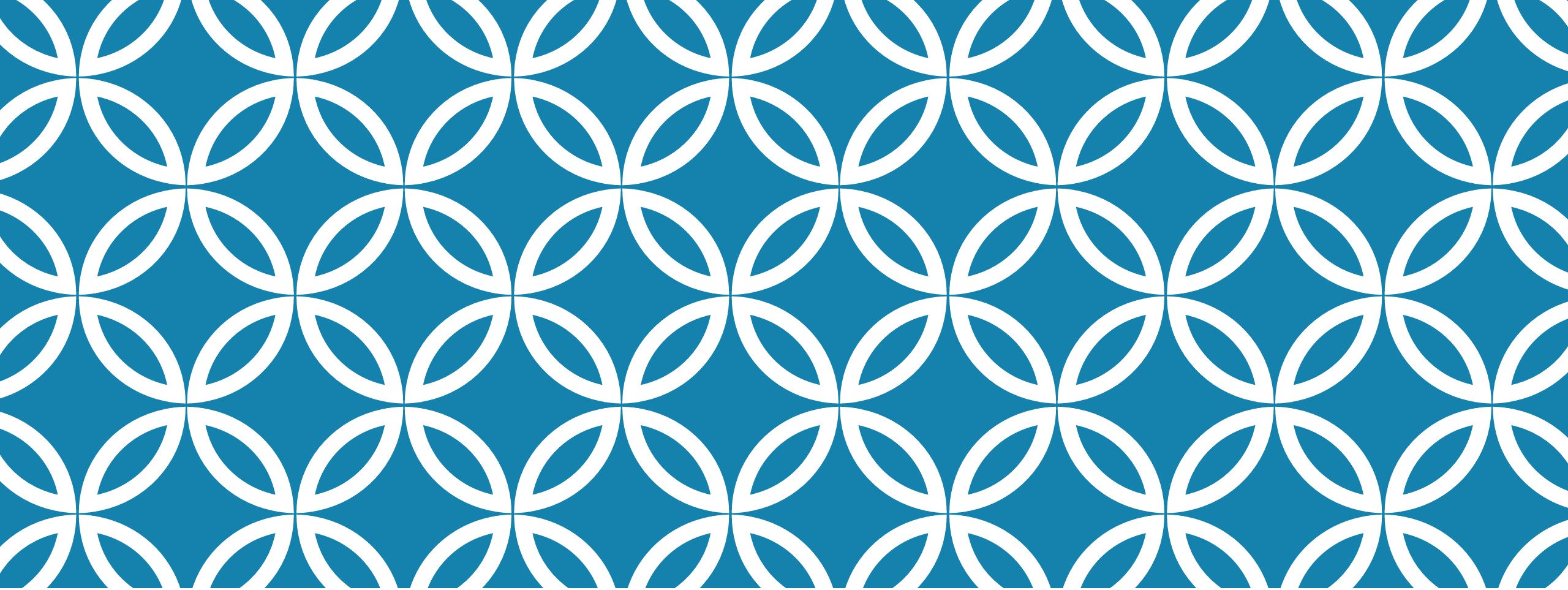


GAME PLAYING

Sukmawati Nu Endah

GAME PLAYING

- Dalam state space search biasa:
  - agent berinteraksi dengan environment (biasanya static & deterministic).
- Terkadang environment yang berisi agent lain:
  - cooperative, competitive
- Melawan agent “musuh”: adversarial search→game
- Adversarial search is used in games where one player (or multiple players) tries to maximize its score but it is opposed by another player (or players)

JENIS JENIS GAME

	deterministic	stochastic
perfect information	chess, checkers, go, othello	monopoly, backgammon
imperfect information		bridge, poker, nu-clear war

Go game

checkers

JENIS JENIS GAME

Dalam sejarah AI, game yang biasanya jadi bahan riset:

- Deterministic
- Perfect information : pemain mengetahui semua informasi state dari game
- 2 pemain
- zero sum - one player's win is the other's loss; kerugian pemain adalah keuntungan bagi pemain lain

GAME SEBAGAI SEARCH

State: konfigurasi papan dan info pemain yang akan berjalan

Successor function: mengembalikan list pasangan (move, state)

Terminal test: menentukan apakah permainan sudah selesai (terminal state)

Utility function: penilaian numerik terhadap terminal state. Mis: menang (+1), seri (0), kalah (-1).

