ĐẠI HỌC QUỐC GIA HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Báo cáo

Heuristic và A* Search

GVHD: **Lương Ngoc Hoàng**

Họ tên SV: **Đặng Thanh Ngân**

MSSV: **22520929**

Lớp: **CS106.O22**

Hồ Chí Minh, ngày 26 tháng 3 năm 2024

Mục lục

I. Giới thiệu thuật toán A*

A* là thuật toán cải thiện hiệu năng từ thuật toán Greedy Best First Search. Khi greedy best-first search chỉ tìm đường dựa trên hàm ước lượng (Heuristic) từ điểm kế tiếp đến đích mà không tính độ dài quãng đường mà nó đã đi vì có thể ước lượng là nhỏ nhưng đường đã đi lại quá lớn.

Ý tưởng : tránh việc xét các nhánh tìm kiếm đã xác định cho đến thời điểm hiện tại là có chi phí cao.

Sử dụng hàm đánh giá

f(n) = h(n) + g(n)

h(n) : là ước lượng từ nút hiện tại tới nút đích

g(n) : chi phí từ nút gốc tới nút hiện tại n

f(n) : là chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích.

Một số điểm mạnh và điểm yếu của thuật toán A*:

Ưu điểm:

-Completeness: A* được đảm bảo tìm ra giải pháp nếu có, với điều kiện là nó khám phá không gian tìm kiếm một cách thông minh và toàn diện.

-Optimality: Khi hàm heuristic được sử dụng trong A* admissible (không bao giờ đánh giá quá cao chi phí để đạt được mục tiêu), thì A* là tối ưu, nghĩa là nó sẽ luôn tìm ra con đường ngắn nhất.

-Efficiency: So với các thuật toán tìm kiếm như tìm kiếm theo chiều sâu hoặc tìm kiếm theo chiều rộng, A* hiệu quả hơn nhiều về độ phức tạp về thời gian và không gian. Điều này là do nó sử dụng phương pháp phỏng đoán để hướng dẫn việc tìm kiếm hướng tới mục tiêu, trước tiên hãy tập trung vào những con đường hứa hẹn nhất.

-Flexibility: A* có thể được tùy chỉnh và điều chỉnh cho phù hợp với nhiều lĩnh vực bài toán khác nhau bằng cách chọn các hàm heuristic thích hợp. Điều này cho phép nó được áp dụng cho nhiều ứng dụng, từ tìm đường trong trò chơi điện tử đến robot và hậu cần.

-Adaptability: A* có thể xử lý các loại đồ thị khác nhau, bao gồm đồ thị có trọng số và không có trọng số, đồ thị có hướng và vô hướng, cũng như các đồ thị có chi phí cạnh khác nhau.

Nhược điểm:

-Sự phụ thuộc chất lượng heuristic: Hiệu quả và tính tối ưu của A* phụ thuộc rất nhiều vào chất lượng của hàm heuristic được sử dụng. Nếu phương pháp heuristic không admissible hoặc quá lạc quan, A* có thể không tìm được giải pháp tối ưu hoặc có thể mất nhiều thời gian hơn để hội tụ.

-Memory Usage: A* lưu trữ thông tin về các nút đã truy cập trong bộ nhớ, điều này có thể trở thành hạn chế trong không gian tìm kiếm rất lớn. Trong trường hợp việc sử dụng bộ nhớ là mối lo ngại, các thuật toán tìm kiếm khác như IDA*hoặc các biến thể tìm kiếm giới hạn bộ nhớ có thể được ưu tiên hơn.

-Complexity: Mặc dù khái niệm cơ bản về A* rất đơn giản nhưng việc triển khai nó một cách chính xác với cấu trúc dữ liệu phù hợp và xử lý các trường hợp khó khăn có thể là một thách thức, đặc biệt đối với người mới bắt đầu.

