Reinforcement Learning (RL) to Optimal Reconfiguration of Radial Distribution System (RDS)

John G. Vlachogiannis[1] and Nikos Hatziargyriou[2]

[1] Industry & Energy (IEI) Lab
Dept. of Informatics & Computer Technology
Technological Educational Institute (TEI) of Lamia, Greece
[2] Power Systems Lab
School of Electrical & Computer Engineering
National Technical University of Athens (NTUA), Greece

Abstract. This paper presents a Reinforcement Learning (RL) method for optimal reconfiguration of radial distribution system (RDS). Optimal reconfiguration involves selection of the best set of branches to be opened, one from each loop, such that the resulting RDS has the desired performance. Among the several performance criteria considered for optimal network reconfiguration, an important one is real power losses minimization, while satisfying voltage limits. The RL method formulates the reconfiguration of RDS as a multistage decision problem. More specifically, the model-free learning algorithm (Q-learning) learns by experience how to adjust a closed-loop control rule mapping operating states to control actions by means of reward values. Rewards are chosen to express how well control actions cause minimization of power losses. The Q-learning algorithm is applied to the reconfiguration of 33-bus RDS busbar system. The results are compared with those given by other evolutionary programming methods.

1 Introduction

The reconfiguration of radial distribution systems (RDS) aims at its optimal operation satisfying physical and operating constraints. One of the criteria for optimal operation is the minimization of the real power losses satisfying simultaneously operating limits of voltages. A number of algorithms based on evolutionary computation techniques [1-5] have been developed to solve this problem. These methods however are inefficient in providing optimal configurations for a whole planning period. In this paper the RDS problem is solved by means of Reinforcement Learning (RL) [6-9]. RL originates from optimal control theory and dynamic programming and aims at approximating by experience solutions to problems of unknown dynamics [8]. From a theoretical point of view, many breakthroughs have been realized concerning the convergence of the RL approach and their application to nonlinear systems [6, 8], leading to very efficient algorithms. Also the rapid increase in computer capacities makes RL methods feasible and attractive in the power system community [6, 8].

In order to apply RL, the reconfiguration problem is formulated as a multistage decision problem. Optimal control settings are learnt by experience adjusting a closed-

G.A. Vouros and T. Panayiotopoulos (Eds.): SETN 2004, LNAI 3025, pp. 439–446, 2004.

loop control rule, which is mapping operating states to control actions (set of branches switched off one by one at each loop of RDS). The control settings are based on rewards, expressing how well actions work over the whole planning period. As a reward function minimization of real power losses is chosen. Moreover all voltage limits must be satisfied. In the paper the model-free learning algorithm (Q-learning) [6] is applied to reactive power control, however the algorithm is general and can be applied to a wide variety of constrained optimization problems in planning or operational planning.

The paper is organized in 5 sections. Section 2 describes the Reinforcement Learning approach. In Section 3, Q-learning algorithm is implemented to optimal reconfiguration of RDS. In Section 4, the results obtained by the application of the Q-learning algorithm to the 33-bus RDS are presented. The results are compared with those obtained by the evolutionary programming algorithm [1], showing the superiority of RL. Moreover the superiority of Q-learning algorithm in providing optimal reconfiguration over the whole planning period is depicted. In Section 5, general conclusions are drawn.

2 Reinforcement Learning (RL)

Reinforcement Learning (RL) techniques are simple iterative algorithms that learn to act in an optimal way through experience gained by exploring an unknown system [6-9]. RL assumes that the "world" can be described by a set of states S and an "agent" can choose one action from a set of actions A. The operating range is divided into discrete learning-steps. At each learning-step the agent observes the current state s of the "world" ($s \in S$), and chooses an action $a \in A$ that tends to maximize an expected long-term value function [6-8]. After taking action (a), the agent is given an immediate reward $r \in \Re$, expressing the effectiveness of the action and observing the resulting state of the "world" $s' \in S$. The particular RL algorithm used in this work is the Q-learning algorithm [6]. The Q-learning optimal value function is defined by means of the Bellman equation, as:

$$Q^*(s,a) = E\left(r(s,a) + \gamma \max_{a'} Q^*(s',a')\right) \tag{1}$$

This represents the expected sum of rewards, when starting from an initial state (s) taking action (a), and performing optimal actions (a′) in next searches, until the optimal value of Q-function, (Q*(s,a)) is reached. The discount parameter γ ($0 \le \gamma \le 1$) is used to exponentially decrease the weight of rewards received in next searches [6-8].

Once we have the optimal value $Q^*(s,a)$, it is easy to determine the optimal action a* using a greedy policy [6-9]. A simple way is to look at all possible actions (a) from a given state (s) and select the one with the largest value:

$$a^* = \arg\max_a Q^*(s,a) \tag{2}$$

The Q-function (Q-memory) is typically stored in a table, indexed by state and action. Starting with arbitrary values, we can iteratively approximate the optimal Q-

function based on our optimality criteria. The table entry for state (s) and action (a) is then updated according to [6]:

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right) \quad (3)$$

It is important to note that the new value for Q(s,a) memory is based both on the current value of Q(s,a), and the values (immediate rewards) of control actions obtained by next searches. So, the parameter α ($0 \leq \alpha \leq 1$) plays a critical role representing the amount of the updated Q-memory (3) and affects the number of iterations. The parameter ($1-\alpha$) represents the total amount of Q-values still remaining as a memory in the Q-function [6].

Table 1. Q-Learning algorithm applied to optimal reconfiguration of RDS.

1) Initialize memory Q(s,a) and immediate rewards r(s,a)=0.0, $\forall s \in S$, $\forall a \in A$
2) Repeat for a given number of operating points over the whole planning period (load variations)
   2.1) Repeat
      2.1.1) Observe the state (s) of load flow solution
      2.1.2) Choose an action vector (a)
      2.1.3) Execute load flow
      2.1.4) Observe state (s´) resulting from load flow and calculate the reward (6)
      2.1.5) Update Q-function using (3)
      2.1.6) Set s←s´
   until optimal Q-function is achieved (no change in reward or in action vector)

3 Q-Learning Applied to Optimal Reconfiguration of RDS

For the purpose of our analysis a two-class classification is assumed. The "world" of RL solution states ($s \in S$) is binary, comprising acceptable operating points characterized by satisfaction of all constraints and unacceptable, when any constraint is violated. The control vectors that combine discrete values of control adjustments are the actions ($a \in A$), and the Q-learning algorithm is the "agent". The algorithm proceeds as follows: An operating point comprising a load and generation pattern including a set of control actions is randomly created. The agent observes the state (s) of the system, as obtained by the load flow solution, and chooses one control action (a) from the control vector. A new load flow is executed. The agent observes the resulting state of the solution (s′) and provides an immediate reward (r): $SxA \rightarrow \Re$ expressing the reduction of power losses. A new control (switching) action is selected next, leading to a new load flow solution and a new reward. Selection of new control actions is repeated until no more changes in the reward value or in control action can be achieved. The goal of the agent is to learn the optimal Q-function (Q*(s,a)) using the mappings of states to actions ($S \rightarrow A$) such that the long-term reward is maximized. The procedure is repeated for a large number of operating states covering the whole planning period. The agent finds the optimal control settings (a*) [6] using the optimal policy described by (3). Table I shows the Q-learning algorithm applied in the optimal reconfiguration of RDS.

3.1 State Vectors

In order to confine the constrained variable within its operating limits, the states of the system are discretized as follows:

When one of the variables (e.g voltage magnitude) lies outside its operating limits, the state is considered as –1 level-state, otherwise it is considered at the zero level-state. Consequently, if we have n-constrained variables, the total number of states is:

$$\overline{\overline{S}} = 2^n \tag{4}$$

In our application the lowest voltage in each loop is constrained within operating limits.

