Outline

Game Playing

Optimal decisions

Minimax

α-β pruning

Imperfect, real-time decisions

Game Playing

Mathematical Game Theory

→ Branch of economics that views any multi-agent environment as a game, provided that the impact of each agent on the others is "significant", regardless of whether the agents are cooperative or competitive.

Game Playing in AI (typical case) :

- Deterministic
- Turn taking
- 2-player
- Zero-sum game of perfect information (fully observable)

Game Playing vs. Search

Game vs. search problem

"Unpredictable" opponent → specifying a move for every possible opponent reply

Time limits → unlikely to find goal, must approximate

Game Playing

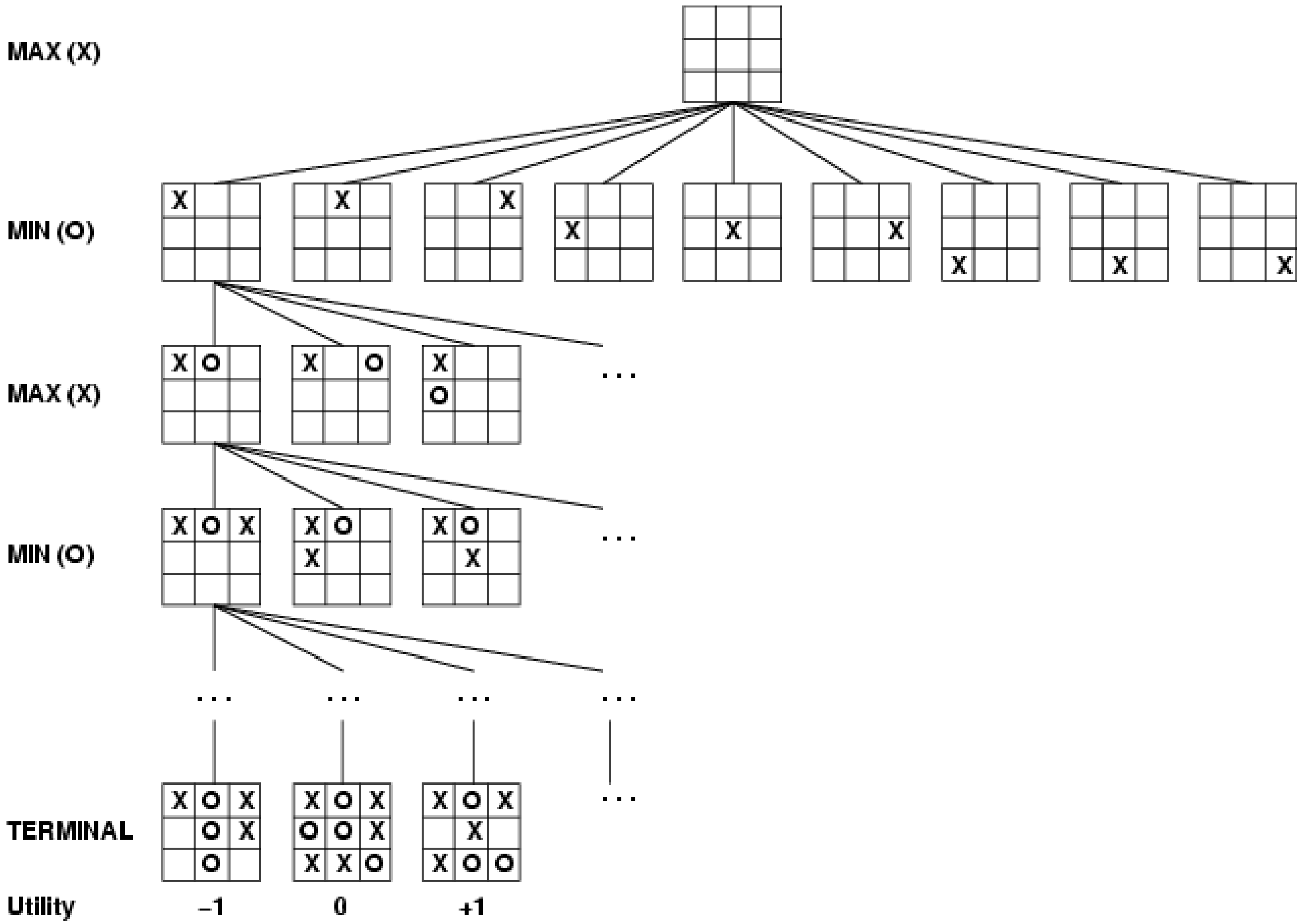
Formal definition of a game:

- Initial state
- Successor function: returns list of *(move, state)* pairs
- Terminal test: determines when game over

  Terminal states: states where game ends
- Utility function (objective function or payoff function): gives numeric value for terminal states

We will consider games with 2 players (**Max and Min**);
Max moves first.

Game Tree Example: Tic-Tac-Toe



Tree from Max's perspective

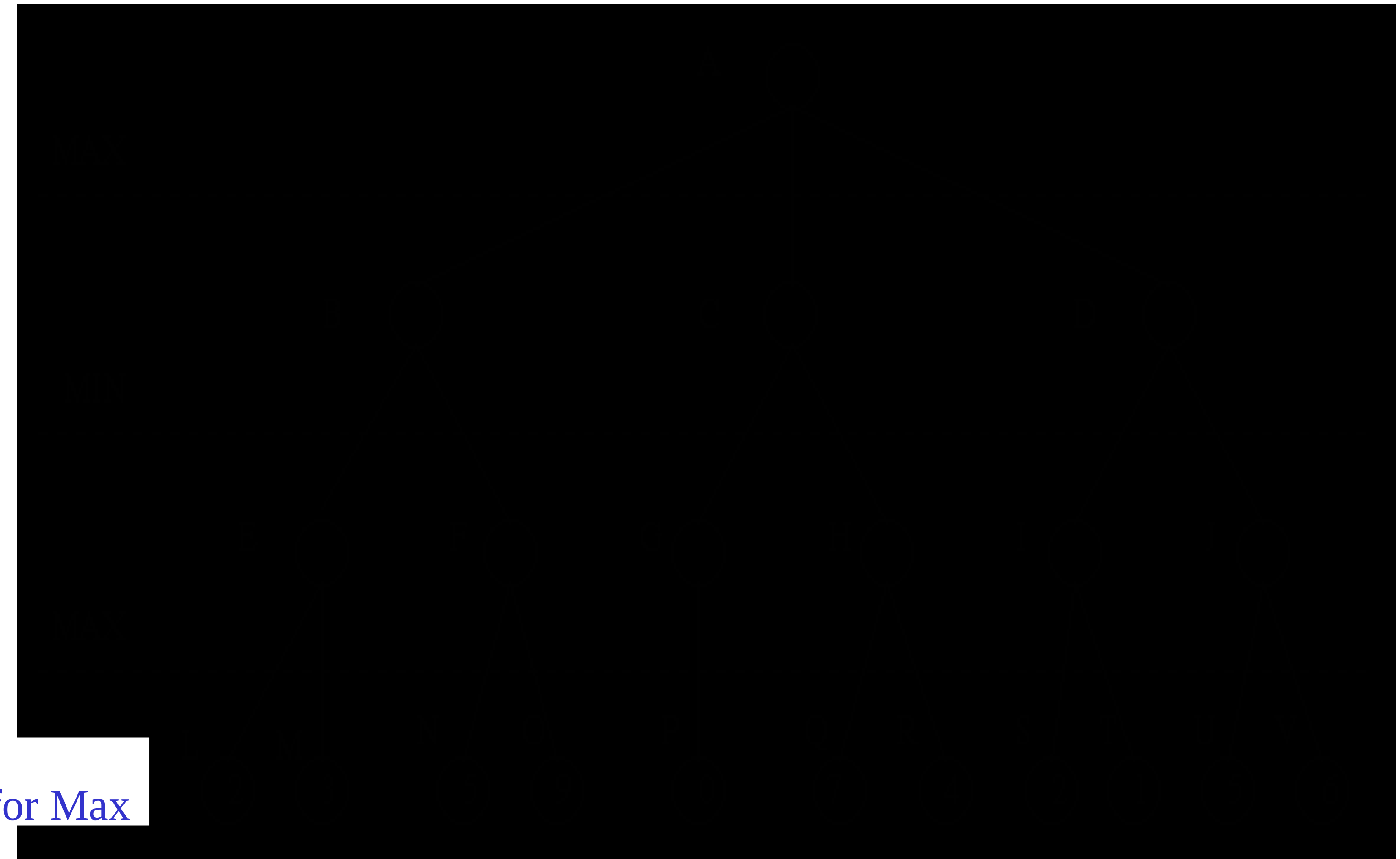
Minimax Algorithm

Minimax algorithm

- Perfect play for deterministic, 2-player game
- Max tries to maximize its score
- Min tries to minimize Max’s score (Min)
- Goal: move to position of highest minimax value

