NEW DEVELOPMENTS IN THE APPLICATION OF AUTOMATIC LEARNING TO POWER SYSTEM CONTROL

Louis Wehenkel, Mevludin Glavic, Damien Ernst
Department of Electrical Engineering and Computer Science
University of Liège - Sart-Tilman B28 - B-4000 Liège
{L.Wehenkel,dernst}@ulg.ac.be , glavic@montefiore.ulg.ac.be

Abstract - In this paper we present the basic principles of supervised learning and reinforcement learning as two complementary frameworks to design control laws or decision policies within the context of power system control. We also review recent developments in the realm of automatic learning methods and discuss their applicability to power system decision and control problems. Simulation results illustrating the potentials of the recently introduced fitted Q iteration learning algorithm in controlling a TCSC device aimed to damp electro-mechanical oscillations in a synthetic 4-machine system, are included in the paper.

Keywords - Optimal control, sequential decision making, Monte-Carlo methods, supervised learning, reinforcement learning, anytime algorithms.

1 INTRODUCTION

Power system control has been evolving recently and it continues to do so driven by the changing conditions and the progress in control theory and computer science. Developments in these fields that have not yet been seriously exploited in the context of power system control are for example the theory of hybrid systems combining automata theory and systems theory in a single framework, and new results in automatic learning theory and stochastic optimization algorithms. In computer science, a switch of paradigm is observed from deterministic algorithms trying to find the exact solution of a (typically rather idealistically posed) problem towards stochastic algorithms which aim rather to furnish with a certain probability a good approximation to the solution of a (typically much more realistically posed) problem. It turns out that in the latter class one finds many interesting and scalable anytime[1] algorithms. One can conceptualize automatic learning as a particular case of this latter framework.

Given the increase in uncertainties and speed of change in the environment the power system engineer has to cope with in present days, we believe and argue that it is time to try to exploit this new family of algorithms more systematically. The main objective of this paper is to provide a tutorial introduction to some of these new developments, and to explain their relevance in power system control applications.

We first review classical results from (stochastic) dynamic programming and (stochastic) optimal control theory and analyze the suitability of this framework to model in a rather generic way many interesting power system decision and control problems. We also revisit the basic principles of supervised and reinforcement learning as two complementary frameworks to design control laws or decision policies.

We then present recent developments in the realm of automatic learning methods and discuss their applicability to a panel of power system control problems. We also explain the recently introduced "Fitted Q iteration" algorithm and the novel "Extra-Trees" method involving ensembles of randomized decision or regression trees.

Simulation results of power system slow and fast dynamics real-time control are included to illustrate the framework. A 4-machine test system, with a Thyristor Controlled Series Capacitor (TCSC), is used for this purpose. Two measurement schemes to design a local and a wide-area controller are considered with realistic off-line training and validation scenarios.

The rest of the paper is organized as follows. Section 2 provides the necessary background from stochastic optimal control theory. Section 3 describes two basic protocols of automatic learning, namely supervised and reinforcement learning and some novel developments and ongoing research work that are of significant importance to the practical applicability of these methods to real control problems. Section 4 presents our application example and Section 5 a concluding discussion.

2 OPTIMAL CONTROL AND SEQUENTIAL DECISION MAKING

We present well-known results from (stochastic) dynamic programming and (stochastic) optimal control theories. Our objective is mainly to set the framework under which the remainder of the paper is shaped while stressing the similarities of the problems of designing a human agent's decision making strategy and an automatic control device's control law. A much more detailed description, including the mathematical assumptions, can be found in [1], from which we also borrow the notation.

2.1 General stochastic system model

Let us consider a discrete time state description of a stochastic dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, h-1 \quad (1)$$

where

[1]Roughly, an "anytime" algorithm is an algorithm which if interrupted at anytime can furnish an approximate solution; these algorithms provide solutions whose quality gradually increases with the amount of CPU time they are allowed to use.

k indexes the discrete time,

x_k is called the system state at time k,

u_k is the control or decision variable applied at time k,

w_k is a random process (also called disturbance),

h denotes temporal horizon of the problem.

Furthermore, we denote by X_k the state space at time k, i.e. the mathematical structure[2] describing a set of possible states of our system model at time k, by $U_k(x_k)$ the set of possible control decisions at time k (possibly a function of the state x_k). The random disturbance model is given in the form of a conditional probability distribution $P_k(\cdot|x_k, u_k)$ that may depend explicitly on the current state and control decision, but not on past values of w.

Supposing the system is initially (at step $k = 0$) in a given state x, its trajectory is then defined by the following random process:

1. set $k = 0$ and $x_k = x$;
2. the control agent selects a control $u_k \in U_k(x_k)$, and a random experiment selects a value of w_k distributed according to $P_k(\cdot|x_k, u_k)$;
3. at time $k + 1$ the system moves to state $x_{k+1} = f_k(x_k, u_k, w_k) \in X_k$ according to its dynamics,
4. the process repeats itself $h - 1$ further times, by replacing the index k by $k + 1$ at stage 2, yielding an h-stages trajectory.

