Rethink AI-based Power Grid Control: Diving Into Algorithm Design

Xiren Zhou, Siqi Wang, Ruisheng Diao, Desong Bian and Di Shi

Global Energy Internet Research Institute North America (GEIRINA)

NEURAL INFORMATION PROCESSING SYSTEMS

Abstract

Recently, deep reinforcement learning (DRL)-based approach has shown promise in solving complex decision and control problems in power engineering domain. We present an in-depth analysis of DRL-based voltage control from aspects of algorithm selection, state space representation, and reward engineering. To resolve observed issues, we propose a novel imitation learning-based approach to directly map power grid operating points to effective actions without any interim reinforcement learning process. The performance results demonstrate that the proposed approach has strong generalization ability with much less training time. The agent trained by imitation learning is effective and robust to solve voltage control problem and outperforms the former RL agents.

Problem formulation

We model the power grid control problem as an MDP as follows:

- Goal: maintain bus voltage values and line flows within predefined bounds.
- State: an infinite state space of continuous-valued state representation. Three types of measurement values can be adopted to construct state space: bus values (bus voltage V_m and bus angle V_a), branch values (line flow S_{line}), and generator values (active power P_g and reactive power Q_g).
- Action: control over voltage setpoints of all plants (continuous-valued).
- Reward function: we define our reward function R by dividing it up into two separate functions R_- and R_+, according to the types of a transition step (either successful or unsuccessful). i.e.,

$$r_t = R(s_t, a_t) = \begin{cases} R_-(s_{t+1}), \text{if } s_{t+1} \notin T \\ R_+(s_{t+1}), \text{if } s_{t+1} \in T \end{cases}$$

Reward design strategies

We proposed the following two general reward design strategies:

1. $R_-(s) = f_{\text{penalty}}(V_m,\ S_{line})$; $R_+(s)$ is a fixed non-negative constant.

2. CartPole-style reward: $R \equiv -1$.

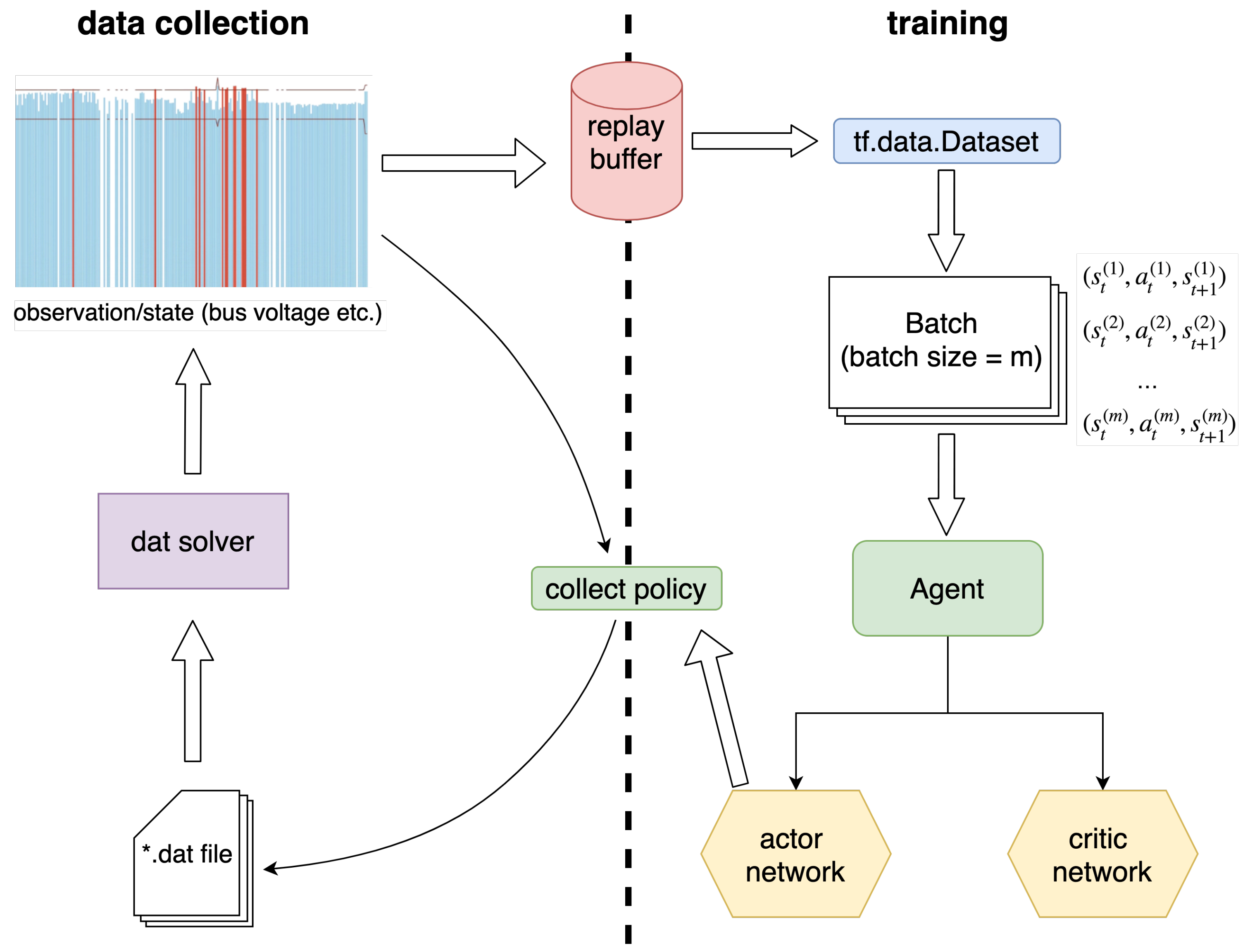
where $f_{\text{penalty}} = \alpha \sum_i \text{line\_overflow}[i] + \beta \sum_j \text{bus\_violation}[j]$ computes the penalty (negative reward) of a given state according to its S_{line} and V_m.

