**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ**

**TRAVELING SALESMAN PROBLEM**

**WITH GENETIC ALGORITHMS**

*Người hướng dẫn*:  **TS. NGUYỄN CHÍ THIỆN**

*Người thực hiện*:  **TRẦN THỊ VẸN – 52100674**

**NGUYỄN VŨ TƯỜNG – 52100944**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ**

**TRAVELING SALESMAN PROBLEM**

**WITH GENETIC ALGORITHMS**

*Người hướng dẫn*:  **TS. NGUYỄN CHÍ THIỆN**

*Người thực hiện*:  **TRẦN THỊ VẸN – 52100674**

**NGUYỄN VŨ TƯỜNG – 52100944**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

# LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn khoa Công nghệ thông tin đã tạo điều kiện cho chúng em được tiếp cận và hoàn thành bài báo cáo. Chúng em xin chân thành cảm ơn thầy Nguyễn Chí Thiện đã hướng dẫn hoàn thành bài báo cáo.

Trong quá trình làm bài báo cáo, do kiến thức cũng như kinh nghiệm còn nhiều hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp của thầy để chúng em có thể học hỏi được nhiều kĩ năng, kinh nghiệm và sẽ ngày càng hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của thầy Nguyễn Chí Thiện. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 01 tháng 01 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Thị Vẹn*

*Nguyễn Vũ Tường*

**TÓM TẮT**

Báo cáo này xác định vấn đề cho thuật toán giải thuật di truyền cho vấn đề người giao hàng (đầu vào, đầu ra theo cấu trúc dữ liệu Java). Biểu diễn Java (lập trình chung) của các thuật toán để giải quyết vấn đề mà không cần nhập thư viện bên ngoài. Phân tích độ phức tạp tiệm cận của các giải pháp(tuân thủ nghiêm ngặt các kế hoạch phân tích.)

Viết chương trình tạo tập dữ liệu đầu vào. Thử nghiệm các chương trình với các tập dữ liệu đầu vào được tạo ra. Vẽ hàm phức tạp lý thuyết và hoạt động thực tế thời gian là hàm của kích thước đầu vào. Đồ họa phải có hoa văn tương tự. Đưa ra kết luận dựa trên kết quả thực nghiệm sau khi thử nghiệm thuật toán.

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc155483638)

[**TÓM TẮT** iii](#_Toc155483639)

[MỤC LỤC 1](#_Toc155483640)

[DANH MỤC HÌNH ẢNH, BẢNG BIỂU 2](#_Toc155483641)

[DANH MỤC BẢNG 2](#_Toc155483642)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 4](#_Toc155483643)

[CHƯƠNG 1 - GIỚI THIỆU 5](#_Toc155483644)

[CHƯƠNG 2 - RELATED WORK 5](#_Toc155483645)

[CHƯƠNG 3 - SƠ LƯỢC VẤN ĐỀ CẦN GIẢI QUYẾT 6](#_Toc155483646)

[CHƯƠNG 4 - PHƯƠNG PHÁP 7](#_Toc155483647)

[CHƯƠNG 5 - DỮ LIỆU ĐẦU VÀO VÀ ĐẦU RA 24](#_Toc155483648)

[CHƯƠNG 6 - TẠO RA DỮ LIỆU ĐẦU VÀO 25](#_Toc155483649)

[CHƯƠNG 7 - KẾT QUẢ THỬ NGHIỆM VÀ KẾT LUẬN 28](#_Toc155483650)

[CHƯƠNG 8 - SO SÁNH VỚI CÁC THUẬT TOÁN KHÁC 30](#_Toc155483651)

[TÀI LIỆU THAM KHẢO 32](#_Toc155483652)

# DANH MỤC HÌNH ẢNH, BẢNG BIỂU

**DANH MỤC HÌNH ẢNH**

[Hình 1: Biểu đồ thời gian chạy của 75 thành phố 28](#_Toc155480264)

[Hình 2: Biểu đồ thời gian chạy của 200 thành phố 29](#_Toc155480265)

# DANH MỤC BẢNG

[Bảng 1: Danh sách ma trận vuông 5 thành phố 8](#_Toc155483594)

[Bảng 2: Random map chromosome 8](#_Toc155483595)

[Bảng 3: Random Guide chromosome 9](#_Toc155483596)

[Bảng 4: Selection (roulette-wheel) 9](#_Toc155483597)

[Bảng 5: Uniform Crossover 9](#_Toc155483598)

[Bảng 6: Danh sách ma trận vuông 6 thành phố 10](#_Toc155483599)

[Bảng 7: Khởi tạo quần thể 10](#_Toc155483600)

[Bảng 8: Tính xác suất được chọn 11](#_Toc155483601)

[Bảng 9: Tính xác suất tích lũy 11](#_Toc155483602)

[Bảng 10: Chọn ra 2 parent 12](#_Toc155483603)

[Bảng 11: Chọn ra 2 parent rồi thực hiện lai chéo 12](#_Toc155483604)

[Bảng 12: Chọn ra 2 parent rồi thực hiện lai chéo 12](#_Toc155483605)

[Bảng 13: Chọn ra 2 parent rồi thực hiện lai chéo cho P2 13](#_Toc155483606)

[Bảng 14: Bảng population mới 13](#_Toc155483607)

[Bảng 15: Bảng population mới cùng khoảng cách mới 13](#_Toc155483608)

[Bảng 16: Bảng parent mới chọn 14](#_Toc155483609)

[Bảng 17: Bảng parent mới chọn và thực hiện lai chéo 14](#_Toc155483610)

[Bảng 18: Bảng parent mới chọn và thực hiện lai chéo trên P2 14](#_Toc155483611)

[Bảng 19: Bảng population gồm 4 con 15](#_Toc155483612)

[Bảng 20: Bảng thế hệ mới chọn ra cái tốt nhất 15](#_Toc155483613)

[Bảng 21: Chọn 2 parent dựa trên random 15](#_Toc155483614)

[Bảng 22: Thực hiện lai chéo dựa trên P1 16](#_Toc155483615)

[Bảng 23: Thực hiện lai chéo dựa trên P2 16](#_Toc155483616)

[Bảng 24: 6 child mới xuất hiện trong bảng 16](#_Toc155483617)

[Bảng 25: Thực hiện lai chéo dựa trên P1 17](#_Toc155483618)

[Bảng 26: Thực hiện lai chéo dựa trên P2 17](#_Toc155483619)

[Bảng 27: Thực hiện lai chéo dựa trên P1 18](#_Toc155483620)

[Bảng 28: Thực hiện lai chéo dựa trên P2 18](#_Toc155483621)

[Bảng 29: Thực hiện lai chéo dựa trên P1 18](#_Toc155483622)

[Bảng 30: Thực hiện lai chéo dựa trên P2 19](#_Toc155483623)

[Bảng 31: Thực hiện lai chéo dựa trên P1 19](#_Toc155483624)

[Bảng 32: Thực hiện lai chéo dựa trên P2 19](#_Toc155483625)

[Bảng 33: Thực hiện lai chéo dựa trên P1 19](#_Toc155483626)

[Bảng 34: Thực hiện lai chéo dựa trên P2 19](#_Toc155483627)

[Bảng 35: Bảng gồm 8 child mới 20](#_Toc155483628)

[Bảng 36: Thực hiện lai chéo dựa trên P1 21](#_Toc155483629)

[Bảng 37: Thực hiện lai chéo dựa trên P2 21](#_Toc155483630)

[Bảng 38: Thực hiện lai chéo dựa trên P1 21](#_Toc155483631)

