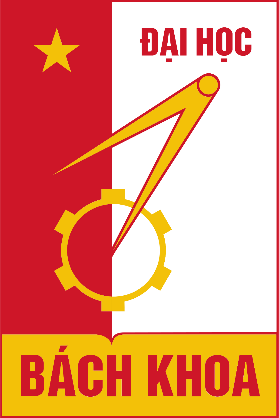
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



**Họ tên sinh viên: Lê Anh Đức**

**Mã số sinh viên: 20190042**

**Giảng viên hướng dẫn: TS. Trịnh Tuấn Đạt**

**Đồ án 1:**

**ÁP DỤNG GIẢI THUẬT DI TRUYỀN VÀ GIẢI THUẬT TỐI ƯU ĐÀN KIẾN CHO BÀI TOÁN NGƯỜI ĐI DU LỊCH**

1. **Giới thiệu**

Bài toán người du lịch (Travelling Salesman Problem - TSP) là một bài toán tối ưu tổ hợp được nghiên cứu trong lĩnh vực tối ưu hóa và khoa học máy tính. Bài toán TSP được mô tả như sau: cho một tập các thành phố, chúng ta cần tìm một chu trình đi qua tất cả các thành phố, mỗi thành phố đúng một lần sao cho tổng khoảng cách đi qua các thành phố là nhỏ nhất. Bài toán TSP có thể được biễu diễn bởi một đồ thị , trong đó là tập các thành phố tương ứng các đỉnh của đồ thị và là tập các đường đi giữa các thành phố tương ứng với cạnh của đồ thị. Mỗi cạnh được gán một giá trị tương ứng là khoảng cách của thành phố đến . Như vậy, bài toán TSP tương đương với việc tìm chu trình Hamilton có độ dài ngắn nhất trên một đồ thị có trọng số. Bài toán TSP thuộc lớp bài toán NP - khó (NP - hard) vì có độ phức tạp tính toán là hàm giai thừa [1] và có thể được giải bằng cách sử dụng thuật toán vét cạn (exhausive algorithm) hoặc thuật toán tìm kiểm xấp xỉ (approximation algorithm).

Thuật toán vét cạn cho phép tìm được chu trình có chiều dài ngắn nhất cho bài toán TSP, đó là tìm tất cả các chu trình Hamilton trong đồ thị và sau đó lấy chu trình có chiều dài ngắn nhất. Với đồ thị có đỉnh sẽ có tối đa chu trình Haminton, tức là độ phức tạp của thuật toán là hàm giai thừa, do vậy khi số đỉnh của đồ thị tăng thì số phép tính trong thuật toán sẽ tăng cấp giai thừa. Ví dụ với đồ thị 25 đỉnh, thuật toán vét cạn cần thực hiện phép tính. Rõ ràng rằng sử dụng thuật toán vét cạn để giải bài toán TSP là không khả thi khi số đỉnh đồ thị tăng lên nhanh.

Một hướng tiếp cận khác để giải bài toán TSP hiệu quả là sử dụng các giải thuật tìm kiếm xấp xỉ đề tìm một chu trình đúng hoặc gần đúng trong một thời gian chấp nhận được. Các giải thuật xấp xỉ có thể được sử dụng để giài bài toán TSP như giải thuật láng giềng gần nhất (nearest neighbour algorithm), giải thuật di truyền (genetic algorithm), thuật toán tối ưu hóa đàn kiến (ant colony optimization).

