



Lecture 8 of 42

CSP Techniques and Game Tree Search Discussion: Project Plans, MP2 Design

Monday, 11 September 2006

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/v9v3>

Course web site: <http://www.kddresearch.org/Courses/Fall-2006/CIS730>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Section 6.2 – 6.8, p. 162 – 189, Russell & Norvig 2nd edition



Lecture Outline

- Reading for Next Class: Sections 6.2 – 6.8, R&N 2^e
- Today: CSP Concluded
- This Week: Intro to Game Theory
 - * Game tree search: minimax, alpha-beta
 - * Ideas from economics
 - * Relationship between heuristics for games and other heuristics
 - * Role of constraints, learning
- Looking Ahead: Knowledge Representation and Logic
 - * Rest of this month and first week of October
 - * Classical knowledge representation
 - * Relation to planning
 - * Where things break in practice
 - * Segue (back) to uncertain reasoning and preferences
- Wednesday: More on Game Theory, Game Tree Search





Constraint satisfaction problems (CSPs): Review

- Standard search problem:
 - * state is a "black box" – any data structure that supports successor function, heuristic function, and goal test
- CSP:
 - * state is defined by variables X_i with values from domain D_i
 - * goal test is a set of constraints specifying allowable combinations of values for subsets of variables
- Simple example of a formal representation language
- Allows useful general-purpose algorithms with more power than standard search algorithms

© 2004 S. J. Russell

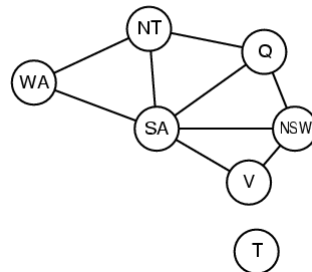
From: <http://aima.eecs.berkeley.edu/slides-ppt/>

Reused with permission.



Constraint graph: Review

- Binary CSP: each constraint relates two variables
- Constraint graph: nodes are variables, arcs are constraints



© 2004 S. J. Russell

From: <http://aima.eecs.berkeley.edu/slides-ppt/>

Reused with permission.





Standard search formulation: Review

Let's start with the straightforward approach, then fix it

States are defined by the values assigned so far

- **Initial state:** the empty assignment { }
 - **Successor function:** assign a value to an unassigned variable that does not conflict with current assignment
→ fail if no legal assignments
 - **Goal test:** the current assignment is complete
1. This is the same for all CSPs
 2. Every solution appears at depth n with n variables
→ use depth-first search
 3. Path is irrelevant, so can also use complete-state formulation
 4. $b = (n - l)d$ at depth l , hence $n! \cdot d^n$ leaves

© 2004 S. J. Russell

From: <http://aima.eecs.berkeley.edu/slides-ppt/>

Reused with permission.



Arc consistency algorithm AC-3: Review

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains

inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \dots, X_n\}$

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)

if RM-INCONSISTENT-VALUES(X_i, X_j) **then**

for each X_k **in** NEIGHBORS[X_i] **do**

add (X_k, X_i) to *queue*

function RM-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff remove a value

removed \leftarrow false

for each x **in** DOMAIN[X_i] **do**

if no value y **in** DOMAIN[X_j] allows (x, y) to satisfy constraint(X_i, X_j)

then delete x from DOMAIN[X_i]; *removed* \leftarrow true

return *removed*

- Time complexity: $O(n^2d^3)$

© 2004 S. J. Russell

From: <http://aima.eecs.berkeley.edu/slides-ppt/>

Reused with permission.





Local search for CSPs

- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
 - * allow states with unsatisfied constraints
 - * operators **reassign** variable values
- Variable selection: randomly select any conflicted variable
- Value selection by **min-conflicts** heuristic:
 - * choose value that violates the fewest constraints
 - * i.e., hill-climb with $h(n)$ = total number of violated constraints

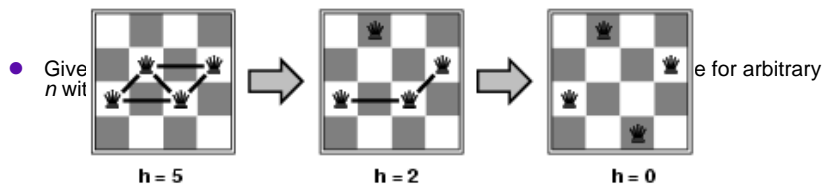
$$P(s|s') = \begin{cases} 1 & \text{if } \Delta E_{s,s'} \geq 0 \\ e^{-\frac{\Delta E}{kT}} & \text{otherwise} \end{cases}$$

© 2004 S. J. Russell
 From: <http://aima.eecs.berkeley.edu/slides-ppt/>
 Reused with permission.



