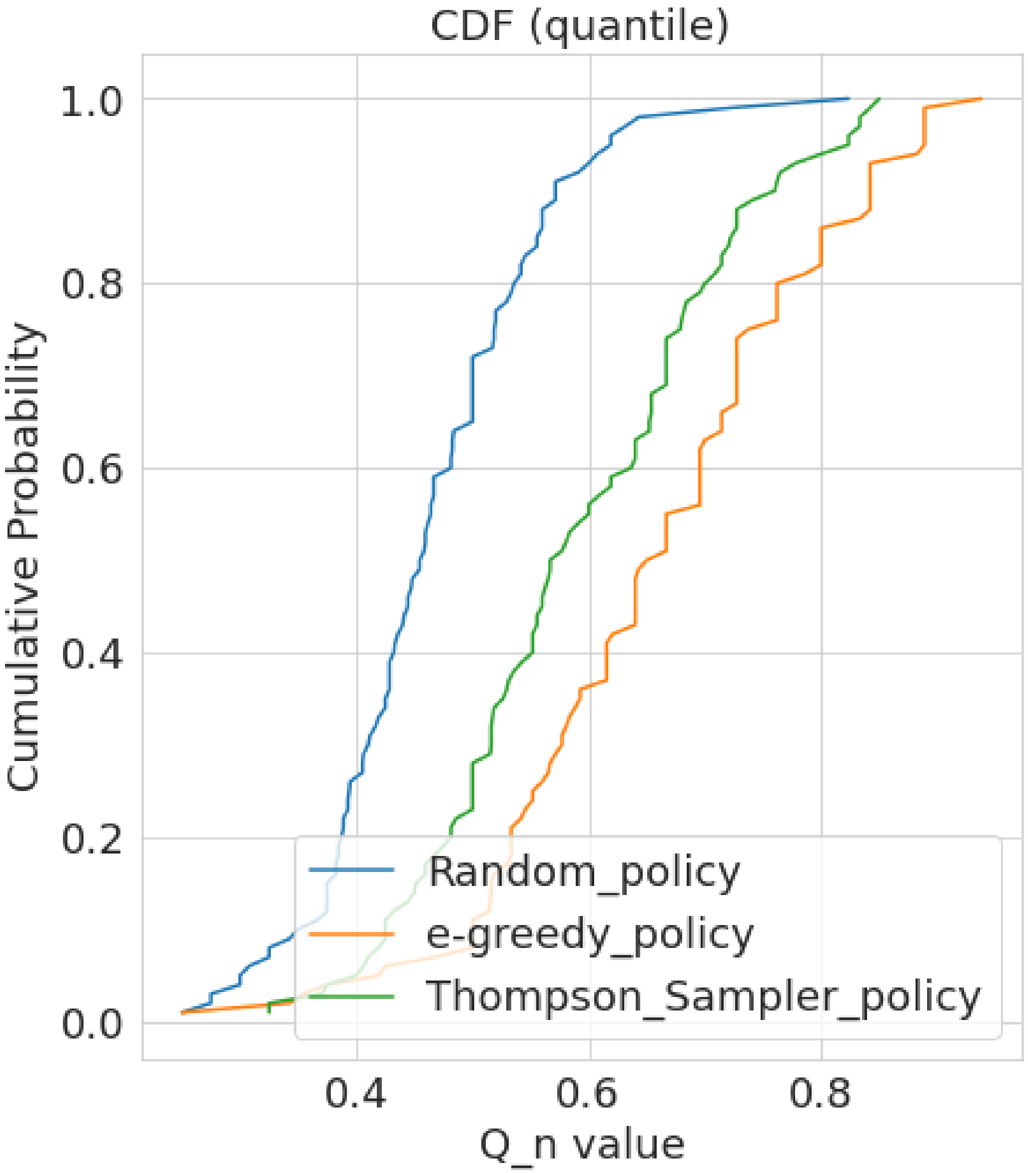
end

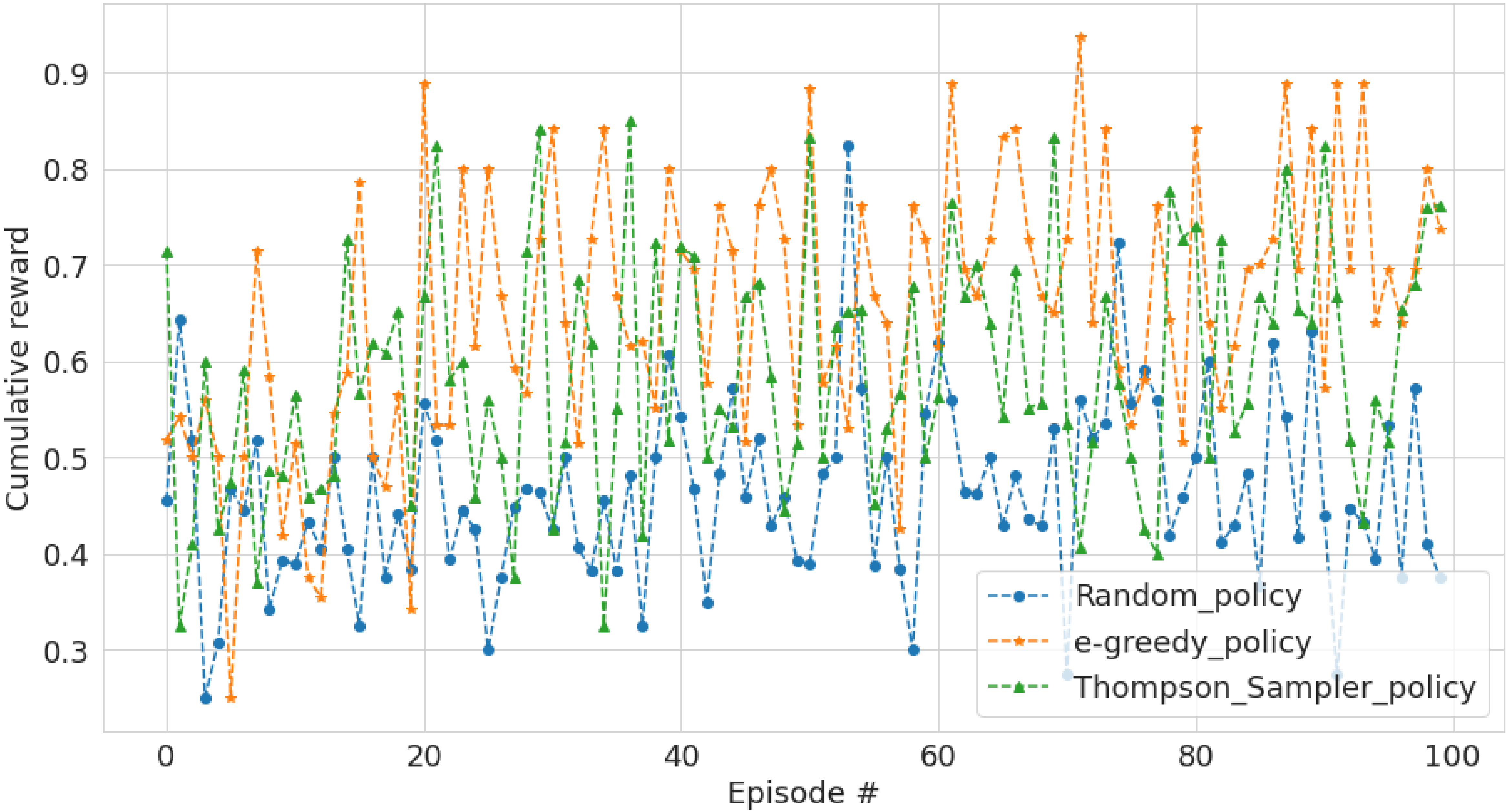
Total reward performance

Algorithm	$\overline{R}_{Total}$ $epi \in [1, 20]$	$\overline{R}_{Total}$ $epi \in [21, 100]$
Random	0.49	0.47
ϵ-greedy	0.50	0.67

