Quick 'n' Dirty Generalization For Mobile Robot Learning

Content Areas: robotics, reinforcement learning, machine learning, real-time systems
Tracking Number: A895

Abstract

The mobile robot domain challenges policy-iteration reinforcement learning algorithms with difficult problems of structural credit assignment and uncertainty. Structural credit assignment is particularly acute in domains where "real-time" trial length is a limiting factor on the number of learning steps that physical hardware can perform. Noisy sensors and effectors in complex dynamic environments further complicate the learning problem, leading to situations where speed of learning and policy flexibility may be more important than policy optimality. Input generalization addresses these problems but is typically too time consuming for robot domains.

We present two algorithms, YB-learning and YB^-, that perform simple and fast generalization of the input space based on bit-similarity. The algorithms trade off long-term optimality for immediate performance and flexibility. The algorithms were tested in simulation against non-generalized learning across different numbers of discounting steps, and YB was shown to perform better during the earlier stages of learning, particularly in the presence of noise. In trials performed on a sonar-based mobile robot subject to uncertainty of the "real world," YB surpassed the simulation results by a wide margin, strongly supporting the role of such "quick and dirty" generalization strategies in noisy real-time mobile robot domains.

1 Introduction

The mobile robot domain remains a challenge for applying reinforcement learning techniques (Mahadevan & Kaelbling 1996). Algorithms based on policy-iteration are faced with physically-grounded problems of structural credit assignment and noise, both with idiosyncratic characteristics. Structural credit assignment is particularly acute in domains where "real-time" trial length is a limiting factor. Physical tasks by nature have time constraints, and robotic hardware is only capable of a small amount of learning (compared to non-mobile systems) before it requires battery charging, maintenance, alteration, or replacement. Noisy sensors and effectors in complex dynamic environments further complicate the learning problem by reducing the credibility of any given step of a learning trial. This composition of characteristics leads to situations where rapid policy flexibility may be more important than policy optimality. Robots that can quickly learn partially-correct behavior are often more useful and robust than those that take a longer time but eventually reach an optimum.

Input generalization is an effective means of addressing all of the above problems, but is often too time consuming for robot platforms. Without an *a priori* bias, a learning system requires a great deal of time to gather the necessary statistics for significant and reliable generalization. In this paper we present two algorithms, YB and YB^-, that attempt to strike a balance between the costs and benefits of generalization. They generalize over the the input space based on simple and fast computation of input bit-similarity. By exploiting the isomorphism between the input representation and the physical structure that a robot perceives with its sensors, this simple technique can be an effective tool for speeding up the early stages of robot learning, during which the vast majority of the input space has not yet been visited. These algorithms trade off long-term optimality for immediate performance and flexibility.

In simulation, YB-learning and YB^- were compared to non-generalized learning across differing amounts of noise and numbers of discounting steps in simulation. YB was shown to speed up early learning, especially in the presence of sensor noise. Trials performed on a sonar-based mobile robot demonstrated a greater advantage for YB in the real world than was seen in simulation.

2 Quick 'n' Dirty Generalization

YB and YB$^-$ are "plug compatible" with most learning systems. They are incorporated into a traditional reinforcement learning algorithm as follows:

given a policy $P(s,a)$, and a current state s_0,

1. Choose an action a (either $maxP(s_0,a)$ or random with appropriate probabiliy);
2. Perform a;
3. Determine the reward;
4. Update the value of P according to the reward, s_0, and a, as appropriate for the chosen algorithm (Base, YB, or YB$^-$);
5. Return to 1.

2.1 The Base learning algorithm

Our basic learning algorithm combines the *reward discounting* and *reward averaging* techniques that are the most common in reinforcement learning ((Mahadevan 1996) (Ok & Tadepalli 1996)).

The value of performing action a in state s is $\frac{R(s,a)}{N(s,a)}$, where R is the total accumulated reward received for previous performances of action a in state s, and $N(s,a)$ is a normalization factor. R and N are updated as follows:

given reward r, received for performance of action a in state s at timestep t, for all i from 0 to the number of discounting steps:

$$R(s,a)_{t-i} = R(s,a)_{t-i-1} + \gamma^i r_t$$

$$N(s,a)_{t-i} = N(s,a)_{t-i-1} + \gamma^i Base$$

where γ is a discounting factor, Base is a normalizing factor greater than or equal to the largest possible reward for a sigle step, and subscripts indicate the value at a particular timestep.

