

Apprenticeship Learning for Initial Value Functions in Reinforcement Learning

Frederic Maire

Faculty of Information Technology
Queensland University of Technology
Box 2434, Brisbane Q 4001
Australia
f.maire@qut.edu.au

Vadim Bulitko

Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8
Canada
bulitko@ualberta.ca

Abstract

Reinforcement Learning has had spectacular successes over the last several decades. While meant to require less human input than supervised learning, reinforcement learning can be substantially accelerated with *a priori* available domain expertise. The ways of providing human knowledge to a reinforcement learning agent vary from crafting state features to initial policy design to initial value function design. We chose the latter and propose a novel approach for acquiring a high-quality initial value function via apprenticeship learning. This approach works well in domain when a body of expert data are available. Our apprentice reinforcement learning (ARL) agent uses dynamic programming to compute values for the states visited by the expert. A Laplacian regularizer is then engaged to extrapolate these onto the entire state space. The result of this process is a high-quality initial value function to be further refined by any value-function based reinforcement learning method. In a grid world domain, ARL was able to speed up TD(λ) learning method by a factor of two from a single observed expert's trace.

1 Introduction and Related Research

For some application domains of reinforcement learning, demonstrations by an expert are readily available. For instance, extensive databases of games between strong players can be easily found for popular board games like chess, go or hex. The task of learning by observing an expert is called *apprenticeship learning*. Most apprenticeship learning approaches try to mimic the observed experts by applying a supervised learning algorithm to learn a direct mapping from the states to the actions as pioneered by [Sammut *et al.*, 1992]. A recent alternative to this is to assume a reward-maximizing nature of the expert and then estimate the unknown reward function employed by the expert as a linear combination over given state features [Abbeel and Ng, 2004]. The premise of this approach is that the reward function, rather than the policy or the value function, is the most succinct, robust, and transferable definition of the task.

Work on *reward shaping* demonstrates that a well-chosen reward function can, indeed, significantly speed reinforcement learning [Laud and DeJong, 2003]. Furthermore, *safe* shaping rewards are differentials in a potential function [Ng *et al.*, 1999]. Therefore, by supplying a non-trivial potential function, the user can speed the learning process. This is not surprising since learning with shaping rewards derived from potential function is equivalent to learning without the shaping rewards but with the initial value function being equal to potential function [Wiewiora, 2003]. For instance, learning with the optimal value function V^* as the potential function is equivalent to learning with V^* as the initial value function – in both cases no further learning is needed.

Over the years, a number of methods for acquiring a non-trivial initial value function have been proposed. They vary from hand-engineering [Korf and Taylor, 1996; Ng *et al.*, 1999] and solving abstracted problems [Holte *et al.*, 1994; Korf and Felner, 2002] to artificial evolution [Ackley and Littman, 1991].

In this paper, we propose a novel method for deriving high-quality initial value functions from existing demonstrations by an expert. Our apprentice learning agent first builds an auxiliary graph whose vertices are points in the state space and whose arcs are transitions used by the expert. After estimating the state-values of the vertices visited by the expert, we extrapolate these values onto all other states with a graph Laplacian operator. This step requires a similarity measure over the state space. The resulting value function constitutes a starting point for any value function based reinforcement learning method. As the initial value function is substantially more informative than a random or optimistic value function frequently used with RL methods, the remaining on-line learning process is substantially accelerated. We evaluate the effectiveness of deriving an initial value function in a standard grid-world domain and find that a *single* observed demonstration by an expert speed up learning as much as 300 additional TD(λ) episodes. This constitutes a two-fold speed-up.

The rest of the paper is organized as follows. In section 2 we detail our method. Section 3 sets up the experimental environment and presents results of the empirical evaluation. A discussion of future research follows.

2 ARL: The Proposed Approach

The presentation will be tailored to non-discounted episodic reinforcement learning tasks with a single absorbing state (called the *goal state* henceforth). While this setting covers a number of planning tasks [Bonet *et al.*, 1997; Aberdeen *et al.*, 2004], combinatorial puzzles [Korf, 1997; Korf and Felner, 2002], games [Tesauro, 1995; Schaeffer *et al.*, 2001], path-finding [Ng *et al.*, 1999; Shimbo and Ishida, 2003; Koenig, 2004] and other applications [Kohl and Stone, 2004], our approach can be extended to handle multiple goal states as well as non-episodic tasks.

