Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB'98):
From Animals to Animats 5, 223-228, R. Pfeiffer, B. Blumberg, J. A. Meyer and S. W. Wilson (eds.)

MIT Press/Bradford Books, 1998.

Efficient Model-Based Exploration

Marco Wiering, Jürgen Schmidhuber
IDSIA
Corso Elvezia 36, Lugano, Switzerland
marco@idsia.ch
juergen@idsia.ch

Abstract

Model-Based Reinforcement Learning (MBRL) can greatly profit from using world models for estimating the consequences of selecting particular actions: an animat can construct such a model from its experiences and use it for computing rewarding behavior. We study the problem of collecting *useful* experiences through exploration in stochastic environments. Towards this end we use MBRL to maximize exploration rewards (in addition to environmental rewards) for visits of states that promise information gain. We also combine MBRL and the Interval Estimation algorithm (Kaelbling, 1993). Experimental results demonstrate the advantages of our approaches.

1. Introduction

In reinforcement learning (RL) problems an animat repeatedly receives input from the environment, selects and executes an action, and may receive reinforcement. At a given time the animat's goal is to select the action leading to maximal future cumulative reward. RL-based methods essentially learn how much long-term reward the animat will receive on average for selecting a particular action in a particular state. The expected future cumulative reward intakes are stored in a value function which builds the basis for generating rewarding behaviors. Direct RL methods like Q-learning learn the value function from experimenting with the current policy without using a world-model. These methods have been shown to learn significantly slower in discrete state spaces than *model-based* RL (Moore and Atkeson, 1993), which first estimates the transition probabilities between states and the immediate rewards for these transitions, and then computes a policy by applying dynamic programming-like techniques (Bellman, 1961) to the estimated model.

The problem. Since some RL animat is only able to learn from what it has experienced, the success of its policy heavily depends on the utility of its experiences. Optimal experimental design (Fedorov, 1972; Dodge et al., 1988) tries to gather those experiences which are most useful for computing good approximate solutions. In RL the problem of selecting alternative non-greedy actions is called exploration or dual control (Dayan and Hinton, 1993). Deviating from the current greedy policy (which always selects the action with the highest Q-value), however, usually causes some loss of immediate reinforcement intake. The animat faces the problem of spending as little time as possible on exploration without affecting its ability to find optimal policies.

Previous work. Schmidhuber (1991a) and Thrun (1992) present comparisons between *directed* and *undirected* exploration methods. *Directed* exploration methods use special exploration-specific knowledge to guide the search through alternative policies. *Undirected* exploration methods use randomized action selection methods to guess useful experiences. Previous research has shown significant benefits of using directed exploration (e.g., Schmidhuber 1991a; Thrun 1992; Storck, Hochreiter and Schmidhuber, 1995, Wilson 1996, Schmidhuber 1997). Koenig and Simmons (1996) shows how undirected exploration techniques can be improved by using the action-penalty rule which makes unexplored actions look more promising — this decreases the advantage of directed exploration.

The Interval Estimation (IE) algorithm (Kaelbling, 1993) uses second order statistics to detect whether certain actions have a potential of belonging to the optimal policy. IE computes confidence intervals of Q-values and always selects the action with largest upper interval boundary. Previous results (Kaelbling, 1993) show that IE works well for action selection in bandit problems (Berry and Fristedt, 1985).

Model-based exploration. Directed exploration methods learn an *exploration value function* in the same way standard RL methods learn a problem-oriented value function. Therefore we may simply define an exploration reward function determining immediate exploration rewards and let the selected RL method learn

exploration Q-values. Most previous methods use Q-learning for learning where to explore, e.g., (Schmidhuber, 1991a; Thrun, 1992; Storck et al., 1995). We extend this work by using model-based RL (MBRL) to learn exploration policies. Since MBRL can outperform its direct RL counterpart (Sutton, 1990; Moore and Atkeson, 1993) we expect that it can also improve learning where to explore: model-based exploration is promising because it allows for immediately selecting actions based on global (as opposed to local) expected information gain.

