Solving Dynamic Distribution Network Reconfiguration using Deep Reinforcement Learning

[1]Ognjen B. Kundačina, [1,3]Predrag M. Vidović, [2]Milan R. Petković,
[1]University of Novi Sad, Faculty of Technical Sciences, Serbia,
[2]Schneider Electric DMS NS Novi Sad, Serbia
[3]Corresponding author, e-mail: pvidovic@uns.ac.rs, phone/fax: +381-21-455-865

Abstract – Distribution Network Reconfiguration (DNR), as a part of the Distribution Management System, plays an important role in increasing the energy efficiency of the distribution network by coordinating the operations of the switches in the distribution network. Dynamic Distribution Network Reconfiguration (DDNR), enabled by the sufficient number of remote switching devices in the distribution network, attempts to find the optimal topologies of the distribution network over the specified time interval. This paper proposes data-driven DDNR based on Deep Reinforcement Learning (DRL). DRL based DDNR controller aims to minimize the objective function, i.e. active energy losses and the cost of switching manipulations while satisfying the constraints. The following constraints are considered: allowed bus voltages, allowed line apparent powers, a radial network configuration with all buses being supplied, and the maximal allowed number of switching operations. This optimization problem is modelled as a Markov Decision Process (MDP) by defining the possible states and actions of the DDNR agent (controller) and rewards that lead the agent to minimize the objective function while satisfying the constraints. Switching operation constraints are modelled by modifying the action space definition instead of including the additional penalty term in the reward function, to increase the computational efficiency. The proposed algorithm was tested on three test examples: small benchmark network, real-life large-scale test system and IEEE 33-bus radial system and the results confirmed the robustness and scalability of the proposed algorithm.

Keywords: Distribution network, Network reconfiguration, Deep reinforcement learning, Active energy loss and switching operations minimization, Multi-objective function.

NOMENCLATURE

A. Variables

$T_{\int ¿^{i}¿}$	– Duration of the considered time interval, in hours,
P,Q	– Active and reactive powers, respectively,
S^{sw}	– Apparent power of switch $sw(sw=1,2,\ldots,N_{SW}$, where N_{SW} is the number of switches in the network)
P_{Loss}	– Active power losses,
U,θ	– Bus voltage magnitude and phase angle, respectively,
x	– Switch status (1 – closed switch, 0 – opened switch),
y	– Change of switch status (1 – status change, 0 – no status change),
α	– Auxiliary variable,
$p\left(s^{'},r\middle|s,a\right)$	– Transition function,
$S_t,s^{'},s_t$	– Random variable denoting the new state in timestep t and its particular values, respectively,
S_{t-1},s,s_{t-1}	– Random variable denoting the previous state in timestep $t-1$ and its particular values, respectively,
R_t,r,r_t	– Random variable denoting the immediate reward in timestep t and its particular values, respectively,
A_{t-1},a,a_{t-1}	– Random variable denoting the action in timestep $t-1$ and its particular values, respectively,
$a^{'},a_t$	– Action in timestep t,
G_t	– Random variable denoting the return in timestep t,
π	– Policy,
q^{π}	– Action value function (Q-function) related to the policy π,
$Q\left(s^{'},a^{'};\theta^{Q}\right)$	– Deep Q-network,
θ^{Q}	– Deep Q-network parameters,

Q_{label}	– Label for Deep Q-network training,
$Q_{target}\left(s',a';\theta^{Q_{target}}\right)$	– Target Deep Q-network,
$\theta^{Q_{target}}$	– Target Deep Q-network parameters,
L	– Squared error loss function.

B. *Indices*

t	– Discrete time interval index ($t=1,2,\ldots,T$, where T is the total number of time intervals),
j,k	– Bus ($j,k=1,2,\ldots,N_N$, where N_N is the total number of buses in a distribution network),
b	– Branch ($b=1,2,\ldots,N_{Br}$, where N_{Br} is the total number of branches),
sw	– Switch ($sw=1,2,\ldots,N_{SW}$, where N_{SW} is the number of switches in a network),
$episode$	– Episode index.

C. *Sets*

S	– Finite set of states,
A	– Finite set of actions,
R	– Finite set of immediate rewards.

D. *Parameters*

g,b	– Element of nodal conductance and susceptance matrices, respectively,
N_{SW}^{max}	– Maximum number of allowed switch operations during an optimization time interval,
R,X	– Branch resistance and reactance, respectively,
S_b^{max}	– Maximum allowed VA power flow in b^{th}branch,
U_j^{min},U_j^{max}	– Minimum and maximum allowed kV voltage at j^{th} bus,
N	– Number of episodes,
T	– Number of time intervals, i.e. number of timesteps per episode,
CLoss	– Cost of energy losses, in \$ per kWh,
CSWs	– Cost of switching action for the s^{th} switch, in \$,
C_U	– Penalty value if the bus voltage constraint is violated in any of the nodes,
C_S	– Penalty value if the branch capacity constraint is violated for any of the branches,
γ	– Discount factor,
ε	– Exploration hyperparameter,
α	– Learning rate hyperparameter for Q-learning.

E. *Abbreviations*

DNR	– Distribution Network Reconfiguration,
DDNR	– Dynamic Distribution Network Reconfiguration,
RL	– Reinforcement Learning,
MDP	– Markov Decision Process,
DRL	– Deep Reinforcement Learning,
DQN	– Deep Q-network,
ReLU	– Rectified Linear Unit.

1. INTRODUCTION

Distribution network reconfiguration is a widely used power application in distribution system operation. Firstly, DNR is used to achieve objectives such as the minimization of power loss and voltage deviations [1, 2], and secondly, it is used for load balancing, Volt/Var optimization, supply restoration, etc. [3]. Therefore, DNR is an important tool in distribution systems for optimal distribution network management. Two types of DNR are static DNR, which is determined for the fixed operation point, and DDNR, which optimizes the network operations over the specified time period. Therefore, DDNR is suitable for real-time applications because of the intermittency of load, generation, and other network conditions.

