An intelligent power flow adjustment method based on reinforcement learning

L Ye¹, Z M Xiang¹, Y Yang¹, Y L Zhu²,4, C Zhang¹ and H F Yao³

¹State Grid Zhejiang Electric Power Co., Ltd.
²State Grid Zhejiang Taizhou Power Supply Co., Ltd.
³State Grid Zhejiang Shaoxing Power Supply Co., Ltd.

Abstract. In order to examine the future power grid operation, the operators usually build future power system model files first according to boundary conditions such as infrastructure plans, maintenance plans, power generation plans, load forecasts, and power plan requirements. As these plans are usually prepared independently, and the scale of the power grid is getting larger and larger, the newly built grid model is usually unsolved, ill, or does not meet the actual grid operation requirements. In this situation, the operators normally have to readjust the units power many times, according to their own experience, to obtain a qualified power flow result. A method for power flow adjustments is proposed in this paper. The Q-Learning (QL) method is used to adjust the power flow from nonconvergence to convergence, and then some direction guidance for subsequent adjustments is provided through certain safety and economic indicators of the power grid (such as network loss, output of the balancing machine, etc.). Finally, two examples of IEEE39 nodes system and Zhejiang power grid are presented to verify the effectiveness and practicability of the proposed method.

1. Introduction
In recent years, the structure of the power grid has become increasingly complex. The power flow calculation of the power system is not easy to converge. The most widely used method to obtain a convergence power flow is artificially adjusted based on experience, which is inefficient and ineffective[1]. The auto-adjustment of power flow can be discribed as an optimization problem. However, the optimization problem is so computationally intensive and complicated that traditional algorithms hardly meet the requirements. Therefore, there is an urgent need for an artificial intelligence method of power flow auto-adjustment.

The optimal power flow (OPF)[2] problems of power systems were initially solved by classical optimization methods such as Newton's method, interior point method, and quadratic programming. A Tabu search algorithm was proposed in [3], which effectively solved the optimal power flow problem, and the generation cost and the active power network loss of the system were minimized. In [4], a distributed parallel algorithm was proposed to speed up the optimization of large-scale system power flow. In [5], the original-dual interior point method was used to adjust the load of the unbalanced power flow nodes. These methods rely heavily on specific mathematical models and have certain limitations. With the rapid development and widespread application of artificial intelligence technology, researchers in the power field have also begun to study the application of artificial intelligence in power systems. At present, reinforcement learning (RL) is one of the important methods to realize the intelligentization of the power grid, and has shown good results in power flow optimization. A multi-step $Q(\lambda)$ online backtracking algorithm for solving the multi-objective optimal power flow problem was proposed in [6], which effectively improved the convergence speed of the algorithm. Distributed $Q(\lambda)$ backtracking algorithms were further proposed in [7] and [8] to solve the multi-objective optimal power flow in complex power grids. On this basis, this algorithm was applied in [9] to study the multi-
objective optimal carbon flow problem of the power grid. In [10], the distributed reinforcement learning algorithm was applied to solve the optimal power flow problem of isolated island microgrids, which has great flexibility and scalability. The optimization of power systems flow were all realized by adjusting the output of the units in [6]-[10].

In the actual power grid, when adjusting units, the personnel usually first adjust the start-stop status of a specific type of unit (such as thermal power, steam turbine), and use the typical output of the unit (such as full power or 0.9 power factor) as its active output value. Therefore, in order to ensure that the unit adjustment is in line with the rationality of the daily grid preparation business, the Q-Learning (QL) method is applied to select the best adjustable unit combination without adjusting the unit output. The power flow is first adjusted from nonconvergence to convergence to provide directional guidance for subsequent further adjustments. In this process, the purpose of the adjustment is only to achieve the convergence of the power flow calculation, and then to provide some directional guidance for the adjustment through certain safety and economic operation indicators of the power grid.

