TO APPEAR IN THE PROCEEDINGS OF FROM ANIMALS TO ANIMATS, THIRD INTERNATIONAL CONFERENCE ON SIMULATION OF ADAPTIVE BEHAVIOR (SAB94), BRIGHTON, UK, AUGUST 8–12, 1994

A COMPARISON OF Q-LEARNING AND CLASSIFIER SYSTEMS

Marco Dorigo[+,*] and Hugues Bersini[*]

[*] IRIDIA - Université Libre de Bruxelles
Avenue Franklin Roosvelt 50, CP 194/6, 1050 Bruxelles, Belgium
bersini, mdorigo@ulb.ac.be

[+] Progetto di Intelligenza Artificiale e Robotica
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
dorigo@elet.polimi.it

Abstract

Reinforcement Learning is a class of problems in which an autonomous agent acting in a given environment improves its behavior by progressively maximizing a function calculated just on the basis of a succession of scalar responses received from the environment. Q-learning and classifier systems (CS) are two methods among the most used to solve reinforcement learning problems. Notwithstanding their popularity and their shared goal, they have been in the past often considered as two different models. In this paper we first show that the classifier system, when restricted to a sharp simplification called *discounted max very simple classifier system* (D_{MAX}-VSCS), boils down to tabular Q-learning. It follows that D_{MAX}-VSCS converges to the optimal policy as proved by Watkins & Dayan (1992), and that it can draw profit from the results of experimental and theoretical works dedicated to improve Q-learning and to facilitate its use in concrete applications. In the second part of the paper, we show that three of the restrictions we need to impose to the CS for deriving its equivalence with Q-learning, that is, no internal states, no don't care symbols, and no structural changes, turn out so essential as to be recently rediscovered and reprogrammed by Q-learning adepts. Eventually, we sketch further similarities among ongoing work within both research contexts. The main contribution of the paper is therefore to make explicit the strong similarities existing between Q-learning and classifier systems, and to show that experience gained with research within one domain can be useful to direct future research in the other one.

1. Introduction

Reinforcement Learning (RL) is a class of problems in which an autonomous agent acting in a given environment improves its behavior by progressively maximizing a function calculated just on the basis of a succession of scalar responses (rewards or punishments) received from the environment. No complementary guidance is provided for helping the exploration/exploitation of the problem space, and therefore the learning agent can rely only on a trial-and-error strategy. Q-learning (Watkins, 1989) and classifier systems (CSs) (Booker, Goldberg, Holland, 1989) have been separately proposed as two general frameworks for treating reinforcement learning problems. Despite their shared goal, only a few researchers (Sutton, 1988; Twardowski, 1993; Roberts, 1993; Wilson, 1994) have discussed the relationship between them, and they are largely regarded as different approaches.

We believe that the reason for this situation is to be found in their different origins. Although Samuel's (1959) work on learning the game of checkers appears to be a common inspiration, Q-learning and temporal differences (TD) methods originated from the behaviorist and cybernetic tradition of cognitive science, paying large attention to neural networks, control theory, and ethology; on the other hand, the CS found its origin back in the symbolic and AI roads of cognitive science, more focused on rule-based systems and symbolic types of representation, reasoning and learning. Indeed, there are some clear indications of these different backgrounds in, for instance, the use of neural networks (or any numerical type of classifier) to attack the issue of generalization for Q-learning and TD, while relying on don't care symbols in CSs. In the field of process control, we observe today a similar kind of convergence between two types of methods, likewise having the two behaviorist and symbolic distinct origins: neural networks and fuzzy systems (Bersini and Gorrini, 1993). A second reason for this situation is that while the Q-learning algorithm is the heart of the first reinforcement strategy to be compared, the bucket brigade (BB) algorithm (Holland, 1980), that is the counterpart of Q-learning in the CS framework, is only a component of the CS, making its restricted analysis harder to achieve.

In this paper our objective is to underline the strong similarities between not only the original CS and Q-learning, but also between ongoing research being developed in the two respective communities. This objective will be pursued in four successive steps.