The agent operates on a directed weighted graph whose vertices are agent’s states. A single state is designated as the goal state. Edges correspond to actions available to the agent. By traversing an edge, the agent incurs a negative reward equal to the edge’s weight. Arriving at the goal state ends the episode and does not deliver any additional reward. The learning task is to learn a state value function such that following this function greedily from any state will maximize the collected reward. This is equivalent to following a shortest path from any start state to the goal.

Notationally, the sum of negative rewards (i.e., the costs) incurred along a path P between states a and b is denoted as $\text{dist}_P(a, b)$. As usual, the value $V_\pi(s)$ of a state s with respect to the policy π is the expectation of the cumulative cost incurred by the agent when following the policy π from the state s .

2.1 Step 1: Collecting Observations

The observed demonstrations by an expert are encoded in the form of trajectories in the state space. The premise of our approach is that an expert agent will follow trajectories that are close to optimal. On this premise, we induce a partial order over the states visited by the expert. The training set (i.e., the set of state trajectories) therefore induces a directed subgraph on the set of visited states. If a sequence of states from an expert agent has a state repeated, we prune out all states between the two occurrences. That is, the sequence $(\dots, s_{t-1}, s_t, s_{t+1}, \dots, s_{t+m}, s_{t+m+1}, \dots)$ with state $s_t = s_{t+m}$ will become $(\dots, s_{t-1}, s_t, s_{t+m+1}, \dots)$.

Once all cycles have been eliminated, we label each state in each sequence with its distance from the goal state. We then merge multiple sequences labeled with their distances into one directed labeled graph. If several sequences pass through the same state, we use the minimal distance as the final label of this state. In other words, we compute the shortest weighted paths to the goal state in this auxiliary graph using Dijkstra algorithm. At this stage, the value of a state that has been visited by an expert is approximated by $\hat{h}(s)$, the length of a shortest path from the goal state to s .

2.2 Step 2: Generalization with a Graph Laplacian

Several approaches in the past generalized high-quality values of sample states onto the entire space. In [Levner and Bulitko, 2004], the researchers applied reinforcement learning methods to acquire a computer vision policy in the domain of tree canopy recognition. A training set of aerial images was used as the starting states of an RL-agent. Since the

“ground truth” was provided for each training image, a rollout (i.e., full subtree expansion) technique allowed to compute the exact values of each input and subsequent states. The resulting very sparse samples were generalized onto the entire abstracted state space using Artificial Neural Networks and hand-crafted state features.

In [Bulitko and Wilkins, 2003], Artificial Neural Networks were used to generalize the values of damage control status (i.e., state) of a naval vessel computed from past scenarios onto the entire state space. Again, hand-built state features were used to reduce the dimensionality of the state space.

Our setting is different from the first approach mentioned above as we assume having access to traces of expert behavior as opposed to the final state that the expert arrived at. An additional difference from this and the second approach is that we take an advantage of the similarity between states by using a Graph Laplacian. This allows us to induce high-quality value functions from a single expert trajectory whereas both aforementioned systems required large bodies of labeled images or damage control scenarios.

We will now present the details of Step 2 of the ARL approach. The training set $\{[s, \hat{h}(s)]\}$ computed in Step 1 is now fed into a Graph Laplacian regularizer [Chung-Graham, 1997] to generalize the data onto all other states. To illustrate the role of the graph Laplacian, consider the toy-sized graph of Figure 1 where vertex 5 represents the goal state. Given the values of the vertices on the path $(7, 4, 2, 1, 5)$, traversed by the expert, we would like to generalize the values of these vertices to non-visited vertices. That is, we are looking for reasonable values for vertex 3 and vertex 6. The value v_3 should be related to the values v_2 and v_7 of its neighbours. Similarly, the value v_6 should be related to v_1, v_4 and v_7 . We determine the values v_3 and v_6 by minimizing the Laplacian cost function:

$$(v_3 - 2)^2 + (v_3 - 4)^2 + (v_6 - 1)^2 + (v_6 - 3)^2 + (v_6 - 4)^2.$$

From here the optimal values are computed as $v_3 = 3$ (which happens to be its correct value) and $v_6 = 2.667$ (which exceeds its correct value of 2). If there were an edge between vertex 3 and vertex 6, we would add the term $(v_3 - v_6)^2$ to this cost function. In general, the cost function to be minimized is of the form:

$$\frac{1}{2} \tilde{v}^T H \tilde{v} - f^T \tilde{v}$$

where \tilde{v} represents the vector of the values of the states not visited by the experts. The solution for \tilde{v} is $H^+ f$, where H^+ is the pseudo-inverse of H . Alternatively, we can formulate the optimization problem as a quadratic program on v (the full vector) with equality constraints for the visited states. The values of the states visited by the experts are clamped to the values found in Step 1. The pseudo-code of Step 2 is found in Algorithm 1.

The use of a set of non-visited states is motivated by the fact we may have access to valuable information on the entire state space. For instance, in game playing, such information is in the form of a database of games where some games were played by experts and other games by non-expert players. The board positions reached by non-expert players are still relevant to learning a strong program for this game as

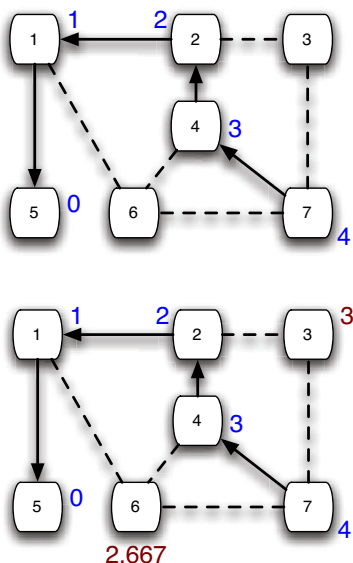


Figure 1: **Top:** a hand-traceable micro-graph used to illustrate the first two steps of our ARL algorithm. The vertex labels are inside the nodes (1 through 6). The expert path from vertex 7 to vertex 5 (the goal) is shown by the solid arrows. The numbers next to vertices of the path are the values of the vertices \hat{h} computed in Step 1. **Bottom:** the values estimated for vertices 3 and 6 in Step 2 are shown.

they correspond to states likely to be visited. If we have a reasonable quality similarity measure between board positions, then we can derive a weighted graph similar to the graph of Figure 1. The weights of the edges are the similarity measures between states and the Laplacian will be defined for the weighted graph. Then we can induced some initial values of the board positions for the games of non-expert players. Despite not having seen expert behavior in these states, the initial values can still be expected to be of a higher quality than a random initial values.

As usually done within the semi-supervised learning framework [Zhu *et al.*, 2005], the machine learning module is not only given a labeled data set consisting of input-output pairs $\{(x_1, v_1), \dots, (x_l, v_l)\}$, but also a (typically

Algorithm 1 $V = \text{InitializeValueFunction}(T)$

Require: T , a collection of experts' trajectories in a discrete state space.

Ensure: V is an initial value function.

- Build auxiliary digraph by inserting an arc between state s_i and s_j whenever s_j is a successor state of s_i in a expert trajectory of T .
 - Compute the distance to a goal of each visited state vertex in the auxiliary digraph.
 - Clamping the computing value of visited states, determine the vector V of values of the other states by minimizing the Laplacian subject to the clamping constraint.
-

much larger) unlabeled data set $\{x_{l+1}, \dots, x_m\}$ where the x_i 's are in some general input space and the v_i 's are real values. In addition to this training set T , we are given a graph of similarity on the input state space. That is, two vertices are connected by an edge if they are deemed similar. From the adjacency relation of the graph, the Laplacian operator L is defined as the matrix whose entries satisfy:

$$L_{i,j} = \begin{cases} 1 & \text{if } i \sim j \\ -d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where d_i represents the degree of vertex i , and $i \sim j$ means that vertex i is adjacent to vertex j . It is easy to check that:

$$v^T L v = -\frac{1}{2} \sum_{i \sim j} (v_i - v_j)^2.$$

The scalar $v^T L v$ quantifies how much v varies locally, or how smooth v is over the vertex set.