2. RL Algorithm Principle
Reinforcement learning is a good artificial intelligence method for solving decision problems. It mainly contains four elements: agent, environment, action, and reward [11]. The agent continuously explores in the environment, updates its behavior according to the rewards obtained, and finally finds the optimal strategy to achieve the goal with high returns. The principle of reinforcement learning is shown in figure 1.

![Figure 1. Principle of reinforcement learning algorithms.](image)

Reinforcement learning includes many algorithms, such as Sarsa, Temporal difference (TD), Q-Learning, and so on. The agent under the Q-Learning algorithm does not need to know the overall environment. It only needs to know the action space available in the current state to learn the global optimal strategy. In view of this, the Q-Learning method is used in this paper.

The Q-Learning method proposed in this paper is based on the principle of machine learning. It is different from traditional operations research, linear programming, nonlinear programming and other optimization models. Therefore, it does not require optimization models such as objective functions. Instead, through the black box test of the non-convergent grid model, the machine learning principle is used to autonomously learn the grid model adjustment scheme.

To apply the Q-Learning method to power flow adjustment, it only need to convert the variables in the power grid into states, actions, and rewards in reinforcement learning. The state can be the node voltage in the power grid, the output of the balancing machine, the output of the unit, etc. The action is the adjustment measures used in the grid, such as the output of the unit, the gear position of the transformer, and the number of switchable capacitor groups, etc. According to their own optimization goals, the reward can be the network loss, voltage, reactive power, balance machine output, unit output balance, and so on. In this paper, only the unit combination is changed by turning on and off the unit, so that the grid network loss is minimized on the premise that the power flow converges.

3. QL-based power flow adjustment algorithm
The Q-Learning algorithm needs to know the state space and action space, and the Q-Learning algorithm guides the agent's actions through a Q-value table. Therefore, in addition to constructing the state space and action space, an instant reward matrix R needs to be constructed to represent the reward and punishment values of the action selected in the current state S, and the Q value table is
calculated from the instant reward matrix $R$.

3.1. State space and action space
It is assumed that the output of the unit is unchanged, and each unit has only two states of starting and stopping, that is, the unit output is only the initial output or 0, so the state in the algorithm is the combination of all unit states:

$$S_i = \{P_{\text{Unit}1}, P_{\text{Unit}2}, \ldots, P_{\text{Unit}N}\}$$

(1)

$S_i$ is the i-th state.

The action space is as follows:

$$S = \{S_1, S_2, \ldots, S_N\}$$

(2)

$S$ is the state space.

When turning on/off of the unit is taken as an action, the action space is:

$$A = \{\text{On}_{\text{Unit}1}, \text{Off}_{\text{Unit}1}, \ldots, \text{Off}_{\text{Unit}N}\}$$

(3)

$A$ is the state space.

When the agent is in a certain state, selecting an action may not change the state. For example, Unit 1 itself is on. If the action of “turn on” is selected to Unit 1, then the state is unchanged and meaningless.

In order to reduce unnecessary repetitions and save running time, each time a state is reached, the same actions as the unit status in the state are removed from the action space, and the remaining actions are combined into an action subspace, and then actions only from the action subspace are selected.

3.2. Reward
The reward function is particularly important in the Q-Learning algorithm, which is a instant evaluation of the action selected in the current state. Setting rewards appropriately can help make decisions better and faster. The optimization goal of this paper is to adjust the power flow from nonconvergent to convergent, while minimizing the network loss of the power grid, so the reward is set as follows:

$$R = \begin{cases} 
\text{large penalty (e.g. } -999\text{), powerflow is not convergent} \\
\text{small penalty (e.g. } -1\text{), } P_{\text{balance unit}} \text{ exceeds upper and lower limits} \\
\text{large reward (e.g. } 100\text{), reach the goal} \\
\text{small reward (e.g. } 1\text{), load/loss decreases} \\
0, \text{ load/loss increases} 
\end{cases}$$

(4)

3.3. Parameters setting
In the Q-Learning algorithm, three parameters such as $\varepsilon$, $\alpha$, and $\gamma$ have a greater impact on the performance of the algorithm. Their functions and settings are as follows:

1) Positive number ($\varepsilon$): The probability of the $\varepsilon$-greedy strategy selecting an action. The agent has a probability of $1 - \varepsilon$ to choose the action according to the optimal value of the Q value table, and a probability of $\varepsilon$ to choose the action randomly. This is to make the agent jump out of a local optimum. In addition, all the values in the Q value table are 0 before the start. Therefore, $\varepsilon$ is set to a large value at the beginning that the probability of random selection of the action is enough large to make the agent be able to fully explore. As the number of iterations increases, $\varepsilon$ is gradually reduced.

2) Learning rate ($\alpha$): The smaller the learning rate is, the more the previous learning effect is retained, generally a smaller value is taken.

3) Discount factor ($\gamma$): The attenuation factor for future rewards. The larger $\gamma$ is, the more emphasis will be placed on past learning experiences, and the smaller $\gamma$ is, the more rewards the current action will be taken into account. Therefore, a value close to 1 is generally taken to $\gamma$. 
3.4. Unit combination optimization
The optimization process of the Q-Learning algorithm is to continuously explore and modify the strategy so as to learn the optimal strategy to achieve the goal. Through continuous learning to update the Q value, after reaching a certain learning times, look for the action that can get the maximum total rewards in the current state from the Q value table. The Q value is updated as follows:

\[
Q = (1-\alpha) \cdot Q(S,A) + \alpha \left[ R + \gamma \cdot \max_a Q(S',a) \right]
\]  

(5)

\(R\) is immediate reward. \(\max_a Q(S',a)\) is the optimal value of all actions in the next state in the previous learning process.

If the agent has previously benefited from a certain action in the next state, when the choice is faced again next time, the correct action can be directly selected to enter this beneficial state.

The flow of Q-Learning algorithm to solve the optimal unit combination is shown in figure 2.

![Figure 2. Process of unit combination optimization.](image)

In order to make the agent learn sufficiently, it is neede that set the appropriate number of learning times to avoid falling into the local optimum due to too few learning times. First, read the initial state of the power grid, select a unit from the action subspace to operate according to the selection \(\epsilon\)-greedy strategy, and perform power flow calculations. Based on the results of the power flow calculations, get a reward in accordance with equation (4), then update the Q value at the corresponding position in the
Q value table according to equation (5). Once the termination state is reached, the learning of this Episode is ended, and the learning of the next Episode is resumed after returning to the initial state. As the number of learning times increases, the value of the Q-value table gradually converges during continuous updating. At this time, the start and stop of the unit is selected according to the optimal value of the Q-value table. The unit combination can be quickly adjusted so that the power flow of the power grid is adjusted from nonconvergent to convergent.

4. Example analysis

Based on the above algorithm, the systems of 10 units and 39 nodes and Zhejiang power grid are taken as two examples. The Q-Learning method is used to select the most effective startup combination of the adjustable units.

4.1. IEEE39 nodes system

In the 39-nodes system with 10 units, the balancing machine unit is unit No.10, and the maximum active output is 572.8MW. Since the output of the balancing machine will be adjusted by itself, it is not viewed as a control variable. If the on/off action of all the remaining 9 units are used as actions, then there are 18 actions. In addition, considering the states of the nine units as a whole as a state in the algorithm, there are 512 states in state space. Thus, the Q-value table is a matrix which rows are actions and columns are states. The size of the Q matrix is $(512, 9)$ dimensions.

First the load factor is adjusted to 0.5, and the network loss before adjustment is 151.129MW, which accounts for 2.76% of the load. After several simulation tests, the settings of each parameter are shown in table 1.

| Positive number $\varepsilon$ | Discount factor $\gamma$ | Learning rate $\alpha$ |
|--------------------------------|--------------------------|-----------------------|
| $\sqrt{\text{Episode}+1}$     | 0.97                     | 0.01                  |

Under the parameter settings as shown in table 1, the simulation result is shown in figure 3. In order to illustrate the impact on the performance of the algorithm, $\varepsilon$ is set to a fixed value of 0.2, and the simulation results is shown in figure 4.

