Generality Vs. Speed Of Convergence In the Cart-Pole Balancer

Jerome B. Soller
C.S. Department

Neil E. Cotter
E.E. Department

UUCS-91-008

Department of Computer Science
University of Utah
Salt Lake City, UT 84112, USA

April

Abstract

This paper compares the speed of convergence to an optimal solution of four controllers for the problem of balancing a pole on a cart. We demonstrate that controllers whose design is tailored specifically to the cart-pole problem (i.e. less general) converge more rapidly to an optimal solution. However, the architectures and learning algorithms for those networks may not perform well for more general tasks. The four controllers, ordered from the least general to the most general, are the Perceptron, the Associative Search Element [1], Jordan's approach [3], and Prejudicial Search applied to the ASE architecture. Two of the above neural networks, the Perceptron and Prejudicial Search [2] are new methods for solving this problem. The perceptron is a simple two input neuron (summing unit) with 2 weights and a step function output. The Prejudicial Search is a method for biasing the search of possible solutions. It guarantees convergence, but allows the search to be biased by heuristics or information about the problem. In this paper, it is combined with the ASE architecture. However, the Prejudicial Search technique can be combined with any architecture and learning algorithm, extending their ability to handle a more general class of problems.

1 Introduction

The pole-balancer problem has been used for many years as the classic test of problems where the only feedback is failure, Reinforcement Learning Problems. The basic idea is a pole with one degree of freedom is attached to a moving cart. The position, velocity, angle, and angular velocity are calculated via simulation and fed into the controller. The controller's task is to apply a force to the cart in a way that does not cause it to cross θ or x boundaries. Four approaches will be applied to this problem ranging from very general to very specific.

The specific approaches are faster, but can not be applied to as many problems. Many of the approaches in the literature stress the speed of convergence, but do not discuss the loss of generality. Different architectures have different cost surfaces, surfaces describing the performance of the system based on a criteria, its weights, and its inputs. Therefore, some learning schemes may work well on certain architectures and not others. By a clever choice of an architecture, a network can be customized to a problem. However, the loss of generality must be recognized.

The least general approach is the 2 input perceptron with a step function output. This approach has the fastest convergence for this problem because a bang/bang controller with 2 inputs can solve this problem. Creating a forward model allows systems to generalize to different situations. The forward model biases the search on a neural network control system. Both Barto and Sutton [1] and Jordan [3] use some kind of forward modelling. Barto and Sutton's Reinforcement learning scheme quantized the space of inputs and limited itself to bang/bang control. However, if a system could be quantized in this fashion, it could be mapped to their system. Jordan's approach is more general in that it allows continous values and does not quantize the space. However, gradient descent/backpropagation techniques, such as those used by Jordan, rely on assumptions about the cost surface. If the surface is not differentiable or has many local minima, gradient descent techniques, such as Jordan's do not work well. In this paper, we present an even more general approach, using prejudicial search. Like simulated annealing, prejudicial search guarantees convergence as time approaches ∞. Unlike simulated annealing, it allows the system to use information about the solution space to speed its search. This technique is applied to this problem, but can be applied to more general problems. It can also be combined with other techniques used for neural network control to avoid local minima problems.

2 The Problem And Current Approaches

2.1 The Cart-Pole Balancer

The system to be controlled consists of a broomstick with one degree of freedom attached to a cart which is attached to a track of finite length [1] [5]. There are four state variables the system can read: θ, $\dot{\theta}$, x(absolute position), and $\dot{x}$. These values come from sensors or differential equation simulation. The constraints the optimal controller must satisfy are:

1. The θ limits are $+12°$ and $-12°$
2. The x limits are +2.4m and -2.4m

The cart starts at the origin with $\theta = 0$ and all velocities = 0. The pole starts to fall, and the system responds by applying a force to the cart. This continues indefinitely until the system fails. Failure occurs when the cart-pole system passes one of the θ or x thresholds. Systems that don't fail find limit cycles where they oscillate around the origin. To keep consistent with Barto and Sutton's experiments [1], the following assumptions are made in the simulation:

1. The phase space is divided into the following 162 regions.
   (a) θ boundaries are at $0°$, $+1°$, $-1°$, $+6°$, $-6°$, $+12°$, and $-12°$.
   (b) $\dot{\theta}$ boundaries are at $+50°$/sec, $-50°$/sec, $+\infty°$/sec, and $-\infty°$/sec.
   (c) X boundaries are at +0.8m, -0.8m, +2.4m, and -2.4m .
   (d) $\dot{X}$ boundaries are at +0.5m/s, -0.5m/s, $+\infty$ m/s, $-\infty$ m/s
2. Using bang/bang control with -10 or 10 Newtons of applied force
3. Simulation of pole balancer dynamics using Euler Integration with time steps of 0.02 seconds.
4. The θ limits are $+12°$ and $-12°$
5. The x limits are +2.4m and -2.4m
6. Learning Constants: $\alpha = 1000$, $\delta = 0.9$
7. The Noise for the ASE below has standard deviation = 0.01 .

The result of these assumptions is a reduced search space using a priori information about the problem.

