Knowledge-Based Systems 144 (2018) 9–15

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Tackling the start-up of a reinforcement learning agent for the control of wastewater treatment plants

Félix Hernández-del-Olmo[a,*], Elena Gaudioso[a], Raquel Dormido[b], Natividad Duro[b]

[a] Department of Artificial Intelligence, National Distance Education University (UNED), Madrid 28040, Spain
[b] Department of Computer Sciences and Automatic Control, National Distance Education University (UNED), Madrid 28040 Spain

ARTICLE INFO

Article history:
Received 22 April 2017
Revised 13 December 2017
Accepted 15 December 2017
Available online 16 December 2017

Keywords:
Reinforcement learning
Wastewater systems
Intelligent agent
Adaptive control

ABSTRACT

Reinforcement learning problems involve learning by doing. Therefore, a reinforcement learning agent will have to fail sometimes (while doing) in order to learn. Nevertheless, even with this starting error, introduced at least during the non-optimal learning stage, reinforcement learning can be affordable in some domains like the control of a wastewater treatment plant. However, in wastewater treatment plants, trying to solve the day-to-day problems, plant operators will usually not risk to leave their plant in the hands of an inexperienced and untrained reinforcement learning agent. In fact, it is somewhat obvious that plant operators will require firstly to check that the agent has been trained and that it works as it should at their particular plant. In this paper, we present a solution to this problem by giving a previous instruction to the reinforcement learning agent before we let it act on the plant. In fact, this previous instruction is the key point of the paper. In addition, this instruction is given effortlessly by the plant operator. As we will see, this solution does not just solve the starting up problem of leaving the plant in the hands of an untrained agent, but it also improves the future performance of the agent.

1. Introduction

Reinforcement learning (RL) is a machine learning paradigm where the agent learns to do better in its environment by interacting with it [29]. RL has been applied to different domains which include: medical applications such as optimization of anemia treatment [12], control of blood glucose variability [10], robotics [27], operations research (e.g. optimization the pricing policy of a cloud service provider [36] or web services [32]). RL has also been successfully applied to the intelligent control of processes. In this domain we could distinguish between process stabilization [30,37,38], process tracking for fault-tolerant controllers (FTC) [17,33,39] or process optimization [1,13]. The latter is the domain we focus on for the control of WasteWater Treatment Plants.

The main function of WWTPs is to provide humans and industries mechanisms for disposing effluents to protect the natural environment. Since WWTPs are significant energy consumers, optimizating them implies cutting operating costs and effluent fines, while rising to the challenges of water quality, sustainable development and even stringent regulations. Traditionally, WWTPs are controlled using standard control techniques [2], fuzzy control [3] or even artificial neural networks [15]. RL has also been successfully used in WWTPs. For example, in [23] a RL approach is proposed in order to increase methane production during anaerobic digestion of wastewater sludge. In [19], we improve the energy and environmental efficiency of a WWTP in N-Ammonia removal proccess by means of an RL agent.

Nevertheless, the high volume of data needed when the agent learns *online* –when the agent learns at the same time it is interacting with its environment– is currently one of the most important limitations for the application of RL [11]. Notice that, different from other kind of machine learning paradigms, there is no supervisor to *label* the data, either for learning or for evaluating the learning. Instead, only a signal that comes from the environment, the reinforcement, tells the agent how it is doing its job in the log run.

In the case of the control of a WWTP, the reinforcement signal can be a measure that combines energy and environmental efficiency [20–22]. In this case, plant operators are in charge of setting up this measure and tunning up the control (or RL agent control) of their WWTP. Nevertheless, as it may seem obvious, operators require that the application works from the very beginning. The problem we face is summarized as follows: how to get these interactive data if we cannot let the agent act on the plant first.

The solution we propose in this paper consists of applying a first stage of *instruction time*: a time in which the agent learns

* Corresponding author.
E-mail address: felixh@dia.uned.es (F. Hernández-del-Olmo).

https://doi.org/10.1016/j.knosys.2017.12.019
0950-7051/

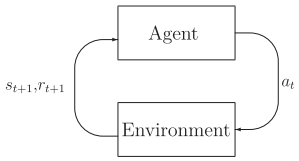

Fig. 1. General schema of a reinforcement learning task.

from the actions taken by the plant operator. After this instruction time, we leave the agent to learn by doing, this is, by interacting with the plant by means of the knowledge acquired during the first instruction stage.

The rest of this paper is organized as follows. Section 2 presents the required reinforcement learning background to grasp the main ideas of this paper. Section 3 presents a brief explanation of our previous work [20] since this paper is a continuation of it. Section 4 focuses on the solution proposed. Section 5 shows the results obtained when we apply the solution proposed and compare them with our previous work. Finally, in Section 6 we discuss these results and some work we have left to do for the near future.

2. Reinforcement learning background

Reinforcement learning (RL) algorithms are based on agent's interaction with its environment. The environment is defined as any external condition that cannot be changed directly by the agent [29], but can be changed through its actions. In fact, this interaction is usually represented as in Fig. 1.

2.1. Elements of reinforcement learning

The usual way the environment is modeled in RL is by means of Markov decision processes (MDP). Here, the MDP environment is modeled as (i) a space of states S, (ii) a space of actions $A(s)$ that can be done over this environment, given that the environment is in state s, and (iii) a set of transition probabilities from one state s to another state s' once the agent has executed action a over this environment $P(s'|s, a)$ besides (iv) the expected reward to be obtained from this environment $E\{r|s', a, s\}$ when changing from state s to state s' having executed action a [29].

