STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

A STUDY ON USING REINFORCEMENT LEARNING FOR TEMPORAL ORDERING OF BIOLOGICAL SAMPLES

IULIANA M. BOCICOR

Abstract. The temporal ordering of biological samples, with the goal of retrieving the temporal evolution of dynamic biological processes, is an important problem within bioinformatics. As the general temporal ordering problem has been proven to be NP-complete, various approximation and heuristic methods are developed to approach it. Reinforcement Learning is an approach to machine intelligence in which an adaptive system can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. This paper aims to investigate a reinforcement learning based approach to the temporal ordering problem and several variations to this approach, based on Q-Learning. The algorithms are experimentally evaluated on a time series gene expression data set and we provide analysis and comparisons of the obtained results.

1. Introduction

Reinforcement Learning [12] is an approach to machine intelligence in which an agent [11] can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time.

The biological *temporal ordering (TO) problem* is formulated as the problem of constructing a sorted collection of multi-dimensional biological data, collection that reflects an accurate temporal evolution of a certain biological process. The final goal is to find certain patterns in the input data that

Received by the editors: December 1, 2012.

2010 *Mathematics Subject Classification.* 68P15, 68T05.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8[**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. Bioinformatics, Temporal Ordering, Reinforcement Learning, Q-Learning.

vary over time and use them efficiently in order to be able to offer a proper characterization of the process in question.

In this paper we aim to investigate several variations to a reinforcement learning based approach for the *TO* problem, approach that we have previously introduced in [3]. The evaluations are made on a data set that was used in [3] and comparisons and analysis will be provided.

The rest of the paper is organized as follows. Section 2 introduces the biological *TO* problem, as well as an existing reinforcement learning based approach. A series of variations to this approach, more specifically to the underlying algorithm are presented in Section 3. Experimental evaluations, analysis and comparisons of the all the algorithms are given in Section 4. Section 5 outlines our conclusions and further work.

2. Background

In this section we will briefly present the *TO* problem, in a bioinformatics framework, then review some fundamental aspects related to the reinforcement learning based approach that we previously introduced for solving this problem [3].

2.1. The Temporal Ordering Problem.

The general *temporal ordering* problem has been tackled within multiple fields. In machine learning it is considered as important as the classification problem, given that, in certain cases, ordering a set of instances can provide more significant information than classifying them. The *TO* problem has been proven to be NP-complete [2], therefore various approximation and heuristic methods could be used to approach it.

Within the bioinformatics and computational biology framework, the *TO* problem refers to constructing a sorted collection of multi-dimensional biological data, collection that reflects an accurate temporal evolution of a certain biological process. A large part of the existing data is static, but biological processes are mostly dynamic. In order to be able to analyze and characterize these processes, scientists need dynamic information and one way to obtain this from static data is by inferring temporal orderings to this data. The *TO* problem is important within bioinformatics, as there are many practical applications for it, one of the most significant being in the field of cancer research, as cancer is inherently a dynamic disease.

2.2. Reinforcement Learning based approach for the *TO* problem.

Reinforcement Learning (RL) [7] is an approach to machine intelligence that combines two disciplines to solve successfully problems that neither discipline can address individually: *Dynamic programming* and *Supervised learning.* RL is a synonym of learning by interaction [9]. During learning, the adaptive system tries some actions (i.e., output values) on its environment, then it is reinforced by receiving a scalar evaluation (the reward) of its actions. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time. In RL, the computer is simply given a goal to achieve and it learns how to achieve that goal by trial-and-error interactions with its environment.

In [3] we introduced a reinforcement learning based technique for identifying a temporal ordering of a series of multi-dimensional biological samples. Even though in the above mentioned work we refer strictly to gene expression data obtained from microarray experiments, the applicability of our method is more general and it can be used with different types of multi-dimensional biological data.

