CSCI 5582
Artificial Intelligence

Lecture 7
Jim Martin

Today 9/19

- Review (and finish) search
- Break
- Game Playing Search

Review

- Optimization/Local Search
- Constraint Satisfaction Search

Local Search

- Hillclimbing
- Random-Restart Hillclimbing
- Simulated Annealing

Constraint Satisfaction

- In CSP problems, states are represented as sets of variables, each with values chosen from some domain
- A goal test consists of satisfying constraints on sets of variable/value combinations
- A goal state is one that has no constraint violations

Approaches to CSPs

- As a kind of backtracking search
- As a kind of iterative improvement

Making Backtracking Work

- What it means to be a goal (or not) can be decomposed
  - In CSPs a state is a goal state if *all* of the constraints are satisfied.
  - A state fails as a goal state if *any* constraint is violated
  - Therefore we can check for violations as variables are assigned values

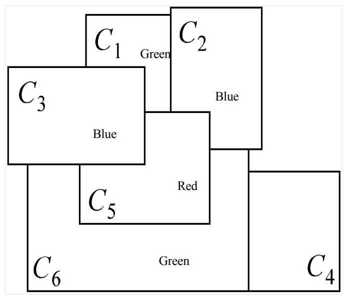
Informed Backtracking CSP Search

- The previous discussion didn't use any notion of heuristic.
- There are two places heuristics can help
  - Which variable to assign next
  - Which value to assign to a variable

Generic CSP Heuristics

- Variables
  - Degree heuristic
    - The one involved in the largest number of constraints
  - Choose the most constrained variable
    - The one with the minimum remaining values
- Values
  - Choose the least constraining value

Graph Coloring



Iterative Improvement

- Sometimes it’s better to look at these problems as optimization problems.
- Where you want to optimize (minimize) the number of constraints violated (to zero would be good)

How?

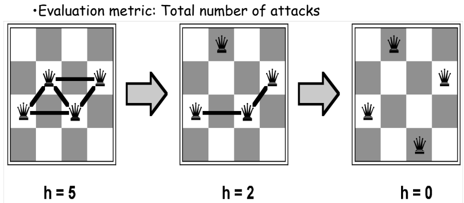
- Randomly assign values to all the variables in the problem (from their domains)
- Iteratively fix the variables (reassign values) that are conflicted.
- Continue until there are no conflicts or no progress

Min Conflict Heuristic

- Randomly choose a variable from the problematic ones.
- Reassign its value to be the one that results in the fewest conflicts
- Continue until there are no conflicts

Min Conflict Example

- States: 4 Queens, 1 per column
- Operators: Move queen in its column
- Goal test: No attacks
- Evaluation metric: Total number of attacks

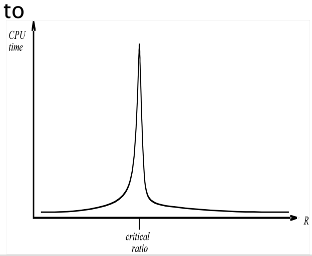


Min Conflict Performance

- Min Conflict seems to have astounding performance.
- For example, it’s been shown to solve arbitrary size (in the millions)
  N-Queens problems in constant time.
- This appears to hold for arbitrary CSPs with the caveat…

Min Conflict Performance

- Except in a certain critical range of the ratio constraints to variables.



Preferences and Constraints

- In practice, applications can get fairly messy
  - Sometimes you want the lowest cost zero conflict solution
  - Sometimes constraints are preferences not true constraints
  - Sometimes some constraints are more important than other constraints. That is, the cost of violating some constraints is more than the cost of violating others.

Admin/Break

- Questions?
- Quiz will be on Thursday for the first 30 minutes or so.
  - Focus is on search
  - Chapters 3,4,5 and 6 (today)

Game Playing Search

- Why study games?
- Why is search a good idea?

9/19/06 CSCI 5582 Fall 2006 19

Typical Assumptions

- Some majors assumptions we've been making:
  - Only an agent's actions change the world
  - World is deterministic and accessible
- Pretty much true in lots of games

9/19/06 CSCI 5582 Fall 2006 20

Why Search?

- Ignoring computational complexity, games are a perfect application for a complete search.
- Of course, ignoring complexity is a bad idea, so games are a good place to study resource bounded searches.

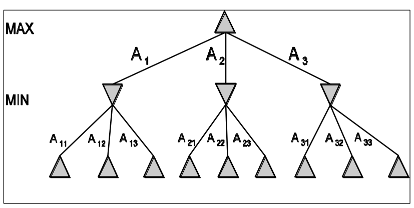
9/19/06 CSCI 5582 Fall 2006 21

MiniMax

- From among the moves available to you, take the best one
- Where the best one is determined by a search using the MiniMax strategy

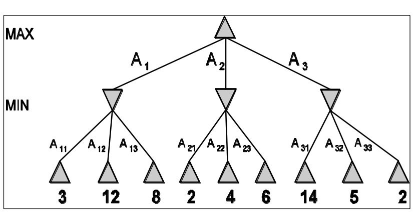
MiniMax

- I'll choose the best move for me (max)
- You'll choose the best move for you (min)



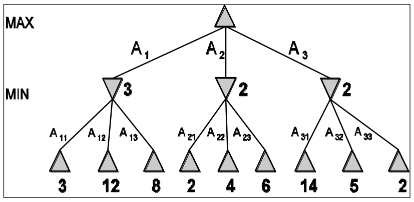
MiniMax

- I'll choose the best move for me (max)
- You'll choose the best move for you (min)



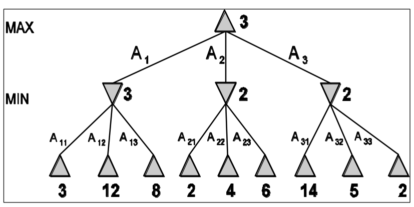
MiniMax

- I'll choose the best move for me (max)
- You'll choose the best move for you (min)



MiniMax

- I'll choose the best move for me (max)
- You'll choose the best move for you (min)



Ideal Case

- Search all the way to the leaves (end game positions)
- Return the leaf (leaves) that leads to a win (for me)
- Anything wrong with that?