Ke-4 hal ini mendefinisikan sebuah game tree.

CONTOH GAME TREE

Andaikan sebuah permainan antara 2 pemain: MAX (agent) dan MIN

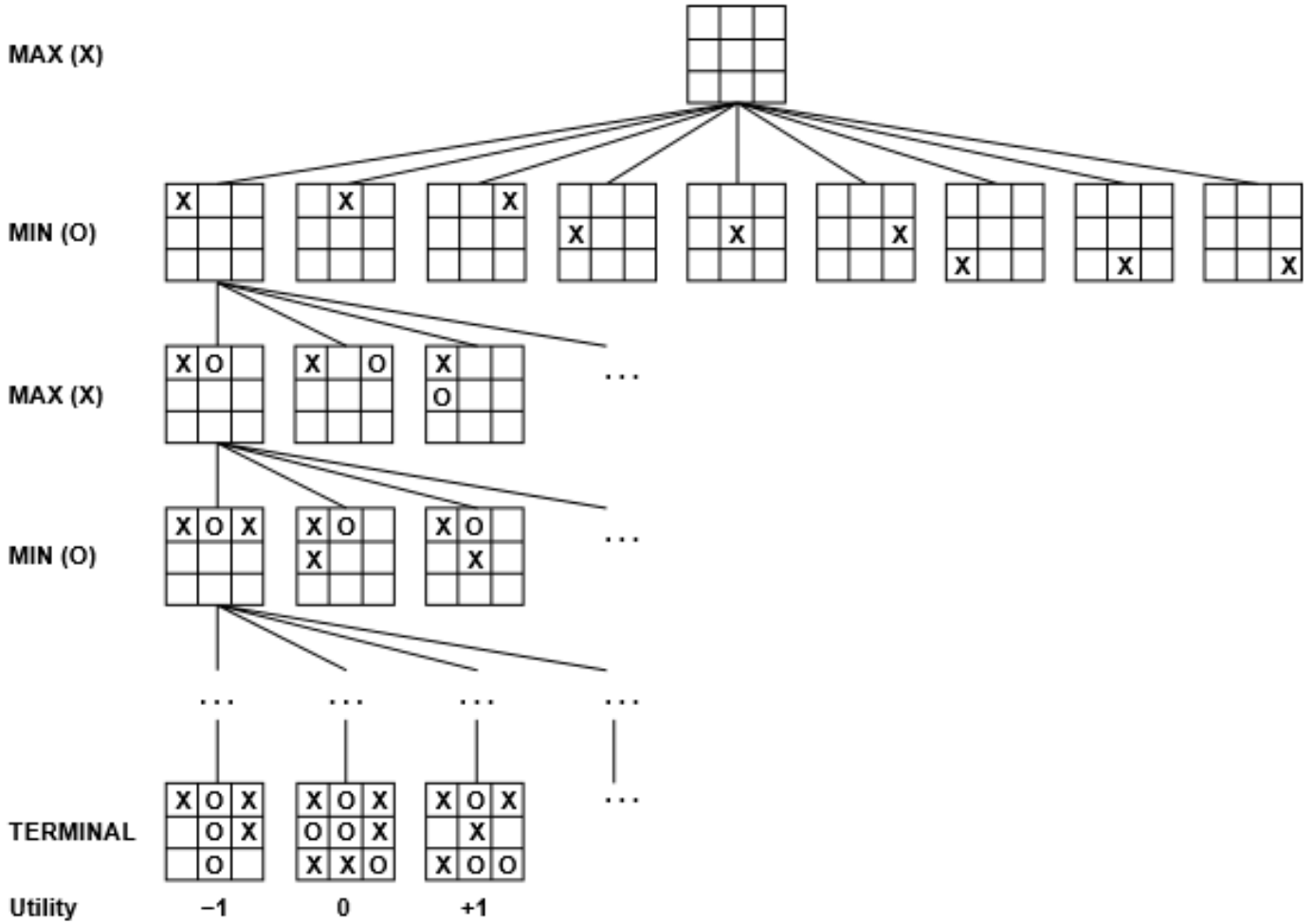
MAX jalan dulu, lalu MIN, dst. sampai game selesai

1 langkah = 2 ply (MAX jalan, MIN jalan)

Kalau search biasa, cari path sehingga mencapai terminal state di mana MAX menang

Tapi langkah MIN di luar kendali si agent MAX!

