On Discontinuous Q-Functions in Reinforcement Learning

Alexander Linden

AI Research Division
German National Research Center for
Computer Science (GMD)
P. O. Box 1316
W-5205 Sankt Augustin
Germany
email: Alexander.Linden@gmd.de

Abstract. This paper considers the application of reinforcement learning to path finding tasks in continuous state space in the presence of obstacles. We show that cumulative evaluation functions (as Q-Functions [28] and V-Functions [4]) may be discontinuous if forbidden regions (as implied by obstacles) exist in state space. As the infinite number of states requires the use of function approximators such as backpropagation nets [16, 12, 24], we argue that these discontinuities imply severe difficulties in learning cumulative evaluation functions. The discontinuities we detected might also explain why recent applications of reinforcement learning systems to complex tasks [12] failed to show desired performance. In our conclusion, we outline some ideas to circumvent the problem.

Keywords: Machine learning, reinforcement learning, evaluation functions, temporal difference learning, backpropagation nets, robotics, path finding, continuous state space, obstacles.

1 Introduction

In robotics and artificial intelligence, path finding is generally included under the heading "path planning". In path planning, one looks for ways of, among other things, generating collision-free and efficient paths for robot manipulators and autonomous vehicles. Its scope also includes issues of traditional control problems and problem solving tasks. Standard methods like heuristic search suffer from their combinatorial complexity and inflexibility in changing environments. To get rid of these deficiencies, heuristic evaluation functions can be used. The method consists in directly selecting actions on the basis of the evaluation, which estimates the success of an action w.r.t. accomplishing a given goal. If this estimation is computed without search in state space and only on the basis of current state information, then "path planning" appears to be an improper expression; instead the term "path finding" has been used in the literature [14].

In the field of reinforcement learning one is concerned with systems that incrementally learn such heuristic evaluation functions through interaction with the environment. Indeed it can be shown that, under certain conditions, the learning algorithms converge to the optimal evaluation function [21, 27, 28, 7]. With an appropriate action generation mechanism, they form the core to path finding. Furthermore, such evaluation functions may be dynamically adapted to changes of the environment [23].

Currently most work on reinforcement learning has been done in the domain of finite and discrete state and action spaces [5, 23, 26, 27, 33, 32]. Tesauro [25] had great success in applying this learning method to the game of backgammon. Much less research has been done on applications requiring continuous state space representations [1, 2, 12]. Here only quite limited tasks have been examined. To the best of our knowledge, reinforcement learning has never been applied to the problem of optimal path finding with changing goals, obstacles and with continuous state and action space. In this context we have observed discontinuous cumulative evaluation functions, that have not yet been recognized elsewhere.

The paper is structured as follows: In the next section, we review learning with delayed reinforcement. Section 3 provides a general formulation of the path finding task. We argue that the optimal cumulative evaluation function should reflect the costs of the optimal path to the goal. Section 4 gives some path finding examples, illustrating the existence of discontinuous cumulative evaluation functions. In section 5, we discuss why the learning of evaluation functions in domains with continuous state spaces is quite problematic. We conclude with considerations for future research in order to circumvent these problems. The result of this paper may explain why recent applications of reinforcement learning to quite complex tasks [12] failed to show the desired performance.

2 Short Review of Reinforcement Learning

The general computational approach to learning from reward and/or punishment is known as reinforcement learning [15]. Very recently strong connections between dynamic programming in operations research [6], heuristic evaluation functions in artificial intelligence [10], adaptive control [11], and reinforcement learning have been discovered [3, 27, 31]. This has caused a rapidly growing interest in this field[1]. Although a multitude of algorithms and architectures is available that implement reinforcement learning [21, 29, 34], we only consider reinforcement learning systems that deal with delayed reinforcement [5, 21, 28]. In addition, all these systems share a central mechanism that allows a computational agent to evaluate situations with respect to a prediction of the accumulated future reinforcement. Minsky [15] called this evaluation function "secondary reinforcement", we will call it *cumulative evaluation function*. Because our focus is the study of this function under certain conditions it is appropriate

[1] See also the May, 1992 special issue on Reinforcement Learning in the Machine Learning Journal

to review reinforcement learning by one exemplary architecture. For this purpose we have chosen the efficient and widely used Q-learning [27, 28, 12] architecture.

