ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

Họ và tên sinh viên: Lê Minh Cường
MSSV : 3117410027

ĐỀ XUẤT THUẬT TOÁN HILL CLIMBING GIẢI BÀI TOÁN NGƯỜI BÁN HÀNG

Giảng viên hướng dẫn: Phan Tấn Quốc

TP. Hồ Chí Minh, tháng 5 năm 2021

MỤC LỤC

XÁC NHẬN CỦA NGƯỜI HƯỚNG DẪN KHOA HỌC

……

……

……

……

……

……

……

Thành phố Hồ Chí Minh, ngày tháng năm

Người hướng dẫn khoa học

(Kí và ghi rõ họ tên)

TS. Phan Tấn Quốc

LỜI MỞ ĐẦU

Nghiên cứu thuật toán là một trong những vấn đề phổ biến trong lĩnh vực Công Nghệ Thông Tin. Chúng ta biết rằng có rất nhiều thuật toán hiện nay, việc tìm hiểu và học hỏi sẽ giúp chúng ta có kiến thức và đẩy mạnh tiềm lực bản thân. Với môn Seminar chuyên đề, chúng ta sẽ nghiên cứu một vấn đề có tên là Traveling Saleman Problems (hay còn gọi là Bài toán người bán hàng). Không chỉ đơn thuần là một bài toán mà nó còn được vấn dụng vào thực tế như là tìm đường đi ngắn nhất giữa các thành phố. Vấn đề đã được nghiên cứu và phát triển nhiều năm, nhưng việc tìm kết quả chính xác cho vấn đề cụ thể vẫn còn là một bí ẩn. Lượng thành phố cao sẽ dẫn tới kết quả chúng ta tìm sẽ không bao giờ có được một cách chính xác hoàn toàn.

Trong báo cáo này, chúng ta sẽ bàn thêm về Hill Climbing một trong những thuật toán nổi tiếng để giải quyết bài toán TSP, như chúng ta biết thì có những dạng thuật toán tương tự như là: Tabu, Thuật toán Bee, Thuật Toán Local Search, Thuật Toán Ant Colony, Thuật Toán GA… Những thuật toán này đều có cách giải quyết riêng của chúng. Nhưng chúng ta sẽ bắt đầu từ những thuật toán đơn giản trước và tăng dần hướng phát triển. Ở chương đầu chúng ta sẽ tìm hiểu TSP, các định nghĩa liên quan, công thức, ứng dụng, lịch sử và tầm quan trọng của bài toán TSP. Chương 2 chúng ta sẽ nghiên cứu thuật toán Hill Climbing và cách giải một ví dụ đơn giản với thuật toán Hill Climbing. Chương 3 sẽ chi tiết cách sử dụng, cách trình bày thuật toán giải thuật bao gồm có mã giả, code để bổ sung cụ thể từng bước làm trong thuật toán. Ở mỗi bước sẽ là mỗi thuật toán khác nhau để hình thành giải thuật Hill Climbing. Chương 4 sẽ sử dụng các bộ test ở TSP Lib một trong những trang nổi tiếng trên mạng hiện nay, tất cả các kết quả đều được kiểm chứng và đó là lời giải tốt nhất ở hiện tại. Và cuối cùng, Chương 5 chúng ta sẽ nêu kết luận, nêu lên hướng phát triển trong tương lai với TSP và Hill Climbing.

Do đây là lần đầu tiên nghiên cứu về vấn đề này, nên có những thiếu sót không mong muốn. Mong mọi người có thể thông cảm và bỏ qua.

CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN NGƯỜI BÁN HÀNG

1.1.Một số định nghĩa

1.1.1. Bài toán người bán hàng

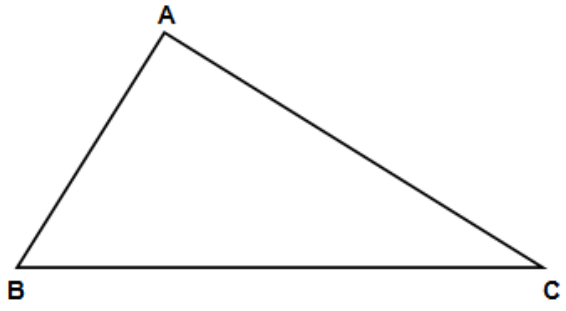
Bài toán người bán hàng (còn được biết tới với tên là nhân viên bán hàng hay viết tắt là TSP) có nội dung cụ thể như sau: Cho danh sách tất cả các thành phố và khoảng cách của mỗi thành phố, bài toán muốn tìm ra tuyến đường đi ngắn nhất có thể sao cho đi qua mỗi thành phố một lần và quay trở về thành phố ban đầu sao cho chi phí là nhỏ nhất. Đây là bài toán thuộc lớp NP-hard trong tối ưu hóa tổ hợp, quan trọng trong lý thuyết khoa học máy tính và hoạt động nghiên cứu.

1.1.2. Định nghĩa 2: Bài toán người bán hàng trong đồ thị

Bài toán người bán hàng có thể được mô hình hóa như là một đồ thị vô hướng có trọng số, vì vậy những thành phố sẽ trở thành đỉnh, những đường đi sẽ trở thành cạnh của đồ thị và khoảng cách của đường đi sẽ trở thành trọng số. Vấn đề trở thành việc bắt đầu và kết thúc tại 1 đỉnh được chỉ đỉnh sau khi đi qua tất cả các đỉnh chỉ một lần. Đây là mô hình đồ thị hoàn chỉnh. Nếu không tồn tại đường đi giữa 2 thành phố, việc thêm đủ cạnh vào để hoàn thành đồ thị sẽ không ảnh hưởng đến kết quả của bài toán.

1.1.3. Hệ đo lường

Trong hệ đo lường TSP, còn được biết tới như delta-TSP hay là $\Delta - TSP$, khoảng cách thỏa bất đẳng thức hình tam giác, có nghĩa là khoảng cách từ A đến B sẽ không bao giờ dài hơn khoảng cách thông qua C $d_{AB} \leq d_{AC} + d_{CB}$



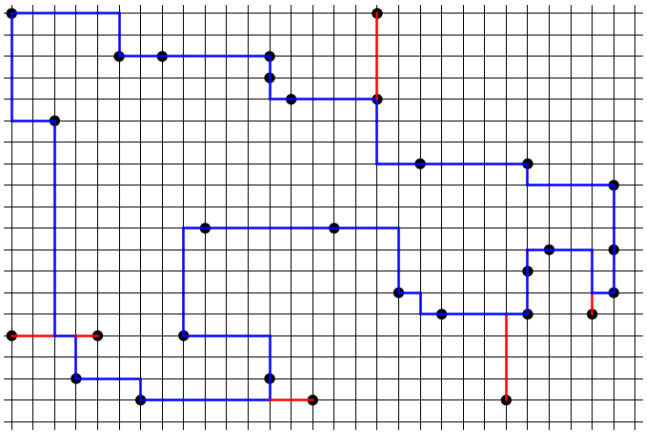
1.1 Hình ảnh thể hiện bất đẳng thức tam giác

Bài toán TSP trong các hệ đo lường khác nhau. Ví dụ:

- Trong hệ đo lường Euclidean khoảng cách từ 2 thành phố là khoảng cách Euclidean giữa 2 điểm tương ứng

Công thức Euclidean giữa 2 điểm bất kì là d = $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ với x_i, y_i là chiều của x và chiều y của 2 điểm.