In [Smola and Kondor, 2003], graph Laplacians are put into the principled framework of Regularization Theory and a family of regularization operators (equivalently, kernels) on graphs that include Diffusion Kernels is proposed. It is worth mentioning that the approach we propose herein can be extended to any of these regularization operators. For instance, consider the diffusion kernel K and the training set T as above. Then we would be maximizing $v^T L v$ subject to $v_i = \hat{h}(x_i)$ for $i \in [1 : l]$.

2.3 Step 3: On-line Refinement via RL

The result of Step 2 is the initial value function $V_0(s) = -h(s)$. This function is then refined during the on-line reinforcement learning with a value-function based algorithm such as the simple value iteration or TD(λ) [Sutton and Barto, 1998]. The initial value function computed during Steps 1 and 2 captures certain expert knowledge induced from the recorded experiences. We believe that our approach is superior to the learning by value iteration by replaying the observed state trajectories of the experts. Indeed, assume we have two experts A and B such that A makes a good decision in the state a and a lower quality decision in the state b . On the other hand, the expert B is strong in the state b and less optimal in the state a . The value of the state a obtained by value iteration on the trajectories of A and B will depend on which expert trajectory was presented last. If B was used last, then the result will not be as accurate as if A was used last. Our approach does not suffer from this ordering problem because of the way we determine the values of the states in the auxiliary graph.

3 Empirical Evaluation

To assess the benefits of the proposed approach, we have tested the performance of the ARL in a classical maze-like grid world such as the one in Figure 2. The primary objective of the empirical studies was to measure the acceleration in reinforcement learning with respect to the amount and quality of observed expert trajectories.

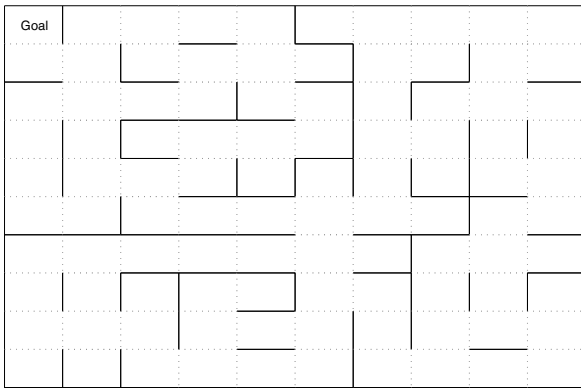


Figure 2: A sample grid-world used for empirical evaluation.

We used a heuristically guided full-width fixed-lookahead search with the user-specified lookahead d as the domain expert and recorded the trajectories it traversed. In the state s , the expert agent with the lookahead d computes the frontier $S(s, d)$ of all states reachable from s in exactly d moves. For each of the state s' on the frontier $S(s, d)$, its heuristic value $h(s')$ is computed. As commonly done, we used the Manhattan distance between the state and the goal state as the heuristic. The agent then takes d moves from state s to the frontier state with the most promising heuristic:

$$\arg \min_{s' \in S(s, d)} h(s').$$

Note that in this example, the distance from s to s' is ignored in the selection as all moves have uniform cost. Upon reaching a dead end, the agent backtracks. The backtracking causes re-visits of states but such loops are removed in Step 1 of the ARL method. As our expert agent remembers its trajectory during each trial, we do not need to adjust the heuristic h as done in real-time search algorithms such as RTA* [Korf, 1990] to avoid infinite cycles.

As we real-time search agents, deeper lookahead leads (on average) to more optimal paths. Thus, by varying d , we were able to adjust the degree of expertise from being a perfect agent (e.g., when the goal state is within the lookahead radius) to the uninformed search (e.g., $d = 0$). In Step 2 of ARL, the heuristic estimates derived from the recorded trajectories were extrapolated onto the entire maze using the Graph Laplacian approach. In Step 3, the initial value function produced was further refined by a temporal difference (TD) algorithm with $\lambda = 0.7$, $\epsilon = 0$, $\gamma = 1.0$ and a learning rate α of 0.1 [Sutton and Barto, 1998, Figure 7.7, p. 174]. The initial state for the TD agents was chosen randomly while the goal state and the maze were fixed.