[Bảng 39: Thực hiện lai chéo dựa trên P2 22](#_Toc155483632)

[Bảng 40: 10 child mới được tạo 22](#_Toc155483633)

[Bảng 41: Thế hệ thứ 3 của quần thể này 22](#_Toc155483634)

[Bảng 42: Thế hệ thứ 4 của quần thể này 23](#_Toc155483635)

[Bảng 43: Bảng thời gian chạy thuật toán với 75 thành phố 28](#_Toc155483636)

[Bảng 44: Bảng thời gian chạy thuật toán với 200 thành phố 29](#_Toc155483637)

# DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC CHỮ VIẾT TẮT**

TSP Traveling salesman problem

ACO Ant colony optimization

NST Nhiễm sắc thể

# CHƯƠNG 1 - GIỚI THIỆU

Sự đa dạng của các phương pháp được sử dụng để giải quyết vấn đề người giao hàng chứng tỏ rằng vấn đề này vẫn còn phức tạp và được các nhà nghiên cứu trong lĩnh vực này quan tâm. Đối với các bài toán có không gian tìm kiếm lớn, tương tự như người bán hàng du lịch, các thuật toán tiến hóa như thuật toán di truyền rất mạnh mẽ và có thể được sử dụng để đạt được tối ưu hóa các giải pháp. Tuy nhiên, thách thức trong việc áp dụng giải thuật di truyền vào bài toán nhân viên bán hàng du lịch là việc lựa chọn các nhà khai thác thích hợp có thể sản xuất các chuyến du lịch hợp pháp. Trong tài liệu, các thuật toán sửa chữa bổ sung đã được giới thiệu và được sử dụng và con cái được tạo ra bởi những người vận hành thuật toán di truyền này được sửa đổi để đảm bảo rằng các nhiễm sắc thể được tạo ra đại diện cho các chuyến tham quan hợp pháp. Thay vì bám vào các thuật toán sửa chữa, phương pháp tiếp cận nhiễm sắc thể kép được đề xuất trong bài viết này. Phương pháp đề xuất có thể được sử dụng để tối ưu hóa bài toán tương tự như bài toán người bán hàng du lịch. Phương pháp tiếp cận nhiễm sắc thể kép đã được thử nghiệm với nhiều vấn đề khác nhau của nhân viên bán hàng du lịch và kết quả cho thấy phương pháp đề xuất có tốc độ hội tụ cao hướng tới hành trình ngắn nhất.

# CHƯƠNG 2 - RELATED WORK

Bài toán Người du lịch Bài toán Người du lịch, tìm đường đi ngắn nhất cho người thương nhân (salesman), hay còn gọi là người chào hàng xuất phát từ một thành phố, đi qua lần lượt tất cả các thành phố duy nhất một lần và quay về thành phố ban đầu với chi phí rẻ nhất, được phát biểu vào thế kỷ 17 bởi hai nhà toán học vương quốc Anh là Sir William Rowan Hamilton và Thomas Penyngton Kirkman, và được ghi trong cuốn giáo trình Lý thuyết đồ thị nổi tiếng của Oxford. Nó nhanh chóng trở thành bài toán khó thách thức toàn thế giới bởi độ phức tạp thuật toán tăng theo hàm số mũ (trong chuyên ngành thuật toán người ta còn gọi chúng là những bài toán NP-khó). Người ta bắt đầu thử và công bố các kết quả giải bài toán này trên máy tính từ năm 1954 (49 đỉnh), cho đến năm 2004 bài toán giải được với số đỉnh lên tới 24.978, và dự báo sẽ còn tiếp tục tăng cao nữa.

Các phương pháp giải quyết bài toán TSP bao gồm:

* Brute Force: Tính toán tất cả các hoán vị của các điểm và chọn hoán vị tối ưu. Tuy nhiên, phương pháp này không thực tế cho các bài toán lớn vì số lượng hoán vị là rất lớn.
* Nearest Neighbor: Bắt đầu từ một điểm bất kỳ, tìm điểm gần nhất và di chuyển đến nó, sau đó tìm điểm gần nhất với điểm đang đứng và tiếp tục cho đến khi tất cả các điểm được ghé thăm. Phương pháp này dễ triển khai và thường được sử dụng cho các bài toán nhỏ.
* Christofides Algorithm: Kết hợp giữa Nearest Neighbor và Minimum Spanning Tree để tìm một lời giải gần đúng với tỷ lệ gần 3/2 so với giải tối ưu. Phương pháp này được sử dụng cho các bài toán lớn.
* Simulated Annealing: Một phương pháp metaheuristics, nó tạo ra một lời giải bắt đầu ngẫu nhiên và sau đó cố gắng tìm một lời giải tốt hơn bằng cách di chuyển qua các lời giải khác theo một quy trình tương tự như quá trình làm mát trong hành trình.
* Genetic Algorithm: Một thuật toán tối ưu hóa được lấy cảm hứng từ các quá trình tiến hóa trong tự nhiên. Thuật toán này sử dụng một quần thể các lời giải và tiến hành các phép lai ghép, đột biến để tìm ra lời giải tốt hơn và nhiều thuật toán khác.

# CHƯƠNG 3 - SƠ LƯỢC VẤN ĐỀ CẦN GIẢI QUYẾT

Trong một TSP đối xứng trong đó khoảng cách giữa hai thành phố là như nhau theo mỗi hướng ngược nhau, tổng số chuyến tham quan có thể có (θ) của n thành phố có thể thu được bằng cách loại bỏ các chuyến tham quan tương tự như sau:

Phương pháp được đề xuất:

Terminal

Condition

Selection (Chọn lựa)

Tạo nhiễm sắc thể bản đồ ban đầu

Genetic Operation

(Lai ghép hoặc đột biến)

Tạo quần thể nhiễm sắc thể hướng dẫn ban đầu

Tạo ra thế hệ NST hướng dẫn mới

Áp dụng các NST dẫn đường vào NST bản đồ và đánh giá

# CHƯƠNG 4 - PHƯƠNG PHÁP

* Bước 1: Tìm khoảng cách giữa các thành phố

Sau khi có được n thành phố thì sẽ tạo ra ma trận vuông khoảng cách giữa các thành phố.

* Bước 2: Tạo ra một random Map chromosome
* Bước 3: Tạo ra random Guide chromosome

Các nhiễm sắc thể dẫn đường được sử dụng để sắp xếp lại nhiễm sắc thể bản đồ. Trong các nhiễm sắc thể hướng dẫn được đề xuất, mỗi cặp gen sẽ xác định hai thành phố nào trong nhiễm sắc thể bản đồ sẽ được hoán đổi. Do đó, các nhiễm sắc thể dẫn đường hoạt động theo cách sắp xếp nhiễm sắc thể bản đồ để đạt được hành trình ngắn nhất.

* Bước 4: Selection (roulette-wheel)

Chọn ra con đường ngắn nhất sau khi đã tính toán.

