COOPERATIVE AND ADAPTIVE ALGORITHMS

WEEK 3

Otman A. Al-Basir, Spring 2016

INFORMED SEARCH

- Tries to use the knowledge known about the problem to prune the search.
- This knowledge is used in the form of a heuristic function.
- The heuristic function should give an estimate of the distance to the goal.

HEURISTIC

Merriam-Webster's Online Dictionary

Heuristic (pron. \hy*u*-ˈris-tik\): adj. [from Greek *heuriskein* to discover.] involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods

The Free On-line Dictionary of Computing (15Feb98)

heuristic 1. <programming> A rule of thumb, simplification or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee feasible solutions and are often used with no theoretical guarantee. 2. <algorithm> approximation algorithm.

From WordNet (r) 1.6

heuristic adj 1: (computer science) relating to or using a heuristic rule 2: of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]

INFORMED METHODS ADD DOMAIN-SPECIFIC INFORMATION

Add domain-specific information to select the best path along which to continue searching

Define a heuristic function ***h*(*n*)** that estimates the "goodness" of a node *n*.

- Specifically, *h*(*n*) = **estimated cost** (or distance) of minimal cost path from *n* **to a goal state**.

The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description.

HEURISTICS

All domain knowledge used in the search is encoded in the **heuristic function** ***h***().

Heuristic search is an example of a "**weak method**" because of the limited way that domain-specific information is used to solve the problem.

Examples:

- 8-puzzle: Number of tiles out of place
- 8-puzzle: Sum of distances each tile is from its goal position

In general:

- $h(n) \geq 0$ for all nodes n
- $h(n) = 0$ implies that n is a goal node
- $h(n) = \infty$ implies that n is a dead-end that can never lead to a goal

WEAK VS. STRONG METHODS

We use the term *weak methods* to refer to methods that are extremely *general* and not tailored to a specific situation.

Examples of weak methods include

- **Means-ends analysis** is a strategy in which we try to represent the current situation and where we want to end up and then look for ways to shrink the differences between the two.
- **Space splitting** is a strategy in which we try to list the possible solutions to a problem and then try to rule out classes of these possibilities.
- **Subgoaling** means to split a large problem into several smaller ones that can be solved one at a time.

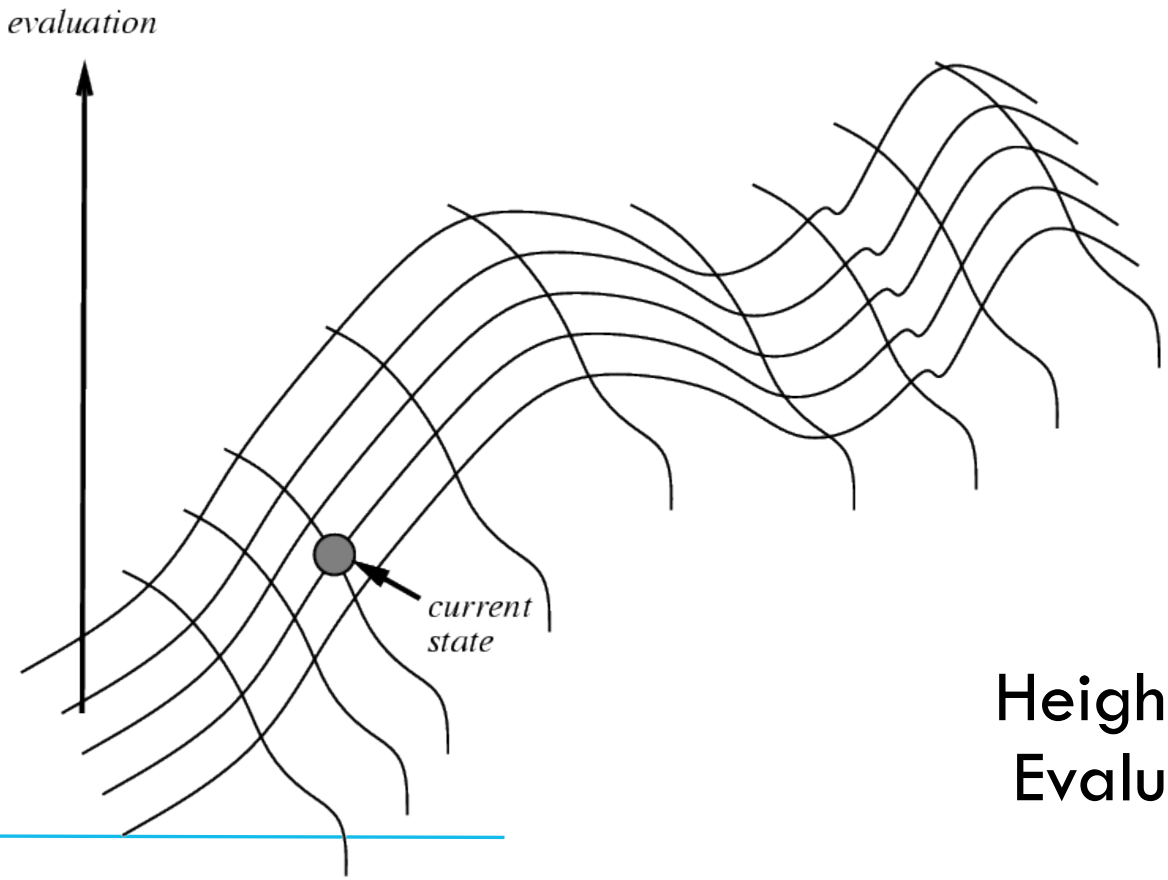
Called "weak" methods because they do not take advantage of more powerful domain-specific heuristics

INFORMED SEARCH

Hill climbing

- Tries to improve the efficiency of depth-first.
- Informed depth-first algorithm.
- It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution.
- It sorts the successors of a node (according to their heuristic values) before adding them to the list to be expanded.
- If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.
- For example, hill climbing can be applied to the travelling salesman problem. It is easy to find an initial solution that visits all the cities but will be very poor compared to the optimal solution. The algorithm starts with such a solution and makes small improvements to it, such as switching the order in which two cities are visited. Eventually, a much shorter route is likely to be obtained

HILL CLIMBING ON A SURFACE OF STATES



Height Defined by Evaluation Function

HILL-CLIMBING SEARCH

Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.

Rule:

If there exists a successor s for the current state n such that

- $h(s) < h(n)$ and
- $h(s) \leq h(t)$ for all the successors t of n,

then move from n to s. Otherwise, halt at n.

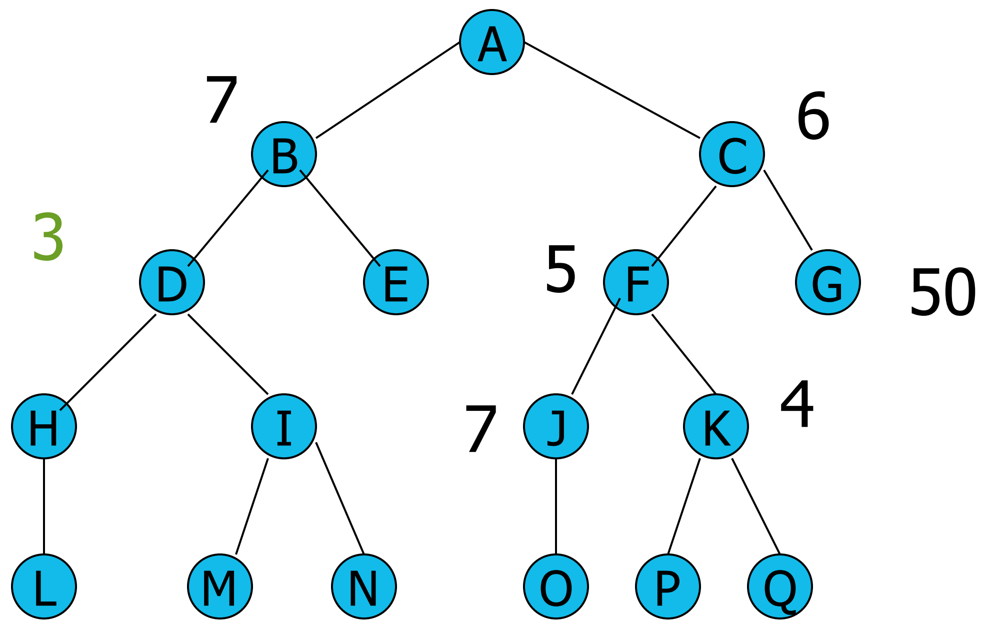
Similar to Greedy search in that it uses *h*(), but does not allow backtracking or jumping to an alternative path since it doesn't "remember" where it has been.

Corresponds to Beam search with a beam width of 1 (i.e., the maximum size of the nodes list is 1).

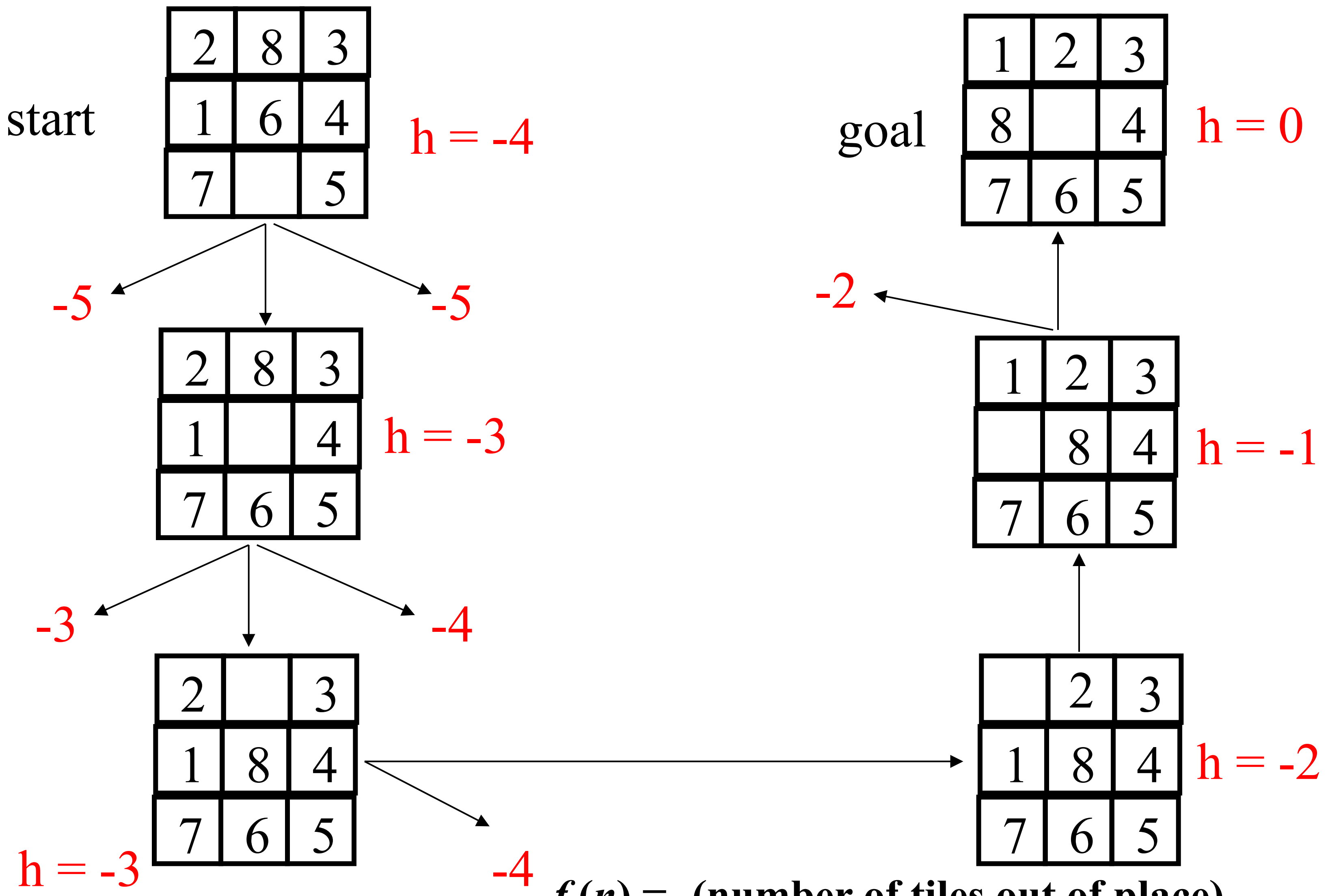
Not complete since the search will terminate at "local minima," “Like climbing Everest in thick fog with amnesia”

HILL CLIMBING

A,C,F, K



HILL CLIMBING EXAMPLE



***f* (*n*) = -(number of tiles out of place)**

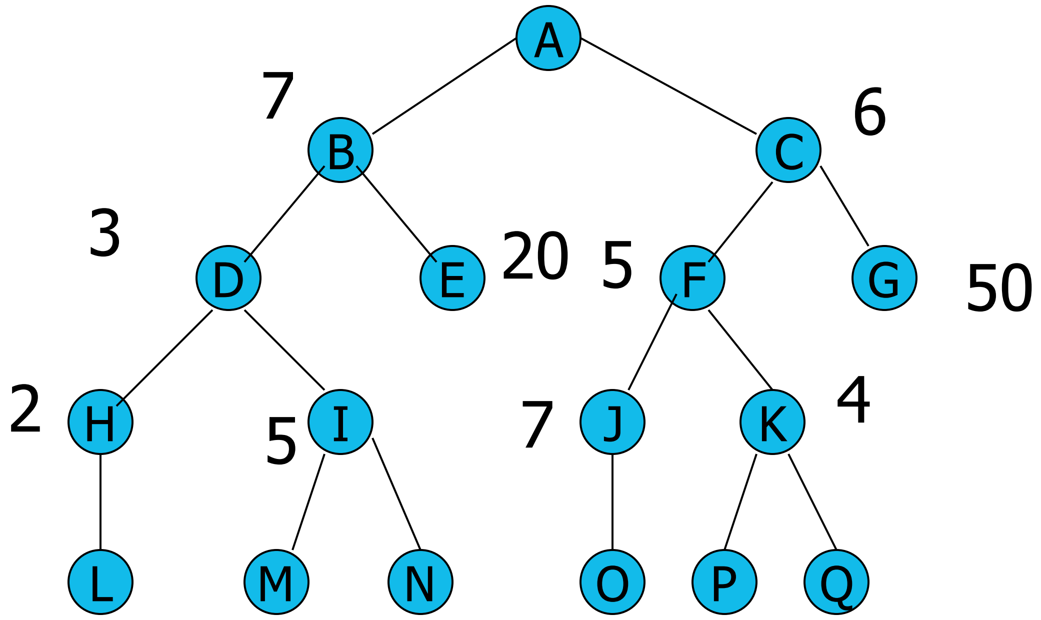
INFORMED SEARCH

Beam search

- Tries to minimize the memory requirement of the breadth-first algorithm.
- Informed breadth-first algorithm.
- Expands only the first *m* promising nodes at each level.

