Báo cáo Mini Project OOP Lý Thuyết
Thiết kế Genetic Algorithm để giải TSP

Nhóm 2

January 2024

1 Giới thiệu

Bài toán TSP, viết tắt của "Traveling Saleman Problem", là một trong những bài toán kinh điển trong lĩnh vực tối ưu hóa và có ứng dụng rất đa dạng trong thực tế. Bài toán đã thu hút sự chú ý và quan tâm nghiên cứu của nhiều nhà toán học và nhà khoa học máy tính để trả lời câu hỏi "Làm sao để một người bán hàng có thể ghé thăm tất cả các địa điểm khác nhau một lần và trở về điểm xuất phát sao cho tổng quãng đường là ngắn nhất?". Trong khí đó thuật toán tiến hóa (genetic algorithm) được các nhà khoa học phát triển dựa trên cơ sở của di truyền học để giải quyết các bài toán về tối ưu. Trong project này, nhóm chúng tôi sẽ sử dụng thuật toán tiến hóa để giải bài toán TSP.

1.1 Mô tả bài toán

Bài toán TSP được mô tả như sau: Có một số N các thành phố và biết được khoảng cách giữa các thành phố. Người đưa thư phải đi qua tất cả các thành phố, hãy tìm lộ trình giúp cho người đưa thư đi qua tất cả các thành phố với lộ trình là ngắn nhất. Để kiểm tra mọi khả năng có thể chọn một đường đi ngắn nhất qua N thành phố ta có N! trường hợp, điều này làm cho bài toán được cho là một trong những bài toán NP-khó.

1.2 Phương pháp giải quyết vấn đề

Để giải quyết được vấn đề đặt ra trong bài toán TSP, nhóm chúng em sử dụng các kỹ thuật của thuật toán tiến hóa (GA) như là lựa chọn tự nhiên, đột biến, lai ghép và di truyền. Cùng với đó, nhóm chúng em cũng sử dụng các kỹ thuật của lập trình hướng đối tượng trên nền tảng là ngôn ngữ lập trình Java để hoàn thiện project. Và cuối cùng là nhom chúng em sử dụng GUI để tạo ra giao diện dùng để giao tiếp với người dùng.

2 Use case diagram

Với yêu cầu đề bài, nhóm chúng tôi đã tạo ra biểu đồ usecase với 3 chức năng chính:

- **Browse help** : chức năng này được sử dụng khi người dùng muốn tìm kiếm hướng dẫn sử dụng để tìm kiếm được lời giải tối ưu nhất theo như mong muốn của người dùng
- **Quit the application** : người dùng sẽ sử dụng chức năng này để thoát khỏi chương trình đã sử dụng khi giải quyết được bài toán mình muốn tìm kiếm lời giải tối ưu
- **Start Solving TSP** : người dùng sẽ sử dụng khi mà muốn chạy thuật toán. Sau khi người dùng nhập vào các thông số được yêu cầu như trong hướng dẫn, rồi nhấn nút Start để chạy thuật toán. Khi thuật toán đang chạy, người dùng có thể nhấn nút Stop để . Sau khi thuật toán hoàn tất, người dùng có thể nhấn nút Print route để in ra đường đi ngắn nhất được tìm thấy bởi chương trình

3 General class diagram

Trong sơ đồ lớp xây dựng thuật toán, nhóm chúng em đã thiết kế thành 4 package chính đó là: components, geneticalgorithm, gui và test

3.1 components package

Trong components package sẽ cung cấp đầy đủ các thực thể cơ bản nhất phục vụ cho việc tìm lời giải của bài toán : node, individual, route, population, IndividualComparator

- Một node là một thành phố trong bài toán TSP
- Một individual là một dãy các số đại diện cho thứ tự thăm các thành phố đã tìm được, và được xem xét như là một lời giải của thuật toán
- Một population là một tập hợp các individuals được xem xét để trở thành lời giải của thuật toán
- Một route là một tổ hợp của các nodes có thứ tự giống như một individual nào đó