2.2 The Perceptron Solution

The simplest neural network model is the perceptron. It consists of a neuron summation unit with weighted inputs. In other words, y(t) = $\sum_{i=1}^{n} x_i(t) \cdot w_i(t)$. If y(t) > 0, the output is a 1. Otherwise, it is a 0. Figure 1 shows the 2 input perceptron applying either 10 or -10 Newtons of force to the cart. The random search box holds the optimal solutions or solutions. Each perceptron serves to divide the solution space into two halves by slicing it with a hyperplane. The perceptron is equivalent to a bang/bang controller, where half of the space will produce one output. In this particular case, and not in general, the x and $\dot{x}$ terms are not needed because most of the optimal controllers cause the system to enter limit cycles around the center. The first simulation peformed was an analysis of the solution space for the 2 weight perceptron. Since there was a high percentage of solutions, it was not practical to use a conventional learning method. Instead, uniform random search was used to characterize the space of solutions. All of the solutions formed a wedge. Figure 2 contains the quadrant with the wedge. Each point on the wedge represents the slope of a hyperplane mapping the input space into a 10 or -10 force applied to the cart. All of the solutions were limited to the quadrant with negative values for both of the gains. Overall, 12 percent of possible weight values worked. If we used a logistic instead of a step function, the weights for an optimal controller would saturate the neurons to produce a bang/bang controller effect. Obviously, the percentage of weights giving viable solutions with a logistic is reduced.

2.3 Barto and Sutton Reinforcement Learning

Barto and Sutton [1] solve the pole-balancer problem with their Reinforcement Learning Model. Figure 3 contains the Barto and Sutton controller. It had two major components, an Associative Search Element(ASE) and an Adaptive Critic Element(ACE). The ASE is the unit that actually controls the force applied to the cart. The 4 element phase space vector is mapped into one of the 162 boxes above. That mapping produces a Z vector, a string of all 0s and one 1 corresponding to each of the 162 possible boxes. Each box i associated with the ASE has two pieces of information attached at time t, the weight, $W_i(t)$ and the eligibility, The weight updates depend heavily on the eligibility, a time trace. The eligibility associated with a given W_i is incremented by a fixed quantity when its box is activated, x_i = 1. Over time, this value decays. If a box is visited frequently, the system will add to the current eligibility. This approach gives the system information that is used to choose which weights are more likely to have caused a failure. The reinforcement at time t is only -1 upon failure and 0 elsewhere. Therefore, the weights of the ASE without the attached ACE are only updated on a failure. Since noise is involved and the system applies binary forces, the weights determine probabilities(certainty factors) that the force will be applied in one direction. However, by using high valued constants, they minimize the effect of the noise.