Notice that the notion of state used here is an extension of the classical notion of state used in deterministic system theory. More specifically, once x_k and u_k are given, the subsequent states are independent of previous states and controls, although they are not necessarily perfectly predictable. This is also called the *Markov property*, because if u_k depends only on x_k the sequence of states $x_0, \ldots, x_{h-1}$ forms a *Markov chain*.

The system is said to be *time-invariant* if the function f_k (and the sets X_k and U_k) and the probability distribution P_k do not depend explicitly on time, in which case we will drop the subscript k in our notation.

2.2 Performance criterion and candidate strategies

We consider an *additive over time* return-criterion. Namely, we define the return over an h-stages "trajectory" $(x_0, u_0, w_0, \ldots, x_{h-1}, u_{h-1}, w_{h-1}, x_h)$ by

$$J_h(x_0, u_0, \ldots, x_h) = g_h(x_h) + \sum_{k=0}^{h-1} r_k(x_k, u_k, w_k). \quad (2)$$

Notice that this performance criterion takes into account a terminal reward $g_h(x_h)$ and at each time step an instantaneous reward $r_k(x_k, u_k, w_k)$. The latter one is potentially stochastic (dependence on the r.v. w_k) and in general time-dependent.

The objective is to define control signals in such a way that the return is maximized. Furthermore, the control policy has to be causal which means that the control applied at time k is not allowed to depend on information which has not yet been gathered at time k. Assuming that the controller has the possibility to observe the state at time k and to store this value for future usage, the most general class of control policies that makes sense (we call these the *admissible* policies) is defined by an h-vector π of conditional probability distributions

$$\pi_k(u_k|x_k, u_{k-1}, \ldots, u_0, x_0), \quad k = 0, 1, \ldots, h-1. \quad (3)$$

The controller can use such a policy to make a random draw of the control u_k at time k depending in some way on its current knowledge.

Once such a control strategy has been selected, the distribution of N-stages trajectories starting from a given initial state $x_0 = x$ is well defined. Thus, the so-called expected return over h-stages

$$J_h^\pi(x) = E\{g_h(x_h) + \sum_{k=0}^{h-1} r_k(x_k, u_k, w_k)\}, \quad (4)$$

where the expectation is taken according to the distribution of trajectories starting from $x_0 = x$ and induced by the system dynamics f_k, noise distribution P_k, and choice of control policy π.

The solution of the optimal control problem consists of exploiting the knowledge of the system dynamics and return function so as to define an optimal policy π^*, i.e. an admissible policy such that for any initial state $x_0 = x$ and any admissible policy π, $J_h^{\pi^*}(x) \geq J_h^\pi(x)$.

We call this very broad class of controllers the *non-anticipating* controllers, to distinguish them from *open-loop* ones which only use the information about x_0 and *closed-loop* ones which only use information about x_k to select u_k.

2.3 Main results from the dynamic programming theory

2.3.1 Optimality of deterministic Markov policies

Under the assumptions given, one can show that the class of policies can be restricted to so-called deterministic Markov policies without sacrificing optimality. These are policies such that $\pi_k(u_k|x_k, u_{k-1}, x_{k-1}, \ldots, x_0)$ can be written as a function (a Dirac probability distribution) of x_k only, which we will denote by $\mu_k(x_k)$.

Hence, the search for an optimal policy π^* can be reduced to the simpler problem of searching for a sequence of h functions $\mu_k^*(\cdot)$. We will call this optimal sequence the optimal closed-loop time-variant control policy.

2.3.2 Sub-optimality of open-loop policies

One can easily show that if the system is deterministic, the optimal closed-loop time-variant policy may also be expressed as a function of only the initial state x_0, i.e. as an open-loop policy.

However, in the general case of a non-deterministic system, closed-loop policies can yield better performances than open-loop ones.

[2] We would like to stress the fact that in this framework the state space can be discrete, Euclidean, or a mixture of both. In power system problems the state space could be modeled as a finite union of finite dimensional Euclidean spaces corresponding to the configurations of the discrete state variables.

2.3.3 Time invariance and infinite horizon

Under the assumption of a time-invariant system and of a return defined over an infinite horizon by $\gamma \in (0;1)$ and

$$J_{\infty}^{\pi}(x_0) = \lim_{h\to\infty} E\{\sum_{k=0}^{h-1} \gamma^k r(x_k, u_k, w_k)\}, \quad (5)$$

one can show that optimality can be reached inside the even simpler set of *time-invariant* closed-loop policies.