3.2 geneticalgorithm package

Trong genetic algorithm sẽ chứa đựng các lớp thuật toán để giải quyết yêu cầu của đề bài. Đó chính là CrossOver, Mutation, và GeneticAlgorithm

- CrossOver : đây là một lớp abstract cung cấp các thuộc tính cơ bản nhất là bố và me phục vụ cho việc sinh ra các lời giải mới từ 2 lời giải bố và mẹ cho trước. Từ đó ta có lớp dẫn xuất OrderedCrossOver từ lớp cơ sở CrossOver để tạo ra con mới lấy một phần gen của cha và mẹ theo tỷ lệ ngẫu nhiên
- Mutation : đây cũng là một lớp abstract dùng để làm khuôn mẫu cho các dạng đột biến từ một lời giải gốc có sẵn để tạo ra một lời giải mới cho bài toán. Từ lớp cơ sở này chúng ta có 2 lớp dẫn xuất đó là SwapMutation và InverseMutation. SwapMutation class sinh ra lời giải mới của bài toán bằng cách đổi chỗ vị trí của 2 thành phố bất kỳ có trong lời giải gốc. Trong khi đó InverseMutation sẽ đảo ngược lại một phần của lời giải gốc để sinh ra lời giải mới

3.3 gui package

Trong gui package, chúng em tạo ra 2 package con đó là app package và controller package để chứa các class của gui.

3.4 test package

4 Detailed class diagram

4.1 components package

4.1.1 Node class

Một node đại diện cho một thành phố trong bài toán TSP

- Thuộc tính
  - x và y là 2 tọa độ của node trên mặt phẳng 2D và có kiểu dữ liệu là int

- Phương thức
  - Node(int x, int y) là constructor dùng để khởi tạo đối tượng thuộc lớp
  - getX(), getY() là các phương thức dùng để lấy các tọa độ của node theo trục Ox và trục Oy
  - getDistance(node Node) : phương thức này dùng để tính toán khoảng cách giữa một node và một node khác bất kỳ
  - toString() là phương thức ghi đè của phương thức cùng tên trong lớp Object dùng để in ra tọa độ x, y tương ứng của node đó

4.1.2 Individual class

Một individual là một tập hợp có thứ tự các thành phố đến thăm

- Thuộc tính
  - chromosome : có kiểu dữ liệu là ArrayList<int> dùng để lưu danh sách các node được đến thăm có thứ tự
  - fitness : có kiểu dữ liệu là double là chỉ số dùng để xác định individual này có tốt hay không để đưa vào các thế hệ sau

- Phương thức
  - Individual(ArrayList<int> chromosome), Individual(int cityNum) và Individual(int cityNum, int placeholder) là ba constructor dùng để khởi tạo đối tượng có kiểu là Individual theo 3 cách khác nhau

- getChromosome(), getFitness() là các gettet trả về các thuộc tính chromosome và fitness của đối tượng cần lấy thông tin để tính toán
- toString() là phương thức ghi đè của phương thức cùng tên trong lớp Object dùng để in ra danh sách thứ tự các node được thăm và quay trở về vị trí xuất phát

4.1.3 Population class

- Thuộc tính
  - Vì một Population la một tập hơn các Individual nên sẽ có một thuộc tính là population có kiểu dữ liệu là ArrayList<Individual> dùng để lưu trữ một danh sách các Individual

- Phương thức
  - Population class có 2 contructor là Population(int populationSize, int cityNum) và Population(int populationSize)
  - getPopulation(): trả về

4.1.4 Route class

- Thuộc tính
  - route : dùng để lưu danh sách các node theo thứ tự có sẵn
  - individual : có kiểu dữ liệu là Individual

- Phương thức
  - Route(individual Individual, Node[] nodes) : đây là construtor dùng để khởi tạo đối tượng thuộc lớp
  - getRoute() :
  - getIndividual() :
  - totalDistance() : phương thức dùng để tính tổng quãng đường đi từ điểm đầu đến điểm cuối và quay trở lại điểm xuất phát