Once the agent has this model of the environment, the optimal policy $\pi(s, a)$ can be solved by several methods, for instance dynamic programming [4]. However, if the model of the environment is not provided to the agent, it can still learn this model by means of the so called *model-free* RL methods [16]. Thus, with these model-free RL methods, the agent must interact with its environment so as to get, step by step, the model of the environment as well as the optimal policy to act upon it.

More specifically, the agent interacts with its environment making decisions according to its observations, via perception and action. At each step t, the agent observes the current state of the environment s_t and chooses an action to execute, a_t. This action causes a transition between states and the environment provides a new state s_{t+1} and a reward r_{t+1} to the agent. The ultimate goal of the agent is to choose those actions that tend to increase its *return*: the long-term sum of the future reward values r_t. This return, in a continuous environment, is usually set as $R_t = \sum_{t'=t}^{\infty} \gamma^{(t'-t)} r_t$, where $0 < \gamma < 1$ stands for a kind of *Optimization Horizon* (OH, as we will see later). In other words, the higher the γ (up to 1), the further the future time considered into the return R_t. In addition, if the application has a large number of states or if it is continuous, we need to approach the problem by means of function approximation [7].

Finally, in the case of the WWTP domain, a complete MDP model is not usually available, thus, a model-free RL approach is required. That is why an agent has to interact with the environment to estimate it. Notice, however, that the goal does not consist of getting the complete estimation of the MDP, but just to calculate the best policy π. This is, the one that gets a higher return. To this end, in [20] we took advantage of the widely known method called *policy iteration* [7,29].

2.2. Reinforcement learning by policy iteration

The objective of reinforcement learning consists of searching for the policy that provides the agent with the action that, for a given state, will carry the agent to a next state maximizing the next and the following rewards (the agent's return). This is called the *optimal policy* $\pi^*(s)$ and this policy is unique [4]. In reinforcement learning, to find this $\pi^*(s)$, the agent must interact with the environment by means of some previous non-optimal policies $\pi_t(s)$.

For the sake of accuracy, we will define some new concepts in the next paragraphs. The return obtained by a policy π interacting in an MDP when the agent starts at state s is defined as $R^\pi(s)$. When working with a known MDP, this return is usually called $V^\pi(s)$ and it is what we need to solve the problem in a *dynamic programming problem* [4]. However, in reinforcement learning (where we do not have the complete MDP available), we must work with the so called Q values: $Q^\pi(s, a)$. Each $Q^\pi(s, a)$ value is defined as the return obtained when the agent follows the policy π starting at the state s and taking a as the next immediate action.

Policy iteration consists of an iterative process in which the policy π_t followed by the agent at time t is monotonically improved at each step letting *policy evaluation* and *policy improvement* processes interact [29]. Policy evaluation consists of calculating a new estimation of $Q^{\pi_t}(s, a)$ after each interaction with the environment under this policy π_t. Policy improvement consists of getting the new policy π_{t+1} from the last policy evaluation $Q^{\pi_t}(s, a)$. Algorithm 1 shows the pseudocode that details this explanation.

Algorithm 1 Reinforcement learning by policy iteration.

```
1:  function PI(π_0, s_0)
2:      t ← 0
3:      π ← π_0
4:      π' ← π
5:      a ← π(s_0)
6:      repeat
7:          s_{t+1}, r_{t+1} ← Environment(s_t, a)              ▷ Interaction
8:          a' ← π'(s_{t+1})
9:          Q^{π'}_{t+1} ← UpdateQ(Q^π_t, s_t, a, r_{t+1}, s_{t+1}, a')   ▷ Policy
    evaluation
10:         π ← π'
11:         for ∀s do
12:             π'(s) ← arg max_a[Q^π_{t+1}(s, a)]   ▷ Policy improvement
13:         t ← t + 1
14:         a ← a'
15:     until π' = π
    return π* = π
```

Fig. 2 depicts the idea behind.

Finally, notice that in Algorithm 1 the function `UpdateQ` is still to be defined. In the next section we will show the two major ways of updating Q: the so-called *on-policy* and *off-policy* control methods.

2.3. On-policy and off-policy control methods

In this section we will focus on line 9 of Algorithm 1: policy evaluation. This point is a very important one, because the way the agent updates Q can make it converge to Q^* or, instead, make it diverge badly. Thus, there are two major approaches: the on-policy

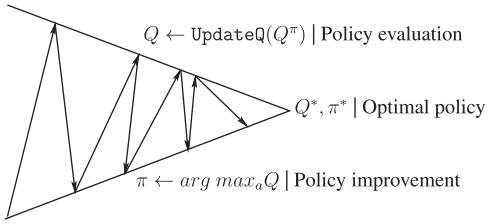

Fig. 2. Policy iteration: two processes, policy evaluation Q above and policy improvement π below, interact one with each other so as to get closer and closer to their optimal values.

SARSA like algorithms [28] and the off-policy Q-learning like algorithms [34].

