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

BT1 - DFS/BFS/UCS for Sokoban

CS106 - Trí Tuệ Nhân Tạo

Đỗ Phương Duy - 23520362

Ngày ... tháng ... năm ...

Mục lục

1 Mô tả Sokoban đã được mô hình hóa ra sao? Trạng thái khởi đầu, trạng thái kết thúc, không gian trạng thái, các hành động hợp lệ, hàm tiến triển (successor function) là gì?

1.1 Trạng thái khởi đầu (Initial State):

```
class Game:
    def __init__(self, window):
        self.window = window
        self.load_textures()
        self.player = None
        self.index_level = 1
        self.load_level()
        self.play = True
        self.scores = Scores(self)
        self.player_interface = PlayerInterface(self.player, self.level)
```

Bao gồm ma trận trạng thái với các giá trị:

- **WALL**: Tường (#)
- **BOX**: Hộp (B)
- **TARGET**: Đích (.)
- **TARGET_FILLED**: Hộp đã đến đích (X)
- **PLAYER**: Vị trí người chơi (&)
- **Ô trống**: (0)

1.2 Trạng thái kết thúc (Goal State):

```
def has_win(self):
    nb_missing_target = 0
    for y in range(len(self.level.structure)):
        for x in range(len(self.level.structure[y])):
            if self.level.structure[y][x] == SOKOBAN.TARGET:
                nb_missing_target += 1
    return nb_missing_target == 0
```

Đạt được khi:

- Tất cả các đích **TARGET** đều được lấp đầy bởi **hộp**
- Không còn đích trống

1.3 Không gian trạng thái (State Space):

Mỗi trạng thái được biểu diễn bởi:

- Vị trí người chơi (`self.level.position_player`)
- Vị trí các hộp trên bản đồ
- Cấu trúc bản đồ (`self.level.structure`)

1.4 Các hành động hợp lệ (Legal Actions):

```
if event.key in [K_UP, K_DOWN, K_LEFT, K_RIGHT, K_z, K_s, K_q, K_d]:
    # Di chuyển người chơi
    self.player.move(event.key, self.level, self.player_interface)
```

Bốn hướng di chuyển cơ bản:

- UP/Z: Di chuyển lên
- DOWN/S: Di chuyển xuống
- LEFT/Q: Di chuyển trái
- RIGHT/D: Di chuyển phải

Một hành động được coi là hợp lệ khi:

- Không đâm vào tường
- Nếu đẩy hộp, phải có không gian trống phía sau hộp

1.5 Hàm tiến triển (Successor Function):

```
def auto_move(self):
        # Các thuật toán tìm kiếm
        strategy = get_move(self.level.structure[:-1],
        ↪  self.level.position_player, 'dfs')
        strategy = get_move(self.level.structure[:-1],
        ↪  self.level.position_player, 'bfs')
        strategy = get_move(self.level.structure[:-1],
        ↪  self.level.position_player, 'ucs')
        # with open("assets/sokobanSolver/Solverlevel_" + str(self.index_level)
        ↪  + ".txt", 'w+') as solver_file:
        #     for listitem in strategy:
        #         solver_file.write('%s, ' % listitem)
```

Hàm này:

- Nhận vào trạng thái hiện tại
- Tính toán các bước di chuyển hợp lệ
- Trả về chuỗi các hành động để đạt mục tiêu

1.6 Các đặc điểm bổ sung:

Hàm lưu trữ trạng thái:

```
class Level:
    def __init__(self, level_to_load):
        self.last_structure_state = None
        self.load(level_to_load)
```

Hỗ trợ nhiều thuật toán tìm kiếm:

- DFS (Depth-First Search)
- BFS (Breadth-First Search)
- UCS (Uniform-Cost Search)

Giao diện trực quan:

```
def update_screen(self):
    # Cập nhật hiển thị trạng thái trò chơi
    self.level.render(self.board, self.textures)
    self.player.render(self.board, self.textures)
```

Mô hình này cho phép:

- Biểu diễn đầy đủ trạng thái trò chơi
- Kiểm tra tính hợp lệ của các bước di chuyển
- Theo dõi tiến trình giải
- Hiển thị trực quan quá trình giải
- Áp dụng nhiều chiến lược tìm kiếm khác nhau

2 Thuật toán DFS, BFS, UCS:

2.1 Bảng thống kê:

Level	DFS	BFS	UCS
1	0.05s | 79	0.12s | 12	0.08s | 12
2	0s | 24	0.01s | 9	0.01s | 9
3	0.2s | 403	0.24s | 15	0.12s |15
4	0s | 27	0.01s | 7	0.01s | 7
5	Crash Máy	322.11s | 20	108.47s | 20
6	0.03s | 55	0.02s | 19	0.02s | 19
7	1.01s | 707	1.23s | 21	0.80s | 21
8	0.33s | 323	0.29s | 97	0.32s | 97
9	0.26s | 74	0.06s | 8	0.01s | 8
10	0.01s | 37	0.05s | 33	0.03s | 33
11	0.01s | 36	0.03s | 34	0.03s | 34
12	0.11s | 109	0.14s | 23	0.14s | 23
13	0.17s | 185	0.2s | 31	0.26s | 31
14	4.51s | 865	4.16s | 23	4.46s | 23
15	0.52s | 291	0.42s | 105	0.42s | 105
16	Crash máy	32.55s | 34	24.24s | 34
17	37.56s | Không có đáp án	37.63s | Không có đáp án	39.93s | Không có đáp án
18	Crash Máy	Crash Máy	Crash Máy

Bảng 1: Bảng kết quả tìm kiếm theo thuật toán(Thời gian chạy | số bước đi)

3 Nhận Xét:

DFS: Thuật toán này thường có số bước đi lớn và đôi khi dẫn đến "Crash Máy" ở các mức độ phức tạp cao (Level 5, 16, 18). Điều này xảy ra do DFS có thể đi sâu vào cây tìm kiếm mà không tìm thấy lời giải, dẫn đến việc tiêu tốn bộ nhớ lớn.
BFS: Ở bài toán Sokoban, mỗi bước đi đều có chi phí là 1 nên BFS đảm bảo tìm thấy lời giải tối ưu nhưng có thể tốn nhiều thời gian và bộ nhớ khi số lượng nút mở rộng quá lớn. Ở Level 5, thuật toán này mất 322.11s, cho thấy vấn đề về độ phức tạp.
UCS: UCS hoạt động hiệu quả hơn DFS nhưng cũng gặp vấn đề tương tự BFS.

Trong tất cả các bản đồ thì em thấy level 18 khó nhất vì cả 3 thuật toán đều bị crash khi giải level này. Level 18 có 10 Target và 10 hộp, mê cung cũng khá trick, con người cũng mất khá nhiều thời gian để giải ra được đáp án.