  → Identify best achievable payoff against best play

Minimax Algorithm

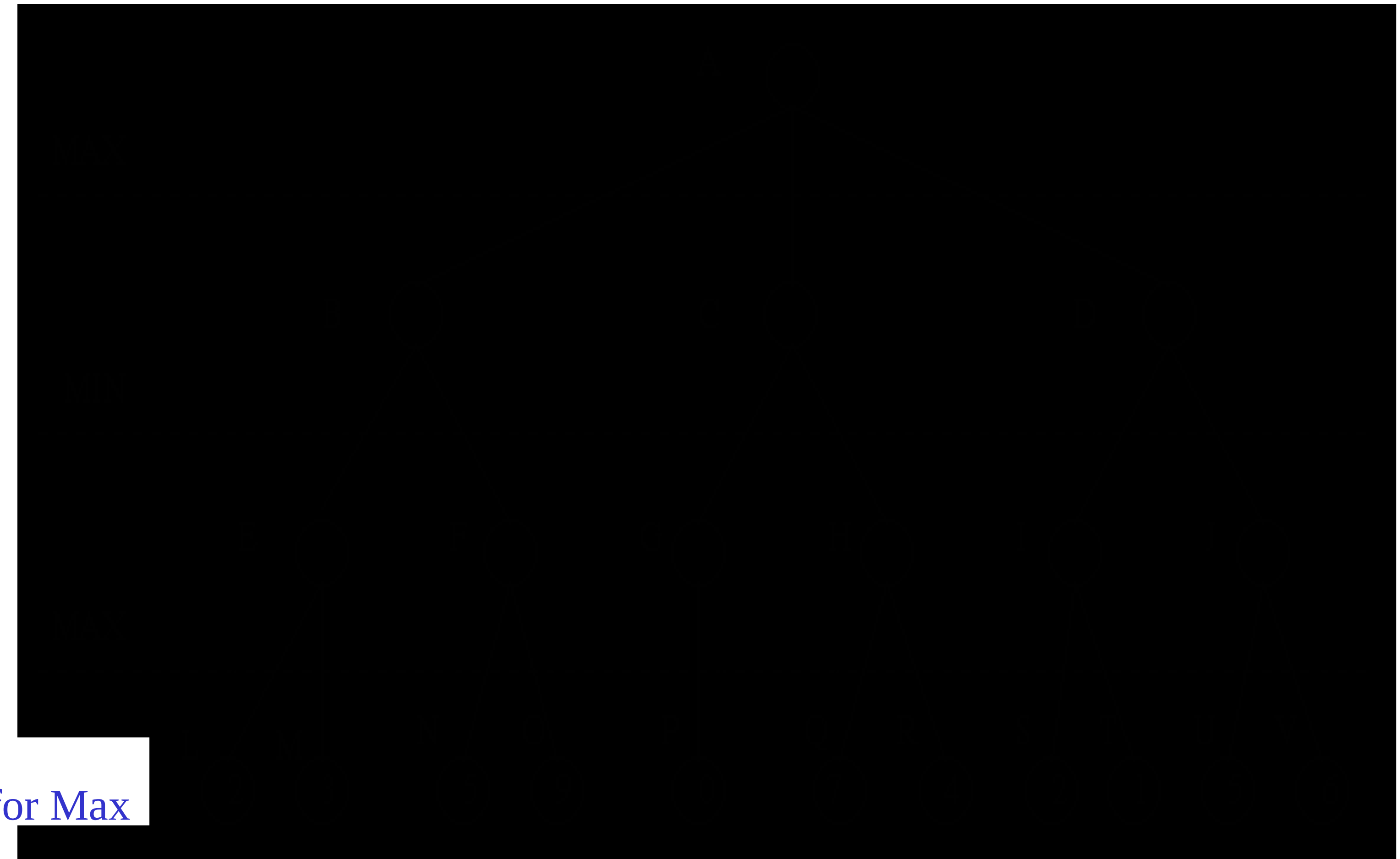


Payoff for Max

Minimax Algorithm (cont'd)

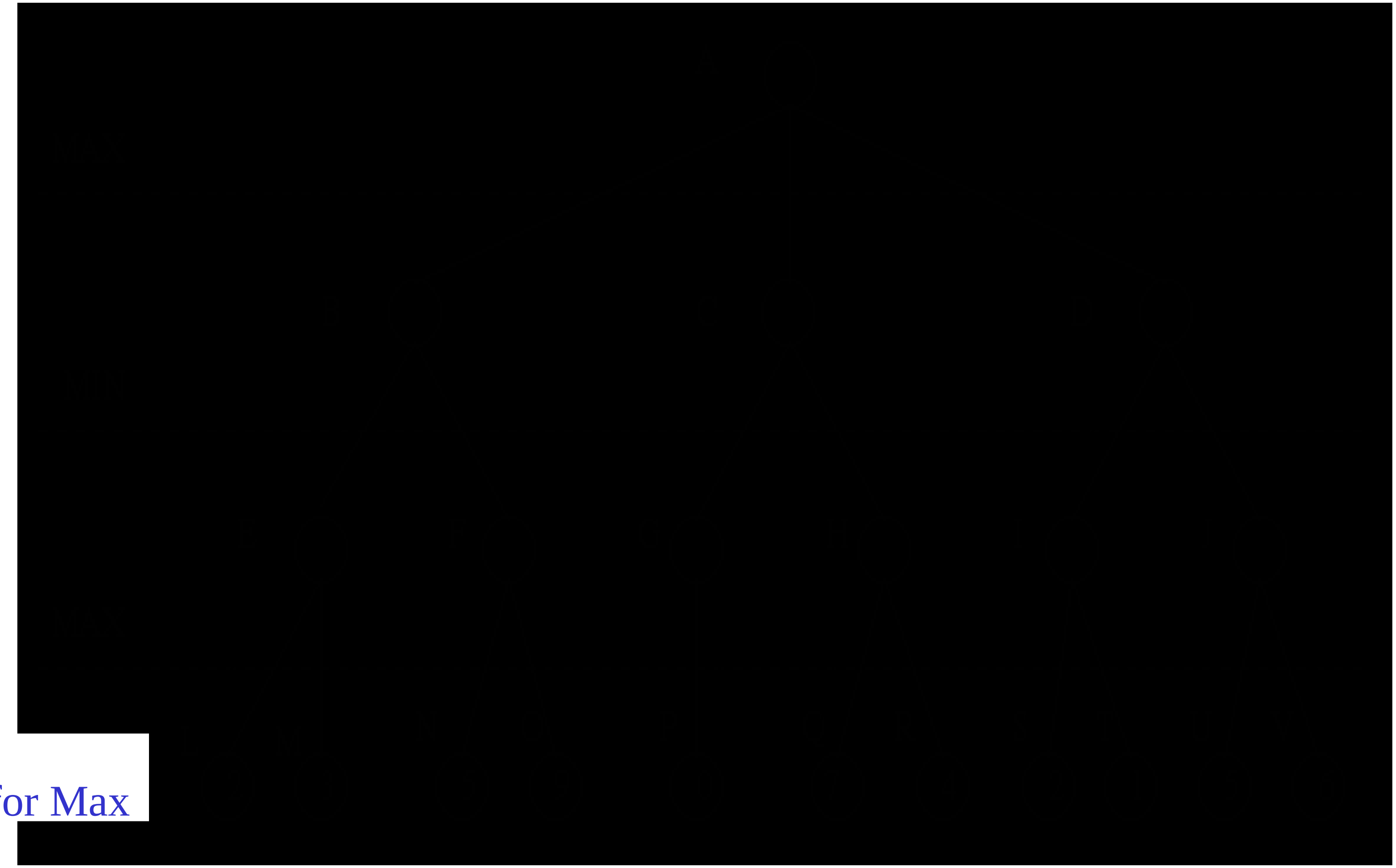
Payoff for Max

Minimax Algorithm (cont’d)



Payoff for Max

Minimax Algorithm (cont’d)



Payoff for Max

Minimax Algorithm (cont’d)

Properties of minimax algorithm:

Complete? Yes (if tree is finite)

Optimal? Yes (against an optimal opponent)

Time complexity? $O(b^m)$

Space complexity? $O(bm)$ (depth-first exploration, if it generates all successors at once)

m – maximum depth of tree; b branching factor

For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games

→ exact solution completely infeasible

m – maximum depth of the tree; b – legal moves;