Figure 3. Percentage change of network loss to load (\(\varepsilon\) attenuating).

Figure 4. Percentage change of network loss to load (\(\varepsilon\) not attenuating).

It can be seen from figure 3, the percentage of the network loss to load in each Episode is not stable at the beginning. As the number of learning times increases, the percentage of the network loss to load begins to converge to 0.49% when more than 7000 Episodes. This is because that it is the learning process at the beginning, the value in the $\varepsilon$-greedy strategy is relatively large. In this case, the actions are randomly selected, and a lot of explorations are made. After training to a certain degree, a certain rule has been learned, and then $\varepsilon$ is gradually reduced. In the subsequent episode, the unit can be
adjusted according to the learned rule to reach the goal quickly. That can be easily observed from figure 4 that the percentage of the network loss to load cannot converge when \( \epsilon \) is a fixed value. This is because there is still a lot of randomness in selecting the action if \( \epsilon \) has not been reduced after learning a certain number of times. Then the optimal value of the Q-value table is not selected, that is, the optimal action is missed. Deviations in the adjustment path result in a large number of adjustments. It can be observed that \( \epsilon \) needs to decrease as the number of learning times increases as shown in table 1.

In order to observe the convergence of the number of adjustments, a small upper limit is set for the number of adjustments of each Episode, and the result is shown in figure 5.

As can be seen from figure 5, the number of adjustments finally converged to three. As the number of learning times increases, the Q-value table is gradually improved. After the Q-value table is updated, it can be quickly adjusted according to the Q-value table. The adjustment path of the units is shown in table 2.

| Step | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 | Unit 7 | Unit 8 | Unit 9 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0    | 250    | 650    | 632    | 508    | 650    | 560    | 540    | 830    | 1000   |
| 1    | 250    | 650    | 632    | 508    | 650    | 560    | 540    | 0      | 1000   |
| 2    | 250    | 650    | 0      | 508    | 650    | 560    | 540    | 0      | 1000   |
| 3    | 250    | 650    | 0      | 508    | 650    | 0      | 540    | 0      | 1000   |

Due to the limited space and the large space of the Q matrix, in order to explain the change of the Q value, the Q \((S, a)\) value at the corresponding position is found according to the adjustment path in table 2. Besides, the change track of the Q value is shown in figure 6.

It can be viewed from figure 6 that the Q values also converge accordingly. It shows that as the number of learning times increases, the Q value table is updated. After a certain number of learning times is reached, the Q value corresponding to each (state, action) in the Q value table no longer
changes. Then, every time a state is reached, only need to select the optimal value in this state in the Q value table to quickly reach the goal. Some indicators such as the network loss equivalents before and after the adjustment are shown in table 3.

**Table 3.** Comparison of indicators before and after adjustment.

| Indicators                        | Comparison              |
|----------------------------------|-------------------------|
|                                  | Before adjustment | After adjustment     |
| Network loss (MW)                | 151.129               | 51.1344              |
| Percentage of load to loss (%)   | 2.76                  | 0.49                 |

It can be seen from table 3 that the grid network loss was reduced from 151.129MW before adjustment to 51.1344MW, and the percentage of the network loss to load was also reduced from the original 2.76% to 0.49%.

In addition, under the same parameter settings, the load factors are adjusted to 0.2 and 0.7 respectively for simulation, and the algorithm can also achieve convergence. The process is the same as above, and will not be expanded one by one here. When the load factor is 0.2, the network loss decreases from 88.216MW to 6.55098MW, and the percentage of load decreases from 1.62% to 0.30%. The number of adjustments finally converges to 14 times. When the load factor is 0.7, the network loss is reduced from 82.8362MW to 27.9267MW, the percentage of the load is also reduced from 1.50% to 0.60%, and the number of adjustments finally converges to 2 times.