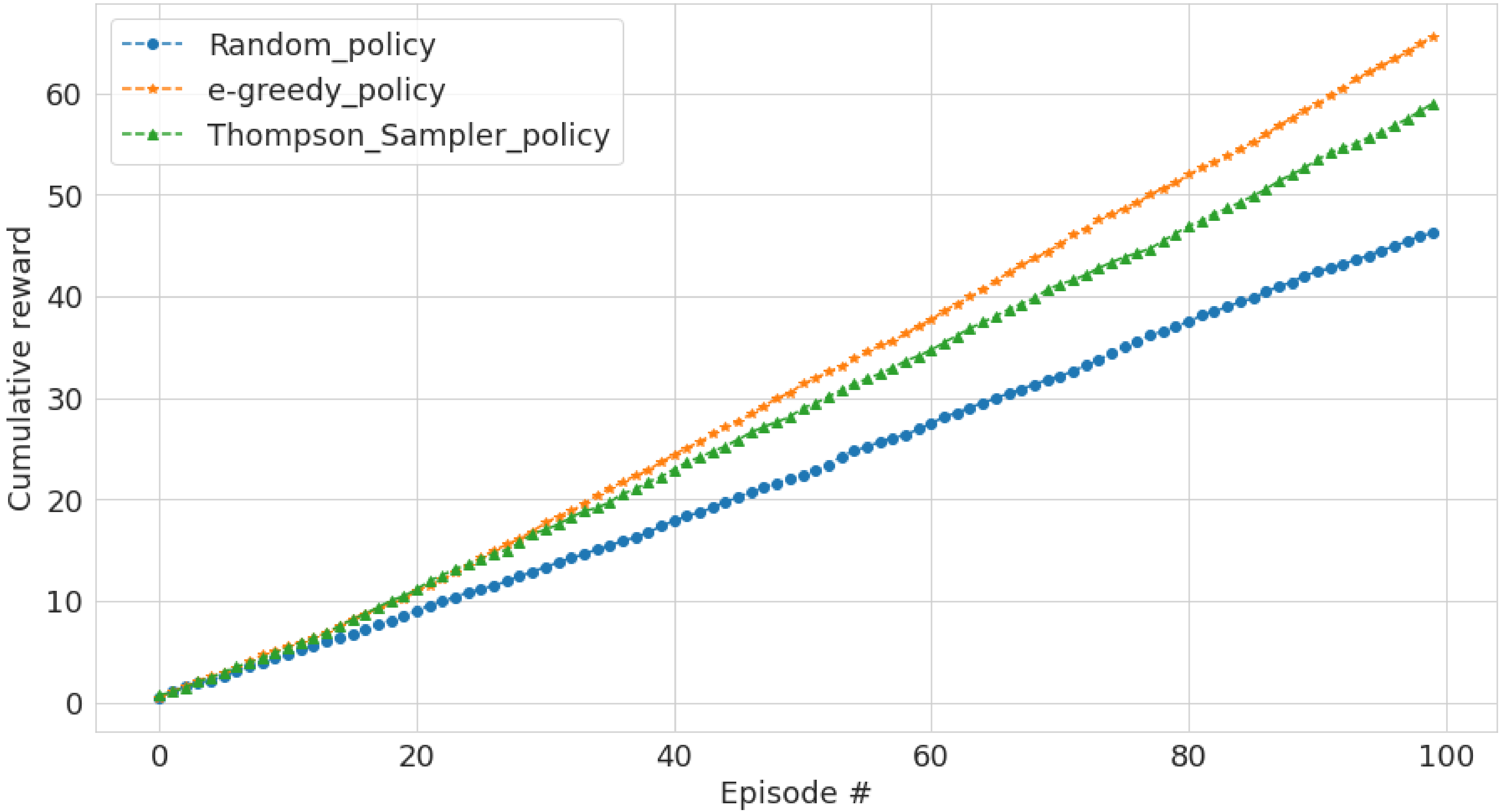
Empirical distribution function across 100 episodes







Cumulative reward
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
Episode #
Random_policy
e-greedy_policy
Thompson_Sampler_policy



Random_policy
e-greedy_policy
Thompson_Sampler_policy
Cumulative reward
Episode #
0
10
20
30
40
50
60
0
20
40
60
80
100