Let us consider a computational agent. At each time step t, it observes the state of its environment $s_t \in S$. Its current policy Π maps s_t onto an action $a_t = \Pi(s_t)$. This action is executed in the environment, which results in a transition $s_{t+1} = \delta(s_t, a_t)$. In addition, the agent receives some spontaneous evaluation[2] $e_t = e(s_t, a_t)$. Some actions might yield a much smaller spontaneous evaluation than others but in the long term be much more favourable. This is the phenomenon of delayed reinforcement. The mechanism of telling how good an action or state actually was in the long term is called *temporal credit assignment* [15].

In order to develop method for learning a policy that is optimal w.r.t. delayed reinforcement, we derive a cumulative evaluation function[3], which tells how good a given action a_t will be in some state s_t. If Π is fixed, the agent loops through a sequence of states $s_{t+1}, s_{t+2}, \ldots$ and receives a corresponding sequence of spontaneous evaluations $e_t, e_{t+1}, e_{t+2}, \ldots$. The evaluation function relative to Π has the form of the discounted sum of future spontaneous evaluations

$$\begin{aligned} Q^{\Pi}(s_t, a_t) &:= \sum_{k=0}^{\infty} \gamma^k e_{t+k} \qquad (1) \\ &= e_t + \sum_{k=1}^{\infty} \gamma^k \cdot e_{t+k} \end{aligned}$$

Note that the discount factor $0 \leq \gamma < 1$ assures the convergence of the infinite sum.

In this sense the optimal policy $\Pi^\star$ yields the maximal evaluation for all states and action pairs:

$$\begin{aligned} Q^\star(s_t, a_t) &= e_t + \max \sum_{k=1}^{\infty} \gamma^k \cdot e_{t+k} \qquad (2) \\ &= e_t + \gamma \cdot \max \sum_{k=0}^{\infty} \gamma^k \cdot e_{t+1+k} \qquad (3) \end{aligned}$$

This immediately leads to Bellman's optimality equation in dynamic programming [3]

$$Q^\star(s_t, a_t) = e_t + \gamma \cdot \max_{a_{t+1}} Q^\star(\delta(s_t, a_t), a_{t+1}) \qquad (4)$$

Watkins [27] has designed a learning rule for autonomous systems based on this equation[4]. Watkins proved that this learning rule is convergent under specific assumptions:

[2] Often it is also called reinforcement, reward, immediate payoff, outcome

[3] Also called quality (Q) function by Watkins [27]

[4] Also Werbos [30, 31] has outlined learning algorithms in the framework of dynamic programming.

- The state space S is finite and discrete.
- In each state $s \in S$, only a finite number of actions is allowed.
- The environment is ergodic, this means that each state can be reached from every other state.
- The distribution of spontaneous evaluations $e(s, a)$ does not change over time.
- Each action in each state will be tried infinitely often. This property can be encoded in the action selection mechanism by the boltzmann distribution.

The algorithm uses a lookup table to store the estimated cumulated evaluations $\hat{Q}$. It works as follows:

1. Initialization:
   a) $\forall_{s,a} \hat{Q}(s, a) := 0$.
   b) The agent starts randomly in some state s_t.
2. Action selection by random exploration:
   a) The agent uses its Q-prediction to score each possible action a^i in $s(t)$ with m_i.
   b) The boltzmann distribution chooses actions randomly with the following probabilities.
   $$\text{Prob}(a^i) = e^{m_i/T} / \sum_k e^{m_k/T}$$
   The probability that action a^i is selected increases with its score m_i, but as $T \to \infty$ these differences vanish, and as $T \to 0$ the action with the highest score is chosen with probability almost 1.
3. The execution of action a_t results in a new state s_{t+1} and a new spontaneous evaluation e_t.
4. The prediction $\hat{Q}$ is updated with the following learning rule
   $$\Delta\hat{Q}(s_t, a_t) := \eta \cdot \left[e_t + \gamma \cdot \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t) \right] \tag{5}$$
   If the difference on the right hand is zero for all s, a then the equation 4 also holds true. The learning rule is a kind of temporal difference learning rule investigated first by Samuel [17, 18] and reformulated and studied intensively by Sutton [21, 22].
5. Increase t, decrease T and goto 2.