First studies regarding DNR tackle problems like power loss reduction [4-6], load balancing [6], energy loss reduction [7], and operation cost reduction [8], in the scope of static DNR. Static DNR including mixed-integer linear programming is presented in [9-13], with the main advantage of finding the global optimum. Heuristic algorithms are, also, static DNR. The main drawback of those applications is the possibility to find only the local optimum. Representative metaheuristic algorithms used for solving the DNR problem are genetic algorithm [13, 14], evolutionary algorithms [15], particle swarm optimization [16], and others.

Using DDNR rather than static DNR can result in larger benefits and increased performance of network operation, with the drawback of requiring the fully automated distribution network to be used. DDNR studies for small distribution networks are presented in [17, 18], both of which have the problem of computation time increasing significantly with the increase of the problem dimension. The "path-to-node" concept is proposed in [19], for the efficient modelling of the radiality of the distribution network. The main drawback of this concept is the significant increase in the number of decision variables with the increase of the optimization problem dimension. DNR can be performed on an hourly, daily, or monthly basis, as in [20–22]; however, these references do not deal with the limits of the number of switching operations in mentioned time periods. DNR studies for annual network reconfiguration which consider the variable loads are presented in [23, 24], where the study [23] additionally deals with minimizing the cost of switching operations, while the study [24] considers the stochastic power generation of distributed generators. Refs. [24, 25] attempt to solve DDNR with the use of genetic algorithm. Per hour DNR solved by rule-based algorithm is presented in [26].

Ref. [27] presents a simple and fast strategy for selecting the candidate solutions of the distribution system reconfiguration problem without solving optimization based or load flow based programs. Dynamic and multi-objective DNR by using the parallel processing method and adaptive population approach is presented in [28]. Ref. [29] proposed single- and multi-objective formulation of DDNR based on the Lagrange relaxation approach. In single-objective formulation, the objective function is loss reduction, while in multi-objective formulation, the objective function is the minimization of costs of energy losses, network reliability and switching operations. Study [30] presents DNR for loss reduction without the network parameter information. DNR is formulated as a Markov Decision Process and solved using an off-policy Reinforcement Learning algorithm trained on historical operation data set. Modern distribution network is being transformed from passive to active due to emerging renewable energy resources and, consequently, using distributed generators for the DNR problem is becoming a study of importance [2, 31]. Study [32] presented DNR for power loss reduction with budget limit as a hard constraint for the planning purposes of distribution networks.

This paper proposes DDNR based on Reinforcement Learning algorithm. It is a multi-objective approach which minimizes the cost of energy losses and switching manipulations. The proposed expression of the DDNR problem in the RL framework, that is, the definition of the state variables leads to lower observability requirements compared to the approach proposed in [30]. The amount of information needed for the algorithm execution is decreased, since the topology information and the information about the power flows in the network are compressed in the single set of variables. This reduces the amount of telemetered measurements needed for the potential real-world execution of the algorithm and it is also convenient from the algorithm training perspective, since the required size of the neural network is reduced. The paper also proposes the way of considering switching operation constraints that improves the algorithm training computational efficiency. The proposed approach assumes selecting the actions from the available subset of the action set, which is updated during the episode so that switching operation constraints are not violated, rather than allowing these actions, yet penalizing them with the large amount of negative reward. This way of selecting actions can be used for optimization problem constraints whose violation can be detected without the feedback from the environment (by evaluating only the agent's action) and it can be applied to similar power system control and optimization problems treated with RL like Volt-Var optimization, energy storage scheduling, supply restauration, etc. Total cost benefits and execution times of the proposed algorithm are compared with the state of the art method from [29]. The main contributions of the paper are:

1. Proposed DDNR is based on a Machine Learning method called Reinforcement Learning,
2. Suggested multi-objective and scalable approach is computationally efficient during the algorithm execution, with the expense of high computation cost during the algorithm training,
3. Proposed definition of the state variables for the RL agent, which leads to lower observability requirements,
4. Proposed, computationally efficient way of considering switching operation constraints by creating the available subset of the action set and updating it during the episode.

The rest of the paper is organized in the following way: In section 2 the main idea of RL is presented, while in section 3 network reconfiguration problem is formulated. Expression of the DDNR problem in the RL framework is described in section 4. Results and discussion, as well as conclusion, are presented in sections 5 and 6, respectively. In section 7, references are presented.

2. MATHEMATICAL BACKGROUND – REINFORCEMENT LEARNING

Reinforcement Learning, as a machine learning technique, deals with how software agents learn to take actions in an environment by experience and exploration, with the goal of finding the optimal strategy that maximizes the obtained long-term reward [33]. In the RL framework, it is assumed that the agent interacts with the generally stochastic environment in discrete timesteps. At the beginning of each timestep, the agent observes the environment, i.e. it receives the state variables from the environment. Based on the state variables, the agent takes an action and sends it to the environment, which changes its state, due to the action received, as well as its internal processes. The environment sends the immediate reward signal for that timestep and the state variables for the next timestep to the agent. The goal of an RL algorithm is to find the (close to) optimal policy, i.e. the action selection that maximizes the long-term reward, while receiving feedback about its immediate performance. This section presents the theoretical background of RL in 2.1 and 2.2 and the Deep Q-learning algorithm in 2.3, which is used in the remaining of the paper.

2.1. Finite Markov Decision Processes

The RL problem is mapped onto an MDP, which is defined as the following tuple:

1. S – finite set of states,
2. A – finite set of actions,
3. R – finite set of immediate rewards,
4. $p(s', r | s, a) = Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$ – transition function,

where the random variables S_t, S_{t-1}, R_t, A_{t-1} symbolize the new state, previous state, received immediate reward and action being taken, respectively, and $s', s \in S$, $r \in R$, $a \in A$ denote the particular values of these random variables. Therefore, the transition function represents the probability of being in the state s' and receiving the immediate reward r, on condition to being in the state s previously and executing the action a.

Return in timestep t, which represents the long-term reward starting from that timestep, is the subject of optimization:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}, \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor.