Solusi berupa contingent strategy untuk setiap kemungkinan langkah MIN



ALGORITMA MIN-MAX (ALGORITMA MINIMAX)

Utility function memiliki aturan yang mirip dengan sebuah fungsi heuristic

Mengevalusi seberapa baik node tersebut untuk pemain

Gambar berikut menunjukkan contoh utility function untuk Tic-Tac-Toe

Nilai positif mengindikasikan state menguntungkan untuk MAX

Nilai negative mengindikasikan state menguntungkan untuk MIN

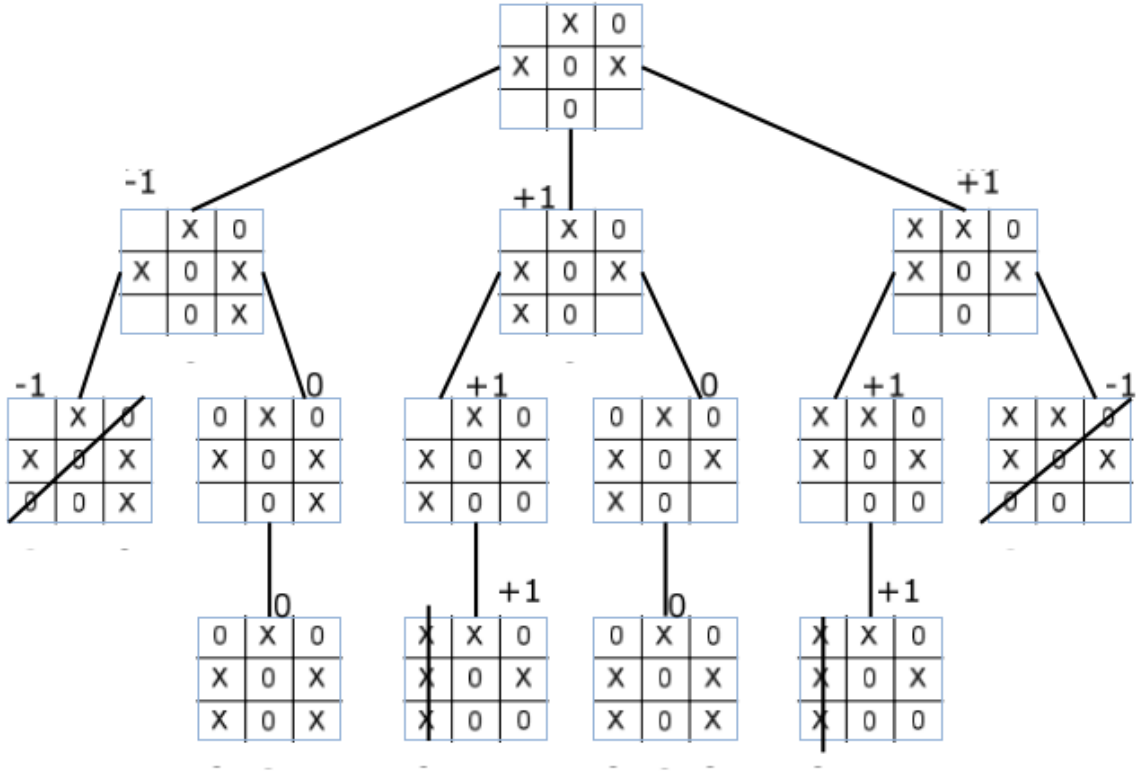


Fig. 5.1 Example of utility function for tic-tac-toe. Positive values indicate states advantageous for MAX and negative values indicate states advantageous for MIN.

LANJUTAN

To find the best move, the system first generates all possible legal moves, and applies them to the current board.

In a simple game this process is repeated for each possible move until the game is won, lost, or drawn.

The fitness of a toplevel move is determined by whether it eventually leads to a win.

The generation of all moves is possible in simple games such as tic-tac-toe, but for complex games (such as chess) it is impossible to generate all the moves in reasonable time.

In this case only the state within a few steps ahead may be generated

LANJUTAN

Algorithm 5.1 MIN-MAX Algorithm

Step 1. Expand the entire tree below the root.