Minimax Algorithm

Limitations

- Not always feasible to traverse entire tree
- Time limitations

Key Improvement

- Use evaluation function instead of utility
  - Evaluation function provides estimate of utility at given position

→ More soon…

α-β Pruning

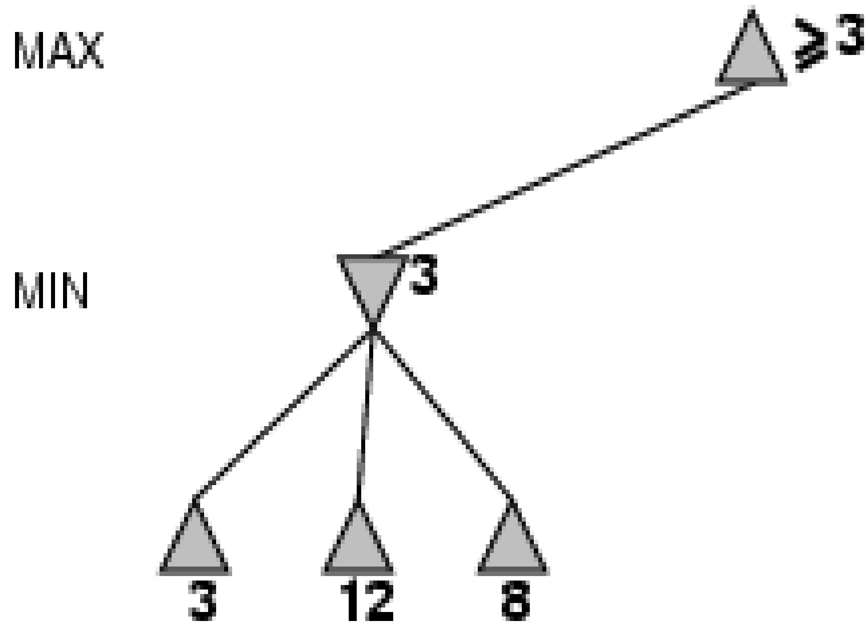
Can we improve search by reducing the size of the game tree to be examined?

→ Yes!!! Using alpha-beta pruning

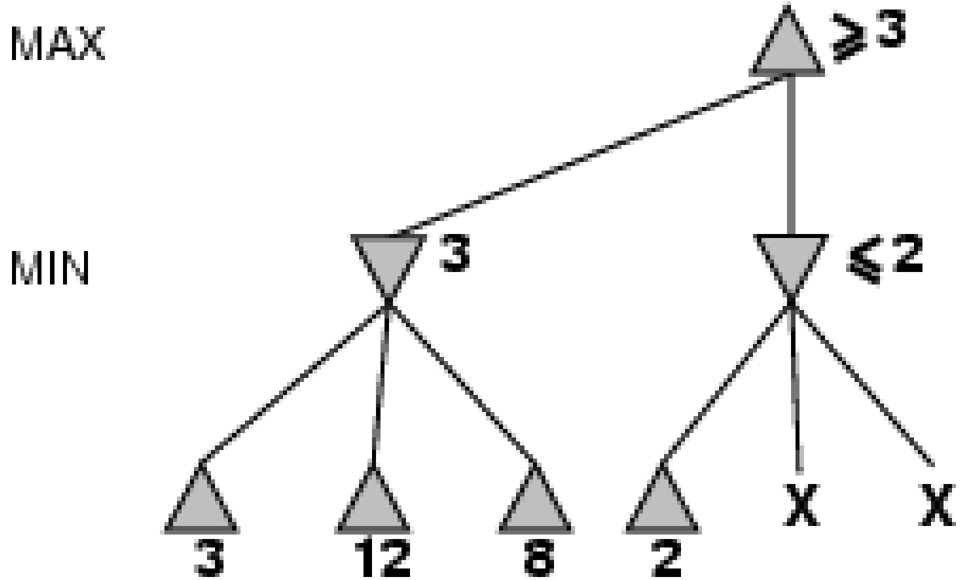
Principle

- If a move is determined worse than another move already examined, then there is no need for further examination of the node.

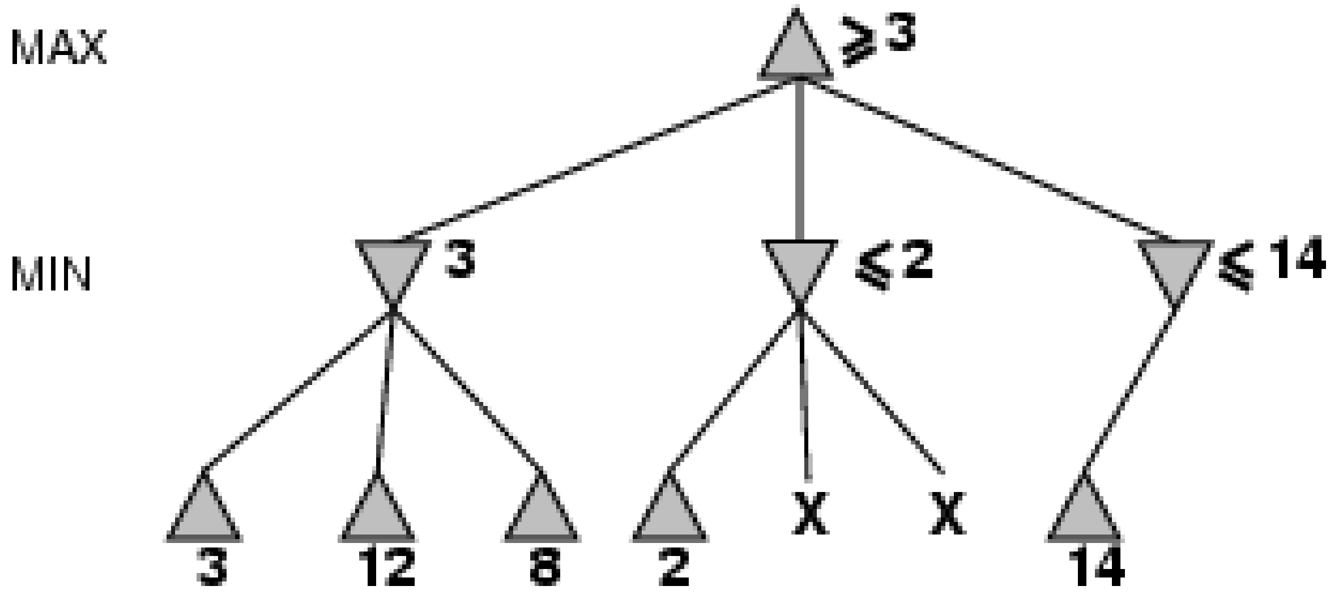
α-β Pruning Example



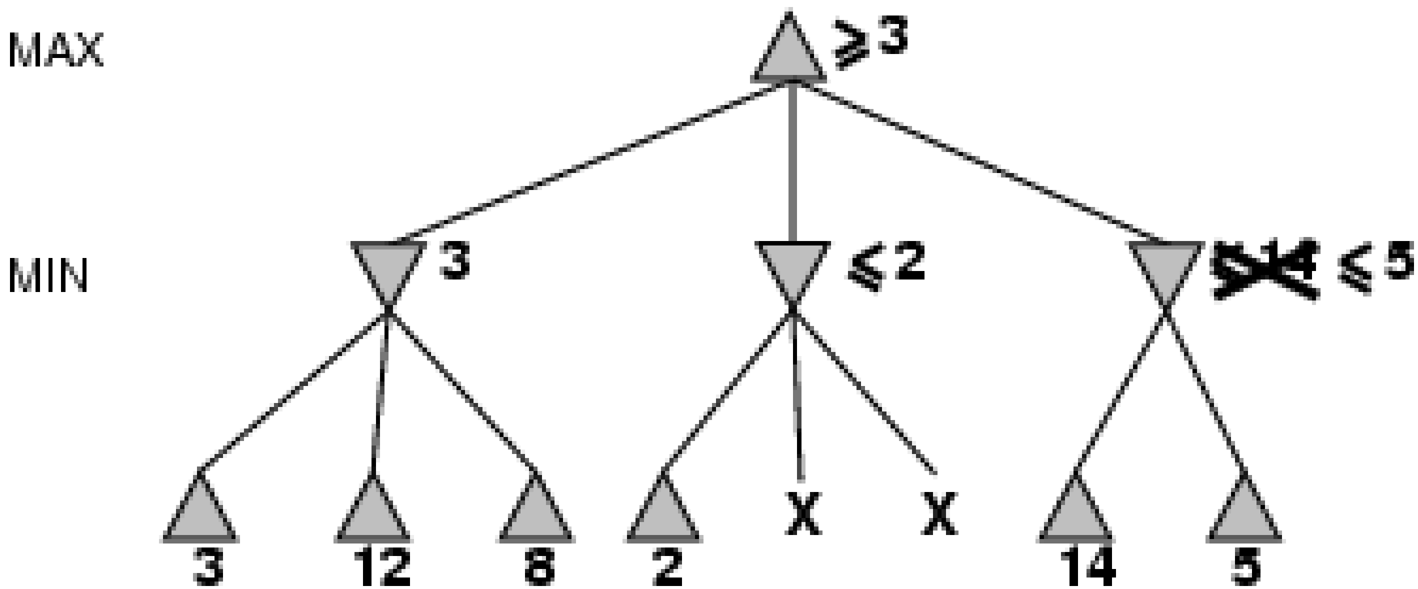
α-β Pruning Example (cont'd)