3.2 Action Vectors

If each control variable u_i is discretized in d_{u_i} levels (e.g branches to be opened one at each loop of RDS), the total number of action-vectors affecting the load flow is:

$$\overline{\overline{A}} = \prod_{i=1}^{m} d_{u_i} \tag{5}$$

m expresses the total number of control variables (e.g total number of branches to be switched out).

3.3 Rewards

Optimal reconfiguration involves selection of the best set of branches to be opened, one from each loop, such that the resulting RDS has the desired performance. Amongst the several performance criteria considered for optimal network reconfiguration, the one selected is the minimization of real power losses, while satisfying operating limits of voltages. Application of the Q-learning algorithm to optimal reconfiguration of RDS is linked to the choice of an immediate reward (r), such that the iterative value of Q-function (3) is maximized, while the minimization of total real power losses (TPRL) is satisfied over the whole planning period. So the immediate reward (r) is computed as:

$$r = - \text{Total Losses} \tag{6}$$

4 Performance Results

The Q-learning algorithm is applied to the optimal reconfiguration of the 33-bus RDS. The line diagram is shown in the Appendix and also in [1] together with the transmission line and load data. The control variables comprise the sets of branches to be opened, one from each loop. There are five loops, therefore each control action of Q-learning vector comprises five branches. Table 2 shows the branches comprising each loop. According to this Table, the total number of control action vectors is calculated

10x7x7x16x11= 86240. Since the lowest voltage magnitudes (v_j) in each loop are constrained within operating limits [0.96pu, 1.05pu], the total number of solution states is calculated 2^5= 32. The Q-learning algorithm (Table 1) can be implemented in a large number of load combinations (operating points) selected over the whole planning period.

Table 2. Branches to be opened at each loop of 33-bus RDS.

Loops	Branches
1	2-3-4-5-6-7-18-19-20-33
2	9-10-11-12-13-14-34
3	8-9-10-11-21-33-35
4	6-7-8-15-16-17-25-26-27-28-29-30-31-32-34-36
5	3-4-5-22-23-24-25-26-27-28-37

We first apply the Q-learning algorithm for a particular load profile. In this case we set the Q-learning parameters α = 0.99 and γ = 0.01. Figure 1 shows the obtained immediate reward r (6) at each Q-learning step, corresponding to the upper values of load. Each Q-learning step corresponds to an iteration of the Q-learning algorithm (Table 1). The agent made approximately 46000 Q-learning steps to find the optimum control actions. The whole computing time was 100 sec in a 1.4-GHz Pentium-IV PC. This figure also depicts the convergence of Q-learning algorithm in a maximum reward value (-0.354), mapping the optimum control action to the best solution state.

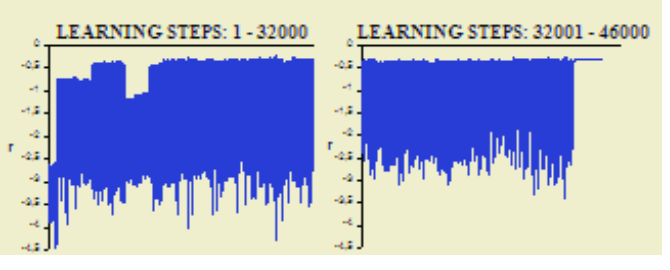

Fig. 1. Immediate rewards of Q-learning algorithm.

Table 3 shows the optimum control action (best set of branches switched out) 7-10-13-31-25 and the total real power losses calculated at 110,05 kW. Moreover, Table 3 shows the voltage magnitude achieved by the evolutionary programming algorithm reported in [1]. The latter proposes as best control action the set of branches to be opened: (6-14-9-32-37). Table 3 also gives the load voltage magnitudes of 33-bus RDS at the base case of branches switched out: (33-34-35-36-37).