- Trong rectilinear TSP thì khoảng cách giữa 2 thành phố là tổng trị tuyệt đối giá trị của sự khác nhau của tọa độ x và y



1.2 Hình ảnh rectilinear TSP

Công thức khoảng cách Mahattan giữa 2 điểm bất kì là d = $|x_1 - y_1| + |x_2 - y_2|$ với x_i, y_i là chiều của x và chiều y của 2 điểm.

1.1.4. Euclidean

Đầu vào có thể là những con số là số thực tùy ý, Euclidean TSP là một trường hợp điển hình của bài toán trong hệ đo lường TSP, bởi vì khoảng cách trong không gian xy tuân theo bất đẳng thức tam giác.

1.1.5. Định nghĩa 3: Đối xứng và bất đối xứng

Trong bài toán người bán hàng với đồ thị đối xứng thì khoảng cách giữa 2 thành phố là như nhau với mỗi chiều ngược lại, ví dụ như: đồ thị vô hướng. Bài toán với đồ thị đối xứng thì có thể có số lượng giải pháp có thể. Trong khi đồ thị bất đối xứng, đường đi có thể không tồn tại cả 2 chiều và khoảng cách cũng có thể khác nhau, ví dụ: đồ thị có hướng. Hầu hết tất cả khoảng cách giữa hai thành phố trong mạng lưới TSP thì như nhau ở cả 2 chiều tức là A đến B bằng khoảng cách từ B đến A. Trường hợp khi mà khoảng cách từ A đến B không bằng khoảng cách từ B đến A đó được gọi là không đối xứng.

1.1.6. Độ phức tạp của thuật toán

Bài toán có độ phức tạp là NP-hard không thể giải bài toán trực tiếp với dữ liệu đầu vào lớn một cách chính xác, mà chỉ có thể giải bài toán gần đúng. Vấn đề sẽ luôn luôn

là NP-hard trong trường hợp khi mà những thành phố nằm trong trục tọa độ xy với hệ khoảng cách tính bằng Eclidean.

1.1.7. Các bài toán liên quan

Một công thức tương đương về lý thuyết đồ thị là: Cho một đồ thị có trọng số hoàn chỉnh (trong đó các đỉnh sẽ đại diện cho các thành phố, các cạnh sẽ đại diện cho các con đường và trọng số sẽ là chi phí hoặc khoảng cách của con đường đó), hãy tìm một chu trình Hamilton với trọng lượng ít nhất.

Yêu cầu quay trở lại thành phố xuất phát không làm thay đổi độ phức tạp tính toán của bài toán, xem bài toán đường đi Hamilton.

Một bài toán khác có liên quan là bài toán nhân viên bán hàng đi du lịch nút cổ chai (TSP nút cổ chai): Tìm một chu trình Hamilton trong đồ thị có trọng số với trọng lượng nhỏ nhất của cạnh cân nhất. Ví dụ, tránh những con phố hẹp có xe buýt lớn. Vấn đề có tầm quan trọng thực tế đáng kể, ngoài các khu vực vận tải và hậu cần rõ ràng. Một ví dụ cổ điển là trong sản xuất mạch in: lập lịch trình của máy khoan để khoan lỗ trên PCB. Trong các ứng dụng khoan hoặc gia công rô bốt, "thành phố" là các bộ phận của máy hoặc lỗ (có kích thước khác nhau) để khoan và "chi phí đi lại" bao gồm thời gian trang bị lại rô bốt (vấn đề trình tự công việc của một máy).

Bài toán tổng quát về nhân viên bán hàng lưu động, còn được gọi là "bài toán về chính trị gia du lịch", đề cập đến các "tiểu bang" có (một hoặc nhiều) "thành phố" và nhân viên bán hàng phải đến thăm chính xác một "thành phố" từ mỗi "tiểu bang". Một ứng dụng được gặp phải trong việc đặt hàng một giải pháp cho vấn đề kho cắt để giảm thiểu sự thay đổi dao. Một người khác liên quan đến việc khoan trong sản xuất chất bán dẫn, xem ví dụ: Bằng sáng chế Hoa Kỳ 7.054.798. Noon and Bean đã chứng minh rằng bài toán nhân viên bán hàng đi du lịch tổng quát có thể được chuyển thành bài toán nhân viên bán hàng đi du lịch tiêu chuẩn với cùng một số thành phố, nhưng một ma trận khoảng cách đã được sửa đổi.

Bài toán sắp xếp thứ tự giải quyết vấn đề thăm một tập hợp các thành phố nơi tồn tại các mối quan hệ ưu tiên giữa các thành phố.

Một câu hỏi phỏng vấn phổ biến tại Google là làm thế nào để định tuyến dữ liệu giữa các nút xử lý dữ liệu; các tuyến đường khác nhau theo thời gian để truyền dữ liệu,

nhưng các nút cũng khác nhau theo khả năng tính toán và khả năng lưu trữ của chúng, làm phức tạp vấn đề về nơi gửi dữ liệu.

Vấn đề người mua đi du lịch giải quyết với người mua được tính phí mua một bộ sản phẩm. Anh ta có thể mua những sản phẩm này ở một số thành phố, nhưng với các mức giá khác nhau và không phải tất cả các thành phố đều cung cấp các sản phẩm giống nhau. Mục tiêu là tìm ra một tuyến đường giữa một tập hợp con các thành phố, điều này giúp giảm thiểu tổng chi phí (chi phí đi lại + chi phí mua hàng) và cho phép mua tất cả các sản phẩm cần thiết.

1.1.8. Các trường hợp đặc biệt

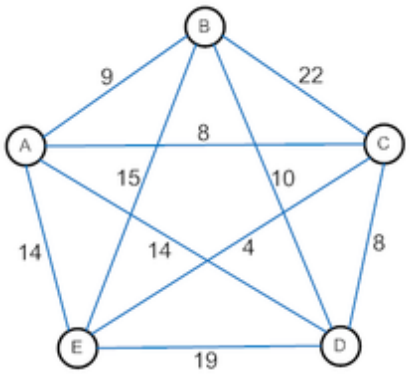
1.1.8.1. Hệ đo lường và tổng quan:

Trong phiên bản đo lường của TSP, công thức khoảng cách d là một hệ đo lường. Nó thỏa với bất đẳng thức tam giác tức là nếu chúng ta có 3 điểm A, B và C muốn đi từ A qua C chúng ta có thể đi một đường khác là A sang B rồi từ B sang C. Như vậy thì sẽ luôn luôn thỏa bất đẳng thức d(AC) <= d(AB) + d(BC). Trong những phiên bản khác, d có thể được gán bất kì giá trị nào của một cạnh, nghĩa là từ A qua B có thể không cần từ B qua A không tuân theo bất đẳng thức tam giác.