From a computational point of view, the *TO* problem was defined as the problem of generating a permutation that maximizes the overall similarity of the sequence of samples considered in the ordering [3]. The RL task associated to the *TO* problem consists in training the agent to find a path from the initial to a final state having the maximum associated overall similarity. During the training step of the learning process the learning agent determines its *optimal policy* in the environment, i.e. the mapping from states to actions that maximizes the sum of the received rewards. The equivalent *action configuration* is viewed as a permutation that gives the temporal ordering for the input samples. For training the *TO* agent [3] a Q-learning approach was used [12] and a new action selection mechanism was defined in order to guide the exploration of the search space [3]. After the training step of the agent has been completed, the solution learned by the agent, which indicates the recovered temporal ordering, is constructed starting from the initial state and following the *Greedy* mechanism. For more details about how the data was pre-processed, about the definitions of the state and action spaces, reward and transition functions or about the action selection mechanism, we refer the reader to [3].

3. Variations of the *RL* based approach

This section aims to present several variations we propose for the *RL* based approach introduced in [3], used to solve the biological *TO* problem. In [3] we introduced an action selection mechanism based on the ϵ-Greedy mechanism [12], which uses a look-ahead procedure, in order to better guide the learning agent through the search space. To investigate how the policy specifying the way in which a new action is chosen in each given state influences the accuracy of the recovered ordering, we firstly try a different action selection policy: the *softmax policy.* A second idea refers to using a different RL approach, one that combines Q-learning with an essential mechanism of RL: *eligibility traces.*

3.1. **The Softmax Action Selection Policy.**

One key aspect of reinforcement learning is a trade-off between *exploitation* and *exploration* [13]. To accumulate a lot of reward the learning system must prefer the best experienced actions, however, it has to try (to experience) new actions in order to discover better action selection mechanisms for the future.

Several rules (policies) for choosing actions in order to make transitions among states during the learning process exist in the literature. The *greedy* policy implies that the learning agent chooses the highest-valued action in each state. An agent using this mechanism only exploits current knowledge to maximize its reward, but does not explore new states that could lead to higher long term rewards. A more effective method, which balances the exploration of new states with exploitation of current knowledge, is ϵ-Greedy [12]. It selects the greedy action with probability $1 - \epsilon$ and, in order to explore the environment, with probability ϵ it chooses an action at random, uniformly, not taking into consideration the action value estimates. Therefore, the main drawback of ϵ-Greedy is that the worst action is as likely to be chosen as the second best one.

A way to counter this disadvantage is to use a policy that chooses better actions more often. This is achieved by the *softmax* action selection policy [12], in which actions are ranked according to their value estimates and each action is chosen with a probability computed using its value. The greedy action will still have the highest probability. The most common softmax method uses a Gibbs, or Boltzmann distribution, where the probability of choosing action a in state s is (for a Q-learning approach):

$$\frac{e^{Q(s,a)/\tau}}{\sum_a e^{Q(s,a)/\tau}} \tag{1}$$

where τ is a positive parameter called *temperature*, which specifies how random actions should be chosen. For high values of the temperature all actions will be almost equiprobable. As the temperature is reduced, the actions that have higher value estimates are more likely to be selected and in the limit, as $\tau \to 0$, the best action is always chosen, this meaning that the softmax policy becomes the same as the greedy policy.

3.2. **Q-learning with eligibility traces.**

Eligibility traces were firstly introduced in [5] and they are a basic mechanism used in RL for handling delay [10]. The idea is that each time a state is visited it is marked by a trace, which then gradually decays over time, exponentially, according to a decay parameter λ ($0 \leq \lambda \leq 1$) and to the discount rate parameter γ. The trace makes the state *eligible* for learning [10].

There are two types of possible implementations for eligibility traces:

- *Accumulating eligibility traces* - the trace increases each time a state is visited. States that are visited more recently and more often are assigned more credit. For a Q-learning approach, the accumulating trace is defined in the following way [10]:

$$e_{t+1}(s,a) = \begin{cases} \gamma\lambda e_t(s,a) + 1, & if \;\; s = s_t \; and \; a = a_t \\ \gamma\lambda e_t(s,a), & otherwise \end{cases} \tag{2}$$

  for all state-action pairs (s,a). Here $e_t(s,a)$ represents the eligibility trace of the state-action pair (s,a) at time t, s_t is the actual state and a_t the actual selected action at time t.