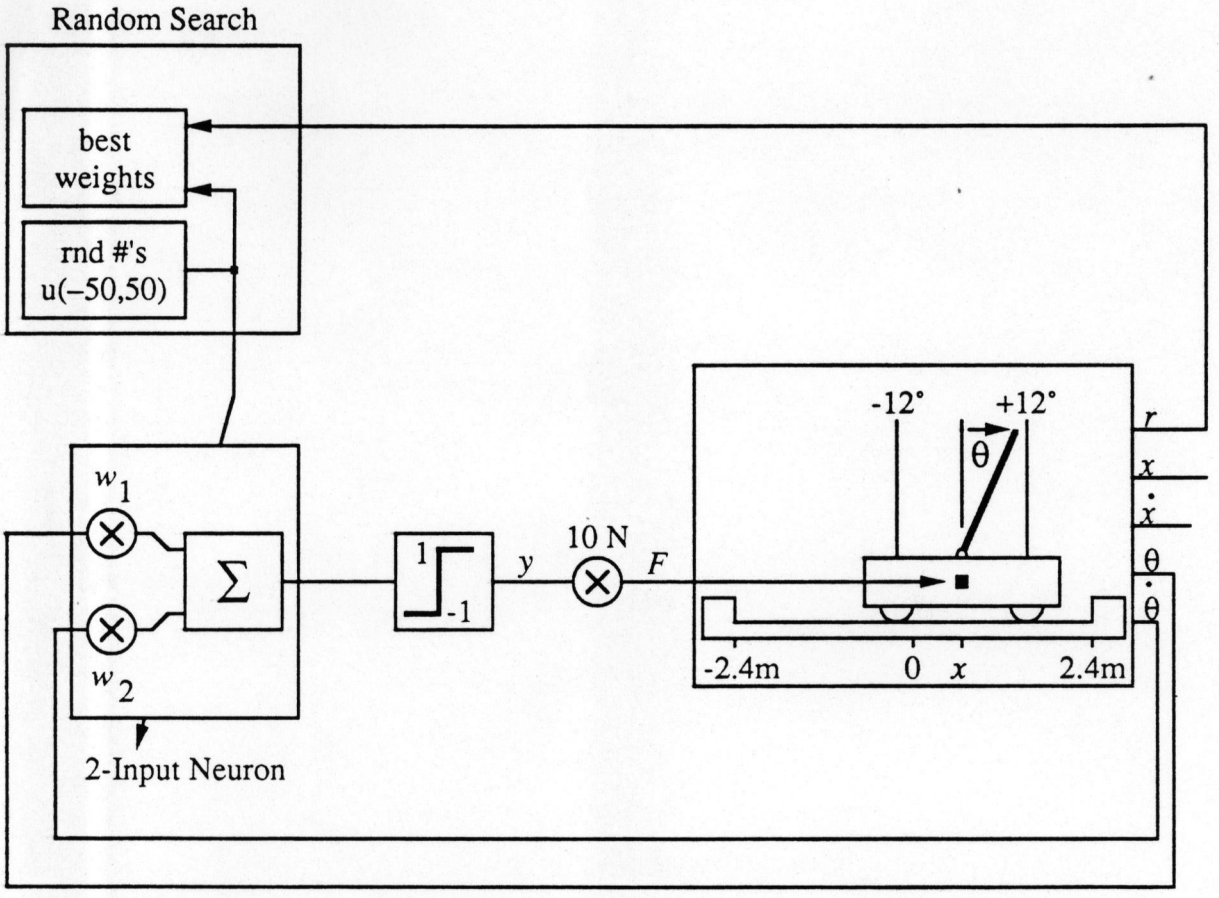

Fig. 1. 2-Input perceptron pole-balancer.