First, a radical simplification is achieved on the CS in order to obtain a primitive version of it called Very Simple CS (VSCS). This is very easily derived when complying with four restrictions: (i) classifiers have one condition and one action; (ii) the message list has length one and contains only messages coming from the environment, that is there are no internal messages; (iii) the don't care symbol is not used in the classifier coding; (iv) the classifier set is complete, that is all state-action pairs (classifiers, in CS jargon) are covered, and therefore no genetic search is necessary.

Second, we will show the equivalence of a slightly modified version of the just derived VSCS, called Discounted Max VSCS (D_{MAX}-VSCS), with Watkins' original Q-learning algorithm. The first good new of this equivalence is immediate: all the theoretical studies developed around the convergence proofs of Q-learning can be likewise applied to the D_{MAX}-VSCS. Also all the experimental works aiming at making possible the use of Q-learning for real applications can be of interest for CS users.

Third, three of the restrictions we need to impose to Holland's original presentation of the CS for showing the equivalence with Q-learning might be regarded as an impoverishment and all three turn out so essential as to be recently rediscovered and reprogrammed by Q-learning adepts. They are on one hand the presence of hidden state and don't care symbols in the coding of the agent, points (ii) and (iii) above, and on the other, point (iv), the existence of a double level plasticity that we will call *parametric* and *structural* plasticity.

The internal states provide the system with a short term memory capacity which allows the agent to decide its actions not only on the basis of its current perceptions, but also as a consequence of its past perceptions and actions. This interesting CS feature has been integrated in Q-learning by people like Lin and Mitchell (1992), Whitehead and Lin (1994), Chrisman (1992) and McCallum (1993). The presence of don't care symbols allows the agent to generalize a certain action policy over a class of environmental states with an important gain in learning speed by data compression: a key requirement in any trial-and-error based learning strategy. This generalization capacity has been investigated by Q-learning users either by means of NN (Lin, 1992) or by statistical clustering techniques (Mahadevan and Connell, 1992).

The double level plasticity, *parametric* and *structural*, whose biological inspiration is largely discussed in (Farmer, 1991) and (Bersini, 1993; Bersini and Varela, 1994), refers to the strong adaptive capacity of a system which automatically adjust its numerical parameters with the simultaneous possibility to modify the structure of the system by the generation of new agents. For example, parametric plasticity accounts for changes in the classifiers strengths, in the actions Q-values, or in the NNs synaptic weights; on the other hand, structural plasticity accounts for new actions in Q-learning, new neurons in a NN, new classifiers in CSs, or other structural modification of the system like a finer division of the state space. The double level plasticity leads to an acceleration of the reinforcement learning discovery of good policies in large problem spaces. This acceleration is obtained by a simultaneous search of a satisfying minimal structural representation of the problem space and, within this minimal representation, the discovery at a computationally reduced cost of the optimal solution. In addition, it makes the system exhibit a larger degree of adaptivity thus escaping from the brittleness of classical AI methods.

Fourth, since the original proposals of both methods, a lot of developments have been carried out in the two communities with astonishing resemblance; examples are hierarchical task decomposition and the fuzzification of the methods. The main contribution of this paper is to make explicit the strong analogy between Q-learning and CSs so that experience gained in one domain can be useful to guide future research in the other.

The paper is organized as follows. In Section 2 we present VSCS and D_{MAX}-VSCS, and we compare D_{MAX}-VSCS to Q-learning. In the second part of the paper we continue our comparative analysis of CSs and Q-learning approaches to reinforcement learning. We compare extensions of the VSCS model, i.e. the original Holland's CS obtained again by replacing the four constraints, with currently investigated extensions to the basic Q-learning[1] model. In Section 3 we discuss hidden state and short-term memory. Section 4 is devoted to the problem of generalization. In Section 5 we discuss parametric and structural plasticity. Finally, in Section 6 we conclude with a brief overview of current research topics.

