Thuật toán tìm kiếm A* và ứng dụng trong trò chơi Sokoban

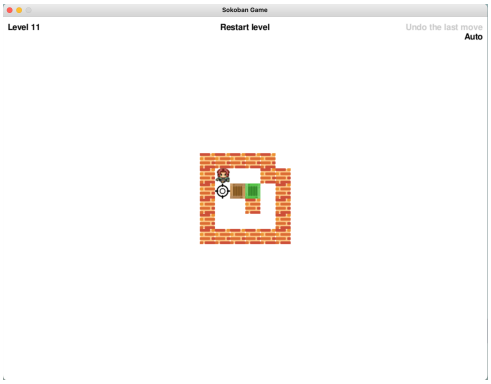
Bùi Minh khoa

Tháng 3/2024

1 Trò chơi Sokoban và định nghĩa thuật toán tìm kiếm A*

Sokoban là trò chơi giải đố logic trong đó người chơi phải điều khiển một nhân vật đi khắp mê cung để đưa các thùng hàng về vị trí yêu cầu.

Như đã đề cập ở bài báo cáo về các thuật toán Uninformed Search, trong đó việc tìm kiếm các trạng thái tiếp theo để mở ra sẽ được giả định không phụ thuộc vào thông tin cho trước, chính vì thế đây là điểm yếu của các thuật toán Uninformed Search. Đó là lúc thuật toán tìm kiếm A* - hay thuật toán Informed Search sẽ khắc phục điểm yếu và tăng hiệu suất tìm kiếm.



Hình 1: Trò chơi Sokoban

Để bắt đầu, quay lại thuật toán UCS, là một thuật toán Uninformed Search, UCS sẽ mở node có chi phí đường đi ngắn nhất. Dù là một thuật toán chắc chắn sẽ tìm ra kết quả, nhưng do nhược điểm không có manh mối về hướng đi đúng nên sẽ không phải là thuật toán có độ tối ưu về mặt thời gian.

Thuật toán Greedy - tham lam là thuật toán tìm kiếm sẽ chọn node tiếp theo để mở dựa trên một tiêu chí nào đó tối ưu, cụ thể ở đây chính là khoảng cách tuyệt đối - có thể được tính bằng khoảng cách Manhattan hay Euclidean. Greedy Search sẽ luôn mở node có độ lớn khoảng cách ngắn nhất từ điểm hiện tại đến đích. Song, Greedy Search vẫn có nhược điểm đó là ta chỉ xem xét khoảng cách chứ không xem xét đến yếu tố độ lớn đường đi.

Thuật toán A* Search sẽ là sự kết hợp giữa UCS và Greedy Search, trong đó, A* sẽ luôn chọn node có giá trị f(n) = g(n) + h(n) với hàm g(n) là chi phí đường đi - cùng cách tính với cost trong UCS và h(n) là heuristics - một thông tin ám thị để hướng cho thuật toán có thể tìm được điểm tối ưu tốt hơn - sẽ được tính bằng khoảng cách Manhattan hay Euclidean giống như Greedy Search.

Cụ thể hơn trong trò chơi Sokoban, với UCS ta sẽ cài đặt cost là số bước không di chuyển thùng. Thuật toán tìm kiếm A* với hàm g(n) là chi phí đường đi sẽ được cài đặt tương tự với cost của UCS. Còn với h(n) - hàm heuristic của A*, sẽ được cài đặt bằng khoảng cách Manhattan và Euclidean giữa các hộp chưa được đặt vào vị trí đích và giữa các vị trí đích chưa có hộp nào được đặt vào. Thuật toán UCS chính là trường hợp đặc biệt của A* với hàm h(n) = 0. Chính vì thế, heuristics là yếu tố cực kì quan trọng ảnh hưởng đến hiệu suất của thuật toán tìm kiếm A*.

2 Hàm Heuristics và các đặc điểm

2.1 Hàm Heuristics và sự đánh đổi giữa độ tối ưu và thời gian

Như đã đề cập ở trên, thuật toán tìm kiếm A* có hàm heuristics là yếu tố rất quan trọng, vì nó đóng vai trò như chỉ thị giúp tìm được lời giải nhanh chóng và tối ưu về mặt thời gian hơn. Hai hàm heuristics được sử dụng trong bài báo cáo này là Manhattan Distance và Euclidean Distance.

Khi hàm h(n) = 0, f(n) sẽ chính là UCS, đồng thời có thể tìm ra lời giải gần với lời giải tối ưu. Trong khi đó khi h(n) » 0, h(n) sẽ lấn át g(n) và khiến f(n) hoạt động gần giống Greedy Search, lúc đó lời giải sẽ nhanh chóng được tìm ra nhưng có khả năng cao sẽ không tối ưu.

2.2 Các hàm Heuristics

Như đã đề cập ở trên, hàm Heuristics được lựa chọn trong bài báo cáo này là Manhattan Distance và Euclidean Distance. Dù hai hàm Heuristics này có sự tương đồng nhất định song cũng có sự chênh lệch về tốc độ do việc tính căn ở Euclidean có thể mất nhiều thời gian.

Ngoài việc cải thiện hiệu suất bằng các thuật toán tìm kiếm cao cấp hơn, ta còn có thể sử dụng Hungarian algorithm như hàm Heuristics cho A*. Hungarian algorithm là thuật toán giải quyết bài toán phân chia công việc, song việc phân chia cũng lại phụ thuộc vào chi phí từ các thùng đến đích - hay khoảng cách, chính là hàm Heuristics hiện tại đang sử dụng, chính vì lý do đó Hungarian sẽ phù hợp để kết hợp với các thuật toán khác cao cấp hơn.