* Bước 5: Uniform Crossover (Lai ghép) và Mutation (Đột biến)

Ví dụ: Có 5 thành phố A, B, C, D, E và khoảng cách giữa các thành phố

Bước 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Thành phố | A | B | C | D | E |
| A | 0 | 167 | 170 | 93 | 145 |
| B | 167 | 0 | 73 | 123 | 206 |
| C | 170 | 73 | 0 | 91 | 152 |
| D | 93 | 123 | 91 | 0 | 92 |
| E | 145 | 206 | 152 | 92 | 0 |

Bảng 1: Danh sách ma trận vuông 5 thành phố

Bước 2: Tạo ra một random map chromosome

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| E | B | C | A | D |
| (0) | (1) | (2) | (3) | (4) |

Bảng 2: Random map chromosome

Bước 3: Tạo ra random Guide chromosome

|  |  |  |
| --- | --- | --- |
| NST dẫn đường | Đường mới | Khoảng cách |
| [1,3,0,2] | [C, A, E, B, D] | 735 |
| [0,1,3,4] | [B, E, C, D, A] | 709 |
| [0,2,0,4] | [D, B, C, E, A] | 586 |
| [2,3,1,4] | [E, D, A, C, B] | 634 |

Bảng 3: Random Guide chromosome

Bước 4: Selection (roulette-wheel)

|  |  |  |
| --- | --- | --- |
| NST dẫn đường | Đường mới | Khoảng cách |
| [1,3,0,2] | [C, A, E, B, D] | 735 |
| [0,1,3,4] | [B, E, C, D, A] | 709 |
| [0,2,0,4] | [D, B, C, E, A] | 586 |
| [2,3,1,4] | [E, D, A, C, B] | 634 |

Bảng 4: Selection (roulette-wheel)

Bước 5: Uniform Crossover

|  |  |  |
| --- | --- | --- |
| NST dẫn đường | Đường mới | Khoảng cách |
| [2,3,1,2] | [E, A, B, C, D] | 568 |

Bảng 5: Uniform Crossover

**Phân tích độ phức tạp của thuật toán**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A – 1 | B – 2 | C – 3 | D – 4 | E – 5 | F – 6 |
| A – 1 | 0 | 1930 | 1484 | 1096 | 991 | 1152 |
| B – 2 | 1930 | 0 | 684 | 1091 | 1211 | 1582 |
| C – 3 | 1484 | 684 | 0 | 459 | 1427 | 884 |
| D – 4 | 1096 | 1091 | 459 | 0 | 1046 | 808 |
| E – 5 | 991 | 1211 | 1427 | 1046 | 0 | 1619 |
| F – 6 | 1152 | 1582 | 884 | 808 | 1619 | 0 |

Bảng 6: Danh sách ma trận vuông 6 thành phố

Tìm đường đi ngắn nhất:

* Số lần lặp lại của giải thuật: 4
* Kích thước của population : 5
* Xác suất của lai ghép chéo: 0.8
* Xác suất của đột biến: 0.1

Bước 1: Khởi tạo quần thể:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5 Random Population | Path length | Fitness score | Fitness rank |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 | 459+808+1619+1211 | 4097 | 2 |
| Parent 2 | 2 -> 4-> 3-> 5-> 6 | 1091+459+1427+1619 | 4596 | 4 |
| Parent 3 | 4 -> 6-> 5-> 3-> 2 | 808+1619+1427+684 | 4538 | 3 |
| Parent 4 | 2 -> 6-> 5-> 3-> 4 | 1582+1619+1427+459 | 5087 | 5 |
| Parent 5 | 6 -> 4-> 3-> 2-> 5 | 808+459+684+1211 | 3162 | 1 |
|  | Chromosomes | Fitness function | Fitness value |  |
|  |  | Total fitness score: | 21480 |  |

Bảng 7: Khởi tạo quần thể

Tính độ phù hợp của những path đã random

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Random Population | Fitness score | Relative Fitness | Probability of selection |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 | 4097 | 21480/4097 = 5.24286 | 5.24286/25.66552 = 0.2043 |
| Parent 2 | 2 -> 4-> 3-> 5-> 6 | 4596 | 21480/4596 = 4.67362 | 4.67362/25.66552 = 0.1821 |
| Parent 3 | 4 -> 6-> 5-> 3-> 2 | 4538 | 21480/4538 = 4.73336 | 4.73336/25.66552 = 0.1844 |
| Parent 4 | 2 -> 6-> 5-> 3-> 4 | 5087 | 21480/5087 = 4.22252 | 4.22252/25.66552 = 0.1645 |
| Parent 5 | 6 -> 4-> 3-> 2-> 5 | 3162 | 21480/3162 = 6.79316 | 6.79316/25.66552 = 0.2647 |
| Sum of relative fitness | | | 25.66552 |  |

Bảng 8: Tính xác suất được chọn

Tính relative Fitness:

Tính xác suất chọn lựa (Probability of selection)

Bước 2: Chọn ra parent tốt nhất:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Random Population | Fitness score | Probability of selection | Probability distribution |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 | 4097 | 5.24286/25.66552 = 0.2043 | 0.2043 |
| Parent 2 | 2 -> 4-> 3-> 5-> 6 | 4596 | 4.67362/25.66552 = 0.1821 | 0.3864 |
| Parent 3 | 4 -> 6-> 5-> 3-> 2 | 4538 | 4.73336/25.66552 = 0.1844 | 0.5708 |
| Parent 4 | 2 -> 6-> 5-> 3-> 4 | 5087 | 4.22252/25.66552 = 0.1645 | 0.7353 |
| Parent 5 | 6 -> 4-> 3-> 2-> 5 | 3162 | 6.79316/25.66552 = 0.2647 | 1 |

Bảng 9: Tính xác suất tích lũy

Sẽ chọn ngẫu nhiên 1 random number = 0.826

Parent 5 có xác suất tích lũy > random number

* Parent 5 được chọn là parent đầu tiên

Chọn ngẫu nhiên 1 random number = 0.094

Parent 1 có xác suất tích lũy > random number

* Parent 1 được chọn là parent thứ 2

Bước 3: Reproduction:

|  |  |
| --- | --- |
| Parent 5 | 6 -> 4 -> 3 -> 2 -> 5 |
| Parent 1 | 3 -> 4 -> 6 -> 5 -> 2 |
| Parent 1 | 6 -> 4 -> 3 -> 2 -> 5 |
| Parent 2 | 3 -> 4 -> 6 -> 5 -> 2 |

Bảng 10: Chọn ra 2 parent

Áp dụng xác suất mà để lai ghép là 0.8, chúng ta sẽ random number, và số này quyết định có thực hiện lai ghép hay không!

Số ngẫu nhiên <= xác suất chéo sau đó chọn điểm chéo ngẫu nhiên tại bất kỳ vị trí nào trong chuỗi gốc.

Số ngẫu nhiên: 0,561 <=0,8

Thực hiện chéo!

Để tránh vi phạm TSP, chúng tôi thực hiện trao đổi trong Parent 1

|  |  |
| --- | --- |
| Parent 1 | 3 -> 4 | 6-> 2-> 5 |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |

Bảng 11: Chọn ra 2 parent rồi thực hiện lai chéo

Hoán đổi được thực hiện để tránh điền thành viên hai lần và vi phạm TSP

|  |  |
| --- | --- |
| Parent 1 | 3 -> 4 | 6-> 2-> 5 |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |

Bảng 12: Chọn ra 2 parent rồi thực hiện lai chéo

Chuyển thành công thành viên thứ 1 từ P2 sang P1

Bạn có thể được chuyển từ P2 sang P1 mà không cần trao đổi.

