Application of Reinforcement Learning to Electrical Power System Closed-Loop Emergency Control

C. Druet, D. Ernst, and L. Wehenkel

Department of Electrical and Computer Engineering - Institut Montefiore
University of Liège - Sart-Tilman B28 - B4000 Liège - Belgium

Phone: +32-4-3662645 - Fax: +32-4-3662984
{druet,ernst,lwh}@montefiore.ulg.ac.be

Abstract. This paper investigates the use of reinforcement learning in electric power system emergency control. The approach consists of using numerical simulations together with on-policy Monte Carlo control to determine a discrete switching control law to trip generators so as to avoid loss of synchronism. The proposed approach is tested on a model of a real large scale power system and results are compared with a quasi-optimal control law designed by a brute force approach for this system.

1 Introduction

Reinforcement learning techniques are currently being investigated for suitability of use in a wide variety of environments. These range from game playing environments such as backgammon where these machine learning techniques have been successfully applied to develop systems capable of Master-level play ([8]) to self-adjusting algorithms for packet routing in computer networks ([1]).

In the field of automatic control *Reinforcement Learning* starts also to be well-known. Its principle is to learn how to control by associating a certain benefit to being in a particular state and taking a particular action in that state. We can divide such control learner (agent) in two categories. First we can plug the agent and wait for him to know enough about the system to control it. Unfortunately this is not always possible. Let us imagine a car driven by an agent which doesn't now anything about driving rules. We will have to buy lots of cars before having a *capable* agent. It leads to the second category, the agent which learns from simulated experience before being used in real-life.

In the electric power system community, Dynamic Security Assessment (DSA) has long been recognized to be an issue of great practical concern ([2]). The recent deregulated practices make the need for effective DSA methods more urgent than ever. What holds true for predictive DSA holds even more true for emergency DSA. Predictive DSA concerns what can be done in study mode and then used in the control center to enhance security. Emergency DSA concerns

© Springer-Verlag Berlin Heidelberg 2000

implementation of control systems to deal with emergency situations. It becomes a necessity, given the trend to operate the systems increasingly closer to their limits and given the difficulties in predicting the operating conditions and the troublesome contingencies likely to occur.

This paper deals with during transient emergency TSA (Transient Stability Assessment) and control. The purpose is to demonstrate that an agent using reinforcement learning techniques can appraise and trigger control actions so as to prevent the electrical power system from serious degradation.

2 Reinforcement Learning

2.1 Basics

The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. Reinforcement learning is learning what to do, i.e. how to map situations to actions, so as to maximize a reward signal ([7]). The goal is to discover the actions which yield the most reward, by trying them out. In the most interesting and challenging case, actions may affect not only the immediate reward but also the next situation and all the subsequent rewards.

One of the challenges in reinforcement learning is the trade-off between exploration and exploitation. To obtain a high reward, an agent must prefer actions that it has tried before and found to be rewarding. But some other actions could be better, i.e. actions not yet taken, it must then try them out. The first behavior is called *exploitation*, the second *exploration*. The challenge is that neither exploration nor exploitation can be pursued exclusively without failing the task ([9]).

As we will use the terms *policy*, *reward function* and *value function*, we will define them. A **policy** (π) defines the agent's way of behaving at a given time. A policy is thus a mapping from perceived states to actions. A **reward function** defines the goal in a reinforcement learning problem. This function defines what are the good and the bad events for the agent. The **value function** is more complicated. It specifies what is good in the long run. The value of a state is the reward (R) an agent can expect to accumulate over the future, starting from that state.

Two kinds of value functions exist: the state-value function $V(s)$, where s denotes the state, and the action-value function[1] $Q(s, a)$ where a denotes the action. The first says how good it is to be in a particular state, the second how good it is to take a particular action in a particular state.

The agent makes its decisions as a function of a signal from the environment (state). Certainly, state signal should include the immediate sensations, e.g. measurements, but it has to contain more than that. This state signal is of course not expected to inform the agent of everything about the environment, or even everything that would be useful to it in making decision. Ideally it must sum-

[1] Also called state-action function.

marize past sensations compactly, in such a way that all relevant information is retained. A state signal that succeeds in retaining all relevant information is said to have the *Markov property*[2]. Of course this is very restrictive, but even when the state signal doesn't have the Markov property, one can consider it as an approximation of a Markov state, i.e. the more the state signal approaches the Markov property, the better the performance from reinforcement learning systems will be.