BEAM SEARCH

Beam, m=2
A,B,C,D,F, H, K



INFORMED SEARCH

Best-first

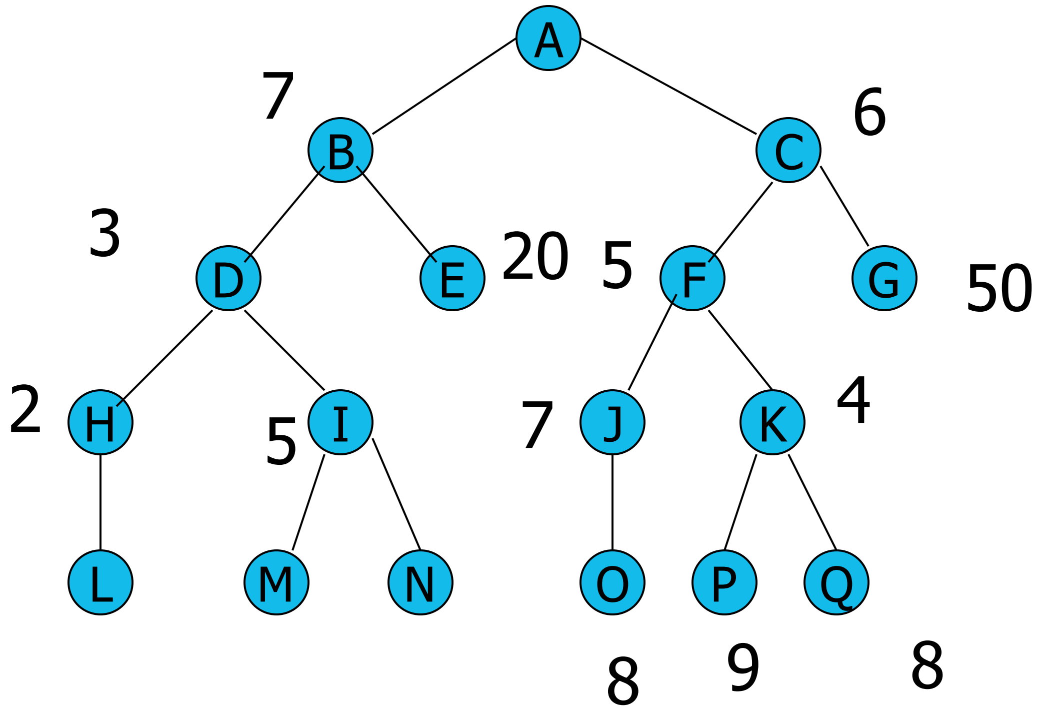
- Adds the successors of a node to the *expand* list.
- All nodes on the list are sorted according to the heuristic values.
- Expand most desirable unexpanded node
- Special Cases: Greedy search, A*

BEST FIRST SEARCH

BFS
A,C,F,K,J,B,D,H

Or

A,C,F,K,B,D,H



GREEDY SEARCH

Greedy search is best first that uses heuristic h(n) =estimate of cost from n to the closest goal node

Greedy search expands nodes that appear to be closest to goal

A*

A* tries to avoid expanding paths that are already expensive.

It uses an evaluation function $f(n)=g(n)+h(n)$

$g(n)$ is cost from start node to n

$h(n)$ is estimate cost from n to the goal

A* uses admissible heuristic $h(n)<= h^*(n)$ (the actual cost from n to the goal, and $h(n)>=0$)

A* is optimal

A heuristic function is *admissible* if it never overestimates the cost of reaching the goal. An admissible heuristic is also known as an *optimistic heuristic*

8-PUZZLE PROBLEM

5		8
4	2	1
7	3	6

n

1	2	3
4	5	6
7	8	

goal

$h_1(n)$ = number of misplaced tiles = 6 is admissible

Heuristic $h(n)$ is *admissible* if:

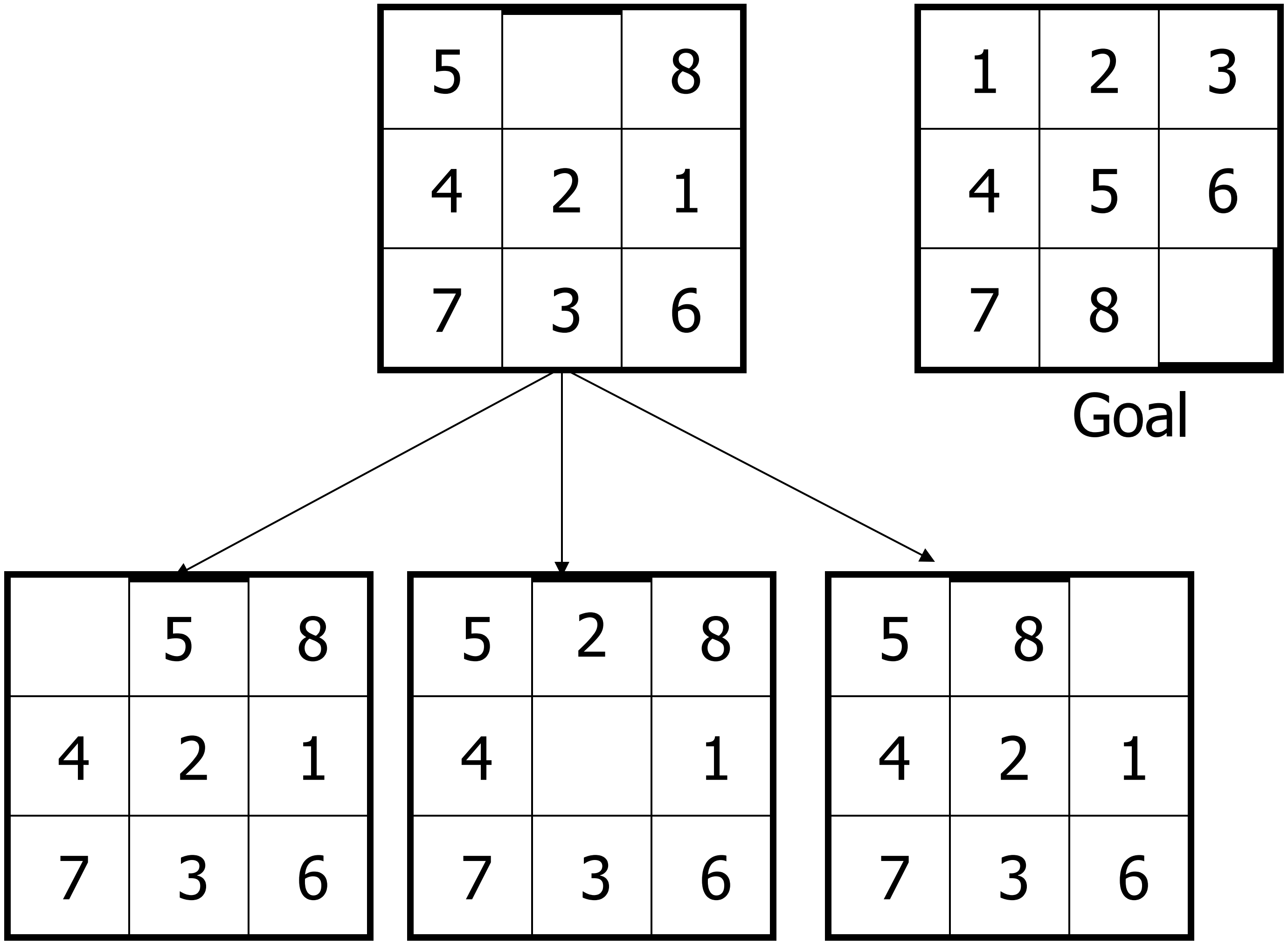
$$0 \leq h(n) \leq h^*(n)$$

An admissible heuristic is always optimistic

$h_2(n)$ = sum of distances of each tile to goal = 13 is admissible

Let h_1 and h_2 be two admissible and consistent heuristics such that for all nodes N: $h_1(n) \leq h_2(n)$.
Then, every node expanded by A* using h_2 is also expanded by A* using h_1.
h_2 is more informed than h_1

$h1(n) < h2(n)$ both are admissible, h2 is more informed than h1



h1=6
h2=12

h1=5
h2=12

h1=6
h2=12

HOW TO OBTAIN ADMISSIBLE HEURISTICS?

Can be derived by relaxing the problem

If the rules of the 8-puzzle are relaxed so a tile can move anywhere, then h1 gives the shortest cost to the goal.

If the rules are relaxed so the tile can move to any adjacent square, then h2 gives shortest cost.

GAME PLAYING AS SEARCH

In Games there is an opponent.

Need to search for best move and wait for the opponent response to search again from the new board configuration for best move.

Time is limited to find goal in each search

TYPES OF GAMES

Perfect information : each player has complete information about the opponent's position and available choices

- Deterministic: chess, checkers
- Chance: Backgammon, monopoly

Imperfect information and chance: poker, bridge

Two players or multiple players

TWO PLAYERS, PERFECT INFORMATION

Player and Opponent.

Take the player’s point of view

Ideally the player expands the game tree taking into consideration all the possible moves of the opponent till end of the game with leaf nodes as win, lose, draw.

MAX MIN STRATEGY

Player – MAX

Opponent – MIN

Minimax principle

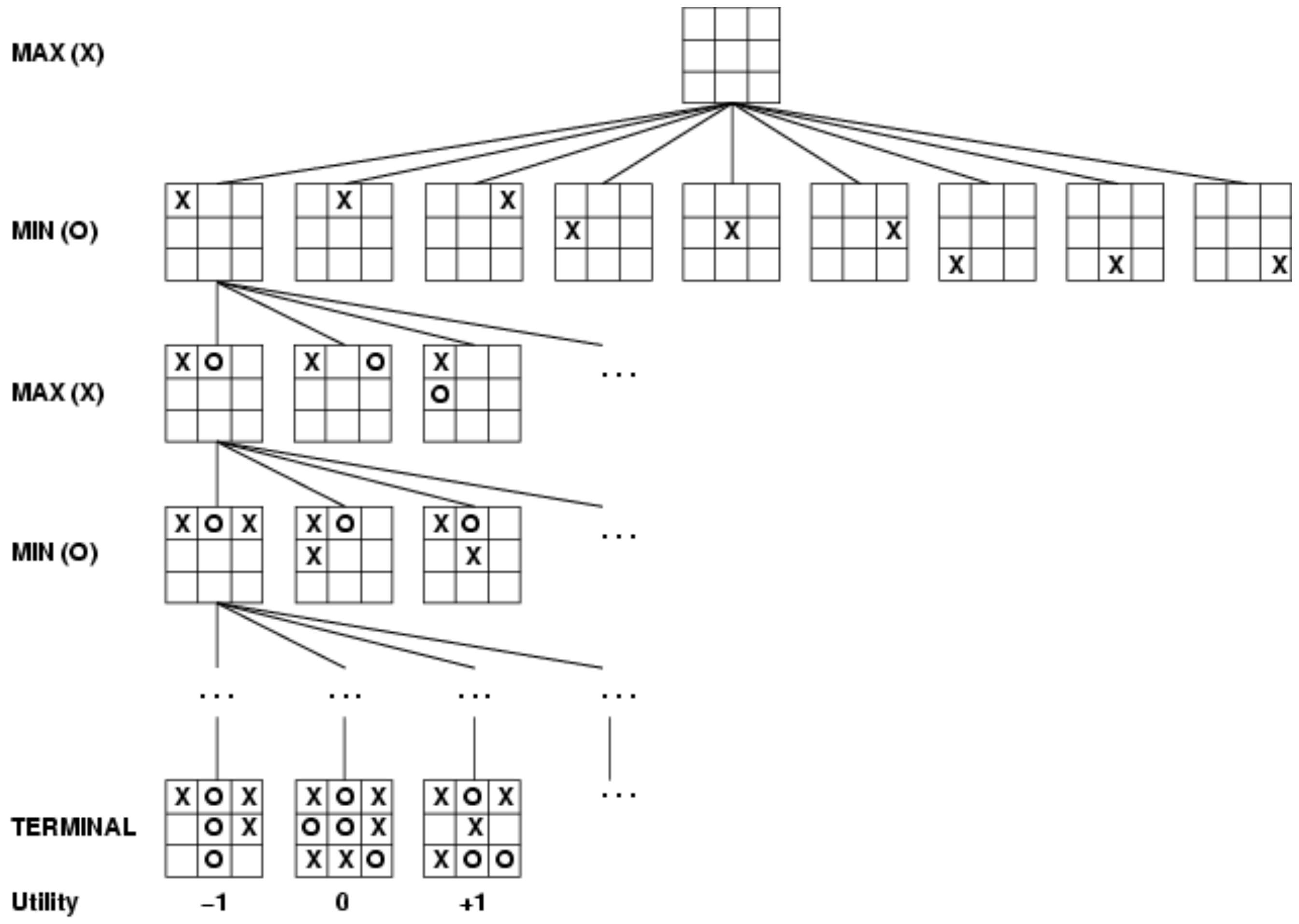
Label each level in Game Tree with **MAX** (player) and **MIN** (opponent)