Vậy first child: 3 -> 4 -> 6 -> 2-> 5

|  |  |
| --- | --- |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |
| Parent 1 | 6 -> 4 | 3-> 2-> 5 |
| Parent 2 | 6 -> 4 | 3-> 5-> 2 |
| Parent 1 | 6 -> 4 | 3-> 2-> 5 |

Bảng 13: Chọn ra 2 parent rồi thực hiện lai chéo cho P2

Vậy second offspring or child: 6 -> 4 -> 3 -> 5-> 2

|  |  |
| --- | --- |
|  | Random Population |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 |
| Parent 2 | 2 -> 4-> 3-> 5-> 6 |
| Parent 3 | 4 -> 6-> 5-> 3-> 2 |
| Parent 4 | 2 -> 6-> 5-> 3-> 4 |
| Parent 5 | 6 -> 4-> 3-> 2-> 5 |
| Child 1 | 3 -> 4-> 6-> 2-> 5 |
| Child 2 | 6 -> 4-> 3-> 5-> 2 |

Bảng 14: Bảng population mới

Bước 4: Đột biến:

Số ngẫu nhiên = 0,273 > 0,1 (Xác suất đột biến)

Nhưng đột biến không được thực hiện trên cả hai con được tìm thấy.

Bước 5: Thay thế:

a) Đánh giá tổng quãng đường tham quan của quần thể ban đầu và hai thế hệ con

b) Chọn lọc những cá thể có hiệu quả sử dụng tốt nhất (bảo quản với áp suất chọn lọc 10%)

|  |  |  |
| --- | --- | --- |
|  | Random Polutation | Total length |
| Parent 1 | 1 -> 3 -> 4-> 6-> 5 -> 2 -> 1 | 4097 + 1484 + 1930 = 7511 |
| Parent 2 | 1 -> 2 -> 4-> 3-> 5 -> 6 -> 1 | 4596 + 1930 + 1152 = 7678 |
| Parent 3 | 1 -> 4 -> 6-> 5-> 3 -> 2 -> 1 | 4538 + 1096 + 1930 = 7564 |
| Parent 4 | 1 -> 2 -> 6-> 5-> 3 -> 4 -> 1 | 5087 + 1930 + 1096 = 8113 |
| Parent 5 | 1 -> 6 -> 4-> 3-> 2 -> 5 -> 1 | 3162 + 1152 + 991 = 5305 |
| C1 | 1 -> 3 -> 4 -> 6 -> 2-> 5 ->1 | 1484 + 459 + 808 + 1582 + 1211 + 991 = 6535 |
| C2 | 1 -> 6 -> 4 -> 3 -> 5-> 2 ->1 | 1152 + 808 + 459 + 1427 + 1211 + 1930 = 6987 |

Bảng 15: Bảng population mới cùng khoảng cách mới

Số ngẫu nhiên: 0,011

Chọn Parent 1 vì 0,2043 > 0,011 (số ngẫu nhiên)

Số ngẫu nhiên: 0,723

Chọn Parent 4 vì 0,753 > 0,723 (số ngẫu nhiên)

Parent đã chọn:

|  |  |
| --- | --- |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 |
| Parent 4 | 2 -> 6-> 5-> 3-> 4 |
| Parent 1 | 3 -> 4-> 6-> 5-> 2 |
| Parent 2 | 2 -> 6-> 5-> 3-> 4 |

Bảng 16: Bảng parent mới chọn

Random number: 0.594 < 0.8 Apply crossover

|  |  |
| --- | --- |
| Parent 1 | 3 -> 4 | 6-> 5-> 2 |
| Parent 2 | 2 -> 6 | 5-> 3-> 4 |
| Parent 1 | 2 -> 4 | 6-> 5-> 3 |
| Parent 2 | 2 -> 6 | 5-> 3-> 4 |
| Parent 1 | 2 -> 6 | 4-> 5-> 3 |
| Parent 2 | 2 -> 6 | 5-> 3-> 4 |

Bảng 17: Bảng parent mới chọn và thực hiện lai chéo

New child 3: 2 -> 6 -> 4 -> 5 -> 3

|  |  |
| --- | --- |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |
| Parent 1 | 2 -> 6 | 5-> 3-> 4 |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |
| Parent 1 | 3 -> 6 | 5-> 2-> 4 |
| Parent 2 | 3 -> 4 | 6-> 5-> 2 |
| Parent 1 | 3 -> 4 | 5-> 2-> 6 |

Bảng 18: Bảng parent mới chọn và thực hiện lai chéo trên P2

Child 4: 3 -> 4 -> 5-> 2-> 6

Mutation:

Random number: 0.607 > 0.1 Mutation Probability

Skip mutation

|  |  |  |
| --- | --- | --- |
|  | Random Polutation | Total length |
| Parent 1 | 1 -> 3 -> 4-> 6-> 5 -> 2 -> 1 | 4097 + 1484 + 1930 = 7511 |
| Parent 2 | 1 -> 2 -> 4-> 3-> 5 -> 6 -> 1 | 4596 + 1930 + 1152 = 7678 |
| Parent 3 | 1 -> 4 -> 6-> 5-> 3 -> 2 -> 1 | 4538 + 1096 + 1930 = 7564 |
| Parent 4 | 1 -> 2 -> 6-> 5-> 3 -> 4 -> 1 | 5087 + 1930 + 1096 = 8113 |
| Parent 5 | 1 -> 6 -> 4-> 3-> 2 -> 5 -> 1 | 3162 + 1152 + 991 = 5305 |
| C1 | 1 -> 3 -> 4 -> 6 -> 2-> 5 ->1 | 1484 + 459 + 808 + 1582 + 1211 + 991 = 6535 |
| C2 | 1 -> 6 -> 4 -> 3 -> 5-> 2 ->1 | 1152 + 808 + 459 + 1427 + 1211 + 1930 = 6987 |
| C3 | 1 -> 2 -> 6 -> 4 -> 5 -> 3->1 | 1930 + 1582 + 808 + 1046 + 459 + 1484 = 7309 |
| C4 | 1 -> 3 -> 4 -> 5-> 2 -> 6 ->1 | 1484 + 459 + 1046 + 1211 + 1582 + 1152 = 6934 |

Bảng 19: Bảng population gồm 4 con

Interation 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Random Polutation | Fitness score | Relative Fitness | Probability of selection | Probability distribution |
| Parent 5 | 6 -> 4 -> 3-> 2 -> 5 | 3162 | 20288/3162 = 6.416 | 0.252 | 0.252 |
| C1 | 3 -> 4 -> 6 -> 2-> 5 | 4060 | 20288/4060 = 4.997 | 0.196 | 0.448 |
| C2 | 6 -> 4 -> 3 -> 5-> 2 | 3905 | 20288/3905 = 5.195 | 0.204 | 0.652 |
| C3 | 2 -> 6 -> 4 -> 5 ->3 | 4863 | 20288/4863 = 4.172 | 0.164 | 0.916 |
| C4 | 3 -> 4 -> 5-> 2 -> 6 | 4298 | 20288/4298 = 4.720 | 0.185 | 1.00 |
| Total Fitness: | | | 20288 |  |  |

Bảng 20: Bảng thế hệ mới chọn ra cái tốt nhất

Random number: 0.409 choose C1: 0.448

Random number: 0.525 choose C2: 0.652

Select 2 parent:

|  |  |
| --- | --- |
| C1 | 3 -> 4 -> 6 -> 2-> 5 |
| C2 | 6 -> 4 -> 3 -> 5-> 2 |

Bảng 21: Chọn 2 parent dựa trên random

Random number: 0.336 <= 0.8 crossover

|  |  |
| --- | --- |
| P1 | 3 -> 4 | 6 -> 2-> 5 |
| P2 | 6 -> 4 | 3 -> 5-> 2 |
| P1 | 6 -> 4 | 3 -> 2-> 5 |
| P2 | 6 -> 4 | 3 -> 5-> 2 |