- *Replacing eligibility traces* - each time a state is visited its trace is reset to 1, disregarding the previous trace information. For a Q-learning approach the replacing trace for a state-action pair is [10]:

$$e_{t+1}(s,a) = \begin{cases} 1, & if \;\; s = s_t \; and \; a = a_t \\ 0, & if \;\; s = s_t \; and \; a \neq a_t \\ \gamma\lambda e_t(s,a), & otherwise \end{cases} \tag{3}$$

  for all state-action pairs (s,a).

There are two approaches that combine Q-learning with eligibility traces: Watkins's $Q(\lambda)$ [14] and Peng's $Q(\lambda)$ [8]. As in this study we use only the former, we will briefly describe it in the following. In Q-learning the agent learns about the greedy policy, but usually, during training, it follows an exploratory policy (e.g. ϵ-greedy). Therefore, in learning about the greedy policy, the eligibility trace information can be used only as long as the greedy policy is followed. This means that eligibility traces are updated using Formula 2 or 3 (depending on the case) for the greedy actions, but the moment an exploratory action is taken the eligibility trace for the respective state-action pair is set to 0. The algorithm is given in Figure 1. We denote in the following by $Q(s,a)$ and $e(s,a)$ the Q-value estimate, respectively the eligibility trace value associated to the state s and action a, by α the learning rate, by γ the discount factor and by λ the decay parameter.

```
Repeat (for each episode)
  Select the initial state $s$ of the agent (as $s_1$).
  Choose action $a$ from $s$ using the given action selection mechanism.
  Repeat (for each step of the episode)
```

```
      Take action $a$, observe the reward $r(s,a)$ and the next state $s'$.
      Choose action $a'$ from $s'$ using the given action selection mechanism.
      $a^* \leftarrow argmax_b Q(s',b)$
      $\delta \leftarrow r(s,a) + \gamma \cdot Q(s',a^*) - Q(s,a)$
      Update $e(s,a)$ //$e(s,a) \leftarrow e(s,a)+1$ or $e(s,a) \leftarrow 1$
      For all s,a:
        $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \delta \cdot e(s,a)$
        If $a' = a^*$
          $e(s,a) \leftarrow \gamma \cdot \lambda \cdot e(s,a)$
        else
          $e(s,a) \leftarrow 0$
      s ← s'
    until $s$ is terminal
  Until the maximum number of episodes is reached or the $Q$-values do not change
```

Figure 1. Watkins's $Q(\lambda)$ algorithm [14].

Another possible implementation of $Q(\lambda)$, called *naive* $Q(\lambda)$ [12], would be the same as Watkins's algorithm, except that for an exploratory action the eligibility traces are not set to 0.

4. Experiments

In this section we provide experimental evaluations of the algorithms described in Section 3. Several tests were made, for each Q-learning algorithm (*traditional Q-learning*, $Q(\lambda)$ and *naive* $Q(\lambda)$), using each type of eligibility trace (*accumulating* and *replacing*) and two different action selection policies (*one step look-ahead procedure* [3] and *softmax* action selection policy).

For the experiments we used a software framework that we have previously introduced for solving combinatorial optimization problems using reinforcement learning techniques [4].

4.1. **Case study.**

The data set we used to test the performance of the different Q-learning based algorithms is a time series composed of gene expression data measuring the levels of expression of almost every yeast gene, at eight different time points, as yeast cells were affected by a given type of environmental change: dithiothrietol (DTT) exposure [6]. A time series is a collection of data resulted from a specific type of biological experiment: samples of tissues are extracted from the same individual at different and known moments in time, during the progression of the biological process. Thus, for a time series data set, the exact time of each sample is provided and the ordering is known. This data set was also used in [3] along with several different time series experiments for yeast or human cells and cancer gene expression data. As described in [3], in order to reduce the dimensionality of the input data (which is, usually, huge in the case of microarray experiments), we firstly pre-process the data by applying a statistical analysis, the final goal being the selection of those features (genes) that are most important for an accurate temporal ordering.