Label leaves with evaluation of player

Go through the game tree

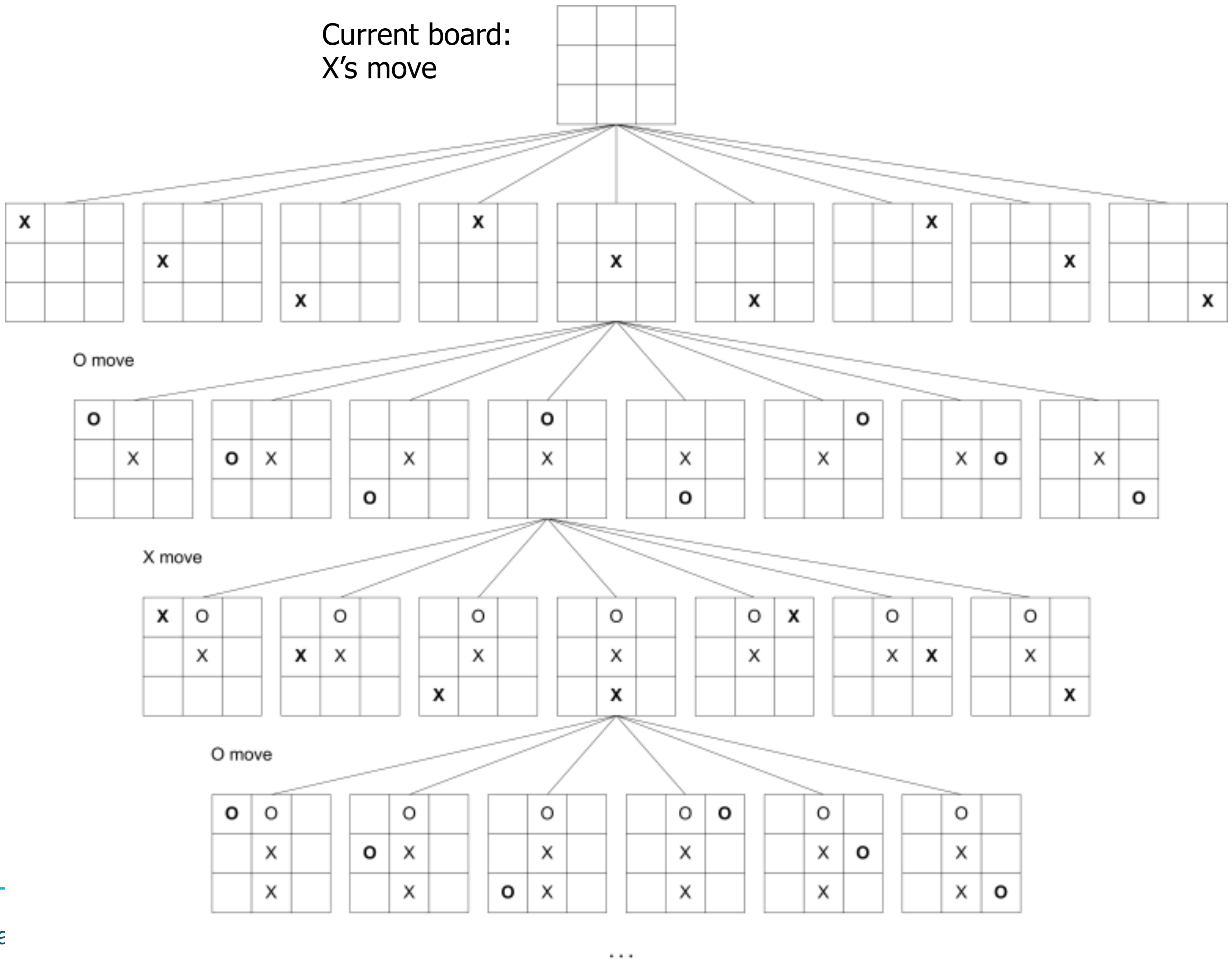
- if father node is **MAX** then label the node with the maximal value of its successors
- if father node is **MIN** then label the node with the minimal value of its successors

MAX MIN FOR TIC TAC TOE



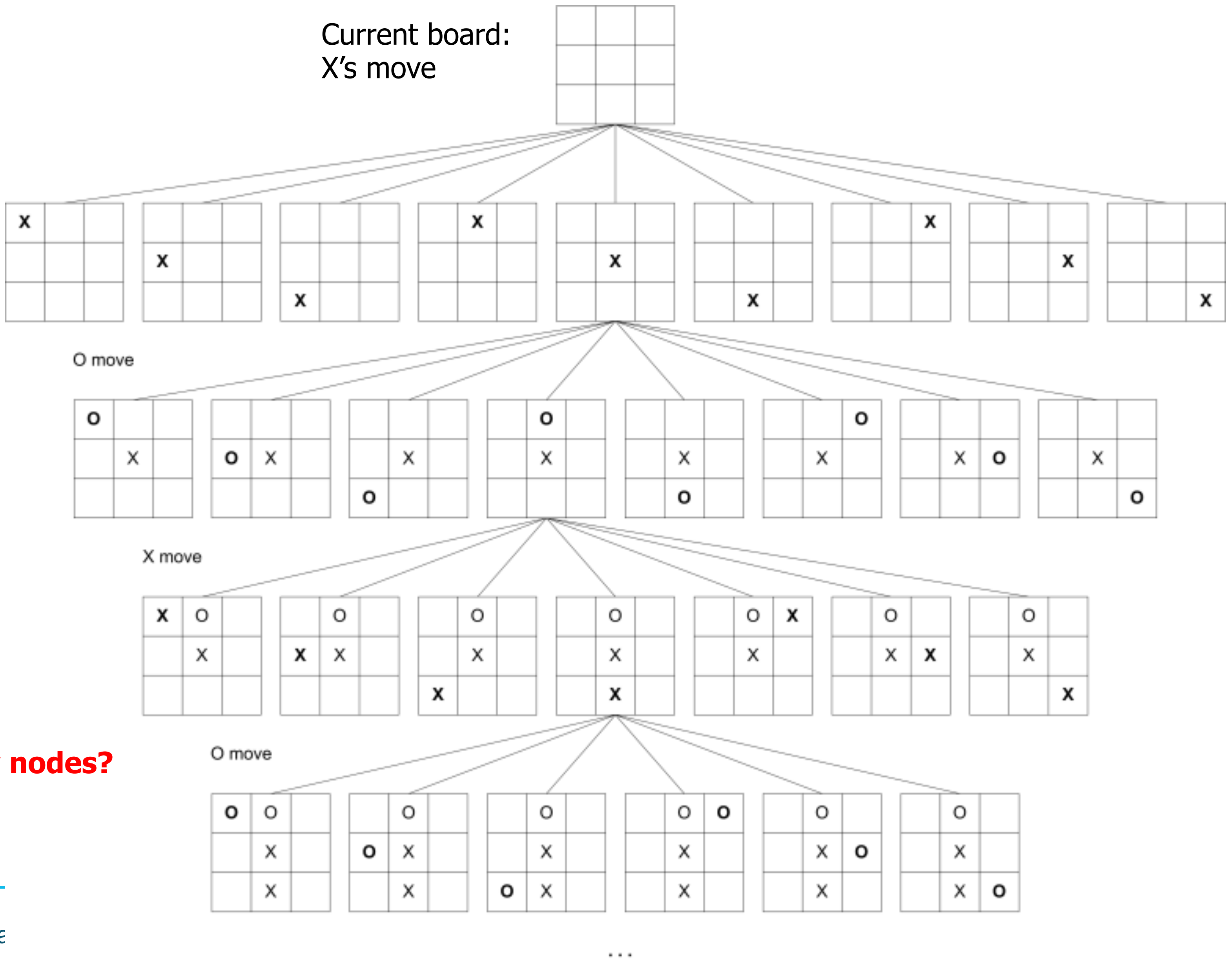
From [13]

GAME TREE
FROM M. T. JONES, *ARTIFICIAL INTELLIGENCE: A SYSTEMS APPROACH*



GAME TREE

FROM M. T. JONES, *ARTIFICIAL INTELLIGENCE: A SYSTEMS APPROACH*

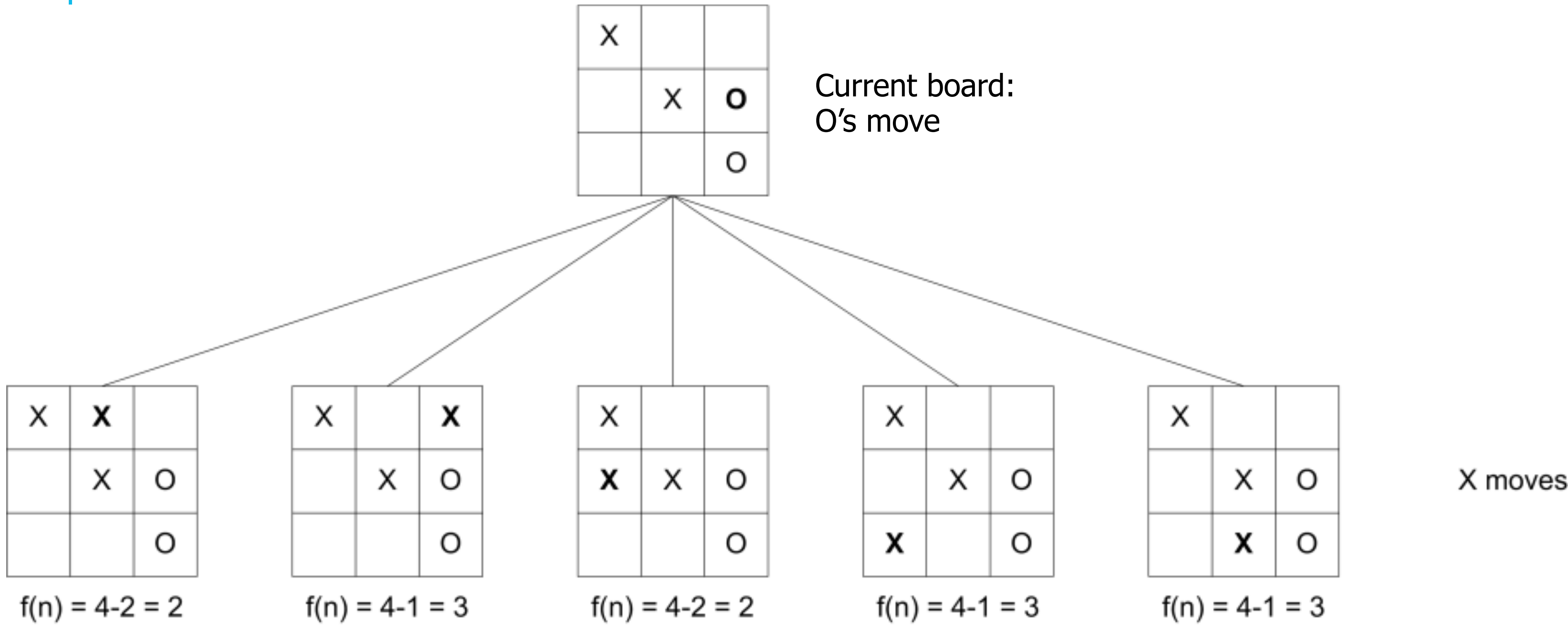


How many nodes?