Objective. Finding optimal solutions is considered a hard and unprosperous activity in artificial intelligence. We will focus on a more modest goal: given some RL problem, how can we learn with minimal experiences a policy whose performance is not more than ϵ percent below the optimum? Kearns and Singh (1998) proved that this goal can be achieved in polynomial time for general Markov decision problems. We will use a slightly adapted version of prioritized sweeping (PS) (Moore and Atkeson, 1993) to learn both an exploration policy and a problem-oriented policy, and combine this approach with (a) frequency-based and recency-based exploration reward functions, and (b) our novel Model-Based Interval Estimation (MBIE) update rule, which combines IE and MBRL.

Outline. Section 2 describes MBRL and introduces our version of prioritized sweeping. Section 3 addresses exploration issues in RL and mentions the exploration reward rules used in the experiments. Section 4 addresses issues related to MBRL for learning exploration policies and introduces MBIE. Section 5 describes experimental results on a task involving a 50×50 maze with one optimal goal and two suboptimal ones. Section 6 concludes.

2. Model-Based Reinforcement Learning

2.1 Markov decision problems

We consider discrete time steps $t = 1, 2, 3, \ldots$, a finite set of states $S = \{S_1, S_2, S_3, \ldots, S_n\}$ and a finite set of actions A. Let s_t denote the state at time t, and $a_t = \Pi(s_t)$ the action, where Π represents the learner's policy mapping states to actions. The transition function P with elements $P_{ij}(a) := p(s_{t+1} = j | s_t = i, a_t = a)$ for $i, j \in S$ defines the transition probability to the next state s_{t+1} given s_t and a_t. A reward function R maps state/action pairs (SAPs) $(i, a, j) \in S \times A \times S$ to scalar reinforcement signals $R(i, a, j) \in \mathbb{R}$. The reward at time t is denoted by r_t. A discount factor $\gamma \in [0, 1]$ discounts later against immediate rewards. The controller's goal is to select actions which maximize the expected long-term cumulative discounted reinforcement, given an arbitrary initial state $\in S$. The value $V^{\Pi}(s)$ is a prediction of the expected discounted cumulative reward to be received in the future, given that the process is currently in state s and policy Π will be used in the future:

$$V^{\Pi}(i) = E(\sum_{k=0}^{\infty} \gamma^k R(s_k, \Pi(s_k), s_{k+1}) | s_0 = i)$$

Action evaluation functions (Q-functions) $Q^{\Pi}(i, a)$ return the expected future discounted reward for current state i, current action a, and subsequently executed policy Π:

$$Q^{\Pi}(i, a) = \sum_j P_{ij}(a)(R(i, a, j) + \gamma V^{\Pi}(j))$$

2.2 Estimating a model

Inducing a model from experiences can simply be done by counting the frequency of observed experiences. Towards this end our animat uses the following variables:

$C_{ij}(a)$:= nr. of transitions from state i to j after executing action a.
$C_i\ (a)$:= number of times the animat has executed action a in state i.
$R_{ij}(a)$:= sum over all immediate rewards received after executing action a in state i and stepping to state j.

A maximum likelihood model (MLM) is computed as follows:

$$\hat{P}_{ij}(a) := \frac{C_{ij}(a)}{C_i(a)} \text{ and } \hat{R}(i, a, j) := \frac{R_{ij}(a)}{C_{ij}(a)} \qquad (1)$$

After each experience the variables are adjusted and the MLM is updated. In deterministic environments one experience per SAP is sufficient to infer the true underlying model. In stochastic environments, however, we need resampling. For resampling a good exploration strategy is essential.

2.3 Prioritized Sweeping (PS)

Dynamic programming (DP) techniques could immediately be applied to the estimated model, but online DP tends to be computationally very expensive. To speed up DP algorithms, some sort of efficient update-step management should be performed. This can be done by prioritized sweeping (PS) (Moore and Atkeson, 1993) which assigns priorities to updating the Q-values of different state/action pairs (SAPs) according to a heuristic estimate of the update sizes. PS keeps track of a *backward model* relating states to predecessor SAPs. Following the update of a state-value, the state's predecessors are inserted in a priority queue. Then the priority queue is used for updating Q-values of actions of those states with highest priority.