Step 2. Using the utility (evaluation) function, evaluate the terminal nodes as wins for the minimizer or maximizer.

Step 3. Select a node all of whose children have been assigned values.

Step 3.1. **if** there is no such node
then the search process is finished.
Return the value assigned to the root.

Step 3.2 **if** the node is a minimizer move
then assign it a value that is the minimum of the values of its children.

Step 3.3 **if** the node is a maximizer move assign it a value that is the maximum of the values of its children.

Step 4. Return to Step 3.

end.

LANJUTAN

Definisi algoritma ini rekursif, dengan base case pada terminal state

Untuk menghitung MINIMAXVALUE pada initial state, harus depth-first search seluruh game tree!

Complete? Ya, kalau game tree-nya finite

Optimal? Ya, asumsi lawan musuh optimal juga. (Kalau tidak? “Lebih optimal”!)

Time complexity? O(bm)

Space complexity? O(bm) (atau O(m) dgn. backtracking)

Teori sih OK...

Untuk catur: $b \approx 35$, $m \approx 100 \rightarrow$ pencarian strategi optimal berdasarkan Minimax tidak feasible!

KETERBATASAN SUMBER DAYA

Biasanya dalam suatu permainan ada batasan waktu

Andaikan ada agent bermain catur yang diberi 100 detik untuk “berpikir” tiap langkah.

Mis. bisa memroses 10^4 node/detik $\rightarrow 10^6$node/langkah

Kita bisa melakukan aproksimasi sbb.:

- Cutoff: batasi depth yang diproses (≈IDS)
- Evaluation function: prediksi dari nilai utility function (tidak perlu sampai ke terminal state)

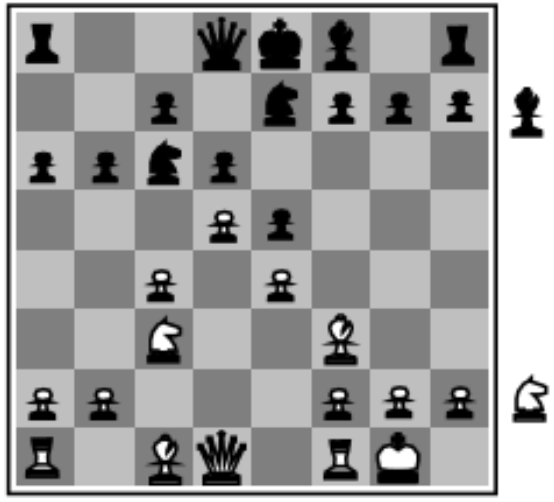
EVALUASI FUNCTION

Biasanya, evaluation function berupa kombinasi linier dari fitur-fitur sebuah state:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

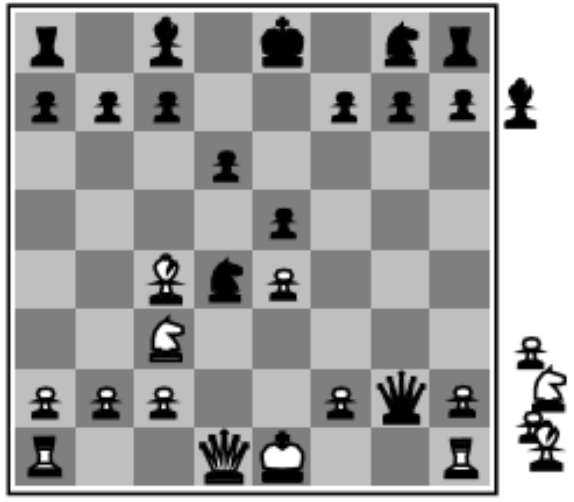
Mis. untuk catur:

- $w_1 = 1$, f_1 = jumlah pion putih - jumlah pion hitam
- $w_2 = 3$, f_2 = jumlah gajah putih - jumlah gajah hitam



Black to move

White slightly better



White to move

Black winning

PRUNING (MEMANGKAS) GAME TREE

Kinerja MINIMAX masih bisa diperbaiki dengan pruning (memangkas) game tree.

Prinsipnya: node (subtree) yang tidak mungkin mempengaruhi hasil akhir tidak perlu ditelusuri.

Pruning demikian dilakukan oleh algoritma alpha-beta pruning

TUGAS

Buatlah resume tentang Algoritma Alpha-Beta Pruning!