The algorithms are compared by examining the accuracy of the recovered orderings, by the number of epochs they need to achieve convergence and by the computational time. In [3] we introduced an evaluation measure, called *Samples Misplacement Degree (SMD)*, which, in our view, asseses the quality of a solution (ordering). We mention that smaller values for the SMD (smaller numbers of misplaced samples) indicate better orderings, the correct ordering having $SMD = 0$. Regarding the parameter setting, we remark that for all types of tests we used the following values: the discount factor for the future rewards is $\gamma = 0.95$; the learning rate is $\alpha = 0.8$; the number of training episodes is $7 \cdot 10^5$; for the tests using eligibility traces the decay parameter is $\lambda = 0.95$. For each of the two action selection policies, tests were made for different values of policy parameter (ϵ - in the case of the one step look-ahead procedure and τ - in the case of softmax): $\{0.1, 0.2, 0.5, 0.8, 0.9\}$. We mention that the experiments were carried out on a PC with an Intel Core i5-2400 Processor at 3.1 GHz (4 CPUs) with 8 GB of RAM.

4.2. **Comparative results.** In the following, we present the results obtained by each type Q-learning algorithm.

The *traditional Q-learning algorithm*, with no eligibility traces, proves a very good performance with both action selection policies. In the case of the *ϵ-Greedy based look-ahead procedure* [3], the optimal solution (the correct ordering $1, 2, 3, 4, 5, 6, 7, 8$) is obtained within very short amounts of time - less than 2 seconds, for all values of ϵ. During the first few epochs of the training process the algorithm obtains various orderings, depending on the value of the parameter ϵ, but it converges very soon to the optimal ordering, in less than 3000 training epochs, on average. This is illustrated in the first image of Figure 2, which depicts the overall similarity of the solutions obtained during the training process. We mention that the overall similarity of 97.051 corresponds to the correct ordering.

The *softmax action selection policy* also leads to the correct ordering, for all values of the temperature parameter, except for $\tau = 0.1$. In this case, the algorithm converges to a different ordering of the samples: $S = 8, 1, 2, 3, 4, 5, 6, 7$, having $SMD(S) = 3$. Still, we observe that the algorithm succesfully recovers