Example: 4-Queens

- **States:** 4 queens in 4 columns ($4^4 = 256$ states)
- **Actions:** move queen in column
- **Goal test:** no attacks
- **Evaluation:** $h(n)$ = number of attacks



© 2004 S. J. Russell
 From: <http://aima.eecs.berkeley.edu/slides-ppt/>
 Reused with permission.



Games versus Search Problems

- **Unpredictable Opponent**
 - * Solution is contingency plan
 - * Time limits
 - ⇒ Unlikely to find goal
 - ⇒ Must approximate
- **Plan of Attack**
 - * Algorithm for perfect play (J. von Neumann, 1944)
 - * Finite horizon, approximate evaluation (C. Zuse, 1945; C. Shannon, 1950, A. Samuel, 1952-1957)
 - * Pruning to reduce costs (J. McCarthy, 1956)

Adapted from slides by S. Russell
UC Berkeley



Types of Games

- **Information: Can Know (Observe)**
 - * ... outcomes of actions / moves?
 - * ... moves committed by opponent?
- **Uncertainty**
 - * Deterministic vs. nondeterministic outcomes
 - * *Thought exercise:* sources of nondeterminism?

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	Clue Concentration Mastermind	bridge, poker, scrabble nuclear war

Adapted from slides by S. Russell
UC Berkeley



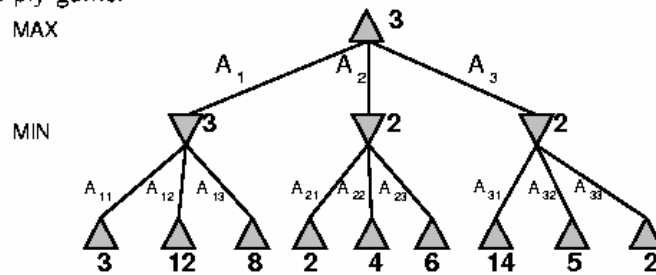


Minimax

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest *minimax value*
= best achievable payoff against best play

E.g., 2-ply game:



Adapted from slides by S. Russell
UC Berkeley

Figure 6.2 p. 164 R&N 2e



Minimax Algorithm: Decision and Evaluation

```
function MINIMAX-DECISION(game) returns an operator
  for each op in OPERATORS[game] do
    VALUE[op] ← MINIMAX-VALUE(APPLY(op, game), game)
  end
  return the op with the highest VALUE[op]
```

```
function MINIMAX-VALUE(state, game) returns a utility value
  if TERMINAL-TEST[game](state) then ← what's this?
    return UTILITY[game](state) ← what's this?
  else if MAX is to move in state then
    return the highest MINIMAX-VALUE of SUCCESSORS(state)
  else
    return the lowest MINIMAX-VALUE of SUCCESSORS(state)
```

Adapted from slides by S. Russell
UC Berkeley

Figure 6.3 p. 166 R&N 2e





Properties of Minimax

- **Complete?**
 - * ... yes, provided following are finite:
 - ⇒ Number of possible legal moves (generative *breadth* of tree)
 - ⇒ "Length of game" (*depth* of tree) – more specifically?
 - * **Perfect vs. imperfect information?**
 - ⇒ Q: What search is perfect minimax analogous to?
 - ⇒ A: Bottom-up breadth-first
- **Optimal?**
 - * ... yes, provided **perfect info** (evaluation function) and **opponent is optimal!**
 - * ... otherwise, guaranteed *if* evaluation function is correct
- **Time Complexity?**
 - * **Depth of tree: m**
 - * **Legal moves at each point: b**
 - * **$O(b^m)$ – NB, $m \approx 100$, $b \approx 35$ for chess!**
- **Space Complexity? $O(bm)$ – why?**

