Báo cáo Bài tập lớn 2: Game Playing
Xây dựng Agent chơi Cờ Vua

Môn: Nhập môn Trí tuệ Nhân tạo - HK2 2024-2025

Phan Đình Tuấn Anh - 2210118 Hồ Anh Dũng - 2310543
Nguyễn Trọng Tài - 2212995 Nguyễn Thiện Minh - 2312097

Trường Đại học Bách Khoa TP.HCM
Khoa Khoa học và Kỹ thuật Máy tính

Tháng 5, Năm 2025

Giới thiệu & Mục tiêu

Đặt vấn đề

- Xây dựng Game Playing Agent cho Cờ Vua.
- Thử thách: Độ phức tạp cao, không gian trạng thái lớn ($\approx 10^{120}$).
- Yêu cầu: Thuật toán Searching truyền thống (không ML).

Mục tiêu chính Bài tập lớn

1. **Engine Đúng luật:** Agent tuân thủ mọi quy tắc.

Giới thiệu & Mục tiêu

Đặt vấn đề

- Xây dựng Game Playing Agent cho Cờ Vua.
- Thử thách: Độ phức tạp cao, không gian trạng thái lớn ($\approx 10^{120}$).
- Yêu cầu: Thuật toán Searching truyền thống (không ML).

Mục tiêu chính Bài tập lớn

1. **Engine Đúng luật:** Agent tuân thủ mọi quy tắc.
2. **Thi đấu Cơ bản:** Thắng Random Agent 10/10.

Giới thiệu & Mục tiêu

Đặt vấn đề

- Xây dựng Game Playing Agent cho Cờ Vua.
- Thử thách: Độ phức tạp cao, không gian trạng thái lớn ($\approx 10^{120}$).
- Yêu cầu: Thuật toán Searching truyền thống (không ML).

Mục tiêu chính Bài tập lớn

1. **Engine Đúng luật:** Agent tuân thủ mọi quy tắc.
2. **Thi đấu Cơ bản:** Thắng Random Agent 10/10.
3. **Phân cấp Độ khó:** Điều chỉnh độ khó (qua search depth).

Giới thiệu & Mục tiêu

Đặt vấn đề

- Xây dựng Game Playing Agent cho Cờ Vua.
- Thử thách: Độ phức tạp cao, không gian trạng thái lớn ($\approx 10^{120}$).
- Yêu cầu: Thuật toán Searching truyền thống (không ML).

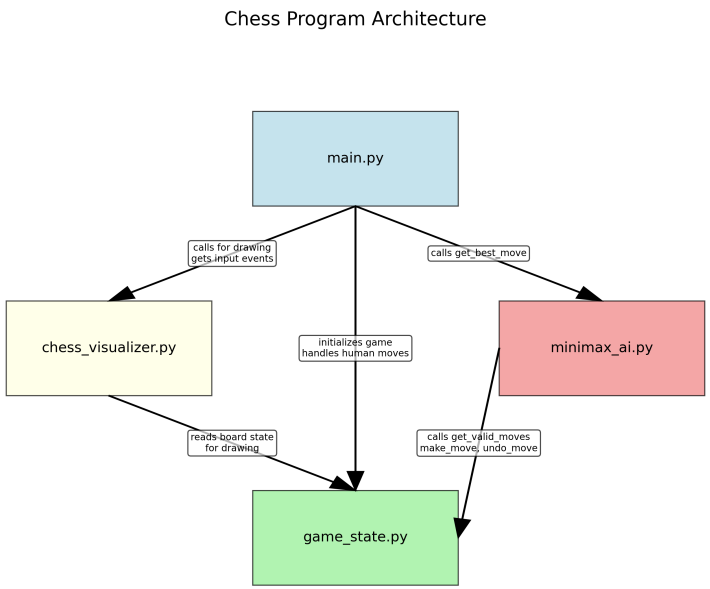
Mục tiêu chính Bài tập lớn

1. **Engine Đúng luật:** Agent tuân thủ mọi quy tắc.
2. **Thi đấu Cơ bản:** Thắng Random Agent 10/10.
3. **Phân cấp Độ khó:** Điều chỉnh độ khó (qua search depth).
4. **Tập trung Searching:** Logic AI dựa trên Searching.