In order to evaluate the quality of a value function V , we computed the average expected return of a policy greedy with respect to V :

$$V^\pi = \frac{1}{|S|} \sum_{s \in S} E_\pi \left\{ \sum_i r_i \mid s_{\text{start}} = s \right\}.$$

A policy greedy with respect to V always selects the action that is expected to lead to the state with the highest V -

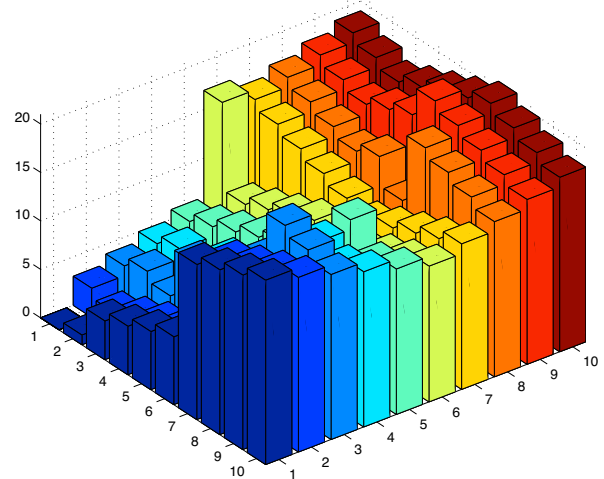


Figure 3: The optimal value function of the maze in Figure 2. The value V^{π^*} of the optimal policy π^* is -12.82 . To make the plot easier to read, we are showing the negative policy value as positive numbers. Hence, visually lower values are better.

value. While it is computationally less expensive to assess the quality of a value function as the mean squared deviation from the optimal value function V^* , [Li *et al.*, 2004b; 2004a] demonstrates that such measure does not lead to maximizing reward collected by the agent. The value of the greedy policy with the zero value function is -27.69 . The value of the optimal policy is -12.83 .

To evaluate the benefits of the apprenticeship-based value function initialization, we compared the policy values from the ARL-initialized policies with TD-learned policy with the initial value function of zero. Initializing the value function with the Manhattan distance immediately yielded a greedy policy whose performance is -18.20 . This initial heuristic is equivalent to the one obtained by approximately 100 episodes of zero-initialized TD-learning as seen in Figure 4.

Figure 6 shows the induced value function from the single trajectory in Figure 5. It is remarkable that the Laplacian operator allows the capture of a rough landscape of the optimal value function from merely a single trajectory: as seen by comparing Figure 3 and Figure 6.

Table 1: **Top:** value of the greedy policy over the value function derived by ARL. A single trajectory of the expert with the shown lookahead was used as the input to ARL. **Bottom:** the number of TD-learning episodes required to learn a value function of the quality produced by ARL from a single expert's trajectory.

Expert's lookahead	1	5	10	30
Policy value V^π	-19.31	-19.28	-16.39	-13.47

Expert's lookahead	1	5	10	30
Equivalent number of TD episodes	95	95	140	300

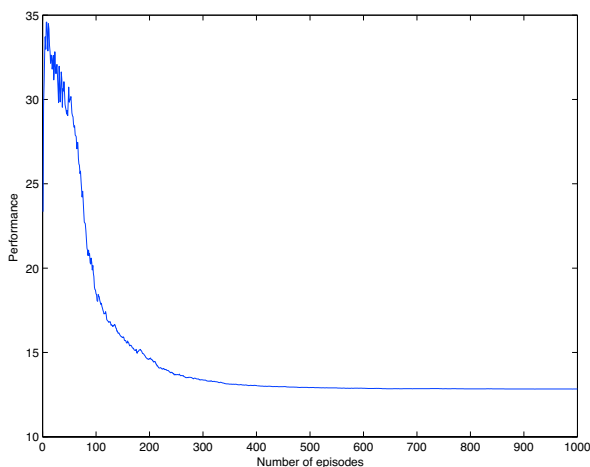


Figure 4: Learning curve of TD($\lambda = 0.7$, $\alpha = 0.1$, $\gamma = 1.0$) initialized with a zero value function. Each point is averaged over 30 independent TD-learning runs starting from random initial states. To make the plot easier to read, we are showing the negative policy value as positive numbers. Hence, visually lower values are better.