Policy π models the action selection in various states. By optimizing the policy, the long-term reward is optimized. The policy is in the general case stochastic, i.e. it maps the probability distribution of actions to states.

Action value function (Q-function) is a mapping of states-action pairs to the real numbers, where the value of the state-action pair represents the expected long-term reward starting from that state, taking that action, and following a concrete policy π afterwards.

$$q^{\pi}(s,a) = E_{\pi}[G_t | S_t = s, A_t = a] = E_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \vee S_t = s, A_t = a\right]. \tag{2}$$

The Q-function can be recursively expressed using the Bellman equations:

$$\begin{aligned} q^{\pi}(s,a) &= E_{\pi}\left[R_{t+1} + \gamma q^{\pi}(S_{t+1}, A_{t+1}) | S_t = s\right] \\ ¿ r &+ \gamma \sum_{s' \in S} p(s' | s, a) \sum_{a' \epsilon A} \pi(a' | s') q^{\pi}(s', a') \end{aligned} \tag{3}$$

Finding the optimal action value function implies finding the optimal policy. The optimal policy is generated from the optimal Q-function by selecting the action with the largest Q-function value in each state. If the problem is formulated as an MDP, then at least one optimal solution exists and an iterative procedure based on the Bellman equations which converges to one of those solutions can be established.

In order to find the exact solution to the MDP, it has to be fully defined, i.e. all of the transition probabilities and immediate rewards has to be known. Additionally, for large state and action spaces, solving MDPs exactly could be computationally infeasible. The main RL idea is to overcome these problems by learning the (close to) optimal policies based on the interaction of the agent with the environment.

2.2. Q-learning

This paper considers model-free RL algorithms, where the optimal policy is learned directly from the accumulated experience, i.e. the history of interaction of the agent with the environment. On the other hand, in the model-based RL the MDP, or the transition probabilities and immediate rewards for all state-action-next state triplets are learned from the accumulated experience and solved to obtain the policy.

Q-learning is a basic model-free RL algorithm, where the Q-function values for each state-action pair are stored in the Q-table and updated during the algorithm training [34]. During the execution of the algorithm, for each state the agent receives from the environment, the action with the greatest Q-function value in the table is selected.
During the algorithm training, actions are selected randomly with the probability ε, and the actions with the largest Q-function values for the corresponding states are selected with the probability $1-\varepsilon$, where $\varepsilon \in [0,1]$ is the exploration hyperparameter. This way the agent searches the state-action space and avoids the local optima problem. Algorithm training is performed by repeating the predefined number of episodes, which consist of timesteps. One timestep contains the information about one interaction of the agent with the environment: current state, action selected by the agent, received reward and the next state. The length of episodes, i.e. the number of timesteps in them is generally variable.

Upon one interaction of the agent with the environment, the values in the Q-table are updated using the following rule, obtained using the idea from the Bellman equations:

$$q(s,a) := (1-\alpha)\,q(s,a) + \alpha\left[r + \gamma \max_{a'} q(s',a')\right], \tag{4}$$

where α is the learning rate hyperparameter.

As well as MDPs, the Q-learning algorithm assumes discrete state and action spaces and for large state and action spaces it may be infeasible to learn the Q-function value for all state-action pairs.

2.3. Deep Q-learning

Deep Q-learning algorithm is one of the basic Deep Reinforcement Learning (DRL) algorithms, that utilizes the advance in deep learning field to improve the RL algorithms. The idea of the algorithm is to use a deep neural network, also called Deep Q-Network (DQN), as an approximator of the Q-function [35]. Inputs of the DQN are state variables, while output neurons provide the Q-function values for each of the actions and for the input state. Therefore, the state space can be continuous, whereas the action space has to be discrete and finite, since the number of output neurons is limited.

Deep Q-learning algorithm introduces target DQN which has the same topology as DQN and in which the parameters of DQN are copied at the predefined period during the training process. Target DQN is used for determining the labels for the DQN training, which is one of the key advantages over the Q-learning algorithm, since it improves the stability of the learning process [35].

Another important feature of DRL algorithms is the experience replay memory, in which the history of agent's interaction with the environment is stored. At each timestep, tuple $(s,a,r,s')$is stored into the experience replay memory, from which the minibatches for DQN training are sampled.

Labels for DQN training are calculated in the following way, using the idea from the Bellman equations, similarly to the Q-learning algorithm:

$$Q_{label}(s,a) = r + \gamma \max_{a'} Q_{target}\left(s',a';\theta^{Q_{target}}\right). \tag{5}$$

θ^{Q} and $\theta^{Q_{target}}$ denote the parameters (weights and biases) of DQN and target DQN. DQN is trained using the minibatch gradient descent algorithm, which minimizes the squared error loss function that expresses the distance between the labels and the outputs of DQN during the training:

$$L\left(\theta^{Q}\right)=E[(Q_{label}(s,a)-Q(s,a;\theta^{Q}))^{2}]. \quad (6)$$

3. DYNAMIC DISTRIBUTION NETWORK RECONFIGURATION PROBLEM FORMULATION

This section provides a standard formulation of the DDNR problem, which will be transformed into the Reinforcement Learning formulation of the same problem in section 4. DDNR mathematical model consists of multi-objective function and constraints. Multi-objective function defined as the total cost of active energy loss and manipulation of switching devices is minimized, subject to the following constraints: active and reactive powers injection constraints, bus voltages constraints, branch capacity constraints, switching operations constraints and network radiality constraint.