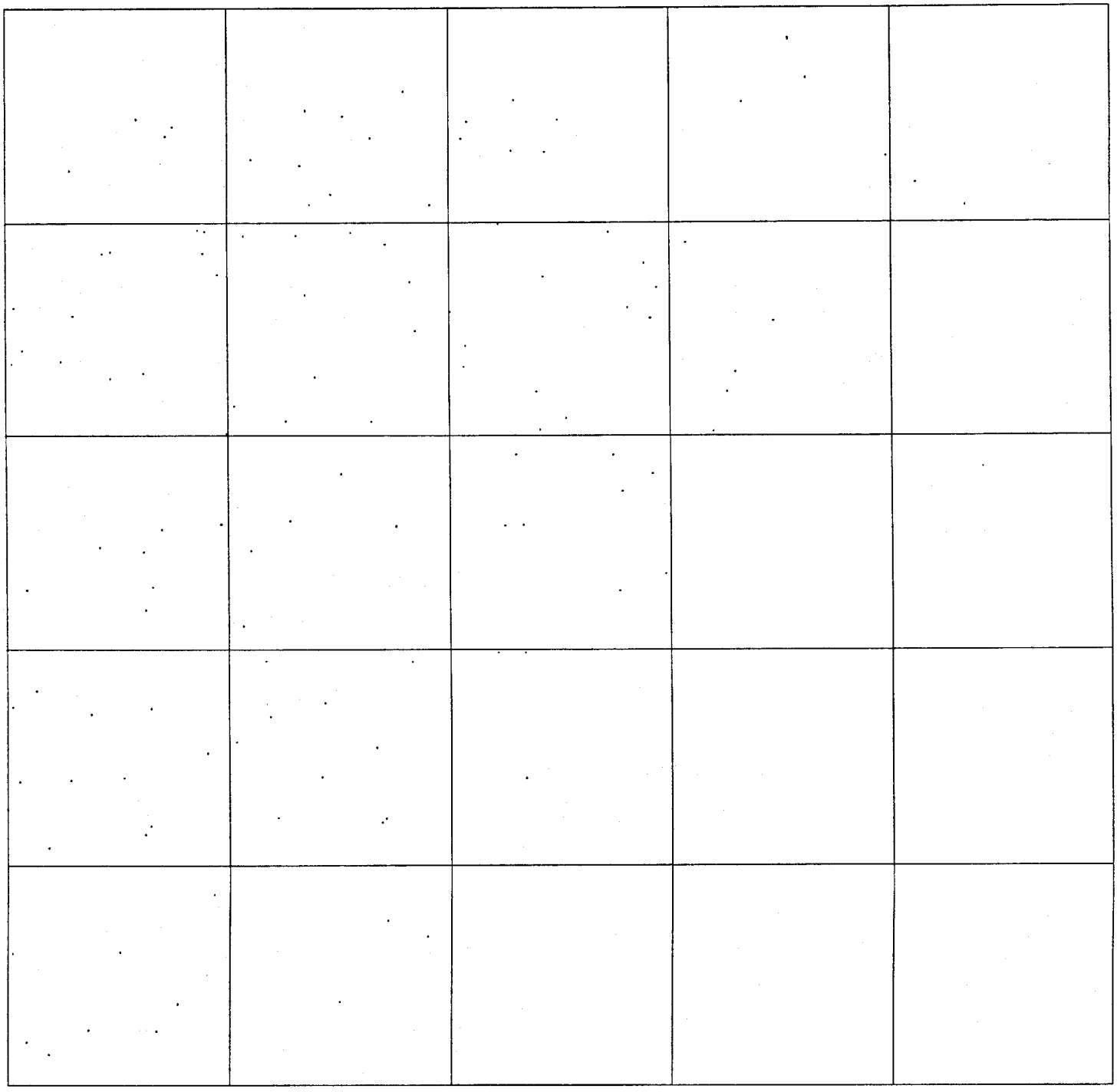

FIGURE 2

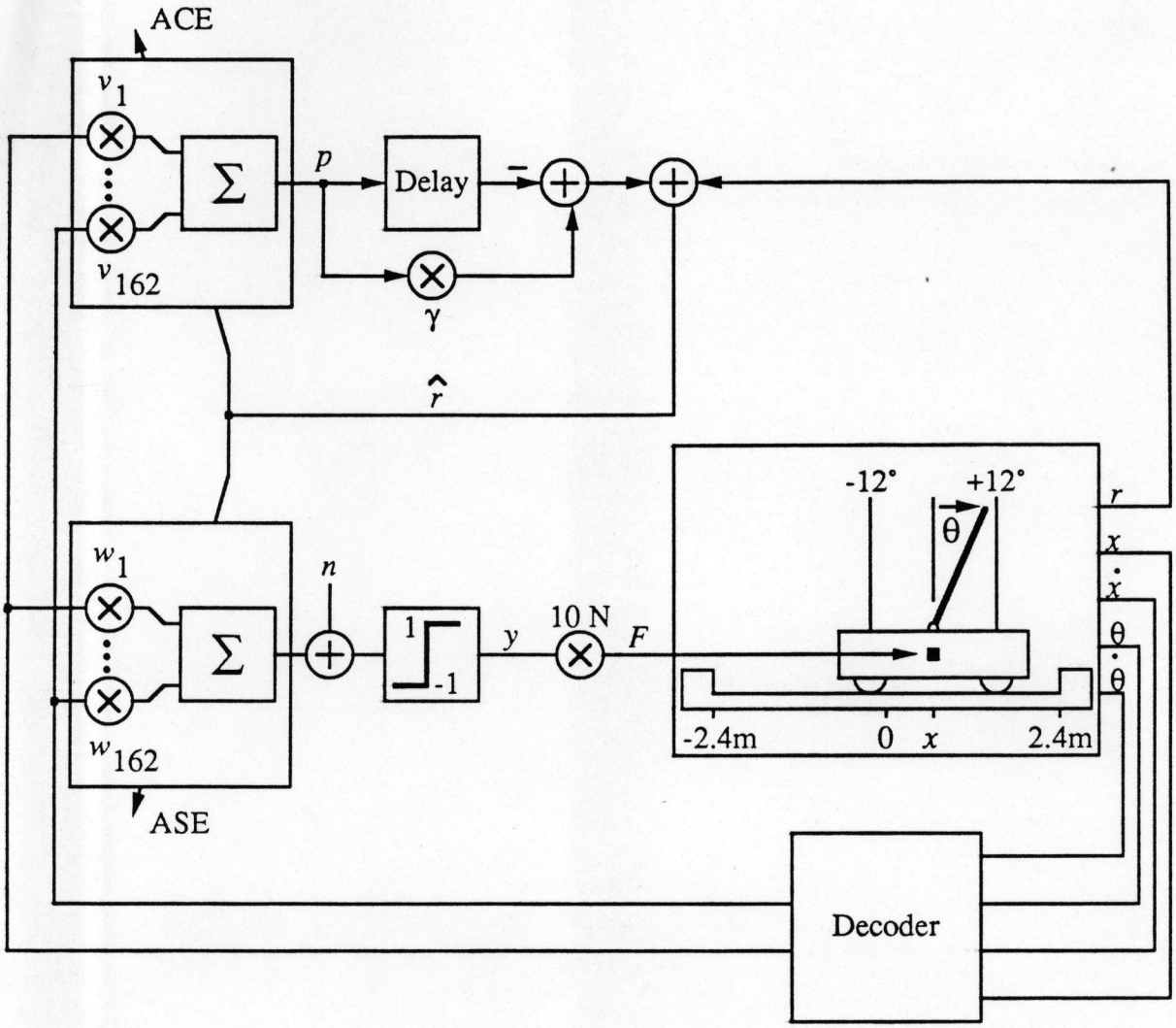

Fig. 3. Adaptive Critic Element pole-balancer of Barto, Sutton, and Anderson.

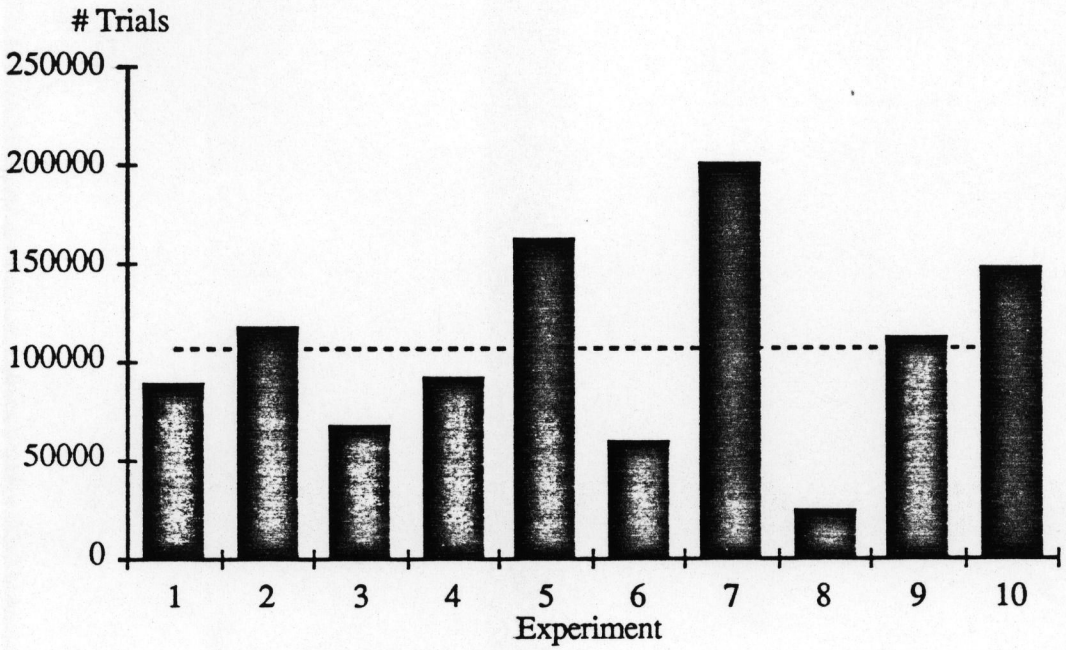

Fig. 4. ASE number of trials to find successful weights. Dashed line is average.