Bảng 22: Thực hiện lai chéo dựa trên P1

Child 5: 6 -> 4 -> 3 -> 2 -> 5

|  |  |
| --- | --- |
| P1 | 3 -> 4 | 6 -> 2-> 5 |
| P2 | 6 -> 4 | 3 -> 5-> 2 |
| P1 | 3 -> 4 | 6 -> 2-> 5 |
| P2 | 3 -> 4 | 6 -> 5-> 2 |

Bảng 23: Thực hiện lai chéo dựa trên P2

Child 6: 3 -> 4 -> 6 -> 5 -> 2

Random number: 0.53 > 0.1

Skip mutation

Tour length:

|  |  |  |
| --- | --- | --- |
|  | Random Polutation | Total length |
| Parent 1 | 1 -> 3 -> 4-> 6-> 5 -> 2 -> 1 | 4097 + 1484 + 1930 = 7511 |
| Parent 2 | 1 -> 2 -> 4-> 3-> 5 -> 6 -> 1 | 4596 + 1930 + 1152 = 7678 |
| Parent 3 | 1 -> 4 -> 6-> 5-> 3 -> 2 -> 1 | 4538 + 1096 + 1930 = 7564 |
| Parent 4 | 1 -> 2 -> 6-> 5-> 3 -> 4 -> 1 | 5087 + 1930 + 1096 = 8113 |
| Parent 5 | 1 -> 6 -> 4-> 3-> 2 -> 5 -> 1 | 3162 + 1152 + 991 = 5305 |
| C1 | 1 -> 3 -> 4 -> 6 -> 2-> 5 ->1 | 1484 + 459 + 808 + 1582 + 1211 + 991 = 6535 |
| C2 | 1 -> 6 -> 4 -> 3 -> 5-> 2 ->1 | 1152 + 808 + 459 + 1427 + 1211 + 1930 = 6987 |
| C3 | 1 -> 2 -> 6 -> 4 -> 5 -> 3->1 | 1930 + 1582 + 808 + 1046 + 459 + 1484 = 7309 |
| C4 | 1 -> 3 -> 4 -> 5-> 2 -> 6 ->1 | 1484 + 459 + 1046 + 1211 + 1582 + 1152 = 6934 |
| C5 | 1 -> 6 -> 4 -> 3 -> 2 -> 5->1 | 5305 |
| C6 | 1-> 3 -> 4 -> 6 -> 5 -> 2-> 1 | 7511 |

Bảng 24: 6 child mới xuất hiện trong bảng

C5 = P5 (Polution2)

C6 = P1 (Polution1)

New offspring polutation:

Random number: 0.392 choose C1 as parent

Random number: 0.352 choose C2 as parent

Random number: 0.692 choose C3 as parent

Random number: 0.734 choose C4 as parent

New parent C3,C4

Random number: 0.245 < 0.8 crossover

|  |  |
| --- | --- |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| C4 | 3 -> 4 -> 5 -> 2 -> 6 |
| P1 | 2 -> 6 | 4 -> 5 -> 3 |
| P2 | 3 -> 4 | 5 -> 2 -> 6 |
| P1 | 3 -> 4 | 6 -> 5 -> 2 |
| P2 | 3 -> 4 | 5 -> 2 -> 6 |

Bảng 25: Thực hiện lai chéo dựa trên P1

New child but like P1 : 3 -> 4 -> 6 -> 5 -> 2

|  |  |
| --- | --- |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| C4 | 3 -> 4 -> 5 -> 2 -> 6 |
| P2 | 2 -> 6 | 4 -> 5 -> 3 |
| P1 | 3 -> 4 | 5 -> 2 -> 6 |
| P2 | 2 -> 6 | 4 -> 5 -> 3 |
| P1 | 2 -> 6 | 5 -> 3 -> 4 |

Bảng 26: Thực hiện lai chéo dựa trên P2

New child but like P4: 2 -> 6 -> 5 -> 3 -> 4

No new offspring found

Random number: 0.367 choose C1

Random number: 0.760 choose C3

New parent C1 và C3

Random number: 0.564 < 0.8 crossover

|  |  |
| --- | --- |
| C1 | 3 -> 4 -> 6 -> 2-> 5 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| C1 | 2 -> 6 -> 4 -> 3-> 5 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |

Bảng 27: Thực hiện lai chéo dựa trên P1

Child 7: 2 -> 6 -> 4 -> 3-> 5

Total length : 7197

|  |  |
| --- | --- |
| C1 | 3 -> 4 -> 6 -> 2-> 5 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| C1 | 3 -> 4 -> 6 -> 2-> 5 |
| C3 | 3 -> 4 -> 6 -> 5 -> 2 |

Bảng 28: Thực hiện lai chéo dựa trên P2

Child 8 but like P1

Random number: 0.392 choose C1 as parent

Random number: 0.644 choose C2 as parent

Random number: 0.724 choose C3 as parent

Random number: 0.967 choose C4 as parent

**Random number: 0.630 choose C2 as parent**

**Random number: 0.102 choose P5 as parent**

Random number: 0.366 < 0.8 => crossover to follow

|  |  |
| --- | --- |
| C2 | 6 -> 4 | 3-> 5-> 2 |
| Parent 5 | 6 -> 4 | 3-> 2 -> 5 |

Bảng 29: Thực hiện lai chéo dựa trên P1

6 -> 4 -> 3-> 5-> 2 => Nothing new

|  |  |
| --- | --- |
| Parent 5 | 6 -> 4 | 3-> 2 -> 5 |
| C2 | 6 -> 4 | 3-> 5-> 2 |

Bảng 30: Thực hiện lai chéo dựa trên P2

6 -> 4-> 3-> 2 -> 5 => Nothing new

Crossover again:

|  |  |
| --- | --- |
| C2 | 6 -> 4 -> 3-> 5 | 2 |
| Parent 5 | 6 -> 4-> 3-> 2 | 5 |

Bảng 31: Thực hiện lai chéo dựa trên P1

6 -> 4 -> 3-> 2 -> 5 => Nothing new

|  |  |
| --- | --- |
| Parent 5 | 6 -> 4-> 3-> 2 | 5 |
| C2 | 6 -> 4 -> 3-> 5 | 2 |

Bảng 32: Thực hiện lai chéo dựa trên P2

6 -> 4 -> 3-> 5 ->2=> Nothing new

Random number: 0.072 choose P5 as parent

Random number: 0.508 choose C2 as parent

Random number: 0.499 choose C2 as parent

Random number: 0.035 choose P5 as parent

**Random number: 0.072 choose P5 as parent**

**Random number: 0.922 choose C4 as parent**

**Random number = 0.694 < 0.8 crossover**

|  |  |
| --- | --- |
| Parent 5 | 6 -> 4-> 3-> 2 -> 5 |
| C4 | 3 -> 4 -> 5-> 2 -> 6 |

Bảng 33: Thực hiện lai chéo dựa trên P1

**New:** 3 -> 4-> 6-> 2 -> 5 same C1

|  |  |
| --- | --- |
| C4 | 3 -> 4 -> 5-> 2 -> 6 |
| Parent 5 | 6 -> 4-> 3-> 2 -> 5 |

