Bài tập cá nhân "Áp dụng các thuật toán tìm kiếm vào 8 quân xe"

Tên sinh viên: Trịnh Đại Nghĩa

Mã số sinh viên: 23110131

Môn học/lớp: Trí tuệ Nhân tạo/ARIN330585_05CLC

Ngày nộp: 15/10/2025

Link github: https://github.com/TrinhDaiNghia/BaiTapCaNhanAI

Search Algorithms Solve 8 Rooks

Tổng Quan

Ứng dụng này là một UI mô phỏng trực quan cách các thuật toán tìm kiếm đặt 8 quân xe cho bài toán 8-Rooks (quân xe), được xây dựng bằng ngôn ngữ lập trình Python với `Customtkinter` và `tkinter`. Người dùng có chọn 1 thuật toán tìm kiếm bất kỳ trong 5 nhóm thuật toán: Uninformed, Informed, Local, Non-deterministic, Constraint Satisfaction Problem.

---

1. Mục đích xây dựng

Bài tập cá nhân này nhằm mục đích ứng dụng những kiến thức đã học về các thuật toán tìm kiếm trong môn học Trí tuệ Nhân tạo vào trò chơi đặt 8 quân xe. Có 5 nhóm thuật toán tìm kiếm được sử dụng, các thuật toán tìm kiếm được sử dụng xếp theo 5 nhóm sau:

- `Uninformed Search`: Nhóm thuật toán tìm kiếm không có thông tin gồm Breadth-First Search (BFS), Depth-First Search (DFS), Depth-Limited Search (DLS), Iterative Deeping Search (IDS)
- `Informed Search`: Nhóm thuật toán tìm kiếm có thông tin gồm Greedy Best-First Search, A* Search, Uniform Cost Search (UCS)
- `Local Search`: Nhóm thuật toán tìm kiếm cục bộ gồm Hill Climbing, Simulated Annealing, Beam Search, Genetic
- `Non-deterministic`: Nhóm thuật toán tìm kiếm trong môi trường phức tạp: AND-OR Search, Belief State Search, Comformant Search
- `Constraint Satisfaction Problem Search`: Nhóm thuật toán tìm kiếm có ràng buộc: Backtracking, Forward-Checking, Look-Ahead (AC-3)

---

2. Tính năng chính

- Giao diện trực quan: Hai bảng 8x8 được hiển thị song song:
  - Board: vẽ quá trình tìm kiếm cách đặt quân.
  - Goal Board: hiển thị trạng thái đích ngẫu nhiên (goal state) cho một vài thuật toán cần thiết. Goal Board sẽ sinh ngẫu nhiên với mỗi lần chạy chương trình.

- Hiển thị kết quả thuật toán: Các quân xe sẽ được vẽ dần sau mỗi bước duyệt ngầm, chi phí - cost sẽ được cập nhật ngay sau mỗi bước đặt quân và thời gian - time sẽ được cập nhật mới khi chạy xong một quá trình đặt quân.
- Các thuật toán được sử dụng:
  - `Uninformed Search`: Breadth-First Search (BFS), Depth-First Search (DFS), Depth-Limited Search (DLS), Iterative Deeping Search (IDS)
  - `Informed Search`: Greedy Best-First Search, A* Search, Uniform Cost Search (UCS)
  - `Local Search`: Hill Climbing, Simulated Annealing, Beam Search, Genetic
  - `Non-deterministic`: AND-OR Search, Belief State Search, Comformant Search
  - `Constraint Satisfaction Problem Search`: Backtracking, Forward-Checking, Look-Ahead (AC-3)
- Nút Clear Board: Dùng để xóa toàn bộ quân xe trên Board và reset lại các lựa chọn thuật toán.
- Nút Run: Chạy thuật toán đã chọn để tiến hành quá trình đặt quân xe.