After performing some action a in state s results in some reward, that reward is averaged with all reward previously received for performing that action in that state. The reward is then spread to the states and actions that immediately preceded the current ones and discounted according to their temporal distance.

R and N must remain separate in order to maintain a true average, since rewards are received (through discounting and, as in YB, similarity) in an "asynchronous" manner and with varying N values that determine a particular reward's influence on the average. This separation also allows accurate integer implementation, which is in keeping with the processing capabilities of most microcontroller-based robots.

2.2 YB-learning

YB generalization is based on bit-similarity of the input states. An unweighted Hamming distance - strictly the number of bits that are the same - is used to calculate a similarity measure, M, between two states:

given Hamming distance H between two states s_1 and s_2, and n inputs,

$$M_{s_1,s_2} = \frac{H}{2^{(H-n)}}$$

The update functions for R and N become:

$$R(s_n,a)_{t-i} = R(s_n, a_{t-i-1} + M_{(s_n,s_{t-i})}\gamma^i r_t$$

$$N(s_n,a)_{t-i} = N(s_n, a_{t-i-1} + M_{(s_n,s_{t-i})}\gamma^i Base$$

Thus, reward is distributed exactly as in the basic algorithm. Any time reward is distributed, however, either to the current state and action or to earlier states and actions through discounting, every state receives a fraction of the reward proportional to its similarity to the state directly receiving reward.

2.3 YB$^-$-learning

We considered a problem which might result from YB generalization as a robot approaches optimal performance. If the reward function is set up in such a way that performing the proper action always results in a positive reward, the spread of reward to similar states will eventually cause some "unlearning" of negative, "prohibitory" knowledge if rewards are normalized. To see if this effect truly had an influence on our learning task, we tested a hybrid algorithm called YB$^-$. In YB$^-$, positive rewards are handled exactly as in the basic algorithm, and negative rewards are handled as in YB. Thus, although the robot may begin to receive a predominance of positive reward, previous knowledge of what actions to avoid will not be slowly wiped out; yet actions with negative results will be rapidly generalized across the state space. While our reward function does dispense negative reward for some proper actions (turns in the right situation), negative reward should become relatively rare as behavior improves.

3 Experimental Domains

Experiments were carried out in either a real or simulated eleven by fourteen foot "corrall" with obstacles projecting from the edges which provided various corners and dead ends (Figure 1). The simulated robot was based on a simple model of our physical robot. The robot's learning task in both simulated and real-world domains was to expeditiously extricate itself from difficult situations such as dead ends so as to travel the

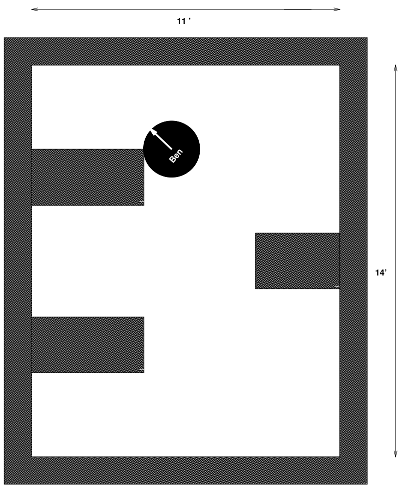

Figure 1: The experimental environment used for both simulation and physical robot testing. The physical environment was constructed with a smooth-walled "corrall" and weighted cardboard boxes. The black circle represents the robot.

greatest possible total distance over the course of the trial. The reward function was simple:

$$R = \begin{cases} 512 & \text{if robot moved at least 100mm} \\ -512 & \text{otherwise} \end{cases}$$

and discount factor γ was 2/3.

Of the seven bits of input, five were derived from the directional sensors and the remaining two represented the direction (left, right, or forward) of the last action taken. The robot had seven directional sonar sensors, angled at 90, 20, 10, 0, -10, -20, and -90 degrees relative to the robot's heading; the 10- and 20-degree sensors on each side were OR-ed together to yield a total of five bits of sensor information. These sensors were thresholded to binary values: 1 if they detected an object within 500mm, 0 if they did not. The five possible actions were: *go-forward*, *turn-left 45 degrees*, *turn-left 90 degrees*, *turn-right 45 degrees*, and *turn-right 90 degrees*.