DDNR problem formulation will be written in the following way [11, 29]:

Objective function:

$$min\{\sum_{t=1}^{T} \mathrm{C}_{\mathrm{Loss}} T_{\int ¿^{t} P^{t}_{Loss}} + \mathrm{C}_{\mathrm{SW}\,s} SW^{t}\} ¿. \quad (7)$$

Active power losses and switching action are defined in the following way:

$$P^{t}_{\mathrm{Loss}}=\sum_{b=1}^{N_{\mathrm{Br}}} \alpha_{b} R_{b} \frac{(P^{t}_{b})^{2}+(Q^{t}_{b})^{2}}{(U^{t}_{j})^{2}}, \quad (8)$$

$$SW^{t}=\sum_{sw=1}^{N_{\mathrm{SW}}} y^{t}_{sw}, \quad (9)$$

where:

$$P^{t}_{b}=\{g_{jk}(U^{t}_{j})^{2}-U^{t}_{j}U^{t}_{k}[g_{jk}\cos(\theta^{t}_{j}-\theta^{t}_{k})+b_{jk}\sin(\theta^{t}_{j}-\theta^{t}_{k})]\}, \quad (10)$$

$$Q^{t}_{b}=\{-b_{jk}(U^{t}_{j})^{2}+U^{t}_{j}U^{t}_{k}[b_{jk}\cos(\theta^{t}_{j}-\theta^{t}_{k})-g_{jk}\sin(\theta^{t}_{j}-\theta^{t}_{k})]\}. \quad (11)$$

a^{t}_{b}– is defined in the following way:

$$\alpha^{t}_{b}=\begin{cases} x^{t}_{sw} \text{ branch } b \text{ has a switch;} \\ 1 \text{ branch } b \text{ does not have a switch,} \end{cases} \quad (12)$$

$$x^{t}_{sw}=\begin{cases} 1 \text{ switch } sw \text{ is closed in interval } t; \\ 0 \text{ switch } sw \text{ is opened in interval } t. \end{cases} \quad (13)$$

Constraints are:

1. Active and reactive injection constraints $(k=1,2,\ldots,N_{N}, b=j-k)$:

$$P^{t}_{j}=U^{t}_{j}\sum_{k=1}^{N_{\mathrm{N}}} U^{t}_{k}[g_{jk}\cos(\theta^{t}_{j}-\theta^{t}_{k})+b_{jk}\sin(\theta^{t}_{j}-\theta^{t}_{k})], \quad (14)$$

$$Q^{t}_{j}=U^{t}_{j}\sum_{k=1}^{N_{\mathrm{N}}} U^{t}_{k}[g_{jk}\sin(\theta^{t}_{j}-\theta^{t}_{k})-b_{jk} ¿\cos(\theta^{t}_{j}-\theta^{t}_{k})]. ¿ \quad (15)$$

2. Bus voltage constraints $(j=1,2,\ldots,N_{N})$:

$$U^{min}_{j}\le U^{t}_{j}\le U^{max}_{j}. \quad (16)$$

3. Branch capacity constraints $(b=1,2,\ldots,N_{Br})$:

$$(P^{t}_{b})^{2}+(Q^{t}_{b})^{2}\le(S^{max}_{b})^{2}. \quad (17)$$

4. Switching operation constraints $(sw=1,2,\ldots,N_{SW})$:

$$\sum_{t=1}^{T}\left|x^{h}-x^{h-1}\right|\le N^{max}_{\mathrm{SW}},\ x^{0} \text{is known from the base case.} \quad (18)$$

This equation is described in detail in [29].

5. Radiality network constraint:

$$\sum_{b=1}^{N_{\mathrm{Br}}} \alpha_b^t = N_{\mathrm{N}} - 1. \tag{19}$$

The proposed multi-objective DDNR problem formulation (7) aims to minimize the total cost of active energy loss and manipulation of switching devices. However, in many practical applications the DDNR, apart from minimizing the cost of active energy loss and manipulation of switching devices, is used to improve voltage deviation [1, 2], load balancing, Volt/Var optimization, supply restoration [3], the distribution network reliability [29], etc. Extension of the DDNR problem formulation assumes incorporating some of these criteria into the objective function. That is achieved by adding the corresponding terms to the objective function while preserving the constraints (14) – (19).

4. DYNAMIC DISTRIBUTION NETWORK RECONFIGURATION BASED ON REINFORCEMENT LEARNING

This section describes the way DDNR is expressed as an RL problem, how the objective function and constraints are considered, and the description of the algorithm training procedure.

4.1. Modelling Dynamic Distribution Network Reconfiguration as a Markov Decision Process

The information flow between the agent and the environment during their interaction is presented in Fig. 1. Episodes consist of T timesteps and, at the beginning of each timestep, active and reactive power consumption data for the next hour is loaded. Then, the Load Flow calculation is executed to create the state variables, which contain the timestep index and the apparent powers of all switches in the network, as shown in (20):

$$s_t = \left(t, S^1, S^2, \ldots, S^{N_{Sw}}\right). \tag{20}$$

This choice of state variables reduces the state space dimensionality and DQN size, since the current network configuration (switch statuses and hence the network topology can be reconstructed using the apparent powers of the switches) and Load Flow results are compressed in a single set of variables, from which the agent can make its own representation of the environment and correlate it with actions and received rewards.

Action space contains all the switch combinations that lead the network in feasible radial configuration, in which all the buses are energized. These switch combinations are enumerated in a unique way so that one output neuron corresponds to one feasible radial combination. This action space definition implies that the radiality network constraint stated in (19) is always satisfied, which accelerates the learning process.

The reward value for each timestep t is equal to the negative sum of the following terms:

1. Active energy losses price $\mathrm{C}_{\mathrm{Loss}} T^t P_{Loss}^t$, used to model the first term in DDNR objective function stated in (7).
2. Price of the switching manipulations needed to conduct the network from the previous configuration to the current one $\mathrm{C}_{\mathrm{SW}\,s} SW^t$, used to model the second term in DDNR objective function stated in (7).
3. Predefined penalty value C_{U} if the bus voltage constraint is violated in any of the nodes, used to model the bus voltage constraints in (16).
4. Predefined penalty value C_{S} if the branch capacity constraint is violated for any of the branches, used to model the branch capacity constraints in (17).

As an alternative to adding the predefined penalty to the reward function if the number of switch manipulations exceeds the predefined limit for any of the switches, we propose the following way to consider the switching operation constraints in (18):

- The subset of available actions is created at the beginning of each episode and it is initially equal to the action set (i.e. all the switch combinations that lead the network into a feasible radial configuration, in which all the buses are energized).