The on-policy SARSA rule updates Q as in Eq. (1). The name SARSA of this updating rule comes from the set of events it uses: $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. The convergence properties of the Algorithm 1 (with the SARSA updating rule) depend strongly on the nature of every π_t [29]. In fact, this feature is very useful when we want the agent not only to benefit from learning the optimal policy π^* in the long run, but also to get the best short-term return while following the current policy π_t. However, this fact can also prevent the agent from reaching the optimal policy π^* if it sticks for too long to this somewhat better short term policy. Moreover, if the agent does not improve the policy quick enough over time, it may never get the optimal π^* or it could even diverge badly [28]. This is where the name *on-policy* comes: the algorithm only takes into account not only the environment where the agent is interacting with, but also the policy π_t where the agent is currently riding on.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\ Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

The off-policy Q-learning rule updates Q as in Eq. (2). The Q-learning algorithm was proposed earlier than SARSA, maybe because it is less difficult to probe its convergence [35]. In fact, in this case, Q approximates Q^* directly, regardless of the policy followed. This dramatically simplifies the analysis of the algorithm and it enabled early convergence proofs [29,34,35]. Moreover, this feature can make the agent improve its behavior faster than SARSA whenever we cannot take control of the current agent's policy π_t. In fact, this happens because –with Q-learning– the agent's learning does not rely –as directly as with SARSA– on the current policy π_t. However, despite it and different from SARSA, this fact can make the agent behave worse in the short term because it does not take into account the on-policy π_t that the agent is currently following. In fact, this is why it is called *off-policy*: the algorithm just takes into account the environment where the agent is interacting with, but not the policy π_t that the agent is currently following.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\ max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

Despite the advantages of the on-policy SARSA-like algorithms described before, for the purpose of improving our agent's behavior while following the plant operator policy (the actions taken by the operator), we found off-policy control methods more suitable. In fact, we want to apply a previous *instruction time* in order to teach the agent to act better before it can freely operate the WWTP (see Section 1). Therefore, in Section 4, we will introduce this *instruction time* as a light modification of the Algorithm 1 (with an off-policy updating rule) trying to change its good nature as little as possible.

Nevertheless, let us first explain our application and the problem we must face in a more detailed way. In the next section, we will explain how we try to minimize the operation cost of a WWTP plant by means of a RL agent. To this end, we will detail how we try to keep the water quality (effluent from the WWTP) in a legal range while trying to save the highest amount of energy.

3. Energy and environmental efficiency in wastewater treatment plants by means of reinforcement learning

Energy and environmental efficiency are critical aspects in WWTPs. From an environmental point of view, energy efficiency is an important issue as different reports show about best practices on this subject [6]. In fact, several approaches have been proposed in order to improve the WWTPs in terms of lowering energy consumption [8,9,24].

But also, from an economic point of view, efficiency in the energy use can lower operating costs while keeping, or even rising, the water quality of the WWTP effluent. In particular, in WWTPs, a strict regulation has been imposed on the nitrogen levels of the effluent. In order to get rid of this nitrogen, active sludge process (ASP) with nitrification / denitrification stages is the most widely-used technique [18]. However, WWTPs are significant energy consumers in this ASP, especially for the N-ammonia removal.

In more detail, the common variable to control an ASP is the dissolved oxygen (DO) concentration [14]. DO level in the aerobic reactors has direct influence on the microorganisms' activity that are part of the active sludge. Aeration mechanisms supply oxygen to the sludge to increase DO so that organic matter is degraded and nitrogen concentration is lowered. Thus, the ASP is urged to keep a proper concentration of DO. Often, in many plants, DO concentration is kept high enough to ensure good effluent quality. However, this approach is expensive and it is therefore desired to operate the aerobic reactors of the plant at lower DO concentration to reduce energy consumption. Notice that the process for the N-ammonia removal is the most important energy consumer in a WWTP, being the responsible of nearly 50% of the energy consumption of the plant.

Therefore, in our WWTP, the objetive is, on the one hand, to satisfy the effluent requirements defined by local regulations to keep the total nitrogen under a limit [26,31] and, on the other hand, to keep maintenance and operation expenses as low as possible. These expenses are due to the disposal of the wasted sludge [25] and mainly due to the energy consumed by blowers (for aeration) and pumps in the WWTP. If we add all these expenses up, we will obtain a final and total operation cost $OC(t)$ in money units. In our application these units are *in euros*. This operation cost is the one we use in our plant as the reinforcement signal r_t for the agent, and this signal is the one we intend to minimize. To this end, we defined our $Q(s, a)$ value as a combination of several non-linear interactions between NH_4 and O_2 in order to grasp much of its underlying non-linearity. In more detail [19]:

$$Q(s = \{NH_4, O_2\}, a) = \sum_{i=0}^{N_{nh_4}} \sum_{j=0}^{N_{o_2}} \alpha_{ij}[a] * NH_4^i * O_2^j \quad (3)$$

N_{nh_4} and N_{o_2} parameters depend on the complexity we want to add to the model. In this particular experiment we set $N_{nh_4} = 2$ and $N_{o_2} = 2$, because there is a trade off between model complexity and hardware resources. Also, parameters $\alpha_{ij}[a]$ are to be learned by the agent by means of neuro-dynamic programming techniques [4,5,7,21].

In Fig. 3 we show the control we simulated in our WWTP. Let us first recall from Algorithm 1 that the time t is sliced. In our application the size of this time slice is 15 min and $\gamma = 0.92105$. Now, if we focus again on Fig. 3, we can observe the following signals exchanged between the environment and the agent:

(i) the state s_{t+1}: the oxygen $O_2(t+1)$ and the N-ammonia $NH_4(t+1)$;
(ii) the reinforcement r_{t+1} given by the operation cost of the plant $OC(t+1)$ after performing
(iii) a previous action a_t: the $DO_{set-point}(t)$.