3 The Path Finding Domain

In this section, we transfer the ideas of reinforcement learning to the path finding domain with continuous state spaces. This is not to show any implementation and experiments but to have a framework for illustrating a remarkable property of cumulative evaluation functions in this domain.

In general, a path finding task can be formulated as follows:

There are

- a set S of states,
- a nonempty set $G \subset S$ of goal states,
- a set A of actions,
- a transition function $\delta : S \times A \to S$ that maps a state and an action to a successor state,
- a spontaneous evaluation function $e : S \times A \to I\!R$ that assigns real-valued costs to the execution of some action in some state. The costs are a kind of (negative) reinforcement or punishment. In this case one wishes to choose those actions with lead to minimal cumulative evaluation. This is implemented by taking the minimum over equations 2, 4, and 5 instead of the maximum.

δ and e may be partial so that setups can be covered where not all actions are executable in every state, for example if obstacles are in the way.

The task consists in finding a sequence of actions that leads the agent from some given start state to a goal state. In addition, this sequence should be kept as cheap as possible by minimizing the cumulative evaluation function.

4 Path finding in continuous state spaces

The framework above may apply analogously to state spaces which are subsets of continuous spaces, however the convergence theorem of Watkins (see section 2) does not work here.

We can show by illustration that nearly all tasks with forbidden regions (in our case implied by the obstacles) in state space have a discontinuous evaluation function. We illustrate this first by the simplest scenario we could think of.

Let the agent be a robot arm with one revolute joint (fig. 1). Its workspace contains a goal and an obstacle. The states are the positions of the arm and are represented by pairs of angles $(\varphi_{\text{ge}}, \varphi_{\text{oe}}), \varphi_{\text{oe}} \neq 0$, thus containing information about the location of the goal and the obstacle relative to the end effector position. The pairs with $\varphi_{\text{oe}} = 0$ form the forbidden region here. Goal states are all states with $|\varphi_{\text{ge}}| \leq \tau, \tau > 0$. Actions are limited joint transitions $a \in [-a_{\max}, -\varepsilon] \cup [\varepsilon, a_{\max}]$ with $a_{\max} < \pi$ and $0 < \varepsilon \leq \tau$, and $\delta((\varphi_{\text{ge}}, \varphi_{\text{oe}}), a) = (\varphi_{\text{ge}} + a, \varphi_{\text{oe}} + a)$. The actions are required to be at least of magnitude ε, so the arm is forced to move at each step so long the goal is not reached. $\varepsilon \leq \tau$ guarantees that the goal state interval cannot be jumped with a smallest step. The costs of an action in a state depend only on the action: $e(s, a) = |a|$. To prevent collisions with the obstacle, an action a is disallowed iff one of the following conditions holds:

- $\varphi_{\text{oe}} > 0$, $a < 0$, and $\varphi_{\text{oe}} + a \leq 0$
- $\varphi_{\text{oe}} < 0$, $a > 0$, and $\varphi_{\text{oe}} + a \geq 0$

This means that δ and e are only partial.

To be able to find a (nearly) optimal path to a goal state from every initial state that it is put into, the agent should learn a policy. This can be achieved

Fig. 1. LEFT: There is no obstacle between end effector and goal, therefore the best way is the direct way. RIGHT: As soon as the obstacle is in the way between end effector and goal the best and only way is the circumvention.