- During the episode, the number of operations of each switch in that episode is updated. Prior to action selection, actions that would violate the switching operation constraints if selected are removed from the subset of available actions.
- In each timestep, the action with the largest Q-value is selected from the subset of available actions, instead of from the action set.

This approach increases the training efficiency compared to the approach that would penalize the exceeded number of switch operations, since the computational effort to learn policies that do not select actions that violate the switching operation constraints is not needed, because those actions cannot be chosen in our approach.
It is convenient to consider the switching operation constraints using this approach, since only the selected action (switch combination) is needed to be known, to conclude if the constraint is violated. The subset of available actions can be determined without executing the actions. Bus voltage and branch capacity constraints cannot be modelled using this approach generally, since the action has to be executed and the feedback from the environment is required for any conclusions about the constraint violations.

Active and reactive injection constraints in (14) and (15) are satisfied in the Load Flow calculation results and therefore are not considered in the reward function. Values of the state variables and rewards are normalized, for the purpose of improving the neural network training.

4.2. Training algorithm

During the algorithm training N episodes are repeated, with each episode consisting of a predefined number of timesteps T, where in each timestep the interaction between the agent and the environment takes place, as described in 4.1. The variety of training scenarios is created by sampling the daily load curves randomly from some predefined distribution. The way the exploration hyperparameter ε is updated during the training can also be tuned. In this paper, we used the linear decrease of ε until the $0.8\,N^{th}$ episode, and the constant value afterwards. A detailed representation of the agent training procedure is displayed in Algorithm 1.

Algorithm 1: Deep Q-Network training

```
Initialize the Deep Q-network's $Q(s \mid \theta^{Q})$ parameters randomly
Initialize the target Deep Q-network's $Q'(s \mid \theta^{Q_{target}})$ parameters using
  the original network's parameters
Initialize the experience replay buffer
for $episode = 1, 2, \ldots, N$ do
    Sample daily load curves randomly
    Initialize $\epsilon$
    Initialize the subset of available actions to the action set
    Run the initial Load Flow calculation
    Send the initial state $s_1$ to the agent
    for $t = 1, 2, \ldots, T$ do
        rand = random number between 0 and 1
        if $rand > \epsilon$ then
            Select the action $a$ with the largest Q-value in the subset of
              available actions
        else
            Select random action $a$ from the subset of available actions
        end
        Update the subset of available actions
        Update the network configuration according to $a_t$
        Run the Load Flow calculation
        Collect the immediate reward $r_t$ and the next state $s_{t+1}$ data
        Store tuple $(s_t, a_t, r_t, s_{t+1})$ in the experience replay buffer
        Sample the minibatch of tuples from the experience replay buffer
        Create labels for Deep Q-Network training using (5)
        Update Deep Q-Network parameters by minimizing the loss
          function given in (6)
        Set loads for the next time instant for each bus
    end
    if update target network period then
        $\theta^{Q_{target}} = \theta^{Q}$
    Update $\epsilon$
end
```

Once trained, the algorithm execution reduces to $T+1$ Load Flow calculations and T neural network evaluations, which are almost instantaneous, resulting in computationally efficient control procedure, which can be used either standalone, or as a part of the more complex power systems application.

5. RESULTS AND DISCUSSION

In this section, results and discussion are presented for Benchmark test examples – 5.1, real-life large-scale distribution network – 5.2 and IEEE 33-bus radial system – 5.3, along with the choice of RL and Deep Learning hyperparameters. Proposed algorithms were implemented in Python, deep neural networks were modelled and trained using the PyTorch Deep Learning framework, and power system related modelling and calculations were completed using OpenDSSDirect, the Python interface to OpenDSS distribution system simulation software [36]. The algorithms were executed on a 64-bit Windows 10 with the following hardware configuration: AMD A8-6410 APU with AMD Radeon R5 Graphics 2.00 GHz, 4 cores and 8GB of RAM.

5.1. Benchmark test examples

Fig. 2 illustrates a 15-bus test benchmark where a slack bus is the bus with the marker 0 and the other 14 buses are of the PQ type. Loads are defined by the chronological daily diagram – represented by the full lines in Fig. 3 and peak load (1MVA). The length of all branches is the same (4.5 km). All branches are balanced with the direct sequence impedance $r+jx=(0.224+j0.109)\,\Omega/km$. All branches have switching devices. Number of switch manipulations is not constrained.

The cost of energy losses and switching operations in the time interval are [23, 37]:
– Cost of energy losses (C_{Loss}): $6.5625 cents/kWh;
– Cost of switching manipulations (C_{sw}): $1 per manipulations.

Penalty values used to model the bus voltage constraints in (16) and the branch capacity constraints in (17) are: $\mathrm{C_U}=\mathrm{C_S}=10$.

DQN used for this test example consists of the input layer, four hidden layers and the output layer. The input layer has 15 neurons, one for the timestep index variable and 14 for the apparent powers of each switch. Each hidden layer has 256 neurons and the output layer consists of 186 neurons, one for each switch combination that leads to feasible radial configuration. Rectified Linear Unit (ReLU) activation function is applied on each of the hidden layers and the output layer. The neural network is trained using the Adam optimizer, with the learning rate of 0.00001 and the minibatch size of 128. Adding batch normalization on several hidden layers was tried, but it neither improved nor deteriorated the training results.

The algorithm was trained on 60000 episodes. During the training, loads for each hour were uniformly sampled from the intervals defined by dashed curves in Fig. 3. The used target DQN update frequency is 10 episodes and the minibatches for DQN training were sampled from experience replay memory, which has the capacity of 1000000 samples. The initial value of the exploration parameter ε is 1 and it decreases linearly to the episode index, until it reaches the value 0.1 in 48000th episode. The value of the discount factor γ is 0.99, as in [35].

Fig 4. presents the average value of DQN loss function defined in (6), over the episode. As the training advances, the DQN loss decreases, which implies that the Q-function is being approximated successfully. Additionally, the testing performance of the RL algorithm with increasing training episodes is presented in Fig. 4 by displaying the amount of the normalized received reward per episode and its moving average. These rewards were obtained by executing the episodes with the exploration parameter ε equal to zero, after each training episode. The training showed asymptotic convergence within 20000 episodes.