---

3. Yêu cầu

- Ngôn ngữ lập trình: `Python`
- Các thư viện được xài: `customtkinter`, `tkinter`, `math`, `random`.
- Thư viện cần cài đặt: Customtkinter Cài trong Command hoặc PowerShell: `pip install customtkinter`

---

4. Cách sử dụng

- Chạy ứng dụng có tên file: `23110131_TrinhDaiNghia_BaitapCanhan.py`
- Chọn thuật toán: có tất cả 5 combobox cho 5 nhóm thuật toán.
  - Chọn 1 thuật toán trong 5 nhóm thuật toán.
  - Khi chọn xong, các combobox còn lại sẽ tự động khóa
- Chạy thuật toán:
  - Nhấn nút `Run` để bắt đầu chạy thuật toán.
  - Quá trình vẽ đặt quân sẽ hiển thị trên Board (bàn cơ bên trái UI).
  - Cost sẽ cập nhật liên tục sau mỗi lần đặt quân. Sau khi hoàn tất sẽ hiển thị thời gian - Time chạy thuật toán.
- Xóa kết quả và chạy lại:
  - Nhấn `Clear Board` để xóa bàn cờ hiện tại và có thể chạy lại thuật toán hoặc chọn lại 1 thuật toán khác.

---

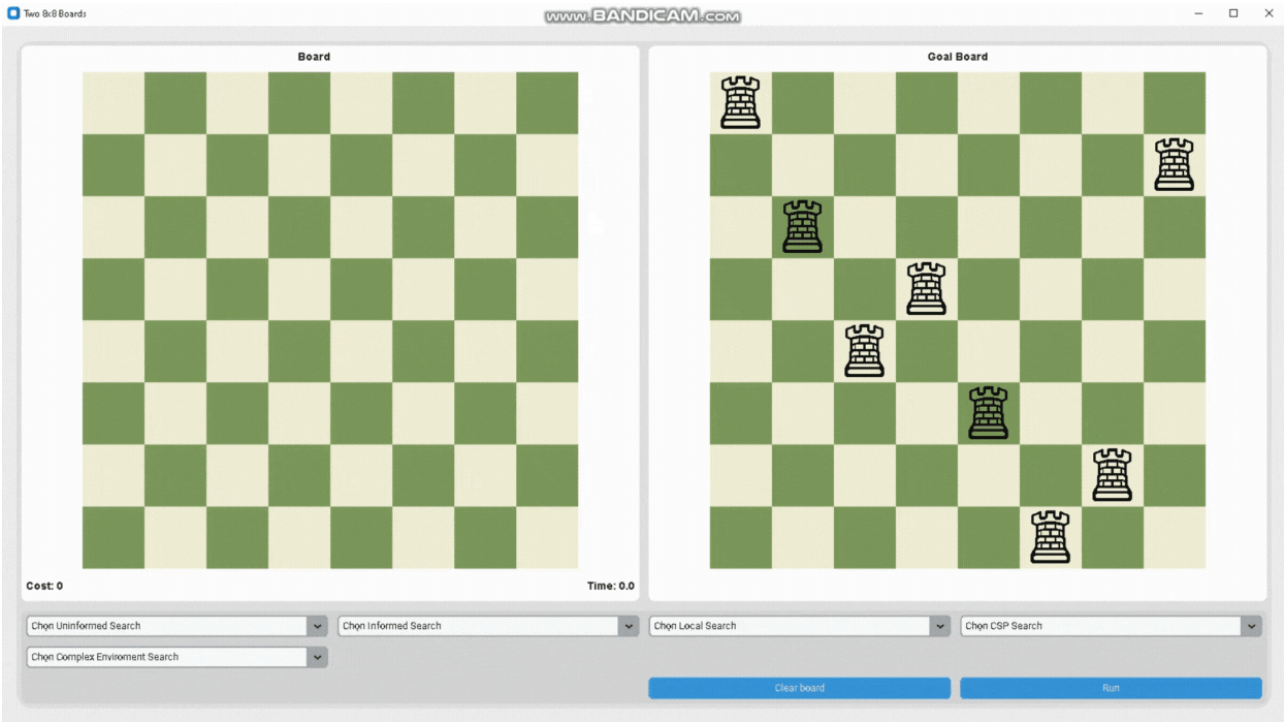
5. Cấu trúc file `23110131_TrinhDaiNghia_BaitapCanhan.py`

