TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Báo cáo Lab 1

Cơ sở trí tuệ nhân tạo

Nguyễn Hiển Đạt

Lớp: 21CLC06

MSSV: 21127591

Hồ Chí Minh, tháng 7 năm 2023

Mục lục

1) Mô tả các hàm

1.1) Class UniformCostSearch

- def __init__(self, initial_state, n): Hàm khởi tạo class UniformCostSearch với 2 giá trị đầu vào là state ban đầu initial_state và số lượng queens n.
- def does_conflict(self, current_state): Hàm kiểm tra xem state hiện tại có cặp queens nào tấn công nhau hay không. Nếu có return false.
- def generate_successors(self, parent_state): Hàm tạo state mới dựa vào state trước đó parent_state.
- def solve(self): Hàm thực hiện giải thuật UCS. Đầu tiên khởi tạo frontier và thêm initial_state vào. Thực hiện UCS tới khi nào tìm ra giải pháp. Trong vòng lặp, tạo successors của *current_state (1)* và thêm vào frontier. Sau đó kiểm tra *current_state* **hiện tại** (được pop từ *frontier* ra, không phải *current_state (1)*) có thỏa mãn yêu cầu không. Nếu có thì return current_state hiện tại.

1.2) Class AstarSearch

- def __init__(self, initial_state, n): Hàm khởi tạo class AstarSearch với hai giá trị đầu vào là state ban đầu initial_state và số lượng queens n.
- def count_conflicts(self, current_state): Hàm tính số lượng cặp queens tấn công nhau ở state hiện tại current_state. Đây cũng chính là giá trị heuristic.
- def does_conflict(self, current_state): Hàm kiểm tra xem state hiện tại có cặp queens nào tấn công nhau hay không, nếu có return false.
- def generate_successors(self, parent_state): Hàm tạo successor mới dựa vào state trước đó parent_state.
- def solve(self): Hàm thực hiện giải thuật A*. Cách thức hoạt động khá giống với UCS nhưng phải tính thêm heuristic bằng hàm count_conflicts.

1.3) Class GeneticAlgorithm

- def __init__(self, initial_state, board_size): Hàm khởi tạo class GeneticAlgorithm với hai giá trị đầu vào là initial_state và board_size.

- def count_conflicts(self, current_state): Hàm tính số lượng cặp queens tấn công nhau ở state hiện tại current_state.
- def initialize_population(self, population_size): Hàm khởi tạo một population. Đầu tiên tạo một số ngẫu nhiên giữa 2 và *population_size*. Sau đó, nó sinh ra *num* giải pháp trong đó mỗi giải pháp là một danh sách các số nguyên từ 0 đến *self.board_size - 1*. Hàm sau đó tính toán số cặp queens tấn công nhau trong mỗi solution và thêm nó vào heap.
- def goal_test(self, population): Hàm trả về solution đầu tiên thỏa mãn không có cặp queens nào tấn công nhau.
- def randomize(self, population): Hàm tạo ngẫu nhiên một population. Population này gồm một số lượng ngẫu nhiên (lớn hơn 2 và nhỏ hơn độ dài population) các solution từ population đầu vào.
- def crossover(self, parent1, parent2): Hàm trả về *child1*,*child2* bằng cách tráo đổi các thành phần của cặp *parents1* và *parents2*.
- def mutate(self, state): Hàm thay đổi giá trị tại *mutation point* (được khởi tạo ngẫu nhiên) bằng một giá trị ngẫu nhiên khác
- def solve(self): Hàm thực hiện giải thuật Genetic Algorithm. Đầu tiên khởi tạo một *population* sau đó push vào heap. Thực hiện GA tới khi nào tìm ra giải pháp. Trong vòng lặp, khởi tạo ngẫu nhiên một *random_population* bằng cách gọi hàm *randomize*. Sau đó thực hiện *crossover* và *mutate* đối với population trước khi push vào heap. Kiểm tra xem có solution nào thỏa mãn goal state không. Nếu có thì return solution đó, còn không thì tiếp tục cho đến khi nào tìm ra solution thỏa mãn.

2) Kết luận

2.1) Checklist

- ✓ UCS
- ✓ A*
- ✓ GA

2.2) Bảng số liệu

	Running time(ms)			Memory(MB)		
Algorithms	**N=8**	**N=100**	**N=500**	**N=8**	**N=100**	**N=500**
UCS	Intractable	Intractable	Intractable	Intractable	Intractable	Intractable
A*	12.0075	Intractable	Intractable	0.74	Intractable	Intractable
GA	38,9980	Intractable	Intractable	0.28	Intractable	Intractable

2.3) Nhận xét

- UCS tốn rất nhiều bộ nhớ. Với n=8 máy tính em (16GB RAM) đã bị tràn RAM sau khi chạy được vài phút. Em chỉ chạy được trường hợp n=7 với thời gian trung bình sau 3 lần đo là 22629,4212 ms.

- A* hoạt động tốt với n không quá lớn (n=80 chương trình của em vẫn ra kết quả) tuy nhiên khi lên n=100, cả 3 lần chạy đều quá 12 tiếng. Vì vậy em cho rằng với số lượng queens này, A* không thể giải quyết bài toán trong khoảng thời gian chấp nhận được.

- GA hoạt động tốt với n không quá lớn, tuy nhiên khi lên n=100, em chạy 3 lần thì chỉ có một lần ra kết quả trong vòng 1 giờ đồng hồ, còn 2 lần kia đều chạy quá 12 tiếng. Lần chạy ra có lẽ là do khởi tạo ngẫu nhiên initial state gần với goal state. Vì vậy em cho rằng với số lượng queens này, GA không thể giải quyết bài toán trong khoảng thời gian chấp nhận được.