Bảng 34: Thực hiện lai chéo dựa trên P2

**New child C8:** 6 -> 4 -> 5-> 2 -> 3

Total length: 6385

Random number: 0.274 > 0.1 no mutation

Till now:

|  |  |  |
| --- | --- | --- |
|  | Total length |  |
| Parent 5 | 5305 |  |
| C1 | 6535 |  |
| C2 | 6987 |  |
| C3 | 7309 |  |
| C4 | 6934 |  |
| C5 | 5305 |  |
| C6 | 7511 |  |
| C7 | 7197 | Replace C3 |
| C8 | 6385 | Replace C2 |

Bảng 35: Bảng gồm 8 child mới

Random number: 0.929 choose C4 as parent

Random number: 0.790 choose C3 as parent

Random number: 0.940 choose C4 as parent

Random number: 0.710 choose C3 as parent

Random number: 0.314 choose C1 as parent

Random number: 0.361 choose C2 as parent

**Random number: 0.362 choose C2 as parent**

**Random number: 0.617 choose C3 as parent**

Random number: 0.769 < 0.8 => crossover

|  |  |
| --- | --- |
| C2 | 6 -> 4 -> 3 -> 5-> 2 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| C2 | 6 -> 4 | 3 -> 5-> 2 |
| C3 | 2 -> 6 | 4 -> 5 -> 3 |

Bảng 36: Thực hiện lai chéo dựa trên P1

New child: 2 -> 4 -> 3 -> 5-> 6

Total: 7965 (unfit)

|  |  |
| --- | --- |
| C3 | 2 -> 6 | 4 -> 5 -> 3 |
| C2 | 6 -> 4 | 3 -> 5-> 2 |

Bảng 37: Thực hiện lai chéo dựa trên P2

New child C9: 6 -> 4 -> 2 -> 5-> 3

Total: 7173 Replace C7

Random number: 0.756 > 0.1 no mutation

Random number: 0.417 choose C1 as parent

Random number: 0.638 choose C2 as parent

**Random number: 0.298 choose P5 as parent**

**Random number: 0.687 choose C3 as parent**

Random number: 0.652 <0.8 crossover

|  |  |
| --- | --- |
| Parent 5 | 6 -> 4-> 3 -> 2 -> 5 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| Parent 5 | 2 ->4 -> 3 -> 6 -> 5 |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |

Bảng 38: Thực hiện lai chéo dựa trên P1

New child C10: 2 ->6 -> 3 -> 4 -> 5

Total length: 6892

|  |  |
| --- | --- |
| C3 | 2 -> 6 -> 4 -> 5 -> 3 |
| Parent 5 | 6 -> 4-> 3 -> 2 -> 5 |
| C3 | 6 -> 2 -> 4 -> 5 -> 3 |
| Parent 5 | 6 -> 4 -> 3 -> 2 -> 5 |

Bảng 39: Thực hiện lai chéo dựa trên P2

Nothing child: 6 -> 4 -> 2 -> 5 -> 3 same C9

Total length: 7173

|  |  |  |
| --- | --- | --- |
|  | Total length |  |
| Parent 5 | 5305 |  |
| C1 | 6535 |  |
| C2 | 6987 |  |
| C3 | 7309 |  |
| C4 | 6934 |  |
| C5 | 5305 |  |
| C6 | 7511 |  |
| C7 | 7197 | Replace C3 |
| C8 | 6385 | Replace C2 |
| C9 | 7173 |  |
| C10 | 6892 |  |

Bảng 40: 10 child mới được tạo

Iteration 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Polutation | Fitness score | Relative Fitness | Probability of selection | Probability distribution | Length |
| P5 | 6 -> 4 -> 3-> 2 -> 5 | 3162 | 19717/3162 = 6.236 | 0.246 | 0.246 | 5305 |
| C8 | 6 -> 4 -> 5-> 2 -> 3 | 3749 | 19717/3749 = 5.259 | 0.207 | 0.453 | 6385 |
| C10 | 2 -> 6 -> 3-> 4 -> 5 | 3971 | 19717/3971 = 4.965 | 0.196 | 0.649 | 6892 |
| C9 | 6 -> 4 -> 2 -> 5-> 3 | 4537 | 19717/4863 = 4.346 | 0.171 | 0.820 | 7173 |
| C4 | 3 -> 4 -> 5-> 2 -> 6 | 4298 | 19717/4298 = 4.587 | 0.181 | 1.00 | 6934 |
| Total Fitness: | | | 19717 |  |  |  |

Bảng 41: Thế hệ thứ 3 của quần thể này

Iteration 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Polutation | Fitness score | Relative Fitness | Probability of selection | Probability distribution | Length |
| P5 | 6 -> 4 -> 3-> 2 -> 5 | 3162 | 18931/3162 = 5.987 | 0.237 | 0.237 | 5305 |
| C11 | 5 -> 4 -> 3-> 2 -> 6 | 3771 | 18931/3771 = 5.020 | 0.199 | 0.436 | 5753 |
| C12 | 5 -> 6 -> 3-> 2 -> 4 | 4278 | 18931/4278 = 4.425 | 0.175 | 0.611 | 6365 |
| C8 | 6 -> 4 -> 5-> 2 -> 3 | 3749 | 18931/3749 = 5.050 | 0.200 | 0.811 | 6385 |
| C10 | 2 -> 6 -> 3-> 4 -> 5 | 3971 | 18931/3971 = 4.767 | 0.189 | 1.00 | 6892 |
| Total Fitness: | | | 18931 |  |  |  |

Bảng 42: Thế hệ thứ 4 của quần thể này

Thuật toán di truyền được mô tả chi tiết trong bài viết đang thực hiện giải bài toán người du lịch (TSP). Dưới đây là một phân tích từng bước:

Bước 1: Khởi tạo dân số ban đầu

Bước 2: Chọn bố và mẹ

Bố mẹ được chọn dựa trên điểm thích hợp của họ.

Sử dụng phân phối xác suất để chọn.

Số ngẫu nhiên được tạo ra để chọn bố mẹ.

Sử dụng crossover dựa trên xác suất crossover (0.8).

Bước 3: Sản xuất con

Chọn hai bố mẹ, và crossover được áp dụng để tạo ra hai con.

Crossover một điểm được sử dụng với một điểm crossover ngẫu nhiên.

Mutation được áp dụng để duy trì một giải pháp hợp lệ cho TSP.

Bước 4: Đột biến

Đột biến được áp dụng với một xác suất nhất định (0.1) cho mỗi con.

Số ngẫu nhiên được tạo ra để quyết định liệu có xảy ra đột biến hay không.

Bước 5: Thay thế

Tổng chiều dài của hành trình của dân số ban đầu và con cái được đánh giá.

Chọn các cá thể có hiệu suất tốt nhất để tạo ra thế hệ tiếp theo, với tỷ lệ bảo toàn là 10%.

Phép lặp:

Quá trình được lặp lại cho một số lần lặp xác định (4 trong trường hợp này).

Dân số mới được tạo ra, và các cá thể tốt nhất được chọn cho thế hệ tiếp theo.

Kết quả cuối cùng: Sau nhiều lần lặp, thuật toán hội tụ thành một tập giải pháp, mỗi giải pháp đại diện cho một hành trình với điểm thích hợp đi kèm.

***Độ phức tạp của thuật toán:***

* Space Complexity:

Độ phức tạp về không gian chủ yếu phụ thuộc vào kích thước của quần thể (population size) và số lượng thành phố trong TSP. Mỗi cá thể trong quần thể là một giải pháp tiềm năng, và mỗi giải pháp cần lưu trữ thông tin về thứ tự các thành phố. Nếu có N thành phố và P cá thể trong quần thể, độ phức tạp về không gian sẽ là O(N×P).