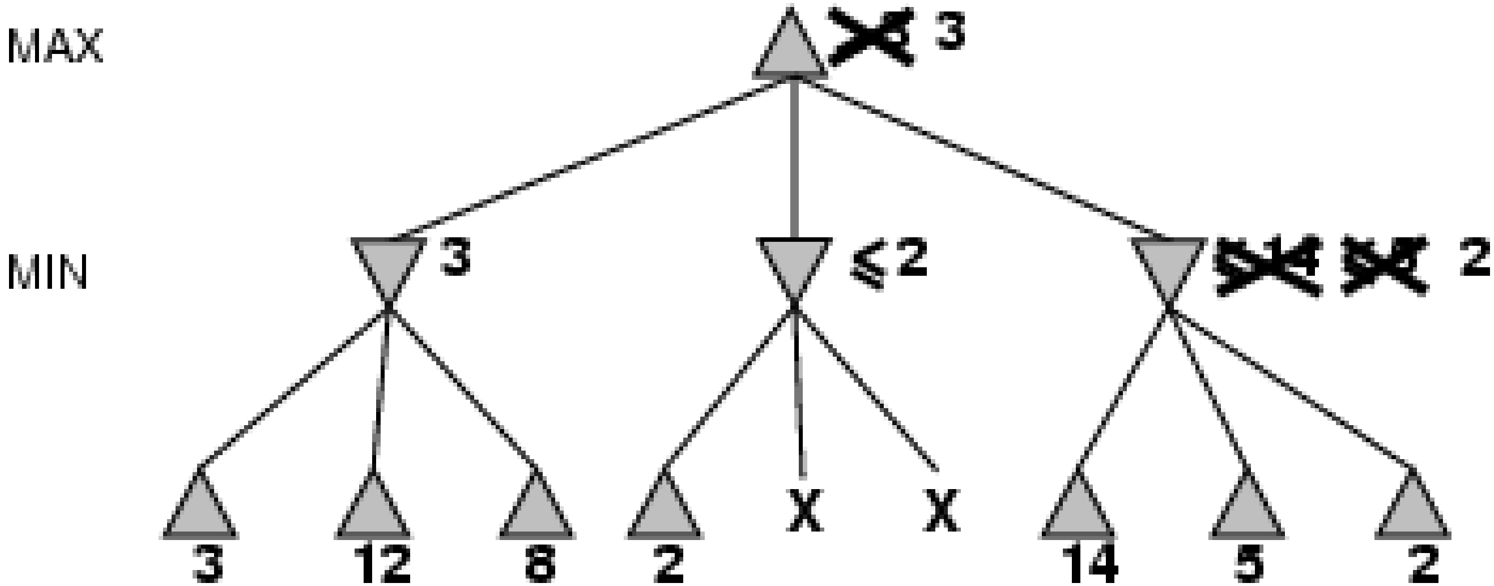
α-β Pruning Example (cont'd)



α-β Pruning Example (cont’d)



α-β Pruning Example (cont’d)



Alpha-Beta Pruning ($\alpha\beta$ prune)

Rules of Thumb

- α is the best (highest) found so far along the path for Max
- β is the best (lowest) found so far along the path for Min
- Search below a MIN node may be alpha-pruned if the its $\beta \leq \alpha$ of some MAX ancestor
- Search below a MAX node may be **beta-pruned** if the its $\alpha \geq \beta$ of some MIN ancestor.

Alpha-Beta Pruning Example

1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.

Alpha-Beta Pruning Example

1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.

Alpha-Beta Pruning Example

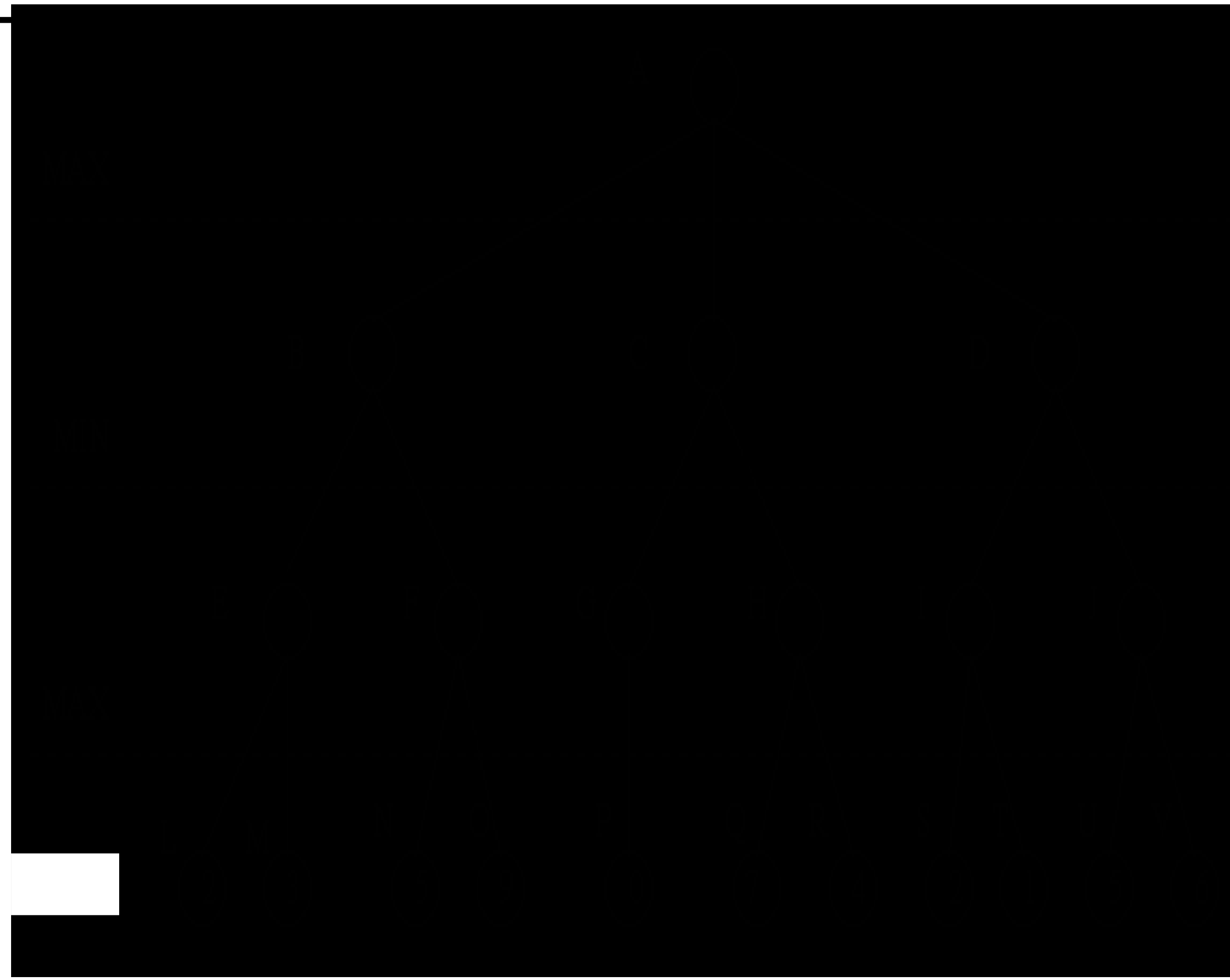
1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.