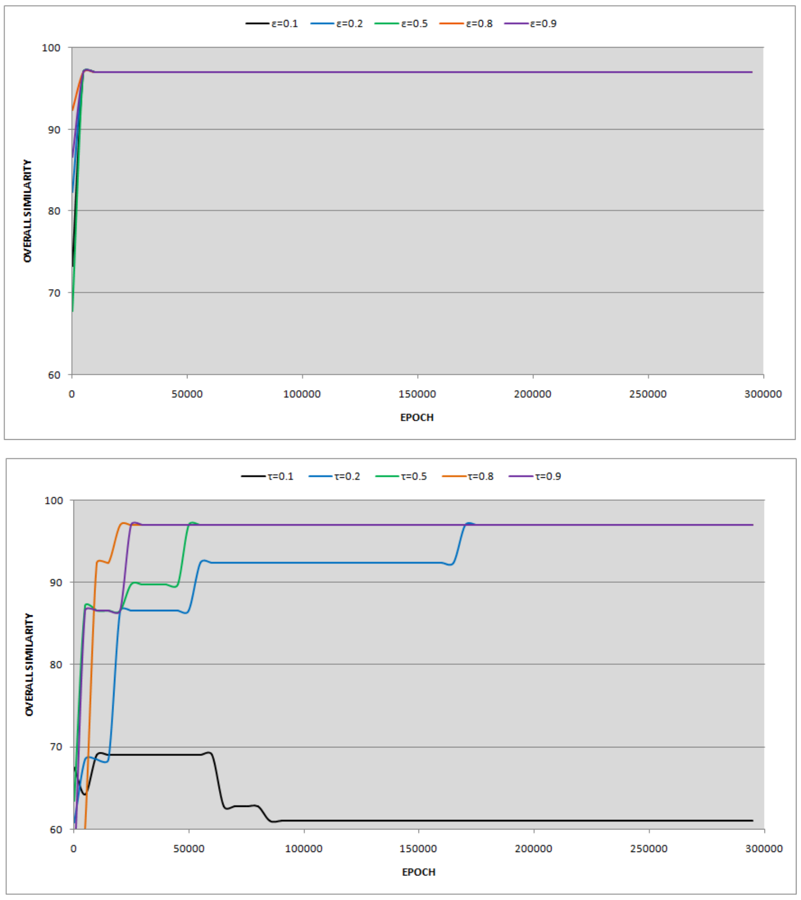

FIGURE 2. Q-Learning: the learning process.

the order of a subset of 7 samples, out of the set of 8. Another observation is that for the softmax policy the convergence is slower than with the other used policy, but as the temperature parameter increases the number of epochs needed to reach the solution decreases. This is illustrated in the second image of the Figure 2: for $\tau = 0.2$ the convergence is achieved after 170000 epochs, while for $\tau = 0.9$ only 20000 epochs are necessary. As for the computational time, we remark that even for smaller values of τ the solution is retrieved in less than 1 minute.

As soon as eligibility traces are introduced, the behaviour of the Q-learning algorithm changes radically: for certain values of ϵ or τ it does not converge at

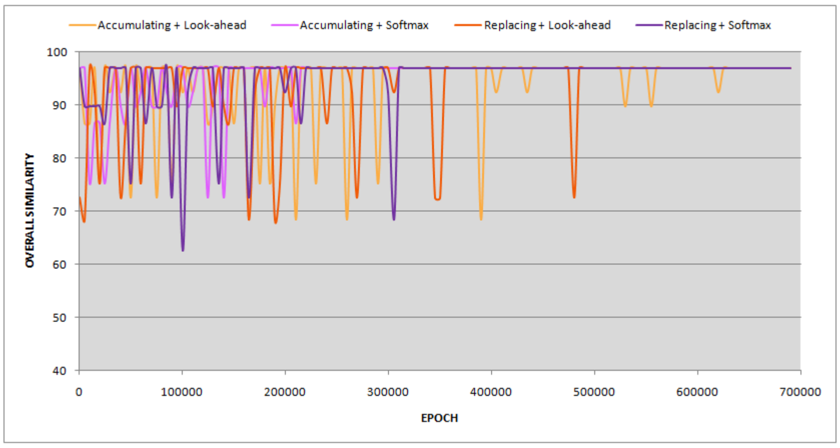

FIGURE 3. $Q(\lambda)$: the learning process for $\epsilon = 0.8$ and $\tau = 0.8$.

all, while for other values it retrieves the correct ordering, but within greater amounts of time.

Watkins's $Q(\lambda)$ [14] performs somewhat similarly for the two types of eligibility traces. In this case, the softmax policy achieves better convergence than the alternative. $Q(\lambda)$ in conjunction with *softmax* has the following behaviour: for $\tau = 0.1$ the algorithm converges to different orderings than the correct one, which have, however, values less than 4 of the SMD measure. As the temperature increases over 0.8 the algorithm slowly converges to the correct ordering, after 250000 training epochs. When used with the *look-ahead action selection procedure*, for small values of ϵ, it does not achieve convergence. But, for $\epsilon \geq 0.8$ it begins to converge to the correct solution after 250000 episodes, equivalently 140 seconds. Still, in some rare cases, there are individual epochs when it slightly deviates from the solution, but it soon returns to the maximum overall similarity ordering. These are illustrated in Figure 3, which shows the overall similarity of the solutions obtained during the training process, for both action selection strategies (using $\epsilon = 0.8$ and $\tau = 0.8$) and both types of traces. It can be observed that the algorithms using the softmax policy (represented in 2 different shades of purple) completely converge, while the ones using the look-ahead procedure (represented in light and dark orange) slighlty deviate from the solution once in a while.