-Không phải lúc nào cũng tốt nhất cho mọi tình huống: Mặc dù A* có hiệu quả cao trong nhiều tình huống nhưng vẫn có những trường hợp các thuật toán khác có thể hoạt động tốt hơn.

-Optimality vs Speed Trade-off: Trong một số trường hợp, việc hy sinh sự tối ưu để lấy tốc độ có thể là điều đáng mong muốn. A* đảm bảo tính tối ưu nhưng có thể chậm hơn so với các thuật toán như thuật toán Dijkstra, đặc biệt trong các trường hợp việc đánh giá heuristic tốn kém hoặc hàm heuristic không phù hợp cho vấn đề.

Tóm lại, mặc dù A* là một thuật toán mạnh mẽ và được sử dụng rộng rãi với nhiều ưu điểm, nhưng điều cần thiết là phải xem xét các hạn chế và tính phù hợp của nó đối với các bài toán cụ thể trước khi áp dụng nó. Việc xem xét cẩn thận các đặc điểm của vấn đề, chất lượng và các yêu cầu về hiệu suất là rất quan trọng để đạt được kết quả tốt nhất với A*.

II. So sánh UCS và A* áp dụng trong Sokoban. So sánh thuật toán A* khi áp dụng Heuristic khác nhau.

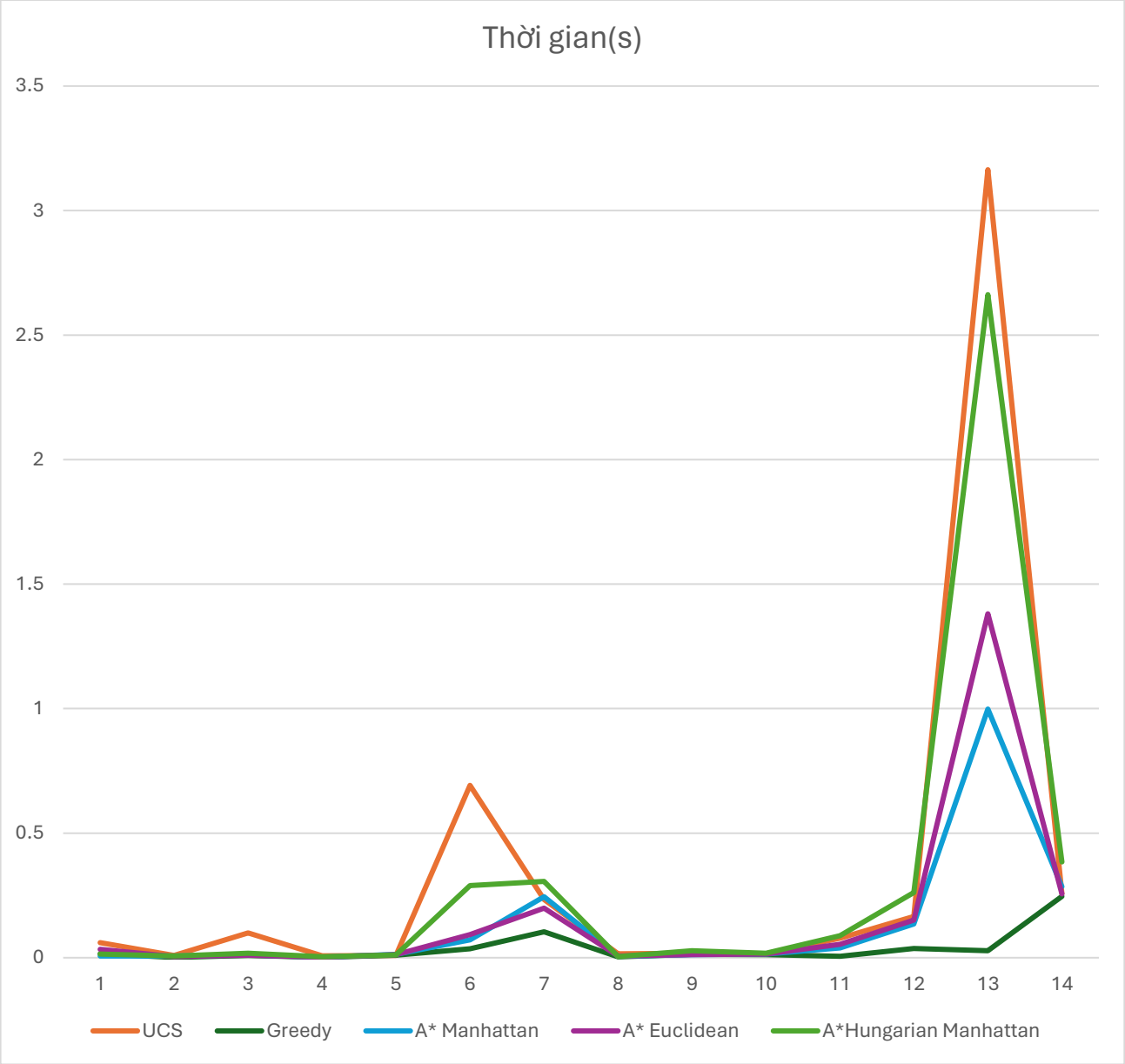
1. Giới thiệu áp dụng các thuật toán trong bài Sokoban