Alpha-Beta Pruning Example

1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

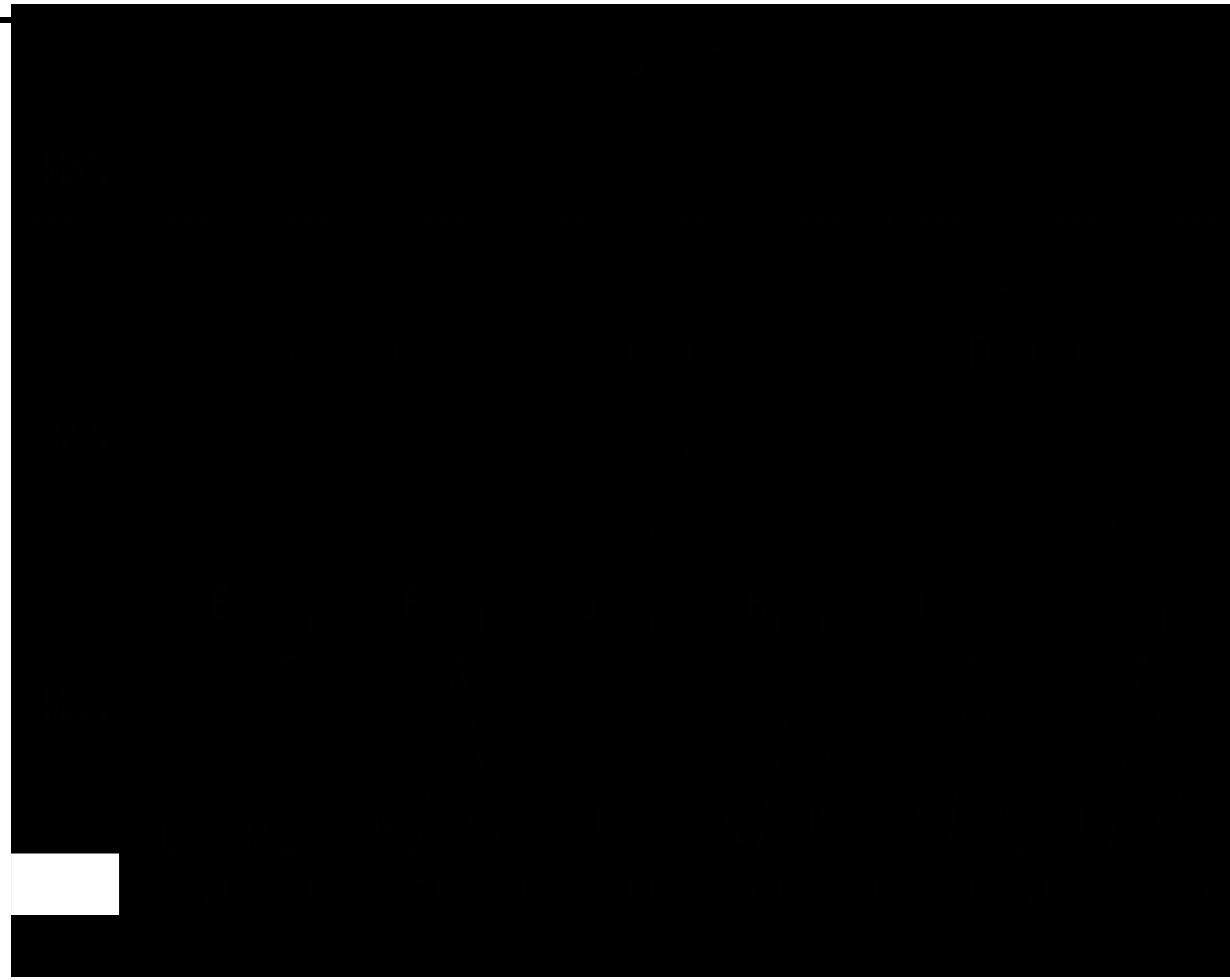
2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.



Alpha-Beta Pruning Example

1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.



α-β Search Algorithm

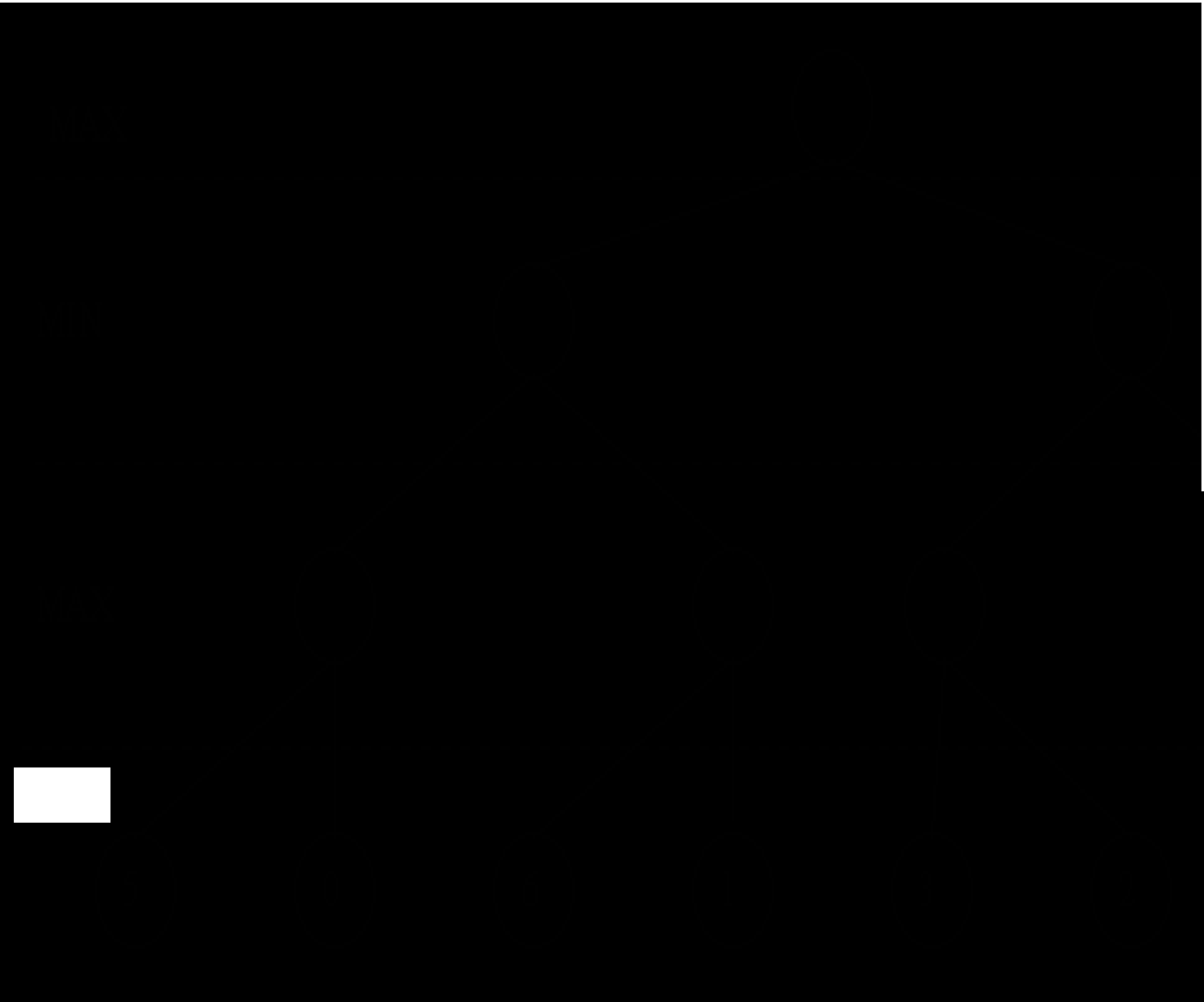
1. If terminal state, compute e(n) and return the result.
2. Otherwise, if the level is a **minimizing** level,
    - Until no more children or $\beta \le \alpha$
        - $v_i \leftarrow \alpha$ - β search on a child
        - If $v_i < \beta, \beta \leftarrow v_i$.
    - Return *min* (v_i)
3. Otherwise, the level is a **maximizing** level:
    - Until no more children or $\alpha \ge \beta$,
        - $v_i \leftarrow \alpha$ - β search on a child.
        - If $v_i > \alpha$, set $\alpha \leftarrow v_i$
    - Return $max(v_i)$

pruning

pruning

See page 170 R&N

Another Example



1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.

Example

1. Search below a MIN node may be alpha-pruned if the beta value is <= to the alpha value of some MAX ancestor.

2. Search below a MAX node may be beta-pruned if the alpha value is >= to the beta value of some MIN ancestor.

Properties of α-β Prune

Pruning does not affect final result

Good move ordering improves effectiveness of pruning b(e.g., chess, try captures first, then threats, froward moves, then backward moves…)

With "perfect ordering," time complexity = $O(b^{m/2})$

→ doubles depth of search that alpha-beta pruning can explore

Example of the value of reasoning about which computations are relevant (a form of metareasoning)

Resource limits

Suppose we have 100 secs, explore 10^4 nodes/sec

→ 10^6 nodes per move

Standard approach:

evaluation function

= estimated desirability of position

cutoff test:

e.g., depth limit

What is the problem with that?