To bias the search of the ASE, the reinforcement approach uses an Adaptic Critic Element. This element's purpose is to give the ASE reinforcement at each time step and not just after a failure. Its own weights are updated based on the correctness of its prediction in a deterministic fashion when a failure occurs. It also uses its own form of time trace, which strengthens weights frequently used. This leads to the encouragement of limit cycles. If a box is reached repeatedly in the phase space in 1 trial, the system usually has entered a limit cycle. The ACE method is closely related to the temporal differences methods [4]. With the ACE, the optimal solution to this problem is found within 3000 failure trials.

Figure 4 shows the number of trials until success for 10 simulations of the ASE. Later, the performance of the Prejudicial Search will be compared with the ASE using the same architecture.

2.4 Jordan's Solution

Jordan [3] also uses a forward model to bias the search of the controlling neural network. His architecture is more general because it allows a random starting position and continous control. The training for his forward model and his controller both depend on backward error propagation. The actual controller, shown in Figure 5, consists of one summation neuron with an output logistic that is mapped to the applied force. It has four inputs, x, $\dot{x}$, θ, and $\dot{\theta}$. The training of those four weights will determine the control scheme. The action and the outputs of the forward model are fed into the hidden layer. There is complete interconnection between the first and second layer. The forward model learns to use the following state information with the action unit's output:

1. $|x|$
2. sgn(x)
3. $|\dot{x}|$
4. sgn($\dot{x}$)
5. $|\theta|$
6. sgn(θ)
7. $|\dot{\theta}|$
8. sgn($\dot{\theta}$)

Each one of the above state values is connected to every neuron in the hidden layer. Likewise, the action unit is also connected to every neuron in the hidden layer. The hidden layer neurons' outputs are connected to a temporal difference unit, which evaluates the correctness