-UCS: Tìm đường đi có chi phí nhỏ nhất, thuật toán này sẽ vét hết các node cạnh, và chọn ra những node có đường đi ngắn nhất. Từ đó có thể tìm ra được đường đi ngắn nhất từ vị trí ban đầu của các box để đưa chúng đến đích.
-Greedy Best First Search (GFS): Thuật toán tìm kiếm tham lam, nó sẽ dựa vào hàm Heuristic (ước lượng khoảng cách từ node đến đích) để chọn ra những node ưu tiên được xử lý. Điều này sẽ giúp tìm được kết quả nhanh chóng, nhưng có thể không thể tìm ra được lời giải hoặc lời giải tối ưu cho bài toán. Thuật toán này phụ thuộc rất nhiều vào hàm Heuristic.
-A*: là thuật toán kết hợp giữa UCS và GFS, ưu tiên những node có chi phí nhỏ và có khoảng cách trông nhỏ nhất đến đích.

Trong Sokoban, bài toán liên quan đến việc tìm kiếm đường đi hay di chuyển một vật trong ma trận thì heuristic được tính theo Euclidean hoặc Manhattan là tối ưu. Nó đảm bảo tình admissible (0<=h(n)<=h*(n)).
-Euclidean là khoảng cách L2 từ các hộp đến đến đích. Manhattan là khoảng cách L1 từ các hộp đến đích.
-Trong code sẵn được cung cấp, việc tính 2 khoảng cách này từ các hộp ngẫu nhiên chưa được đặt đúng vị trí đến các đích. Để cải thiện việc này thì bài báo cáo này sẽ áp dụng thuật toán Hungarian để tìm 2 khoảng cách dựa vào những hộp gần với vị trí đích nhất. Thuật toán Hungarian phát biểu là có n công việc và từng n người làm n công việc với chi phí khác nhau, bài toán đặt ra là tìm cách chia mỗi người một công việc sao cho chi phí là nhỏ nhất. Áp dụng thuật toán này trong Sokoban, chúng ta sẽ tính hàm heristic dựa vào khoách cách Manhattan hay Eucliean sao cho chi phí là nhỏ nhất. Điều này sẽ đảm bảo tính admissible cho hàm (ước lượng chi phi đi tới đích luôn nhỏ hơn chi phí thật sự).
-Trong các đồ thị để minh họa kết quả khảo sát, đã bỏ những level gây nhiễu (level 5) và những level sau với thời gian giải lâu, phước tạp (level 18), hoặc không có lời giải có thể khiến cho các thuật toán bị đứng (level 17).