LIMITED DEPTH AND EVALUATION FUNCTION

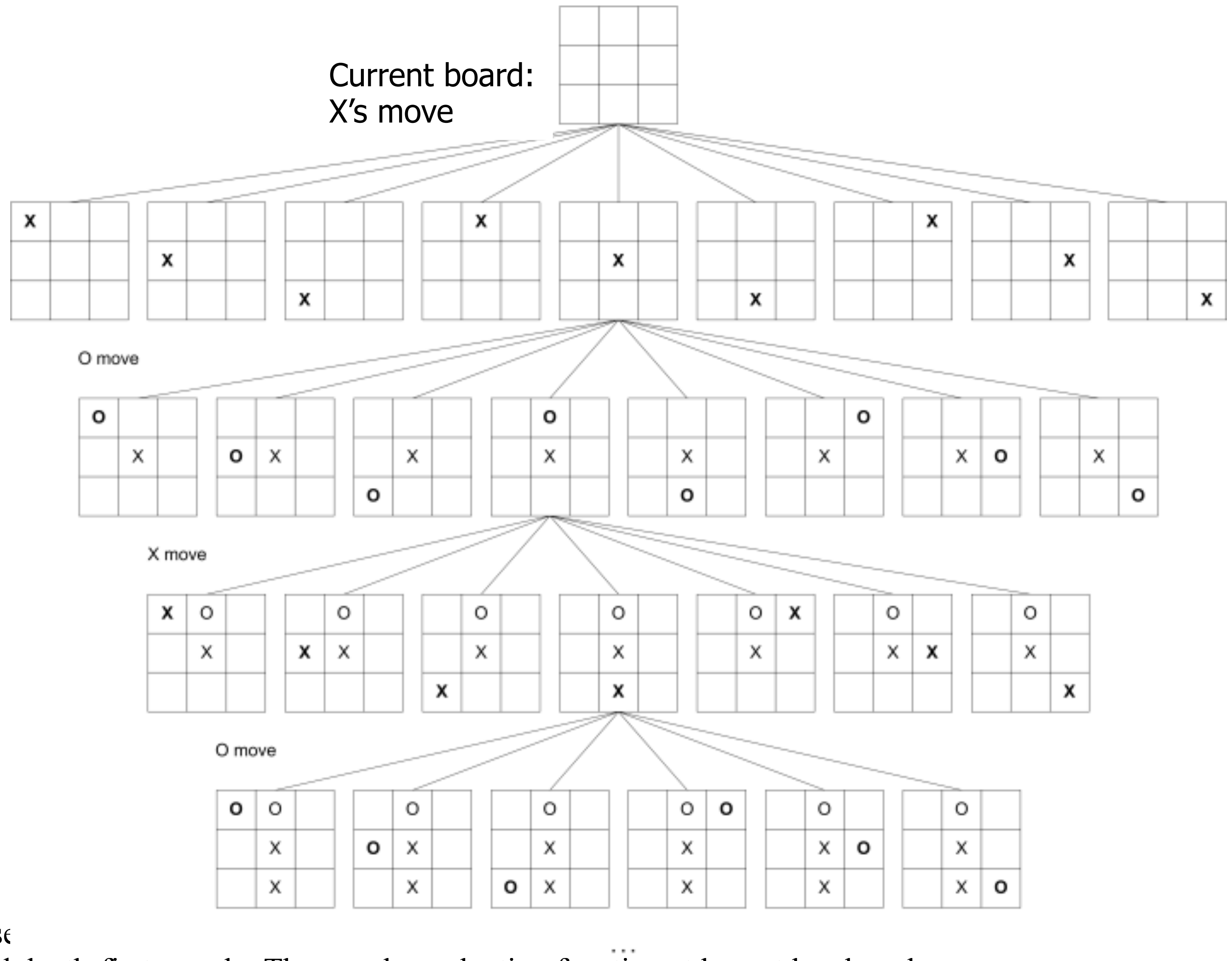
Expanding the complete game tree is only feasible for simple games. Tic Tac Toe has average branching factor *b* of *5* and average (moves) depth *d* of 9. Checkers has *b* =35 and *d* =*100*. Taking into consideration symmetrical and repeated states the game tree may be less than that. Still for checkers $10^{40.}$

For Chess it is only feasible to explore a limited depth of the game tree and to use a board evaluation function for the worth of this node to the player.



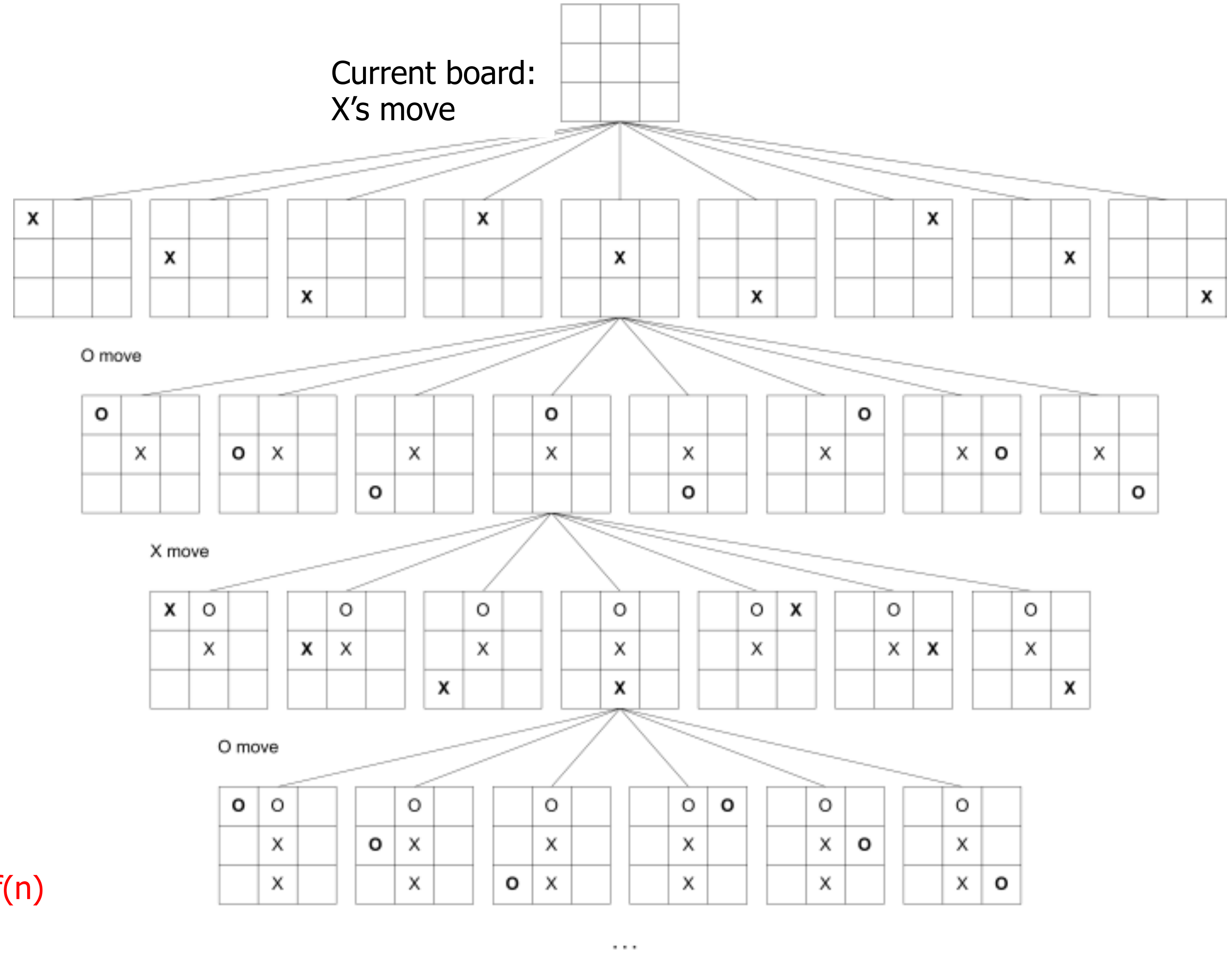
Evaluation function *f*(*n*) measures "goodness" of board configuration *n*. Assumed to be better estimate as search is deepened (i.e., at lower levels of game tree).

Evaluation function here: "Number of possible wins (rows, columns, diagonals) not blocked by opponent, minus number of possible wins for opponent not blocked by current player."



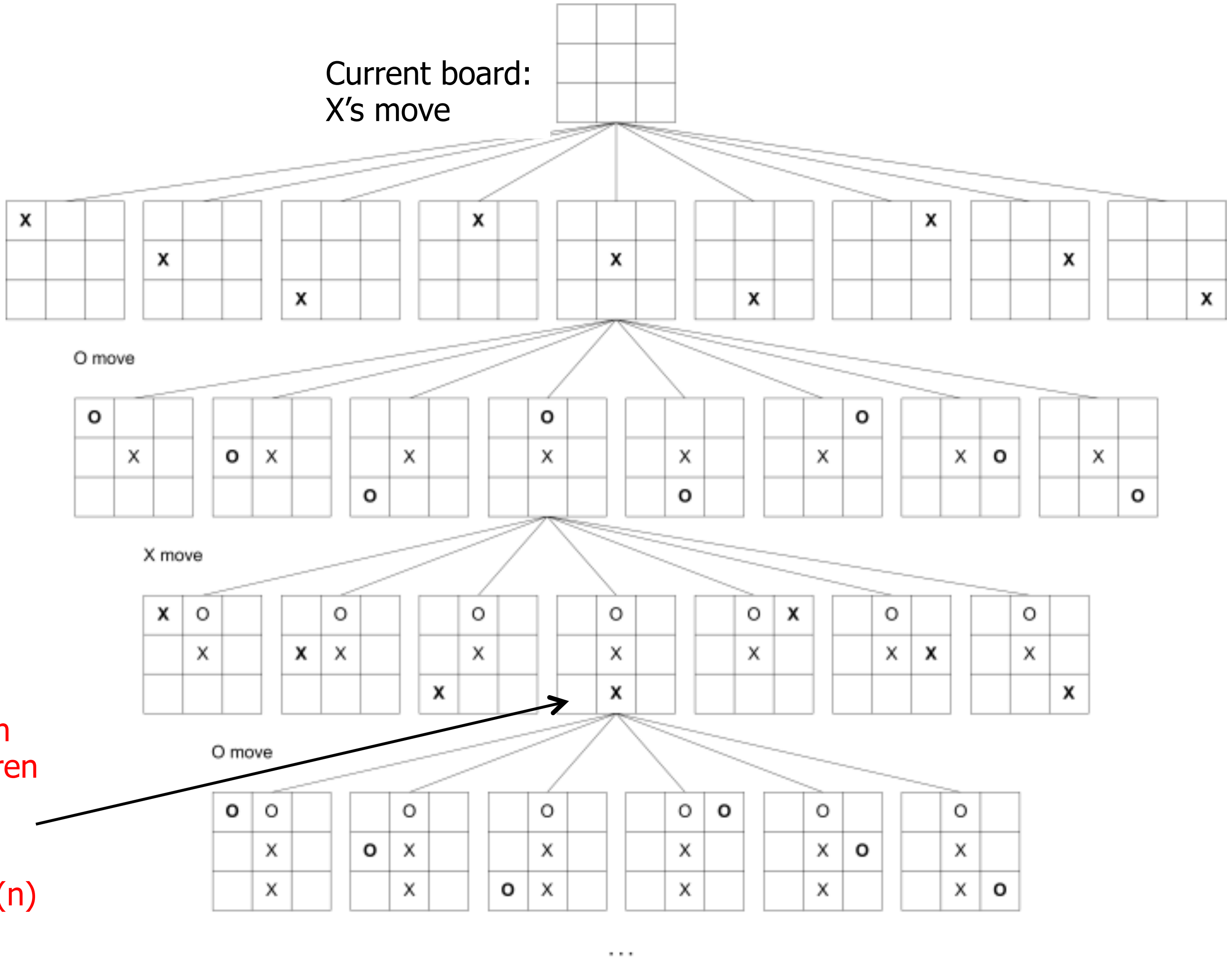
Minimax se
in a limited depth-first search. Then apply evaluation function at lowest level, and propagate results back up the tree.

