ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BT2 - A* for Sokoban

Trí tuệ nhân tạo

Sinh viên: Đỗ Phương Duy - 2352362

Ngày ... tháng ... năm ...

Mục lục

1 Thiết kế hàm Heuristic:

1.1 Code:

```python
def heuristic_combined(posPlayer, posBox):
    """Combined heuristic considering both box-goal and player-box distances"""
    # Tính khoảng cách hộp đến đích
    box_to_goal = 0
    completes = set(posGoals) & set(posBox)
    sortposBox = list(set(posBox).difference(completes))
    sortposGoals = list(set(posGoals).difference(completes))

    for i in range(len(sortposBox)):
        box_to_goal += (abs(sortposBox[i][0] - sortposGoals[i][0])) + (abs(sortposBox[i][1] - sortposGoals[i][1]))

    # Tính khoảng cách người chơi đến hộp gần nhất
    if sortposBox:
        player_to_box = float('inf')
        for box in sortposBox:
            dist = abs(posPlayer[0] - box[0]) + abs(posPlayer[1] - box[1])
            player_to_box = min(player_to_box, dist)
    else:
        player_to_box = 0

    # Kết hợp hai khoảng cách với trọng số
    return box_to_goal + 0.5 * player_to_box
```

1.2 Phân tích và nhận xét:

Đây là heuristic cải tiến hơn so với hàm heuristic được cài sẵn với những điểm khác biệt chính:

1.2.1 Xét hai yếu tố:

- **box_to_goal**: Khoảng cách từ hộp đến đích (giống heuristic gốc)
- **player_to_box**: Khoảng cách từ người chơi đến hộp gần nhất (yếu tố mới)

1.2.2 Tính khoảng cách người chơi - hộp:

- Tìm hộp gần người chơi nhất trong số các hộp chưa đến đích
- Sử dụng khoảng cách Manhattan
- Nếu tất cả hộp đã ở đích thì khoảng cách này = 0

1.2.3 Trọng số 0.5

- Khoảng cách người chơi - hộp được nhân với 0.5

- Điều này làm cho yếu tố này ít quan trọng hơn khoảng cách hộp - đích
- Lý do: Di chuyển hộp tốn nhiều chi phí hơn di chuyển người chơi

1.2.4 Ưu điểm:

- Ưu điểm của heuristic này:
- Thực tế hơn vì xét cả vị trí người chơi
- Ưu tiên di chuyển các hộp gần đích trước
- Khuyến khích người chơi di chuyển đến gần hộp cần đẩy
  - So với heuristic gốc:
  - Heuristic gốc chỉ trả về 1 (khoảng cách hộp-đích)
  - Heuristic kết hợp trả về 1.5 (tính thêm khoảng cách người-hộp)
  - Giá trị cao hơn phản ánh đúng hơn chi phí thực tế để giải quyết trạng thái này

2 Bảng thống kê

	UCS			Heuristic cài sẵn			Heuristic em thiết kế		
Level	**Thời gian**	**Số nút**	**Số bước đi**	**Thời gian**	**Số nút**	**Số bước đi**	**Thời gian**	**Số nút**	**Số bước đi**
1	0.08	2115	12	0.02	361	13	0.04	285	12
2	0.01	183	9	0.01	112	9	0.01	91	9
3	0.12	1465	15	0.02	155	15	0.02	111	15
4	0.01	174	7	0.01	98	7	0.00	62	7
5	108.47	1355329	20	0.19	1864	22	0.21	1264	21
6	0.02	612	19	0.03	514	19	0.09	465	19
7	0.80	17283	21	0.20	2071	21	0.15	1714	21
8	0.32	5357	97	0.74	5303	97	0.75	5010	97
9	0.01	181	8	0.01	106	8	0.01	95	8
10	0.03	526	33	0.03	477	33	0.05	469	33
11	0.03	668	34	0.03	638	34	0.06	633	34
12	0.14	3220	23	0.13	1521	23	0.10	1289	23
13	0.26	6202	31	0.25	4593	31	0.22	3937	31
14	4.46	72422	23	2.84	22920	23	1,94	18743	23
15	0.42	6328	105	0.79	5515	105	0.74	5351	105
16	24.24	150971	33	1.13	3274	42	0.50	2738	33
17	Không có đáp án			Không có đáp án			Không có đáp án		
18	Crash máy			Crash máy			Crash Máy		

2.1 So sánh với UCS:

A* hiệu quả hơn UCS nhờ heuristic vì giúp ưu tiên các trạng thái "hứa hẹn" dẫn đến đích và giảm số lượng nút cần khám phá
Tuy nhiên A* phụ thuộc vào chất lượng hàm heuristic và có thể không hiệu quả nếu heuristic không tốt.
Trong thực tế, A* hiệu quả hơn đáng kể về mặt thời gian và không gian tính toán so với UCS trong bài toán Sokoban (mặc dù ở một vài case kết quả trả về chưa phải là lời giải tối ưu như lv5 ở heuristic_combined em thiết kế; lv1, 5, 16 ở hàm heuristic cài sẵn) vì:

- Không gian trạng thái lớn
- Có thể thiết kế heuristic tốt dựa trên đặc điểm của bài toán