1. **Giải thuật di truyền (Genetic algorithm)**

Giải thuật di truyền (Genetic Algorithm - GA) được phát triển bởi Holland và cộng sự trong thập niên 1960 tại trường đại học Michigan với ý tưởng dựa trên quá trình tiến hóa và chọn lọc tự nhiên. Đặc tính quan trọng nhất của giải thuật GA là không sử dụng các điều kiện truyền thống như điều kiện liên tục hay khả vi làm điều kiện tiên quyết. Để tìm nghiệm, GA thực hiện tìm kiếm song song đồng thời trong “quần thể” (population), trong đó khái niệm “nhiễm sắc thể” (chromosome) được sử dụng như là nghiệm của bài toán.

|  |
| --- |
| Ý tưởng chính của giải thuật di truyền: |
| 1. Vào: một quân thế và một hàm đánh gia độ thích nghi (fitness) 2. Ra: một nhiễm sắc thể, tức là nghiệm của bài toán 3. Khởi tạo quần thể, tỉ lệ đột biến , xác suất chọn lọc . 4. Mã hóa nhiễm sắc thể 5. Repeat 6. Đánh giá độ thích nghi 7. Chọn lọc 8. Lai ghép 9. Đột biến 10. Until (thõa mãn điều kiện dừng). 11. Trả về nghiệm |

Giải thuật tối ưu hóa đàn kiến (Ant Colony Optimization - ) là một phương pháp tìm kiếm nghiệm tối ưu xấp xỉ dựa trên ý tưởng mô phỏng cách tìm đường đi của các con kiến tự nhiên từ tổ tới nguồn thức ăn của chúng. Khi tìm đường đi, đàn kiến trao đổi thông tin gián tiếp và hoạt động theo phương thức tự tổ chức. Cụ thể, khi đi tìm mồi các con kiến sẽ rải vết mùi (pheromone) dùng để đánh dấu đường đi. Bằng cách cảm nhận vết mùi, các kiến có thể lần theo đường đi đến nguồn thức ăn được các con kiến khác khám phá theo phương thức chọn ngẫu nhiên có định hướng theo nồng độ vết mùi. Kiến chịu ảnh hưởng vết mùi của các con kiến khác (đường đi có nồng độ vết mùi càng cao thì xác suất được kiến chọn càng lớn) về quyết định chọn đường đi chính là ý tưởng thiết kế thuật toán .

Trong báo cáo, tôi nghiên cứu áp dụng giải thuật GA và giải thuật bằng thực nghiệm đề xác định giải thuật nào giải bài toán TSP hiệu quả hơn theo nghĩa đạt được chất lượng nghiệm và thời gian tìm kiếm nghiệm tốt hơn.

1. **Áp dụng GA và ACO cho bài toán TSP**
2. **Áp dụng GA cho TSP**

thể được chọn ngẫu nhiên theo một dạng mã hóa nào đó mà thường được gọi là nhiễm sắc thể. Mỗi nhiễm sắc thể được đánh giá độ thích nghi thông qua một hàm thích nghi (fítness). Chọn lọc là quá trình chọn các nhiễm sắc thể tốt theo nghĩa của hàm thích nghi để lai ghép sinh ra thế hệ tiếp theo. Lai ghép và đột biến nhằm sinh ra một thế hệ mới tốt hơn thế hệ trước đó. Rõ ràng, GA là giải thuật dựa trên ý tưởng của quá trình tiến hóa và chọn lọc tự nhiên nhằm sinh ra thế hệ tiếp theo tốt hơn thế hệ trước đó theo nghĩa của hàm thích nghi.

Ký hiệu là tập gồm thành phố, ký hiệu là khoảng cách giữa 2 thành phố và . Trong nghiên cứu này, tôi thử nghiệm với đồ thị vô hướng, do đó tôi giả sử rằng và như vậy nghiệm của bài toán TSP là một hoán vị của thành phố. Để áp dụng giải thuật GA cho bài toán TSP, tôi định nghĩa các phép mã hóa, chọn lọc, lai ghép và đột biến như sau:

* Mã hóa nhiễm sắc thể: Phương pháp biểu diễn đường dẫn được sử dụng để biểu diễn các nghiệm (nhiễm sắc thể) của bải toán. Ví dụ với , các nghiệm có thể là các hoán vị .
* Hàm thích nghi: Mục tiêu của bài toán là tìm chu trình ngắn nhất đi qua tất cả các thành phố với mỗi thành phố đúng một lần, do vậy hàm thích nghi của giải thuật được định nghĩa như công thức (1). Điều này có nghĩa rằng những cá thể tốt là những cá thể có hàm thích nghi là bé.
* Chọn lọc các nhiễm sác thể: Để chọn lọc các nhiễm sắc thể cho thế hệ sau thì mỗi nhiễm sắc thể cần được đánh giá độ thích nghi. Sau đó, các nhiễm sắc thể được sắp xếp giảm dần theo hàm thích nghi. Giả sử là số cá thể được giữ lại và cũng chính là số cá thể được chọn để lai ghép, khi đó xác suất để chọn cá thể thứ ) được định nghĩa như công thức (2).
* Lai ghép nhiễm sắc thể: Nếu sử dụng toán tử lai ghép cho bài toán TSP như giải thuật di truyền nhị phân [2], [3] thì giải thuật sinh ra lỗi. Ví dụ nếu 2 cá thể , và điểm ghép thì sẽ tạo ra 2 con là và , . Hiển nhiên 2 con tạo ra không phải là 2 chu trình. Do vậy, tôi định nghĩa toán tử lai ghép như sau:
* Chọn một vị trí ngẫu nhiên trong 2 cá thể và 2 cá thể hoán đổi 2 số nguyên ở vị trí được chọn để tạo ra 2 cá thể mới.
* Tiếp tục hoán đổi 2 số nguyên trong 2 cá thể tạo ra nếu bị trùng giá trị cho đến khi không có giá trị trùng trong mỗi cá thể.

Ví dụ với 2 cá thể cha, , và mẹ, , với điểm ghép bắt đầu . Khi đó toán tử lai ghép được thực hiện như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hai cá thể cha và mẹ: | Bước 1 | Bước 2 | Bước 3 | Bước 4 |
|  |  |  |  | 345216 |

* Đột biến nhiễm sắc thể: Toán tử đột biến được thực hiện với một xác suất nhỏ nhằm tránh bẫy cục bộ, đó là hoán đổi 2 vị trí bất kỳ của một nhiễm sắc thể sau khi thực hiện toán tử lai ghép.

1. **Áp dụng giải thuật ACO cho TSP**

Giải thuật là một lớp các giải thuật dựa trên ý tưởng mô phỏng cách tìm đường đi của các con kiến tự nhiên từ tổ tới nguồn thức ăn của chúng. Giải thuật cho bài toán TSP được mô tả như sau:

|  |
| --- |
| Giải thuật ACO cho TSP |
| 1. Vào: Một đồ thị có trọng số 2. Ra: Một chu trình 3. Khởi tạo tham số, ma trận vết mùi , khởi tạo con kiến 4. Repeat 5. for to do 6. Kiến thứ xây dựng lời giải 7. Cập nhật vết mùi theo luật cập nhật cục bộ 8. end for 9. Cập nhật vết mùi theo luật cập nhật tổng thể 10. Cập nhật chu trình tối ưu nhất 11. Until (thõa mãn điều kiện dừng) 12. Trả về chu trình tìm được |

Ban đầu mỗi kiến được khởi tạo ngẫu nhiên ở một thành phố xuất phát. Trong quá trình tìm nghiệm, mỗi con kiến ở thành phố chọn thành phố lân cân dựa trên xác suất chuyển trạng thái (random-proportional rule) được định nghĩa bởi (3):

(3)

trong đó là vết mùi của cạnh là giá trị hueristic của cạnh (i,j), là tập các thành phố lân cận mà kiến chưa ghé thăm và là một tham số xác định quan hệ giữa vết mùi và độ dài của các cạnh . Sau khi tất cả các con kiến hoàn thành chu trình, thuật toán sẽ tiến hành cập nhật tổng thể nhằm thay đổi vết mùi trên các cạnh của đồ thị theo luật (4):

tham số bay hơi của vết mùi, là chiểu dài của chu trình tạo bởi kiến và là số kiến. Mục đích của luật cập nhật tổng thể là cập nhật càng nhiều giá trị vết mùi cho các chu trình ngắn

* Luật chuyển trạng thái: Luật chuyển trạng thái của mỗi con kiến di chuyển từ thành phố đến thành phố dựa trên công thức (5):

Với ưu chu dài tham số xác định trước và là một giá trị được xác định dựa theo (3).