2. Thời gian

Level	*Box*	**Thời gian (s)**				
		UCS	Greedy	A* Manhattan	A* Euclidean	A*Hungarian Manhattan
1	2	0.06	0.0162	0.0065	0.0337	0.0142
2	2	0.0086	0.001	0.0065	0.004	0.0071
3	4	0.1001	0.0085	0.0111	0.0098	0.0185
4	1	0.0076	0.0015	0.002	0.0021	0.004
5	3	196.32	0.0365	0.0932	0.1043	0.1839
6	1	0.0095	0.0097	0.0134	0.0125	0.0114
7	2	0.692	0.0356	0.0714	0.0934	0.2892
8	2	0.2338	0.1037	0.2454	0.1991	0.3068
9	3	0.0153	0.003	0.0061	0.0067	0.005
10	2	0.0184	0.0134	0.0124	0.0121	0.0279
11	2	0.0159	0.0122	0.0149	0.0145	0.0184
12	2	0.0787	0.0055	0.0397	0.0538	0.0881
13	2	0.1656	0.0375	0.135	0.1515	0.2612
14	3	3.1642	0.0285	0.9984	1.381	2.6623
15	3	0.2572	0.2462	0.2849	0.256	0.3841
16	7	23.8839	Không tìm ra lời giải	0.3065	0.4558	3.7302
17	2	index out of range	30.2527	30.5327	30.38	46.1885
18	2	Máy đứng	Máy đứng	Máy đứng	Máy đứng	Máy đứng



- Nhận xét:

-Trong nhiều bài toán thì thuật toán Greedy tìm ra giải pháp nhanh nhất so với tất cả các thuật toán. Sau đó là A* Manhattan. Thuật toán UCS có thời gian giải lâu nhất trong các thuật toán, A* Manhattan áp dụng thuật toán Hungarian cũng chiếm nhiều thời gian.