Assuming moreover that U does not depend on x, the optimal policy can be obtained by determining the solution $Q(x,u)$ of the so-called Bellman equation

$$Q(x,u) = \underset{w}{E}\left\{ r(x,u,w) + \gamma \max_{u' \in U} Q\left(f(x,u,w), u'\right) \right\}, \quad (6)$$

and by posing

$$\mu^*(x) = \arg\max_{u\in U} Q(x,u). \quad (7)$$

Notice that the function defined by

$$V(x) = \max_{u\in U} Q(x,u), \quad (8)$$

is the expected return obtained if we start at state $x_0 = x$ and use the optimal closed-loop policy μ^* for a sufficiently large (in principle infinite) number of times. This function is called the value function in the dynamic programming literature and satisfies the following Bellman equation

$$V(x) = \max_{u\in U} \left[\underset{w}{E}\left\{ r(x,u,w) + \gamma V(f(x,u,w)) \right\} \right]. \quad (9)$$

The *value iteration* algorithms consist in solving iteratively this latter equation and deriving therefrom the Q-function by using (8) together with (6) and the system model, and deriving from it the optimal control policy by equation (7).

The exact solution is possible in practice only if the state-space is finite, and of sufficiently small size. In the case of continuous state-spaces, an approximate solution can be obtained by discretizing the state-space into a large enough number of bins. However, if the dimensionality of the continuous state-space increases then the number of bins would grow exponentially with the number of dimensions and very quickly make this approach infeasible. This difficulty has been called by Bellman the *curse of dimensionality* [2]. One possible solution is then to use so-called approximation architectures in order to reduce the problem dimensionality and Monte-Carlo techniques to estimate solutions to the Bellman equation in the most interesting regions of the state-space. This is actually one of the tasks of automatic learning based control discussed in the Section 3.

Variations of the Bellman equations exist for the time-variant and/or finite horizon problems. Their solution yields a sequence of Q_h-functions and therefrom the time-variant closed-loop control laws. Actually, in the case of a time invariant system the infinite horizon control law can be obtained as the limit of a sequence of policies of problems of growing time-horizon [3].

2.4 Discussion

2.4.1 Closed-loop vs open-loop

We saw that in the case of a deterministic system, both open and closed-loop approaches can solve the problem without sacrificing optimality. Using the Bellman equation to find the optimal control in closed-loop form may be cumbersome, but furnishes in a single step the control law valid for any initial condition.

On the other hand, the search for a single optimal sequence of control actions for a given initial condition x_0 amounts, in the case of a finite horizon of length h, to the determination of a single control sequence $(u_0(x_0), \ldots, u_{h-1}(x_0))$. This can often be done using efficient convex programming techniques, and so does not necessarily suffer from the curse of dimensionality. In the context of an infinite horizon, this approach can also be applied by refreshing at each time-step the open-loop policy computed for a sufficiently large but finite horizon; this is actually the so-called Model-Predictive-Control (MPC) approach, which has a better robustness than purely open-loop control and is often more tractable than solving the Bellman equation [4].

As concerns the deterministic vs non-deterministic modeling, we believe that in many power system problems the Markov assumption is more reasonable than assuming that the system is deterministic. For example, in cases where rather than observing the system state one observes only some measurements (e.g. in local control) or in longer term problems, where external disturbances related to weather and market conditions come into consideration, closed-loop control policies can be much more effective than open-loop ones.

2.4.2 Multi-step vs single step optimization

Let us briefly discuss the practical implications of formulating optimization problems over a certain time-horizon rather than using a purely static approach which optimizes at time k only with respect to instantaneous constraints and instantaneous performance criteria, such as the optimal power flow approach for instance. First of all, the static approach is obviously a particular case of the dynamic one. But in the dynamic programming approach it is possible to take into account in addition to static constraints also dynamic constraints in terms of possible transitions from time k to $k+1$ (the system dynamics). These constraints are handled in a static approach only after the fact, leading to suboptimal results. More importantly, in the dynamic approach it is possible to formulate optimization criteria which *integrate* over time, i.e. which are physically meaningful, contrary to instantaneous values which are not very meaningful in many cases. We hence believe that many power system problems, such as congestion management for instance, which are typically handled in a single step approach, could take advantage of a dynamic (multi-step) approach [5].

3 REVISITING AUTOMATIC LEARNING

For the sake of simplicity of our discussion, we now consider the determination of a time-invariant closed-loop control law (or a reactive decision strategy) maximizing the long term discounted return of a time-invariant system which state is perfectly observed. According to what precedes we denote this function by $\mu^*(x)$.

We can make several assumptions about our knowledge of this problem. For example, in the weakest setting we would assume that we can only observe the behavior of the system under control by some agent, and possibly that we can make some limited experimentations with the system by trying out some particular control actions in some particular states. In some more favorable occasions we will also assume that we have a so-called *generative* system model, which allows us to compute for any given initial state the probability of any successor state given any value of the control at time $k = 0$, as well as the expected value of the function $r(x, u, w)$. Notice that sequential sampling from such a distribution also allows us to generate sample trajectories.