3.1 Simulation

Our Pioneer simulator is implemented in Macintosh Common Lisp, with fairly simple sonar and collision detection. Two types of uncertainty were tested in simulation, at levels varying from 0% to 15%. One was sensor noise: bits of the input were changed with a probability equal to the noise percentage; the other was actuator uncertainty: the heading of the robot deviated from the commanded value by the noise percentage. All noise levels were also tested at three levels of discounting. Preliminary research indicated that performance peaked with a discount level of three to five steps; this was borne out in the full trials.

For each of the twenty-four combinations of noise type (sensor only vs. sensor and actuator noise), noise level (0%, 5%, 10%, or 15%), and discounting steps (3, 4, or 5), 20 trials of 5,000 steps were run in simulation.

3.2 Mobile Robot

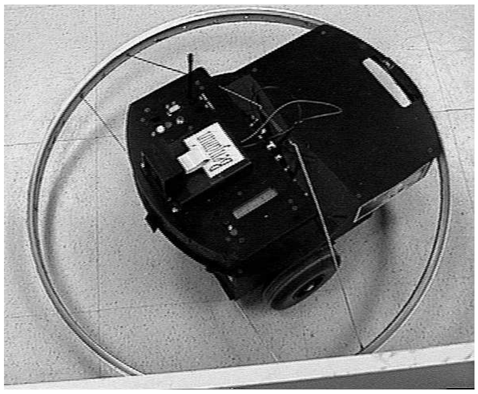

Figure 2: Ben, the mobile robot used in the learning experiments: an RWI Pioneer with two wheels, 7 sonars, and a mechanically extended body radius to allow free rotation even while in contact with objects.

The physical testbed for our generalization ideas is a Pioneer I robot from Real World Interface, programmed with IS Robotics MARS/L (Multiple Agency Reactivity System), an on-board Common Lisp system extended to support multiple concurrent processes and behaviors. This system allowed us to use the same Lisp code in the simulator and on the robot. The Pioneer is a rectangular base steered with two wheels and a caster. Its sensors include seven sonars, as described above (Figure 2). The robot is also equipped with a Fast Track Vision System not used in the current experiment. Much of the robot's body is devoid of sensors, and as it is not rotationally symmetric, it is possible for the robot to get into situations where complicated sequences of maneuvers are necessary for it to detect an obstacle and continue moving. Exiting these situations is quite difficult, even for hand-coding (akin to parallel-parking a truck with no rear view); to avoid the problem, we added a circular ring that made the robot rotationally symmetric so that it could rotate freely in any situation.

The only differences between the code for the simulator and the real robot allowed the robot to run somewhat continuously rather than in discrete steps and compensated for sonar uncertainty. Timing issues involved in programming the robot to move in discrete steps caused each step to take an inordinate amount of time, and timing uncertainties made analysis of learning results difficult. Instead of using discrete actions and pausing between them, the robot performed each action for a fixed amount of time (3 seconds) and computed the reinforcement just prior to switching to the next action, without stopping in between. This scheme allowed us to guarantee that timing issues did not bias learning results, while speeding up the trials somewhat.

The compensation for sonar uncertainty alleviates to a small extent one of the difficulties of sonar perception. Ultrasound sensors are infamous for their specularity properties; as the sonar beam impacts a surface at an oblique angle, it bounces off and returns a falsely long reading (Drumheller 1987, Kuc & Di 1986). Our strategy consisted of having the robot accumulate readings while vacillating between +15 and -15 degrees from the current heading. This provided repeated readings from slightly different angles which increased accuracy of obstacle detection. Enough time was given for the robot to "wiggle" a bit, while executing the *forward* action and immediately following *turn-left* and *turn-right* actions, before the a new state was sensed.

4 Experimental Results

4.1 Simulation Results

Figure 3 summarizes the results of the experiments run in our simulator. It presents data from trials which varied by noise level and algorithm, but not by level of discounting. Our results showed a performance peak somewhere between three and four steps of discounting; we thus chose three as the number of discounting steps for our physical robot trials and present only this data from the simulator for clarity and comparison.