The optimal solution of Q-learning compared to the evolutionary programming optimal solution is better, since all voltage constraints are satisfied and the total of real power losses are smaller (110.05 kW compared to 118.37 kW [1]).

The Q-learning algorithm also provides on-line control under non-autonomous environment [8]. Such case study is considered when the system load varies over a period. The load variation is cyclic with period of 50 Q-learning steps (ls) and it is modeled according to the equation:

$$z(ls) = z_{max} \cdot \sin\left(\frac{2 \cdot \pi \cdot ls}{50}\right) \quad (7)$$

z stands for the real or reactive parts of load. In this case we set the Q-learning parameters $\alpha = 0.1$ and $\gamma = 0.98$.

Table 3. Load bus voltages under the optimal control action given by Q-learning and programming evolutionary algorithm

Method	Q-learning	Evolutionary Programming [1]	Base Case
Optimum Control Action	7-13-10-31-25	6-14-9-32-37	33-34-35-36-37
Total real power losses	110.05 kW	118.37 kW	181.53 kW
Load Bus		Voltages	
1	1.020	1.020	1.020
2	1.017	1.017	1.017
3	1.007	1.007	1.003
4	1.005	1.003	0.996
5	1.003	1.000	0.990
6	0.998	0.991	0.974
7	0.998	0.982	0.972
8	0.986	0.984	0.961
9	0.981	0.981	0.956*
10	0.982	0.982	0.952*
11	0.986	0.983	0.952*
12	0.987	0.984	0.951*
13	0.985	0.980	0.946*
14	0.976	0.979	0.945*
15	0.977	0.979	0.944*
16	0.974	0.977	0.944*
17	0.969	0.973	0.943*
18	0.968	0.972	0.944*
19	1.015	1.015	1.016
20	0.997	0.995	1.012
21	0.992	0.990	1.010
22	0.988	0.986	1.009
23	1.001	1.004	0.999
24	0.989	0.997	0.990
25	0.980	0.993	0.985
26	0.982	0.989	0.972
27	0.982	0.987	0.970
28	0.980	0.978	0.960
29	0.978	0.971	0.953*
30	0.976	0.968	0.949*
31	0.974	0.965	0.944*
32	0.966	0.965	0.943*
33	0.967	0.971	0.942*

** Voltage magnitude violates lower limit*

The progressive learning of the control agent over the whole planning period is illustrated in Figure 2. The convergence of Q-learning algorithm took about 78000 steps. Each Q-learning step corresponds to an iteration of the Q-learning algorithm (Table 1). The whole computing times were 190 sec in a 1.4-GHz Pentium-IV PC. This figure also depicts the convergence of Q-learning algorithm in a maximum range of rewards [-0.467, -0.155] over the whole planning period, mapping the optimum control action to the best solution state. The greedy-optimum control action includes the branches to be switched out (6-10-8-32-37), satisfying all voltage constraints over the whole planning period.

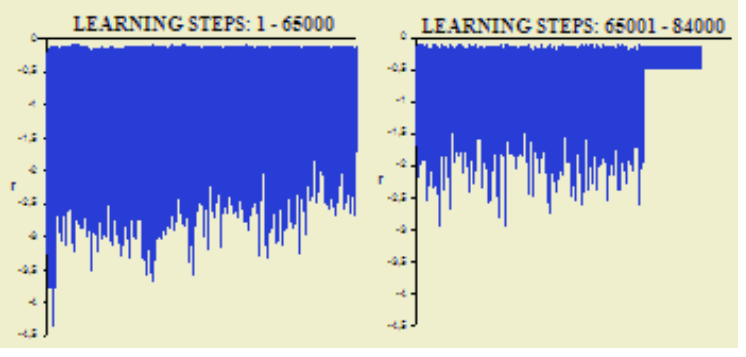

Fig. 2. Immediate rewards of Q-learning algorithm over the whole planning period.