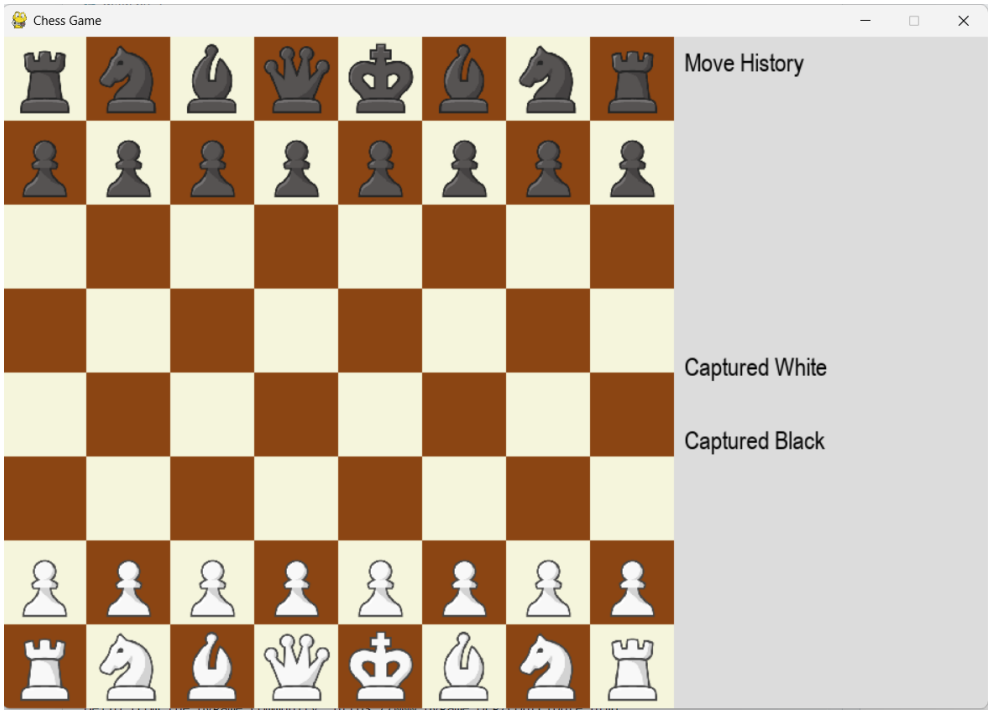
Kiến trúc Hệ thống & Giao diện

Modules chính:

- `game_state.py`
- `minimax_ai.py`
- `visualizer.py`
- `main.py`



Hình: Sơ đồ kiến trúc (Báo cáo Hình 1)



Hình: Giao diện chương trình (Báo cáo Hình 3).

Engine Cờ Vua (game_state.py)

Biểu diễn Trạng thái & Quân cờ

- Bàn cờ: List 8x8 (`'wp'`, `'bn'`, `'--'`).
- Quản lý: Lượt đi, vị trí vua, quyền nhập thành, en passant, lịch sử.
- Kết thúc: Checkmate, stalemate (hết nước, lặp 3 lần, thiếu quân).

Sinh & Kiểm tra Nước đi Hợp lệ (get_valid_moves)

- GĐ 1: Sinh nước đi "tiềm năng"(pseudo-legal).
- Xử lý đặc biệt: Nhập thành, Bắt Tốt qua đường, Phong cấp (chọn Q/R/B/N).
- GĐ 2: Kiểm tra hợp lệ: Mô phỏng, đảm bảo Vua không tự chiếu.