2. The very simplified classifier system and Q-learning

A very simple CS complies with four restrictions: (i) classifiers have one condition and one action, (ii) the message list (ML) length is constrained to one: $|ML|=|ML_e|=1$ where the subscript e denotes the fact that the ML slot is reserved for environmental messages, (iii) classifiers are symbols on $\{0,1\}^*$, that is, no don't cares (#s) are allowed, and finally (iv) the classifier set contains one copy of all the possible classifiers (state-action pairs), and therefore there is no need to use the genetic algorithm to modify the covering of the state space.

---

[1] In this paper we follow the following convention: (i) basic Q-learning is Watkin's Q-learning (1989) in tabular form; (ii) the classifier system, is Holland's CS (Booker, Goldberg, Holland, 1989).

Constraint (ii) says that only one message is possible at each time step, and that this message comes from the sensors (this is often called an environmental message and represents the state of the environment as perceived by the agent). Constraint (i) is in some way connected to (ii) because it would make no sense to have more than one condition when there is a single slot on the message list. Constraint (iii) removes the machinery which CSs use to generalize. Constraint (iv) allows a one-to-one correspondence between classifiers and Q-learning state-action pairs. The VSCS algorithm which results from these restrictions is given in Fig.1.

```
Initialization
   Create a classifier for each state-action pair;
   t:=0;
   Set St(cc,ac), the strength at time t of
   classifier c, to an initial value;
               {cc is the condition part of classifier
                c, while ac is its action part}.
Repeat for ever
   Read(m) {m is the sensor message};
   Let M be the matching set;
   Choose the firing classifier c ∈ M, with a
   probability given by St(cc,ac)/∑d∈M St(cd,ad);
   Change classifiers strength according to the
   implicit bucket brigade;
   t:=t+1;
   Execute(ac);
```

Figure 1. *The very simple classifier system. The implicit bucket brigade is explained in the text.*

In VSCS the equation which rules the change in strength of a classifier c is the following:

$$S_{t+1}(c_c,a_c) = (1-\alpha)\cdot S_t(c_c,a_c) + R + \alpha\cdot S_{t+1}(c_d,a_d) =$$

$$S_t(c_c,a_c) + \alpha\cdot\left(\frac{R}{\alpha} + S_{t+1}(c_d,a_d) - S_t(c_c,a_c)\right) \tag{1}$$

where $S_{t+1}(c_c,a_c)$ is the strength of classifier c at time t+1, conditions are called c and actions a, and subscripts c and d identify the classifiers to which conditions and actions belong (e.g., c_d is the condition part of classifier d). Equation (1) essentially says that, each time a classifier is activated its strength changes, and that this change amounts to the algebraic sum of outgoing payments (the -α times strength component), environmental rewards, and ingoing payments. The VSCS reinforcement algorithm is the same as the reinforcement algorithm used in Wilson (1985) except that to create the parallel with Q-learning only one classifier is reinforced on each time-step. Wilson's algorithm was termed the 'implicit bucket brigade' in Goldberg (1989). The sense of the word 'implicit' is that there is no direct connection between a classifier activated at time t and one activated at time t+1. In the standard bucket brigade a classifier activated at time t+1 is, most of the times[2], activated by messages sent by a precise set of classifiers at time t. Conversely, this is never the case in the implicit bucket brigade, as there are no internal messages. In the implicit bucket brigade a classifier activated at time t+1 implicitly owes its activation to the previously active classifiers, as they were the cause[3] for the new environmental message.

If we now consider 1-step Q-learning, the equation which rules the change of the Q-value for the state-action pair (x,a) is

$$Q_{t+1}(x,a) = (1-\alpha)\cdot Q_t(x,a) + \alpha\cdot\left(R + \gamma\cdot \underset{b}{\mathrm{MAX}}\, Q(y,b)\right) =$$

$$Q_t(x,a) + \alpha\cdot\left(R + \gamma\cdot \underset{b}{\mathrm{MAX}}\, Q(y,b) - Q_t(x,a)\right) \tag{2}$$

where y is the state obtained when executing action a in state x. In both equations R is the external reward.
We can continue our analogy noting that, in the constrained CS we have chosen to deal with, the condition part c_c of classifier c represents the state of the system, and its action part a_c represents the action. That is, in the analogy with Q-learning we have $c_c \Leftrightarrow x$, $a_c \Leftrightarrow a$, and $c_d \Leftrightarrow y$. If we substitute x and y in the corresponding symbols of equation (1), we obtain

$$S_{t+1}(x,a) = S_t(x,a) + \alpha\cdot\left(\frac{R}{\alpha} + S_{t+1}(y,b) - S_t(x,a)\right) \tag{3}$$

where $b = a_d$. This is the VSCS rule for changing strengths of state-action pairs.