Our PS. Moore and Atkeson's PS (M+A's PS) calculates the priority of some state by checking all transitions to updated successor states and identifying the

one whose update contribution is largest. Our variant allows for computing the *exact* size of updates of state values since they have been used for updating the Q-values of their predecessors, and yields more appropriate priorities. Unlike our PS, M+A's PS cannot detect large state-value changes due to many small update steps, and will forget to process the corresponding states.

Our implementation uses a set of predecessor lists $Preds(j)$ containing all predecessor states of state j. We denote the priority of state i by $|\Delta(i)|$, where the value $\Delta(i)$ equals the change of $V(i)$ since the last time it was processed by the priority queue. To calculate it, we constantly update all Q-values of predecessor states of currently processed states, and track changes of $V(i)$. The details are as follows:

Our Prioritized Sweeping:

1) Promote the most recent state k to the top of the priority queue
2) $\forall\ a$ do:
   3 $Q(k,a) := \sum_j \hat{P}_{kj}(a)(\hat{R}(k,a,j) + \gamma V(j))$
4) While $n < U_{max}$ AND the queue is not empty
   5 Remove the top state s from the queue
   6 $\Delta(s) := 0$
   7 $\forall$ Predecessor states i of s do:
      8 $V'(i) := V(i)$
      9 $\forall\ a$ do:
         10 $Q(i,a) := \sum_j \hat{P}_{ij}(a)(\hat{R}(i,a,j) + \gamma V(j))$
      11 $V(i) := \max_a Q(i,a)$
      12 $\Delta(i) := \Delta(i) + V(i) - V'(i)$
      13 If $|\Delta(i)| > \epsilon$
         14 Promote i to priority $|\Delta(i)|$
   15 $n := n + 1$
16) Empty priority queue, but keep the $\Delta(i)$ values

Here U_{max} is the maximal number of updates to be performed per update-sweep. The parameter $\epsilon \in \mathbb{R}^+$ controls update accuracy. Note that another difference to M+A's PS is that we remove all entries from the queue after having processed the maximal number of states.

3. Exploration

3.1 Undirected Exploration

Undirected exploration methods rely on pseudo-random generators, e.g., the *Max-random* and *Boltzmann* exploration rules. We will use Max-random which outperformed Boltzmann in a number of experiments (Thrun, 1992; Caironi and Dorigo, 1994).

Max-random exploration rule. The Max-random rule, also known as ϵ-greedy (Sutton, 1996) or pseudo-stochastic (Caironi and Dorigo, 1994), is the simplest one. It uses a single parameter P_{max} denoting the probability of selecting the action with highest Q-value, and selects a random action otherwise. In case there are multiple actions with highest Q-value, we select one of them stochastically.

3.2 Directed Exploration

Directed exploration techniques use knowledge of what an animat has learned to direct the exploration behavior to the most interesting parts of the state space. All they require is a local reward function determining which experience is interesting (e.g. Schmidhuber, 1991a,b, 1997). It takes the place of the standard MDP reward function. The MDP transitions and the experiences stay the same, but now we learn two Q-functions: the *exploration Q-function* and the *exploitation Q-function*. Here we look at two types of exploration: recency-based, and frequency-based.

Recency-based. Select the action which has been selected least *recently*. The local reward function for exploring SAP (s_t, a_t) is:

$$R^E(s_t, a_t, *) := \frac{-t}{K_T}, \tag{2}$$

where K_T is a scaling constant and t the current time step. The asterisk stands for the don't-care symbol — the exploration reward is *local* because it depends on the current state/action pair only.

Frequency-based. Explore actions which have been executed least *frequently*. The local reward function is simply:

$$R^E(s_t, a_t, *) := -\frac{C_{s_t}(a_t)}{K_C}, \tag{3}$$

where K_C is a scaling constant.