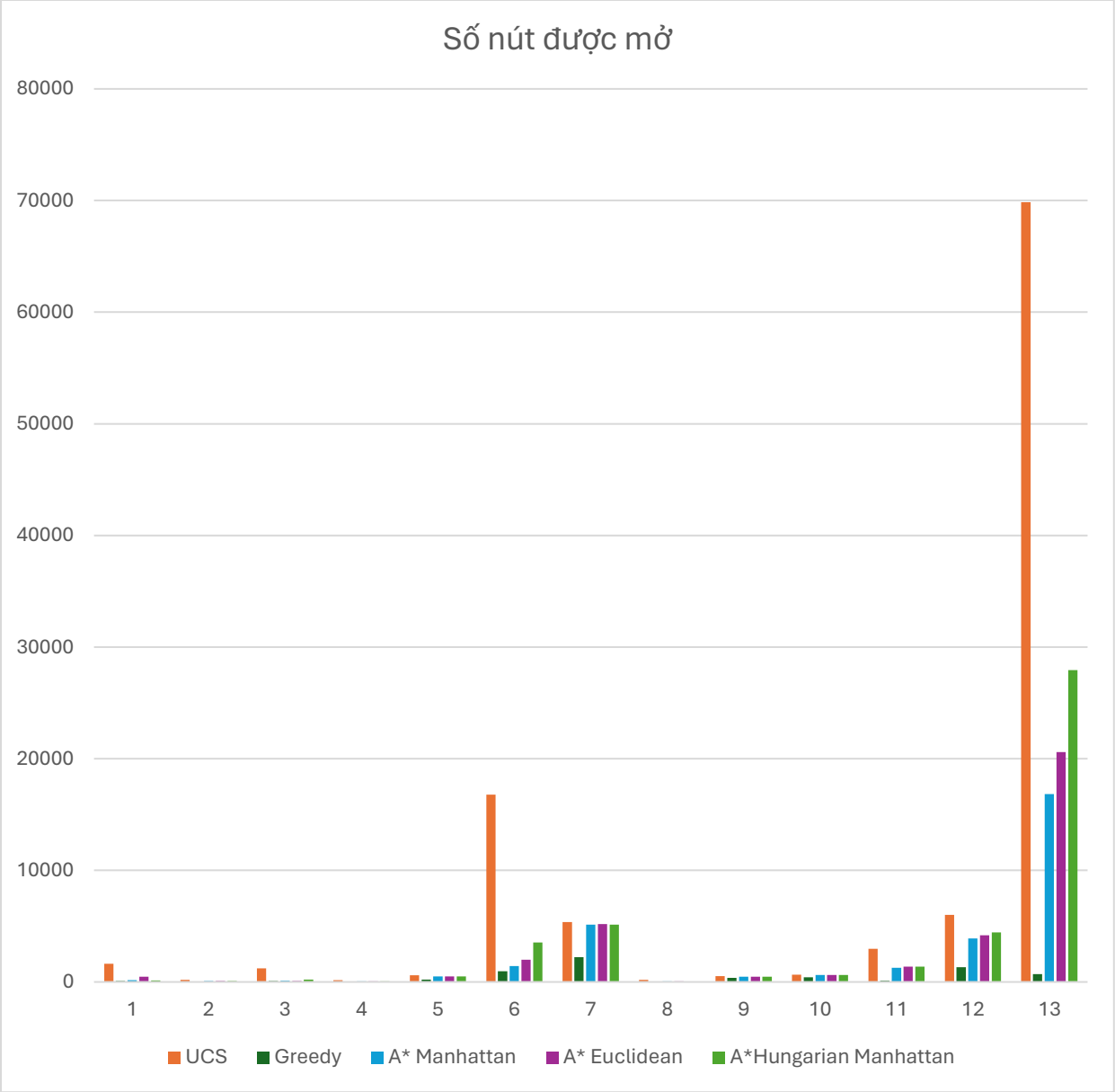
-Level 5 thuật toán UCS tìm lời giải rất lâu vì phải kiểm tra tất cả các node để tìm đường đi ngắn nhất và không gian lớn, nhiều box nên cần phải xử lý nhiều. Khi áp dụng hàm heuristic tìm ra được lời giải nhanh hơn rất nhiều vì nó giúp điều chỉnh thuật toán đi đúng hướng.

-Khi bài toán không có lời giải thì UCS bị tràng dữ liệu (index out of range).

3. Số nút đã được mở

Số nút đã được mở

Level	UCS	Greedy	A* Manhattan	A* Euclidean	A*Hungarian Manhattan
1	1634	83	171	463	128
2	181	40	83	83	87
3	1218	85	108	91	208
4	171	43	55	55	55
5	1278998	692	901	812	1155
6	595	210	489	505	489
7	16767	959	1416	1979	3533
8	5356	2219	5129	5172	5129
9	174	19	57	56	41
10	524	373	461	461	473
11	657	423	627	627	635
12	2966	98	1269	1363	1369
13	6011	1327	3902	4188	4435
14	69842	700	16837	20590	27937
15	6325	4354	5371	5528	5541
16	141814	Không tìm được lời giải	2722	3797	18151
17	Index out of range	169158	169158	169158	169158
18	Máy đứng	Máy đứng	Máy đứng	Máy đứng	Máy đứng



- Nhận xét:

-Tuy Greedy tìm ra được giải pháp rất nhanh và mở rộng ít node hơn hẳn các thuật toán khác nhưng level 16 các thuật toán đều có thể tìm ra được lời giải nhưng Greedy không tìm được.