4.2 geneticalgorithm class

4.2.1 CrossOver

Đây là abstract class dùng để sinh ra các lòi giải ngẫu nhiên cho bài toán bao gồm các thuộc tính và phương thức

- Thuộc tính
  - CrossOver class có 2 thuộc ính là parent1 và parent2 có kiểu dữ liệu là Individual. Đây là hai bố mẹ được chọn ra để sinh ra lời giải con ngẫu nhiên

- Phương thức
  - CrossOver(Individual parent1, Individual parent2) : đây là constructor method dùng để khởi tạo - getChild() : đây là abstract method chưa được cài đặt chi tiết là sẽ được các lớp dẫn xuất ghi đè

4.2.2 OrderedCrossOver class

Đây là lớp dẫn xuất của lớp CrossOver

- Thuộc tính
  - Lớp dẫn xuất này kế thùa các thuộc tính của lớp cơ sở là parent1 và parent2 có kiểu là individual

- Phương thức
  - OrderedCrossOver(Individual parent1, Individual parent2) : đây là constructor method dùng để khởi tạo đối tượng thuộc lớp
  - getChild() : đây là phương thức ghi đè phương thức của lớp cơ sở, cài đặt chi tiết các hành động cần thực hiện để sinh ra một lời giải con ngẫu nhiên mới

4.2.3 Mutation class

Đây là một abstract class, quy định khuôn mẫu cho các class tạo đột biến để sinh ra lời giải mới từ một lời giải gốc ban đầu.

- Thuộc tính
  - Lớp chỉ có một thuộc tính là original có kiểu dữ liệu là individual chứa lời giải gốc dùng để làm cơ sơ cho lời giải đột biến

- Phương thức
  - Mutation(Individual original) : đây là constructor method dùng để khởi tạo đối tượng thuộc lớp
  - mutate() : đây là abstract method chưa được cài đặt chi tiết và sẽ được các lớp dẫn xuất ghi đè

4.2.4 SwapMutation

Đây là một lớp dẫn xuất từ lớp cơ sở là lớp Mutation có các thuộc tính và phương thức

- Thuộc tính
  - Lớp này sẽ kế thừa các thuộc tính của lớp dẫn xuất đó là thuộc tính original có kiểu dữ liệu là Individual

- Phương thức
  - SwapMutation(Individual original) : đây là constructor method dùng để khởi tạo đối tượng thuộc lớp - mutate() : đây là phương thức được ghi đè từ phương thức cùng tên của lớp cơ sở dùng để tạo đột biến sinh lời giải con bằng cách đổi chỗ hai phần tử bất kỳ của danh sách có thứ tự của các nodes.

4.2.5 InverseMutation

Đây là một lớp dẫn xuất khác được kế thừa từ lớp cơ sở Mutation

- Thuộc tính
  - InverseMutation class sẽ kế thừa các thuộc tính của lớp cơ sở là thuộc tính original có kiểu dữ liệu là Individual

- Phương thức
  - InverseMutation(Individual original) : đây là constructor method dùng để khởi tạo đối tượng thuộc lớp - mutate() : đây là phương thức được ghi đè từ phương thức cùng tên của lớp cơ sở dùng để tạo đột biến sinh lời giải con bằng cách đảo ngược một đoạn của danh sách các node một cách ngẫu nhiên

4.2.6 GeneticAlgorithm

Đây là class chứa lời giải chi tiết của thuật toán tiến hóa dùng để giải bài toán TSP

- Thuộc tính
  - mutationNum:
  - crossOverNum:
  - populationSize :

- cityNum : lưu số lượng các thành phố mà người dùng nhập vào từ bàn phím

- Phương thức
  - GeneticAlgorithm():
  - generateNodes() : dùng để sinh ngẫu nhiên các node ở trên mặt phẳng 2D
  - initializePopulation(): - updateFitness(Population population, Node[] nodes):
  - evovle(Population population):

4.3 gui package

4.4 test package

5 Phân công công việc