with Q-Learning using a slight modification of the algorithm in section 2: When a goal state is reached in step 3, the arm is set to a new non-goal state after step 4. This has the additional effect that γ becomes superfluous since the minimal costs are achieved by getting to a goal state with a finite number of actions, without changing the direction of movement. With the optimal policy, the Q-function is as follows (from now on, its second argument is always assumed to represent an allowed action):

$$Q^\star((\varphi_{\mathrm{ge}}, \varphi_{\mathrm{oe}}), a) = \begin{cases} 0, & \text{if } |\varphi_{\mathrm{ge}}| \leq \tau \\ |a| + \max_{b \in A} Q^\star((\varphi_{\mathrm{ge}} + a, \varphi_{\mathrm{oe}} + a), b), & \text{otherwise} \end{cases} \tag{6}$$

The actual values then are as follows:

$$Q^\star((\varphi_{\mathrm{ge}}, \varphi_{\mathrm{oe}}), a) = \begin{cases} 0, & |\varphi_{\mathrm{ge}}| \leq \tau \\ \varphi_{\mathrm{ge}} - \tau, & \varphi_{\mathrm{ge}} - \tau > 0 > \varphi_{\mathrm{oe}} \\ -\varphi_{\mathrm{ge}} - \tau, & \varphi_{\mathrm{oe}} > 0 > \varphi_{\mathrm{ge}} + \tau \\ \varphi_{\mathrm{ge}} - \tau, & \varphi_{\mathrm{oe}} > \varphi_{\mathrm{ge}} - \tau > 0 \\ -\varphi_{\mathrm{ge}} - \tau, & 0 > \varphi_{\mathrm{ge}} + \tau > \varphi_{\mathrm{oe}} \\ 2\pi - \varphi_{\mathrm{ge}} - \tau, & \varphi_{\mathrm{ge}} - \tau \geq \varphi_{\mathrm{oe}} > 0 \\ 2\pi + \varphi_{\mathrm{ge}} - \tau, & 0 > \varphi_{\mathrm{oe}} \geq \varphi_{\mathrm{ge}} + \tau \end{cases} \tag{7}$$

Obviously, the values of $Q^\star$ are independent of the action. Figure 2 shows them graphically (to keep the graphics clear, we took $\tau = 0$ here) and demonstrates the existence of discontinuities.

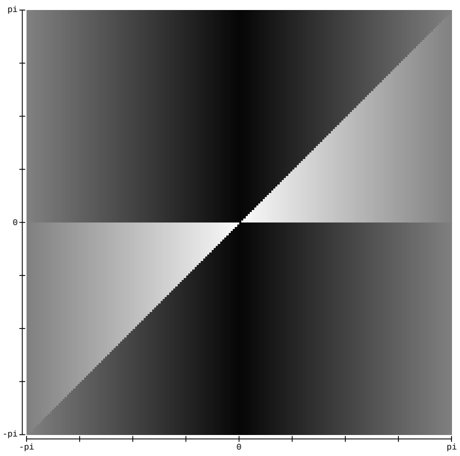

Fig. 2. LEFT: Values of $Q^\star$ for the one-jointed robot arm. RIGHT: Density plot of the left part of the figure. (dark = low costs, light = high costs)

If there were no obstacle (the state representation would then consist only in φ_{ge}), $Q^\star$ would be a continuous function with a maximum at $\varphi_{\mathrm{ge}} = \pi$. The obstacle introduces two discontinuities, one with regard to φ_{oe} alone (due to the differences of positioning the obstacle on the shorter or the longer path between goal and agent), one with regard to both state components simultaneously (due to the, in general, different lengths of the optimal paths starting from states immediately left and right of the obstacle). If the obstacle position were not integrated in state representation, only the second discontinuity would remain.

Further examples for discontinuous evaluation functions are shown in Figure 3. In the left part of the figure, the discontinuity occurs if the horizontal position of the obstacle is integrated in the state description; the optimal path and its costs change dramatically when the corridor becomes too narrow. For a two-joint arm with two obstacles in its workspace (right part of the figure), there are discontinuities even if the position of the (point form) obstacles are not included in the state description (large cost differences between the second segment of the arm being right or left of obstacle$_1$, and between any segment being left or right of obstacle$_2$). But it gets a lot more complex if they are included; for example, the optimal path changes if obstacle$_2$ is moved to the left until the arm cannot pass obstacle$_1$ any more by turning the first joint a bit to the right and turning the second one to the left. Thus it has to take a different, much more costly path to the goal.