→add quiescence search:

→quiescent position: position where

– next move unlikely to cause large change in players' positions

Cutoff Search

Suppose we have 100 secs, explore 10^4 nodes/sec
→ 10^6 nodes per move

Does it work in practice?
$b^m = 10^6$, $b=35$ → $m=4$

4-ply lookahead is a hopeless chess player!
- 4-ply ≈ human novice
- 8-ply ≈ typical PC, human master
- 12-ply ≈ Deep Blue, Kasparov

Other improvements…

Evaluation Function

Evaluation function

- Performed at search cutoff point
- Must have same terminal/goal states as utility function
- Tradeoff between accuracy and time → reasonable complexity
- Accurate
  - Performance of game-playing system dependent on accuracy/goodness of evaluation
  - Evaluation of nonterminal states strongly correlated with actual chances of winning

Evaluation functions

For chess, typically linear weighted sum of features

$$Eval(s) = w_1\, f_1(s) + w_2\, f_2(s) + \ldots + w_n\, f_n(s)$$

e.g., $w_1 = 9$ with

$f_1(s)$ = (number of white queens) – (number of black queens), etc.

Key challenge – find a good evaluation function:

Isolated pawns are bad.

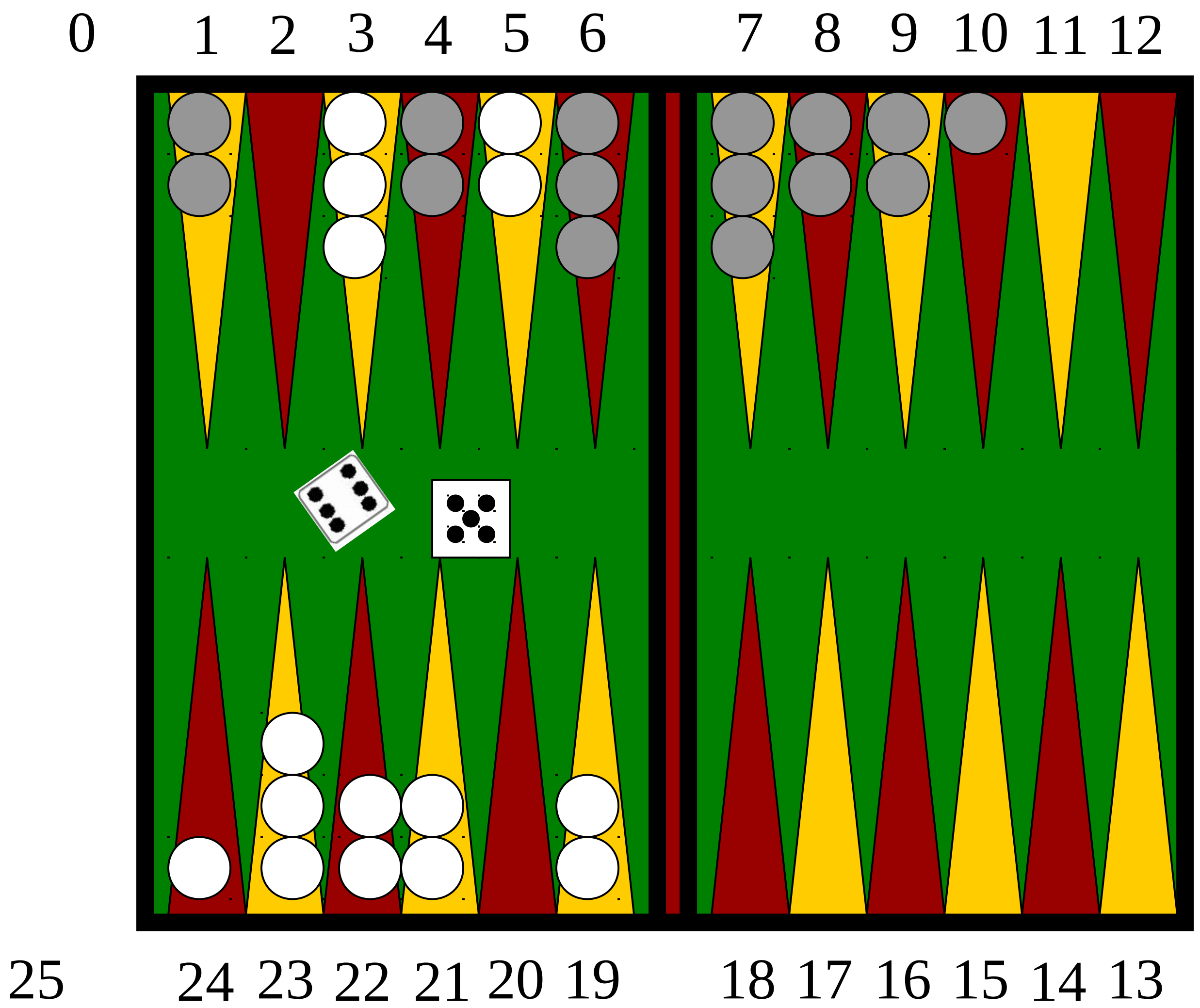
How well protected is your king?

How much maneuverability to you have?

Do you control the center of the board?

Strategies change as the game proceeds

When Chance is involved:
Backgammon Board



Expectiminimax

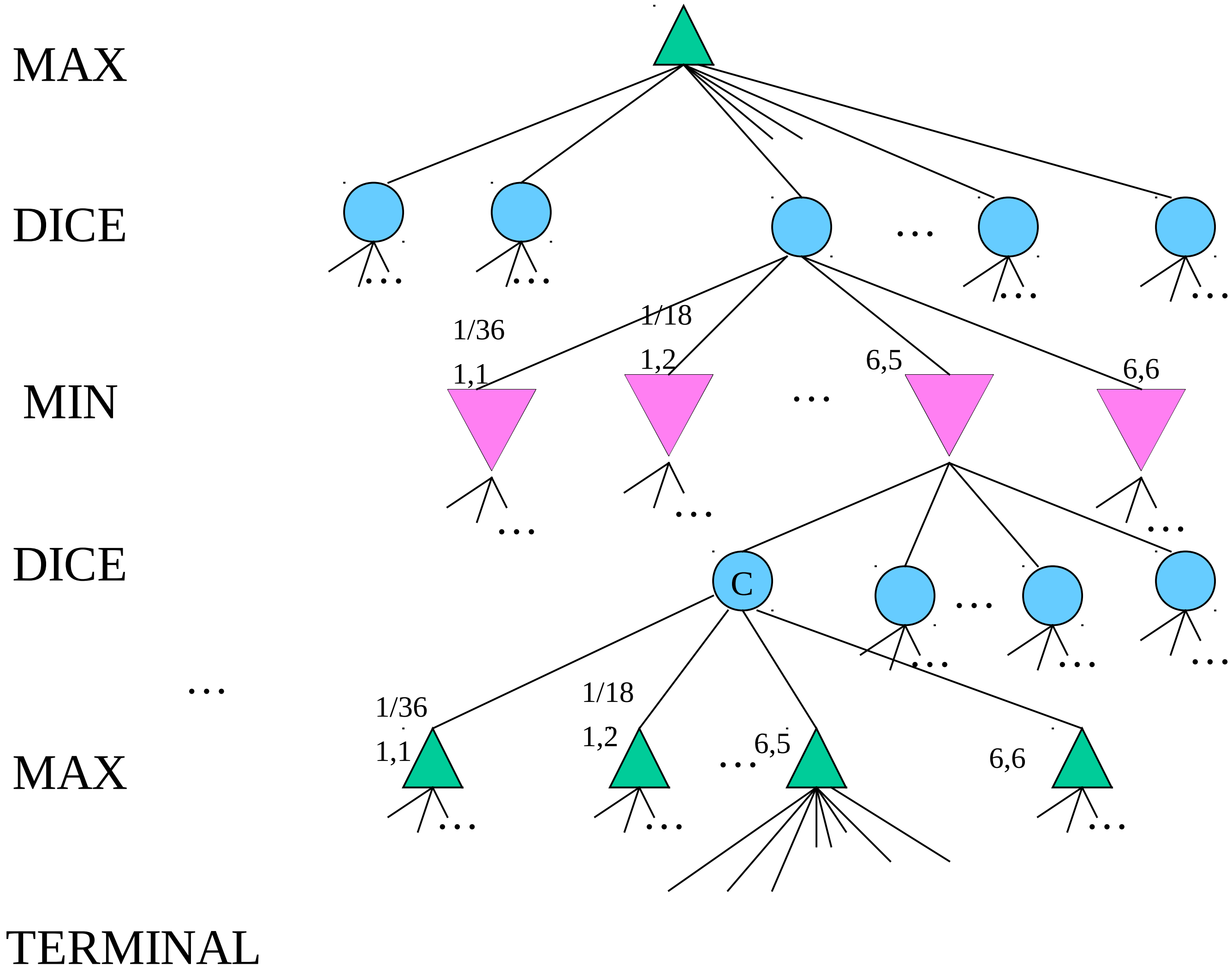
Generalization of minimax for games with chance nodes