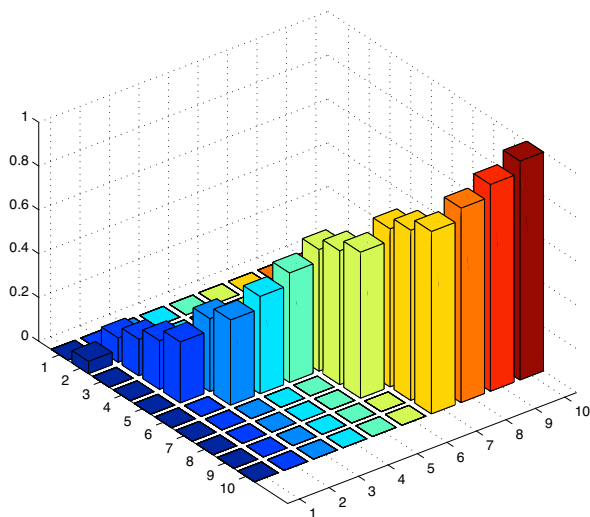


Figure 5: Values of states computed in Step 1 from a single a trajectory of our expert search agent with the lookahead of 5.

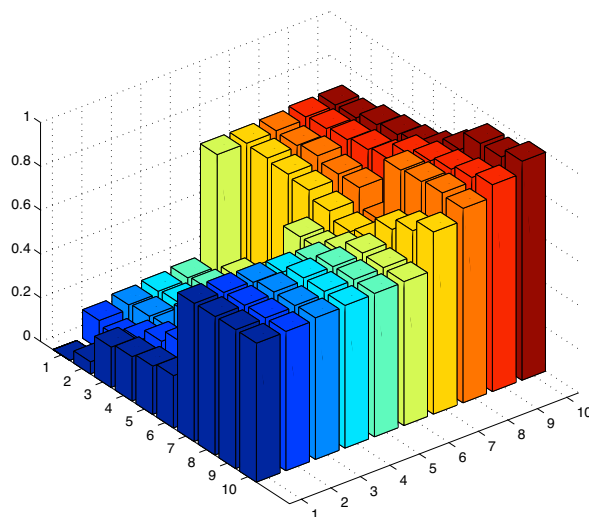


Figure 6: The results of Step 1 (Figure 5) are generalized onto the entire space with the Laplacian in Step 2.

The top portion of Table 1 shows how the quality of the initial value functions produced by ARL from a single expert's trajectory depends on the expert's lookahead depth. The bottom portion demonstrates the number of TD learning episodes required to produce a value function of the same quality as those produced by ARL.

We have experimented with the number of trajectories used as the input to ARL. In this micro-maze, the gains in performance are negligible. We expect that in larger and more complex state spaces leading to sparser sampling with each trajectory, ARL will be able to benefit substantially from a collection of recorded expert problem-solving traces.

4 Future Work and Conclusions

The promising initial results encourage several follow-up research directions. First, we would like to extend the approach to non-episodic tasks. Second, it is of interest to investigate the extent to which this novel method applies to approximated (as opposed to tabular) value function. Third, we are presently working on scaling up this approach to challenging real-world tasks such as the game of Hex wherein human players are presently far superior to computer players and massive amounts of past games are available for apprenticeship learning.

In summary, we have proposed a novel effective approach for generalizing problem-solving traces from expert agents into high-quality initial value functions. Practical evaluation in a standard grid world micro-domain demonstrated that even as few as a single recorded expert trajectory speeds up temporal differences with eligibility traces TD(λ) by as many as 300 learning episodes. That is equivalent to halving the convergence process.