Current board:
X's move



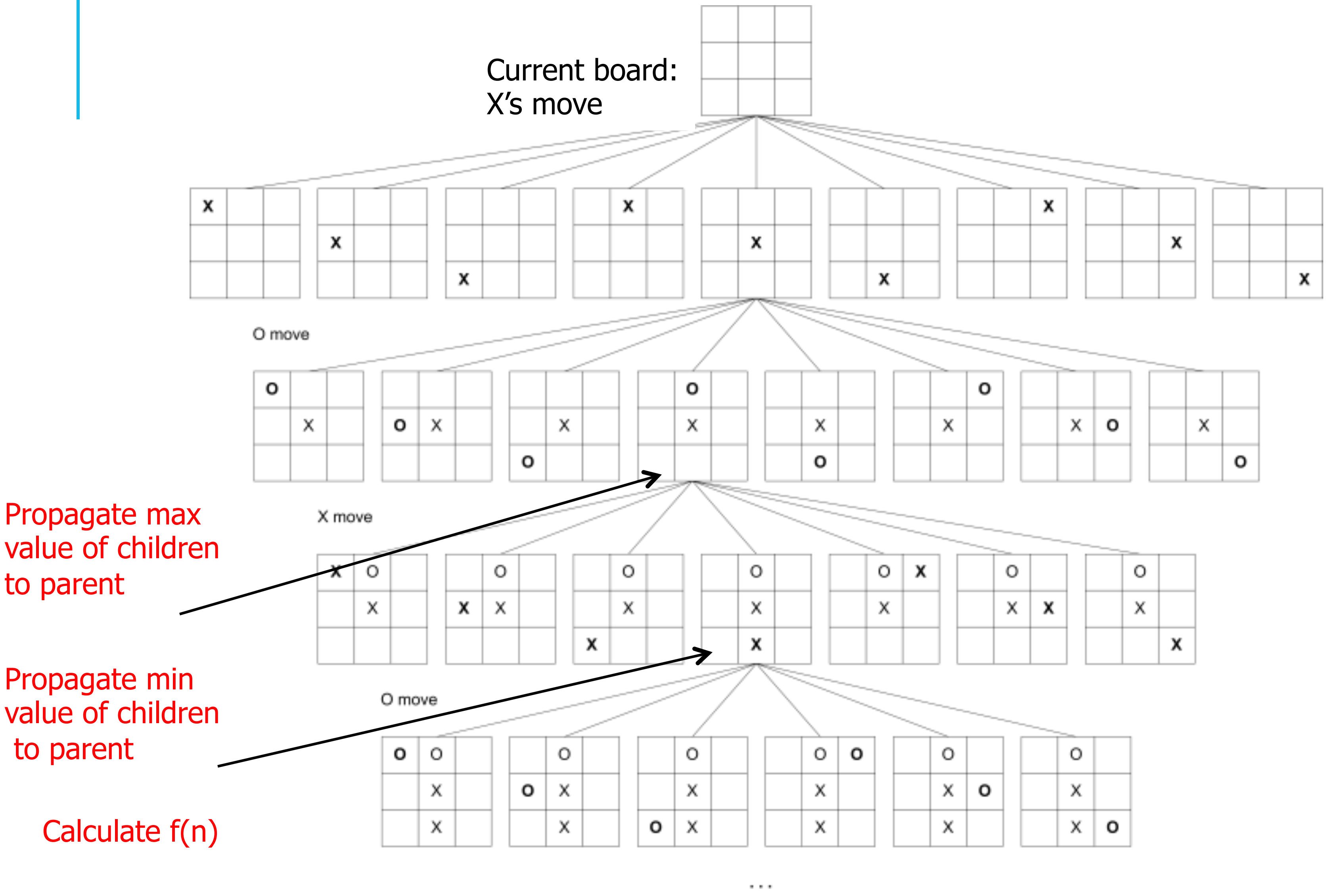
Calculate f(n)

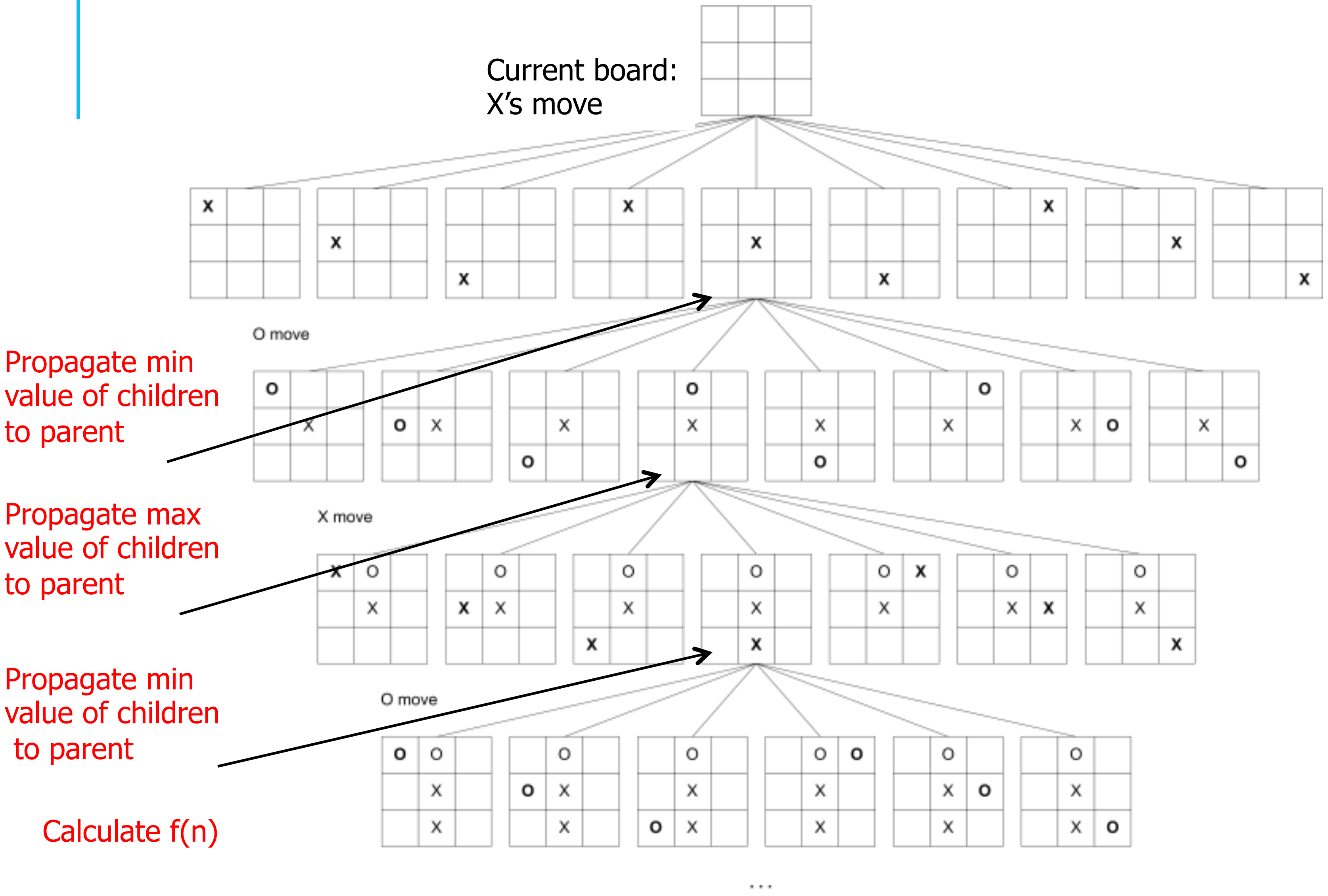
Current board:
X's move

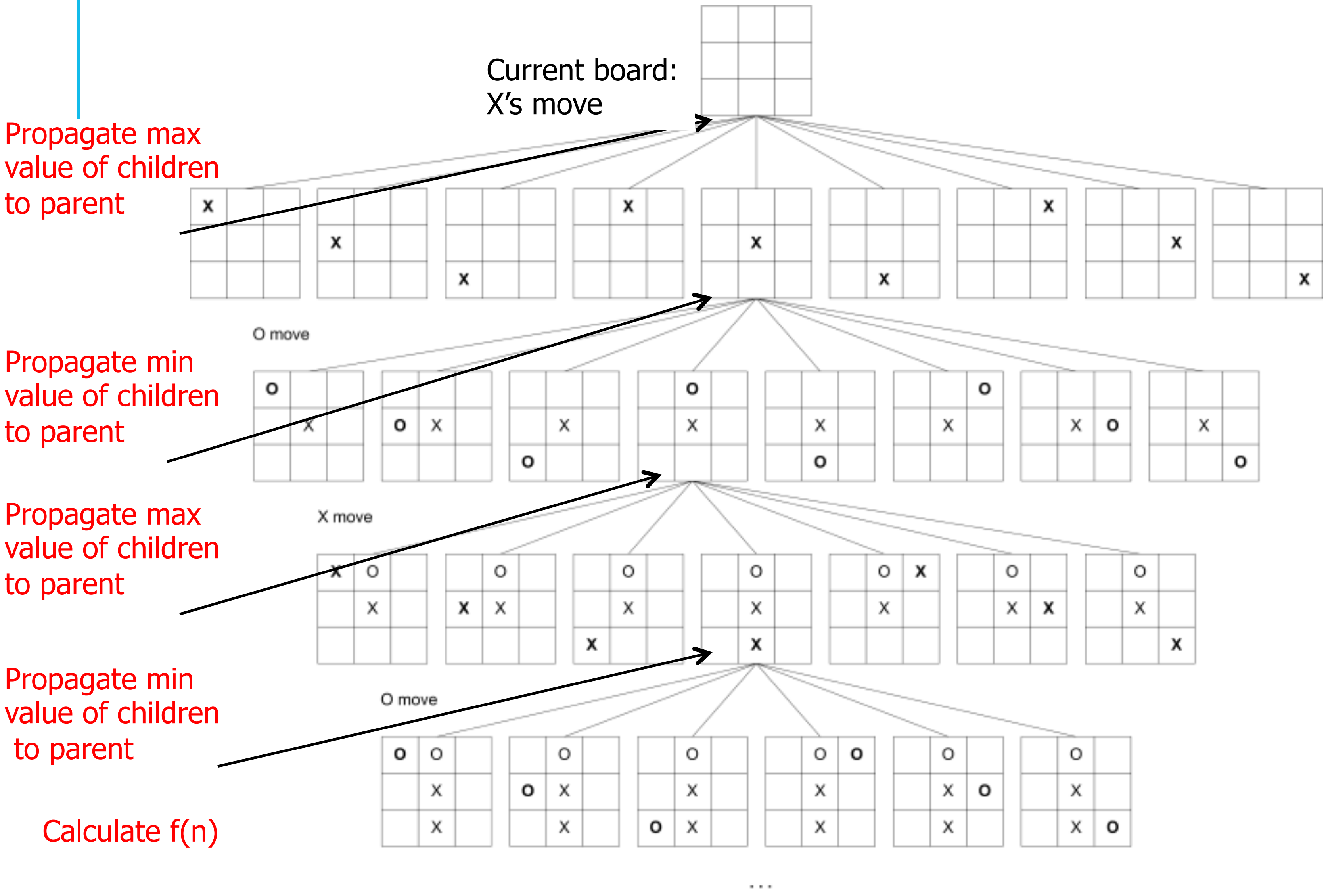


Propagate min value of children to parent

Calculate f(n)

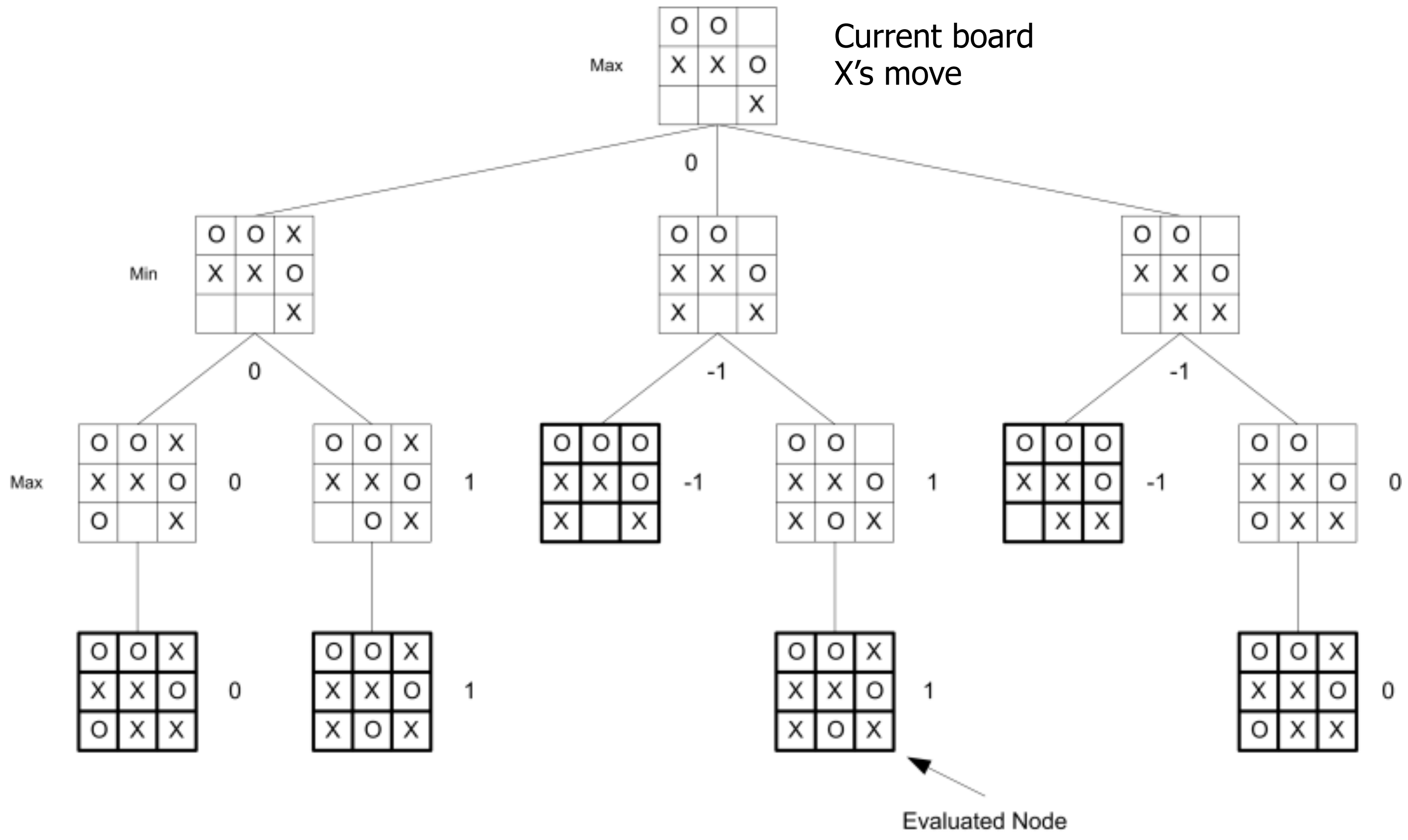

Current board:
X's move
O move
X move
O move
Propagate max value of children to parent
Propagate min value of children to parent
Calculate f(n)
...


