Chapter 11:
Case Studies

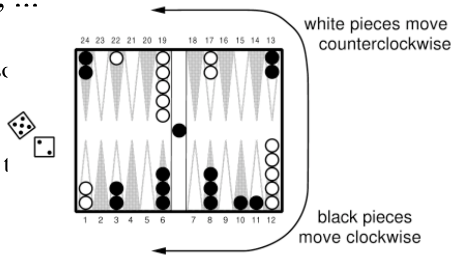
Objectives of this chapter:

- Illustrate trade-offs and issues that arise in real applications
- Illustrate use of domain knowledge
- Illustrate representation development
- Some historical insight: Samuel’s checkers player

TD Gammon

Tesauro 1992, 1994, 1995, ...

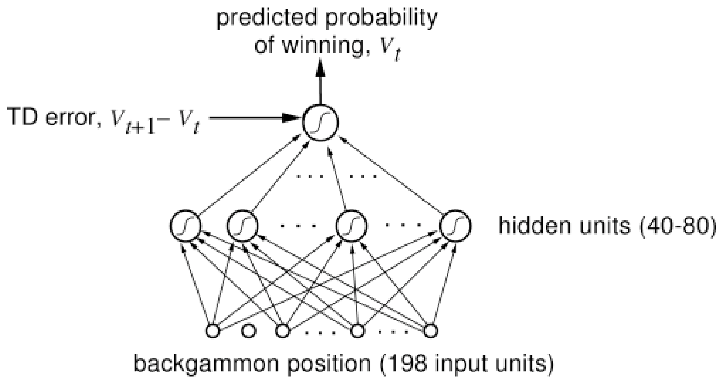
- White has just rolled a 5 and a 2 sc can move one of his pieces 5 and one (possibly the same) 2 steps
- Objective is to advance all pieces t points 19-24
- Hitting
- Doubling
- 30 pieces, 24 locations implies enormous number of configurations
- Effective branching factor of 400


white pieces move counterclockwise
24 23 22 21 20 19
18 17 16 15 14 13
1 2 3 4 5 6
7 8 9 10 11 12
black pieces move clockwise

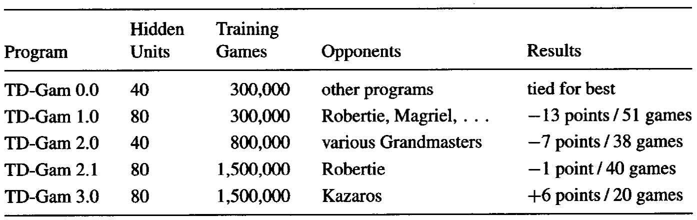
A Few Details

- Reward: 0 at all times except those in which the game is won, when it is 1
- Episodic (game = episode), undiscounted
- Gradient descent TD(λ) with a multi-layer neural network
  - weights initialized to small random numbers
  - backpropagation of TD error
  - four input units for each point; unary encoding of number of white pieces, plus other features
- Use of afterstates
- Learning during self-play

Multi-layer Neural Network


predicted probability of winning, V_t
TD error, $V_{t+1} - V_t$
hidden units (40-80)
backgammon position (198 input units)

Summary of TD-Gammon Results



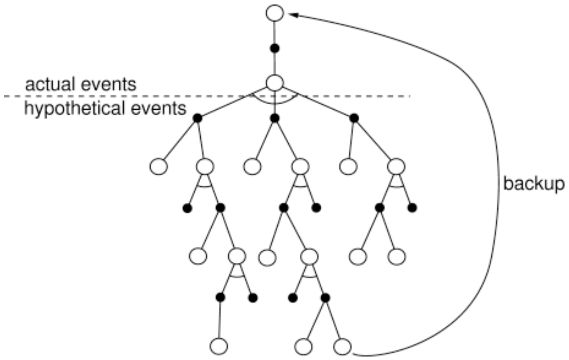
Program	Hidden Units	Training Games	Opponents	Results
TD-Gam 0.0	40	300,000	other programs	tied for best
TD-Gam 1.0	80	300,000	Robertie, Magriel, . . .	−13 points / 51 games
TD-Gam 2.0	40	800,000	various Grandmasters	−7 points / 38 games
TD-Gam 2.1	80	1,500,000	Robertie	−1 point / 40 games
TD-Gam 3.0	80	1,500,000	Kazaros	+6 points / 20 games

Samuel's Checkers Player

Arthur Samuel 1959, 1967

- Score board configurations by a "scoring polynomial" (after Shannon, 1950)
- Minimax to determine "backed-up score" of a position
- Alpha-beta cutoffs
- Rote learning: save each board config encountered together with backed-up score
  - needed a "sense of direction": like discounting
- Learning by generalization: similar to TD algorithm

Samuel's Backups



The Basic Idea

> ". . . we are attempting to make the score, calculated for the current board position, look like that calculated for the terminal board positions of the chain of moves which most probably occur during actual play."