Remembering that $Q(x,a)$ represents the value of action a in state x and that action a in state x causes the system to move to state y, while in the bucket brigade $S(x,y)$ represents the strength of the rule which, if activated, causes a state transition from state x to state y, we obtain D_{MAX}-VSCS changing formula (3) to meet formula (2). This is easily done if we observe that the two formulas, (2) and (3), differ only in the way the error part is computed, and in particular:

(i) Q-learning evaluates the next state y by choosing the value of the best action in the next state, while in VSCS it is evaluated by the strength of the state-action pair actually used,

(ii) in VSCS the evaluation of the next state is not discounted, that is, it is not multiplied by γ.

---

[2] Also in the standard bucket brigade it can happen that a classifier is activated by an environmental message, and therefore the standard bucket brigade contains the implicit bucket brigade as a special case.

[3] They are the only cause in static environments, one of the causes in dynamic environments.

Regarding point (i), in a companion paper (Bersini, Nowé, Caironi, Dorigo, 1994) we have proposed a generalization of formula (2), which allows the definition of a more general class of RL algorithms. An original aspect of the generalization we propose is the use of a *next state evaluation* (NSE) operator which generalizes the MAX operator. We experimented with different values for NSE; among them we tried the ACT operator, which corresponds to using the Q-value of the move actually chosen, as it is done by the BB of formula (3). Experimental results have shown that this choice gives rise to a worse performance than with the MAX operator, but that it does not seem to affect the convergence properties of the algorithm.

With respect to the discounted nature of the algorithm (i.e., $\gamma < 1$), this is usually justified (Schwartz, 1993) by Q-learning adepts with the need to guarantee the boundness of the final expected value of an action. This can be guaranteed either by the presence of a reachable absorbing state in the problem state space or, in the absence of such a state, by $\gamma < 1$ (see Watkins and Dayan, 1992; Tsitsiklis, 1993; Schwartz, 1993; for a more formal analysis of this issue). However, apart from these theoretical considerations, it can be very useful in practice, even in presence of a reachable absorbing state, to choose either $\gamma<1$ or $\gamma=1$, depending on the problem nature. When the problem relates to a path minimization (like the maze problem), $\gamma < 1$ is highly desirable, since in that case the final expected cost will be as big as the number of steps to reach the goal. On the contrary, in problems in which a possible solution is a cyclic behavior, like the cart-pole, $\gamma=1$ is an interesting option, since all the Q-values in that case converge towards a same value, and therefore the probability of falling in a cycle is considerably increased (see Twardowski (1993) for an experimental confirmation).

Our comparison of the Q-learning algorithm with the CS approach has made clear that the CS is a much more complex model than Q-learning. It is interesting that most of the researchers which have been studying Q-learning were unsatisfied with it in ways that led them to generalizations which were already present in the original CS model. In the following of this paper we will try to discuss some of these similarities.

3. Hidden state and short-term memory

In this section we consider the implications of removing the first two constraints on our VSCS; that is, classifiers have more than one condition[4], but still one action, and the message list has extra slots beside the one for the external message. This new kind of CS, which we call VSCS-M (VSCS with memory), although still less general than Holland's original CS, is more powerful than VSCS in that it allows the system, at least in principle, to have short-term memory and to solve therefore hidden-state problems. In the Q-learning community a similar direction was taken by Lin and Mitchell (1992) in their work on Q-learning with recurrent networks. To compare the two approaches we refer to Fig.2. In Fig.2a are shown those components of a VSCS-M which are interesting for our comparison. In particular, it should be noted that the message list comprises the following.