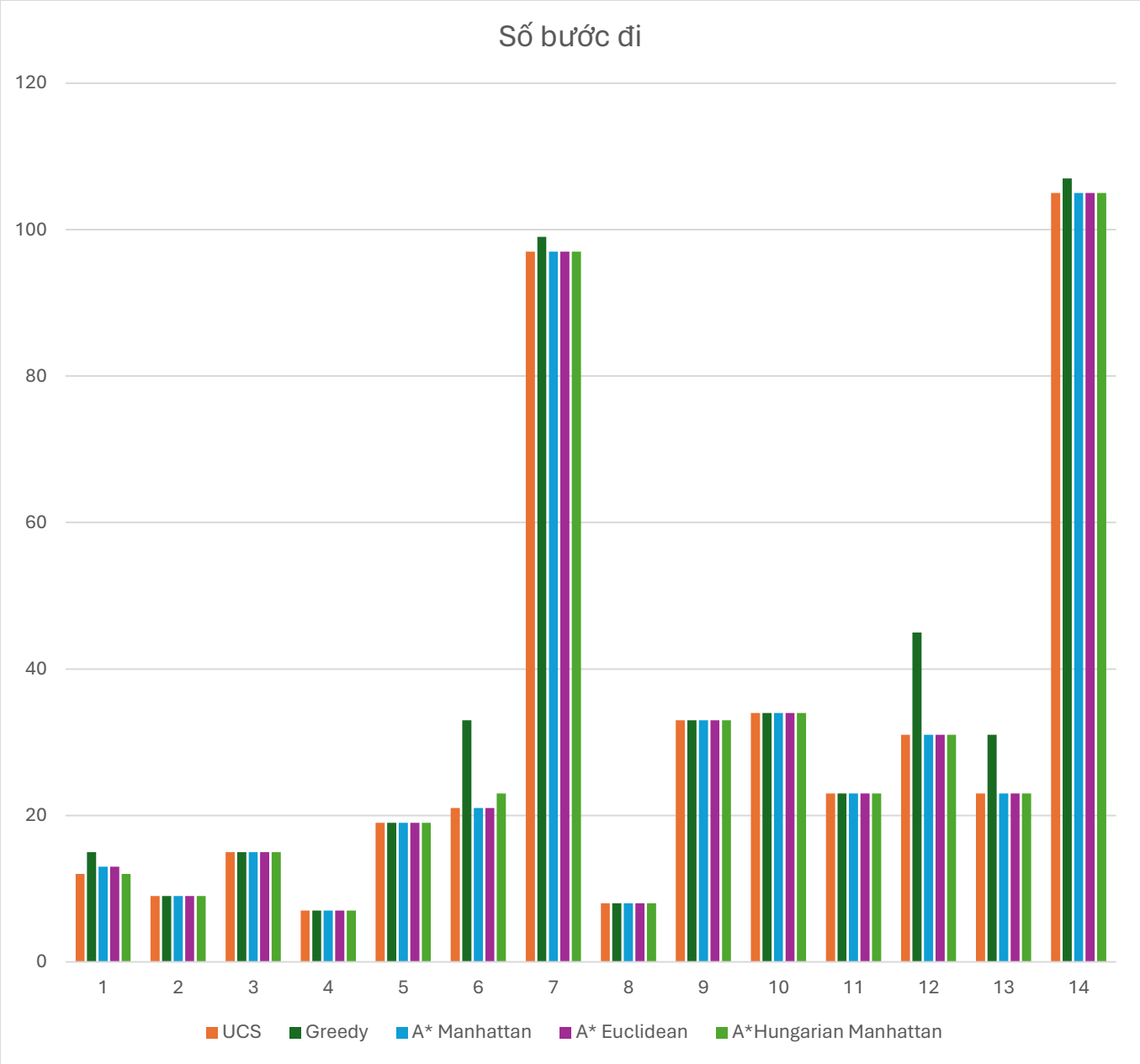
-Các thuật toán A* cần mở ít node hơn, trong đó thuật toán UCS cần mở rất nhiều node hơn hẳn các thuật toán A* khác (ví dụ level 7) vì nó không có định hướng nên cần phải mở hết các node để tìm ra được đích.

-Trong thuật toán A* thì khi áp dụng tính heuristic theo khoách cách Manhattan (xanh dương) giúp mở được ít node hơn các cách tính heuristic khác.