2.2 On-Policy Monte Carlo Control

Monte Carlo methods require only experience-sample sequences of states, actions and rewards from on-line or simulated interaction with an environment. Learning from *on-line* experience is striking because it requires no prior knowledge of the environment's dynamics, yet can still reach an optimal behavior in the long run. Learning from simulated experience is also powerful. Although a model is required, the model only generates sample transitions, not all the possible transitions.

We assume experience is divided into episodes. An episode is a sequence of state-action pairs and rewards. All episodes terminate no matter what actions are selected. Monte Carlo methods are incremental in an episode-by-episode sense.

The idea of the Monte Carlo Control method is to maintain both approximate policy and approximate value function. In this scheme, the value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function. These two changes work against each other as each creates a target for

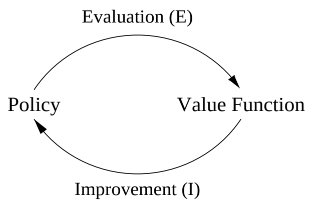

Fig. 1. Approach Scheme

the other, but together they cause both policy and value function to approach optimality.

$$Q^0 \xrightarrow{I} \pi_0 \xrightarrow{E} Q^1 \xrightarrow{I} \pi_1 \xrightarrow{E} Q^2 \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{E} Q^* \xrightarrow{I} \pi^*$$

Policy evaluation (E) is done in the following manner: for each state-action pair (s, a) appearing in the episode k,

$$Q^{k+1}(s,a) = Q^k + \alpha[R - Q^k(s,a)]$$

[2] Or to be Markov

where R is the reward following that state-action pair. Two techniques have been tested in this paper:

1. $\alpha = \frac{1}{n_{s,a}}$, where $n_{s,a}$ is the number of times the agent has passed through the state-action pair (s, a)[3], which corresponds to estimate the state-action values as a sample average of observed rewards;
2. $\alpha = C^{st}$ indicating that the estimates never completely converge but continue to vary in response to the most recently received rewards[4].

Policy improvement (I) can be done making the policy greedy, i.e. a policy that selects the action which has the higher expected reward, with respect to the current action-value function.

$$\pi_k(s) = \arg\max_a Q^k(s, a)$$

This procedure is modified by introducing the exploration part in it. Instead of using the greedy policy described above, one can use an ϵ-greedy policy, i.e. a policy that selects another action than the one with the higher expected reward with a probability of ϵ.

The general algorithm used to implement the Monte Carlo control is thus based on the following scheme.

1. Initialize $Q^0(s, a) \to \pi_0$.
2. Generate an episode using π_k.
3. Evaluate the policy and update the state-action function $\to Q^{k+1}(s, a)$.
4. Improve the policy $\to \pi_{k+1}$.
5. Return to point 2 (loop forever).

The method is called *on-policy* Monte Carlo control because it attempts to estimate the value of a policy while using it. In opposition is the *off-policy* Monte Carlo control method which uses separate policies: one policy is used to generate the episodes (*behavior* policy) which can be unrelated to the second policy that is evaluated and improved, the *estimation* policy.

3 The Practical Problem

Transient stability concerns the ability of an electrical power system to maintain synchronism when subjected to a severe transient disturbance, e.g. a three phase short-circuit cleared by opening a line. The resulting system response involves large excursions of generator rotor angles and is influenced by the nonlinear power angle relationship. When the excursions becomes too large, a loss of synchronism may occur: the system is driven to instability.

[3] This method is called *every-visit Monte-Carlo method* by opposition to the *first-visit Monte-Carlo method* which consists in using only the rewards associated to the first visit to (s, a) in each episode.

[4] This is desirable in a nonstationary environment.