Table 1 presents load, losses, and switch manipulations for the proposed approach in 24-hour time optimization, as well as the comparison of the same results with the state of the art method from [29]. The method from [29] minimizes the costs of energy losses, switching manipulations, and outages. For this comparison, the method from [29] is executed with the cost of outages is set to zero. Graphical representation of switch manipulations are presented in Fig. 5. The final results are additionally compared with the method from [19], which is presented in Table 2. By comparing the results of the proposed algorithm and the method from [29], it can be concluded that the proposed switching actions are not the same, but the total costs only differ slightly. The execution time of the proposed algorithm is 0.148s, which is two orders of magnitude smaller than the execution time of the method from [29].

Table 1 – Total load, active power losses and switch manipulations in the 24-hour time optimization period (O–open; C–close)

		Proposed approach		Method from [29]	
Hour	Load [kW]	Losses [kW]	Manipulation of switches	Losses [kW]	Manipulation of switches
1	4554.0	114.06	4(O), 14(C)	131.73	No manipulations
2	4305.0	101.79	No manipulations	115.32	No manipulations
3	3876.0	82.49	No manipulations	89.97	No manipulations
4	3326.0	61.20	No manipulations	62.95	No manipulations
5	3205.0	57.03	No manipulations	57.81	No manipulations
6	3693.0	75.01	No manipulations	80.35	No manipulations
7	7542.0	316.63	10(O), 14(O), 4(C), 13(C)	316.63	10(O), 13(C)

8	7909.0	348.71	No manipulations	348.71	No manipulations
9	8279.0	384.08	No manipulations	384.08	No manipulations
10	6067.0	213.94	10(C), 13(O)	220.01	No manipulations
11	9369.0	465.82	No manipulations	465.82	10(C), 13(O)
12	9054.0	440.61	No manipulations	440.61	No manipulations
13	8823.0	423.20	No manipulations	423.20	No manipulations
14	8823.0	423.20	No manipulations	423.20	No manipulations
15	7670.0	375.02	No manipulations	324.18	10(O), 13(C)
16	9801.0	493.87	No manipulations	493.87	10(C), 13(O)
17	7440.0	278.42	4(O), 14(C)	278.42	4(O), 14(C)
18	7049.0	252.96	No manipulations	252.96	No manipulations
19	7317.0	276.55	No manipulations	276.55	No manipulations
20	6643.0	227.18	No manipulations	227.18	No manipulations
21	5651.0	178.27	No manipulations	178.27	No manipulations
22	5163.0	147.68	No manipulations	147.68	No manipulations
23	4793.0	126.72	No manipulations	126.72	No manipulations
24	4554.0	114.06	No manipulations	114.06	No manipulations

Table 2 – Total losses, number of switch manipulations and total cost in the 24-hour time optimization period

	Proposed approach	Method from [29]	Method from [19]
Total losses [kW]	5978.48	5980.28	5967.56
Number of switch manipulations	10	10	22
Total cost [$]	402.3	402.4	413.6

Based on Table 2, it can be concluded that the proposed approach and the method from [29] provide better solutions than the method from [19]. Solutions from the proposed approach and the method from [29] are less expensive and they are achieved with the lower number of switch manipulations.

Total losses for 24 hours for 15-bus test benchmark without DNR are 6477.81 kW, with the total cost of $425.1. The proposed dynamic DNR losses are reduced to 5978.48 kW and the total cost is reduced to $402.3. Loss reduction per hours, for proposed approach and method from [29], is presented in Fig. 6.

In Fig. 6 it can be seen that there is no reduction of losses in the interval from 10 to 16 hours. In Table 1 it can be seen that the network configuration from 10 to 16 hours is the same as the basic network configuration, as in Fig. 1 (in the first hour switch 4 was opened and switch 14 was closed; in the seventh hour switch 4 was closed and switch 14 was opened and switch 10 was opened and switch 13 was closed; in tenth hour switch 10 was closed and switch 13 was opened). Consequently, there is no reduction of losses from 10 to 16 hours. For the method from [29] in the 10th hour, there is a negative loss reduction (see Table 1). From 17 to 24 hours, loss reduction is the same for both methods, because network configuration in that interval is the same, too.

Fig. 7 shows the voltage profile before and after the reconfiguration. It is noted that the voltage profile is improved after applying the configuration found by the proposed algorithm. Bus loads are taken from the daily load profile presented in Fig. 3 in the 17th hour.

Graphical representation of switch manipulations, when the number of switch manipulations is constrained on two, is presented in Fig. 7. Losses cost is $403.5, while the switching manipulation cost is $12, summing up to the total cost of $415.5, meaning that this solution is more expensive than the solution presented in Table 1.

5.2. Real-life large-scale distribution network

The considered real-life large-scale radial distribution network consists of 4 feeders, 1015 branches and 1008 loads. This network is used for the presentation of both the efficiency and robustness of the developed DDNR. The distribution network is equipped with 31 remotely controlled switches, where 24 of them are normally closed and 7 are normally open. This is a challenging number of remotely controlled switches, with the purpose of showing the scalability of the developed algorithm.

DQN used for this test example consists of the input layer, four hidden layers and the output layer. The input layer has 32 neurons, one for the timestep index variable and 31 for the apparent powers of each switch. Each hidden layer has 1024 neurons and the output layer consists of 3567 neurons, one for each switch combination that leads to a feasible radial configuration. The algorithm was trained on 100000 episodes. The initial value of the exploration parameter ε is 1 and it decreases linearly to the episode index, until it reaches the value 0.1 in 80000th episode. Rest of the parameters are the same as in the case of the 15-bus test benchmark network.

Table 3 presents losses and switch manipulations for the proposed approach in 24-hour time optimization. The execution time of the proposed algorithm is 3.635s, which is two orders of magnitude smaller than the execution time of the method from [29] when used for networks with 31 remotely controlled switches. By analyzing the testing performance of the RL algorithm with increasing training episodes, the training showed asymptotic convergence within 30000 episodes.