We will consider that the problem is complex "enough", in the sense that we have no direct analytical means to solve the problem (i.e. determine the function $\mu^*(x)$) from whatever knowledge we have. We believe that this is the generic situation in the context of power system control applications.

We will try to identify an approximation to $\mu^*(x)$, that we will denote by $\hat{\mu}^*(x)$. The data that we will use is a table (or database) of system transitions, i.e. *four-tuples* of the form (x_k, x_{k+1}, u_k, r_k), obtained from observations of the real system or from simulations. Our objective is to define methods which under some circumstances will provide a good approximation of $\mu^*(x)$ from a reasonable amount of data and which quality further improves when the amount of data grows and, if possible, converges to the optimal control policy in asymptotic conditions.

3.1 Supervised learning protocol

3.1.1 Principle

Supervised learning is usually defined as follows [6]:

> *Given a training set of input/output pairs, determine a function (or model, or algorithm) to compute the outputs given the inputs which not only is accurate on the training set, but also generalizes well to unseen cases.*
>
> *If the output variable is discrete, one talks about classification, if it is numerical one talks about regression.*

In the context of supervised learning, it is thus assumed that there is a teacher which provides the learning agent with examples of correct decisions (outputs) for a representative set of states (inputs). The learning agent has then to generalize this information to unseen situations and, in some problems, to cope with randomness that appears because the inputs are not providing complete information about the outputs. This would be the case, for example, when the teacher uses complete information about the system state to determine its control actions while the learning agent has only access to partial information in the form of local measurements.

3.1.2 Application to power system control

In our context, this protocol could be used if we dispose of a database where $u_k = \mu^*(x_k)$ or at least a good enough approximation of it. Depending upon whether the control space is discrete or continuous we would then use a classification or a regression technique in order to generate a generalized version of the (x_k, u_k) pairs given in the database, in the form for example of a decision tree or a neural network.

This could allow us to understand how a human operator controls a system, to compare different operators of different systems, to replace a controller using centralized information by a controller (as close as possible in terms of decisions taken) using only a subset of information available in a local data acquisition system. In particular, such a framework could possibly be used in power systems in order to replace a single centralized controller taking a set of coordinated actions, by a set of distributed and independent local controllers trying to imitate the central controller. We could also use this approach in order to build from a database obtained by an off-line optimization algorithm a much faster control algorithm which uses restricted inputs and can be used in real-time.

In other words, supervised learning will allow us to imitate an already existing (human or algorithmic) controller, and to generalize it in different fashions by assuming and exploiting a panel of different input representations to the controlling agent.

3.2 Reinforcement learning protocol

Supervised learning alone is unable to learn a good control policy from a set of system transitions (four-tuples) strongly corrupted by erroneous (i.e. suboptimal) control actions. Thus, it is not sufficient when addressing a new control problem for which there is no experience yet about the optimal way of solving it or when there is no alternative way to determine what is an optimal control action in a given context.

In such situations, we need to call for a more sophisticated approach, which is called reinforcement learning in the computer science literature. Reinforcement learning has been designed to work also when the sole feedback information about system performance is given in terms of instantaneous rewards r_k and successor states x_{k+1} associated with a state-action pair (x_k, u_k). Actually, while supervised learning only exploits the x_k and u_k parts of the four-tuples given in the database to determine an approximation of $\mu^*(x)$ and assumes that $u_k \approx \mu^*(x_k)$, reinforcement learning exploits also the parts x_{k+1} and r_k, and does not make any assumption about the quality of u_k. Provided that the database is rich enough, reinforcement learning is then able to determine (explicitly or implic-

itly) information about the system dynamics $f(\cdot,\cdot,\cdot)$ and reward function $r(\cdot,\cdot,\cdot)$ and from this latter an approximation of the optimal control policy, even if the policy used to generate the system transitions of the database is far from optimal. Since reinforcement learning is less well known in the power system community, we will briefly describe the main principles of this approach and then discuss its possible application modes.

3.2.1 Principle

Reinforcement learning approaches can be of different kinds. One the one hand, the *model based* approaches work in two successive steps : (i) they use the database to create a model of the system (in the form of a Markov Decision Process) and of the reward function, (ii) they use these models together with standard dynamic programming procedures to determine a Q-function and corresponding policy. On the other extreme, we have the so-called *policy search* methods, which aim at directly approximating the optimal policy. In between these to extremes, one has the Q-learning framework [7] which essentially uses the database to determine an approximation of the Q-function and therefrom the control policy. We will explain below one particular approach to Q-learning which we have developed recently [3, 8]. We refer the interested reader to [9] and [10] for a general introduction and discussion of other reinforcement learning methods and the ongoing research in the field.