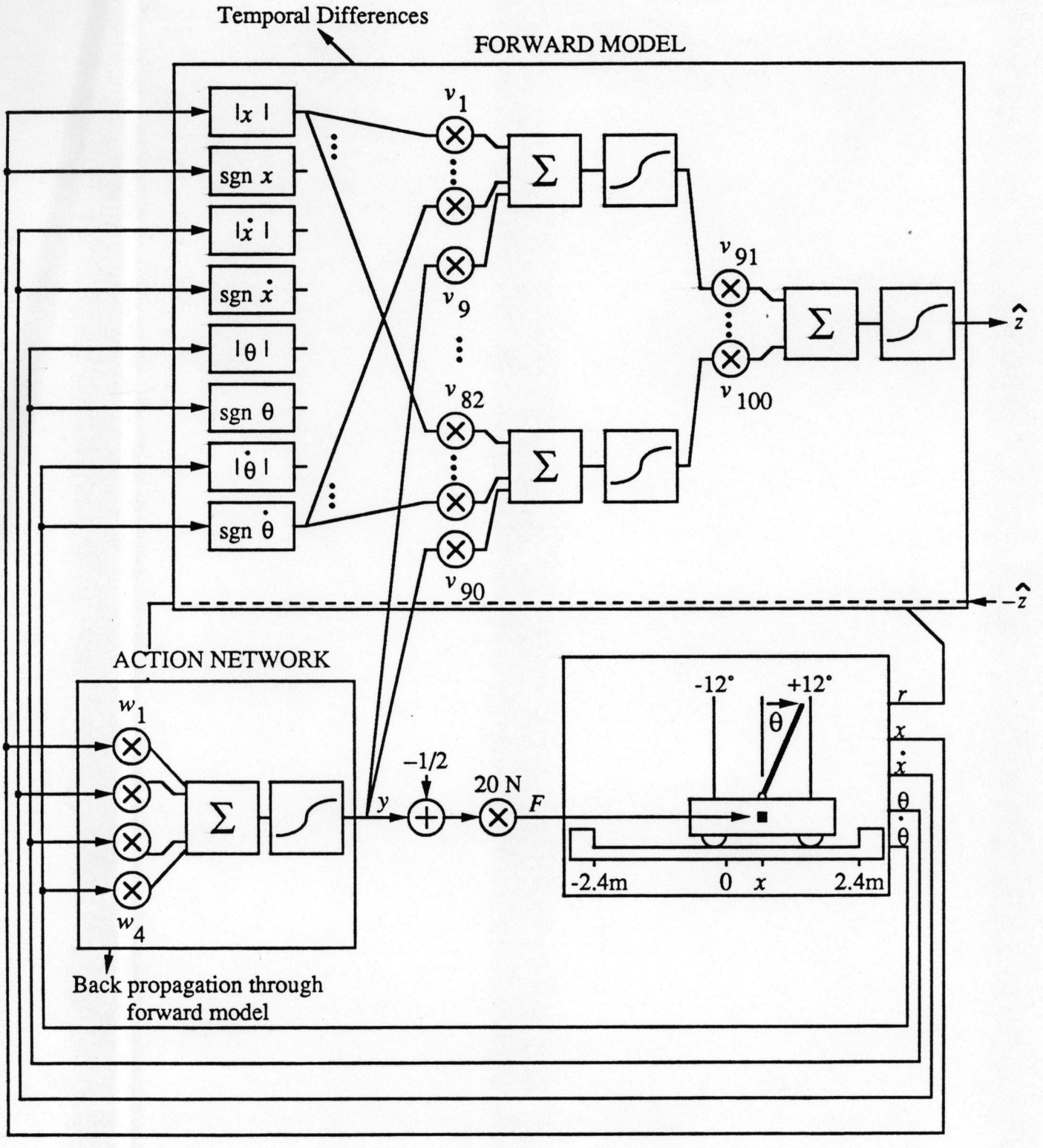

Fig. 5. Temporal differences Pole-Balancer of Jordan and Jacobs.

of the prediction made by the forward unit and learns by backward error propagation. The forward model is trained at a faster rate to adapt to the changes in his force controller, which it is modelling.

Although Jordan's model [3] is more general than Barto and Sutton's [1], it does use some knowledge of the problem. He takes advantage of the symmetry of the broomstick balancer problem by extracting the sign and the direction of the four state variables. By presenting the start positions one at a time, he added noise and reduced the chances of local minima problems. However, minima problems were not completely eliminated. 18 out of 20 of Jordan's runs converged to a solution. The remaining two ended in local minima. If prejudicial search had been applied, the minima problems would have been eliminated.

3 The Prejudicial Search

There are many different methods for searching a solution space. The more general methods take longer, but the less general one will not find the best or even good solutions for certain classes of problems. Random search is the most general because it uses no information about the problem. This proves to be a disadvantage because it fails to focus in regions where good solutions have been found in the past. Simulated annealing is another search that guarantees to find the global minimum of the error surface(optimal solution for the solution space) as time approaches ∞. However, it is difficult to map control systems into an appropriate form and it takes a long time to find good solutions. Prejudicial search [2] also guarantees those convergence properties, but usually at a faster rate. It biases its search towards areas of interest, but it can move to other regions.

If we have some information about where the minimum of a function is located, we can use that information to choose weight vectors. For the kth iteration of learning in the system, we can guarantee success by insuring that every point in the domain is selected with a finite probability density greater than or equal to $1/(1 + k)$. We need to bound our solution space. For instance, if we are at the 49th iteration, there is a probability at least 0.2 for choosing a uniformly random weight vector in the solution space. The rest of the 0.98 probability can be distributed based on a biased search technique. When a uniform weight choice is made, only better solutions are accepted. However, there are no restrictions on the biased search technique; it can based on a priori knowledge of the system or information gathered from the system's failure, such as eligibility traces. Although we apply this technique to control systems, it can be applied to any optimization problem. A formal rigorous proof of the prejudicial search's convergence properties can be found in "Prejudicial Searches and the Pole Balancer" [2].