In Figure 3, the relative performance of YB learning and the Base algorithm are compared. The performance measure is the cumulative distance traveled by the robot at each timestep. This measure is far smoother than the instantaneous learning rate, and allows us to easily determine important points in the trade-off between fast early learning and long-range optimality. As can be seen, YB learning has some initial advantage at all noise levels, and this advantage is extended as the noise level increases. In all cases where there was at least 5% noise, YB-learning outperformed the Base % algorithm for at least the first 1400 time steps; in our tests with only sensor noise, YB remained superior to the Base algorithm for all 5,000 steps of our trials at noise levels of 10% and 15%. Figure 4 demonstrates the important features typical of our results, in absolute terms. YB has an initial advantage which is gradually reduced; in this particular case the Base algorithm has achieved superiority by step 1900. YB^- always performed worse than both YB and the Base algorithm by a very large margin.

We examined statistical significance of our results through a one-way ANOVA at each point on the graph (that is, at intervals of 100 timesteps). Under the sensor-noise-only condition, with 0% noise, YB was significantly

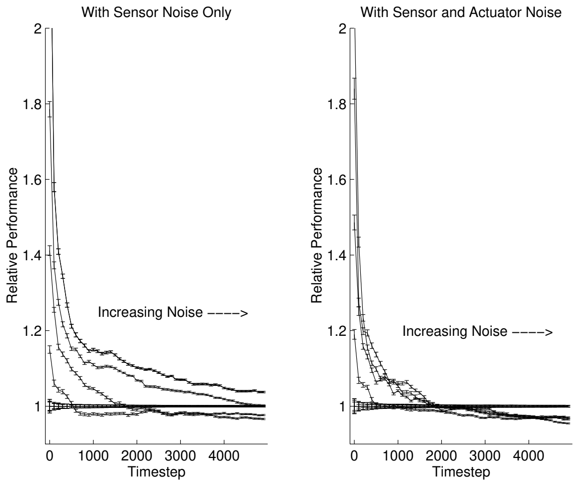

Figure 3: Summary of Results from Simulator Trials. The graphs show performance relative to the Base algorithm, which appears as a straight line at performance level 1. The other curves, from left to right, represent relative performance of YB at 0% noise, 5% noise, 10% noise, and 15% noise. The number of discount steps for these curves was 3; error bars are one standard deviation above and below the curve.

($p < 0.05$) better than Base only during the first one hundred timesteps, and significantly worse than Base after the 4900th timestep; with 5% noise, YB was significantly better until step 800 and worse after 4000; at 10% noise YB was better until step 3800 and never worse during our trials, and at 15% noise YB was significantly better through all 5000 steps. Under the sensor and actuator noise condition, at 0% noise YB was significantly better until step 100 and worse after step 4100; at 5% noise it was better until step 900 and worse after step 3600; at 10%, better until 1300 and worse after 3800; and at 15%, better until 800 and worse at 4500 and 5000.

The statistical data confirm that while increasing *sensor* noise increases the performance of YB relative to Base, the addition of *actuator* uncertainty reduces this lead. Figure 5 illustrates the difference in absolute effect between sensor noise alone and the combination of sensor noise and actuator uncertainty. The large gap visible only in the top graph between the 0% noise condition and the noisy conditions along with the differential benefits of YB indicate intricacies in the difference between sensor and actuator uncertainty that we are beginning to investigate.

4.2 Mobile Robot Results

Trials on the physical robot were run for 1500 steps with three discounting steps. The results represent 11 trials each of the YB and Base algorithms. A preliminary experiment indicated that trials of YB^- would not be

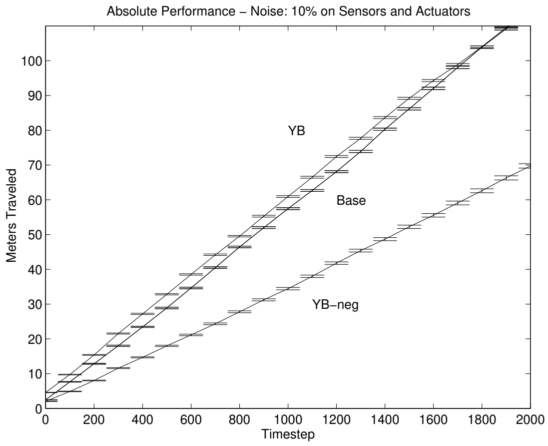

Figure 4: A Typical Simulation Result in Absolute Terms. YB has an initial advantage over Base that lasts until around the 1700th step. YB^- performs worse by a substantial margin. The difference between YB and Base is significant ($p < 0.05$) in this graph until timestep 1300.