3.2.2 Supervised learning based Q-learning

The idea developed in [3, 8] aims at allowing the use of any supervised learning method (for regression) to provide the approximation (and generalization) of the Q-function in an iterative fashion. It works as follows:

- Initialization: Set $i = 0$ and $\hat{Q}_0(x,u) \equiv 0$.
- Basic iteration:
  - Set $i = i + 1$
  - Create a table of input/output pairs from the dataset: $\text{in}_k = x_k$ and $\text{out}_k = r_k + \gamma \max_{u' \in U} \hat{Q}_{i-1}(x_{k+1}, u')$
  - Apply a supervised learning algorithm to build $\hat{Q}_i(x,u)$ from the table of input/output pairs: $\hat{Q}_i(x,u)$ = Sup.Learn$\{(\text{in}_1, \text{out}_1), (\text{in}_2, \text{out}_2), \ldots\}$
- Finalization: if stopping conditions are satisfied return $\hat{Q}_i(x,u)$, otherwise go back to the basic iteration.

In essence, this algorithm (called *Fitted Q Iteration*) consists of solving the Bellman equation (6) iteratively by approximating the expectation operator through the use of a supervised learning algorithm on the set of four-tuples. In practice, the method converges after a number of iterations which strongly depends on the problem under concern. Its CPU time is thus several times larger than that of a single call to a supervised learning algorithm on the same database. The main advantage of this approach with respect to other reinforcement learning approaches is that it can be applied to any kind of database and that it can take advantage of any batch-mode supervised learning algorithm (see §3.3.1).

3.2.3 Application to power system control

The basic assumption in the reinforcement learning framework is that the database is representative in the sense that the (x_k)-parts are representative of the interesting regions of the state-space, that the (u_k)-parts are sufficiently diverse to allow the identification of optimal control actions from the corresponding instantaneous rewards and successor states r_k and x_{k+1}. If the system is non-deterministic, then these latter should also be conditionally representative of the distribution of successor states and rewards for a given state-action pair. In practice this implies that the database size (number of system transitions) needed in this protocol is significantly larger than in the case of the supervised learning protocol (where, however, we need to be sure that the (u_k)-parts are the optimal actions for the corresponding state x_k).

In the context of power system modeling and control problems, reinforcement learning can be used to exploit information obtained from different contexts:

- *Learning from a power system simulator:* it is often interesting, although not strictly necessary, to couple the learning agent with the simulator so as to exploit the result of learning in order to influence the way the subsequent four-tuples are generated. Thus the learning from simulations can be totally autonomous (see e.g. [11, 12] for some examples) or it can take advantage of some "teaching" mechanism which chooses in some way the most interesting simulation scenarios, so as to speed up learning as much as possible.
- *Learning from an actual controlling agent:* it is also possible to exploit in reinforcement learning the information gathered from a real system monitoring. For example, one could collect information about the control decisions taken by a human operator or by an existing automatic control device (together with rewards and successor states) and then use a reinforcement learning agent to learn a control policy from this information. The resulting policy may in principle outperform the original controlling agent. In this way it is also possible to collect information from several suboptimal controllers of a system and inject it into the learning agent.
- *On-line learning:* finally, it is possible even to couple the reinforcement learning agent directly with a real system, provided that safeguards are imposed in order to avoid that the agent (initially far from an optimal controller) creates catastrophic situations.

Of course, data can also be collected from any combination of these contexts. Even, in the context of uncertain system dynamics one can generate simulated data under different modeling hypotheses and inject them into a batch mode reinforcement learning agent to infer a robust controller.

We refer the interested reader to some of our recent publications concerning the applications of reinforcement learning to power system control [12, 13, 14, 15].

3.2.4 *Adaptive and distributed control*

In the context of on-line learning, a reinforcement learning agent continuously collects four-tuples at each time step and can infer from the associated rewards and successor states information about the real system performance and adapt its control policy to it. Furthermore, if there are several control agents using reinforcement learning connected to a single system, they can learn all in parallel and each one of them can adapt its performance progressively, hopefully leading to some kind of coordinated distributed control. If possible, it is of course advised to pre-train such multi-agent systems using an off-line simulator before plugging them on a real system.

3.3 *New developments in automatic learning algorithms*

3.3.1 *Supervised learning*

From a practical viewpoint, a very significant progress in the recent years concerns the design of new supervised learning algorithms which are able to cope with extremely high-dimensional input-spaces. In particular support-vector machines [16] are a kind of kernel-based model developed and extensively studied in the neural network community, whereas ensembles of decision/regression trees have been developed in the machine learning and statistics communities [17, 18]. The main motivation for the development of such algorithms was to cope with time-series, text and image classification problems where the number of ground variables (attributes) is typically very high (e.g. a 100×100 color image is basically represented by $100\times100\times3$=30000 input variables) which makes the use of classical supervised neural network or decision tree algorithms fruitless.