- Khởi tạo, cấu hình UI với `Customtkinter`.
- Vẽ bảng 8x8 gồm BoardA - bàn cờ dùng để vẽ và Goal Board - bàn cờ mục tiêu (goal state).
- Các hàm tiện ích: vẽ quân xe, update cost/time, clear board.
- Định nghĩa các hàm thuật toán tìm kiếm (BFS, DFS, UCS, A*, Hill Climbing, v.v...) và các hàm con cần thiết.
- Khởi tạo các comboBox dùng để chứa và xếp các thuật toán theo đúng nhóm thuật toán.
- Khởi tạo 2 nút `Run` và `Clear Board`

---

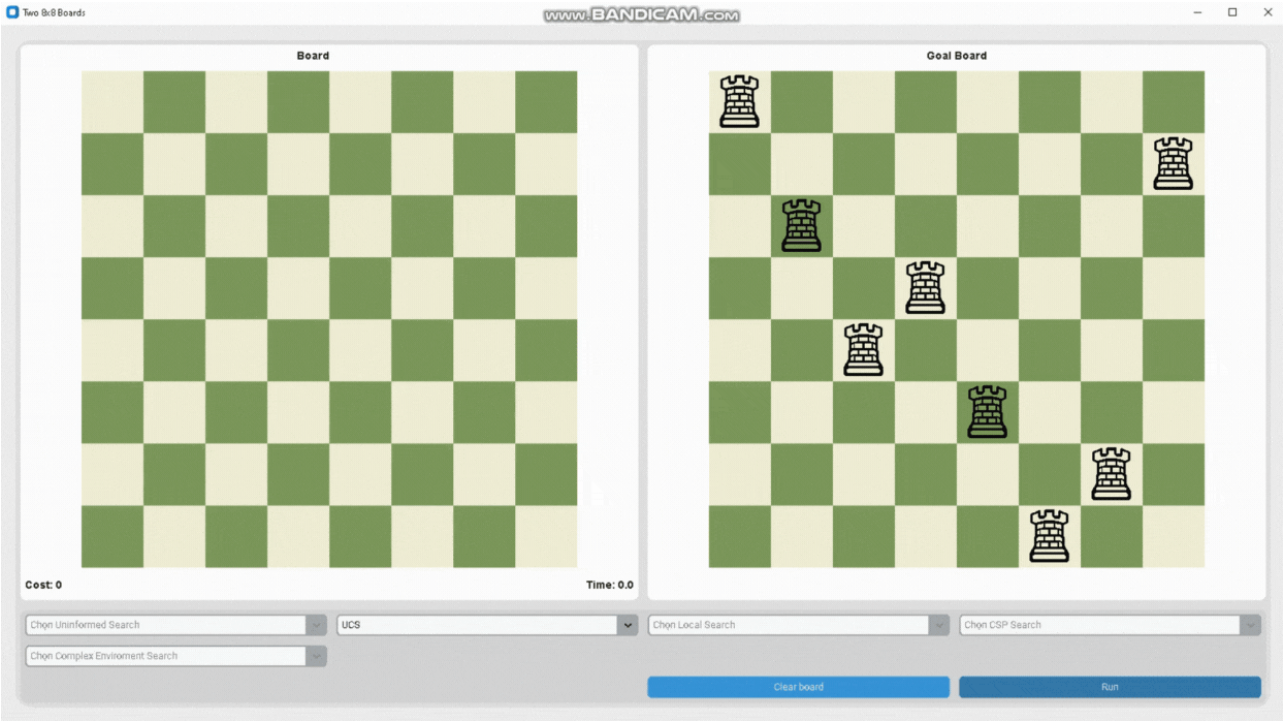
6. Kết quả chạy thử các thuật toán theo từng nhóm.