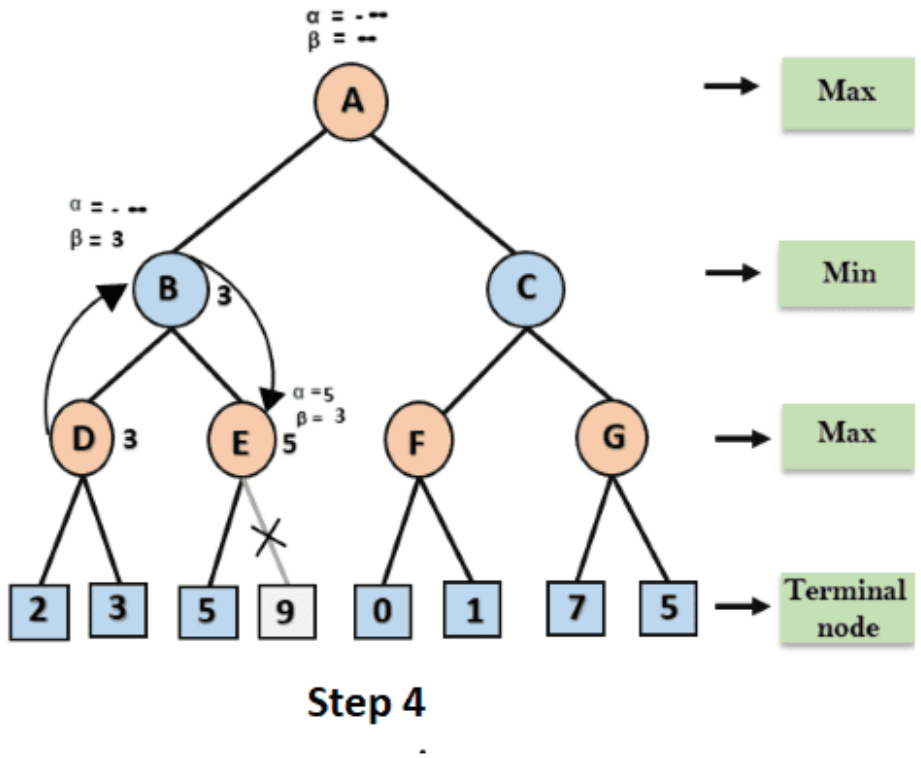
Thuật toán Cốt lõi: Minimax & Alpha-Beta Pruning

Minimax:

- Duyệt cây trò chơi (depth-limited).
- Max (AI) vs Min (Đối thủ).
- Truyền giá trị lá lên để chọn nước tốt nhất.

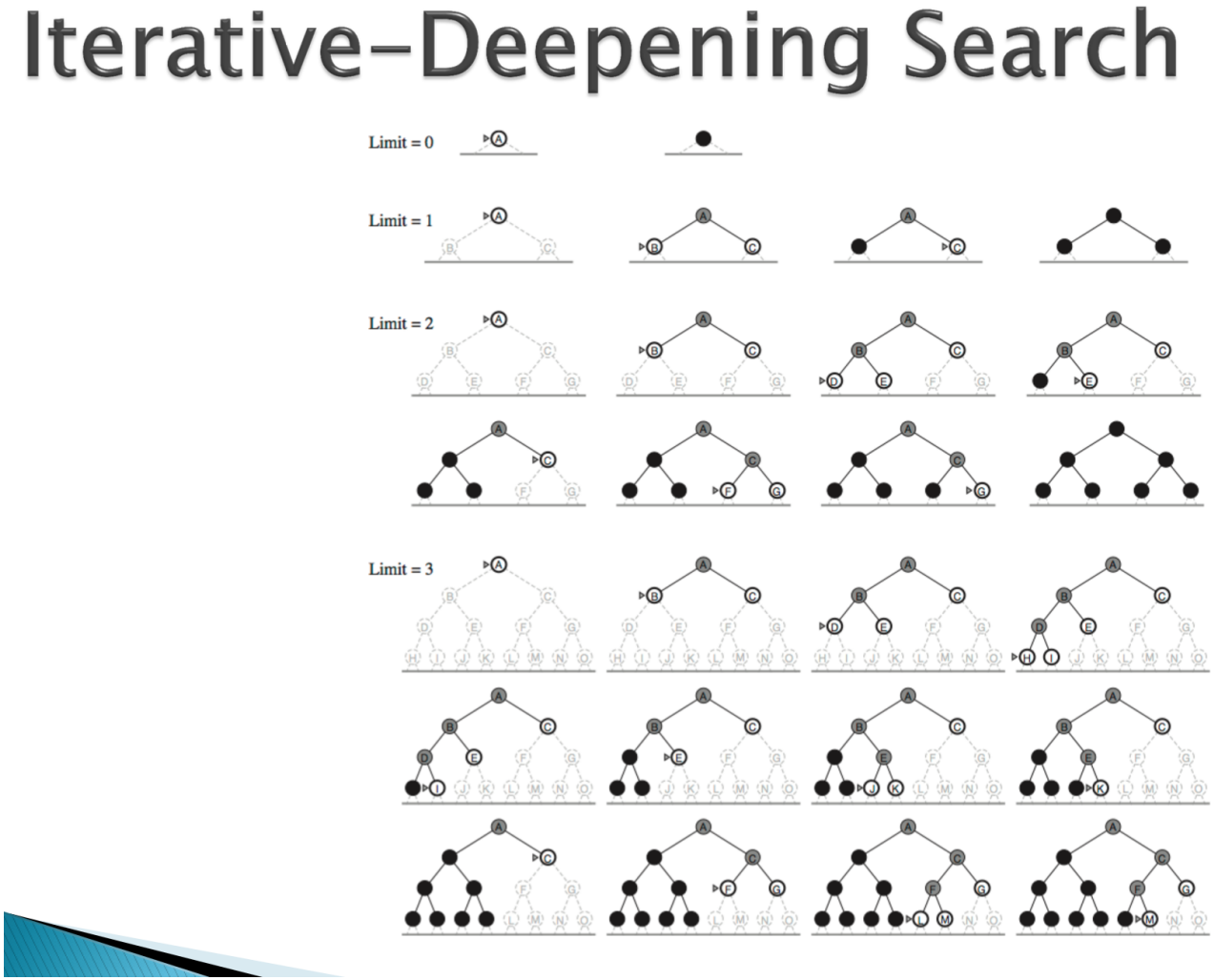
Alpha-Beta Pruning:

- Tối ưu Minimax, giảm node duyệt.
- Duy trì cửa sổ $[\alpha, \beta]$.
- Cắt tỉa nhánh khi $\alpha \geq \beta$.



Hình: Minh họa Alpha-Beta. Nguồn: studydx.com

Tối ưu hóa: Iterative Deepening Search (IDS)



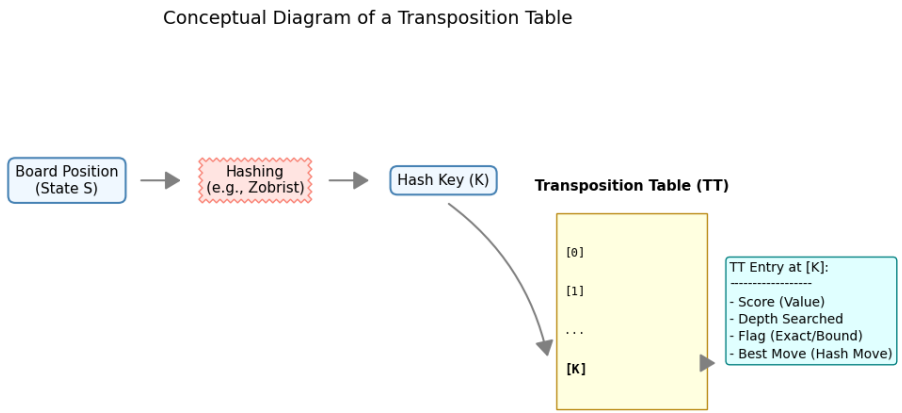
Hình: Minh họa IDS. Nguồn: AI StackExchange

Tối ưu hóa: Transposition Table (TT)

Mục đích & Hoạt động [1]

Cache kết quả trạng thái đã duyệt, tránh tính toán lại (transpositions).