(i) One slot for the external message; we call this ML_e, and we have $|ML_e|=1$.
(ii) One slot for the action message. This is the message which causes the action; appending it on the message list gives the system a one-step memory. We call this ML_a, and we have $|ML_a|=1$.
(iii) The rest of the list is for internal messages; we call this ML_i, and we have $|ML_i|=n-2$.

As we said, the CS of Fig.2a has two new characteristics: classifiers have two conditions, and all three kinds of messages are matched against classifiers in the classifier set. (Matching classifiers produce two kinds of messages: (i) internal messages, which are appended to ML_i, and action messages which, after entering a probabilistic conflict resolution module, are reduced to a single winning message. This winning message, beside causing an action to be performed, is also appended to ML_a.)

Lin, in his neural net implementation of Q-learning (Lin, 1992), uses a feedforward neural net for each action. Each net receives as input a message, which is the equivalent of the environmental message in CSs, and proposes an action with a given strength. Thereafter, an action is chosen with a probabilistic choice which resembles the one going on in the conflict resolution module of CSs.

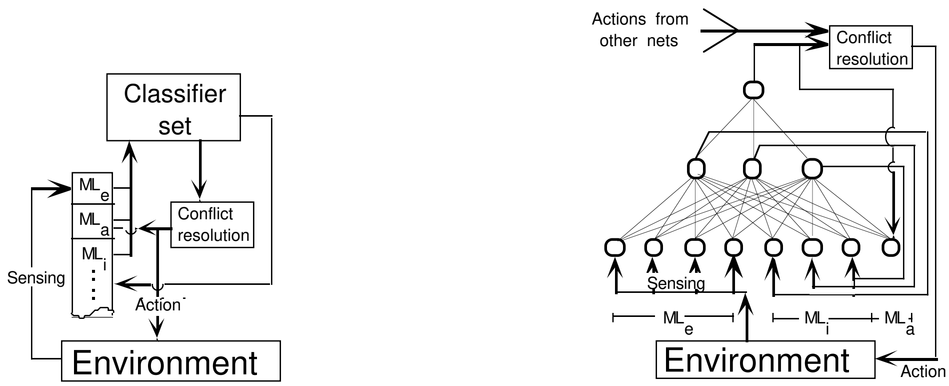
Consider now the recurrent architecture of Fig.2b; a similar architecture was proposed by Lin and Mitchell (1992). Feedback connections link the hidden units and the output unit to the input units. It is clear that, at least from a functional point of view, hidden units and feedback connections play the same role of internal messages and of the message appended to ML_a respectively.

4. Generalization

To be a viable tool for real world applications, reinforcement learning techniques need effective methods to partition the state-action space. This point has been the subject of research by both the CSs and the Q-learning communities. In this section we briefly discuss the role of don't cares (see constraint (iii) in Section 1), and the correspondent machinery found in Q-learning research.

---

[4] It can be shown that while moving from one to two conditions increases the representational power of a rule based system, all formalisms with two or more conditions are, from a representational power point of view, equivalent.



a) b)

Figure 2. *A comparison of VSCS-M with recurrent-net Q-learning.*
a) VSCS-M, the CS with constraints only on the use of # symbols and on the complete covering of the state-action space.
b) Q-learning augmented with recurrent neural nets.

In CSs generalization has been addressed by the use of the don't care (#) operator. A classifier attribute occupied by a don't care is considered to be irrelevant, and it is therefore by means of don't cares that generalization over classes of equivalent state-action pairs is achieved.

However, it is still an open problem how to develop set of rules which can effectively exploit this generalization possibility (so called default hierarchies, in which general rules cover default situations, and specialized rules are activated by exceptions, see Riolo, 1987; 1989). Moreover, CSs are inherently discrete systems (although fuzzy-CSs have been proposed to deal with continuous valued inputs and outputs, see (Valenzuela-Rendón, 1991; Bonarini, 1993)), which makes it difficult to achieve the kind of generalization in which a similar input should cause a similar output. This is more easily achieved in Q-learning systems enriched by neural nets, like Lin's system (1992).