* Luật cập nhật vết mùi tổng thể. Sau khi tất cả các cong kiến hoàn thành chu trình cần cập nhật lại vết mùi của toàn bộ các con kiến, dựa theo chu trình có độ dài ngắn nhất

(6)

trong đó là một tham số bay vết mùi.

* Luật cập nhật vết mùi cục bộ: Khi mỗi con kiến đi qua một cạnh (i,j) nào đó, luật cập nhật vết mùi cục bộ được thực hiện như luật (7):

Trong đó là một tham số, là tham số được xác định bởi thực nghiệm. Trong báo cáo này tôi chọn .

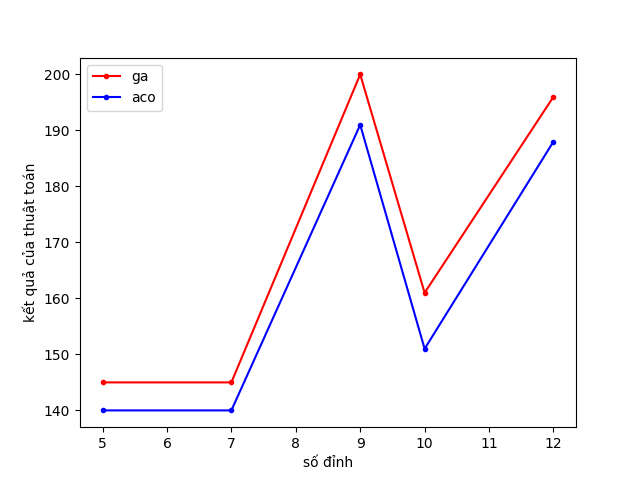
1. **Thực nghiệm**
2. Chọn các tham số cho thuật toán:

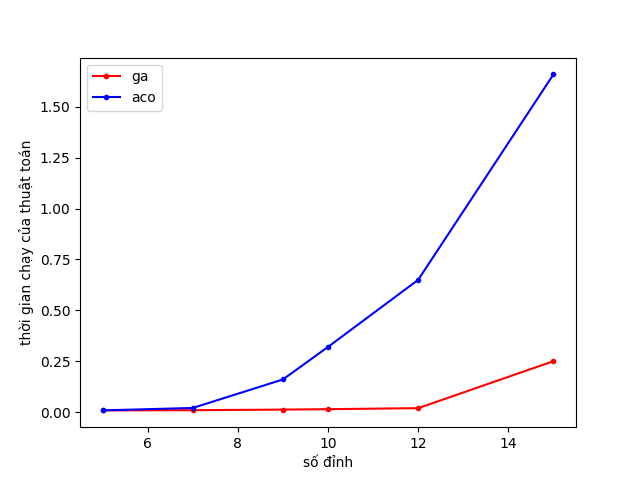
Sử dụng các giải thuật quy hoạch động trên bitmask cho những đồ thị nhỏ, chọn được các tham số cho 2 giải thuật như sau: Đối với TSP, chọn được xác suất đột biến là , đối với thuật toán ACO chọn được , số con kiến được khởi tạo bằng khoảng 80% số đỉnh.