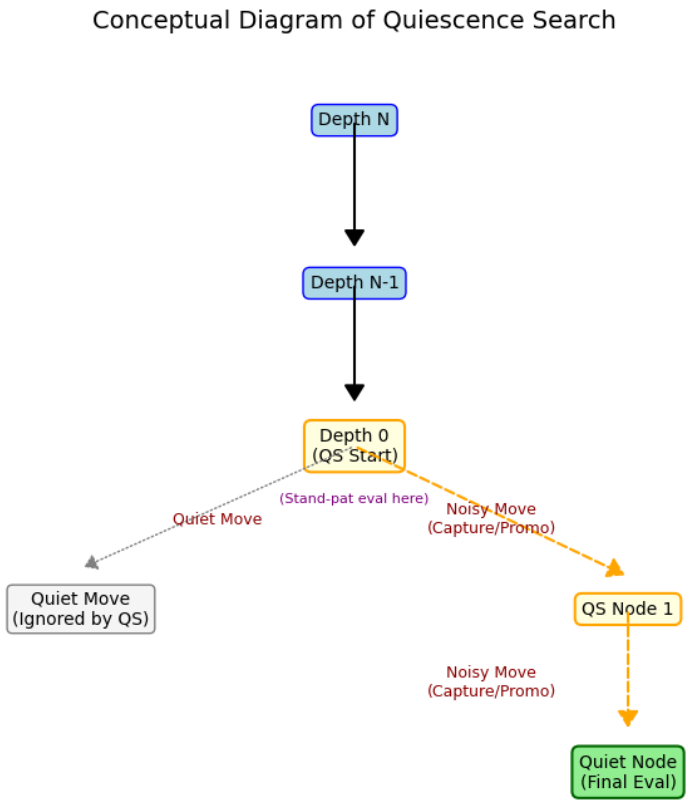
- **Khóa (Key):** Chuỗi đại diện trạng thái (như Zobrist Hash).
- **Dữ liệu lưu:** ‘score‘, ‘depth‘, ‘flag‘ (EXACT, ...), ‘best_move‘.
- **Sử dụng:** Trả score, thu hẹp $[\alpha, \beta]$, ưu tiên ‘best_move‘ [2].



Hình: Khái niệm TT .

Implementation: Python dict. Hiệu quả $D \geq 4$.

Tối ưu hóa: Quiescence Search (QS)



Hình: Sơ đồ QS: Tại depth 0, QS khám phá noisy moves.

Tối ưu hóa: Sắp xếp Nước đi (Move Ordering) & MVV-LVA

MVV-LVA (Most Valuable Victim - Least Valuable Attacker)
Heuristic for Prioritizing Capture Moves

Attacker Piece (Value)	Victim Piece (Value)	MVV-LVA Score (VictimVal*K - AttackerVal)	Priority for Ordering
Pawn (100)	Queen (900)	(900 * 10) - 100 = 8900	Highest
Knight (320)	Queen (900)	(900 * 10) - 320 = 8680	Very High
Pawn (100)	Rook (500)	(500 * 10) - 100 = 4900	High
Bishop (330)	Bishop (330)	(330 * 10) - 330 = 2970	Medium (Even Trade)
Queen (900)	Pawn (100)	(100 * 10) - 900 = 100	Low (Likely Bad)
Rook (500)	Knight (320) (Protected)	(320 * 10) - 500 = 2700	Medium (If unprotected)

Hàm Lượng giá (`evaluate_board()`)

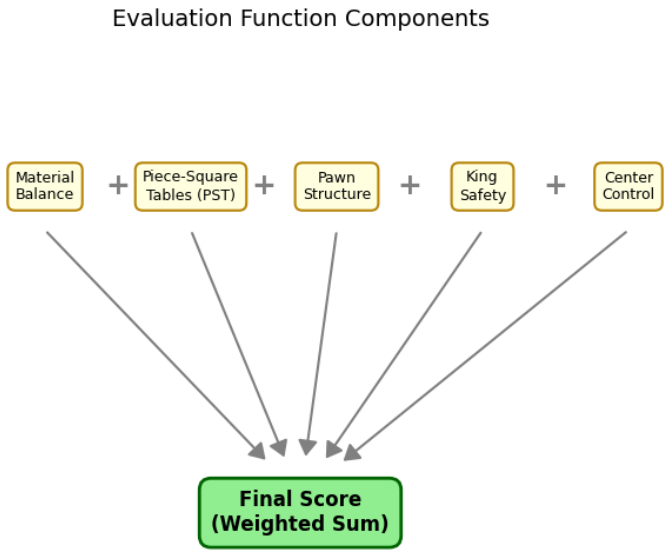
Mục đích

Gán score cho trạng thái tĩnh (góc nhìn Trắng). Điểm dương lợi Trắng, âm lợi Đen. Cốt lõi cho Minimax.

Heuristics chính:

- Material Balance
- PST
- Pawn Structure
- King Safety
- Center Control

Tổng hợp: `Score` $= \sum(w_i \times h_i)$



Hình: Thành phần hàm lượng giá .