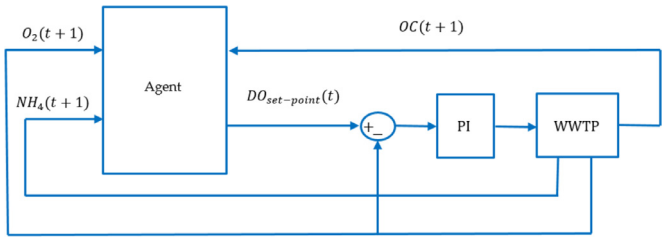

Fig. 3. Control loop to autonomously set the DO set-point using the RL agent.

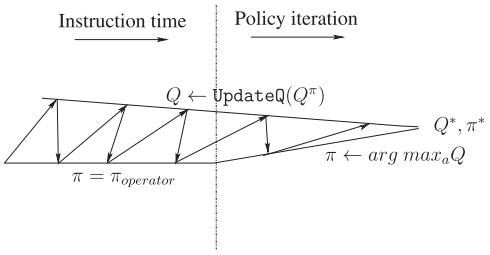

Fig. 4. Instruction time, then policy iteration: during the instruction time, the agent follows the operator's policy. In spite of it, the off-policy evaluation of the policy allows a monotone improvement of Q. After the instruction time, the agent follows the usual policy iteration.

In fact, the output of the agent, the action a_t, consists of one out of three values for the DO set-point: 1.20 mg/L, 1.85 mg/L, and 1.5 mg/l. The final DO value is controlled by a common PID controller (see PI in Fig. 3). Notice that we chose these three values of DO set-points because they are, respectively, the optimal DO set-points for each of the three possible weather conditions: dry-weather, rainy-weather and stormy-weather.

The results obtained with this RL agent (with these settings) showed significant improvement in the N-Ammonia removal process in a simulated WWTP [19]. However, in this previous work we only applied policy iteration, Algorithm 1, without any modification. Therefore, in this previous work the agent started to act on the plant from the very beginning. This is, the RL agent started to operate with zero knowledge about its particular WWTP. The objective of this paper is to check whether we can improve this behavior when the RL agent receives a first *instruction time* without any previous interaction with the plant. To this end, we will present a modification of Algorithm 1. We devote the next section to explain in detail this new algorithm.

4. Performance and starting-up improvement by means of initial instruction

In the last section we summarized the agent and the environment we are working on. Now we want to apply a previous *instruction time* in order to teach the agent to act better before it can freely operate the WWTP. In this section we will detail such an algorithm.

Firstly, let us introduce our modification to Algorithm 1. This modification introduces an *instruction time* during which the plant operator tells the agent what to do *effortlessly*, while she/he is controlling the plant. This way, during this *instruction time* the agent learns *in an off-policy way* while following the plant operator's policy. We will call this Algorithm *ITPI*, this is: *first Intruction Time, then Policy Iteration*. Algorithm 2 shows the pseudocode that details this proposal. Fig. 4 depicts the idea behind. Notice that in Algorithm 2 (*ITPI*) the initial policy is provided by the plant operator $\pi_{operator}$. Thus, there is no π_0 required.

Focusing on the ITPI Algorithm 2 we can see that during the instruction time, and thanks to the off-policy nature of the Q update, we take into account only the previous action a_t and the previous state s_t plus the *provoked* next state s_{t+1} and new reinforcement r_{t+1}. Notice that we do not consider the new action a' into the Q_t update. This allows the agent to *learn* the MDP associated to its environment even though it is following the operator's policy $\pi_{operator}$. Notice that the convergence of ITPI is warranted by two facts: (i) the fact that PI converges [7,29] and (ii) the fact that PI always comes after the initial instruction *IT* phase (see Fig. 4).

In the next section we will show the experiments we conducted with Algorithm 2. We will see that this previous instruction learning time is very useful for the agent.

Algorithm 2 Instruction time before policy iteration.

function ITPI($\pi_{operator}, s_0$)
  $t \leftarrow 0$
  $\pi \leftarrow \pi_{operator}$
  $\pi' \leftarrow \pi$
  $a \leftarrow \pi(s_0)$
  while $t \in$ `InstructionTime` **do** ▷ Instruction time
    $s_{t+1}, r_{t+1} \leftarrow$ `Environment`(s_t, a) ▷ Interaction
    $a' \leftarrow \pi_{operator}(s_{t+1})$
    $Q_{t+1}^{\pi_{operator}} \leftarrow$ `UpdateQ`$(Q_t^{\pi_{operator}}, s_t, a, r_{t+1}, s_{t+1})$ ▷ *off-policy* update Q
    $t \leftarrow t+1$
    $a \leftarrow a'$
  repeat ▷ Policy Iteration
    $s_{t+1}, r_{t+1} \leftarrow$ `Environment`(s_t, a) ▷ Interaction
    $a' \leftarrow \pi'(s_{t+1})$
    $Q_{t+1}^{\pi'} \leftarrow$ `UpdateQ`$(Q_t^{\pi}, s_t, a, r_{t+1}, s_{t+1})$ ▷ *off-policy* evaluation
    $\pi \leftarrow \pi'$
    for $\forall s$ **do**
      $\pi'(s) \leftarrow$ `arg max`$_a[Q_{t+1}^{\pi}(s,a)]$ ▷ Policy improvement
    $t \leftarrow t+1$
    $a \leftarrow a'$
  until $\pi' = \pi$
return $\pi^* = \pi$