In what concerns the *naive* $Q(\lambda)$, when run using *accumulating* eligibility traces and the *look-ahead policy* it does not converge for lower values of ϵ, but for $\epsilon \geq 0.8$ it soon converges to the correct solution, after 15000 episodes, in less than 3 seconds. In this case, the *softmax* selection mechanism leads to

	Accumulating traces		**Replacing traces**	
	Look-ahead	**Softmax**	**Look-ahead**	**Softmax**
$Q(\lambda)$	$\epsilon < 0.8 \Rightarrow$ div. $\epsilon \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\tau = 0.1 \Rightarrow S'$ $\tau \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\epsilon < 0.8 \Rightarrow$ div. $\tau \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\tau = 0.1 \Rightarrow S'$ $\epsilon \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)
naive $Q(\lambda)$	$\epsilon < 0.8 \Rightarrow$ div. $\epsilon \geq 0.8 \Rightarrow$ S ($>$ 250000 epochs)	all values of $\tau \Rightarrow$ div.	all values of $\epsilon \Rightarrow$ div.	$\tau = 0.1 \Rightarrow S'$ $\tau > 0.1 \Rightarrow$ div.

TABLE 1. Results obtained by the $Q(\lambda)$ and naive $Q(\lambda)$ algorithms.

divergence for all values of the temperature parameter. For *replacing* traces, the situation is different. The algorithm does not converge for any of the tested values of ϵ, in the case of the *intelligent look-ahead procedure* and exhibits the same behaviour when combined with the *softmax policy.* Exception is the case when $\tau = 0.1$, when naive $Q(\lambda)$ converges to a different solution than the correct one and the convergence is achieved within less than 40 seconds.

The obtained results for $Q(\lambda)$ and naive $Q(\lambda)$ are synthesized in Table 1. Here S denotes the convergence to the correct ordering: $S = 1, 2, 3, 4, 5, 6, 7, 8$; S' denotes the convergence to a different ordering than the correct one and the abbreviation "div." is used to indicate divergence.

4.3. **Discussion.** We have experimented with three Q-learning based algorithms and two types of action selection policies in order to obtain results for the temporal ordering problem. The original model is a Q-learning algorithm which uses an intelligent ϵ-Greedy based look-ahead action selection mechanism [3]. The other two algorithms combine Q-learning with eligibility traces [5]. As action selection policies, we used the look-ahead procedure we introduced in [3], as well as the softmax selection policy [12].

The obtained results demonstrate that the traditional Q-learning algorithm performs better, for the considered problem, than the two algorithms that use Q-learning in conjunction with eligibility traces: Watkins's $Q(\lambda)$ [14] and naive $Q(\lambda)$ [12]. Both these algorithms are able to retrieve the correct solution, for certain values of the considered action selection policy parameters, but convergence is much slower than in the case of Q-learning without eligibility traces. A possible explanation for this behaviour would be the fact that in our representation of the environment the full set of states is never completely known and therefore eligibility traces can only be updated for a known subset of states. This leads us to the conclusion that, for the TO problem, Q-learning with no eligibility traces is more appropriate.

With regard to the action selection policies, we remark that in the case of the Q-learning algorithm the intelligent action selection procedure [3] converges faster than the softmax selection mechanism, as this procedure efficiently guides the exploration of the search space. On the other hand, for the look-ahead mechanism the training process during an episode has a time complexity of $\theta(n^2)$, while for the softmax policy this complexity is $\theta(n)$, where n is the number of samples considered in the ordering process. We will further investigate how an intelligent action selection mechanism, based on softmax instead of ϵ-Greedy, could influence the outcome.

5. Conclusions and Further Work

In the present study we investigated several variations to a reinforcement learning based approach for the biological temporal ordering problem, tackled from a computational perspective. Three Q-learning based algorithms have been experimentally evaluated and compared.