Hàm Lượng giá: Piece-Square Tables (PST)

Khái niệm & Mục đích [3, 4]

- Bảng 8x8/quân: điểm thưởng/phạt cho vị trí.
- Mã hóa nguyên tắc chiến lược vị trí tối ưu.
- Vua: PST khác nhau trung cuộc/tàn cuộc.

Đặc điểm Project [5]:

- Quân nhẹ $\rightarrow$ trung tâm.
- Tốt $\rightarrow$ tiến sâu.
- Vua: Dùng PST MID/END.

Bảng: Ví dụ: PST Mã Trắng

-50	-40	-30	-30	-30	-30	-40	-50
-40	-20	0	0	0	0	-20	-40
-30	0	10	15	15	10	0	-30
-30	5	15	20	20	15	5	-30
-30	0	15	20	20	15	0	-30
-30	5	10	15	15	10	5	-30
-40	-20	0	5	5	0	-20	-40
-50	-40	-30	-30	-30	-30	-40	-50

Kết quả: Tính đúng luật & Thi đấu Random Agent

Kiểm tra tính đúng luật (YC 1)

✓ Agent tuân thủ đầy đủ luật Cờ Vua.

Thi đấu với Agent Ngẫu nhiên (YC 2)

- Agent (D2) vs Random: **Thắng 100% (10/10)**.
- TB thắng: 15.66s/ván; 31.1 ply/ván.

Kết quả: Phân cấp Độ khó

Phân cấp Độ khó (YC 3)

- Độ khó: D2 ("Easy"), D3 ("Medium"), D4 ("Hard").
- Tăng Depth $\rightarrow$ Sức chơi, tính toán chiến thuật tăng.
- Quyết định "thông minh"hơn ở Depth cao.

✓ *YC phân cấp độ khó qua search depth được đáp ứng.*

Kết quả: Hiệu năng Tìm kiếm (Bảng)

Hiệu năng TB AI vs Random (max 50 nước đi đầu của AI):

Bảng: Bảng 2: Hiệu năng tìm kiếm TB theo độ sâu

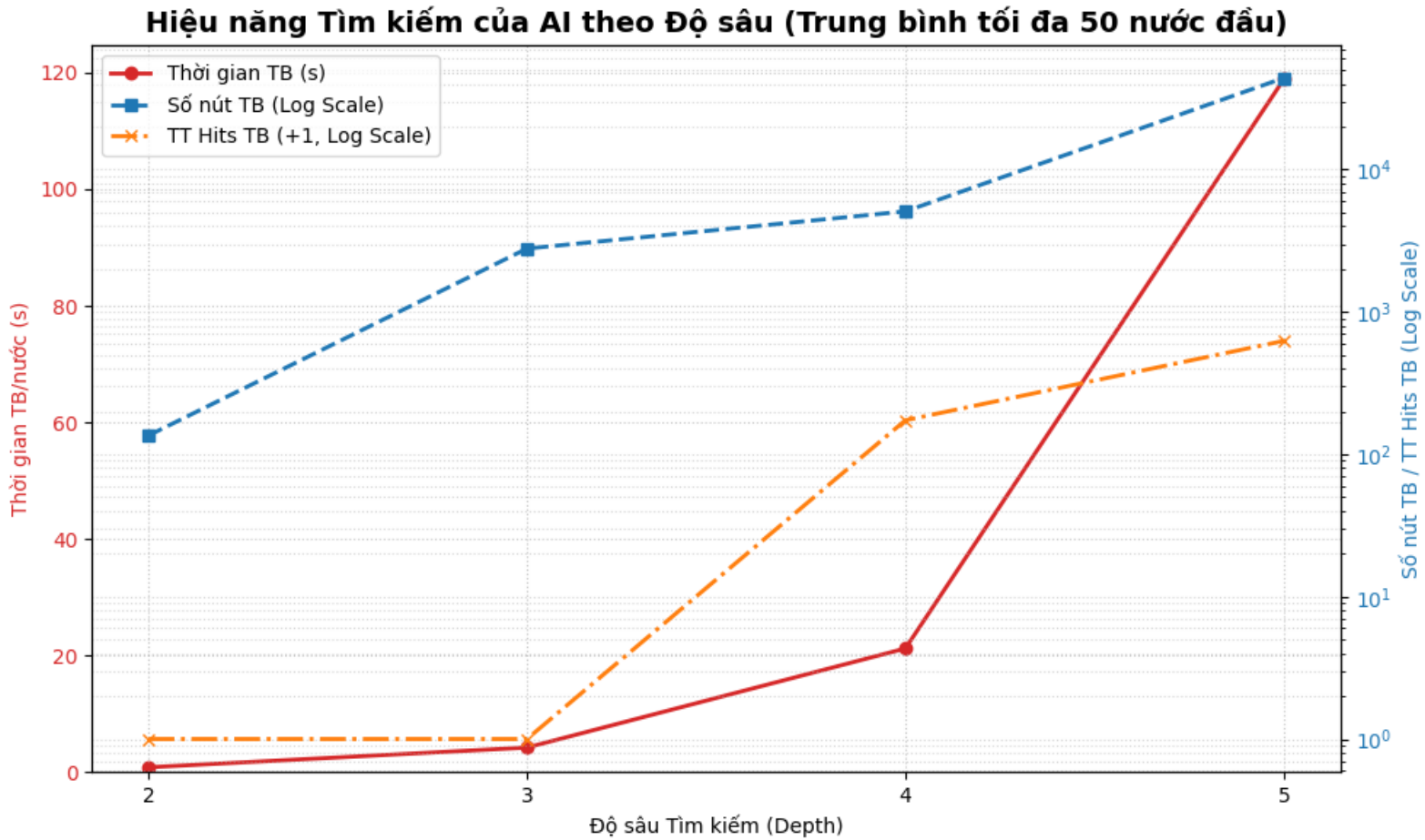
Depth	Time/M (s)	Nodes (k)	QNodes (k)	TT Hits	Moves Eval
2	0.82	0.14	0.42	0.0	10
3	4.20	2.79	3.27	0.0	15
4	21.21	5.08	10.71	172.3	11
5	118.87	44.00	61.45	625.2	9