A. L. Samuel

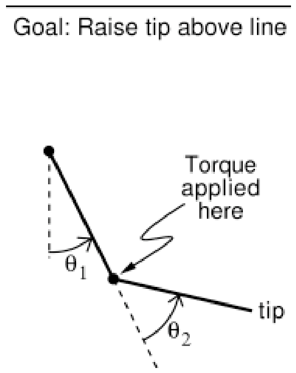
***Some Studies in Machine Learning Using the Game of Checkers*, 1959**

More Samuel Details

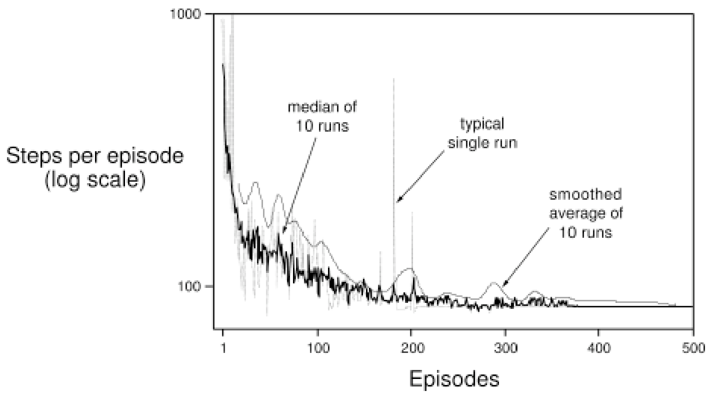
- Did not include explicit rewards
  - Instead used the “piece advantage” feature with a fixed weight
  - No special treatment of terminal positions
  - This can lead to problems . . .
- Generalization method produced “better than average” play; “tricky but beatable”
- Ability to search through feature set and combine features
- Supervised mode: “book learning”
- Signature tables

The Acrobot

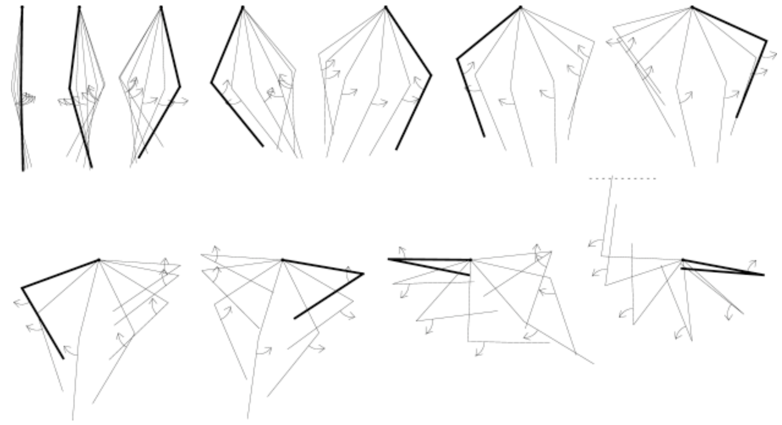
Spong 1994



Acrobot Learning Curves for Sarsa(λ)

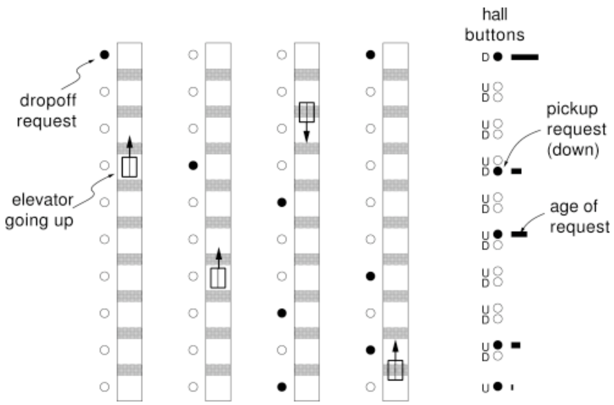


Typical Acrobot Learned Behavior



Elevator Dispatching

Crites and Barto 1996



Semi-Markov Q-Learning

Continuous-time problem but decisions in dicrete jumps

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad \text{becomes} \quad R_t = \int_0^{\infty} e^{-\beta\tau} r_{t+\tau} d\tau$$