1.1.8.2. Đi qua lặp lại và không lặp lại

Mục tiêu của TSP là tìm một chu trình đi qua tất cả các cạnh chỉ một lần. Vậy thì có chu trình nào chứa lặp lại đỉnh mà đã đi qua được hay không? Trong phiên bản TSP không lặp lại thì, chu trình phải đi qua mỗi thành phố chỉ một lần duy nhất. Nhưng trong phiên bản lặp lại thì chúng ta có thể chấp nhận được việc lặp lại việc đi qua lại thành phố thêm một lần nữa, nếu kết quả là tốt hơn.

1.1.9. Ví dụ về một trường hợp với đồ thị đối xứng



1.3 Hình ảnh bài toán TSP

Hình 1. Minh họa đồ thị và đường đi bắt đầu từ đỉnh C {C, E, A, B, D, C}

Cho đồ thị vô hướng liên thông có 9 đỉnh và 26 cạnh như Hình 1; Lấy một đường đi bất kì tổng chi phí của đoạn đường bắt đầu từ C gồm CEABDC là 4 + 14 + 9 + 10 + 8 = 45. Vậy kết quả cuối cùng đi từ C là 45 nếu chúng ta ưu tiên đi những đường ngắn từ đỉnh hiện tại tới đỉnh tiếp theo.

1.2.Ứng dụng của bài toán người bán hàng

Ngoài việc là một "polytope" của một vấn đề tối ưu hóa tổ hợp khó khăn từ một phức tạp điểm lý thuyết của xem, có những trường hợp quan trọng của các vấn đề thực tế có thể được xây dựng như các vấn đề TSP và nhiều vấn đề khác là những khái quát của vấn đề này.

Bên cạnh việc khoan mạch in bảng mô tả ở trên, vấn đề có cấu trúc TSP xảy ra trong phân tích cấu trúc của các tinh thể, (Bland và Shallcross, 1987), các đại tu động cơ tuốc bin khí (Pante, Lowe và Chandrasekaran, 1987), trong xử lý vật liệu trong một nhà kho (Ratliff và Rosenthal, 1981) , trong việc cắt giảm các vấn đề chứng khoán, (Garfinkel, 1977), các phân nhóm của các mảng dữ liệu, (Lenstra và Rinooy Kạn, 1975), trình tự các công việc trên một máy tính duy nhất (và Gilmore Gomory, 1964) và phân công các tuyến đường cho máy bay của một hạm đội quy định (Boland, Jones, và Nemhauser, 1994).

Biến thể có liên quan về vấn đề nhân viên bán hàng đi du lịch bao gồm các nguồn tài nguyên hạn chế đi du lịch vấn đề nhân viên bán hàng trong đó có các ứng dụng trong lập kế hoạch với thời hạn tổng hợp (Pekny và Miller, 1990). Nghiên cứu này cũng cho thấy giải thưởng thu thập đi vấn đề nhân viên bán hàng (Balas, 1989) và các vấn đề Orienteering (Golden, Levy và Vohra, 1987) là trường hợp đặc biệt của tài nguyên hạn chế TSP.

Quan trọng nhất là vấn đề nhân viên bán hàng đi du lịch thường thể hiện như một bài toán con trong nhiều vấn đề tổ hợp phức tạp, là nổi tiếng và quan trọng nhất trong số đó là vấn đề định tuyến xe, có nghĩa là, vấn đề xác định cho một đội xe mà khách hàng sẽ được phục vụ bởi mỗi chiếc xe và theo thứ tự mỗi chiếc xe nên đến các khách hàng được giao. Đối với các cuộc điều tra có liên quan, xem Christofides (1985) và Fisher (1987).

Thuật toán tối ưu hóa đã được áp dụng cho nhiều vấn đề tối ưu hóa tổ hợp khác nhau, từ phân bậc hai với Protein gấp hoặc định tuyến xe (Vehicle routing problem) và rất nhiều phương pháp có nguồn gốc đã được thích nghi với các vấn đề năng động trong thực tế các biến ngẫu nhiên, các vấn đề ngẫu nhiên, đa mục tiêu và triển khai song song. Nó cũng đã được sử dụng để sản xuất các giải pháp gần tối ưu cho vấn đề nhân viên bán hàng đi du lịch. Họ có lợi thế hơn mô phỏng luyện kim (Simulated annealing) và thuật toán di truyền (Genetic algorithm) của phương pháp tiếp cận vấn đề tương tự khi đồ thị có thể thay đổi tự động; thuật toán đàn kiến có thể chạy liên tục và thích ứng với những thay đổi trong thời gian thực. Đây là quan tâm trong mạng định tuyến (Network routing) và các hệ thống giao thông đô thị.

Các thuật toán ACO đầu tiên được gọi là hệ thống Ant [17] và nó nhằm mục đích để giải quyết vấn đề nhân viên bán hàng đi du lịch, trong đó mục đích là để tìm ngắn chuyến đi vòng quanh để liên kết một loạt các thành phố. Các thuật toán chung là tương đối đơn giản và dựa trên một tập hợp các kiến, mỗi người làm của vòng các chuyến đi có thể cùng các thành phố. Ở mỗi giai đoạn, các kiến lựa chọn để di chuyển từ một thành phố khác theo một số quy tắc:

1. Nó phải đến mỗi thành phố đúng một lần.

2. Một thành phố xa xôi có ít cơ hội được lựa chọn (khả năng hiển thị.

3. Cường độ cao hơn đường mòn pheromone đặt ra trên một cạnh giữa hai thành phố, lớn hơn xác suất mà cạnh đó sẽ được chọn.

4. Đã hoàn thành cuộc hành trình của nó, là kiến gia gửi pheromone hơn trên tất cả các cạnh nó đi qua, nếu cuộc hành trình ngắn.

5. Sau mỗi lần lặp, những con đường mòn các kích thích tố bay hơi.

1.3.Lịch sử nghiên cứu vấn đề/ Tổng quan

Nguồn gốc của bài toán người bán hàng vẫn chưa được biết rõ. Một cuốn sổ tay dành cho người bán hàng xuất bản năm 1832 có đề cập đến bài toán này và có ví dụ cho chu trình trong nước Đức và Thụy Sĩ, nhưng không chứa bất kì nội dung toán học nào.

Bài toán người bán hàng được định nghĩa trong thế kỉ 19 bởi nhà toán học Ireland William Rowan Hamilton và nhà toán học Anh Thomas Kirkman. Trò chơi

Icosa của Hamilton là một trò chơi giải trí dựa trên việc tìm kiếm chu trình Hamilton. Trường hợp tổng quát của TSP có thể được nghiên cứu lần đầu tiên bởi các nhà toán học ở Vienna và Harvard trong những năm 1930, đặc biệt là Karl Menger, người đã định nghĩa bài toán, xem xét thuật toán hiển nhiên nhất cho bài toán, và phát hiện ra thuật toán láng giềng gần nhất là không tối ưu.