In all cases, the cumulative evaluation function is discontinuous in the states. Since all such functions are defined at least on the state space, the problem of discontinuities through forbidden regions applies to every method dealing with delayed reinforcement, not only to Q-Learning.

Fig. 3. LEFT: A robot has to decide whether to go to the corridor or to circumvent the big obstacle. Here the discontinuity is at the least d at which the robot can pass the way. RIGHT: A two-jointed robot arm with two obstacles has a very complex structure of discontinuities in its cumulative evaluation function.

5 The Problem of Learning Evaluation Functions in Continuous Domains

We have shown in the last section that optimal cumulative evaluation function may be discontinuous in the presence of forbidden regions in state space. Furthermore in the case of continuous state spaces, lookup tables are not very useful. They might store all evaluations to some degree of precision, but they cannot generalize between them. It is obvious that practical applications can require a degree of discretization resulting in impractical sizes of the lookup table.

These considerations have led many researchers to use compact representations [12, 25]. They allow the handling of continuous-valued inputs quite naturally. Examples of such compact representations are parameterized structures [24, 25] like a backpropagation net [16], a polynomial data fitter or other neural networks. They are able to generalize or more technically interpolate among states, which means that the agent needs only to explore a fraction of states and infers the evaluation of unseen states from experience.

There are some general problems known with the use of compact representations, which also apply to learning evaluation functions:

1. The function approximator might be theoretically incapable of representing the optimal evaluation function to a sufficient degree.
2. The gradient descent method usually applied to train the function approximator might get stuck in a local optimum, even if the above case does not apply.

Additional problems arise in the context of the temporal difference updating method. There is currently no convergence proof available for continuous input spaces. Thus it is unknown wether training with temporal difference in the continuous case converges at least to a local optimum. We expect this situation to get worse a lot in the presence of discontinuous cumulative evaluation functions.

Apart from these problems, function approximators such as backpropagation nets have severe difficulties to simply represent discontinuous functions. Backpropagation nets represent inherently continuous functions and an approximation of a discontinuous functions would require the weights to grow arbitrarily large [13]. In such a case, the weights would be subject to oscillations. The explanation is that large weights imply a steep and sharp error surface and therefore large error derivatives. With a fixed learning rate, an overshooting [9, 13] of local minima in at least some dimensions happens. Anyway a limited number of hidden units allows the approximation of any function only to a certain precision.

6 Conclusion

Currently we see two general kinds of approaches to circumvent the problem of discontinuity, problem dependent and independent ones. The two problem independent approaches are

- to use discontinuous function approximators. Known discontinuous function approximators are competitive mechanisms, which assigns different single function approximators to different input regions [8, 20, 19]. Unclear is whether they could detect regions of discontinuities in input space.
- to design relative cumulative evaluation functions, which has been already mentioned by Tesauro [25] for other purposes.

Approaches using domain knowledge include the proper choice of coding schemes and representation issues. At least in simpler cases, it seems feasible to encode the inputs in a way that make discontinuities vanish. With a reformulation of the task representation in terms of a hierarchy and a structure in state and action spaces, the discontinuities might play a less significant role.

Another point is that in the presence of sensor noise or actuator nondeterminism uncertainty arises. This uncertainty leads to estimates of evaluations that mediate between neighbouring states, which flaten the discontinuities. The more uncertainty exists the smoother the evaluation function surface will be.

From the outlook presented above, we are quite optimistic. Nevertheless it is obvious that discontinuities of evaluation functions are a limitation to the applicability of reinforcement learning procedures to domains characterized above. The can prevent a solution from being optimal. In this context it is our belief that the experiment of Lin [12] has discontinuities and thus failed to show the desired performance. But it is beyond the scope of this paper to prove this. The message should be instead that one has to be aware of possible discontinuities and take them into account.

Acknowledgements

I gratefully acknowledge many discussions with Frank Weber and all other members of our research group. Discussions with Long-Ji Lin and Sebastian Thrun at CMU contributed to the conclusion of this work.