5 Conclusions

In this paper the RL method was applied to the optimal reconfiguration of RDS. An iterative Q-learning algorithm is implemented in order to provide the optimal control action (set of branches to be opened one from each loop of RDS), satisfying all operating limits of the constrained variables (voltages) simultaneously with the minimum total of real power losses. Optimal control settings are learnt by experience adjusting a closed-loop control rule, which is mapping operating states to control actions by means of reward values. As a reward function the total of real power losses was chosen. The Q-learning algorithm was applied to the 33-bus RDS. The results have shown that Q-learning algorithm is able to provide better control settings than other evolutionary programming algorithms. Moreover, the RL approach provided on-line optimal reconfiguration of the 33-bus RDS over the whole planning period.

References

1. Venkatesh, B., Ranjan, R.: Optimal Radial Distribution System Reconfiguration using Fuzzy Adaption of Evolutionary Programming. Int. J. Electrical Power & Energy Systems 25 (2003) 775-780.
2. Baran, M.E., Wu, F.F.: Network reconfiguration in distribution systems for loss reduction and load balancing. IEEE Trans. on Power Delivery 4 (1989) 1401-1407.

3. Shirmohamaddi, D., Hong, H.W.: Reconfiguration of electric distribution networks for resistive line losses reduction. IEEE Trans. on Power Delivery, 4 (1989) 1484-1491.
4. Peponis, G.P., Papadopoulos, M.P., Hatziargyriou, N.D.: Distribution networks reconfiguration to minimize resistive line losses. IEEE Trans. on Power Delivery, 10 (1995) 1338-1342.
5. Kashem, M.A., Ganapathy, V., Jasmon, G.B., Buhari, M.I.: A novel method for loss minimization in distribution networks. Proc of Inter. Conf. on Electric Utility Deregulation and Restruct. and Power Tech., London (2000) 251-255.
6. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning 8 (1992) 279-292.
7. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4 (1996) 237-285.
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, Adaptive Computations and Machine Learning. MIT Press Cambridge MA (1998).
9. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific Belmont MA (1996).

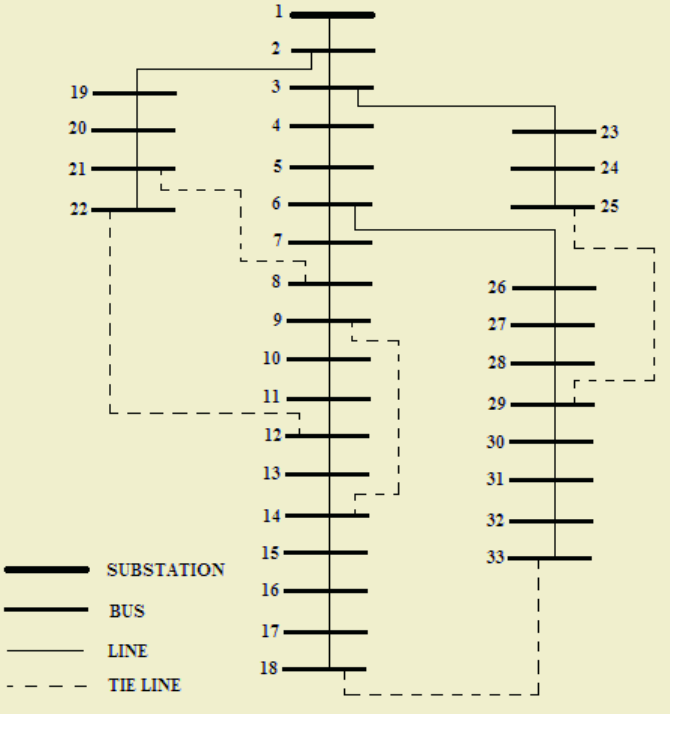

Fig. A1. 33-bus radial distribution system.