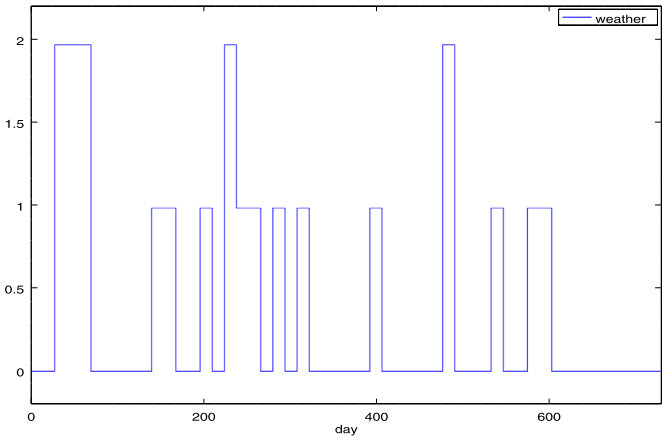

Fig. 5. Weather over the first 2 years. Notes: 0, dry weather; 1, rain weather; 2, storm weather.

5. Experiment and results

The weather in the experiment varies in the following way: it rains (randomly) 20% of the time and it storms (also randomly) 10% of the time; thus, the weather is dry 70% of the time. In Fig. 5, we show the weather day by day over the first year.

Now, let us recover from Section 3 that we have three optimal DO set-points (for each weather condition): 1.20 mg/l for

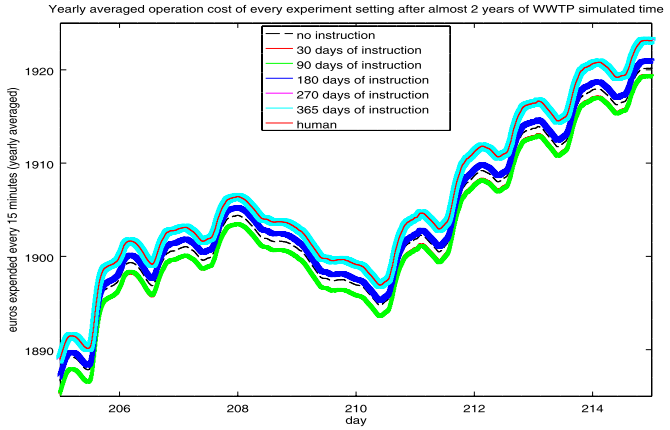

Fig. 6. Yearly averaged operation cost of each of the 7 settings; less than a year of simulated time. Notice that the lines for "30 days of instruction" and "90 days of instruction" are both so close that they are hardly distinguishable. The same occurs with the lines for "270 days of instruction", "365 days of instruction" and "human" because they are all the same (notice that day<270, so they are following $\pi_{operator}$).

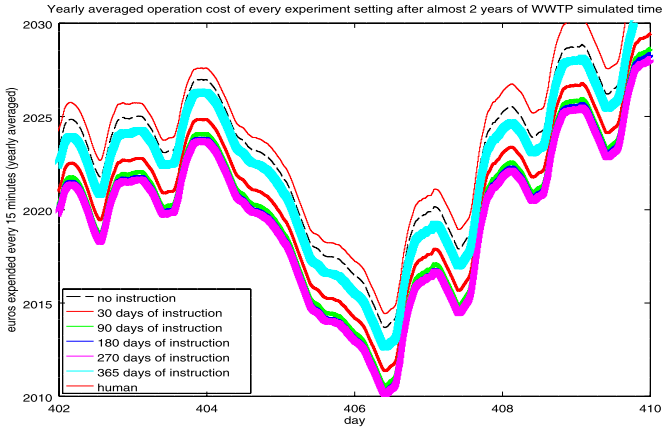

Fig. 7. Yearly averaged operation cost of each of the 7 settings; around a year of simulated time.

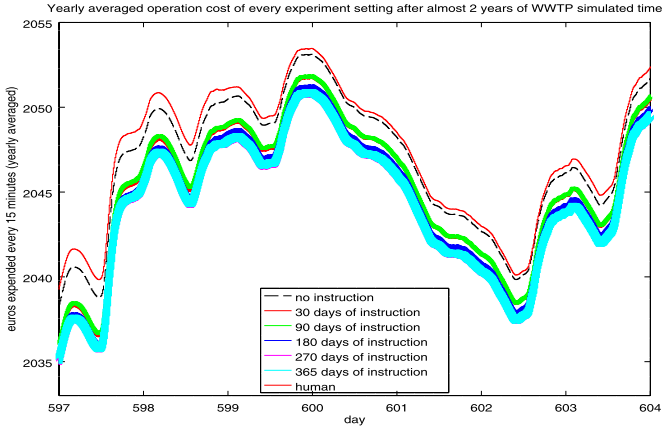

Fig. 8. Yearly averaged operation cost of each of the 7 settings; almost 2 years of simulated time.

dry-weather, 1.85 mg/l for rain-weather, and 1.5 mg/l for stormy-weather. With respect to our RL agent (or *agent* for short), we will focus on two main results: (i) learning speed of the agent, (ii) energy and environmental efficiency (= operation cost improvement) obtained by means of its behavior.