3 So sánh hiệu suất và nhận xét độ tối ưu ở mỗi màn chơi với UCS và A*

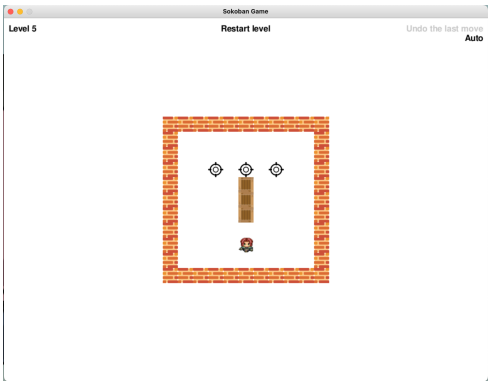
Thực nghiệm được thực hiện trên chip Apple M2, 16GB RAM

Bảng 1: So sánh hiệu suất của UCS, A*

Level	Manhattan	Euclidean	UCS
1	0.01s, 15	0.01s, 15	0.08s, 12
2	0.01s, 9	0.01s, 9	0.01s, 9
3	0.01s, 15	0.01s, 15	0.12s, 15
4	0.01s, 7	0.01s, 7	0.01s, 7
5*	0.05, 28	0.05, 28	53.45s, 20
6	0.01s, 19	0.01s, 19	0.03s, 19
7	0.05s, 25	0.04s, 25	0.60s, 21
8	0.13s, 99	0.13s, 99	0.25s, 99
9	0.00s, 8	0.00s, 8	0.02s, 8
10	0.02s, 33	0.02s, 33	0.02s, 33
11	0.01s, 34	0.02s, 34	0.03s, 34
12	0.01s, 23	0.01s, 23	0.10s, 23
13	0.04s, 45	0.04s, 45	0.20s, 31
14*	0.05s, 31	0.05s, 55	2.78s, 23
15	0.24s, 107	0.24s, 107	0.30s, 105
16	0.49s, 38	0.97s, 38	13.87s, 34
17	24.7s, 0	24.61s, 0	error
18	error	error	error

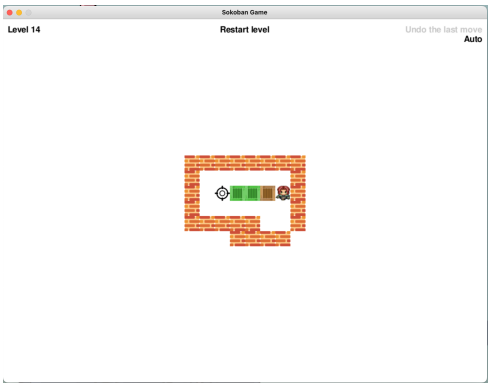
Giá trị đầu tiên là thời gian cần để chạy màn chơi trên máy tính thực nghiệm, giá trị thứ hai là số node được mở ra trong màn chơi.

Thông qua bảng thống kê trên, ta thấy được hiệu suất của mỗi thuật toán. Việc kết hợp Heuristics và UCS đã giúp tăng mạnh về mặt hiệu suất tìm kiếm lời giải. Ví dụ như màn chơi số 5, màn chơi có rất nhiều khoảng trống, việc sử dụng chi phí đường đi làm tiêu chí duy nhất để chọn node tiếp theo để mở như UCS sẽ khiến thuật toán tốn rất rất nhiều thời gian. Trong khi đó với A*, kết hợp chi phí đường đi và khoảng cách từ thùng đến đích đã giúp thuật toán hoàn thành nhanh hơn gấp nhiều lần.



Hình 2: Màn chơi số 5

Với màn số 14, dù Manhattan và Euclidean là 2 hàm Heuristics có bản chất rất giống nhau, nhưng lại cho ra số node đã mở là chênh lệch rất lớn: 31 so với 55.



Hình 3: Màn chơi số 14

Bảng 2: Độ tối ưu ở mỗi màn của UCS và A*

Level	**Manhattan**	**Euclidean**	**UCS**
1	Không	Không	Có
2	Có	Có	Có
3	Có	Có	Có
4	Có	Có	Có
5	Không	Không	Có
6	Có	Có	Có
7	Không	Không	Có
8	Có	Có	Có
9	Có	Có	Có
10	Có	Có	Có
11	Có	Có	Có
12	Có	Có	Có
13	Không	Không	Có
14	Không	Không	Có
15	Không	Không	Có
16	0.49s, 38	Không	Có
17	Có	Có	error
18			

Qua bảng thống kê về độ tối ưu của mỗi thuật toán, ta thấy thực nghiệm này càng làm rõ được sự đánh đổi về độ tối ưu và thời gian cảu Heuristics. Với UCS, thuật toán sẽ tìm ra lời giải gần tối ưu nhất, ví dụ như trong thực nghiệm này UCS đã chiến thắng A* về mặt số bước tối ưu. Khi hàm Heuristics được bổ sung vào, ta thấy được sự đánh đổi về mặt tối ưu song thời gian lại ngắn hơn rất rất nhiều ở những màn chơi phức tạp.

Màn 18 vẫn là màn chơi mà A* không thể tìm ra được lời giải vì độ phức tạp của nó. Để có thể giải được màn chơi khó như thế, ta có thể áp dụng các thuật toán cao cấp hơn.