Table 3 – Active power losses and switch manipulations in the 24-hour time optimization period (O–open; C–close)

Hour	Losses [kW]	Manipulation of switches
1	182.30	238(O), 900 (O), 994(O), 1011 (C), 1013(C), 1014 (C)
2	160.98	No manipulations
3	127.19	No manipulations
4	92.46	No manipulations
5	86.08	No manipulations
6	196.43	No manipulations
7	742.05	695(O), 742(O), 900 (C), 1015 (C)
8	783.53	No manipulations
9	826.54	No manipulations
10	368.21	947 (O), 1014 (O), 695 (C), 994 (C)
11	1008.27	No manipulations
12	965.66	No manipulations
13	932.39	No manipulations
14	891.17	1011 (O), 1013 (O), 238 (C), 947 (C)
15	746.81	No manipulations
16	1044.24	No manipulations
17	786.83	947 (O), 1013 (C)
18	553.33	No manipulations
19	744.87	No manipulations
20	645.53	1015 (O), 1014 (C)
21	509.13	191 (O), 1010 (C)
22	455.83	No manipulations
23	313.87	No manipulations
24	178.78	No manipulations

Based on Table 3, total power losses are 13342.49 kW. The cost is \$875.6[1]; on the other hand, switching manipulations cost \$24. Total money losses, using the proposed DDNR, are \$899.6. Those results demonstrate that the proposed DDNR is applicable to real-life large-scale distribution networks.

5.3 IEEE 33-bus radial system

In this section, a numerical test is performed by using the IEEE 33-bus radial system [38] – Fig. 8. Branch resistances, reactances, as well as the peak loads are taken from [38]. Loads between buses 2 and 18 are scaled with the pink full line in Fig. 3, loads between buses 19 and 25 are scaled with the green full line in Fig. 3, and loads between buses 26 and 33 are scaled with the blue full line in Fig. 3. This network is modified to match the developed DDNR. The IEEE33 test system is equipped with 20 remotely controlled switches, where 15 of them are normally closed and 5 are normally open, as shown in Fig. 8. Remotely controlled switches are marked with "*s*" and the unique index. If the remotely controlled switch is placed on the full line it is closed, otherwise, it is open. Five lines were added to the original scheme and they are presented with dashed lines in Fig. 8.

The algorithm was trained on 60000 episodes with the same training parameters as in the case of the 15-bus test benchmark network. Table 4 presents losses and switch manipulations for the proposed approach in the 24-hour time optimization period. The execution time of the algorithm is 0.272 s.

[1] Costs of energy losses and switching manipilations are the same as in section 5.1.

Table 4 – Active power losses and switch manipulations in the 24-hour time optimization period (O–open; C–close)

Hour	Losses [kW]	Manipulation of switches
1	34.87	6 (O), 35 (C)
2	30.41	No manipulations
3	23.50	No manipulations
4	16.03	No manipulations
5	14.60	No manipulations
6	20.85	No manipulations
7	56.49	No manipulations
8	62.11	8 (O), 33 (C)
9	70.44	No manipulations
10	50.05	No manipulations
11	95.42	14 (O), 34 (C)
12	87.05	No manipulations
13	81.22	No manipulations
14	81.22	No manipulations
15	57.06	No manipulations
16	108.47	No manipulations
17	86.87	27 (O), 37 (C)
18	75.20	No manipulations
19	81.43	No manipulations
20	65.80	No manipulations
21	47.51	No manipulations
22	38.90	No manipulations
23	33.00	No manipulations
24	29.45	No manipulations

Based on Table 4, total power losses are 1347.95 kW. The cost is \$88.5[2] , while the switching manipulations cost equals \$8. Total money losses, using the proposed DDNR, are \$96.5. By analyzing the testing performance of the RL algorithm with increasing training episodes, the training showed asymptotic convergence within 15000 episodes.

Fig. 10 shows the voltage profile before and after the reconfiguration. It is noted that in most of the buses, the voltage is enhanced after applying the configuration found by the proposed algorithm. Bus loads are taken from the daily load profile presented in Fig. 3 in the 17th hour.

6. CONCLUSION

This paper presents a multi-objective formulation for DDNR, which minimizes the total cost of energy losses and switching operations. The proposed DDNR problem solution is based on DRL. It has been displayed that the suggested definition of state variables, which leads to lower observability requirements, can be used for treating the DDNR problem in the RL framework successfully. Additionally, the proposed, computationally efficient way of considering switching operation constraints by creating the available subset of the action set and updating it during the episode also turned out viable for this problem. The ideas from suggested way of modelling the DDNR problem as MDP can be used for solving similar power system control and optimization problems as well. Once trained, the RL algorithm demonstrates faster execution compared to the state of the art method, while yielding approximately equal total cost savings.

The presented results indicate that the developed algorithm is scalable, i.e. the computation time does not increase exponentially with the problem dimension. Therefore, the developed DDNR algorithm is capable of handling the real-life large-scale distribution networks.

7. REFERENCES

[2] Costs of energy losses and switching manipilations are the same as in section 5.1.