Observing reinforcement learning algorithms, we are in presence of the two traditional and opposite ways to obtain partitions of the input space: bottom-up and top-down (Mitchell, 1982). In the bottom-up approach, the input space is initially very fine grained, and the partitions are created by clustering together input regions showing similar properties. In the top-down approach, the initial input space is uniformly coded and is progressively subdivided in finer parts. The bottom-up type of generalization has been investigated by Q-learning users following two approaches. The first one is the already cited approach of Lin (1992), who uses a neural net for each action. In the second, proposed by Mahadevan and Connell (1992), state-action points are clustered using statistical clustering methods. Since top-down methods amount to a progressive structural modification of the problem space, they will be discussed in the next chapter.

5. The double plasticity

In this section we discuss the last constraint we introduced to get VSCS, that is constraint (iv) of Section 1, regarding the coverage of state-action space. Q-learning and CS are both concerned with populations of agents which collectively achieve a certain objective: they are adaptive and distributed types of control. Holland's biological inspiration, that is ecosystems and genetic mechanisms, led him to address structural plasticity by the GA. However, an ecosystem does not perform any kind of distributed control; instead, each individual selfishly tries to survive in an environment constituted by the other individuals. A perhaps more adequate biological inspiration could have been the immune system in which new cells are produced at a very high speed (Bersini, 1993; Bersini and Varela, 1994) to improve the capacity of the system to achieve, as a whole and in an adaptive way, its vital functions. We hypothesize that, when interested in the collective performance of a system, its structural adjustments should aim at compensating for the current weakest parts of the system. This is in contrast with Holland's GA approach in which the generation of new actors is biased towards the best ones.

In CS, the GA is responsible for two types of structural changes, depending on whether it applies to the condition or to the consequent part of the classifiers: a change in the coding of the state space and a change in the set of actions. These two types of changes have also been investigated in the Q-learning community, but within the "collective and compensatory" perspective just presented. Aware of the exponential dependency of the search duration with respect to the size of the state-action space, different methods have been proposed in order to progressively divide the space so to obtain a final solution in the smallest possible space. Chapman and Kaelbling's G-algorithm (1991) recursively splits the state space using statistical measures of differences in reinforcements received. Munos, Patinel and Goyeau

(1994) make a similar recursive splitting of the state space following a different criterium. They split an action-state pair if the action acting in that particular state presents variations of great amplitude, indicating that the reinforcement received by that action is highly fluctuating and then requires a subdivision of the state (a similar approach was taken by Dorigo, 1993, in his CS called ALECSYS). This is typical of a mechanism which, contrary to GA, focuses on the current unsatisfactory part of the system to guide its improvement by a structural addition.

Regarding the action part, Bersini (1993) has implemented an adaptive control strategy in which the adaptation relies on the parametric and structural types of change occurring at different time scales (the inspiration is the immune system two-level adaptive strategy). First, the Q-learning tries to find for each state the best actions among a preliminary set of actions (the same in each state) during a certain number of Q-learning iterations. Then, after a certain number of steps, new actions are recruited in the states in which the current actions show bad Q-values as compared to other actions acting in the same state or actions acting in neighboring states. The new actions are selected so as to be the opposite of bad ones they replace (for instance *move left* instead of *move right* or a negative value instead of a positive one) and they are given the Q-value of the best action already present in the state (so that new actions are immediately tested). Q-learning is then activated again on the basis of the new set of actions.

6. Conclusions

We have seen that D_{MAX}-VSCS and Q-learning share

- the class of problems to which they are applied,
- the apportionment of credit mechanism.