Suppose system takes action a from state s at time t_1,
and next decision is needed at time t_2 in state s' :

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[\int_{t_1}^{t_2} e^{-\beta(\tau - t_1)} r_\tau d\tau + e^{-\beta(t_2 - t_1)} \max_{a'} Q(s',a') - Q(s,a)\right]$$

Passenger Arrival Patterns

Up-peak and Down-peak traffic

- Not equivalent: down-peak handling capacity is much greater than up-peak handling capacity; so up-peak capacity is limiting factor.
- Up-peak easiest to analyse: once everyone is onboard at lobby, rest of trip is determined. The only decision is when to open and close doors at lobby. Optimal policy for pure case is: close doors when threshold number on; threshold depends on traffic intensity.
- More policies to consider for two-way and down-peak traffic.
- We focus on down-peak traffic pattern.

Control Strategies

- Zoning: divide building into zones; park in zone when idle. Robust in heavy traffic.
- Search-based methods: greedy or non-greedy. Receding Horizon control.
- Rule-based methods: expert systems/fuzzy logic; from human "experts"
- Other heuristic methods: Longest Queue First (LQF), Highest Unanswered Floor First (HUFF), Dynamic Load Balancing (DLB)
- Adaptive/Learning methods: NNs for prediction, parameter space search using simulation, DP on simplified model, non-sequential RL

The Elevator Model

(from Lewis, 1991)

Discrete Event System: continuous time, asynchronous elevator operation

Parameters:

- Floor Time (time to move one floor at max speed): 1.45 secs.
- Stop Time (time to decelerate, open and close doors, and accelerate again): 7.19 secs.
- Turn Time (time needed by a stopped car to change directions): 1 sec.
- Load Time (the time for one passenger to enter or exit a car): a random variable with range from 0.6 to 6.0 secs, mean of 1 sec.
- Car Capacity: 20 passengers

Traffic Profile:

- Poisson arrivals with rates changing every 5 minutes; down-peak

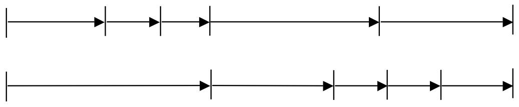
State Space

- 18 hall call buttons: 2^{18} combinations
- positions and directions of cars: 18^4 (rounding to nearest floor)
- motion states of cars (accelerating, moving, decelerating, stopped, loading, turning): 6
- 40 car buttons: 2^{40}
- Set of passengers waiting at each floor, each passenger's arrival time and destination: unobservable. However, 18 real numbers are available giving elapsed time since hall buttons pushed; we discretize these.
- Set of passengers riding each car and their destinations: observable only through the car buttons

Conservatively about 10^{22} states

Actions

- When moving (halfway between floors):
  - stop at next floor
  - continue past next floor
- When stopped at a floor:
  - go up
  - go down
- Asynchronous



Constraints

standard

- A car cannot pass a floor if a passenger wants to get off there
- A car cannot change direction until it has serviced all onboard passengers traveling in the current direction
- Don’t stop at a floor if another car is already stopping, or is stopped, there

special heuristic

- Don’t stop at a floor unless someone wants to get off there
- Given a choice, always move up

→ **Stop and Continue**

Performance Criteria

Minimize:

- Average wait time
- Average system time (wait + travel time)
- % waiting > T seconds (e.g., T = 60)
- Average squared wait time (to encourage fast and fair service)

Average Squared Wait Time

Instantaneous cost:

$$r_\tau = \sum_p \left(\text{wait}_p(\tau)\right)^2$$

Define return as an integral rather than a sum (Bradtke and Duff, 1994):

$$\sum_{t=0}^{\infty} \gamma^2 r_t$$

becomes

$$\int_0^{\infty} e^{-\beta\tau} r_\tau d\tau$$

Algorithm

Repeat forever:

1. In state x at time t_x, car c must decide to STOP or CONTINUE
2. It selects an action using Boltzmann distribution (with decreasing temperature) based on current Q values
3. The next decision by car c is required in state y at time t_y
4. Implements the gradient descent version of the following backup using backprop:

$$Q(x,a) \leftarrow Q(x,a) + \alpha\left[\int_{t_x}^{t_y} e^{-\beta(\tau - t_x)} r_\tau d\tau + e^{-\beta(t_y - t_x)} \max_{a'} Q(y,a') - Q(x,a)\right]$$