Hassler Whitney ở đại học Princeton đưa ra tên bài toán người bán hàng ngay sau đó. Trong những năm 1950 và 1960, bài toán trở nên phổ biến trong giới nghiên cứu khoa học ở châu Âu và Mỹ. George Dantzig, Delbert Ray Fulkerson và Selmer M. Johnson ở công ty RAND tại Santa Monica đã có đóng góp quan trọng cho bài toán này, biểu diễn bài toán dưới dạng quy hoạch nguyên và đưa ra phương pháp mặt phẳng cắt để tìm ra lời giải. Với phương pháp mới này, họ đã giải được tối ưu một trường hợp có 49 thành phố bằng cách xây dựng một chu trình và chứng minh rằng không có chu trình nào ngắn hơn. Trong những thập niên tiếp theo, bài toán được nghiên cứu bởi nhiều nhà nghiên cứu trong các lĩnh vực toán học, khoa học máy tính, hóa học, vật lý, và các ngành khác.

Năm 1959, Jillian Beardwood, J.H. Halton và John Hammersley đã xuất bản một bài báo có tựa đề "Con đường ngắn nhất qua nhiều điểm" trên tạp chí của Hiệp hội Triết học Cambridge. Định lý Beardwood – Halton – Hammersley cung cấp một giải pháp thực tế cho vấn đề người bán hàng lưu động. Các tác giả đã suy ra một công thức tiệm cận để xác định độ dài của con đường ngắn nhất cho một nhân viên bán hàng bắt đầu tại nhà hoặc văn phòng và ghé thăm một số địa điểm cố định trước khi quay lại điểm bắt đầu. Trong những thập kỷ tiếp theo, vấn đề đã được nghiên cứu bởi nhiều nhà nghiên cứu từ toán học, khoa học máy tính, hóa học, vật lý và các ngành khoa học khác. Tuy nhiên, vào những năm 1960, một cách tiếp cận mới đã được tạo ra, rằng thay vì tìm kiếm các giải pháp tối ưu sẽ tạo ra một giải pháp mà độ dài của nó bị giới hạn bởi bội số của độ dài tối ưu, và làm như vậy sẽ tạo ra các giới hạn thấp hơn cho vấn đề; các giới hạn thấp hơn này sau đó sẽ được sử dụng với các phương pháp tiếp cận nhánh và ràng buộc. Một phương pháp thực hiện điều này là tạo một cây bao trùm tối thiểu của đồ thị và sau đó nhân đôi tất cả các cạnh của nó, điều này tạo ra ràng buộc rằng độ dài của chuyến tham quan tối ưu nhiều nhất là gấp đôi trọng lượng của cây bao trùm tối thiểu.

Năm 1976, Christofides và Serdyukov độc lập với nhau đã tạo ra một bước tiến lớn theo hướng này: thuật toán Christofides-Serdyukov đưa ra một giải pháp mà trong trường hợp xấu nhất, dài hơn tối đa 1,5 lần so với giải pháp tối ưu. Vì thuật toán rất đơn giản và nhanh chóng, nhiều người hy vọng nó sẽ nhường chỗ cho một phương pháp giải gần như tối ưu. Đây vẫn là phương pháp phù hợp với trường hợp xấu nhất. Tuy nhiên, đối với một trường hợp đặc biệt khá chung chung của vấn đề, nó đã bị đánh bại bởi một biên độ nhỏ vào năm 2011.

Richard M. Karp đã chỉ ra vào năm 1972 rằng bài toán chu trình Hamilton là NP-đầy đủ, có nghĩa là độ cứng NP của TSP. Điều này cung cấp một giải thích toán học cho khó khăn tính toán rõ ràng trong việc tìm kiếm các chuyến tham quan tối ưu.

Tiến bộ vượt bậc đã đạt được vào cuối những năm 1970 và 1980, khi Grötschel, Padberg, Rinaldi và những người khác tìm cách giải quyết chính xác các trường hợp có tới 2.392 thành phố, sử dụng máy bay cắt, phân nhánh và ràng buộc.

Trong những năm 1990, Applegate, Bixby, Chvátal và Cook đã phát triển chương trình Concorde đã được sử dụng trong nhiều giải pháp thu âm gần đây. Gerhard Reinelt đã xuất bản TSPLIB vào năm 1991, một tập hợp các trường hợp chuẩn có độ khó khác nhau, đã được nhiều nhóm nghiên cứu sử dụng để so sánh kết quả. Vào năm 2006, Cook và những người khác đã tính toán một chuyến tham quan tối ưu qua một phiên bản 85.900 thành phố được đưa ra bởi một bài toán bố cục vi mạch, hiện là phiên bản TSPLIB đã được giải quyết lớn nhất. Đối với nhiều trường hợp khác với hàng triệu thành phố, có thể tìm thấy các giải pháp được đảm bảo nằm trong khoảng 2-3% của một chuyến tham quan tối ưu.

Vào năm 2020, một thuật toán xấp xỉ được cải tiến một chút đã được phát triển. Năm 1972, Richard M. Karp chứng minh rằng bài toán chu trình Hamilton là NP-đầy đủ, kéo theo bài toán TSP cũng là NP-đầy đủ. Đây là một lý giải toán học cho sự khó khăn trong việc tìm kiếm chu trình ngắn nhất.

Một bước tiến lớn được thực hiện cuối thập niên 1970 và 1980 khi Grötschel, Padberg, Rinaldi và cộng sự đã giải được những trường hợp lên tới 2392 thành phố, sử dụng phương pháp mặt phẳng cắt và nhánh cận.

Trong những năm 1990, Applegate, Bixby, Chvátal và Cook đã phát triển chương trình Concorde đã được sử dụng trong nhiều giải pháp thu âm gần đây. Gerhard Reinelt đã

xuất bản TSPLIB vào năm 1991, một tập hợp các trường hợp chuẩn có độ khó khác nhau, đã được nhiều nhóm nghiên cứu sử dụng để so sánh kết quả. Vào năm 2006, Cook và những người khác đã tính toán một chuyến tham quan tối ưu qua 85,900 thành phố được đưa ra bởi một bài toán bố cục vi mạch, hiện là phiên bản TSPLIB đã được giải quyết lớn nhất. Đối với nhiều trường hợp khác với hàng triệu thành phố, có thể tìm thấy các giải pháp được đảm bảo nằm trong khoảng 2-3% của một chuyến tham quan tối ưu.

Vào năm 2020, một thuật toán xấp xỉ được cải tiến một chút đã được phát triển.

1.4.Kết luận

Bài toán người bán hàng là một trong những vấn đề hay kể cả trong toán học và các lĩnh vực nghiên cứu khoa học, không chỉ góp phần vào việc xây dựng các đường đi ngắn nhất giữa các thành phố, làm tối ưu hóa cách giải nghĩa là cải thiện được tuyến đường đi trên thực tế. Với độ phức tạp là NP hard như vậy sẽ cần có những cách giải và lời giải hợp lý nếu không bài toán sẽ không thể nào giải ra được kết quả vì có rất nhiều trường hợp.

