**Thực nghiệm & Báo cáo**

# 1. Bảng số liệu tổng hợp

• Bảng cho thuật toán BFS (UCS có cost = 1 )

| **Puzzle** | **Solution Path** | **Execution Time (s)** | **Expanded Nodes** | **Generated Nodes** | **Duplicated Nodes** | **Memory (bytes)** | **Pieces** | **Steps** | **Empty Spaces** | **Solver** | **Nodes/sec** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| capability1 | 0u | 0.000016 | 2 | 3 | 0 | 0 | 1 | 1 | 32 | IW(2) | 187,804.656716 |
| capability2 | 0r | 0.000024 | 3 | 6 | 0 | 0 | 1 | 1 | 34 | IW(2) | 167,772.160000 |
| capability3 | 0l | 0.000067 | 3 | 6 | 0 | 0 | 1 | 1 | 34 | IW(2) | 59,705.395018 |
| capability4 | 0d | 0.000031 | 2 | 2 | 0 | 0 | 1 | 1 | 32 | IW(2) | 96,791.630769 |
| capability5 | 0u0u | 0.000050 | 6 | 15 | 0 | 0 | 1 | 2 | 30 | IW(2) | 139,810.133333 |
| capability6 | 0l0u0u | 0.000223 | 21 | 64 | 0 | 0 | 1 | 3 | 30 | IW(2) | 98,689.505882 |
| capability7 | 0r0u0u | 0.000208 | 21 | 64 | 0 | 0 | 1 | 3 | 30 | IW(2) | 105,819.596330 |
| capability8 | 1l0u0u0u | 0.015640 | 723 | 4,087 | 0 | 0 | 2 | 4 | 28 | IW(3) | 46,292.205494 |
| capability9 | 1l2r0u0u0u | **-93.658692** | 8,308 | 57,867 | 0 | 0 | 3 | 5 | 26 | IW(4) | **-88.715738** |
| capability10 | 1l2r0u0u0u | 0.117329 | 3,962 | 25,029 | 0 | 0 | 4 | 5 | 20 | IW(5) | 33,776.780892 |

• Bảng cho thuật toán BFS cải tiến( dùng radix tree xử lí trùng )

| **Puzzle** | **Solution Path** | **Execution Time (s)** | **Expanded Nodes** | **Generated Nodes** | **Duplicated Nodes** | **Memory (bytes)** | **Pieces** | **Steps** | **Empty Spaces** | **Solver** | **Nodes/sec** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| capability1 | 0u | 0.000023 | 2 | 3 | 0 | 83 | 1 | 1 | 32 | IW(2) | 129,720.742268 |
| capability2 | 0r | 0.000032 | 3 | 5 | 1 | 148 | 1 | 1 | 34 | IW(2) | 125,203.104478 |
| capability3 | 0l | 0.000034 | 3 | 5 | 1 | 148 | 1 | 1 | 34 | IW(2) | 117,323.188811 |
| capability4 | 0d | 0.000023 | 2 | 2 | 0 | 50 | 1 | 1 | 32 | IW(2) | 129,720.742268 |
| capability5 | 0u0u | 0.000063 | 6 | 9 | 6 | 278 | 1 | 2 | 30 | IW(2) | 111,212.606061 |
| capability6 | 0l0u0u | 0.000148 | 12 | 16 | 19 | 504 | 1 | 3 | 30 | IW(2) | 87,803.465378 |
| capability7 | 0r0u0u | 0.000131 | 12 | 16 | 19 | 504 | 1 | 3 | 30 | IW(2) | 99,318.673953 |
| capability8 | 1l0u0u0u | 0.001849 | 94 | 156 | 349 | 5,075 | 2 | 4 | 28 | IW(3) | 51,380.900064 |
| capability9 | 1l2r0u0u0u | 0.010471 | 349 | 700 | 1,732 | 22,983 | 3 | 5 | 26 | IW(4) | 33,425.314784 |
| capability10 | 1l2r0u0u0u | 0.008468 | 239 | 479 | 1,070 | 16,067 | 4 | 5 | 20 | IW(5) | 28,342.285666 |
| capability11 | 0u0r1l2r3d3l0u0u3d0  l0u0u0u | 1.282861 | 28,478 | 37,785 | 210,229 | 1,233,868 | 4 | 13 | 23 | IW(5) | 22,199.599275 |
| capability12 | 0r1d1d1l0u0u0u0u0l0u0u | 0.004807 | 218 | 238 | 783 | 7,732 | 2 | 11 | 26 | IW(3) | 45,558.604107 |
| capability13 | 1d1d1r1r0u0u0u0u | 0.004775 | 220 | 278 | 849 | 9,036 | 2 | 8 | 30 | IW(3) | 46,282.264030 |
| impassable1 | 0l3r1d2l2l2l3r1r5d3d1r1u1r2r2  d2d1l1l1u2u2r3r4u0l4r4r0u0u5l5  l3d3d4r5l4d2d1d1r1r2r0r0u0u0r0u0u | 0.937662 | 19,328 | 21,325 | 84,942 | 718,688 | 6 | 46 | 13 | IW(7) | 20,614.040396 |
| impassable2 | 0l4u0l4r4r5d4d2d1r2l2l2d4u0r5u  0r0r2d2d2r4l2u4u2u0l0l5d2r2r  2r4d1d4l4d1d1r3l3l3u1u4u0  l4r4r0u0u5l5l2d2d4r5l4d1d1r0r3d3r0u0u3r0r0u0u | 2.561682 | 53,079 | 56,697 | 230,556 | 1,909,627 | 6 | 64 | 13 | IW(7) | 20,720.760927 |