5. $x \leftarrow y,\ t_x \leftarrow t_y$

Computing Rewards

Must calculate

$$\int_0^{\infty} e^{-\beta(\tau - t_x)} r_\tau d\tau$$

- "Omniscient Rewards": the simulator knows how long each passenger has been waiting.
- "On-Line Rewards": Assumes only arrival time of first passenger in each queue is known (elapsed hall button time); estimate arrival times

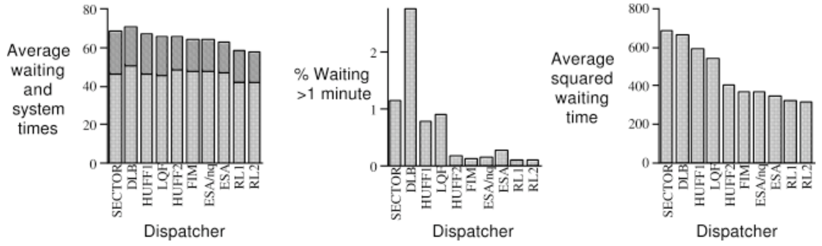
Neural Networks

47 inputs, 20 sigmoid hidden units, 1 or 2 output units

Inputs:

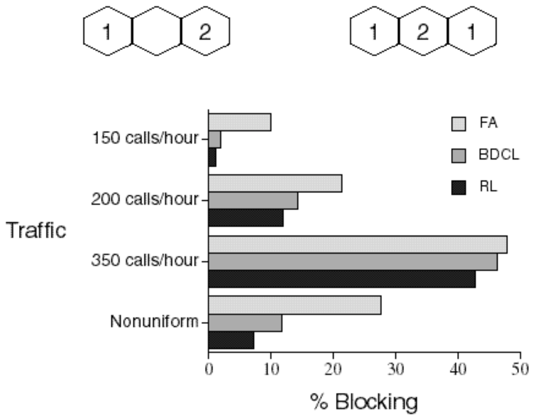
- 9 binary: state of each hall down button
- 9 real: elapsed time of hall down button if pushed
- 16 binary: one on at a time: position and direction of car making decision
- 10 real: location/direction of other cars: “footprint”
- 1 binary: at highest floor with waiting passenger?
- 1 binary: at floor with longest waiting passenger?
- 1 bias unit ≡ 1

Elevator Results



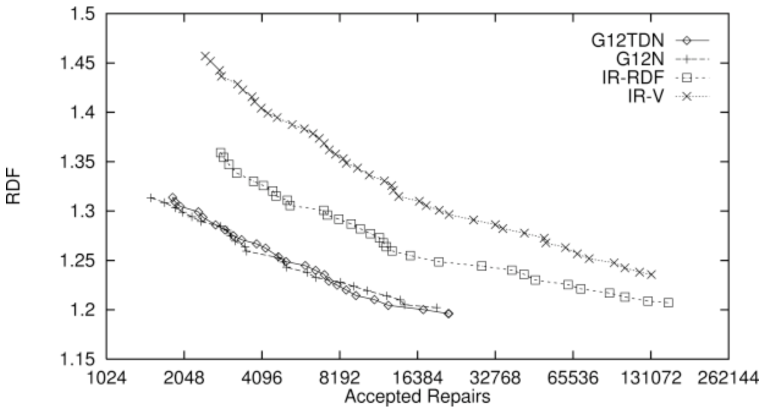
Dynamic Channel Allocation

Singh and Bertsekas 1997

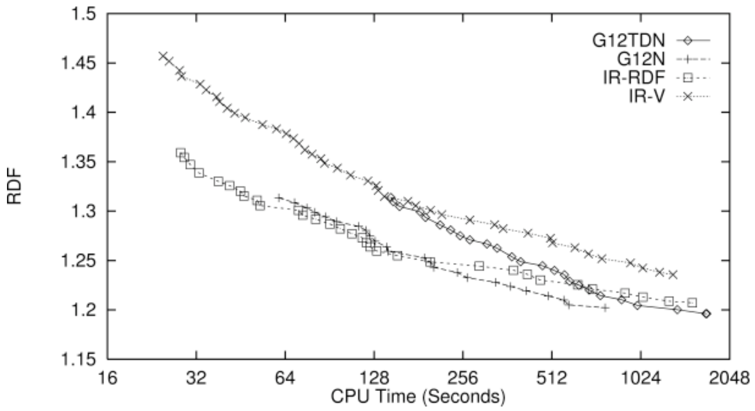


Job-Shop Scheduling

Zhang and Dietterich 1995, 1996



Job-Shop Scheduling



Autonomous Helicopter Flight

A. Ng, Stanford; H. Kim, M. Jordon, S. Sastry, Berkeley

Model-Based Direct Policy Search

- Identify model of helicopter dynamics as flown by human pilot
- Model using locally-weighted linear regression
- Estimate values via Monte Carlo evaluation
- Simple stochastic hillclimbing to adapt policy neural network
- Does not store a value function
- To hover: 30 evaluations of 35 seconds of flying time each