- Uninformed Search:



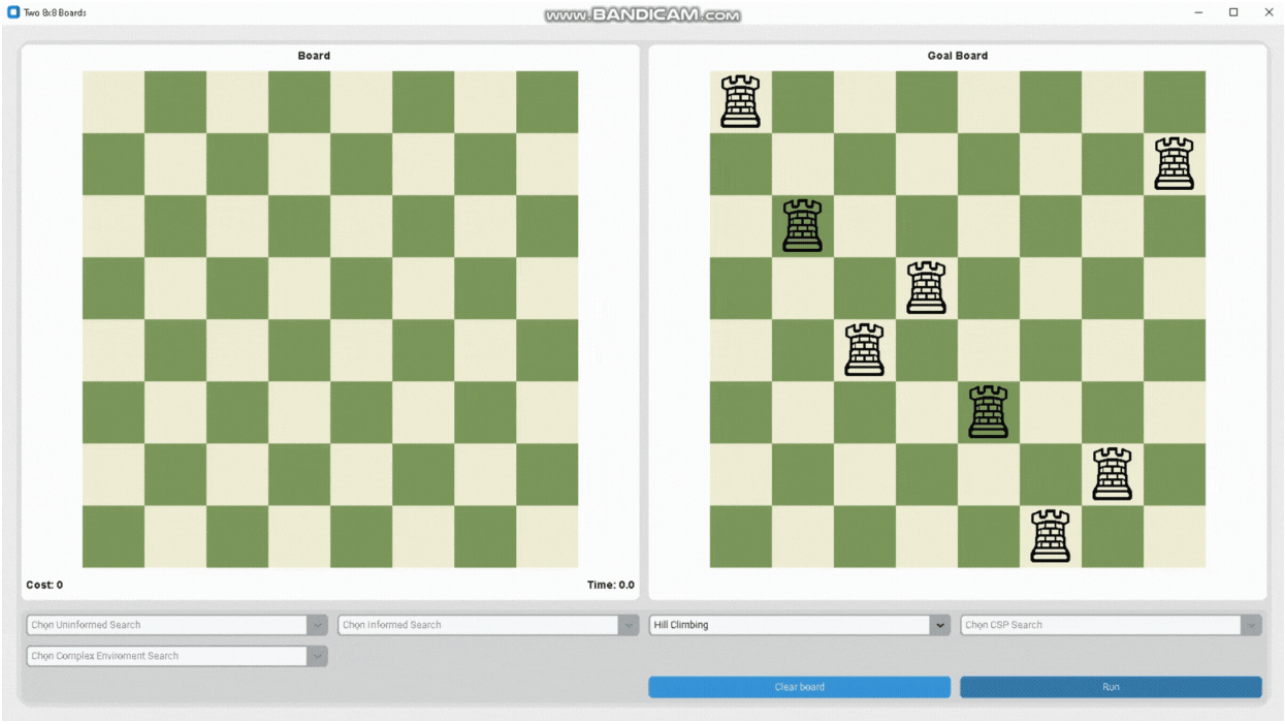
  - Đánh giá: Uninformed Search BFS, DFS, DLS, IDS (Không dùng bàn cờ goal):
    - Đặc trưng: Mở rộng trạng thái tuần tự (FIFO cho BFS, LIFO cho DFS/DLS/IDS), không sử dụng heuristic.
    - Steps Explored: BFS mở rộng nhiều trạng thái nhất nhưng tìm đường đi ngắn nhất về số bước. DFS/DLS/IDS ít steps hơn nhưng dễ rơi vào dead-end, đặc biệt DLS giới hạn sâu.
    - Path Length: BFS thường trả về đường đi ngắn nhất; DFS/DLS/IDS có thể trả về đường đi dài hơn. IDS thường là giải pháp tối ưu về độ dài nhưng có overhead do lặp lại các depth-limited search.
    - Path Cost: Mỗi bước được tính 1, do đó Path Cost ≈ Path Length.
    - Time Taken: BFS tốn nhiều thời gian với N lớn; DFS nhanh hơn nhưng không đảm bảo ngắn nhất; IDS có thời gian trung bình cao do lặp lại search.
    - Khả năng tìm giải pháp: BFS luôn tìm nếu tồn tại; DFS/DLS có thể fail nếu giới hạn sâu quá thấp; IDS đáng tin cậy với N nhỏ và trung bình.