The selective tripping of generating units for a severe disturbance which weakens the system transfer capabilities can be used as a method of improving system stability. The rejection of an appropriate amount of generation in the system reduces power to be transferred over the critical transmission interface. Since generating units can be tripped rapidly, the method constitutes a very effective means of improving transient stability ([6]).

The illustration is based on a lightly modified Brasilian power system. The resulting system comprises 63 machines, 1180 busses and 1968 lines and is modeled in its usual detailed way. The generation shedding scheme is applied to the Itaipu transmission system (figure 2): 8 machines of 700 WM, at 60Hz side of Itaipu).

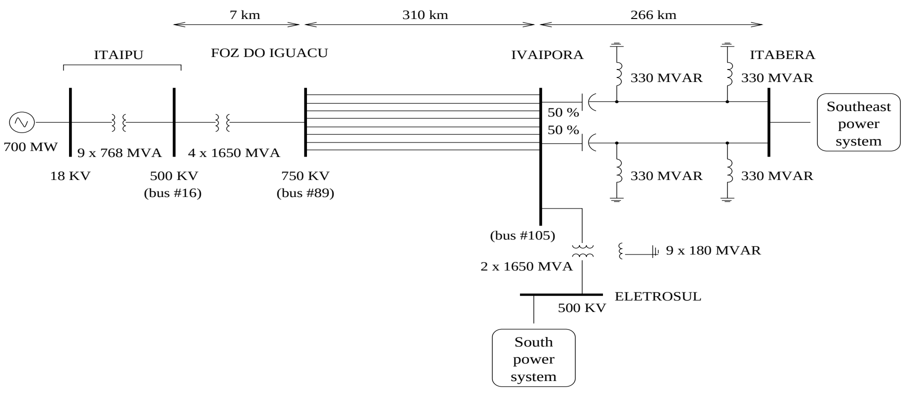

Fig. 2. Topology of the Itaipu transmission system (60 Hz)

What this experience attempts to demonstrate is that it is possible to use reinforcement learning method to control the shedding of the units at Itaipu to maintain stability in terms of synchronism. The control center Itaipu receives measurements of the δ (relative electrical angle, *deg*: $\delta - \delta_{initial}$), ω (angular speed, rad/s), P_m (mechanical power, MW), P_e (electrical power, MW) and the status of the 8 units (on, stopping, off). This is clearly one of the possible descriptions of the state of the system. At each time step, one can decide to shed units. In our experiments one can only shed one machine by time step. As the units are identical, two control actions are possible: to shed one of the 8 units or to wait (*shed-one-unit* or *no-shedding*). Once the control action is decided, a delay of $50\,ms$ is introduced before the transition actually occurs, so as to better suit the real-life case[5]. A loss of synchronism will correspond to a bad event (negative reward). A positive reward will be associated to a stable system (system stable x seconds after the fault).

The Measurements. For want of real-world measurements, the illustration is based on time-domain simulation using the ST600 program of Hydro-Québec

[5] The *stopping* status of a unit: the control action is decided but the action will be effective after the transmission delay.

([10]). Real-time measurements are thus artificially created. The acquisition of these measurements is supposed to have an observation rate of 5 ms.

The Contingencies. A simulation, i.e. an episode, consists of a three-phase short-circuit applied at bus # 89. The starting time t_s is randomly chosen between $t_{s\ min}$ and $t_{s\ max}$ and the clearing time t_e is randomly chosen between $t_{e\ min}$ and $t_{e\ max}$ to cover a wide variety of cases as a light warranty of generality. The non-zero t_s and the use of a contingency without short-circuit (*no-fault* contingency) prevent the systematic shedding of units. Several post-fault configurations (*x-lines-tripped* contingencies) are also used.

The Agent. Starting from a state s and taking action a one cannot determine the resulting state s', thus we must use a state-action function. The state-action function is an associative table $((s, a) \leftrightarrow r)$; the continuous variation spaces of δ, ω, P_m, P_e are thus discretized.

The policy followed by the agent is an ϵ-greedy policy. After evaluation of the expected reward of each action, the agent chooses the action with the highest reward. If several actions match the highest reward, it chooses one randomly among them. To maintain exploration, a random action is selected with a probability of ϵ. The choice of this ϵ is difficult as we want to keep as much generation as possible. This leads to a very small ϵ [6], so as to avoid excessive shedding.