Current board:
X's move
O move
X move
O move
Propagate min value of children to parent
Propagate max value of children to parent
Propagate min value of children to parent
Calculate f(n)
. . .

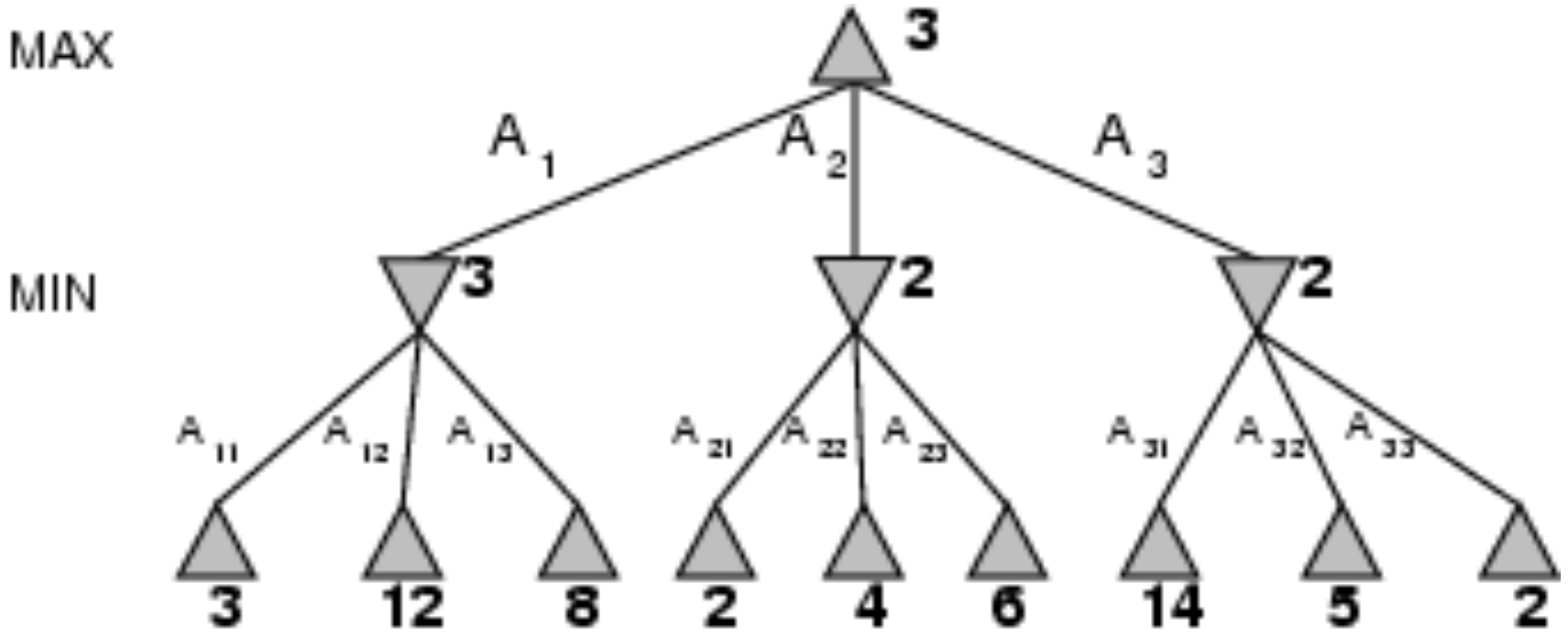

Current board:
X's move
Propagate max value of children to parent
O move
Propagate min value of children to parent
X move
Propagate max value of children to parent
O move
Propagate min value of children to parent
Calculate f(n)
. . .

MINIMAX ALGORITHM: EXAMPLE

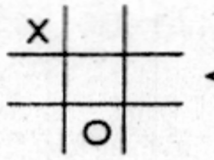
FROM M. T. JONES, *ARTIFICIAL INTELLIGENCE: A SYSTEMS APPROACH*



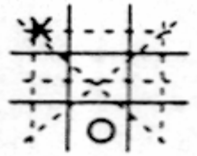
MAX MIN BACKED UP VALUES



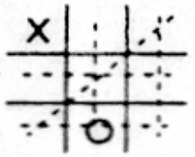
APPLYING MINIMAX TO TIC-TAC-TOE



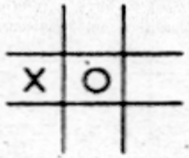
X has 6 possible win paths:



O has 5 possible wins:

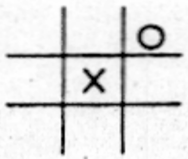


E(n) = 6 – 5 = 1



X has 4 possible win paths;
O has 6 possible wins

E(n) = 4 – 6 = –2



X has 5 possible win paths;
O has 4 possible wins

E(n) = 5 – 4 = 1

Heuristic is E(n) = M(n) – O(n)
where M(n) is the total of My possible winning lines
O(n) is total of Opponent's possible winning lines
E(n) is the total Evaluation for state n

Figure 4.16 Heuristic measuring conflict applied to states of tic-tac-toe.

BACKUP VALUES

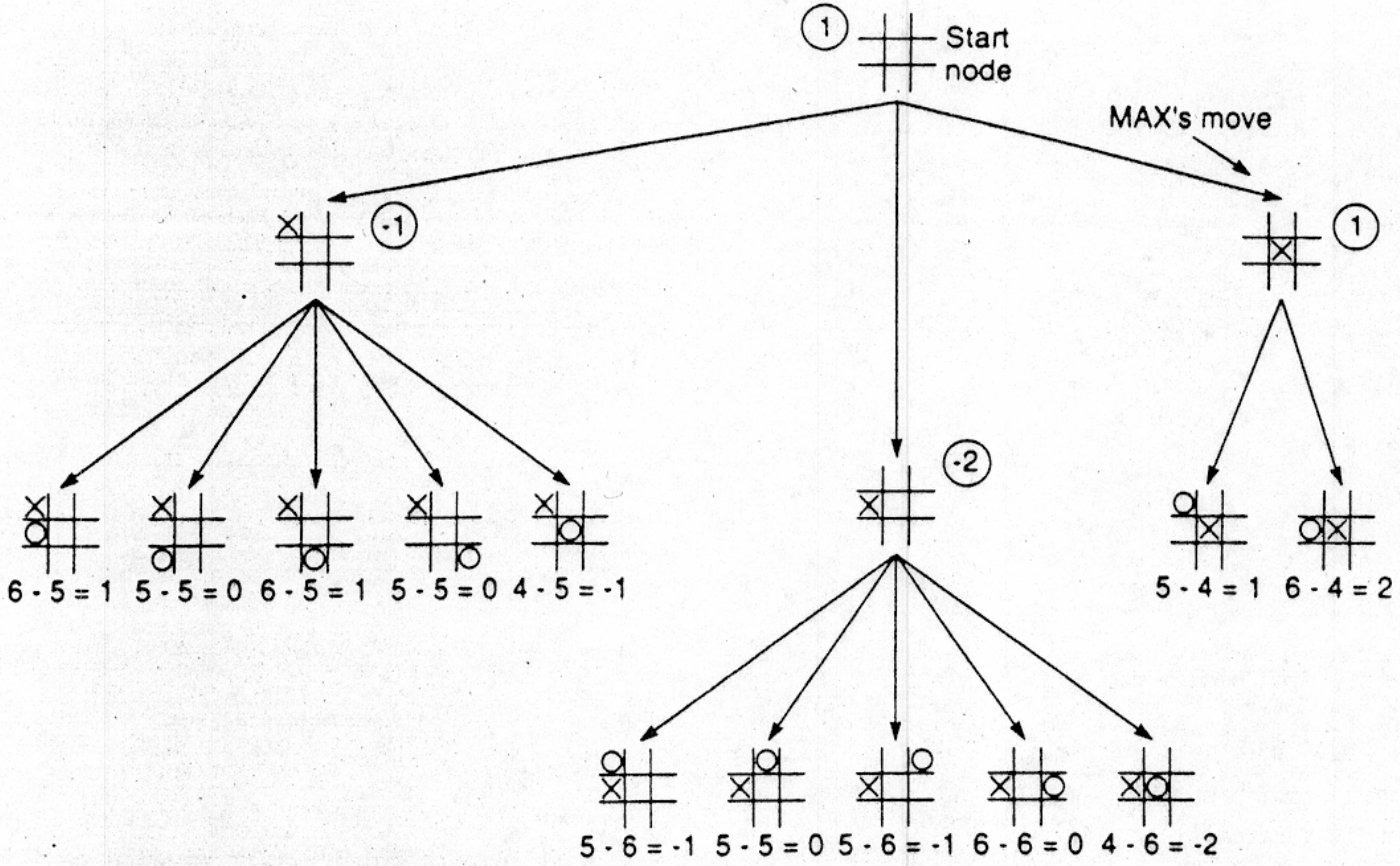


Figure 4.17 Two-ply minimax applied to the opening move of tic-tac-toe.

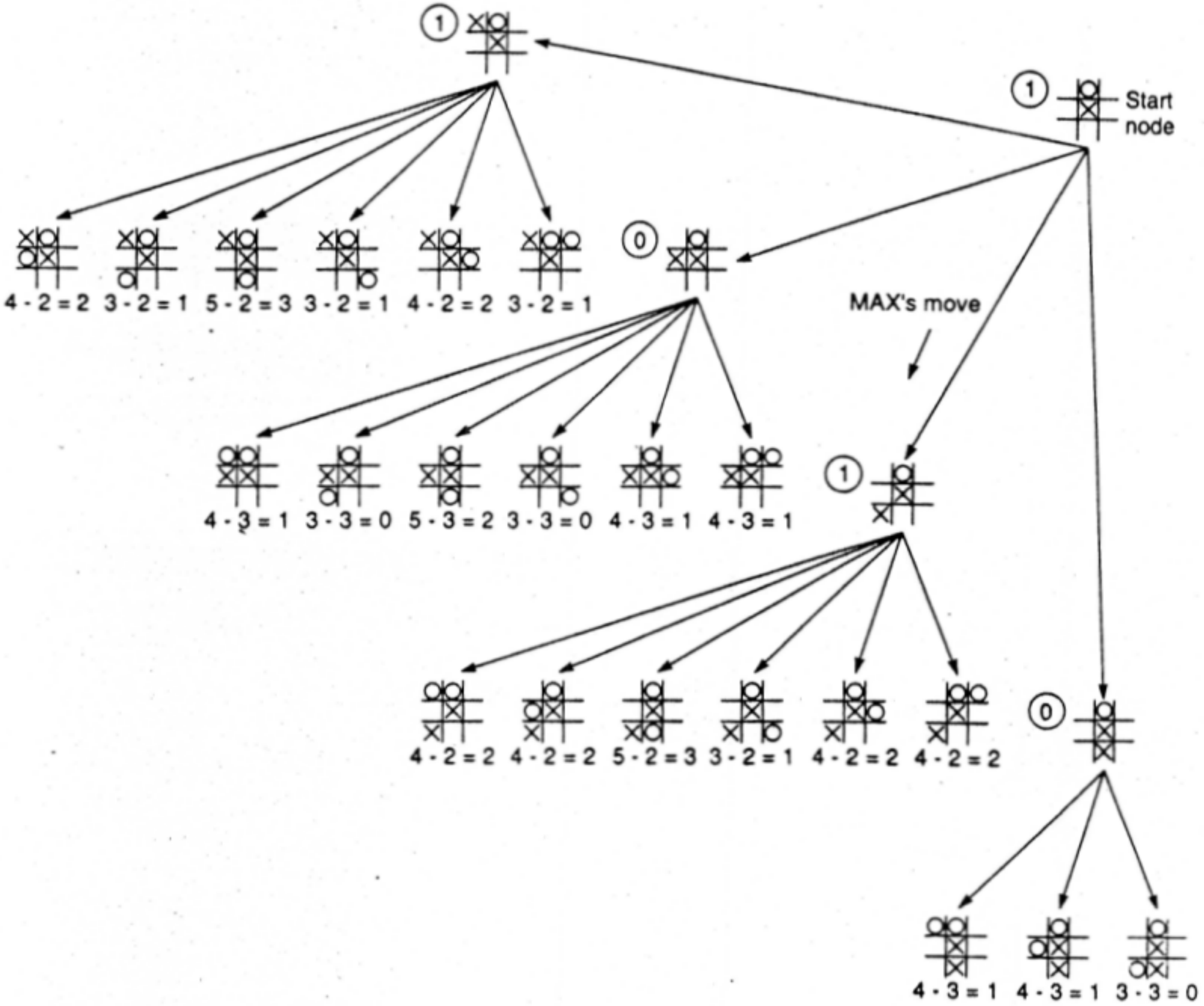


Figure 4.18 Two-ply minimax applied to **X**'s second move of tic-tac-toe.