More Realistic

- Search ahead to a non-leaf (non-goal) state and evaluate it somehow
- Chess
  - 4 ply is a novice
  - 8 ply is a master
  - 12 ply can compete at the highest level
- In no sense can 12 ply be likened to a search of the whole space

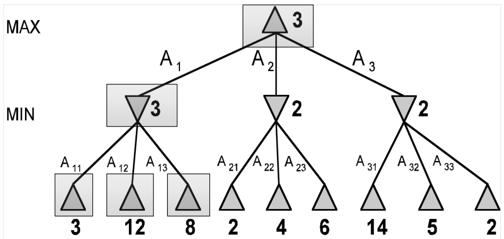
Evaluation Functions

- Need a numerical function that assigns a value to a non-goal state
  - Has to capture the notion of a position being good for one player
  - Has to be fast
  - Has to be fast
  - Typically a linear combination of simple metrics

MiniMax Implemented

- Depth-first, left to right, recursive, depth-limited search
- Only the leaves are evaluated
- Return values represent the best value found below that point in the tree (not the specific moves taken)

Once More



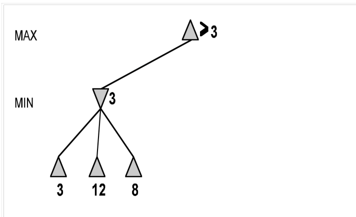
Extensions

- Pruning
- Openings and Closings
- Managing Time

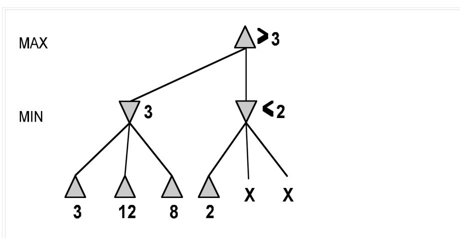
Alpha-Beta Pruning

- Often you can ignore entire sections of the search space and come up with the same answer
- Specifically, if you're exploring a line of play that leads to a worse position for you than another one you've already discovered, then don't explore that line anymore.

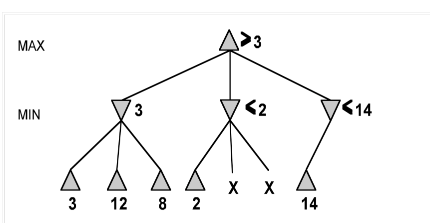
Alpha-Beta Pruning



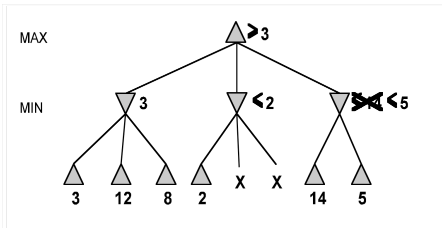
Alpha Beta Pruning



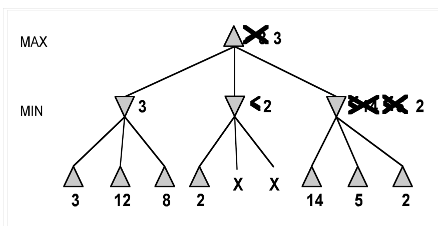
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



How does AB Help?

- After all, all it does is return the same answer you would have gotten otherwise.
- So what's the big deal?

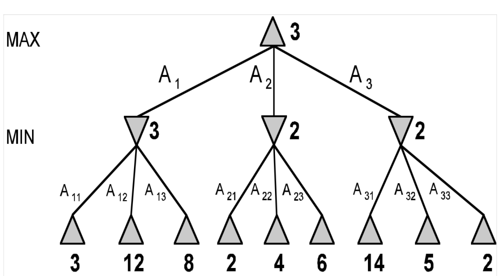
How does AB Help

- It helps if you can use the time saved to look deeper into the tree.
- Moving from b^d to $b^{d/2}$ means that that you can go to 8 ply in the same time it took to go to 4
- Or going from novice to master with the same smarts
- This assumes that you have some way to manage the clock

Improving AB

- Recall that MiniMax is normally a depth-first, left to right, procedure.
- AB works better if the most informative nodes are evaluated first (on the left, as opposed to the right).

Improving AB



Dealing with Time

- In tournament play, you have a time constraint.
- Need some effective way to manage the clock.
- I.e. you need to be sure that you have a move to make when the bell goes off.

Iterative Deepening (again)

- Run MiniMax inside an ID wrapper.
- Remember the best move from previous rounds
- Keep iterating until some time limit is reached
- Key point: You always have an answer available

Openings and Closings

- Do we really need a search right from the start?
- Or at the end?

Game Trivia

- Chinook had closing book with 400 Billion positions
- Deep Blue examined ~100 Billion boards per move
- Often reached 14 ply
- Better pruning/move ordering beats faster better hardware

Next Time

- Quiz (30 min)
- Start on Chapter 7