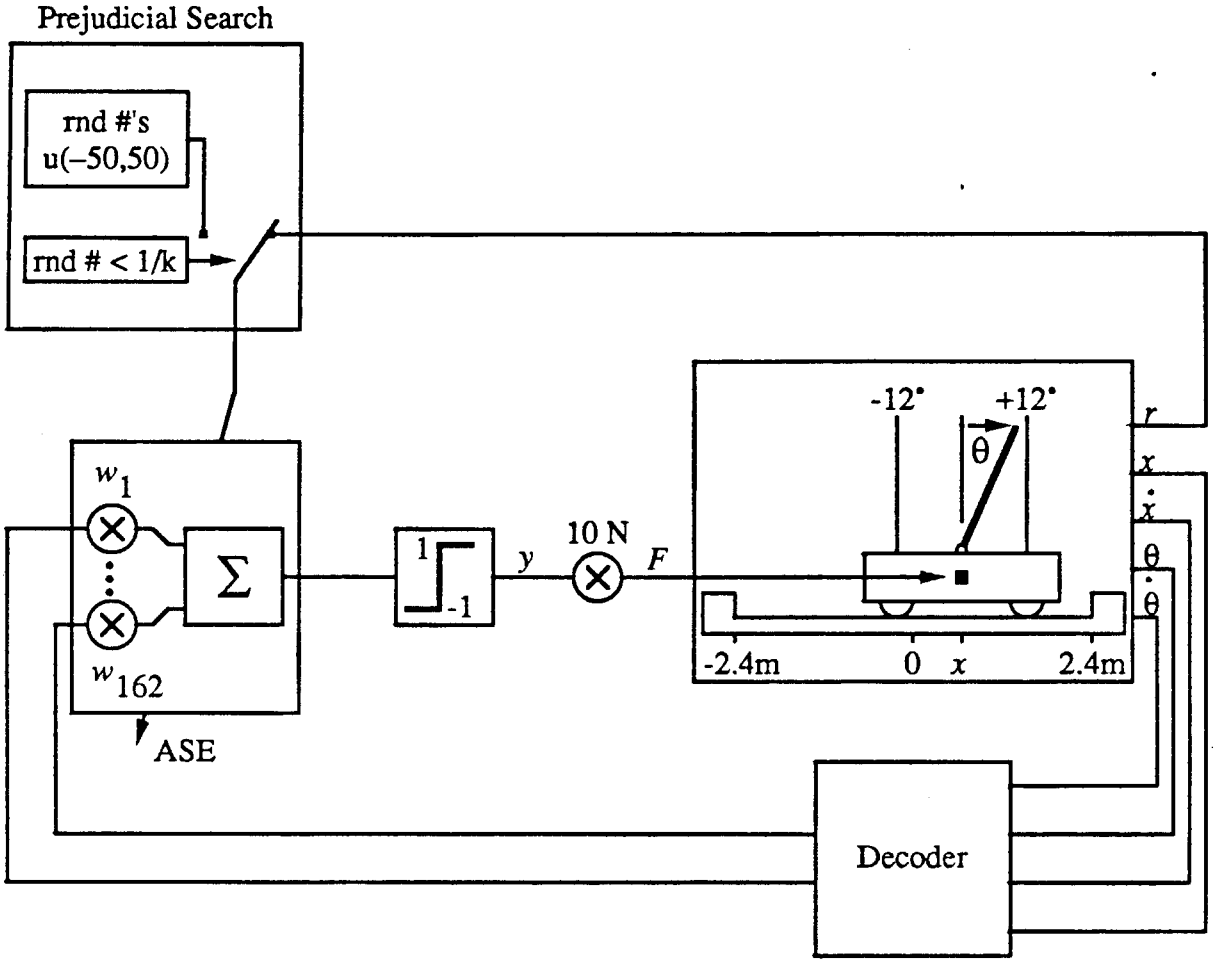

Fig. 6. Prejudicial search pole-balancer.

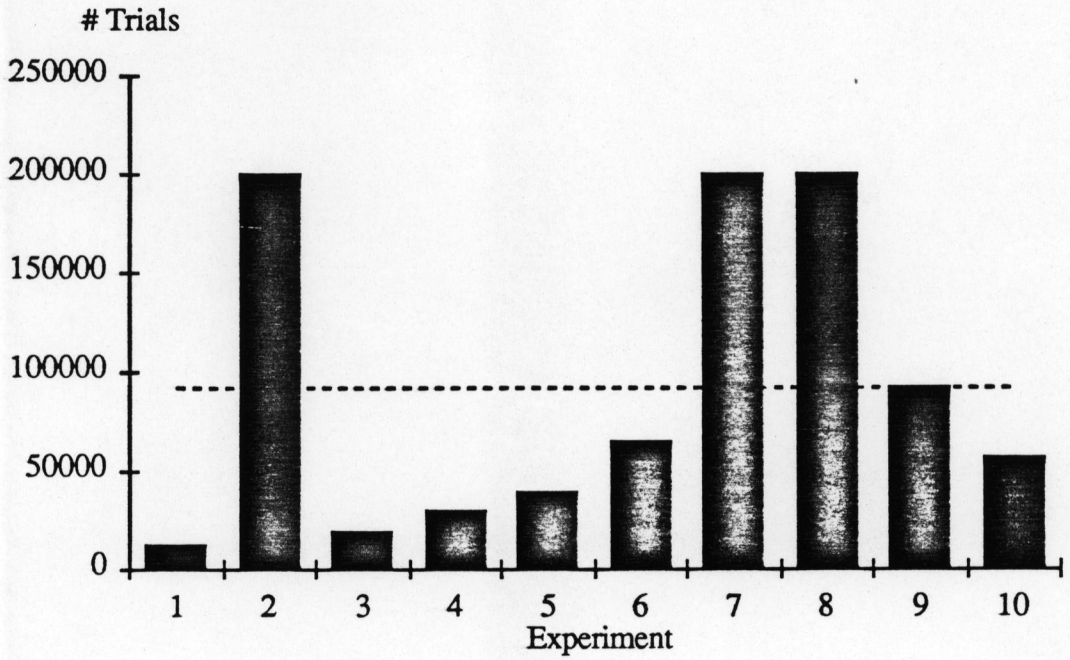

Fig. 7. Prejudicial search number of trials to find successful weights. Dashed line is average.