1. Samman MA, Mokhlis H, Mansor NN, Mohamad H, Suyono H, Sapari NM. Fast Optimal Network Reconfiguration With Guided Initialization Based on a Simplified Network Approach. IEEE Access 2020; 8:11948–11963.
2. Fathi V, Seyedi H, Ivatloo BM. Reconfiguration of distribution systems in the presence of distributed generation considering protective constraints and uncertainties. International Transactions on Electrical Energy Systems 2020; 1–25.
3. Mishra S, Das D, Paul S. A comprehensive review on power distribution network reconfiguration. Energy Systems 2017; 8:227–284.
4. Civanlar S, Grainger JJ, Yin H, Lee SSH. Distribution feeder reconfiguration for loss reduction. IEEE Transactions on Power Delivery 1988; 3:1217–1223.
5. Shirmohamadi D, Hong HW. Reconfiguration of electric distribution networks for resistive line loss reduction. IEEE Transactions on Power Delivery 1989; 2:1492–1498.
6. Baran M, Wu FF. Network reconfiguration in distribution systems for loss reduction and load balancing. IEEE Transactions on Power Delivery 1989; 2:1401–1407.
7. Taleski R, Rajicic D. Distribution network reconfiguration for energy loss reduction. IEEE Transactions on Power Systems 1997; 1:398–406.
8. Zhou Q, Shirmohammadi D, Liu WHE. Distribution feeder reconfiguration for operation cost reduction. IEEE Transactions on Power Systems 1997; 2:730–735.
9. Borghetti A. A mixed-integer linear programming approach for the computation of the minimum-losses radial configuration of electrical distribution networks. IEEE Transactions on Power Systems 2012; 3:1264–1273.
10. Ahmadi H, Marti JR. Distribution system optimization based on a linear power flow formulation. IEEE Transactions on Power Delivery 2015; 1:25–33.
11. Lavorato M, Franko JF, Rider MJ, Romero R. Imposing radiality constraints in distribution system optimization problems. IEEE Transactions on Power Systems 2012; 1:172–180.
12. Jabr RA, Singh R, Pal BC. Minimum loss network reconfiguration using mixed integer convex programming. IEEE Transactions on Power Systems 2012; 2:1106–1115.
13. Haghighat H, Zeng B. Distribution system reconfiguration under uncertain load and renewable generation. IEEE Transactions on Power Systems 2016; 4:2666–2675.
14. Gupta N, Swarnkar A, Niazi KR. Distribution network reconfiguration for power quality and reliability improvement using genetic algorithms. International Journal of Electrical Power & Energy Systems 2014; 54:664–671.
15. Tsai M, Hsu F. Application of grey correlation analysis in evolutionary programming for distribution system feeder reconfiguration. IEEE Transactions on Power Systems 2010; 2:1126–1133.
16. Wu W, Tsai M. Application of enhanced integer coded particle swarm optimization for distribution system feeder reconfiguration. IEEE Transactions on Power Systems 2011; 3:1591–1599.
17. Golshannavaz S, Afsharnia S, Aminifar F. Smart distribution grid: optimal dayahead scheduling with reconfigurable topology. IEEE Transactions Smart Grid 2014; 5:2402–2411.
18. Li Z, Chen X, Yu K, Zhao B, Liu H. A novel approach for dynamic reconfiguration of the distribution network via multi-agent system. In: Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies 2008; 1305–1311.
19. Ramos ER, Exposito AG, Santos JR, Iborra FL. Path-based distribution network modeling: application to reconfiguration for loss reduction. IEEE Transactions on Power Systems 2005; 2:556–564.
20. Broadwater RP, Khan AH, Shaalan HE, Lee RE. Time varying load analysis to reduce distribution losses through reconfiguration. IEEE Transactions on Power Systems 1993; 1:294–300.
21. Chen CS, Cho MY. Energy loss reduction by critical switches. IEEE Transactions on Power Delivery 1993; 3:1246–1253.
22. López E, Opazo H, García L, Bastard P. On line reconfiguration considering variability demand: applications to real networks. IEEE Transactions on Power Systems 2004; 1:549–553.
23. Shariatkhah MH, Haghifam MR, Salehi J, Moser A. Duration based reconfiguration of electric distribution networks using dynamic programming and harmony search algorithm. International Journal of Electrical Power & Energy Systems 2012; 1:1–10.
24. Zidan A, El-Saadany EF. Distribution system reconfiguration for energy loss reduction considering the variability of load and local renewable generation. Energy 2013: 698–707.

25. Milani AE, Haghifam MR. An evolutionary approach for optimal time interval determination in distribution network reconfiguration under variable load. Mathematical and Computer Modelling 2013; 1–2:68–77.
26. Mazza A, Chicco G, Andrei H, Rubino M. Determination of the relevant periods for intraday distribution system minimum loss reconfiguration. International Transactions on Electrical Energy Systems 2015; 10:1992–2023.
27. Karimianfard H, Haghighat H. An initial-point strategy for optimizing distribution system reconfiguration. Electric Power Systems Research 2019; 176:105943–105950.
28. Jafari A, Ganjehlou HG, Darbandi FB, Mohammdi-Ivatloo B, Abapour M. Dynamic and multi-objective reconfiguration of distribution network using a novel hybrid algorithm with parallel processing capability. Applied Soft Computing Journal 2020; 90:106146–106165.
29. Kovački NV, Vidović PM, Sarić AT. Scalable algorithm for the dynamic reconfiguration of the distribution network using the Lagrange relaxation approach. International Journal of Electrical Power & Energy Systems 2018; 94:188–202.
30. Gao Y, Shi J, Wang W, Yu N. Dynamic Distribution Network Reconfiguration Using Reinforcement Learning. IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) 2019; 1–7.
31. Liu Z, Liu Y, Qu G, Wang X, Wang X. Intra-Day Dynamic Network Reconfiguration Based on Probability Analysis Considering the Deployment of Remote Control Switches. IEEE Access 2019; 7:145272–145281.
32. Ahmadi I, Ahmadigorji M, Tohidifar E. A novel approach for power loss reduction in distribution networks considering budget constraint. International Transactions on Electrical Energy Systems 2018; 12:1–27.
33. Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press, 2018.
34. Watkins CJ, Dayan P. Q-learning. Machine learning 1992; 3–4:279–292.
35. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al. Human-level control through deep reinforcement learning. Nature 2015; 7540:529–533.
36. Dugan R, McDermott T. An open source platform for collaborating on smart grid research. Power and Energy Society General Meeting 2011 IEEE 2011; 1–7.
37. Shih-An Y, Chan-Nan L. Distribution feeder scheduling considering variable load profile and outage costs. IEEE Transactions on Power Systems 2009; 2:652–60.
38. Kashem MA, Ganapathy V, Jasmon GB, Buhari MI. A novel method for loss minimization in distribution networks. International Conference on Electric Utility Deregulation and Restructuring and Power Technologies. Proceedings 2000; 251-256.