CHƯƠNG 2 NGHIÊN CỨU VỀ THUẬT TOÁN HILL CLIMBING

2.1.Định nghĩa chung về Hill Climbing

Hill Climbing là một kĩ thuật tối ưu hóa bài toán, thuộc cùng nhóm với bài toán local search. Hill Climbing là một heuristic search được dùng cho những vấn đề tối ưu hóa toán học trong lĩnh vực Trí Tuệ Nhân Tạo. Nó là thuật toán lặp (iterative algorithm), tức là là nó sẽ bắt đầu tại một lời giải bất kì tức nhiên lời giải đó không phải là một lời giải tối ưu (global maximums) và cho đó là một lời giải hiện tại của bài toán, sau đó nó sẽ cố gắng tìm một lời giải tốt hơn lời bằng việc tạo ra sự thay đổi trong lời giải hiện tại. Nếu nó tạo ra một lời giải tốt hơn, thì lời giải mới này sẽ được cho là lời giải hiện tại và sẽ lặp lại cho đến khi mà kết quả lời giải không thể tốt hơn được nữa.

- Trong định nghĩa trên, vấn đề tối ưu hóa toán học được tiến hành là hill climbing giải quyết vấn đề mà chỗ đó cần tối đa hóa hay giảm thiểu hóa một hàm cho trước bằng việc chọn một giá trị từ việc nhập giá trị đó vào hàm cho kết quả thỏa yêu cầu hay nhu cầu một phần của bài toán. Ví dụ ở đây là chúng ta muốn giảm thiểu hóa khoảng cách di chuyển của người bán hàng.
- Heuristic search có nghĩa là một thuật toán tìm kiếm không phải tìm kiếm tối ưu cách giải quyết vấn đề. Mà là nó sẽ tìm kiếm một cách giải quyết tốt trong một khoảng thời gian hợp lý.
- Hàm heuristic là hàm mà sẽ liệt kê và xếp hạng tất cả những phương án thay thế cho phương án hiện tại ở một bước, hay giai đoạn nào đó trong lúc tìm kiếm dựa trên những thông tin có sẵn, nó sẽ giúp thuật toán chọn ra được phương án tốt nhất trong những phương án thay thế đó. Giả dụ như tại bước A có thể đi qua B và C, thì thuật toán sẽ đánh giá là nên đi qua từ A qua B hay từ A qua C là tốt hơn.

2.2.Mô tả toán học

Hill climbing cố gắng tối đa hóa (giảm thiểu hóa) hàm mục tiêu $f(x)$ trong đó x là vector của biến liên tục hoặc rời rạc. Tại mỗi lần lặp, hill climbing sẽ chỉnh từng thuộc tính trong x và xác định bất cứ thay đổi nào cải thiện được giá trị của $f(x)$. Với hill

climbing thì bất cứ sự thay đổi nào mà cải thiện được $f(x)$ thì đều được chấp nhận, quá trình này sẽ được lặp cho tới khi mà không có sự thay đổi nào được tìm thấy để cải thiện được giá trị của $f(x)$. Khi đó thì x được gọi là locally optimal. Nếu hiểu theo nghĩa của bài toán TSP thì x là đường đi có thể đi và $f(x)$ là tổng giá trị đường đi qua các đỉnh và về thành phố ban đầu. Và mục tiêu là cải thiện $f(x)$ về giá trị nhỏ nhất

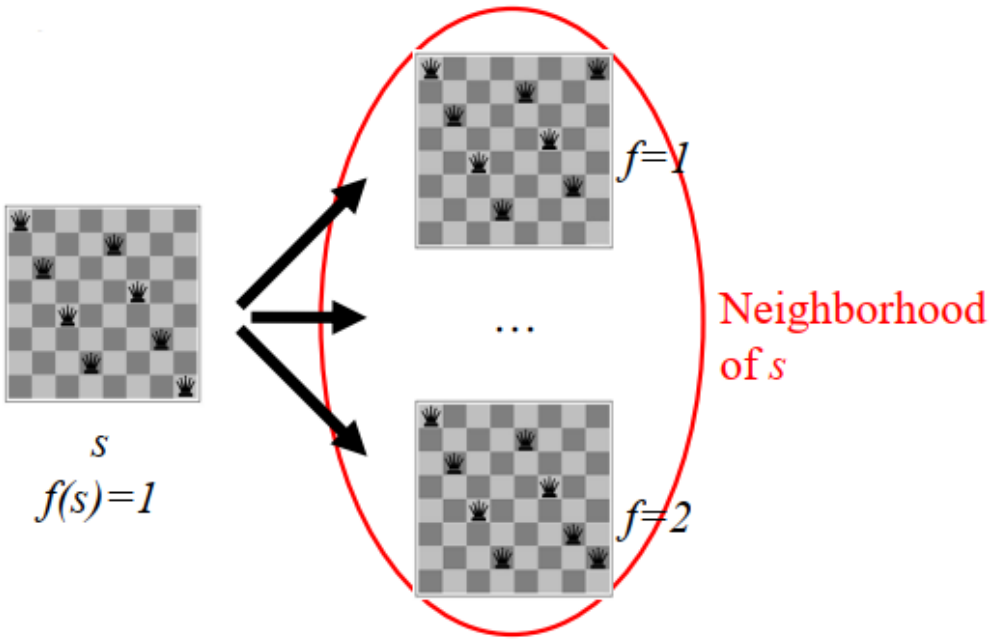
2.3.Đặc điểm của bài toán Hill Climbing

- Nó tiến hành thuật toán tham lam: Có nghĩa là nó sẽ di chuyển về một hướng mà ở đó giá trị của hàm giá trị được tối ưu. Tiếp cận bằng giải thuật tham lam cho phép thuật toán tạo nên kết quả local maximum hoặc local minimum.
- Không đệ quy: Thuật Hill Climbing chỉ làm việc với giải pháp hiện tại (current solution) và giải phép có thể (impossible solution được tạo ra từ việc kế thừa kết quả của current solution). Nó không lấy lại kết quả của giải phá trước đó nên không cần quay lui để tìm lại.
- Feedback mechanism: Thuật toán có phản hồi kết kỹ thuật tức là nó giúp quyết định được hướng di chuyển (Khi nào thì nên leo lên và khi nào thì xuống núi). Những phản hồi này giúp được nâng cao thông qua việc tạo và thử nghiệm các chiến lược.
- Tăng dần sự thay đổi: Thuật toán giúp trao dồi giải pháp hiện tại bằng việc tang dần sự thay đổi. Có nghĩa là khi ở một trạng thái hiện tại, chúng ta làm nhiều bước để tìm ra tất cả những giải pháp có thể từ giải pháp hiện tại, sau đó chúng ta sẽ chọn giải pháp tốt nhất trong những giải pháp vừa mới sinh ra đó để so sánh với giải pháp hiện tại, nếu như tốt hơn thì sẽ thay thế giải pháp hiện tại. Cứ như vậy sẽ làm thay đổi nhiều bộ giải pháp theo giải pháp hiện tại.