We have also seen that both models are subject to research directed to enrich them with more powerful capabilities to make them useful for real world applications. These researches bring the D_{MAX}-VSCS back to its original form, namely Holland's CS; on the other hand, Q-learning is being enriched by functionally similar mechanisms. We have seen that the CS model is an inherently discrete model (but see Valenzuela-Rendón (1991), and Bonarini (1993) for fuzzy implementations of the CS), while Q-learning can smoothly go from the discrete implementation (e.g., Watkin's tabular Q-learning or other more efficient kinds of discrete partition of the state space like Chapman and Kaelbling (1991) or Munos et al. (1994)), to continuous neural net implementations. Regarding generalization capabilities the two models have followed rather different directions, with CSs using don't cares symbols, and Q-learning a whole set of different techniques. Both have tackled the short-term memory problem with functionally similar mechanisms: internal messages in CSs, recurrent nets in Q-learning.

Finally, both have proposed the idea that to build working systems it could be useful to use a *divide and conquer* approach, in which the global task is decomposed into many simpler tasks which are then used as modules of a designed architecture. This can be found in the work of Dorigo and Schnepf (1993), Dorigo (1992), Dorigo and Colombetti (1994), Colombetti and Dorigo (1992), for CSs, and in the work of Lin (1993), Dayan and Hinton (1992), and Mahadevan and Connell (1992), for Q-learning. In all these works the proposed architecture is a hierarchic architecture in which simple tasks are learned by reinforcement learning and coordination is designed (in Dorigo's and Lin's work also coordination is learned).

Finally, in Table 1 we summarize the characteristics of the two models as discussed in the paper.

Table 1. *Characteristics of the two models as discussed in the paper*

	Classifier system	Q-learning
Discrete representation	Holland's style CS	Tabular Q-learning
Continuous representation	Fuzzy CS	Q-learning with neural nets
Short-term memory	Internal messages	Recurrent neural nets
Generalization	Don't cares	Neural nets Statistical clustering
Double plasticity	Genetic algorithm plus bucket brigade	Incremental net division New action recruitment G-algorithm Partitioned Q-learning
Hierarchy	Explicitly designed with learning of both basic tasks and coordination (Dorigo and Colombetti, 1994)	Explicitly designed with learning of both basic tasks and coordination (Lin, 1993)

References

Bersini H., 1993. Immune network and adaptive control. *Toward a Practice of Autonomous Systems - Proceedings of the First ECAL*, Varela and Bourgine (Eds.), 217–225, MIT Press.

Bersini H. and Gorrini V., 1993. FUNNY (FUzzY or Neural Net) methods for adaptive process control. *Proceedings of EUFIT '93*, ELITE Foundation, Aachen, Germany, 55-61.

Bersini H. and Varela F., 1994. The immune learning mechanisms: Recruitment reinforcement and their applications. *To appear in Computing with Biological Metaphors*, R. Patton (Ed.), Chapman and Hall

Bersini H., Nowé A., Caironi P.V.C., Dorigo M., 1994. A family of reinforcement learning algorithms. *Tech. Rep. IRIDIA/94-1*, Université Libre de Bruxelles, Belgium.

Bonarini A., 1993. ELF: Learning incomplete fuzzy rule sets for an autonomous robot. *Proceedings of EUFIT '93*, ELITE Foundation, Aachen, Germany, 69–75.

Booker L., Goldberg D.E., and Holland J.H., 1989. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40, 1-3, 235–282.

Chapman D. and Kaelbling L.P., 1991. Input generalization in delayed reinforcement learning: An algorithm and performance comparison. *Proceeding of the Twelth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 726–731.

Chrisman L., 1992. Reinforcement learning with perceptual aliasing: The perceptual distinction approach. *Proceeding of the Tenth National Conference on Artificial Intelligence* (AAAI-92), 183–188.

Colombetti M. and Dorigo M., 1992. Learning to control an autonomous robot by distributed genetic algorithms. *Proceedings of From Animals To Animats, Second International Conference on Simulation of Adaptive Behavior (SAB92)*, Honolulu, 305–312, Bradford Books, MIT Press.

Dayan P. and Hinton G.E., 1992. Feudal reinforcement learning. In C.L. Giles, S.J. Hanson and J.D. Cowan (Eds), *Advances in Neural Information Processing Systems 5*, 271-278. San Mateo, CA, Morgan Kaufmann.