The algorithms were tested on a time series gene expression data set, consisting of samples extracted from yeast cells affected by dithiothrietol exposure, at eight different time points [6]. The tests showed that the traditional Q-learning algorithm performs well with both considered action selection policies (intelligent look-ahead procedure [3] and softmax [12]), retrieving the correct ordering in less than 1 minute, for all tested values of the parameters, except for one case. The two algorithms that combine Q-learning with eligibility traces are also able to obtain the correct solution, but only for certain parameter settings and after a high number of training epochs.

We plan to extend the evaluation of the Q-learning based algorithms for the other data sets we used in [3], to further develop the analysis. We will also investigate possible improvements of these models by adding various local search mechanisms or combining the softmax policy with the intelligent action selection procedure introduced in [3], by testing different values for the decay parameter λ (in the case of $Q(\lambda)$ and naive $Q(\lambda)$), by decreasing the action selection parameters (ϵ and τ) during the training process or by extending the model to a distributed RL approach.

ACKNOWLEDGEMENT

I thank my thesis advisor, Professor Gabriela Czibula for the guidance, ideas and suggestions for this paper. This work was possible with the financial support of the Sectoral Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title Modern Doctoral Studies: Internationalization and Interdisciplinarity.

References

[1] Chapman D., Kaelbling L. P. *Input generalization in delayed reinforcement learning: an algorithm and performance comparisons*, Proc. of the 12th International Joint Conference on Artificial Intelligence **2**, Morgan Kaufmann Publishers Inc., 1991, pp. 726-731, Sydney, New South Wales, Australia.

[2] Cohen W. W., Schapire R. E., Singer Y. *Learning to order things*, J Artif Intell Res **10**, 1999, pp. 243-270.

[3] Czibula G., Bocicor M.I., Czibula I.G. *Temporal Ordering of Cancer Microarray Data through a Reinforcement Learning based Approach*, Evolutionary Bioinformatics, 2012. Submitted for review.

[4] Czibula I. G., Czibula G, Bocicor M. I., *A Software Framework for Solving Combinatorial Optimization Tasks*, Studia Universitatis "Babes-Bolyai", Informatica, Proc. of KEPT 2011, Special Issue **LVI**(3), Babes-Bolyai University, 2011, pp. 3-8.

[5] Klopf A.H. *Brain function and adaptive systems. A heterostatic theory. Technical Report AFCRL-72-0164*, Air Force Cambridge Research Laboratories, Bedford, MA, 1972.

[6] Gasch A. P., Spellman P. T., Kao C. M., Carmel-Harel O., Eisen M. B., Storz G., Botstein D., Brown, P. O. *Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes*, Molecular Biology of the Cell **11**(12), 2000, pp. 4241-4257.

[7] Lin L.J. *Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching*, Machine Learning **8**, 1992, pp. 293-321.

[8] Peng J. *Efficient Dynamic Programming-Based Learning for Control*, PhD thesis, Northeastern University, Boston, MA, 1993.

[9] Perez-Uribe A. *Introduction to Reinforcement learning*, http://lslwww.epfl.ch/∼anperez/RL/RL.html, 1998.

[10] Singh S. P., Sutton R. S. *Reinforcement Learning with Replacing Eligibility Traces*, Machine Learning **22**, 1996, pp. 123-158.

[11] Susnea I., Vasiliu G., Filipescu A., Radaschin A. *Virtual Pheromones for Real-Time Control of Autonomous Mobile Robots*, Studies in Informatics and Control **18**(3), 2009, pp. 233–240.

[12] Sutton R.S., Barto A. G. *Reinforcement Learning: An Introduction*, MIT Press 1998.

[13] Thrun S. *The Role of Exploration in Learning Control*, Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches, Van Nostrand Reinhold, 1992, Florence, Kentucky.

[14] Watkins, C. J. C. H. *Learning from Delayed Rewards*, PhD thesis, Cambridge University, Cambridge, England, 1989.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogălniceanu street, 400084 Cluj-Napoca, Romania

E-mail address: `iuliana@cs.ubbcluj.ro`