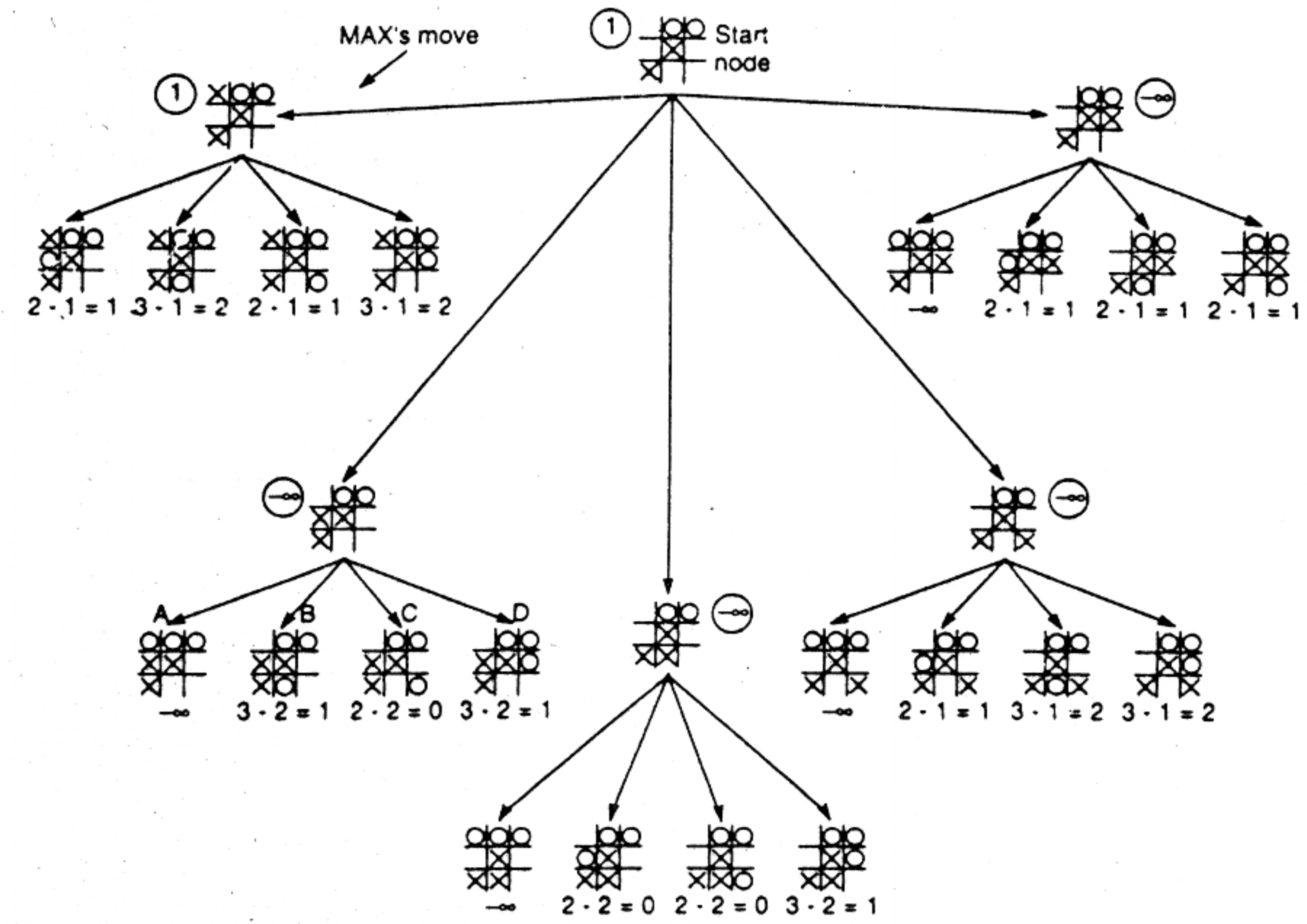


Figure 4.19 Two-ply minimax applied to **X**'s move near end game.

MAX MIN STRATEGY

Complete: Yes (if tree is finite)

Time complexity: $O(b^d)$

Space complexity: $O(bd)$ (depth-first exploration)

Optimal: Yes (against an optimal opponent)

ALPHA BETA PRUNING

- A way to reduce the number of nodes that need to generated and evaluated
- A branch and bound idea

Alpha Cutoff: when the value of a min position is less than or equal to the alpha-value of its parent, stop generating further successors

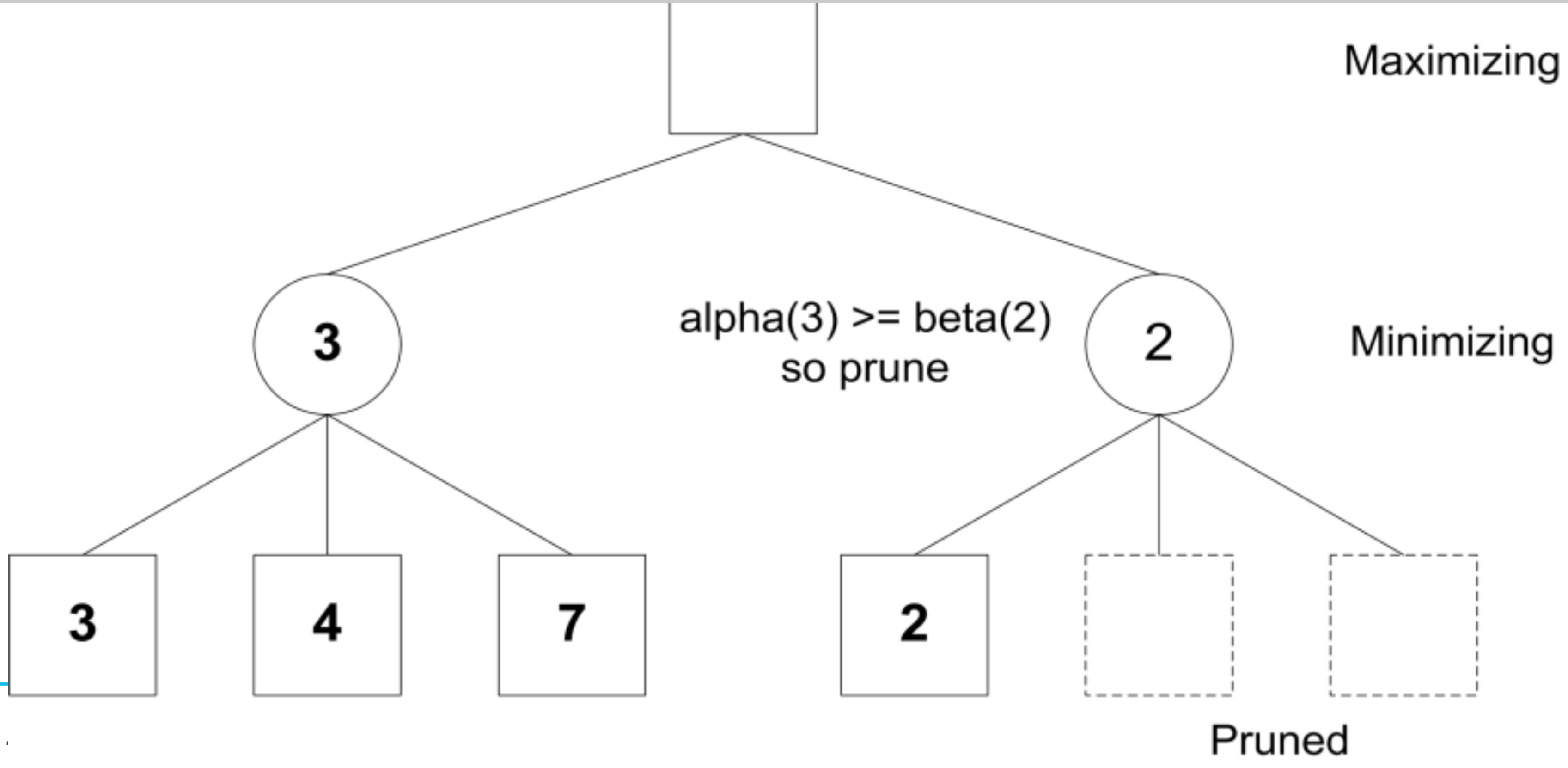
Beta Cutoff: when the value of a max position is greater than the beta-value of its parent, stop generating further successors.

A-B PRUNING EXAMPLE

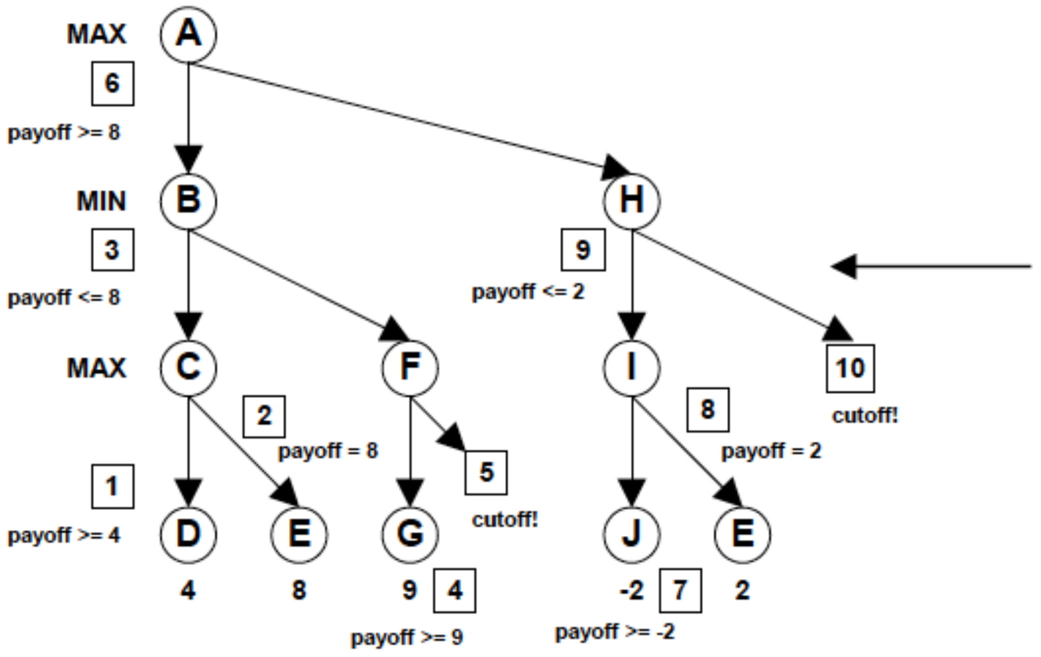
α-= value of the best possible move you can make, that you have computed so far

β = value of the best possible move your opponent can make, that you have computed so far

If at any time, α **>=** β , then your opponent's best move can force a worse position than your best move so far, and so there is no need to further evaluate this move



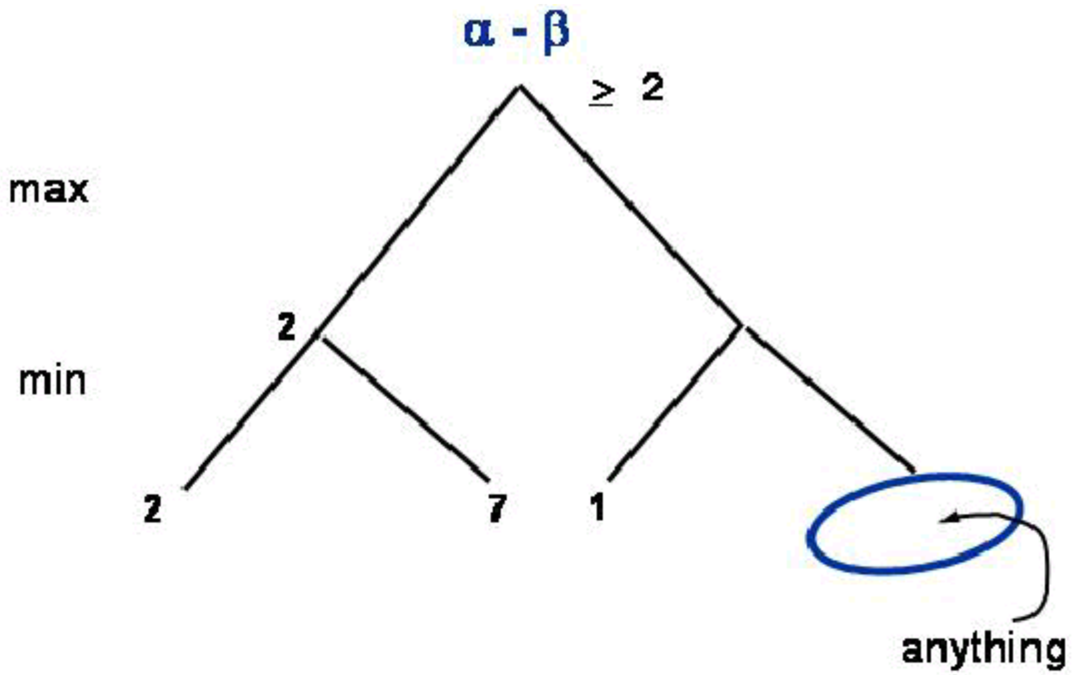
PRUNING- AN EXAMPLE



Numbered Boxes show Salient decisions

- Prune at F after exploring G; F has payoff >= 9 (MAX); there exists a MIN ancestor with payoff <= 8; hence, MIN would prevent us from ever getting to F.
- Prune at H after exploring I; H has payoff <= 2 (MIN); there exists a MAX ancestor with payoff >= 8; hence, MAX would prevent us from ever getting to H.