Within the context of this paper we advocate the use of the novel algorithms based on ensembles of extremely randomized trees [19, 20]. Indeed, these are rather efficient and very flexible and have been coupled successfully with reinforcement learning according to the iterative Q-function fitting approach described above. They have yielded state of the art performances both in the context of reinforcement learning benchmarks [8] and in the context of supervised classification and regression benchmarks [21]. The principle of these methods consists of building a set of randomized decision or regression trees from a training set and then to construct a synthetic model which aggregates the predictions of these trees, as an average over the ensemble of trees of the output variable.

3.3.2 *Reinforcement learning*

Reinforcement learning is presently a rather active research field and a lot of progress has to be expected from the ongoing work. A significant progress in the recent years stems from the possibility to apply these methods now to problems with large and continuous state spaces, because they have been able to exploit modern non-parametric supervised learning algorithms [3]. Convergence proofs are being provided in more general situations than the classical finite state space assumption [22] and also non-asymptotic reinforcement learning theory is being developed [23]. Another hot research topic concerns the theoretical analysis of these methods in the context of multi-agent systems [24].

4 AN APPLICATION EXAMPLE

It not possible to demonstrate here all capabilities of the algorithms discussed in the previous section. To illustrate our discussions we apply the fitted Q iteration algorithm to control a TCSC device aimed to damp inter-area oscillations in the 4-machine test system [25] shown in Fig. 1. The TCSC block diagram is given in Fig. 2.

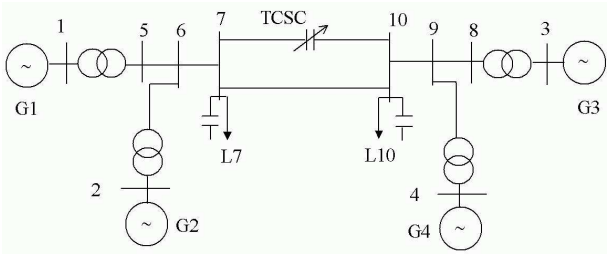

Figure 1: Four-machine test system

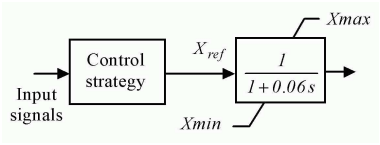

Figure 2: Block diagram of the TCSC

The system exhibits one inter-area mode of approximately 0.72 Hz. Observability analysis [25] reveals that the mode is most observable in current magnitude in the line connecting nodes 6 and 7 (Fig. 1) and among local variables (local to the TCSC controller) in active power flow through the TCSC. We consider two control schemes: one relying on local inputs only and another relying on local and remote inputs [26, 27].

4.1 *Four-tuples generation, state and reward definition*

To collect four-tuples we have considered 1000 scenarios for both control schemes. Each scenario starts with the power system being at rest and is such that at 1 s a self-clearing short circuit at bus 7 occurs. The fault duration is chosen at random in the interval [0,200] ms. A scenario stops either when instability is reached or when t is greater than 60 s. The period between two samplings is chosen equal to 50 ms which means that we generated roughly 1,100,000 four-tuples for each control scheme.[3]

In the control scheme that relies on local measurements we define the pseudo-state that will be used inside the algorithm by the following expression

$$x_k = (P_{ek}, P_{ek-1}, P_{ek-2}, u_{k-1}, u_{k-2}). \qquad (10)$$

The aim of the control is to maximize damping of the electrical power oscillations in the line. Thus, we define the reward by

$$r_k = \begin{cases} -|P_{ek} - \overline{P_e}| \text{ if } |P_{ek}| \leq 250MW \\ -1000 \text{ if } |P_{ek}| > 250MW \end{cases} \qquad (11)$$

[3] We do not collect four-tuples during the fault period because the TCSC is not supposed to control the faulted system.

where $\overline{P_e}$ represents the steady-state value of the electric power transmitted through the line, and the condition is used to detect instability.

In the control scheme that relies on a synthetic input signal (local: active power flow in the line, and remote: current magnitude in the line 6-7) the pseudo-state is defined by expression

$$x_k = (P_{ek}, P_{ek-1}, I_{(6,7)k}, I_{(6,7)k-1}, u_{k-1}). \quad (12)$$

and the reward by

$$r_k = \begin{cases} -|I_{(6,7)k} - \overline{I_{(6,7)}}| \text{ if } |I_{(6,7)k}| \leq 2.3kA \\ -1000 \text{ if } |I_{(6,7)k}| > 2.3kA \end{cases}. \quad (13)$$

In either case, the control set is discretized in four values equal to $U = \{-61.57, 41.05, 20.03, 0.0\}$. The training scenarios are generated by applying a randomly chosen control signal at each time step in the simulator.