We will compare the agent's behavior with different times of instruction against the operator's behavior as a test base. Thus, we simulated a plant operator together with the plant by changing the DO set-point to the optimal position whenever the weather changes. Something important to notice is that, different from the plant operator, the RL agent doesn't have access to the weather condition information, it only accesses the oxygen and the N-ammonia inputs (see Fig. 3). In fact, notice that the operator not only has complete access to the real weather conditions of the plant, he also takes into account this information immediately, and, also immediately (of course ideally), the operator takes care of the oxygen set-point of the plant.

As we said in Section 4, instruction to the agent is given through the operator's behavior $\pi_{operator}$. Thus, during the instruction time, the agent's behavior cannot be distinguished from the operator's behavior. We will compare the following settings: (i) no instruction at all, (ii) 30 days of instruction, (iii) 90 days of instruction, (iv) 180 days of instruction, (v) 270 days of instruction, (vi) a year of instruction, (vii) operator's behavior. In order to get relevant results, we run a simulation of the agent in the WWT simulated plant for 2 (simulated) years. Note that the instruction is given to the agent during the first days of the first year.

In order to show the results, we will use 3 kind of graphs. Firstly, a yearly moving average of the operation cost will show how the optimization efficiency of each of the 7 agent's settings evolves with time. Secondly, so that we can dive into the agent's behavior, we will show the agent's action (by means of its set-point) and its environment reaction (by means of the N-ammonia) in a selected time-frame window. Finally, to sum up, we will show a comparison of the total saved cost of each agent's behavior against the operator's one.

The yearly moving average operation cost of these 7 settings is shown in 3 time-frame windows from less than a year time to almost 2 years time, see Figs. 6–8. Notice that the line width of each setting in the graphs is proportional to the days of instruction the agent received. Thus, "365 days of instruction" is the widest line. Each time-frame is chosen to have a change of weather in the middle.

Focusing on Fig. 6, we can check that, because we are in an early time-frame (less than a year), some settings, 270 and 365 days of instruction, are still bound to the operator's behavior. Other setting, 180 days of instruction, did not have time to learn by itself (interacting with the WWT plant) yet. The rest of settings, 90 and 30 days of instruction (they both so close together that they are hardly distinguishable) have finally been able to overtake the no-instruction setting.

Going forward in (simulated) time, let us locate ourselves after the first year, see Fig. 7. At this time we can see that, finally, no agent's setting is bound to the operator's behavior. However, "365 days of instruction" did not have enough time to learn by itself (interacting with the WWT plant) yet. Notice that if we sort the operation costs from the worst setting to the best one, we get: operator, no-instruction, 365, 30, 90, 180 and 270. It is true that 90, 180 and 270 are together very close to each other. However, this little difference will be detected in the graph displayed with color.

Almost at the end of the first 2 years, see Fig. 8, "365 days of instruction" is in the group of the best: 180, 270 and 365. This group is followed by 90 and 30 days of instruction. Finally, no-instruction and operator are the ones with the worst operation costs.

Let us now understand why these differences in operation costs by diving into the agent's behavior. Focusing on Fig. 9, in the graph, the 180-days-of-instruction agent has had enough time to learn. Also, the weather changes from stormy to dry on day 406, and the operator behavior reflects it with its change of set-point from 1.85 mg/l to 1.2 mg/l on that day. Notice that the agent only minds the level of N-ammonia and oxygen (not the weather), so it does

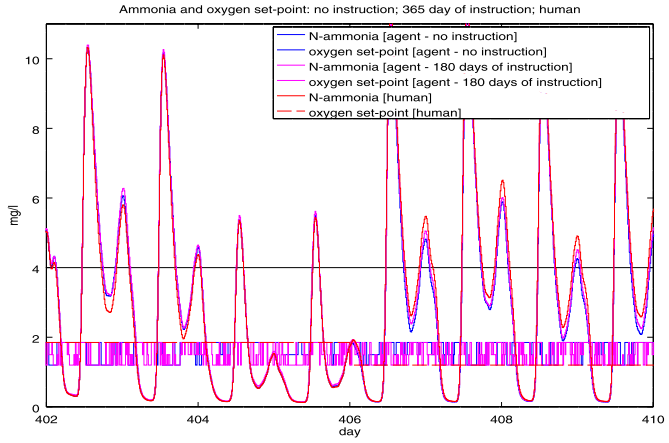

Fig. 9. Operator's behavior vs agent's behavior with and without instruction.

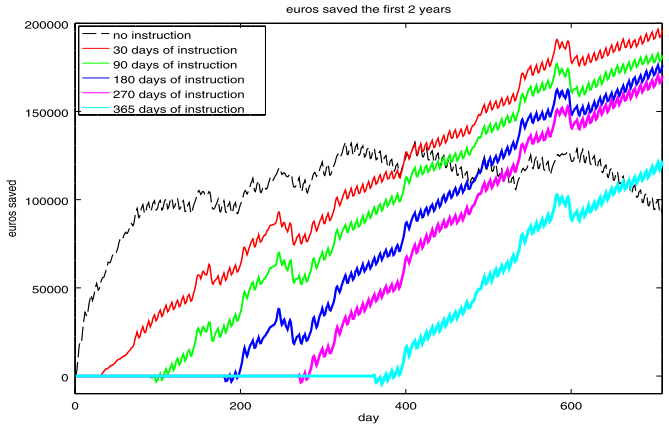

Fig. 10. Total operation cost saved by each agent's setting with regard to the operator for the first 2 years.

not present a substantial change of behavior on day 406. In fact, notice the more dynamic behavior of the agent against the operator's one. However, there are some differences between the 180-instructed agent and the agent no-instructed at all. For instance, between 405 and 406 days, there should be plenty of time to be relaxed because the levels of N-ammonia are low. Between these days, 180-instructed agent is near 1.2 mg/l, in order to save energy. It does not happen with the non-instructed agent (nor the operator, who is even less dynamic).