References

1. C. W. Anderson. Learning and problem solving with multilayer connectionist systems. Technical Report COINS TR 86-50, Dept. of Computer and Information Science, University of Massachusetts, Amherst, MA, 1986.
2. A. G. Barto. *Simulation Experiments with Goal-Seeking Adaptive Elements.* COINS, Amherst, Massachusetts, 1984. AFWAL-TR-84-1022.
3. A. G. Barto, S. J. Bradtke, and S. P. Singh. Real-time learning and control using asynchronous dynamic programming. Technical report, University of Massachusetts, Departement of Computer Science, Amherst MA 01003, August 1991.
4. A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. Technical Report COINS 89-95, Department of Computer Science, University of Massachusetts, MA, September 1989.
5. A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*, pages 539–602. MIT Press, Massachusetts, 1990.
6. R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.
7. P. Dayan. The convergence of TD(λ) for general λ. *Machine Learning Journal*, 8(3/4), May 1992. Special Issue on Reinforcement Learning.
8. D. Fox, V. Heinze, K. Möller, and S. B. Thrun. Learning by error driven decomposition. In *Proc. of NeuroNimes, France*, 1991.
9. G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.
10. R. E. Korf. Real-time heuristic search: New results. In *AAAI-88*, pages 139 – 143, 1988.
11. P. R. Kumar. A survery of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23:329–380, 1985.
12. L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning Journal*, 8(3/4), 1992. Special Issue on Reinforcement Learning.
13. A. Linden. Untersuchung von Backpropagation in konnektionistischen Systemen. Diplomarbeit, Universität Bonn, Informatik-Institutsbericht Nr. 80, 1990.
14. J. del R. Millán and C. Torras. A reinforcement connectionist approach to robot path finding in non-maze-like environments. *Machine Learning Journal*, 8(3/4), May 1992. Special Issue on Reinforcement Learning.
15. M. Minsky. Steps toward artificial intelligence. In E.A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 406–450. McGraw-Hill, 1961.
16. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II.* MIT Press, 1986.

17. A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, pages 210–229, 1959. Reprinted in E.Ã. Feigenbaum and J. Feldman (Eds.) 1963, *Computers and Thought*, McGraw-Hill, New York.
18. A. L. Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal on Research and Development*, pages 601–617, 1967.
19. F. J. Śmieja and H. Mühlenbein. Reflective modular neural network systems. Technical Report 633, GMD, Sankt Augustin, Germany, February 1992.
20. F.J. Śmieja. Multiple network systems (MINOS) modules: Task division and module discrimination. In *Proceedings of the 8th AISB conference on Artificial Intelligence, Leeds, 16–19 April, 1991*, 1991.
21. R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, 1984.
22. R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.
23. R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. To appear in: Proceedings of the Seventh International Conference on Machine Learning, June 1990, 1990.
24. R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement Learning is Direct Adaptive Optimal Control. In *Proceedings of the 1991 American Control Conference*, 1991.
25. G. Tesauro. Practical issues in temporal difference learning. *Machine Learning Journal*, 8(3/4), 1992. Special Issue on Reinforcement Learning.
26. S. B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, Pennsylvania, January 1992.
27. C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.
28. C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning Journal*, 8(3/4), May 1992. Special Issue on Reinforcement Learning.
29. P. Werbos. Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3:179–189, 1990.
30. P. Werbos and J. Titus. An empirical test of new forecasting methods derived from a theory of intelligence: The predicition of conflict in latin america. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8:657–666, 1978.
31. Paul J. Werbos. Backpropagation and neurocontrol: A review and prospectus. In *Proceedings of IJCNN89 Washington*, pages I 209–216, 1989.
32. S. D. Whitehead. A study of cooperative mechanisms for faster reinforcement learning. Technical Report 365, University of Rochester, Computer Science Department, Rochester, NY, March 1991.
33. S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. *Neural Computation*, 2:409–419, 1990.
34. R. J. Williams. Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, Boston, 1987.

This article was processed using the LaTeX macro package with LLNCS style