* Time Complexity:

Khởi tạo quần thể ban đầu: Thực hiện P lần cho mỗi cá thể.

Tính độ thích nghi (Fitness Function): Thực hiện cho mỗi cá thể, chi phí O(N) cho mỗi cá thể, tổng cộng là O(N×P).

Lựa chọn (Selection), Lai ghép (Crossover) và Đột biến (Mutation): Những bước này thường được thực hiện qua nhiều thế hệ, với số lần lặp G (số thế hệ). Mỗi thế hệ thực hiện các hoạt động trên tất cả cá thể trong quần thể. Do đó, độ phức tạp thời gian cho mỗi thế hệ là O(N×P), và tổng thời gian qua G thế hệ là O(G×N×P).

Tổng kết, độ phức tạp của thuật toán phụ thuộc chủ yếu vào số lượng thành phố N, kích thước của quần thể P, và số lượng thế hệ G. Độ phức tạp thời gian là O(G×N×P) và độ phức tạp không gian là O(N×P).

# CHƯƠNG 5 - DỮ LIỆU ĐẦU VÀO VÀ ĐẦU RA

**Đầu vào: Ma trận vuông các thành phố trong file csv**

0,1930,1484,1096,991,1152

1930,0,684,1091,1211,1582

1484,684,0,459,1427,884

1096,1091,459,0,1046,808

991,1211,1427,1046,0,1619

1152,1582,884,808,1619,0

**Đầu ra: Interation, population, invidual và length, best tour.**

Iterations: 5

Cities: 6

PopulationSize: 5

crossoverProbability: 0.8

mutationProbability: 0.1

Iteration 5 - Population:

Individual 1: [5, 2, 3, 4, 6, 1], Length: 5305

Individual 2: [5, 2, 3, 4, 6, 1], Length: 5305

Individual 3: [5, 2, 3, 4, 6, 1], Length: 5305

Individual 4: [5, 2, 3, 4, 6, 1], Length: 5305

Individual 5: [5, 2, 3, 4, 6, 1], Length: 5305

Selected parent index: 0, Probability Distribution: 0.2

Selected parent index: 3, Probability Distribution: 0.8

Selected parent index: 1, Probability Distribution: 0.4

Selected parent index: 4, Probability Distribution: 1.0

Selected parent index: 0, Probability Distribution: 0.2

Best tour: [5, 2, 3, 4, 6, 1], Length: 5305

Final Best Tour: [5, 2, 3, 4, 6, 1], Length: 5305

Runtime: 0.069 seconds

# CHƯƠNG 6 - TẠO RA DỮ LIỆU ĐẦU VÀO

Nhập số lượng thành phố: Mã yêu cầu người dùng nhập vào số lượng các thành phố thông qua giao diện dòng lệnh.

Khởi tạo và ghi dữ liệu vào File: Mã khởi tạo một BufferedWriter để ghi dữ liệu vào tệp tin với tên được cung cấp. Dòng đầu tiên của tệp tin được viết là "88888," để đánh dấu khoảng cách tự thành phố đến chính nó. Sau đó, mã viết các chỉ số của thành phố (từ 1 đến số lượng thành phố đã nhập) vào tệp tin.

Tạo ma trận khoảng cách: Một ma trận hai chiều distances được khởi tạo với kích thước (numCities + 1) x (numCities + 1), và tất cả giá trị được thiết lập mặc định là -1. Mã sau đó lặp qua ma trận để thiết lập khoảng cách giữa các thành phố.

Tính toán và ghi khoảng cách: Mỗi cặp thành phố (i, j) sẽ được gán một khoảng cách ngẫu nhiên (trừ khi i == j, khi đó khoảng cách là 88888, tức là không thể di chuyển từ một thành phố đến chính nó). Khoảng cách này được ghi vào tệp tin, và cùng một khoảng cách sẽ được sử dụng cho cả (i, j) và (j, i) để đảm bảo tính đối xứng.

Đóng tệp và scanner: Sau khi quá trình ghi tệp hoàn tất, BufferedWriter và Scanner sẽ được đóng lại.

Xử lý ngoại lệ: Nếu có lỗi IO xảy ra (ví dụ như tệp không tìm thấy, vấn đề quyền truy cập), mã sẽ bắt và in ra thông báo lỗi.

Các bước được áp dụng:

class DataGenerator

method generateTestData(filename)

try

open file with BufferedWriter using filename

initiate a Scanner for user input

print "Enter number of cities: "

read numCities from user input

write "88888," to file

for i from 1 to numCities

write i to file

if i is less than numCities then

write "," to file

end if

end for

write new line to file

create a Random object

declare distances as a two-dimensional array of size numCities+1 by numCities+1

// Initialize all distances to -1

for each row i from 1 to numCities

for each column j from 1 to numCities

set distances[i][j] to -1

end for

end for

// Calculate and write distances

for each row i from 1 to numCities

write i to file

for each column j from 1 to numCities

if distances[i][j] is -1 then

if i equals j then

set distances[i][j] to 88888

else

generate a random number between 1 and 100

set distances[i][j] and distances[j][i] to this random number

end if

end if

write "," and distances[i][j] to file

end for

write new line to file

end for

close scanner

catch any IOException

print the error stack trace

end try

end method

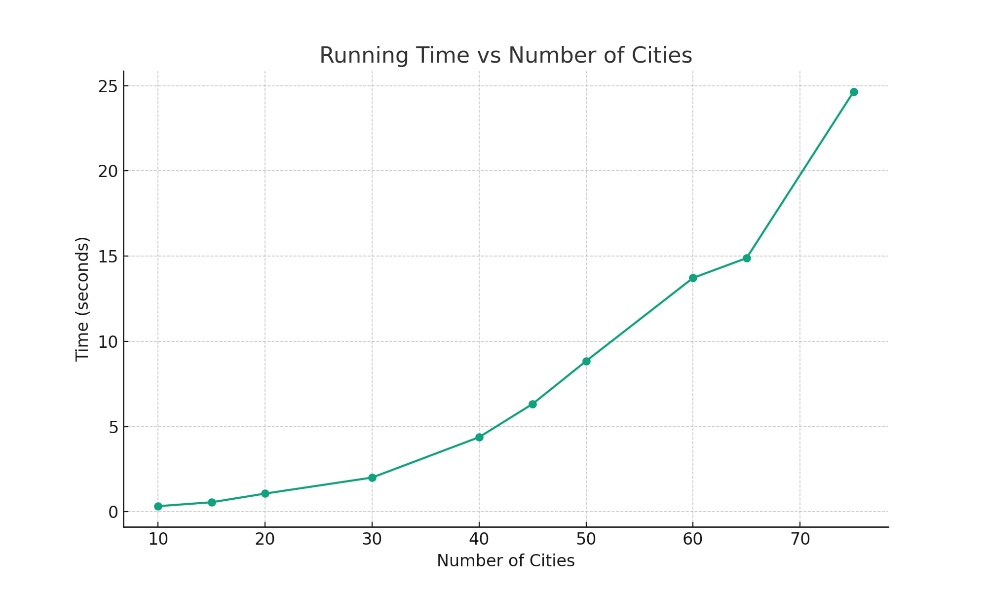
end class

# CHƯƠNG 7 - KẾT QUẢ THỬ NGHIỆM VÀ KẾT LUẬN

**Thống kê thời gian chạy của thuật toán Genetic với TSP:**

|  |  |
| --- | --- |
| Số lượng thành phố | Thời gian chạy |
| 10 | 0.328 |
| 15 | 0.557 |
| 20 | 1.067 |
| 30 | 2.007 |
| 40 | 4.375 |
| 45 | 6.326 |
| 50 | 8.841 |
| 60 | 13.723 |
| 65 | 14.885 |
| 75 | 24.646 |