The Discretizations are very simple. As the eight generators are exactly identical, only one combination of δ, ω and P_a ($P_m - P_e$) is enough to represent each unit's state. In addition, the number of *running* units U_r (0-8) and the number of *stopping* units U_s complete the description of the state.

The space of δ, ω and P_a are respectively divided into 41, 21 and 15 ranges (respectively 40, 20 and 15 of identical length for each variable plus an additional category to represent the *no-unit-running* case).

The Rewards. The goal of the agent is not only to save the system but also to preserve as much generation as possible. When the system is stable, the reward is positive and corresponds to the total amount of electrical power produced at the end of the simulation so as to distinguish good stable situations (6 units producing) from less good stable situations (e.g. 4 units only). When the system is unstable, the reward is negative and corresponds to the loss of the starting generation, i.e. -5600 (MW).

4 The Simulation Results

Our experiments demonstrate that one can apply successfully reinforcement learning techniques to the transient stability emergency problem. On the figure 3 (Contingency: *4-lines-tripped*, $50 < t_e < 60\,ms$, $0 < t_s < 100\,ms$, $\alpha = \frac{1}{n_{s,a}}$, $\epsilon = 0.005$), one can see the evolution of the stability of the system (*dots* are stable episodes and *crosses* unstable episodes) and the evolution of the final production P_f when the number of simulations increases k.

[6] $\epsilon < \frac{\text{number of units}}{\text{maximum number of time steps}}$.

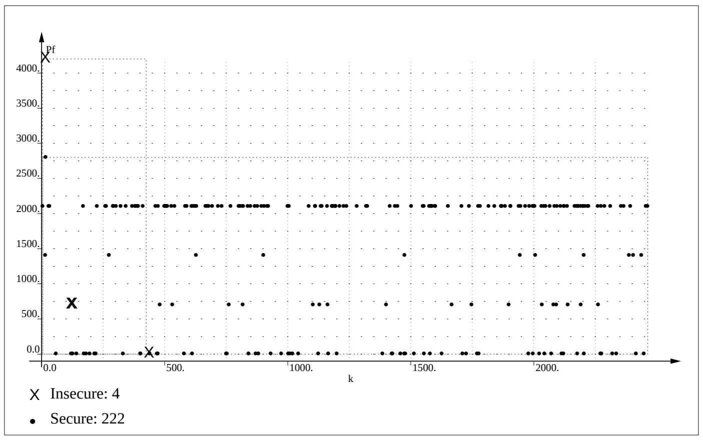

Fig. 3. MW not rejected **versus** evolution of learning

The convergence is very fast in terms of saving the system (a small number of unstable cases with low values of k). This agent learns also very fast to shed as few generation as possible (P_f already high after 250 episodes). The goal is of course to have higher MW not rejected points when the number of episodes increases (later we will see that ideally the value of P_f should still increase up to 2800 MW).

4.1 $\alpha = C^{st}$ Versus $\alpha = \frac{1}{n_{s,a}}$

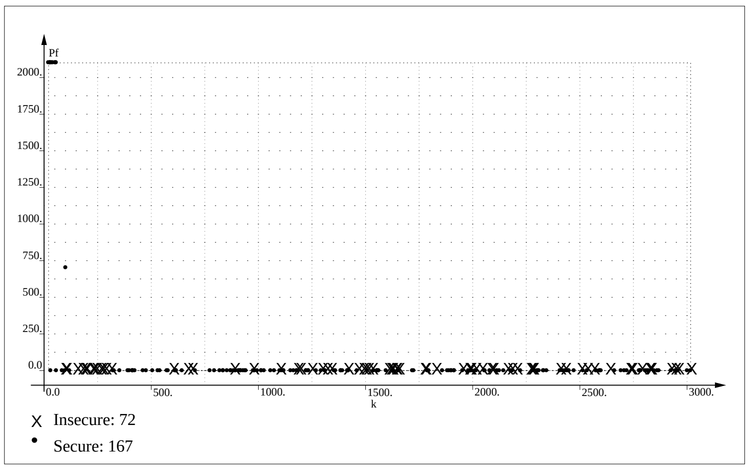

Fig. 4. MW not rejected **versus** evolution of learning for $\alpha = C^{st}$ $(= 0.2)$