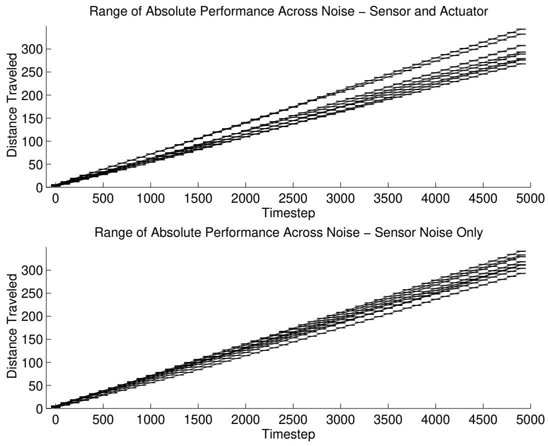

Figure 5: The Effect of Noise on Absolute Performance. The highest lines are noiseless conditions, the lowest are noise at 15%. Note that the condition with only sensor noise (bottom) has a smooth degradation, while the condition with both sensor and actuator uncertainty (top) has a discontinuity between noiseless and noisy trials.

fruitful.

As indicated in Figure 6, the physical-robot results follow the pattern of the simulator results, but the magnitude of YB's benefit over Base is much greater than even that of YB's best performance in simulation. None of our trials reached a point at which Base learning performed better than YB. Figure 4.2 indicates a continuing divergence of YB and Base learning, rather than the gradual surpassing of YB by the Base algorithm seen in simulation (Figure 4). A one-way ANOVA shows the results to be highly significant, with $p < .002$ for all 1500 steps. The trend seen in simulation of YB's greater robustness to noise than non-generalized learning was not only repeated, but surpassed in our real-world experiments. Over the 1500-step course of our robot trials, the performance of YB was an average of 127% of the Base algorithm.

5 Discussion

Our results can perhaps best be contextualized by the fact that each robot trial of 1500 steps lasted close to two hours. Though the simulator results from trials of 5,000 steps show that the Base algorithm eventually reaches a higher level of performance than YB, real-world time constraints lead to better performance of YB on a physical robot. For most robotic tasks, the length of a step has a lower bound imposed by the physics of motion and responsiveness of the environment. The only way to speed up learning beyond this point is through some type of generalization. Our results show that even a very simple generalization technique can have a strong effect on performance of a physical robot.

YB showed its greatest advantage in noisy situations. Its performance relative to the non-generalized Base-case increased with uncertainty of sensing and action. This was demonstrated through adding various levels of noise to the simulation, but was seen most clearly in the experiments of the physical robot operating with real-world uncertainty. The advantage of YB over Base on the real robot was consistently about twice the best advantage seen in simulation throughout the 1500 timesteps trial length.

Definitive attribution YB's advantage to a faster learning rate, greater policy flexibility, and/or some other factor of generalization that filters out noise is a subject for further research. The discontinuity between sensor-only and sensor-and-actuator uncertainty is perhaps a good place to begin examination of some of the noise issues. Future experiments in a more dynamic environment will cast light on issues of policy flexibility. We would like to examine the issue of recognizing the point of changing superiority of learning algorithms in order to switch from YB to Base to reap the advantages of both at different stages of learning.

6 Related Work

Chapman & Kaelbling (1991) addressed input generalization in a video game domain, where a statistical generalization algorithm started with a single most general state and iteratively split it. Mahadevan & Connell (1991) compared two generalization techniques on a mobile robot performing a box pushing behavior; one clustered states starting from a fully differentiated state space based on statistics accumulated over time, and the other spread reward according to a weighted Ham-

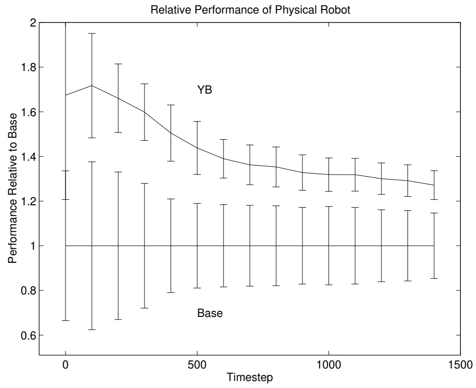

Figure 6: Relative Performance of YB and Base on a Physical Robot. YB's advantage is much greater than the best case in simulation.