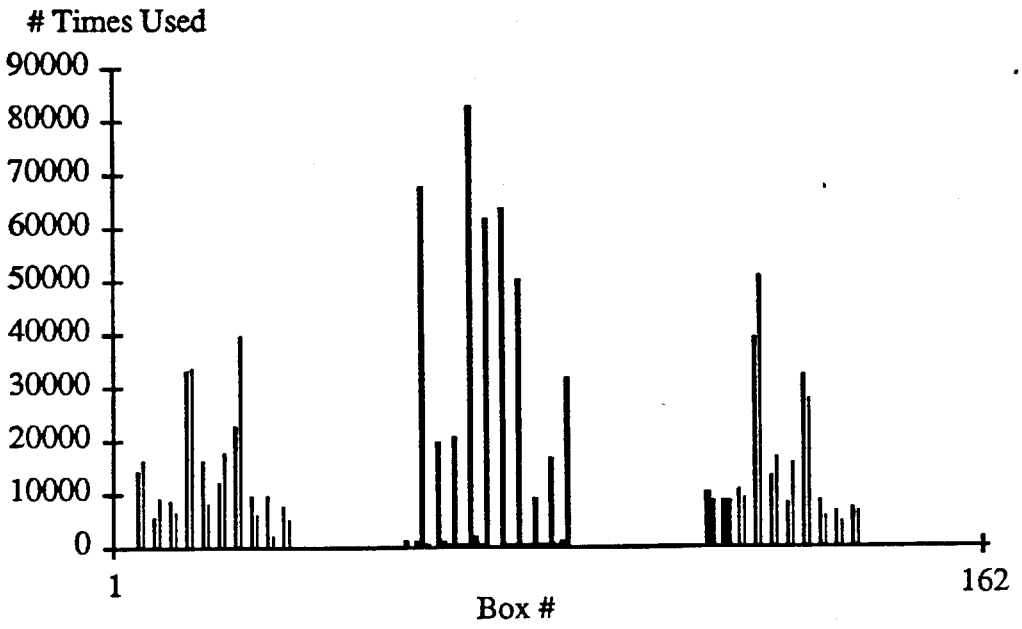

Fig. 8. Number of times boxes were entered during prejudicial search.

4 Applying the Prejudicial Search

The architecture from the Barto and Sutton ASE(no ACE) [1] is also used with prejudicial search to demonstrate how prejudicial search can improve the ASE's performance. This combination is illustrated in Figure 6. The first test chose 100,000 random weight vectors in the range [-50,50]. The best performance was 27.08 seconds before a failure. Ignoring the noise factor, there are 2^{162} possible solutions. This shows the difficulty of finding optimal solutions by random search for this particular architecture. However, an average of 80 out of 162 regions are used in any successful trial. Over the set of successful trials, certain regions are never reached. This implies a greater percentage of solutions. To provide a statistically accurate comparison, the ASE and the ASE with prejudicial search each ran 10 times from an initial random weight vector until a successful trial or a fixed limit on the iterations was reached. The limit Barto and Sutton [1] used was 10,000 seconds. For our simulations, the upper bound was 20,000 seconds. If a failed trial has lasted beyond 5,000 seconds, it is near the optimal solution. Any given set of trials was run until a success or 200,000 failures. If a failed trial occurred, 200,000 seconds were counted for its contribution to the average. On average, the plain ASE converged to an optimal controller after 106,987 failures. The one failure kept the pole balanced for 12,655 seconds. The ASE with prejudicial search used the same update rule as the plain ASE for times when a uniform choice is not taken. As stated above, uniform choices of the weights will only be accepted when better solutions are chosen. Therefore, the addition of the occasional uniform search will not force us to leave a good area. On the other hand, it may allow us to leave a local minima. The ASE with the prejudicial search found the optimal solution within an average of 91,145 failures. The three failures kept the pole balanced for 6633, 18038, and 11067 seconds.

One of the major reasons these solutions worked well is the quantization of the solution space. For the successful solutions, a limit cycle was entered, where it entered the same regions repeatedly and ignored other regions. Therefore, only those weights are important for the solution. Figure 8 shows the boxes(regions) used and the number of times each box is entered in a sample sucessful run.

5 Future Work

This paper examined the tradeoffs between speed and generality of a solution. Four techniques are used to solve the main pole-balancer problem. The most specific approach, the perceptron, found solutions very easily. The architecture used by Barto and Sutton [1] was more general, but was limited to problems that could be broken into regions with uniform characteristics. Jordan's approach [3] was more general still, but could run into minima problems. The prejudicial search [2] was the most general because it guaranteed convergence to a global minima as time approaches ∞, and it could be combined with any of the above architectures.

To make this simple task harder, we could allow continuous weight ranges and variable starting positions. Unless the starting position is near the extremes, it can be ignored. A more valid generalization would be variable thetas because nonlinear elements of the system would dominate for larger thetas. The problem is important because it provides a basis for designing systems that are general enough to handle more complex control systems. Some examples include multiple degree of freedom/multiple constraint robotic systems going through a continuous motion. By combining prejudicial search with other learning methods, the type of cost surface will not be constrained. Therefore, a more general class of systems can be controlled effectively. In future work, prejudicial search will be combined with other paradigms, such as genetic algorithms, gradient descent, and forward models, in systems with a more general architecture.

References

[1] A.G. Barto, R.S. Sutton, and C.S. Anderson. Neuronlike adaptive elements that can solve difficult learning problems. *IEEE Transactions Systems, Man, and Cybernetics*, SMC-13:834–846, 1983.

[2] Neil E. Cotter, Thierry M. Guillerm, Jerome B. Soller, and Peter R. Conwell. Prejudicial Searches and the Pole Balancer. *submitted to IJCNN*, 1991.

[3] M.I. Jordan and R.A. Jacobs. Learning to control an unstable control system with forward modelling. *Advances in Neural Information Processing Systems 2*, 324–331, 1990.

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

[5] B. Widrow and F.W. Smith. Pattern recognizing control systems. *Computer and Information Sciences*, 288–317, 1964.