4.2 The value of training parameters

For computational reasons, we have discretized the electrical power and current in 100 values within their interval of variation, although this is not necessarily optimal. The discount parameter γ was chosen equal to 0.95. Within the fitted Q iteration framework we have used as supervised learning method the Extra-Trees method of [21]. It has three parameters M (the number of trees that are built), K (the number of random cut-directions considered at each tree node), and n_{min} (the minimum number of elements to split a node) that have been chosen here respectively equal to 50, the dimensionality of the input space, and 2.

4.3 Simulation results

Fig. 3 represents the evolution of generator 1 internal angle relative to generator 3 when the system is subjected to a 100 ms self-clearing short-circuit for three cases: uncontrolled (dashed), when the TCSC is controlled by a "classical" controller (solid-thin), and when the TCSC is controlled by means of fitted Q iteration algorithm (local measurements) after 1000 learning scenarios (solid-thick).

To design the "classical" controller we used the procedure of [26], yielding the following transfer function

$$K(s) = \frac{100s(1+0.12s)^2}{(1+s)(1+0.6s)^2(1+0.06s)}. \quad (14)$$

This transfer function is between the input (active power flow in the line) and the TCSC reference reactance.

To assess the robustness of the proposed control, the learned control law is used to control the system when subjected to a different fault scenario (100 ms short-circuit followed by tripping the line 7-10). The system response is illustrated in Fig. 4 (solid-thick) together with uncontrolled system response (solid-thin). In spite of the change in system configuration, the controller succeeds to control efficiently the system being subjected to the "unseen" scenario.

Figure 5 represents the controlled system response to the self-clearing short-circuit when a local and a remote signal are used together as input to the fitted Q iteration algorithm (solid-thick), together with the learned controller using local measurements only (dashed). It illustrates also the system response (solid-thin) when a communication delay of 50 ms (modeled by first-order Padé approximation [27]) is imposed on the remotely acquired signal. We notice that this (reasonable) delay does not have significant detrimental effect to the designed controller.

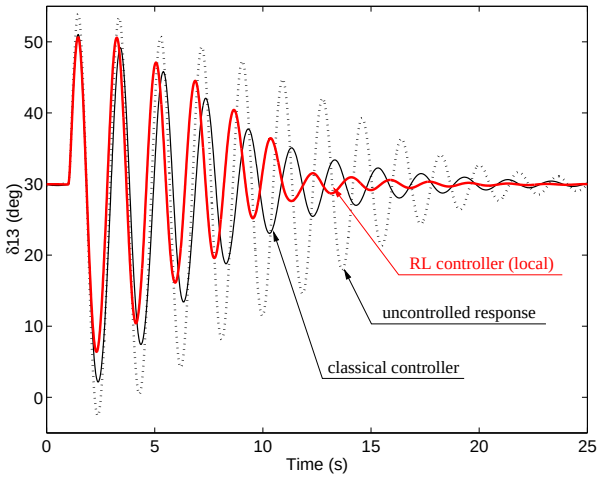

Figure 3: The system responses to 100 ms, self-clearing, short circuit

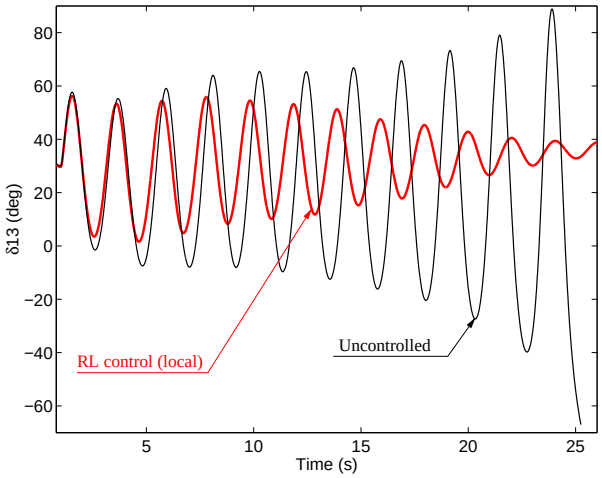

Figure 4: The system responses to 100 ms short circuit followed by tripping the line

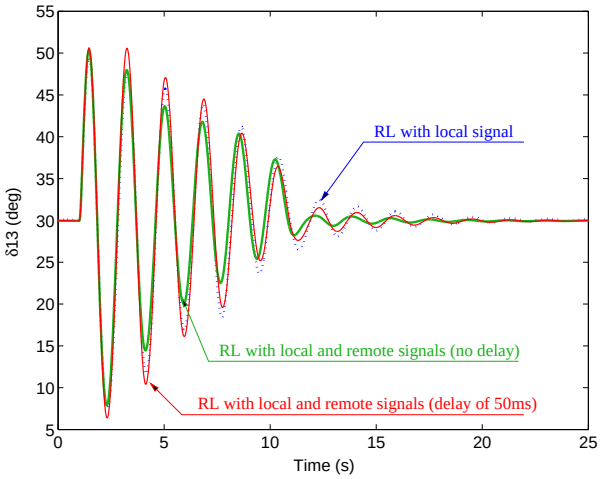

Figure 5: The system responses when controlled by controller using local and remote signals, only local signal and with communication delay