Quadrupedal Locomotion

Nate Kohl & Peter Stone, Univ of Texas at Austin

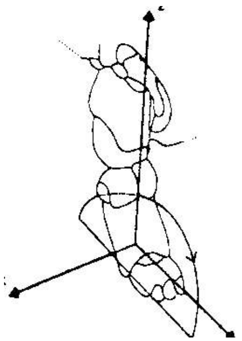
All training done with physical robots: Sony Aibo ERS-210A

Before Learning

After 1000 trials, or about 3 hours

Direct Policy Search

Policy Parameterization:



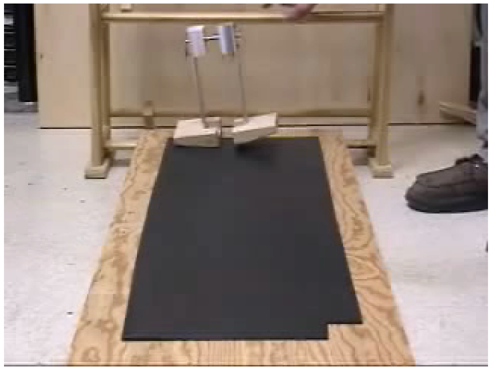
- ❒ Half-elliptical locus for each foot
- ❒ 12 parameters:
  - Position of front locus (x, y, z)
  - Position of rear locus (x, y, z)
  - Locus length
  - Locus skew (for turning)
  - Height of front of body
  - Height of rear of body
  - Time for each foot to move through locus
  - Fraction of time each foot spends on the ground

Simple stochastic hillclimbing to increase speed

Learning Control for Dynamically Stable Walking Robots

Russ Tedrake, Teresa Zhang, H. Sebastion Seung, MIT

Start with a Passive Walker



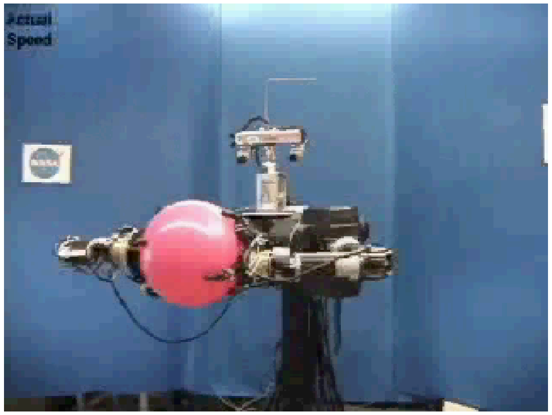
Value Function + Policy Adaptation

- ❒ Sophisticated form of Actor-Critic algorithm
- ❒ Passive walker + 4 actuators: roll and pitch of each foot
- ❒ Value function and policy represented by linear function approximators
- ❒ Behavior is periodic; learning tunes “return map”
- ❒ Goal: walk on the flat like the passive walker walks on a slope

http://hebb.mit.edu/~russt/robots

Grasp Control

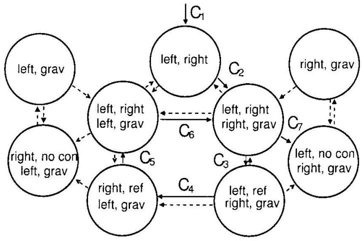
R. Platt, A. Fagg, R. Grupen, Univ of Mass



Umass Torso: "Dexter"

Control Basis Approach

- A set of parameterized closed-loop controllers
- Multiple controllers can operate at the same time
- Sequencing controllers and combinations of controllers can generate a variety of behavior
- ADP done in a smallish abstract state space:


C1
left, grav
left, right
C2
right, grav
left, right
left, grav
left, right
right, grav
C7
C6
right, no con
left, grav
C5
C3
left, no con
right, grav
C4
right, ref
left, grav
left, ref
right, grav