Dorigo M., 1992. ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems. *Technical Report No.92-011*, Politecnico di Milano, Italy.

Dorigo M., 1993. Genetic and non-genetic operators in ALECSYS. *Evolutionary Computation*, 1, 2, 151–164, MIT Press.

Dorigo M. and Colombetti M., 1994. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence,* to appear.

Dorigo M. and Schnepf U., 1993. Genetics-based machine learning and behavior-based robotics: A New Synthesis. *IEEE Transactions on Systems, Man, and Cybernetics*, 23, 1, 141–154.

Farmer D. 1991. A Rosetta Stone to connectionism. *In Emergent Computation*, Forrest S. (Ed.) MIT Press.

Goldberg D.E. 1989. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley.

Holland J.H., 1980. Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal of Policy Analysis and Information Systems*, 4, 2, 217-240.

Lin L-J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8, 3-4, 293–322.

Lin L-J., 1993. Hierarchical learning of robot skills by reinforcement. *Proceedings of 1993 IEEE International Conference on Neural Networks*, IEEE, 181–186.

Lin L-J. and Mitchell T.M., 1992. Memory approaches to reinforcement learning in non-Markovian domains. *Tech.Rep. CMU-CS-92-138*, Carnegie Mellon University, Pittsburgh, PA, May 1992.

Mc Callum R.A., 1993. Overcoming incomplete perception with utile distinction memory. *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, 190–196.

Mahadevan S. and Connell J., 1992. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55, 2, 311–365.

Mitchell T.M., 1982. Generalization as search. *Artificial Intelligence*, 18, 2, 203-226.

Munos R., Patinel J. and Goyeau P., 1994. Partitioned Q-learning. *Proceedings of From Animals To Animats, Third International Conference on Simulation of Adaptive Behavior (SAB94)*, Brighton, United Kingdom, August 8–12, 1994.

Riolo R.L., 1987. Bucket Brigade performance: II. Default hierarchies. *Proceedings of the Second International Conference on Genetic Algorithms*, J.J. Grefenstette (Ed.), Lawrence Erlbaum, 184–195.

Riolo R.L., 1989. The emergence of default hierarchies in learning classifier systems. *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (Ed.), Morgan Kaufmann, 322–327.

Roberts G., 1993. Dynamic planning for classifier systems. *Proceedings of Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 231–237.

Samuel A.L., 1959. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 3, 210–229. Reprinted in E.A. Feigenbaum and J.Feldman (Eds.) *Computers and thought.* New York: McGraw-Hill.

Schwartz A., 1993. A reinforcement learning method for maximizing undiscounted rewards. *Proceedings of Tenth*

International Conference on Machine Learning, Morgan Kaufmann, San Mateo, CA, 298–305.

Sutton R.S., 1988. Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 1, 9–44.

Tsitsiklis J.N, 1993. Asynchronous stochastic approximation and Q-learning. *Internal Report from Laboratory for Information and Decision Systems and the Operations Research Center, MIT.*

Twardowski K., 1993. Credit assignment for pole balancing with learning classifier systems. *Proceedings of Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 238–245.

Valenzuela-Rendón M., 1991. The fuzzy classifier system: A classifier system for continuously varying variables. *Proceeding of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 346–353.

Watkins C. J. C. H., 1989. Learning with delayed rewards. *Ph. D. dissertation*, Psychology Department, University of Cambridge, England.

Watkins C. J. C. H. and Dayan P., 1992. Technical Note: Q-learning. *Machine Learning*, 8, 3-4, 279–292.

Whitehead S.D. and Lin L.J., 1994. Reinforcement learning in non-Markov environments. *Artificial Intelligence*, to appear.

Wilson S.W., 1985. Knowledge growth in an artificial animal. *Proceedings of the First International Conference on Genetic Algorithms and their Applications,* J.J. Grefenstette (Ed.), Lawrence Erlbaum, 16–23.

Wilson S.W., 1994. ZCS: a zeroth level classifier system. *Evolutionary Computation*, MIT Press, to appear.