Adapted from slides by S. Russell
UC Berkeley



Resource Limits

Suppose we have 100 seconds, explore 10^4 nodes/second
⇒ 10^6 nodes per move

Standard approach:

- *cutoff test*
e.g., depth limit (perhaps add *quiescence search*)
- *evaluation function*
= estimated desirability of position

Adapted from slides by S. Russell
UC Berkeley





Static Evaluation Function Example: Chess



Black to move
White slightly better



White to move
Black winning

For chess, typically *linear* weighted sum of features

$$\text{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

e.g., $w_1 = 9$ with

$f_1(s) = (\text{number of white queens}) - (\text{number of black queens})$

etc.

Adapted from slides by S. Russell
UC Berkeley

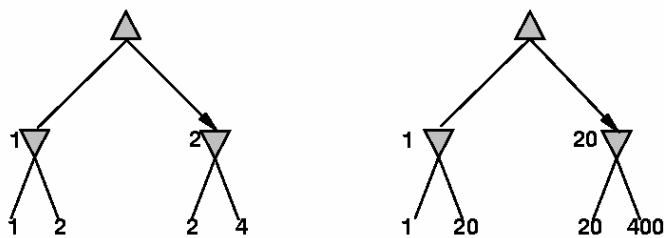
Figure 6.8 p. 173 R&N 2e



Do Exact Values Matter?

MAX

MIN



Behaviour is preserved under any *monotonic* transformation of EVAL

Only the order matters:

payoff in deterministic games acts as an *ordinal utility* function

Adapted from slides by S. Russell
UC Berkeley





Cutting Off Search [1]

MINIMAXCUTOFF is identical to MINIMAXVALUE except

1. TERMINAL? is replaced by CUTOFF?
2. UTILITY is replaced by EVAL

Does it work in practice?

$$b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4$$

4-ply lookahead is a hopeless chess player!

4-ply \approx human novice
8-ply \approx typical PC, human master
12-ply \approx Deep Blue, Kasparov

Adapted from slides by S. Russell
UC Berkeley



Cutting Off Search [2]

● Issues

* Quiescence

- ⇒ Play has “settled down”
- ⇒ Evaluation function unlikely to exhibit wild swings in value in near future

* Horizon effect

- ⇒ “Stalling for time”
- ⇒ Postpones inevitable win or damaging move by opponent
- ⇒ See: Figure 5.5 R&N

● Solutions?

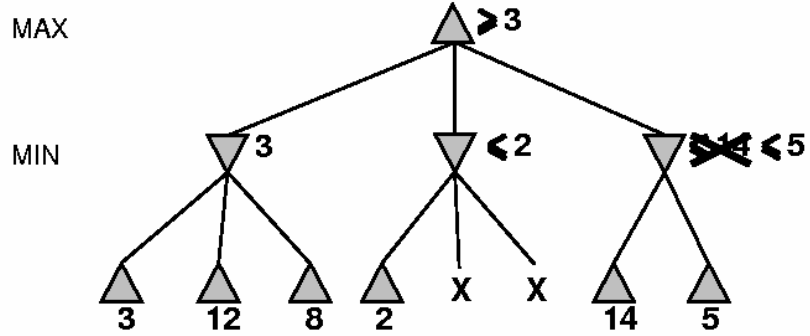
- * Quiescence search: expand non-quiescent positions further
- * “No general solution to horizon problem at present”

Adapted from slides by S. Russell
UC Berkeley





Why Prune?



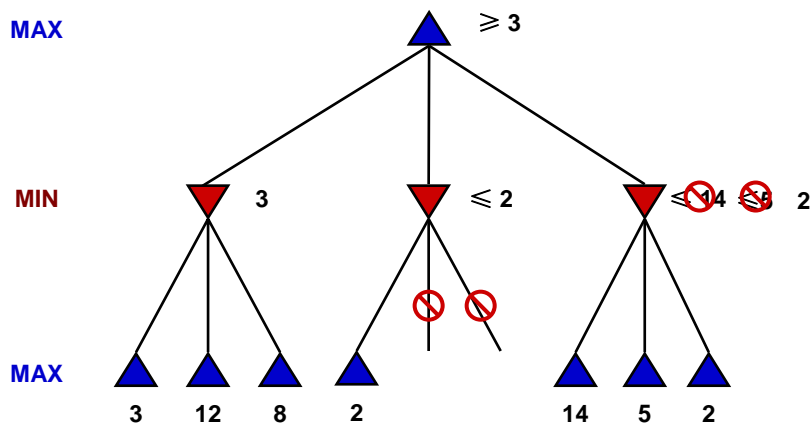
Adapted from slides by S. Russell
UC Berkeley