2.4.Thế nào gọi là neighbor trong Hill Climbing

N-Queen:

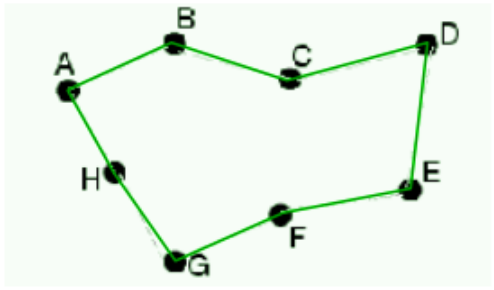
Nếu trong bài toán N hậu thì lời giải lân cận sẽ là các nước đi của con hậu trong bàn cờ, mỗi lẫn di chuyển con hậu sẽ sinh ra một neighbor mới.



TSP:

Còn với TSP thì việc chúng ta hoán vị 2 thành phố thì cũng tạo ra một lời giải quyết neighbor. Và trong báo này chúng ta trình bài 3 chiến thuật để sinh ra lời giải hàng xóm.

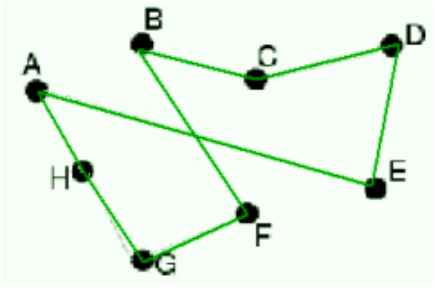
- state: A-B-C-D-E-F-G-H-A
- *f* = length of tour
- One possibility: 2-change



A-B-C-D-E-F-G-H-A

flip

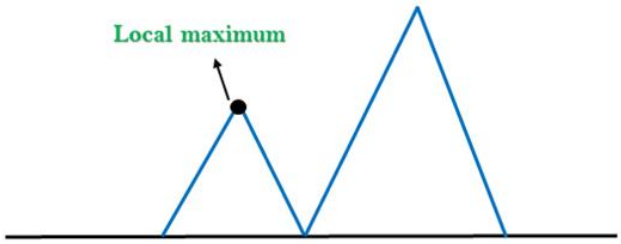
A-E-D-C-B-F-G-H-A



2.5.Đặc điểm các vùng trong Hill Climbing

2.5.1. Local maximum

Là trạng thái tốt hơn tất cả những trạng thái lân cận. Tuy nhiên, vẫn tồn tại một trạng thái tốt nhất đó gọi là global maximum. Local maximum là tốt hơn bởi vì ở đây giá trị của hàm mục tiêu cao hơn những giá trị hàng xóm của nó.

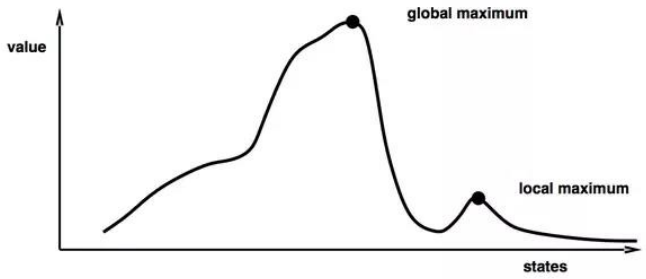


2.1 Hình ảnh local maximum

Như hình có thể thấy rằng xung quanh trạng thái local maximum thì đó chính là trạng thái tốt nhất, nếu như mà thuật toán hill climbing không tốt thì khi tìm tất cả những trạng thái xung quanh local maximum thì chính là kết quả cuối cùng của bài toán, trong khi kết quả thực chất có thể sẽ tìm thấy đâu đó.

2.5.2. Global maximum

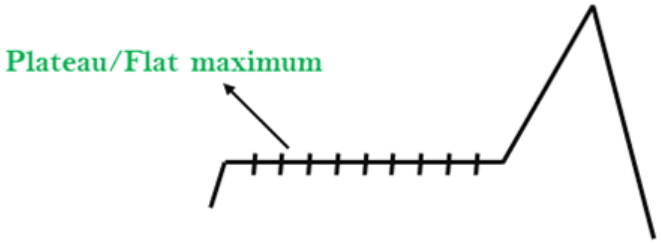
Đây là trạng thái tốt nhất hay còn gọi là kết quả của bài toán. Ở trạng thái này thì đường đi, đi qua các thành phố là tốt nhất. Tức nhiên sẽ có nhiều trạng thái Global maximum tức nhiều đường đi.



2.2 Hình ảnh Global maximum

2.5.3. Plateua/ flat local maximum

Là vùng mà của không gian trạng thái, mà ở đây tất cả các giá trị hàng xóm đề có kết quả là như nhau.

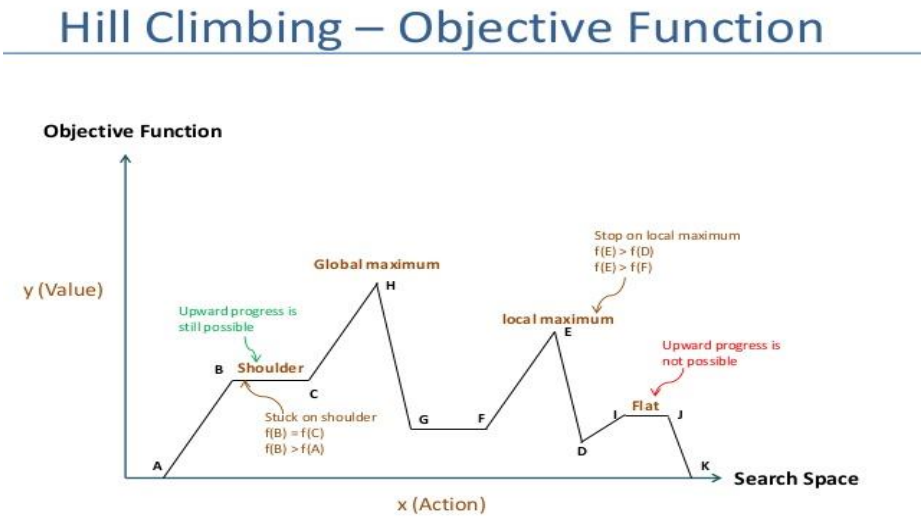


2.3 Hình ảnh Plateau/flat

2.5.4. Shoulder

Nó là một vùng cao nguyên có cạnh dốc nếu tiếp tục tìm trong vùng này vẫn có thể leo lên được dốc cao hơn để tìm ra được kết quả global maximum. Ở vùng này gần như đã gần chạm được tới được mục tiêu nếu như tiếp tục leo lên cao hơn.