References

- [Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.
- [Aberdeen *et al.*, 2004] Douglas Aberdeen, Sylvie Thiebaut, and Lin Zhang. Decision-theoretic military operations planning. In *Proceedings of International Conference on Automated Planning and Scheduling*, 3–13, Whistler, Canada, 2004.
- [Ackley and Littman, 1991] David H. Ackley and Michael L. Littman. Interactions between learning and evolution. In *Artificial Life II*, volume 10, 487–509. AddisonWesley, Redwood City, CA, Santa Fe Institute Studies in the Sciences of Complexity, 1991.
- [Bonet *et al.*, 1997] Blai Bonet, Gabor Loerincs, and Hector Geffner. A fast and robust action selection mechanism for planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 714–719, 1997. AAAI Press / The MIT Press.
- [Bulitko and Wilkins, 2003] V. Bulitko and D.C. Wilkins. Qualitative simulation of temporal concurrent processes using Time Interval Petri Nets. *Artificial Intelligence*, 144(1-2):95 – 124, 2003.
- [Chung-Graham, 1997] F. Chung-Graham. *Spectral Graph Theory*. Spectral Graph Theory. Number 92 in CBMS Regional Conference Series in Mathematics. AMS, 1997.
- [Holte *et al.*, 1994] R.C. Holte, C. Drummond, M.B. Perez, R.M. Zimmer, and A.J. MacDonald. Searching with abstractions: A unifying framework and new high-performance algorithm. In *Proceedings of the Canadian Artificial Intelligence Conference*, 263–270, 1994.
- [Koenig, 2004] Sven Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS'04)*, 864 – 871, 2004.
- [Kohl and Stone, 2004] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, 611–616, July 2004.
- [Korf and Felner, 2002] R. Korf and A. Felner. Disjoint pattern database heuristics. *Artificial Intelligence*, 134(1-2):9–22, 2002.
- [Korf and Taylor, 1996] R. E. Korf and L. A. Taylor. Finding optimal solutions to the twenty-four puzzle. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1202–1207, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.
- [Korf, 1990] R.E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.
- [Korf, 1997] R. Korf. Finding optimal solutions to Rubik’s cube using pattern databases. In *Proceedings of the Workshop on Computer Games (W31) at IJCAI-97*, 21–26, Nagoya, Japan, 1997.
- [Laud and DeJong, 2003] Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: an analysis of shaping. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. AAAI Press, 2003.
- [Levner and Bulitko, 2004] Ilya Levner and Vadim Bulitko. Machine learning for adaptive image interpretation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI) and Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, 870 – 876, San Jose, California, 2004.
- [Li *et al.*, 2004a] Lihong Li, Vadim Bulitko, and Russell Greiner. Batch reinforcement learning with state importance. In *Proceedings of European Conference on Machine Learning, Poster Section*, 566–568, Pisa, Italy, 2004.
- [Li *et al.*, 2004b] Lihong Li, Vadim Bulitko, and Russell Greiner. Focus of attention in sequential decision making. In *Proceedings of National Conference on Artificial Intelligence (AAAI), Workshop on Learning and Planning in Markov Processes - Advances and Challenges*, San Jose, California, 2004.
- [Ng *et al.*, 1999] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of ICML*, 1999.
- [Sammur *et al.*, 1992] C. Sammur, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proceedings of ICML*, Aberdeen, 1992. Morgan Kaufmann.
- [Schaeffer *et al.*, 2001] Jonathan Schaeffer, Markian Hlynka, and Vili Jussila. Temporal difference learning applied to a high-performance game-playing program. In *Proceedings of IJCAI*, 529–534, 2001.
- [Shimbo and Ishida, 2003] Masashi Shimbo and Toru Ishida. Controlling the learning process of real-time heuristic search. *Artificial Intelligence*, 146(1):1–41, 2003.
- [Smola and Kondor, 2003] A. Smola and R. Kondor. Kernels and regularization on graphs, 2003.
- [Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [Tesauro, 1995] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3), March 1995.
- [Wiewiora, 2003] Eric Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.
- [Zhu *et al.*, 2005] Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty. Semi-supervised kernels via convex optimization with order constraints. In *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.