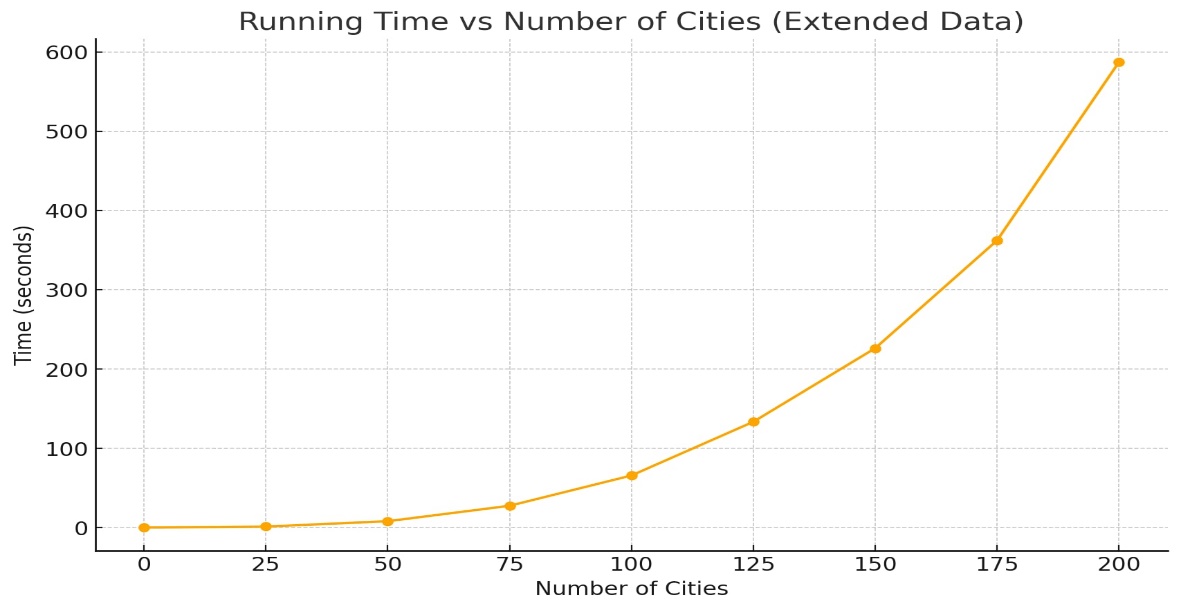
Bảng 43: Bảng thời gian chạy thuật toán với 75 thành phố



Hình 1:Biểu đồ thời gian thực hiện giải thuật của 75 thành phố

|  |  |
| --- | --- |
| Số lượng thành phố | Thời gian chạy |
| 0 | 0 |
| 25 | 1.297 |
| 50 | 8.144 |
| 75 | 27.567 |
| 100 | 65.776 |
| 125 | 133.446 |
| 150 | 226.217 |
| 175 | 362.171 |
| 200 | 586.994 |

Bảng 44: Bảng thời gian chạy thuật toán với 200 thành phố



Hình 2:Biểu đồ thời gian thực hiện giải thuật của 200 thành phố

# CHƯƠNG 8 - SO SÁNH VỚI CÁC THUẬT TOÁN KHÁC

**Thuật toán Ant Colony:**

Bài báo đề xuất cách cải tiến thuật toán Ant Colony để hỗ trợ tìm ra đường đi ngắn hơn cho bài toán người bán hàng. Bài toán người bán hàng yêu cầu tìm ra đường đi ngắn nhất cho người bán hàng đi qua các thành phố và cuối cùng quay về lại thành phố xuất phát, mỗi thành phố chỉ được ghé thăm một lần, biết rằng tất cả các thành phố đều có đường đi đến với nhau và khoảng cách giữa các thành phố là biết trước. Có rất nhiều thuật toán giải quyết được bài toán này. Một trong những thuật toán được nghiên cứu nhiều và giải quyết khá hiệu quả cho bài toán này là thuật toán Ant Colony (thuật toán đàn kiến). Thuật toán Ant Colony có những hỗ trợ tìm kiếm rất mạnh mẽ và tỏ ra khá thích hợp với những bài toán có không gian tìm kiếm cực lớn. Tuy nhiên khi áp dụng thuật toán Ant Colony cho bài toán người bán hàng thì chi phí vẫn còn khá cao. Vì vậy nhóm chúng tôi thiết kế giải thuật cải tiến thuật toán Ant Colony để tìm lời giải tối ưu hơn cho bài toán người bán hàng. Chúng tôi đã tiến hành xây dựng thử nghiệm với nhiều bộ dữ liệu đầu vào để so sánh giữa thuật toán cải tiến với thuật toán Ant Colony nhằm đánh giá một cách chính xác và khách quan nhất. Kết quả cho thấy thuật toán cải tiến của nhóm chúng tôi đề xuất đã có cải thiện đáng kể về chi phí so với thuật toán Ant Colony.

**Thuật toán BruteForce:**

Trong bài toán TSP, một người du lịch cần phải thăm một loạt các thành phố, mỗi thành phố đúng một lần, và cuối cùng trở về điểm xuất phát, sao cho tổng quãng đường di chuyển là ngắn nhất.

Khái quát: Thuật toán Brute Force tạo ra tất cả các hoán vị có thể của các thành phố (trừ thành phố xuất phát) để tìm ra một lộ trình sao cho tổng quãng đường di chuyển là ít nhất.

Tạo hoán vị: Liệt kê mọi hoán vị có thể của các thành phố. Ví dụ, nếu có 4 thành phố {A, B, C, D}, các hoán vị có thể là ABCD, ABDC, ACBD, và cứ thế.

Tính toán quãng đường: Đối với mỗi hoán vị, tính tổng quãng đường di chuyển của người du lịch. Điều này bao gồm cả quãng đường từ điểm xuất phát đến thành phố đầu tiên trong hoán vị và từ thành phố cuối cùng trở về điểm xuất phát.

So sánh và chọn lựa: So sánh tổng quãng đường của tất cả các hoán vị và chọn lựa hoán vị có tổng quãng đường ngắn nhất.

**Độ phức tạp thời gian của thuật toán Brute Force cho TSP là O(n!), với n là số lượng thành phố. Điều này có nghĩa là thời gian thực thi tăng lên một cách nhanh chóng với mỗi thành phố thêm vào.**

**So sánh:**

* **5 thành phố với BruteForce:**

Best tour: [2, 1, 3, 6, 4, 5], Length: 57

Final Best Tour: [2, 1, 3, 6, 4, 5], Length: 57

Runtime: 0.263 seconds

Brute Force Runtime: 0.003 seconds

Final Best Tour Brute: [1, 2, 5, 4, 6, 3]

Length of Best Tour Brute: 57

* **10 thành phố với BruteForce:**

Best tour: [5, 11, 10, 4, 1, 6, 9, 3, 7, 8, 2], Length: 195

Final Best Tour: [5, 11, 10, 4, 1, 6, 9, 3, 7, 8, 2], Length: 195

Runtime: 0.469 seconds

Brute Force Runtime: 109.366 seconds

Final Best Tour Brute: [1, 8, 6, 9, 7, 3, 5, 2, 4, 10, 11]

Length of Best Tour Brute: 127

# TÀI LIỆU THAM KHẢO