4. Learning Exploration Models

We can use all RL methods to learn exploration value functions. While most previous approaches used standard Q-learning for learning it (Schmidhuber, 1991a; Thrun, 1992; Storck et al., 1995), we prefer to use prioritized sweeping (PS) instead. PS allows for quickly learning exploration models which may be useful for learning Q-values estimating global information gain, taking into account yet unexplored regions of the state-space.

Replacing reward. The nice thing about using MBRL is that all explorative transition rewards can be based entirely on the current reward. Suppose that an animat selects an action which has already been executed several times. Computing the local exploration reward by averaging over all previous transition rewards would not result in the desired reward measure. For instance, with frequency-based exploration rewards the exploration model's estimate $\hat{R}^E(i,a,*)$ would be the average over $1, 2, \ldots, C_i(a)$ instead of just $C_i(a)$. Direct RL cannot circumvent this problem, but with MBRL we can just replace the estimated reward $\hat{R}(i,a,j)$ by $R^E(i,a,j)$ for all j with $\hat{P}_{ij}(a) > 0$, that is, we update all rewards for

outgoing transitions from the current state/action pair to take the latest available information into account.

Never-ending exploration. The exploration utilities continually change; there is no such thing as a stable, optimal exploration function. This is not a fundamental problem — in fact the goal of exploration is to search for alternative paths in order to find better and better policies; therefore the exploration policy should never converge. In particular applications, however, e.g., in limited life-time scenarios (Berry and Fristedt, 1985; Thrun, 1992; Schmidhuber et al., 1997), we want to increase or switch to exploitation after some time.

Interval Estimation. To explore efficiently, an animat should not repeatedly try out actions that certainly cannot belong to the optimal policy. To reduce the set of optimal action candidates we extend the interval estimation algorithm (IE) (Kaelbling, 1993) to make it amenable for MBRL.

Standard IE selects the action with the largest upper bound for its Q-value. To compute upper bounds it keeps track of the means and standard deviations of all Q-values.

MBIE, however, uses the model to compute the upper bound of Q-values. Given a set of outgoing transitions from SAP (i, a), MBIE increases the probability of the best transition (the one which maximizes $\gamma V(j) + R(i, a, j)$), depending on its standard deviation. Then MBIE renormalizes the transition probabilities and uses the result for computing the Q-values. The following algorithm can be substituted for line 3 and 10 in our PS algorithm:

Model Based Interval Estimation:

a $m := Argmax_{j:\hat{P}_{ij}(a)>0}\{R(i,a,j) + \gamma V(j)\}$
b $n := C_i(a)$
c $P := \hat{P}_{im}(a)$
d $P^+_{im}(a) := (P + \frac{z^2_\alpha}{2n} + \frac{z_\alpha}{\sqrt{n}}\sqrt{P(1-P) + \frac{z^2_\alpha}{4n}})/(1 + \frac{z^2_\alpha}{n})$
e $\Delta_P := P^+_{im}(a) - \hat{P}_{im}(a)$
f $\forall j \neq m$
g $P^+_{ij}(a) := \hat{P}_{ij}(a) - \frac{\Delta_P C_{ij}(a)}{C_i(a) - C_{im}(a)}$
h $Q(i,a) := \sum_j P^+_{ij}(a)(\hat{R}(i,a,j) + \gamma V(j))$

Here z_α is a variable which determines the size of the confidence bounds. Step 8d elaborates on the commonly used $z_\alpha\sqrt{P(1-P)/n}$, and is designed to give better results for small values of n — see (Kaelbling, 1993) for details.

MBIE hybrids. Although IE seems promising it does not clearly outperform Q-learning with Boltzmann exploration due to problems of estimating the variance of a changing Q-function in the beginning of the learning phase (Kaelbling, 1993). Since MBIE also relies on initial statistics we propose to circumvent such problems by starting out with some other exploration method and switching to IE once some appropriate condition holds.