Examples: Backgammon, bridge

Calculates expected value where probability is taken over all possible dice rolls/chance events

- Max and Min nodes determined as before
- Chance nodes evaluated as weighted average

Game Tree for Backgammon

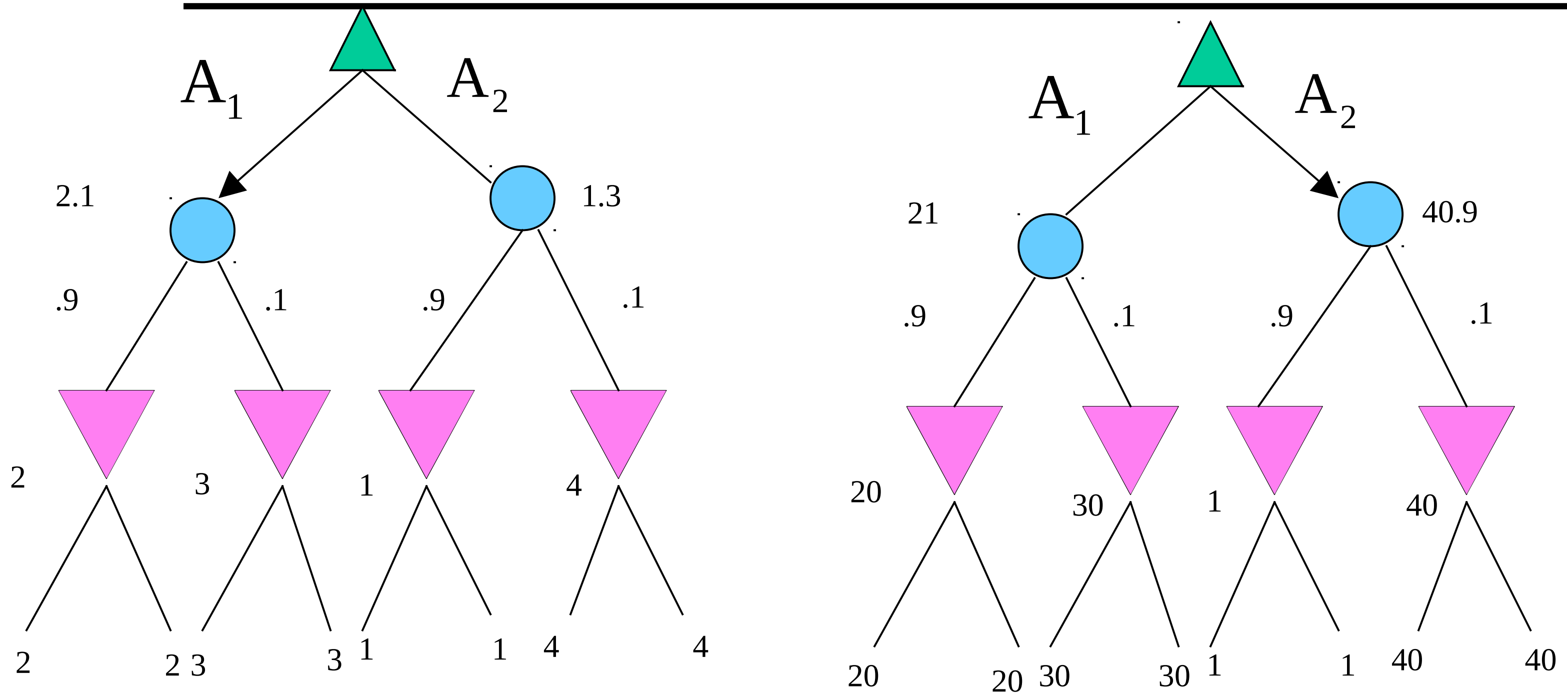


Expectiminimax

Expectiminimax(n) =

Utility(n)	for n, a terminal state
$max_{s \in Succ(n)}\ \text{expectiminimax}(\boldsymbol{s})$	for n, a Max node
$min_{s \in Succ(n)}\ \text{expectiminimax}(\boldsymbol{s})$	for n, a Min node
$\Sigma_{\boldsymbol{s} \in \boldsymbol{Succ}(\boldsymbol{n})} \boldsymbol{P}(\boldsymbol{s}) * \text{expectiminimax}(\boldsymbol{s})$	for n, a chance node

Expectiminimax



Chess: Case Study

Combinatorics of Chess

- Opening book
- Endgame
  - database of all 5 piece endgames exists; database of all 6 piece games being built
- Middle game
  - Positions evaluated (estimation)
    - 1 move by each player = 1,000
    - 2 moves by each player = 1,000,000
    - 3 moves by each player = 1,000,000,000

Positions with Smart Pruning

Search Depth		Positions
2		60
4		2,000
6		60,000
8		2,000,000
10	(<1 second DB)	60,000,000
12		2,000,000,000
14	(5 minutes DB)	60,000,000,000
16		2,000,000,000,000

How many lines of play does a grand master consider?

Around 5 to 7

Formal Complexity of Chess

How hard is chess?

- Obvious problem: standard complexity theory tells us nothing about finite games!
- Generalizing chess to NxN board: optimal play is PSPACE-hard

Game Tree Search

How to search a game tree was independently invented by Shannon (1950) and Turing (1951).

Technique called: MiniMax search.

Evaluation function combines material & position.

- Pruning "bad" nodes: doesn't work in practice
- Extend "unstable" nodes (e.g. after captures): works well in practice (Selection extension)

History of Search Innovations

Shannon, Turing	Minimax search	1950
Kotok/McCarthy	Alpha-beta pruning	1966
MacHack	Transposition tables	1967
Chess 3.0+	Iterative-deepening	1975
Belle	Special hardware	1978
Cray Blitz	Parallel search	1983
Hitech	Parallel evaluation	1985
Deep Blue	ALL OF THE ABOVE	1997

Evaluation Functions

Primary way knowledge of chess is encoded

- material
- position
  - doubled pawns
  - how constrained position is

Must execute quickly - constant time

- parallel evaluation: allows more complex functions
  - tactics: patterns to recognize weak positions
  - arbitrarily complicated domain knowledge

Learning better evaluation functions

- Deep Blue learns by tuning weights in its board evaluation function

  » $f(p) = w_1 f_1(p) + w_2 f_2(p) + \ldots + w_n f_n(p)$

- Tune weights to find best least-squares fit with respect to moves actually chosen by grandmasters in 1000+ games.
- The key difference between 1996 and 1997 match!
- Note that Kasparov also trained on

  "computer chess" play.

Transposition Tables

Introduced by Greenblat's Mac Hack (1966)

Basic idea: caching

- once a board is evaluated, save in a hash table, avoid re-evaluating.
- called “transposition” tables, because different orderings (transpositions) of the same set of moves can lead to the same board.

Transposition Tables as Learning

Is a form of root learning (memorization).

- positions generalize sequences of moves
- learning on-the-fly
- don't repeat blunders: can't beat the computer twice in a row using same moves!

Deep Blue --- huge transposition tables (100,000,000+), must be carefully managed.

Special-Purpose and Parallel Hardware

Belle (Thompson 1978)

Cray Blitz (1993)

Hitech (1985)

Deep Blue (1987-1996)

- Parallel evaluation: allows more complicated evaluation functions
- Hardest part: coordinating parallel search
- Deep Blue never quite plays the same game, because of “noise” in its hardware!