- Informed Search:



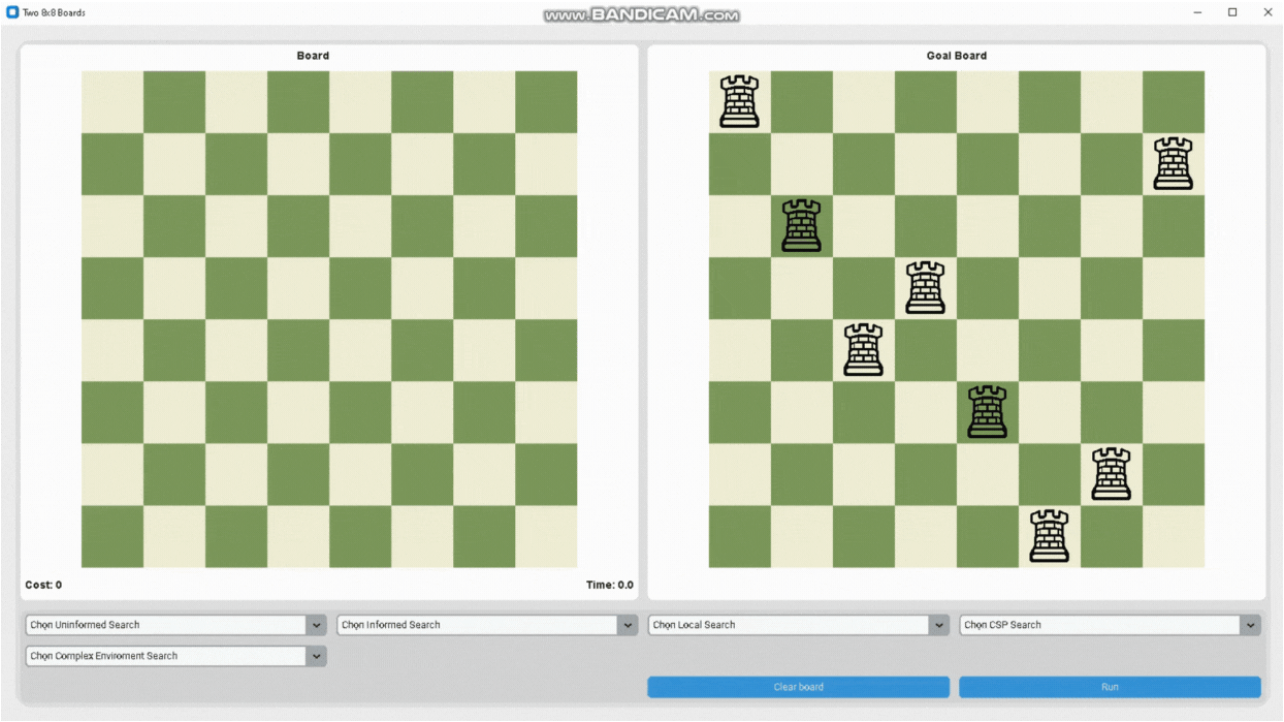
  - Đánh giá: Informed Search: UCS, Greedy, A* (Có dùng Cost, Heuristic, Goal), riêng UCS không dùng Goal:
    - Đặc trưng: Sử dụng cost (UCS) hoặc heuristic (Greedy, A*) để hướng tới trạng thái goal.
    - Steps Explored: UCS mở rộng trạng thái dựa trên tổng chi phí, ít thừa hơn BFS; Greedy mở rộng ít hơn nhưng không đảm bảo tối ưu. A* cân bằng giữa chi phí và heuristic, thường ít steps nhất trong nhóm này.
    - Path Length: UCS và A* thường cho path tối ưu hoặc gần tối ưu; Greedy đôi khi dài hơn.
    - Path Cost: UCS và A* tính chính xác theo hàm cost_UCS; Greedy chỉ dựa trên heuristic nên cost thực tế có thể cao hơn.
    - Time Taken: UCS và A* chậm hơn Greedy do tính toán chi phí/phương án nhiều hơn.
    - Khả năng tìm giải pháp: A* và UCS đảm bảo tìm solution; Greedy có thể bị stuck nếu heuristic không hoàn hảo