We have observed that the second technique ($\alpha = \frac{1}{n_{s,a}}$) converges better and faster than the first technique ($\alpha = C^{st}$). The reason for that is the way we defined the problem. As the state of the system is always the same at the begin-

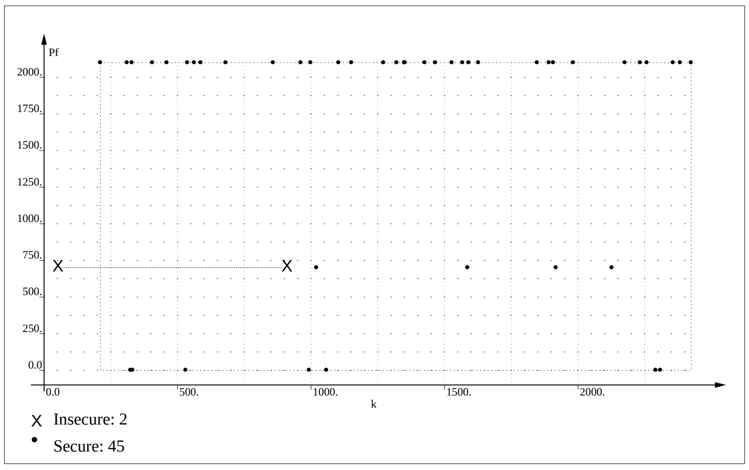

Fig. 5. MW not rejected **versus** evolution of learning for $\alpha = \frac{1}{n_{s,a}}$

ning of the simulation, as the contingencies are always the same, as the models behind the simulator are fixed, the problem can be considered as a stationary problem. Thus there is no reason to use a constant α which is particularly useful in a non-stationary environment.

Looking on the figures 4 and 5 (Contingency: *3-lines-tripped*, $40 < t_e < 50\,ms$, $0 < t_s < 100\,ms$, $\epsilon = 0.005$), one can see that the technique using $\alpha = C^{st}$ ($=$ 0.2) does not even converge to save the system (a lot of crosses during the whole learning). but that the one using $\alpha = \frac{1}{n_{s,a}}$ converges very fast (small number of unstable cases with low values of k). Moreover it successfully succeeds in saving as much production as possible (a already high P_f for low k) despite $\alpha = C^{st}$ cannot.

4.2 Quality of the Control

For this experiment, we use two contingencies, the *no-fault* contingency and the *4-lines-tripped* contingency. The fault duration varies between $40\,ms$ and $100\,ms$.

Analysis of the reinforcement learning control. The state-action values are equal to the sample average of the observed rewards ($\alpha = \frac{1}{n_{s,a}}$). According to the procedure previously established, 2500 simulations have been run. After these simulations, the reward is not corrected anymore and the ϵ-greedy decision process is changed to greedy. The decision process is thus deterministic[7].

These two modifications having been done, we can study the evolution of the controlled system to a *no-fault* contingency. The simulation shows that the method has learned that it was not necessary to exclude machines to stabilize the system: the best reward associated to the current state s is always bound to the *no-action*. Note that in such a scenario, if the decision to take an action is

[7] Except when several actions have the same reward in a state s.

not taken at the first instant, no action will be taken at all. This is the direct consequence that the state $s = (\delta, \omega, P_a, U_r, U_s)$ is here constant if any action is decided. 8 machines are thus still in activity at the end of the simulation, the reward is maximum and equal to 5600 ($8 * 700\, MW$).

For the *4-lines-tripped* contingency each state of the period preceding the fault is constant and the same as the constant state of a *no-fault* contingency. Such an observation suggests that the instant of appearance of the fault does not influence the control process of the system. The only relevant parameter is the fault duration (t_e). The following table represents the number of shed machines (U_n) and the production (P_f) at the end of the simulation for different values of t_e. Even if all the machines are sometimes shed (8), no unstable simulations occurs: the control process has always been able to stabilize the system. Note that except for $t_e = 85\, ms$, the number of shed machines is an increasing function of the fault duration.