Finally, we would like to highlight and compare the total operation costs of each of the 6 agents' behavior. In Fig. 10, we show the added saved costs against the operator's operation cost for the first 2 years day after day. As the reader may see, the tendency is that the more instructed the agent, the steeper the graph of saved cost. However, because of being bound to the operators' behavior, the more instructed the agent, the later it starts interacting with the WWT plant and learning by itself. It happens that a more instructed agent will eventually catch up with a less instructed one, but we wonder: is it worth the time spent in instruction?

The answer looks clear from this experiment: it is. Even as few as 30 days of instruction are worth it. In fact, as it may be seen in Fig. 10, the non-instructed agent even degrades during the second year and even loses (against the operators' behavior) some of the costs saved over the first year. In fact, it looks as if the acquired knowledge were less solid than the one acquired when the agent is firstly instructed.

In the next section, we will highlight some of the main conclusions of these results.

6. Conclusions and future work

When the goal is to improve the energy and environmental efficiency and optimize the operation cost of an WWTP, it is important to do it quickly and autonomously so that the agent can adapt to changing environmental conditions with a minimal intervention of the plant operator.

In previous works, we presented a reinforcement learning approach to reduce the operation cost in an N-ammonia removal process in a WWTP. In this paper, we show that even an initial instruction of as few as 30 days improves dramatically the agent's learning speed and its final objective: reducing much more the operation cost of the WWT plant (see Fig. 10). Even more, we show that the longer the instruction time the better the operation cost (see Fig. 8).

However, we wonder if we should instruct the agent the longer the better. In fact, it looks clear that the more time the operator spends instructing the agent the less time the agent is learning by its own interaction with the WWT plant. Thus, we see this as a trade off between a firstly fruitful *instructed learning stage* and a more expensive, if non-optimal, but much more productive in the long run, *interactive learning* with the WWT plant.

In fact, it does not look easy to find the best time to let the agent *fly alone*. Perhaps, in this first approach, it seems that more than 30 days of instruction don't pay off the lack of time working and interacting with the WWT plant (see Fig. 10). Moreover, the earlier the agent starts working the earlier the WWT plant can start to benefit from it. Even more, as a deep detail, see Fig. 10 just at the end of each agent's instruction time. We can see there that the more instructed the agent, the harder it is for the agent to start getting a positive saved cost (starting, in fact, with a negative saved cost, see first non-instruction days of 90, 180, 270 and 365 agent in Fig. 10). In fact, it looks as if the agent felt stranger with the interaction with the plant as more and more instruction it has previously received. Anyway, eventually, the more instructed agent starts to learn faster and more effectively than the less instructed one after a while.

Therefore, we are currently working around all these questions. Among other things, to get faster learning speeds with even less instruction time.

Acknowledgments

This work was supported in part by the UNED project GID2016-6-1, the Spanish Ministry of Economy and Competitiveness under Projects DPI2014-55932-C2-2-R and ENE2015-64914-C3-2-R and FEDER funds.

References

[1] S. Adam, L. Busoniu, R. Babuska, Experience replay for real-time reinforcement learning control, Trans. Sys. Man Cyber Part C 42 (2012) 201–212.
[2] L. Åmand, G. Olsson, B. Carlsson, Aeration control – a review, Water Sci. Technol. 67 (2013) 2374–2398, doi:10.2166/wst.2013.139.
[3] P. Baroni, G. Bertanza, C. Collivignarelli, V. Zambarda, Process improvement and energy saving in a full scale wastewater treatment plant: air supply regulation by a fuzzy logic system, Environ. Technol. 27 (2006) 733–746.
[4] D. Bertsekas, Dynamic Programming and Optimal Control, 2, third ed., Athenas Scientific, 2007.
[5] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, first ed., Athena Scientific, 1996.
[6] M. Brandt, R. Middleton, S. Wang, Energy efficiency in the water industry: A compendium of best practices and case studies UKWIR report 10/CL/11/3, 2010, Technical Report, Water Research Foundation: London, UK.
[7] L. Buşoniu, R. Babuška, B. De Schutter, D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, Boca Raton, Florida, 2010.
[8] B. Chachuat, N. Roche, M. Latifi, Dynamic optimisation of small size wastewater treatment plants including nitrification and denitrification processes, Comput. Chem. Eng. 25 (2001) 585–593.