4. Số bước đi để đến đích

Số bước đi

Level	UCS	Greedy	A* Manhattan	A* Euclidean	A*Hungarian Manhattan
1	12	15	13	13	12
2	9	9	9	9	9
3	15	15	15	15	15
5	20	28	22	22	20
4	7	7	7	7	7
6	19	19	19	19	19
7	21	33	21	21	23
8	97	99	97	97	97
9	8	8	8	8	8
10	33	33	33	33	33
11	34	34	34	34	34
12	23	23	23	23	23
13	31	45	31	31	31
14	23	31	23	23	23
15	105	107	105	105	105
16	34	Không có lời giải (lặp vô hạn)	42	36	34
17	Index out of range	Không có lời giải	Không có lời giải	Không có lời giải	Không có lời giải
18	Máy đứng	Máy đứng	Máy đứng	Máy đứng	Máy đứng



- Nhận xét:
  -UCS tìm ra được đường đi có số bước ngắn nhất, ngược lại thì Greedy luôn có số bước lớn hơn hoặc bằng với số bước nhỏ nhất.
  -Trong hầu hết các level thì thuật toán UCS và các thuật toán A* đều tìm được đường đi bằng nhau.
  -Trong level 1 và 5 thì hai thuật toán A* không áp dụng Hungarian có số bước lớn hơn, trong khi đó A* Hungarian Manhattan tìm được đường đi bằng với đường đi ngắn nhất. Nhưng trong level 7 thì đường đi của thuật toán này không bằng với đường đi ngắn nhất.

5. Dung lượng bộ nhớ sử dụng

Dung lượng bộ nhớ (MB)

Level	UCS	Greedy	A* Manhattan	A* Euclidean	A*Hungarian Manhattan
1	59.71	58.77	58.85	59.26	59.3
2	60.02	58.96	59.09	59.45	59.55
3	60.48	59.23	59.42	59.69	59.84
5	145.24	59.83	60.96	61.11	62.29
4	60.3	59	59.24	59.46	59.61
6	60.32	59.42	60.47	60.69	61.9
7	66.19	59.86	60.79	61.41	62.49
8	66.26	60.2	61.71	61.83	62.8
9	66.44	59.82	61.43	62.01	62.97
10	66.45	59.94	61.51	62.02	62.99
11	66.46	59.95	61.48	62.07	63.03
12	66.46	59.95	61.54	62.09	63.05
13	66.47	60.18	61.96	62.11	63.07
14	80.86	60.24	70.45	72.62	75.73
15	80.92	60.88	70.96	72.48	75.92
16	168.8	Không có lời giải	70.66	72.59	77.25
17	Index out of range	97.28	98.35	97.95	99.13
18	Máy đứng	Máy đứng	Máy đứng	Máy đứng	Máy đứng

- Nhận xét:

-Nhìn chung, thuật toán Greedy tốn ít dung lượng nhất, sau đó là thuật toán A* khi áp dụng Manhattan tiếp theo là A* Euclidean và A* Manhattan áp dụng thuật toán Hungarian. Thuật toán UCS tốn kém dung lượng nhất, đặc biệt với những bản đồ có độ khó cao, phức tạp, cần duyệt nhiều node như level 5 thì cần nhiều dung lượng.

6. Rút ra kết luận

-Xét về các level thì thuật toán A* với hàm heuristic là khoảng cách Manhattan là tối ưu nhất. Tiết kiệm được nhiều thời gian, hạn chế được số node cần được mở vì vậy ít tốn bộ nhớ hơn. Và đường đi cũng gần bằng với đường đi ngắn nhất.

-Áp dụng thuật toán Hungarian làm cho bài toán thực hiện với thời gian lớn hơn. Vì phải tìm chi phí từ hộp đến đích nhỏ nhất trong các trường hợp. Điều này cũng cho thấy rằng không phải những tất vị trí đích gần gần hộp nhất là vị trí chính xác cho lời giải.