This is done as follows: we start with frequency-based exploration and keep tracking the cumulative change of the problem-oriented value function. Once the average update of the V^{Π} function (computed over the most recent N time-steps) falls below $\eta \in \mathbb{R}^+$, we (I) copy the rewards and Q-values from the problem-oriented model to the exploration model, and (II) switch to IE: we apply asynchronous value iteration (Bellman, 1961) to the model; the iteration procedure calls MBIE for computing Q-values and ends once the maximal change of some state value is less than $\epsilon \in \mathbb{R}^+$.

Simultaneous policy learning. The model-based learner simultaneously learns both exploration policy and problem-oriented policy. After each experience we update the model and use PS to recompute the value functions.

5. Experiments

5.1 Maze-tasks

To compare the different RL methods discussed in previous chapters we use a 50×50 maze shown in Figure 1. It consists of about 20% blocked states and 20% punishing states (these are inserted randomly). Blocked states are represented by black fields, penalty states by grey fields, free states by white fields. In each state the animat can select one of four actions: *go north, go east, go south, go west.* The starting state (S) is located 1 field north/east of the south-west corner. There are three absorbing goal states, two of them are suboptimal. The optimal goal state (G) is located 1 field south/west of the north-east corner, the suboptimal goal states (F) are located in the north-west and south-east corners. Once the animat hits the goal it is reset to its initial position. Selected actions are replaced by random actions with 10% probability.

Reward function. Actions leading to a blocked state are not executed and punished by a reward of -2. Steps leading to free states are punished by a reward of -1. Actions leading to a penalty state yield a reward of -10. If the animat finds the optimal goal state it will receive a reward of 1000. For finding a suboptimal goal state it gets a reward of 500. The discount factor γ is 0.99.

Comparison. We compare the following exploration methods: Max-random, directed model-based exploration techniques using frequency-based and recency-based reward rules, and MBIE. The latter starts out with model-based exploration using the frequency-based reward rule, and switches to IE once the value function hardly changes any more.

The goal is to learn good policies as quickly as possible. We computed an optimal policy using value iteration (Bellman, 1961) and tested this optimal policy by executing it for 1,000 steps. We computed its average

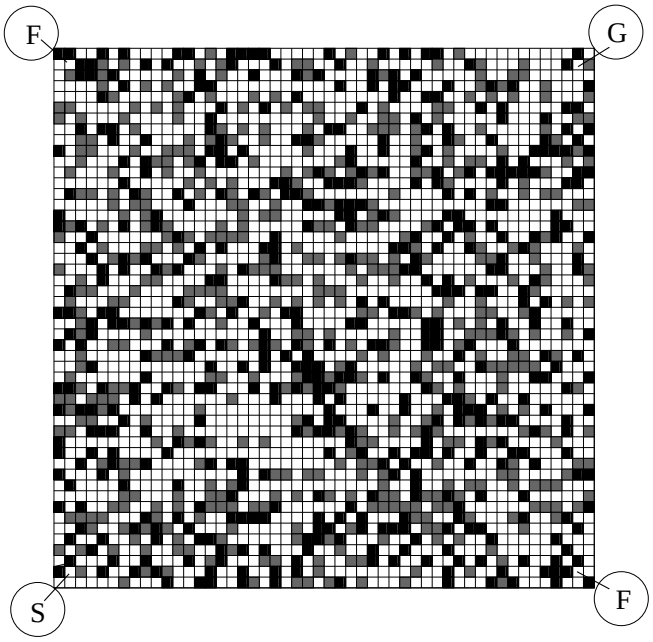

Figure 1 *The 50 × 50 maze used in the experiments. Black squares denote blocked fields, grey squares penalty fields. The animat starts at S and searches for a minimally punishing path to the optimal goal G. Good exploration is required to avoid focusing on suboptimal goal states (F).*

reinforcement intake by testing it 10,000 times, which resulted in 7590 ± 2.