5 CONCLUSIONS

In this paper we reviewed recently developed automatic learning methods from the viewpoint of optimal control. In particular, supervised learning based on ensembles of randomized trees as well as the fitted Q iteration method provide a class of *anytime* algorithms, which gracefully improve the quality of their output as a function of increased computing power and data. Their combination leads to a very powerful framework for approaching optimal control and sequential decision problems. As an alternative to analytical optimization methods, they have several significant advantages. First of all, they make only weak assumptions about the properties of the system and control agents, and are therefore applicable to a very broad class of practical problems. Secondly, they can be applied off-line or on-line and may exploit data provided by system simulators or collected by measurements on the actual system. Contrary to many analytical optimization techniques, they are able to handle uncertainties and partial observability. Finally, while the amount of training data needed to obtain good performances obviously depends on the problem specifics, the computational complexity of these methods increases only slowly (roughly linearly) with problem dimensionality and training data size.

We have discussed the application of the framework to power system problems, and provided simulation results obtained on an academic but non-trivial power system control problem. We thus hope that this paper will help to better understand the potentials of automatic learning in power system decision and control and to foster further research and applications in this context, so as to take advantage of the ongoing progress in computer science.

ACKNOWLEDGMENTS

Damien Ernst acknowledges the support of the Belgian FNRS (Fonds National de la Recherche Scientifique) where he is a post-doctoral researcher.

REFERENCES

[1] D. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, MA: Athena Scientific, 2000, vol. I.

[2] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[3] D. Ernst, P. Geurts, and L. Wehenkel, "Iteratively extending time horizon reinforcement learning," in *Proc. of the 14th European Conference on Machine Learning*, Dubrovnik, Croatia, September 2003.

[4] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice - A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[5] L. Wehenkel, "Whither dynamic congestion management ?" in *Proc. of Bulk Power System Dynamics and Control - VI*, Cortina d'Ampezzo, Italy, August 2004.

[6] ——, *Automatic Learning Techniques in Power Systems*. Kluwer Academic, 1998.

[7] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, Cambridge University, England, 1989.

[8] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *To appear in Journal of Machine Learning Research*, vol. 6, pp. 503–556, April 2005.

[9] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: a Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[10] R. Sutton and A. Barto, *Reinforcement Learning. An Introduction*. MIT Press, 1998.

[11] D. Ernst, "Near optimal closed-loop control. Application to electric power systems," Ph.D. dissertation, University of Liège, Belgium, March 2003.

[12] D. Ernst, M. Glavic, and L. Wehenkel, "Power system stability control: reinforcement learning framework," *IEEE Transactions on Power Systems*, vol. 19, no. 1, pp. 427–435, February 2004.

[13] D. Ernst and L. Wehenkel, "FACTS devices controlled by means of reinforcement learning algorithms," in *Proc. of PSCC'2002*, Sevilla, Spain, June 2002.

[14] M. Glavic, D. Ernst, and L. Wehenkel, "Combining a stability and a performance oriented control in power systems," *IEEE Transactions on Power Systems*, vol. 20, no. 1, February 2005, 2 pages.

[15] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel, "Approximate value iteration in the reinforcement learning context. Application to electrical power system control," *Submitted for publication*, 2005, 30 pages.

[16] C. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. MIT Press, Cambridge, MA, 2000.

[17] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms : bagging, boosting, and variants," *Machine Learning*, vol. 36, pp. 105–139, 1999.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2001.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[20] P. Geurts, "Contributions to decision tree induction: bias/variance tradeoff and time series classification," Ph.D. dissertation, University of Liège, Belgium, May 2002.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Submitted for publication*, 2004, 35 pages.

[22] D. Ormoneit and S. Sen, "Kernel-Based Reinforcement Learning," *Machine Learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[23] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *Journal of Machine Learning Research*, vol. 5, pp. 1–25, 2003.

[24] J. Hu and M. P.Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, no. 4, pp. 1039–1069, 2003.

[25] P. Kundur, *Power System Stability and Control*. McGraw Hill, 2000.

[26] G. Rogers, *Power System Oscillations*. Kluwer Academic, 2000.

[27] J. H. Chow, J. J. Sanchez-Gasca, H. Ren, and S. Wang, "Power system damping controller design using multiple input signals," *IEEE Control Systems Magazine*, pp. 82–90, August 2000.