ALPHA-BETA PRUNING



α is lower bound on score

β is upper bound on score

A-B MINI-MAX

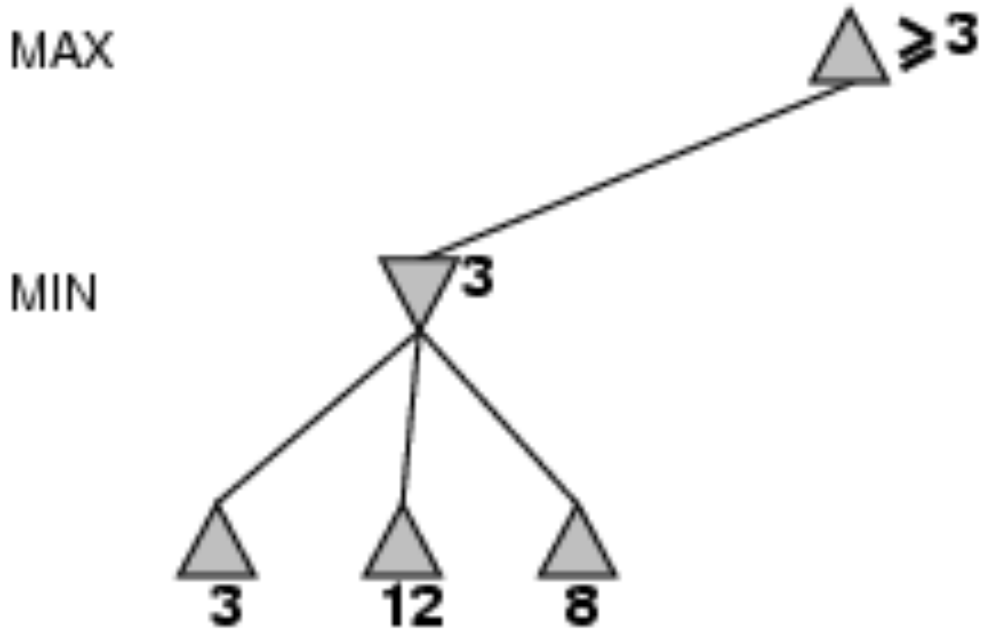
- α–values at Max nodes never decrease.
- β –values at Min nodes never increase.

A-B MINI-MAX

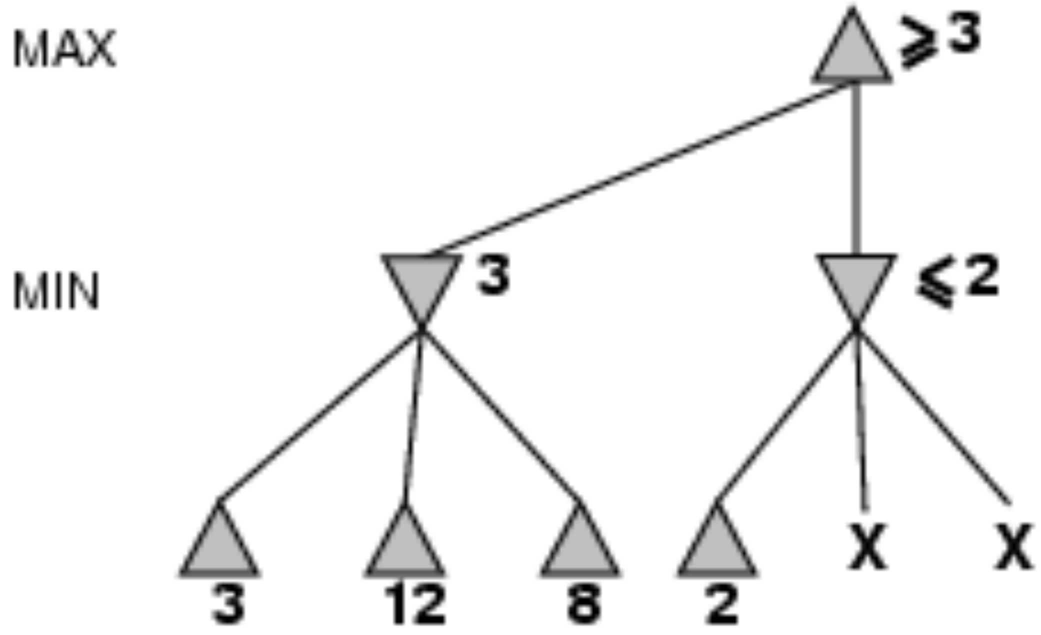
Keep track of a, b values and send on to children :

1) Search discontinued below any Min node with β <= α of one of it's ancestors. Set final value of node to be this b value.
2) Search discontinued below any Max node with α >= β of one of it's ancestors. Set final value of node to be this a value.

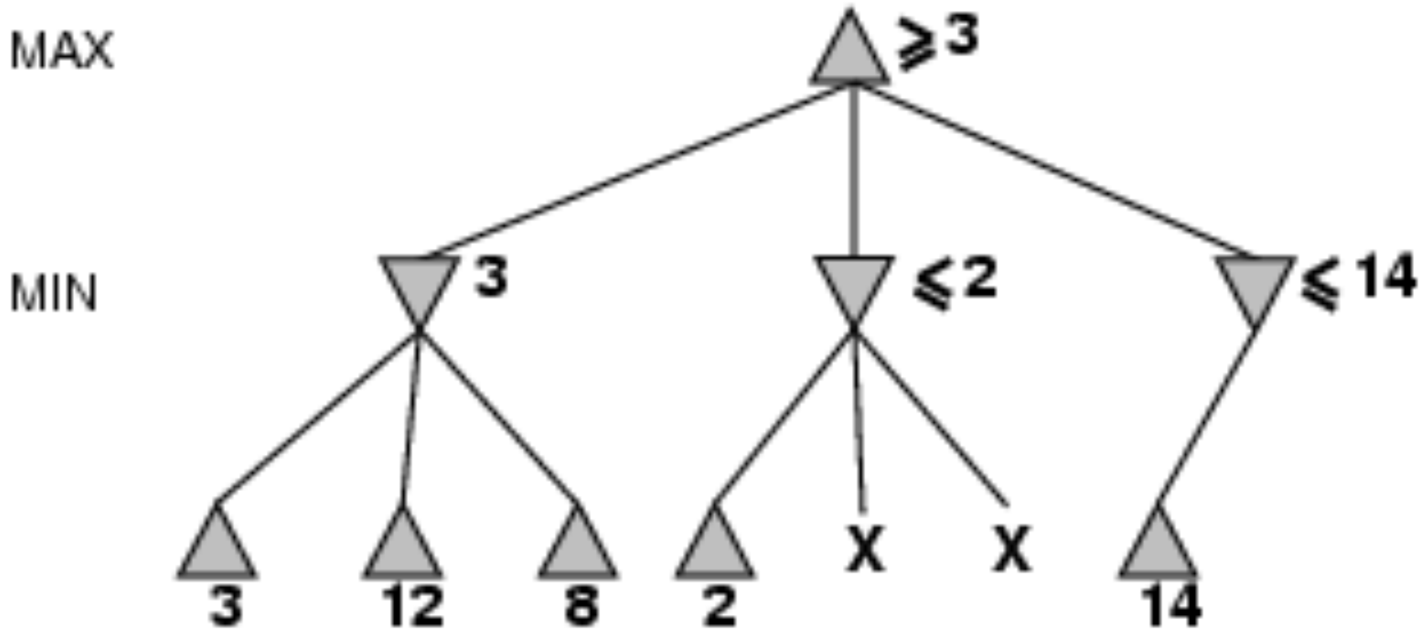
A-B PRUNING EXAMPLE [13]



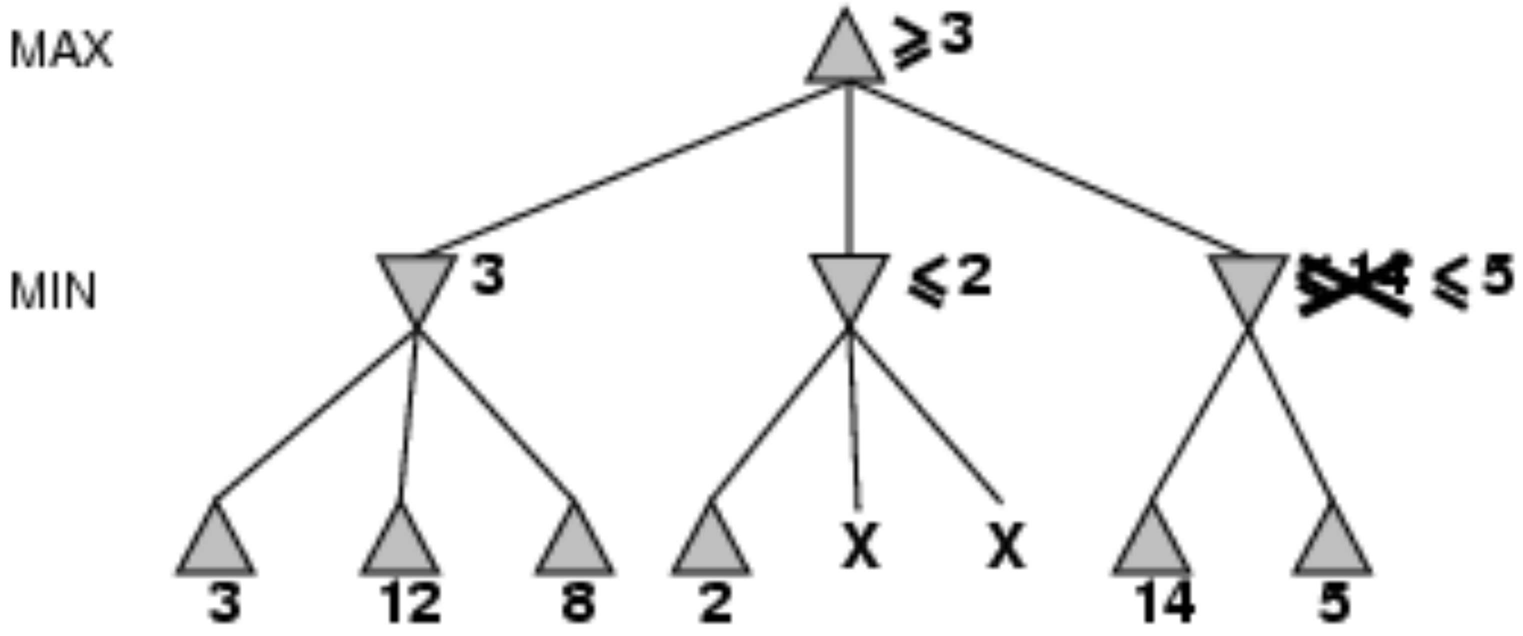
A-B PRUNING EXAMPLE



A-B PRUNING EXAMPLE



A-B PRUNING EXAMPLE



ALPHA BETA PRUNING

It is guaranteed to return exactly the same value as the Min-Max algorithm. It is a pure optimization without any approximations or tradeoffs.

In a perfectly ordered tree, with the best moves on the left, alpha beta reduces the cost of the search from order b^d to order $b^{(d/2)}$, that is, we can search twice as deep.

Worst case it won't prune any nodes.

META-HEURISTICS

Meta-heuristics are algorithms that combine heuristics in a higher level framework aimed at efficiently and effectively exploring the search space,

Types:

- Trajectory methods, which handles one solution at a time,
- Population-based, which handles multiple solutions.

REFERENCES

1. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. ”Optimization by Simulated Annealing”. *Science* 220, 671-680, 1983.
2. F. Glover. ”TabunSearch Part I”. ORSA Journal on Computing, vol. 1, no. 3, pp. 190-206, 1989.
3. F. Glover. “TabunSearch Part II”. ORSA Journal on Computing, vol. 2, no. 1, pp. 4-32, 1989.
4. I. Rechenberg. “Cybernetic Solution Path of an Experimental Problem”. Library Translation1122, August 1965. Farnborough Hants: Royal AircraftEstablishment. English translation of lecture given at the Annual Conference of the WGLR, Berlin, 1964.
5. L. J. Fogel, A.J. Owens and M.J. Walsh.”Artificial Intelligence through a Simulation of Evolution”. In: Maxeld, Callahan, and Fogel, pp. 131-155, 1965.
6. J. H. Holland.”Adaptation in Natural and Artificial Systems”. University of Michigan Press, Ann Arbor MI, 1975.

REFERENCES

7. J. Koza.”Genetic Programming”. MIT Press, Cambridge MA, 1992.

8. M. Dorigo.”Optimization, Learning and Natural Algorithms”. Ph.D. thesis, DEI, Politecnico di Milano, Italy, pp. 140, 1992.

9. J. Kennedy and R. C. Eberhart. “Particle Swarm Optimization”. Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995.

10. R. C. Eberhart and J. Kennedy. “A New Optimizer using Particle Swarm Theory,” Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43, 1995.

11. A. E. Eiben, R. Hinterding and Z. Michaleweiz.”Parameter Control in Evolutionary Algorithms”. IEEE Transactions on Evolutionary Computing, vol. 3, issue 2, pp. 124-141, 1999.

12. S. Russel and P. Norving.”AI: A modern approach”, Prentice Hall, 2003