Combinatorial Optimization

Types of Search Problems

- Shortest Path (SampleWorld, tile puzzle, driving directions)
  - given operators and costs
  - want least-cost path to goal
  - goal depth/cost is unknown
- Constraint Satisfaction (map coloring, n-queens)
  - any goal is fine
  - fixed depth
  - explicit constraints
- Combinatorial Optimization (TSP, set cover)
  - want least-cost goal
  - maximum depth = number of variables
  - every leaf is a solution

Hill Climbing

```
$Sol \leftarrow$ a random solution
for $limit$ runs do
    $New \leftarrow$ random neighbor of $Sol$
    if $New$ is better than $Sol$ then
        $Sol \leftarrow new$
```

Related Techniques

- best neighbor – gradient-descent
- random restarts
- simulated annealing
- population optimization – genetic algorithms

Adversarial Search

Search Twist

Decisions with an adversary:

- chess, tic-tac-toe, go
- adversary prevents some solution paths
- we want best assured outcome

Minimax

Each *ply* in search tree corresponds to half a *move*

Terminal states are labeled with value

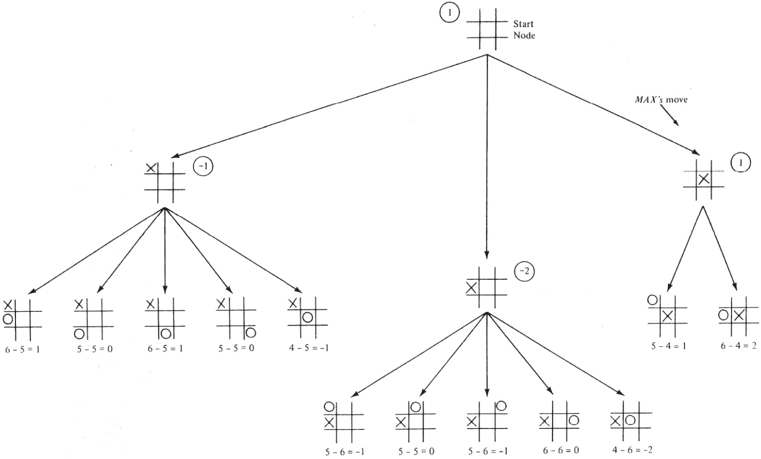
Instead of full solution tree, use a *static evaluation function* on non-terminal states

Evaluation for Tic-tac-toe

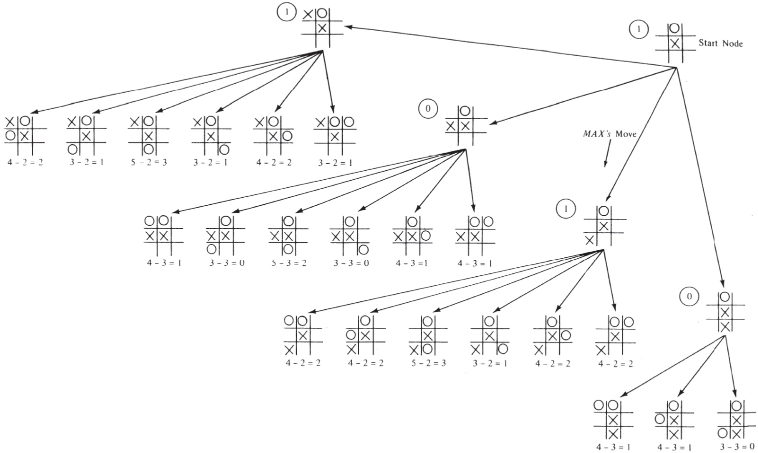
A *3-length* is a complete row, column, or diagonal

$$\begin{aligned} \text{value of position} &= \infty \text{ if win for me,} \\ \text{or} &= -\infty \text{ if win for you,} \\ \text{otherwise} &= \text{\# 3-lengths for me } - \\ &\quad\ \ \text{\# 3-lengths for you} \end{aligned}$$

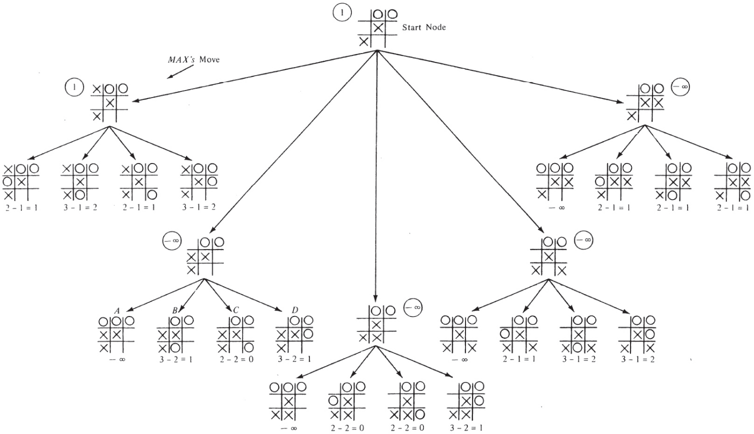
Two-Ply Search from Game Start



Two-Ply Search: Second Move



Two-Ply Search: Third Move


1
Start Node
MAX's Move
1
−∞
2 − 1 = 1
3 − 1 = 2
2 − 1 = 1
3 − 1 = 2
−∞
2 − 1 = 1
2 − 1 = 1
2 − 1 = 1
−∞
−∞
A
B
C
D
−∞
3 − 2 = 1
2 − 2 = 0
3 − 2 = 1
−∞
−∞
2 − 1 = 1
3 − 1 = 2
3 − 1 = 2
−∞
2 − 2 = 0
2 − 2 = 0
3 − 2 = 1