[9] S. Cristea, C. de Prada, D. Sarabia, G. Gutiérrez, Aeration control of a wastewater treatment plant using hybrid nmpc, Comput. Chem. Eng. 35 (2011) 638–650.
[10] M. De Paula, G.G. Acosta, E.C. Martínez, On-line policy learning and adaptation for real-time personalization of an artificial pancreas, Expert Syst. Appl. 42 (2015) 2234–2255.
[11] M.P. Deisenroth, D. Fox, C.E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 408–423.
[12] P. Escandell-Montero, M. Chermisi, J.M. Martínez-Martínez, J. Gómez-Sanchis, C. Barbieri, E. Soria-Olivas, F. Mari, J. Vila-Francés, A. Stopper, E. Gatti, J.D. Martín-Guerrero, Optimization of anemia treatment in hemodialysis patients via reinforcement learning, Artif. Intell. Med. 62 (2014) 47–60, doi:10.1016/j.artmed.2014.07.004.
[13] P. Escandell-Montero, D. Lorente, J.M. Martínez-Martínez, E. Soria-Olivas, J. Vila-Francés, J.D. Martín-Guerrero, Online fitted policy iteration based on extreme learning machines, Knowl. Based Syst. 100 (2016) 200–211, doi:10.1016/j.knosys.2016.03.007.
[14] B. Holenda, E. Domokos, A. Redey, J. Fazakas, Dissolved oxygen control of the activated sludge wastewater treatment process using model predictive control, Comput. Chem. Eng. 32 (2008) 1270–1278.
[15] W. Hu, The application of artificial neural network in wastewater treatment, in: IEEE 3rd International Conference on Communication Software and Networks, IEEE, 2011, pp. 338–341.
[16] L. Kaelbling, M.L. Littman, A. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.
[17] L. Liu, Z. Wang, H. Zhang, Adaptive fault-tolerant tracking control for mimo discrete-time systems via reinforcement learning algorithm with less learning parameters, IEEE Trans. Autom. Sci. Eng. 14 (2017) 299–313.
[18] E. Metcalf, H. Eddy, Wastewater engineering: treatment and reuse, 2003.
[19] F. Hernández-del Olmo, E. Gaudioso, R. Dormido, N. Duro, Energy and environmental efficiency for the n-ammonia removal process in wastewater treatment plants by means of reinforcement learning, Energies 9 (2016) 1–17.
[20] F. Hernandez-del Olmo, E. Gaudioso, R. Dormido, N. Duro, Energy and environmental efficiency for the n-ammonia removal process in wastewater treatment plants by means of reinforcement learning, Energies 9 (2016), doi:10.3390/en9090755.
[21] F. Hernandez-del Olmo, E. Gaudioso, A. Nevado, Autonomous adaptive and active tuning up of the dissolved oxygen setpoint in a wastewater treatment plant using reinforcement learning, Syst. Man Cybern. Part C 42 (2012) 768–774.
[22] F. Hernandez-del Olmo, F. Llanes, E. Gaudioso, An emergent approach for the control of wastewater treatment plants by means of reinforcement learning techniques, Expert Syst. Appl. 39 (2012) 2355–2360.
[23] L. Pettigrew, A. Delgado, Neural network based reinforcement learning control for increased methane production in an anerobic digestion system, in: Proceedings of the 3rd IWA Specialized International Conference Ecotechnologies for Wastewater Treatment, 2016, pp. 1–4.
[24] S. Revollar, P. Vega, R. Vilanova, Economic optimization of wastewater treatment plants using non linear model predictive control, IEEE, 2015. System Theory, Control and Computing (ICSTCC), 2015 19th International Conference on, 583–588.
[25] J. Rojas, T. Zhelev, Energy efficiency optimisation of wastewater treatment: study of atad, Comput. Chem. Eng. 38 (2012) 52–63.
[26] P. Samuelsson, B. Halvarsson, B. Carlsson, Cost-efficient operation of a denitrifying activated sludge process, Water Res. 41 (2007) 2325–2332, doi:10.1016/j.watres.2006.10.031.
[27] H. Shahbazi, K. Jamshidi, A.H. Monadjemi, H. Eslami, Biologically inspired layered learning in humanoid robots, Knowl. Based Syst. 57 (2014) 8–27, doi:10.1016/j.knosys.2013.12.003.
[28] S. Singh, T. Jaakkola, M.L. Littman, C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, Mach. Learn. 38 (2000) 287–308.
[29] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, Mit Press, Cambridge, 1998.
[30] K. Vamvoudakis, Non-zero sum nash q-learning for unknown deterministic continuous-time linear systems, Automatica 61 (2015) 274–281.
[31] R. Vilanova, R. Katebi, N. Wahab, N-removal on wastewater treatment plants: A process control approach, J. Water Resource Prot. 3 (2011) 1–11.
[32] H. Wang, X. Wang, X. Zhang, Q. Yu, X. Hu, Effective service composition using multi-agent reinforcement learning, Knowl. Based Syst. 92 (2016) 151–168.
[33] Z. Wang, L. Liu, H. Zhang, G. Xiao, Fault-tolerant controller design for a class of nonlinear mimo discrete-time systems via online reinforcement learning algorithm, IEEE Trans. Syst. Man Cybern. 46 (2016) 611–622.
[34] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292.
[35] C.J.C.H. Watkins, Learning from delayed rewards, 1989 Ph.D. thesis.
[36] B. Xu, T. Qin, G. Qiu, T.-Y. Liu, Optimal pricing for the competitive and evolutionary cloud market, in: Proceedings of the 24th International Conference on Artificial Intelligence,In IJCAI'15, AAAI Press, 2015, pp. 139–145. http://dl.acm.org/citation.cfm?id=2832249.2832269
[37] H. Zhang, L. Cui, Y. Luo, Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network adp, IEEE Trans. Cybern. 43 (2013) 206–216.
[38] H. Zhang, Q. Wei, D. Liu, An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games, Automatica 47 (2011) 207–214.
[39] H. Zhang, Q. Wei, Y. Luo, A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm, IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 38 (2008) 937–942.