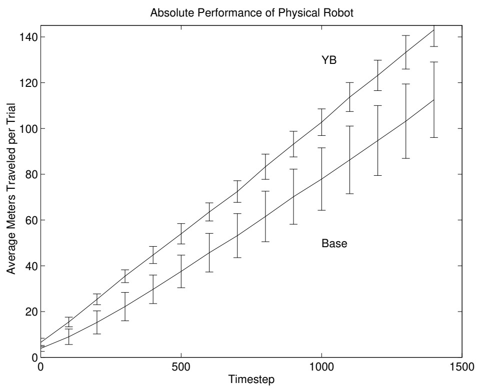

Figure 7: Absolute Performance of YB and Base on a Physical Robot. Note that the curves continue to diverge, rather than converge as in Figure 4. The difference between YB and base is significant ($p < 0.002$) at all timesteps.

ming distance in a manner similar to YB. The reward thus distributed was not scaled by similarity but given in full only to states above a predetermined threshold. Setting the weight of individual bits and the threshold required application of knowledge of the structure of the input space which is not necessary for YB. No comparison was made to non-generalized learning. Neural networks are a popular alternative for automated input generalization; they were used in several mobile robot navigation-learning demonstrations, including Lin (1993) and Mitchell & Thrun (1993). Mahadevan (1996) summarizes a great deal of earlier work on generalization; it and Ok & Tadepalli (1996) examine the issues of discount- and average-based reinforcement learning that led to our Base algorithm.

7 Conclusion

When learning is subject to real-world time constraints and uncertainty, algorithms that trade off long-term optimality for faster learning of suboptimal policies can exhibit better performance. If task dynamics constrain the time of an individual learning step, the only way to speed up learning is through some type of generalization. In early learning stages during simulation, the YB algorithm for very simple, fast generalization showed a significant performance increase over non-generalized learning in situations with noise. The performance of YB relative to non-generalized learning increased as noise increased. In real-world trials on a physical robot, YB showed a relative performance stronger than its best performance in simulated trials, 27% better on average than non-generalized learning during trials of the length supported by our robot hardware.

These robot learning results offer compelling evidence of the benefits of "quick and dirty" generalization in situations with real-world time constraints and uncertainty, and demonstrate some of the factors that influence the concept of "optimality" when selecting algorithms for use in such robotic domains.

References

Chapman, D. & Kaelbling, L. P. (1991), Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons, *in* 'Proceedings, IJCAI-91', Sydney, Australia.

Drumheller, M. (1987), 'Mobile Robot Localization Using Sonar', *IEEE Transactions on PAMI*.

Kuc, R. & Di, Y. (1986), Intelligent Sensor Approach to Differentiating Sonar Reflections From Corners and Planes, *in* 'International Congress on Intelligent Autonomous Systems', Amsterdam.

Lin, L.-J. (1993), Scaling Up Reinforcement Learning for Robot Control, *in* 'Proceedings, Tenth International Conference on Machine Learning', Morgan Kaufmann.

Mahadevan, S. (1996), Average-Reward Reinforcement Learning: Foundations, Algorithms and Empirical Results, *Machine Learning*, bf 22, 159–196.

Mahadevan, S. & Connell, J. (1991), Automatic Programming of Behavior-based Robots using Reinforcement Learning, *in* 'Proceedings, AAAI-91', Pittsburgh, PA, pp. 8–14.

Mahadevan, S. & Kaelbling, L. P. (1996), 'The National Science Foundation Workshop on Reinforcement Learning', *AI Magazine* **17**(4), 89–97.

Mitchell, T. M. & Thrun, S. B. (1993), Explanation Based learning: A Comparison of Symbolic and Neural Network Approaches, *in* 'Tenth International Workshop on Machine Learning', Morgan Kaufmann, pp. 197–204.

Ok, D. & Tadepalli, P. (1996), Auto-exploratory Average Reward Reinforcement Learning, *in* 'Proceedings, 14th International Conference on Artificial Intelligence (AAAI–92)', AAAI Press/MIT Press, Portland, Oregon, pp. 881–88.