Figure 6.5 p. 168 R&N 2e



Alpha-Beta (α - β) Pruning: Example [1]

What are α , β values here?



Adapted from slides by S. Russell
UC Berkeley

Figure 6.5 p. 168 R&N 2e





Alpha-Beta (α - β) Pruning: Modified Minimax Algorithm

Basically MINIMAX + keep track of α , β + prune

```

function MAX-VALUE(state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
  inputs: state, current state in game
         game, game description
          $\alpha$ , the best score for MAX along the path to state
          $\beta$ , the best score for MIN along the path to state

  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     $\alpha \leftarrow \text{MAX}(\alpha, \text{MIN-VALUE}(s, \textit{game}, \alpha, \beta))$ 
    if  $\alpha \geq \beta$  then return  $\beta$ 
  end
  return  $\alpha$ 

function MIN-VALUE(state, game,  $\alpha$ ,  $\beta$ ) returns the minimax value of state
  if CUTOFF-TEST(state) then return EVAL(state)
  for each s in SUCCESSORS(state) do
     $\beta \leftarrow \text{MIN}(\beta, \text{MAX-VALUE}(s, \textit{game}, \alpha, \beta))$ 
    if  $\beta \leq \alpha$  then return  $\alpha$ 
  end
  return  $\beta$ 

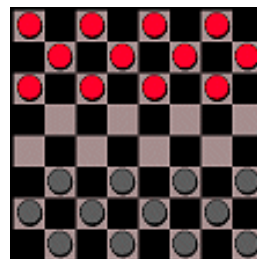
```

Adapted from slides by S. Russell
UC Berkeley



Digression: Learning Evaluation Functions

- **Learning = Improving with Experience at Some Task**
 - * Improve over task *T*,
 - * with respect to performance measure *P*,
 - * based on experience *E*.
- **Example: Learning to Play Checkers**
 - * *T*: play games of checkers
 - * *P*: percent of games won in world tournament
 - * *E*: opportunity to play against self
- **Refining the Problem Specification: Issues**
 - * What experience?
 - * What *exactly* should be learned?
 - * How shall it be *represented*?
 - * What specific algorithm to learn it?
- **Defining the Problem Milieu**
 - * Performance element: How shall the results of learning be applied?
 - * How shall performance element be evaluated? Learning system?





Summary Points

- **Introduction to Games as Search Problems**
 - * **Frameworks**
 - ⇒ Two-player versus multi-player
 - ⇒ Zero-sum versus cooperative
 - ⇒ Perfect information versus partially-observable (hidden state)
 - * **Concepts**
 - ⇒ Utility and representations (e.g., static evaluation function)
 - ⇒ Reinforcements: possible role for machine learning
 - ⇒ Game tree
- **Family of Algorithms for Game Trees: Minimax**
 - * **Propagation of credit**
 - * **Imperfect decisions**
 - * **Issues**
 - ⇒ Quiescence
 - ⇒ Horizon effect
 - * **Need for pruning**



Terminology

- **Constraint Satisfaction Problems (CSP): Min-Conflicts**
- **Game Graph Search**
 - * **Frameworks**
 - ⇒ Two-player versus multi-player
 - ⇒ Zero-sum versus cooperative
 - ⇒ Perfect information versus partially-observable (hidden state)
 - * **Concepts**
 - ⇒ Utility and representations (e.g., static evaluation function)
 - ⇒ Reinforcements: possible role for machine learning
 - ⇒ Game tree: node/move correspondence, search ply
- **Family of Algorithms for Game Trees: Minimax**
 - * **Propagation of credit**
 - * **Imperfect decisions**
 - * **Quiescence**
 - * **Horizon effect**
 - * **Alpha-beta pruning**