Tổng kết lại chúng ta có được hình ảnh như dưới đây



2.4 Hình ảnh shoulder

2.6.Các loại Hill Climbing

2.6.1. Simple Hill Climbing

Nó sẽ tính toán từng lời giải lận cận theo thứ tự và chọn lời giải đầu tiên cái mà tối ưu lời giải hiện tại và lời giải tối ưu này sẽ được gán làm lời giải hiện tại. Và cứ thế tiếp tục lời giải tiếp theo thứ tự.

Giải thuật:

Bước 1: Tính toán lời giải ban đầu. Nếu là đáp án thì dừng và thông báo thành công. Ngược lại, lấy giá trị đó làm lời giải hiện tại

Bước 2: Lặp cho tới khi không tìm thấy bất cứ lời giải lân cận nào có thể thay thế cho lời giải hiện tại.

- Chọn lời giải chưa từng được dùng làm lời giải hiện tại và sử dụng nó để tạo lời giải mới.
- Tiến hành xử lý lời giải mới:
    - Nếu giá trị của lời giải hiện tại là kết quả thì dừng và thông báo thành công.
    - Nếu giá trị của lời giải mới tốt hơn lời giải hiện tại, thì thay thế lời giải hiện tại thành lời giải mới.
    - Nếu không có lời giải tốt hơn lời giải hiện thì sẽ tiếp tục lặp cho tới giải quyết được bài toán

Bước 3: Kết thúc

2.6.2. Steepest Ascent Hill Climbing

Nó sẽ tính toán hết tất cả lời giải lân cận và chọn lời giải tốt nhất trong tất cả lời giải đó. Sau đó lời giải tốt đó sẽ được so sánh với lời giải hiện tại. Nếu tốt hơn sẽ được thay thế cho lời giải hiện tại. Cứ thế lặp lại liên tục cho đến khi trúng điều kiện dừng.

Giải thuật

Bước 1: Tính toán lời giải ban đầu. Nếu là đáp án thì dừng và thông báo thành công. Ngược lại, lấy giá trị đó làm lời giải hiện tại

Bước 2: Lặp cho tới khi không tìm thấy bất cứ lời giải lân cận nào có thể thay thế cho lời giải hiện tại.

- Chọn lời giải chưa từng được dùng làm lời giải hiện tại và sử dụng nó để tạo lời giải mới.
- Khởi tạo lời giải tốt nhất bằng lời giải hiện tại và sử dụng nó để tạo ra những lời giải mới. Như vậy chúng ta sẽ ra được danh sách các lời giải lân cận
- Trong những lời giải mới đó chọn ra 1 lời giải tốt nhất.
- Tiến hành xử lý lời giải tốt:
    - Nếu giá trị của lời giải hiện tại là kết quả thì dừng và thông báo thành công.

- Nếu giá trị của lời giải tốt cải thiện tốt hơn lời giải hiện tại, thì thay thế lời giải hiện tại thành lời giải mới.
- Nếu không có lời giải tốt hơn lời giải hiện thì sẽ tiếp tục lặp cho tới giải quyết được bài toán

Bước 3: Kết thúc

2.6.3. Stochastic Hill Climbing

Nó sẽ không phải tính toán tất cả những lời giải lân cận. Thay vào đó nó sẽ tạo ngẫu nhiên một lời giải ngẫu nhiên và quyết định chọn lời giải đó làm lời giải hiện tại hay là nên tạo một lời giải mới. (Việc chọn dựa vào cách mà lời giải đó thỏa yêu cầu từng bài toán, ví dụ như TSP thì khoảng cách sẽ giảm dần, trong khi với bài toán).

Giải thuật:

Bước 1: Tính toán lời giải ban đầu. Nếu là đáp án thì dừng và thông báo thành công. Ngược lại, lấy giá trị đó làm lời giải hiện tại

Bước 2: Lặp cho tới khi không tìm thấy bất cứ lời giải lân cận nào có thể thay thế cho lời giải hiện tại.

- Chọn lời giải chưa từng làm lời giải hiện tại.
- Áp dụng hàm kế thừa cho lời giải hiện tại để tạo lời giải mới.
- Tất cả các lời giải được tạo ra từ Bước trên sẽ so sánh với lời giải hiện tại và sẽ được chọn làm lời giải hiện tại nếu nó cải thiện được lời giải hiện tại.
- Nếu giá trị của lời giải hiện tại là kết quả thì dừng và thông báo thành công.

Bước 3: Thoát.

2.7.So sánh thuật toán Hill Climbing với những thuật toán khác

Như chúng ta đã biết chúng ta có rất nhiều thuật toán để giải bài toán TSP. Ở đây chúng ta sẽ điểm sơ một vài thuật toán đó.

Chúng ta có:

- Dynamic Programming Algorithm
- Simulated Annealing Algorithm
- Genetic Algorithm
- Partial Swarm Optimization
- Ant Colony Algorithm

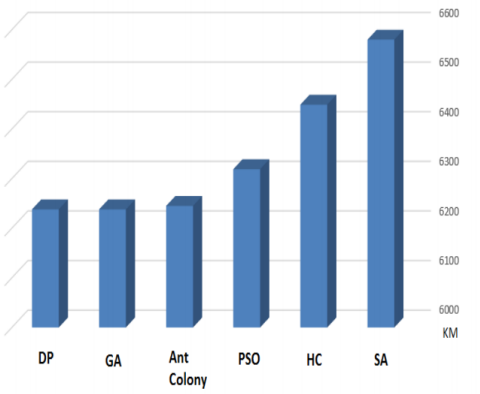


Figure 8. Comparing length of paths obtained from different algorithms in the dataset including 10 cities

Hình ảnh đánh giá độ dài mà thuật toán tìm được trên bộ test 10 thành phố

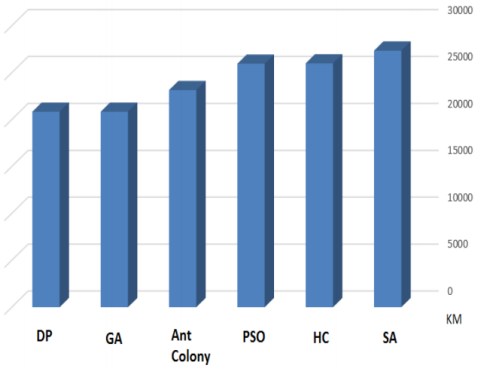


Figure 9. Comparing length of paths obtained from different algorithms in Iran59 dataset

Hình ảnh đánh giá độ dài mà thuật toán tìm được trên bộ test 59 thành phố

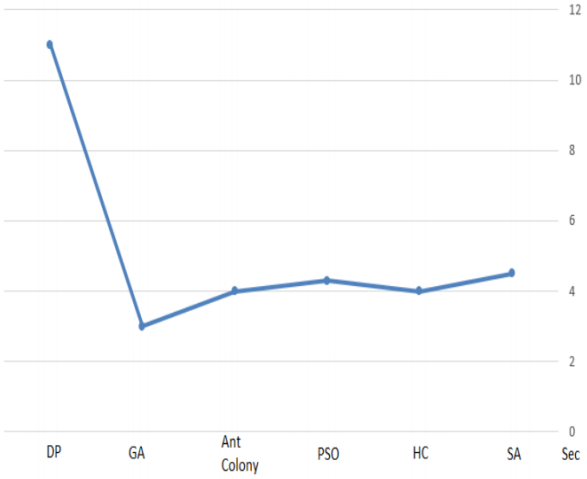


Figure 10. Comparing computation time in the dataset including 10 cities.