The overflow of the ith line and the voltage violation of the jth bus are defined as following:

$\text{line\_overflow}[i] = \max\{S_{line}[i] - \text{line\_limit}[i], 0\}^2$

$\text{bus\_violation}[j] = \max\{(V_m[j] - \text{bus\_lower\_limit}[j])(V_m[j] - \text{bus\_upper\_limit}[j]), 0\}$

RL-based power grid control


data collection
training
observation/state (bus voltage etc.)
replay buffer
tf.data.Dataset
Batch (batch size = m)
$(s_t^{(1)}, a_t^{(1)}, s_{t+1}^{(1)})$
$(s_t^{(2)}, a_t^{(2)}, s_{t+1}^{(2)})$
...
$(s_t^{(m)}, a_t^{(m)}, s_{t+1}^{(m)})$
dat solver
collect policy
Agent
*.dat file
actor network
critic network

Imitation learning-based power grid control

Algorithm 1 Imitation Learning for Training a Power Grid Control Agent

```
Initialize: policy network $\pi$ with random weights $\theta$
$D$ = COLLECT SUCCESSFUL STEPS($S_{\text{train}}$)
$l_{\text{objective}} = 1$ # set an objective average episode's length to terminate the runtime.
while True do
    Train $\theta$ with an optimizer (eg., Adam) on dataset $D$ for an epoch
    $l$ = EVALUATE EPISODE LENGTH($S_{\text{test}}$, $\pi_\theta$)
    if $l \leq l_{\text{objective}}$ then
        terminate

procedure COLLECT SUCCESSFUL STEPS($S$)
    Initialize $D = \emptyset$
    Set policy $\pi_{\text{collect}}$ to be an arbitrary policy (eg., random policy, trained SAC policy, etc.)
    $n$ = number of successful steps to collect (eg., 10000)
    $t_{\text{limit}} = 1000$ # set a horizon limit
    while $|D| < n$ do
        Randomly sample a state $s_0$ from $S$
        for $t = 0, 1, 2..., t_{\text{limit}} - 1$ do
            $a_t = \pi_{\text{collect}}(s_t)$
            $s_{t+1} \leftarrow$ perform $a_t$ on $s_t$
            if $s_{t+1} \in T$ then
                $D = D \cup \{(s_t, a_t)\}$
                break
    return $D$

procedure EVALUATE EPISODE LENGTH($S$, $\pi$)
    Initialize $L = \emptyset$
    $n_{\text{episodes}} = 50$ # set number of episodes(cases) to evaluate
    $t_{\text{limit}} = 50$ # set a horizon limit
    for $i = 1, 2, ..., n_{\text{episodes}}$ do
        Randomly sample a state $s_0$ from $S$
        for $t = 0, 1, 2..., t_{\text{limit}} - 1$ do
            $a_t = \pi(s_t)$
            $s_{t+1} \leftarrow$ perform $a_t$ on $s_t$
            if $s_{t+1} \in T$ or $t + 1 = t_{\text{limit}}$ then
                $L = L \cup \{t + 1\}$
                break
    return $\mathbb{E}_{l \in L}[l]$
```

Experiment results

Table 1: Training time needed til finding the optimal policy under different R_+ for strategy 1

R_+	0	1	10∼20	50	80∼1000
training steps	fail to converge	3.4k	1.3k	1k	0.9k

Table 2: Training time needed til finding the optimal policy under different R_+ for strategy 2

R_+	-1	0	1	1000
training steps	fail to converge	fail to converge	fail to converge	0.9k

- A higher positive reward makes the agent learn lessons more quickly and efficiently.
- A well-designed meaningful R_- does not play a such significant role as R_+.
- **The agent learns significantly from the last successful steps. Those unsuccessful steps give little helpful information to the RL agent. (this directly motivates the imitation learning-based method.)**

Table 3: Policy performance details

Policy	Number of unsolvable cases	Avg steps to solve a case
random policy	**train set: 127/9433(1.35%)** **test set: 13/1000(1.3%)**	train set: 11.53 test set: 11.98
SAC agent(normal distributed action)	train set: 138/9433(1.46%) test set: 16/1000(1.6%)	train set: 3.17 test set: 2.91
SAC agent(greedy action)	train set: 226/9433(2.40%) test set: 22/1000(2.2%)	**train set: 1** **test set: 1**
imitation learning agent	train set: 196/9433(2.08%) test set: 21/1000(2.1%)	**train set: 1** **test set: 1**

- Random policy has highest solvable rate since adding randomness can alleviate the partial observability problem.
- Imitation learning agent has strong generalization ability and outperforms SAC agent.

Conclusion

In this work, we revisited the previous DRL-based voltage control problem of power grid. We performed an in-depth analysis on algorithm selection, state space representation, and reward engineering. Based upon the analysis result, we realize that the agent mostly learn lessons from the positive rewards of the last successful steps. Thus, we optimize the reward design which results in a sample-efficient SAC-based approach that converges to the optimal policy very quickly. Furthermore, we proposed a novel imitation learning-based approach to perform power grid voltage control. The training and testing results show that the trained imitation learning agent has strong generalization ability which even outperforms the RL agent with the same policy network architecture. Meanwhile, the imitation learning based method does not involve any complex hyper-parameter tuning or design of a reward function, and requires less training time to converge to the optimal policy.