- Local Search:



  - Đánh giá: Nhóm Local Search: Hill Climbing, Simulated Annealing, Beam Search, Genetic Algorithm:
    - Đặc trưng: Không đảm bảo mở rộng toàn bộ trạng thái; dựa trên heuristic hoặc quần thể; đôi khi stochastic. Hill Climbing, Simulated Annealing không dùng goal, Beam Search, Genetic Algorithm có dùng
    - Steps Explored: HC ít steps nhưng dễ mắc local optimum; SA thêm stochastic để thoát local optimum, steps nhiều hơn; Beam Search giới hạn k-best, số steps kiểm soát được; GA phụ thuộc vào số thế hệ và size quần thể.
    - Path Length: HC/SA/Beam/GA thường tìm đường đi gần optimal nhưng không đảm bảo ngắn nhất.
    - Path Cost: Heuristic được cập nhật liên tục (update_Cost); GA sử dụng fitness/chi phí khác nhau.
    - Time Taken: HC nhanh; SA lâu hơn; Beam/GA tùy chỉnh được theo beam size/quần thể.
    - Khả năng tìm giải pháp: HC dễ fail nếu stuck; SA cải thiện khả năng tìm; Beam/GA khả năng tìm solution tốt nếu tham số hợp lý.

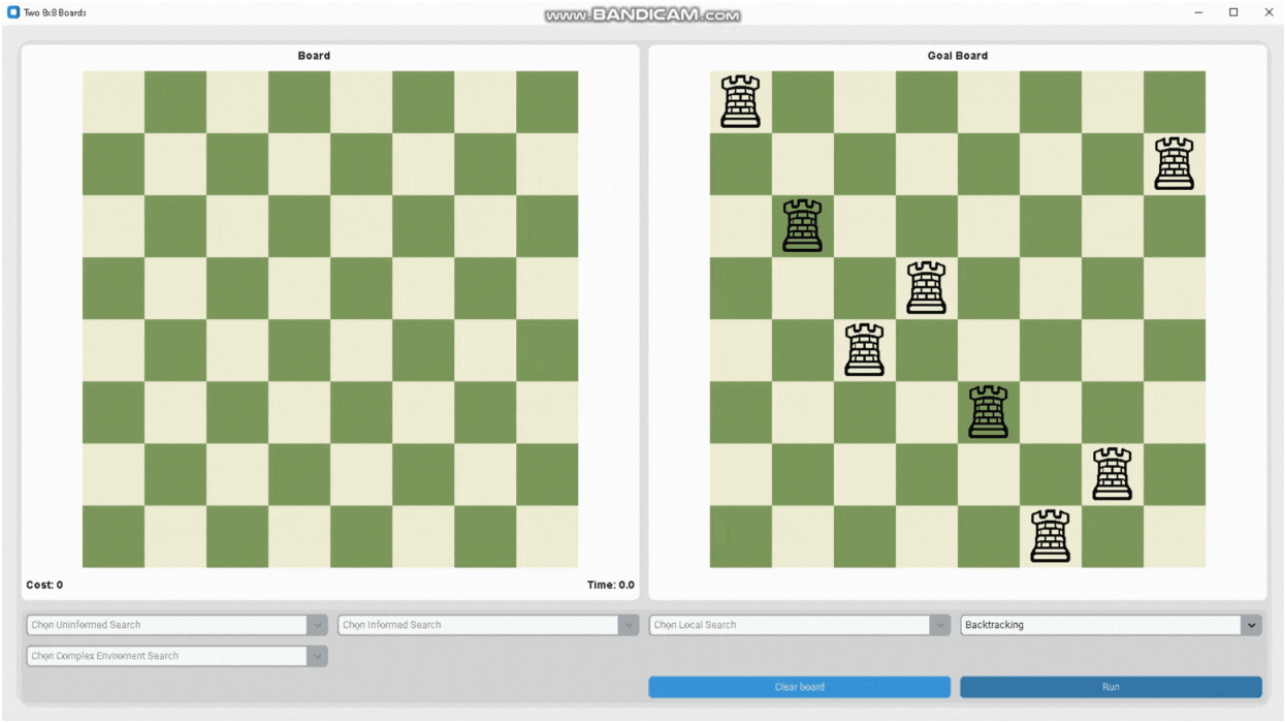
- Non-deterministic:



  - Đánh giá: Belief State Search, Conformant Search
    - Đặc trưng: Xử lý uncertain initial states (belief states), mở rộng tập hợp trạng thái.
    - Steps Explored: Lớn hơn các search thông thường do mở rộng từng belief, kết hợp nhiều state cùng lúc.
    - Path Length: Bằng N; nhưng animate từng step cho từng state trong belief.
    - Path Cost: Cập nhật dựa trên heuristic hoặc số bước apply action trên belief.
    - Time Taken: Cao nhất trong tất cả nhóm do combinatorial belief expansion.
    - Khả năng tìm giải pháp: Tìm solution nếu belief đủ "bao phủ" ban đầu; Conformant Search phụ thuộc sample size để tìm plan.
  - Đánh giá riêng: And-Or Search
    - Mục tiêu thử nghiệm: Tìm kế hoạch từ trạng thái khởi đầu đến goal board trong trò chơi 8 quân XE.
    - Kết quả quan sát:
      - Thuật toán không tìm ra solution trong các lần chạy thử (tất cả các state khởi đầu và belief được sinh ra).
      - Không phải do lỗi code, mà do bản chất AND–OR Search: khi AND node yêu cầu tất cả các outcome con phải dẫn tới solution, trong môi trường 8 quân XE với state được sinh ngẫu nhiên, hầu hết các nhánh không thỏa điều kiện AND → thuật toán dừng sớm.
    - Giải thích:
      - Trong source code, hàm OrSearch gọi AndSearch với tập các outcomes. Nếu bất kỳ outcome nào không có plan, toàn bộ nhánh AND bị loại.
      - Do mỗi state con được sinh ngẫu nhiên qua get_outcomes, xác suất tìm được một nhánh hoàn chỉnh dẫn tới goal board gần như bằng 0.
      - Đây là đặc điểm logic của AND–OR: phải đảm bảo mọi nhánh con của AND node đều thành công → khó tìm solution trong trò chơi 8 quân XE.
    - Kết luận thử nghiệm:

    - And-Or Search không chạy thành công trong môi trường này, nhưng phản ánh đúng bản chất kế hoạch AND–OR, thích hợp cho các bài toán planning có các action deterministically đảm bảo tất cả outcome thành công.
    - Hiệu năng không thể đánh giá bằng steps, path length hay time, vì thuật toán dừng trước khi sinh ra bất kỳ plan hợp lệ nào

- Constraint Satisfaction Problem Search:



    - Đánh giá: Nhóm CSP: Backtracking, Forward Checking, Look-Ahead/AC3 (Không dùng Goal)
        - Đặc trưng: Xử lý bài toán ràng buộc (constraint) trực tiếp, đảm bảo không vi phạm quy tắc 8 quân XE.
        - Steps Explored: Backtracking mở rộng nhiều trạng thái nếu không prune; FC giảm steps nhờ forward checking; Look-Ahead (MAC/AC3) prune mạnh mẽ, steps ít hơn hẳn.
        - Path Length: Luôn bằng N (đặt đủ 8 quân).
        - Path Cost: Path Cost = N hoặc số bước duyệt; tính đúng theo số bước đặt quân.
        - Time Taken: Backtracking lâu nhất, FC nhanh hơn; Look-Ahead nhanh nhất nhờ prune mạnh.
        - Khả năng tìm giải pháp: Tất cả đảm bảo tìm solution nếu tồn tại, Look-Ahead ổn định nhất với N lớn