Cả 2 thuật toán đều chọn số vòng lặp khoảng 50 lần số đỉnh

1. So sánh trên các bộ dữ liệu

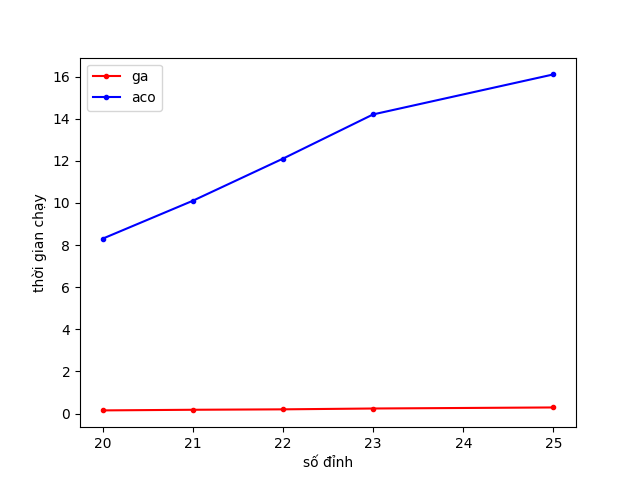
* Với dữ liệu bé hơn 15 đỉnh

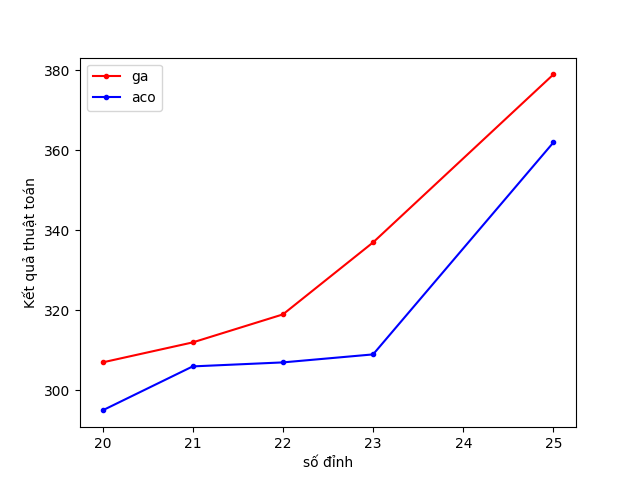




Với những dữ liệu bé hơn 15 đỉnh thì cả 2 thuật toán đều chạy khá đúng với kết quả tối ưu thật sự của bài toán. Thuật toán ACO luôn cho kết quả chính xác hơn thuật toán GA. Nhưng thời gian chạy của thuật toán ACO với những dữ liệu bé hơn 10 thì gần như bằng với thuật toán GA, nhưng đến với những dữ liệu có số đỉnh lớn hơn 12 thì thuật toán ACO bắt đầu chậm hơn thuật toán GA

* Với những dữ liệu lớn hơn 20





Với những dữ liệu lớn hơn 20 đỉnh thì thời gian chạy của thuật toán ACO khá lâu, và chậm hơn nhiều so với thuật toán GA. Tuy nhiên kết quả của 2 thuật toán đã có sự chênh lệch rõ rệt, và cũng không còn gần với kết quả tối ưu thật sự của bài toán nữa. Thật toán ACO, luôn cho kết quả tốt hơn thuật toán GA (độ dài đường đi tìm được ít hơn khoảng 15%).

1. **Tổng kết**

Báo cáo này tìm hiểu 2 giải thuật heuristics cho bài toán TSP là giải thuật di truyền và giải thuật tối ưu đàn kiến. Qua 1 số thực nghiệm trên bộ dữ liệu tôi sinh ngẫu nhiên thì thuật toán ACO cho kết quả chính xác hơn nhưng thời gian lại chạy chậm hơn giải thuật GA khá nhiều với những bộ dữ liệu lớn. Có thể là do việc chọn tham số chưa được hợp lý, tôi sẽ nghiên cứu thuật toán nhằm tối ưu giải thuật cùng với những thực nghiệm trên bộ dữ liệu lơn hơn sau này