For each method we conduct 20 runs of 100,000 learning steps. During each run we measure how quickly and how often the animat's policy collects 95%, 99% and 99.8% of what the optimal policy collects. This is done by averaging the results of 1000 test runs conducted every 2000 learning steps — each test run consists of executing the greedy policies (always selecting actions with maximal Q-value) for 1000 steps.

Parameters. We set the accuracy parameter $\epsilon := 0.1$, and $U_{max} := 100$ for both learning the problem-oriented and exploration value functions. The exploration reward's discount factor γ is set to 0.99 for frequency-based and to 0.95 for recency-based exploration. The constant K_C (used by the frequency-based reward rules) is set to 50; K_T (used by the recency-based reward rule) is set to 1000. We used two values for P_{max} (for Max-random exploration): 0.2 and 0.4. The value of z_α (for MBIE) is set to 1.96 (which corresponds to a confidence interval of 95%). The combination of MBIE and model-based exploration switches to MBIE once the value function has not changed by more than 0.1 (η) on average within the 1000 (U) most recent steps.

5.2 Results

Table 1 shows significant improvements achieved by learning an exploration model. The undirected exploration methods focus too much on suboptimal goals (which are closer and therefore easier to find). This often prevents them from discovering near-optimal policies. On the other hand, exploration model-based learning does favor paths leading to the optimal goal. Using the frequency-based reward rule by itself, the animat always finds the optimal goal although it fails to always find 99.8% optimal policies. Figure 2 shows a large difference between learning performances of exploration models and max-random exploration. It does not clearly show the distinction between frequency-based or recency-based exploration and the extension with MBIE. Switching to MBIE after some time (between 35,000 and 55,000 steps), however, significantly improves matters. First of all, Table 1 shows that this strategy finds optimal or near-optimal policies in 90% of the cases, whereas the others fail in at least 50% of the cases. The second improvement with MBIE is shown in Table 2. MBIE collects much more reward during training than all other exploration methods, thereby effectively addressing the exploration/exploitation dilemma. In fact it is the only exploration rule leading to a positive cumulative reward score.

Exploration Rule	95% (freq)	99% (freq)	99.8% (freq)
MBIE	25K (20)	42K (19)	66K (18)
Frequency-based	24K (20)	50K (16)	66K (10)
Recency-based	27K (19)	55K (18)	69K (9)
Max-random 0.2	43K (4)	52K (4)	68K (4)
Max-random 0.4	— (0)	— (0)	— (0)

Table 1 *The number of steps required by several exploration methods for obtaining p-optimal policies (and how many runs found them at all) (K stands for 1000).*

Exploration Rule	Best run result	Training Performance
MBIE	7.57K ± 0.05K	350K ± 40K
Frequency-based	7.55K ± 0.06K	-45K ± 9K
Recency-based	7.54K ± 0.11K	-120K ± 10K
Max-random 0.2	4.8K ± 1.4K	-190K ± 16K
Max-random 0.4	4.1K ± 0.3K	-62K ± 19K

Table 2 *Average and standard deviation of the best test result during a run and the total cumulative reward during training.*

6. Discussion

Undirected exploration relies on a random generator for selecting actions. When applied to tasks with multiple absorbing goal states, it faces major difficulties in finding the optimal one. Our exploration models, however, allow for globally maximizing exploration rewards. This allows for discovering good policies circumventing suboptimal goal states. We compared two types of exploration rewards: frequency-based, and recency-based. Although frequency-based exploration performed slightly better in our stationary environments, recency-based reward rules

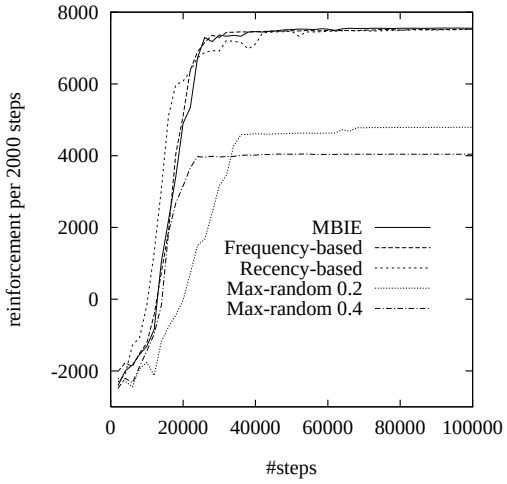

Figure 2 *Cumulative reward during test runs, averaged over 20 simulations.*

may make more sense in non-stationary ones.