---

7. Định hướng phát triển

Dựa trên kết quả thực nghiệm, có thể xác định một số hướng phát triển tiềm năng cho các thuật toán và hệ thống giải quyết trò chơi 8 quân XE:

- Cải thiện hiệu năng của Uninformed Search: Dù BFS và IDS đảm bảo tìm lời giải, chi phí bộ nhớ và số bước duyệt vẫn cao. Việc kết hợp pruning thông minh hoặc heuristic nhẹ có thể giảm đáng kể số trạng thái cần kiểm tra mà vẫn giữ tính đảm bảo của thuật toán.
- Tối ưu hóa Heuristic / Informed Search: Greedy, A*, Beam Search có thể được cải thiện bằng heuristic kết hợp nhiều yếu tố, ví dụ như số xung đột, khoảng cách từ cột hiện tại đến vị trí hợp lý, hoặc cân

nhắc chi phí bước đi. Beam Search có thể thử nghiệm với chiều rộng beam biến thiên theo độ sâu, giúp cân bằng giữa tốc độ và độ phủ của không gian trạng thái.

- Local Search nâng cao: HC và SA có thể kết hợp restart nhiều lần hoặc hybrid với GA để tăng khả năng thoát khỏi cực trị cục bộ. GA có thể thử nghiệm cơ chế đột biến / lai ghép thích nghi dựa vào trạng thái hiện tại, giúp tăng hiệu quả tìm kiếm trong không gian trạng thái lớn hơn (ví dụ 16–32 quân XE).
- CSP nâng cao với ràng buộc động: Forward Checking và Look-Ahead có thể được mở rộng bằng constraint propagation nâng cao, kết hợp với heuristic most-constrained-variable hoặc least-constraining-value để giảm số bước duyệt và tăng khả năng mở rộng sang CSP lớn hơn.
- Khám phá phi xác định và Belief-Based Search: Nhóm And-Or và Belief State Search mở ra hướng nghiên cứu lập kế hoạch phi xác định, robust plan. Đặc biệt, And-Or Search có thể được cải tiến bằng việc tối ưu hóa sinh trạng thái OR và AND nodes, hoặc kết hợp với heuristic đánh giá "độ khả thi" của các nhánh AND–OR, giúp giảm tình trạng dead-end mà vẫn giữ đúng logic AND–OR. Belief State Search có thể thử nghiệm với sampling belief lớn hơn, hoặc kết hợp probabilistic reasoning để tăng khả năng tìm ra kế hoạch tối ưu trong môi trường không chắc chắn.
- Hệ thống: Ngoài ra, hệ thống có thể cải thiện them bằng việc tạo them 1 log History, lưu lại các bước duyệt và đặt quân theo dạng text, cho phép người dùng điều chỉnh N quân tùy ý với N từ 1 đến 8.

Nhìn chung, các định hướng trên không chỉ giúp tăng hiệu quả và tốc độ thực thi, mà còn mở rộng khả năng ứng dụng sang các bài toán CSP phức tạp, lập kế hoạch phi xác định, và các bài toán Local / Metaheuristic trong không gian trạng thái lớn hơn. Đây chính là nền tảng để xây dựng một hệ thống giải thuật mạnh mẽ, linh hoạt và gần thực tế hơn cho trò chơi 8 quân XE cũng như các bài toán mở rộng tương tự.

Cuối cùng em xin cảm ơn Thầy/Cô/Các anh chị đã xem qua bài làm của em. Chúc Thầy/Cô/Các anh chị sức khỏe và thành công. Em xin cảm ơn!