4.2. Zhejiang power grid system

In order to verify the practicability of this algorithm, 9 adjustable generating units from Zhejiang Power Grid which load 51152.8MW are simulated. The results are shown in figure 7 and table 4. It can be seen from figure 7 that after the learning process is finished, the power flow can be adjusted to the ideal situation through 8 times.

**Figure 7.** Change of adjustment times.

**Table 4.** Comparison of indicators before and after adjustment.

| Indicators                        | Comparison            |
|----------------------------------|-----------------------|
|                                  | Before adjustment | After adjustment |
| Power Flow is convergent or not  | No                    | Yes                |
| Network loss (MW)                | /                     | 371.363             |
| Percentage of load to loss (%)   | /                     | 0.73               |

It can be seen from table 4 that the power flow does not converge before adjustment. After the adjustment, the active power loss of the power grid was 371.363MW, accounting for only 0.73% of
the load. Therefore, this algorithm effectively adjusts the nonconvergent power flow to convergence, and the network loss is the smallest of all the startup combinations, which improves the power utilization rate.

5. Conclusion
The Q-Learning method is used in this paper to obtain a unit combination to make nonconvergence power flow converge and the network loss is incidentally reduced. Two simulations are demonstrated to verify that the power flow can be adjusted quickly, accurately and intelligently under the proposed algorithm.

Furthermore, DQN can be used to solve the problems of dimensional disasters. And combined with deep learning (such as CNN), the nonconvergence power flow can be adjusted to convergence by adjusting the output of the adjustable unit, and at the same time, different goals in the power grid can be achieved.

6. References
[1] Yang Y D, Song A J, Liu. H, Qin Z J, Deng J and Qi J J 2018 Parallel computing of multi-contingency optimal power flow with transient stability constraints Protection and Control of Modern Power Systems 3 pp 1-20
[2] Gao H J, L. J Y, Shen X D and Xu R 2017 Optimal power flow research in active distribution network and its application examples Procs. of the CSEE 37 pp 1634-44
[3] Wu H B and Cai L 2003 Optimal power flow of distribution network with distributed generation based on modified Tabu search algorithm Power System Protection and Control 27 pp 23-7
[4] Cheng X G, Li J W, Cao. L X and Liu X L 2003 Pareto of optimization on electricity price discount considering demand price elasticity Automation of Electric Power Systems 27 pp 23-7
[5] Tao H, Bu G Q, Wang H F, Bao W and Guo R P 2014 An optimization method based on weighted least absolute value to restore power flow solvability of bulk power system Automation of Electric Power Systems 38 pp 60-4
[6] Yu T, Hu X B and Liu J 2010 Multi-objective optimal power flow calculation based on multi-step \( Q(\lambda) \) learning algorithm Transactions of China Electrotechnical Society 38 pp 139-45
[7] Yu T, Liu J and Hu X B 2012 Optimal power flow for complex grid using distributed multi-step backtracking \( Q(\lambda) \) learning Transactions of China Electrotechnical Society 27 pp 185-192
[8] Yu T, Liu J, Chan K W and Wang J J 2012 Distributed multi-step \( Q(\lambda) \) learning for optimal power flow of large-scale power grids International Journal of Electrical Power and Energy Systems 42 pp 614-20
[9] Zhang X S, Zheng L M and Yu T 2014 Multi-objective optimal carbon emission flow of power grid based on multi-step \( Q(\lambda) \) learning algorithm Automation of Electric Power Systems 38 pp 118-123
[10] Eleonora R S, Maria L D S, Salvatore F, Ninh N Q and Quyn T A multi-agent system reinforcement learning based optimal power flow for islanded microgrids IEEE 16th International Conference on Environment & Electrical Engineering pp 552-7
[11] W. Christopher, D. Peter 1992 Technical note: Q-Learning Machine Learning 8 pp 279-92

Acknowledgments
This paper is supported by the project of Research on Technology of Intelligent Operation Mode of Power Grid Based on Deep Learning in Electricity Market(5211TZ170006).