Optimal exploration. When it comes to finding optimal policies, methods using external MDP rewards for focusing on actions with large probability of being optimal will often save some SAP visits over methods based solely on local exploration reward rules. Furthermore, cumulative external MDP rewards obtained during training should be taken into account in attempts at approaching the exploit/explore dilemma. That is why we introduced MBIE, a method combining Kaelbling's interval estimation (IE) algorithm (Kaelbling, 1993) and model-based reinforcement learning (MBRL). Since MBIE heavily relies on initial statistics, we switch it on only after an initial phase during which an exploration model is learned (according to, say, the frequency-based local exploration reward rule). In our experiments this approach almost always led to 99.8%-optimal policies.

Future work: function approximators. Our ideas have been presented in the context of discrete state spaces. We can extend them to function approximators in the same way we can extend model-based RL methods to learning in continuous state spaces. This is an interesting topic for future research.

7. Acknowledgments

Many thanks to Rafał Sałustowicz, Nic Schraudolph, and Jieyu Zhao for helpful discussions. We are thankful to the Swiss Center for Scientific Computing (CSCS/SCSC) for providing additional computing power. This work was supported in part by SNF grant 2100-49'144.96 "Long Short-Term Memory".

References

Bellman, R. (1961). *Adaptive Control Processes.* Princeton University Press.

Berry, D. and Fristedt, B. (1985). *Bandit Problems: sequential allocation of experiments.* Chapman and Hall, London/New York.

Caironi, P. V. C. and Dorigo, M. (1994). Training Q-agents. Technical Report IRIDIA-94-14, Université Libre de Bruxelles.

Dayan, P. and Hinton, G. (1993). Feudal reinforcement learning. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 5*, pages 271–278. San Mateo, CA: Morgan Kaufmann.

Dodge, Y., Fedorov, V., and Wynn, H., editors (1988). *Optimal Design and Analysis of Experiments: Proceedings of First International Conference on Optimal Design and Analysis of Experiments.* Elsevier Publishers.

Fedorov, V. V. (1972). *Theory of optimal experiments.* Academic Press.

Kaelbling, L. (1993). *Learning in Embedded Systems.* MIT Press.

Kearns, M. and Singh, S. (1998). Near-optimal performance for reinforcement learning in polynomial time. Retrievable from http://www.research.att.com/∼mkearns/.

Koenig, S. and Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22:228–250.

Moore, A. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130.

Schmidhuber, J. (1991a). Curious model-building control systems. In *Proc. International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE.

Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredom in model-building neural controllers. In Meyer, J. A. and Wilson, S. W., editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books.

Schmidhuber, J. (1997). What's interesting? Technical Report IDSIA-35-97, IDSIA.

Schmidhuber, J., Zhao, J., and Wiering, M. (1997). Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28:1:105–130.

Storck, J., Hochreiter, S., and Schmidhuber, J. (1995). Reinforcement driven information acquisition in nondeterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 2, pages 159–164. EC2 & Cie, Paris.

Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting based on dynamic programming. In *Machine Learning: Proceedings of the Seventh International Workshop.*

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. M. and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1038–1045. MIT Press, Cambridge MA.

Thrun, S. B. (1992). The role of exploration in learning control. In *Handbook of Intelligent Control: Neural, Fuzzy*

and Adaptive Approaches. Florence, Kentucky 41022: Van Nostrand Reinhold.

Wilson, S. W. (1996). Explore/exploit strategies in autonomy. In Meyer, J. A. and Wilson, S. W., editors, *Proc. of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4*, pages 325–332. MIT Press/Bradford Books.