Deep Blue

Hardware

- 32 general processors
- 220 VSLI chess chips

Overall: 200,000,000 positions per second

- 5 minutes = depth 14

Selective extensions - search deeper at unstable positions

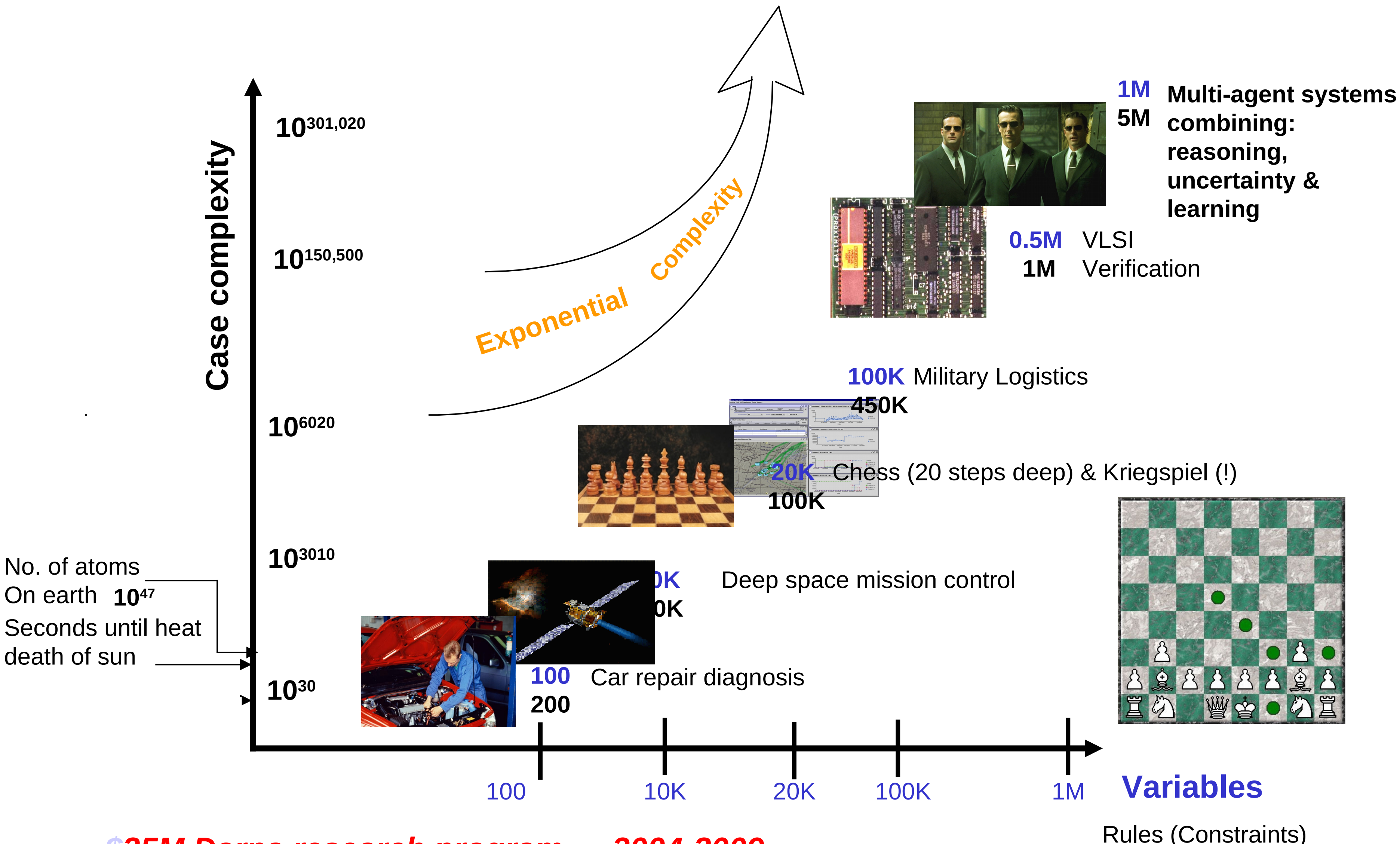
- down to depth 25 !

Tactics into Strategy

As Deep Blue goes deeper and deeper into a position, it displays elements of strategic understanding. Somewhere out there mere tactics translate into strategy. This is the closet thing I've ever seen to computer intelligence. It's a very weird form of intelligence, but you can feel it. It feels like thinking.

– Frederick Friedel (grandmaster), Newsday, May 9, 1997

Automated reasoning --- the path



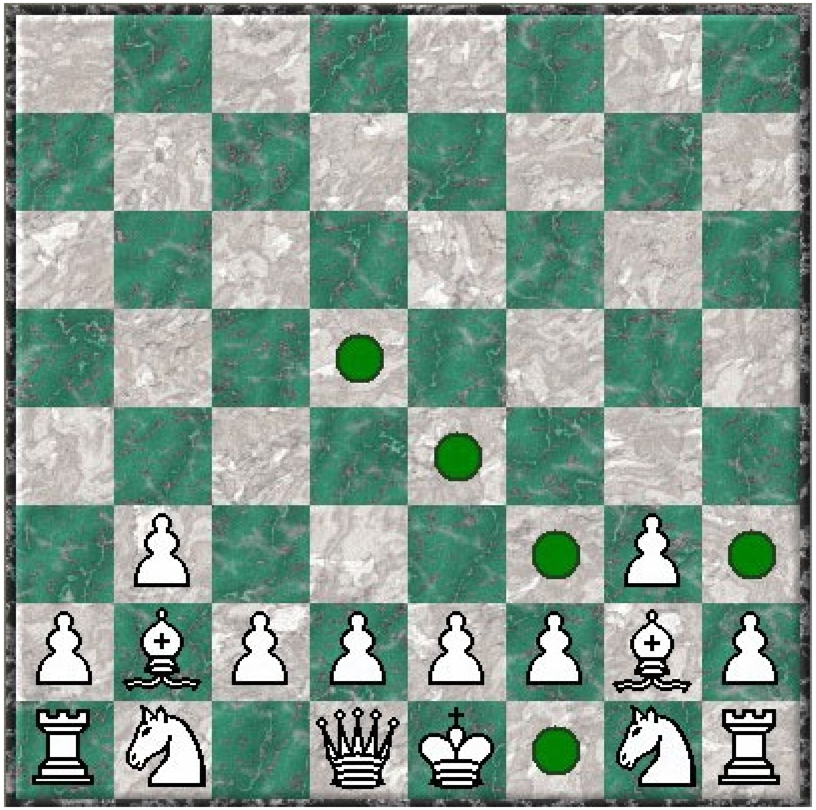
$25M Darpa research program --- 2004-2009

Kriegspiel

Pieces hidden from opponent

Interesting combination of reasoning, game tree search, and uncertainty.

Another chess variant: Multiplayer asynchronous chess.



The Danger of Introspection

When people express the opinion that human grandmasters do not examine 200,000,000 move sequences per second, I ask them, ``How do you know?'' The answer is usually that human grandmasters are not aware of searching this number of positions, or are aware of searching many fewer. But almost everything that goes on in our minds we are unaware of.

– Drew McDermott

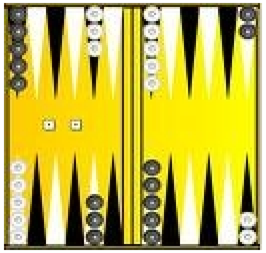
State-of-the-art of other games

Deterministic games in practice

Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a pre-computed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.

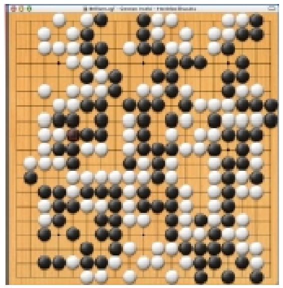
2007: proved to be a draw! Schaeffer et al. solved checkers for "White Doctor" opening (draw) (about 50 other openings).

Othello: human champions refuse to compete against computers, who are too good



Backgamon: TD-Gamon is competitive with World Champion (ranked among the top 3 players in the world). Tesauro's approach (1992) used learning to come up with a good evaluation function. Exciting application of reinforcement learning.

Pl... GO



Go: human champions r... against computers, who ar... b > 300, so most programs use p... bases to suggest plausible mo...

“UCT may prove useful for targeting advertisements on the Web, finding the best settings for an industrial plant or optimizing channel allocation in cellular systems.” – Scientific American

Not true!

Computer Beats Pro at U.S. Go Congress

http://www.usgo.org/index.php?%23_id=4602

On August 7, 2008, the computer program MoGo runnin... ...00 cores) beat professional Go player Myungwan Kim (8p) in a ... on the 19x19 board. The handicap given to the comp... ...ones.

MoGo uses Monte Carlo based methods ... *upper confidence bounds applied*

Summary

Game systems rely heavily on

- Search techniques
- Heuristic functions
- Bounding and pruning technqiues
- Knowledge database on game

For AI, the abstract nature of games makes them an appealing subject for study:

state of the game is easy to represent;

agents are usually restricted to a small number of actions whose outcomes are defined by precise rules

Summary

Game playing was one of the first tasks … in AI as soon as computers became progra… , Turing, Shannon, Wiener tackled c…

Game playin… spawned a number of interesting research id… rch, data structures, databases, heuristics, evaluations …nctions and many areas of computer science.

Games are fun:

Teach your computer how to play a game!