Ghi chú:

- `Nodes/QNodes (k)`: Nút duyệt (nghìn).
- `Moves Eval`: Số nước đi AI thống kê.

✓ *YC 4 - Chỉ dùng Searching: Đảm bảo.*

Kết quả: Hiệu năng Tìm kiếm (Biểu đồ)



Hình: Hình 4: Hiệu năng theo độ sâu (Thời gian, Số nút - thang log).

Kết quả: Hiệu năng Tìm kiếm (Phân tích)

Phân tích từ Bảng 2 & Hình 4:

- **Scaling:** Time/Nodes tăng mũ theo Depth $\rightarrow$ tìm kiếm sâu tốn kém.
- **TT Effectiveness:** TT Hits tăng mạnh $D \geq 4 \rightarrow$ quan trọng khi cây lớn.
- **QS Impact:** QNodes chiếm tỷ trọng lớn ($>Nodes\ D = 5$) $\rightarrow$ cần cho phân tích chiến thuật sâu.
- **ID & Time Limit:** Time D5 ($\sim$119s) $\rightarrow$ ID quan trọng để quản lý thời gian.

Kết luận

Tóm tắt kết quả

- ✓ Xây dựng thành công engine Cờ Vua Minimax Agent.
- ✓ Agent đúng luật, thắng Random 10/10, có cấp độ khó.
- ✓ Tối ưu hóa (Alpha-Beta, IDS, TT, QS, MVV-LVA) hiệu quả.
- ✓ Hoàn thành mục tiêu BTL (chỉ dùng Searching).

Hạn chế & Hướng phát triển Tiềm năng

Hạn chế hiện tại

- Hàm lượng giá cơ bản.
- Thiếu tối ưu tìm kiếm nâng cao (SEE, Killers, History).
- Không có kiến thức chuyên sâu (Opening Book, Endgame Tables).

Hướng phát triển tương lai

- Nâng cấp Hàm lượng giá.
- Tích hợp Opening Book / Endgame Tables.
- Cải thiện Tối ưu hóa Tìm kiếm (Null Move, etc.).
- Tối ưu hiệu suất mức thấp (C++/Rust).

Thank you

Cảm ơn Thầy đã lắng nghe!

Câu hỏi & Thảo luận?