$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f
40	5	2100	55	5	2100	70	7	700	85	7	700
45	5	2100	60	5	2100	75	7	700	90	7	700
50	5	2100	65	7	700	80	8	0	95	8	0

To summarize, we can say that for a *no-fault* contingency the control of the system is optimum because no machines are shed while avoiding loss of stability. For the *4-lines-tripped* contingency, the control also stabilizes the system but the price to pay (the number of shed machines) is sometimes very high. Nothing guarantees that the control designed here is for the *4-lines-tripped* contingencies the optimum or even close to the optimum.

Design of the optimal control. The reinforcement learning procedure that we have applied here to the problem of generation shedding control is self-reliant in the sense that to establish the control law, no human knowledge of the dynamics of the system was required. Exploiting the knowledge of the system we have ([3] and [5]) we are able to establish a near to optimum control law. We described its design in an internal report ([4]).

The following table summarizes the results obtained with simulations carried out using this near optimum control procedure. A comparison with the previous table highlights better quality, but we have to keep in mind that it was only possible to establish this law thanks to the perfect knowledge that we had of the system dynamics and the restricted set of contingencies considered. The comparison is just aimed to give an idea of how far we are from the optimum when we use the reinforcement learning procedure to design a control.

$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f	$t_e(ms)$	U_n	P_f
40	3	3500	55	4	2800	70	4	2800	85	5	2100
45	4	2800	60	4	2800	75	4	2800	90	5	2100
50	4	2800	65	4	2800	80	5	2100	95	5	2100

The disappointing performances of the reinforcement learning control law originate from the state discretization which decreases the degree of observability of the system and thus the ability to associate to each state an optimal control action.

5 Discussion

The interest of reinforcement learning for electrical power system emergency control has been demonstrated. To improve the results, we believe that further research is necessary, in particular in terms of choosing a better reward signal.

Problem enhancement. To be applied in Itaipu this procedure has of course to be improved. First, one must consider noise and other uncertainties in the state signal. Second, the experimentation domain should be enlarged by including other starting states, other topologies and other perturbations.

Method enhancement. The first thing to improve is the discretization of δ, ω and P_a. An alternative to that is the use of continuous variable spaces using more sophisticated machine learning techniques such as regression trees or neural networks (multi-layer perceptron) to generalize the state-action function approximation.

References

1. J.A. Boyan and M.L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, San Francisco CA, 1993.
2. T. Dy Liacco. Security functions in power system control centers. In *IFAC Symp. on Power System Control, New Delhi*, 1979.
3. D. Ernst, A. Bettiol, Y. Zhang, L. Wehenkel, and M. Pavella. Real time transient stability emergency control of the south-southeast brazilian system. *SEPOPE, Salvador, Brazil*, May 1998.
4. D. Ernst and C. Druet. Design of an optimal control law for emergency transient stability, 2000. Internal report.
5. D. Ernst and M. Pavella. Closed-loop transient stability emergency control. *Presented at the On-line Transient Stability Assessment and Control, panel session at the IEEE/PES Winter Meeting 2000, Singapore*, 2000.
6. P. Kundur and G.K. Morison. Techniques for emergency control of power systems and their implementation. In *Proceedings of the IFAC-CIGRE Symp. on Control of Power Plants and Power Systems, Beijing*, pages 679–684, 1997.
7. R.S. Sutton and A.G. Barto. *Reinforcement Learning, an introduction.* The MIT Press, 1998.
8. G.J. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. In *Proceedings of the AAAI Fall Symp. on Games: Planning and Learning*, pages 19–23. AAAI Press Technical Report FS93-02, 1993.
9. S.B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Shool of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213-3890, January 1992.
10. A. Vallette, F. Lafrance, S. Lefebvre, and L. Radakovitz. *ST600 programme de stabilité : manuel d'utilisation version 701.* Hydro-Québec, Vice-présidence technologie et IREQ, 1987.