* Bảng cho thuật toán IWS

| **Puzzle** | **Solution Path** | **Execution Time (s)** | **Expanded Nodes** | **Generated Nodes** | **Duplicated Nodes** | **Memory (bytes)** | **Pieces** | **Steps** | **Empty Spaces** | **Solver** | **Nodes/sec** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| capability1 | 0u | 0.000019 | 2 | 2 | 0 | 50 | 1 | 1 | 32 | IW(1) | 157,286.400000 |
| capability2 | 0r0u | 0.000038 | 4 | 4 | 4 | 115 | 1 | 2 | 34 | IW(1) | 131,896.352201 |
| capability3 | 0l0u | 0.000038 | 4 | 4 | 4 | 115 | 1 | 2 | 34 | IW(1) | 131,072.000000 |
| capability4 | 0d | 0.000021 | 2 | 2 | 2 | 50 | 1 | 1 | 32 | IW(1) | 142,987.636364 |
| capability5 | 0u0u | 0.000037 | 4 | 6 | 1 | 180 | 1 | 2 | 30 | IW(1) | 135,300.129032 |
| capability6 | 0l0u0u0u | 0.000078 | 8 | 8 | 8 | 245 | 1 | 4 | 30 | IW(1) | 115,439.559633 |
| capability7 | 0r0u0u0u | 0.000060 | 6 | 6 | 5 | 180 | 1 | 4 | 30 | IW(1) | 116,972.621514 |

# 2. Biểu đồ

## 2.1. Biểu đồ Line/Scatter tương ứng 3 thuật toán

## 2.2. Biểu đồ sử dụng bộ nhớ (không gian)

# 3. Nhận xét chính (tóm tắt)

• Xu hướng tổng quát:

– Algorithm 1 chạy nhanh với bài nhỏ, nhưng số node sinh rất lớn khi puzzle phức tạp hơn (hơn 57.000 node cho capability9).

– Algorithm 2 (Duplicate Detection) giảm mạnh số node sinh/duyệt so với Algorithm 1 ở các bài có ≥2 pieces; chi phí bộ nhớ phụ tăng nhưng vẫn trong kiểm soát ở tập capability.

– Algorithm 3 (Iterated Width) đạt nghiệm ở bề rộng IW phù hợp với số pieces (thường k ≈ pieces+1 trên bài khó), giúp khống chế tăng trưởng không gian tìm kiếm; thời gian trung vị thường thấp hơn Algorithm 1 và cạnh tranh với Algorithm 2 trên bài nhiều vật thể. Algorithm 3 dùng khái niệm novelty, nên giảm đáng kể số node sinh ra — thể hiện hiệu quả nhất ở các test lớn.

– Các capability 1–7: cả ba thuật toán đều giải rất nhanh; khác biệt rõ rệt chỉ xuất hiện từ capability8 trở đi và các bài impassable\*.

• Ngoại lệ: phát hiện Execution time âm trên 2 mẫu (có khả năng lỗi đo đạc), nên loại khi vẽ biểu đồ thời gian.

# 4. Trả lời các câu hỏi trong yêu cầu báo cáo

• Tăng trưởng thời gian (time complexity):

– #generated/#expanded tăng nhanh theo số pieces và độ dài nghiệm, phù hợp dự báo của BFS/UCS (Algorithm 1).

– Algorithm 2 cắt giảm mạnh nhờ loại trùng trạng thái (radix tree).

– Algorithm 3 tận dụng novelty/IW để thu hẹp không gian cần duyệt, đặc biệt trên bài nhiều vật thể.

• Độ phức tạp không gian (space complexity):

– Algorithm 1: không gian ~ kích thước frontier; tăng nhanh theo branching factor và depth.

– Algorithm 2: thêm bộ nhớ phụ cho cấu trúc visited; đổi lại giảm #generated/#expanded thực tế.

– Algorithm 3: lưu tập tổ hợp atom theo cấp novelty; tổng thể kiểm soát được do nhiều trạng thái bị loại.

• Kết luận:

– Bài đơn giản (≤2 pieces): ba thuật toán tương đương; dùng Algorithm 1 là đủ.

– Bài trung bình/khó (≥3 pieces): ưu tiên Algorithm 2 hoặc 3 tùy ràng buộc bộ nhớ và mục tiêu thời gian.

* Algorithm 1: nhanh nhưng tốn node → kém hiệu quả với puzzle lớn.

Algorithm 2: cân bằng giữa tốc độ và bộ nhớ.

Algorithm 3: tối ưu nhất khi puzzle phức tạp, đặc biệt khi số mảnh > 2.