Hình ảnh đánh giá thời gian mà thuật toán tìm được trên bộ test 10 thành phố

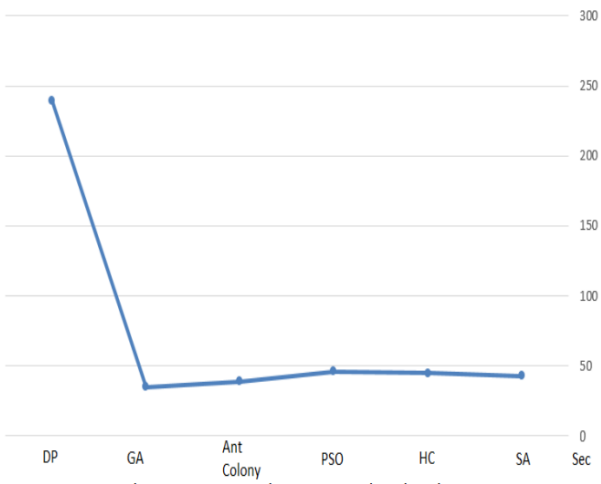


Figure 11. Comparing computation time in Iran59

Hình ảnh đánh giá thời gian mà thuật toán tìm được trên bộ test 59 thành phố

Như chúng ta đã thấy ***Dynamic Programming*** và ***Genetic Algorithm*** mang lại kết quả hiệu quả nhất trong tất cả các thuật toán về tuyến đường mà nó trả về. Còn với thời gian thì ***Genetic Algorithm*** mang lại thời gian nhỏ nhất và cao nhất là ***Dynamic Programming.*** Hill Climbing của chúng ta tuy có thời gian tương đối nhưng mà thuật toán lại không cho kết quả nhỏ hơn về giá trị so với những thuật toán khác. Nhưng đó không phải là vấn đề trong việc tìm hiểu thuật toán của chúng ta.

2.8. Kết luận chương 2

Như chúng ta đã đề cập phía trên Hill Climbing là một thuật toán metaheuristic, nó không giúp tối ưu hóa bài toán một cách tốt nhất, nhưng nó trả về kết quả làm thỏa mãn yêu cầu bài toán. Có nhiều dạng Hill Climbing nhưng việc áp dụng thuật toán như thế nào cho tốt, chiến lược thế nào cũng một phần góp tích cực cải thiện hiệu suất của bài toán.

Hill Climbing là một trong những thuật toán cùng họ với local search, có rất nhiều thuật toán tương tự nhưng nó là nền tảng bước đệm để chúng ta có thể nghiên cứu sâu hơn nhưng thuật toán khác.

CHƯƠNG 3: ĐỀ XUẤT THUẬT TOÁN HILL CLIMBING GIẢI BÀI TOÁN NGƯỜI BÁN HÀNG

3.1.Cơ sở lý thuyết

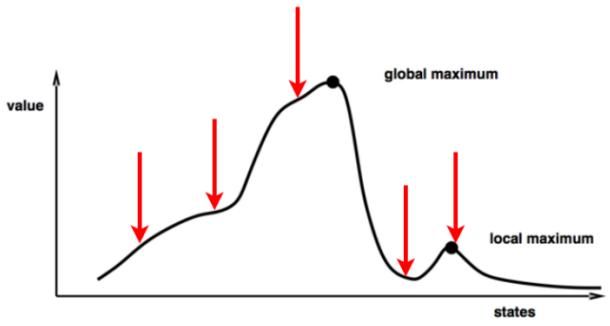
Dựa vào độ phức tạp của bài toán người bán hàng, nếu dữ liệu của thành phố qua nhiều thì bài toán nếu đưa về dạng tìm chính xác kết quả với những thuật toán Vét cạn, đệ quy, quy hoạch động,... Thực sự không khả thi vậy nên cần tìm một lời giải gần đúng ở đây, vậy nên việc áp dụng Hill Climbing vào bài toán là một cách giải quyết vấn đề lớn cho bài toán người bán hàng là hoàn toàn hợp lý. Đáp ứng với yêu cầu và nhu cầu của bài toán.

3.2.Giải bài toán người bán hàng

3.2.1. Trình bài giải thuật

3.2.1.1. Khởi tạo lời giải ban đầu

Lời giải ban đầu là một yếu tố quan trọng của bài toán, việc tạo ra một lời giải nghĩa là chúng ta đang tạo nên vị trí đứng khác nhau trong bài toán TSP, làm tăng khả năng chúng ta tìm được local maximum hoặc có khi thậm chí là global maximum. Nếu chúng ta chạy liên tục hàm Hill Climbing như vậy thì lời giải cuối cùng của chúng ta nhận được sẽ khác nhau

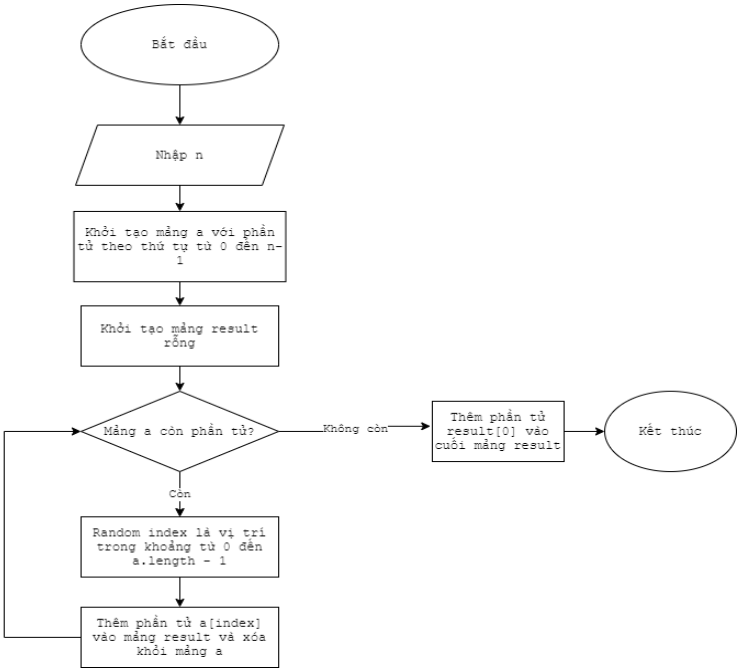


3.1 Hình ảnh mô tả các vị trí có thể ban đầu của initial node

Mũi tên đỏ là những vùng chúng ta có thể bước đến ngay ban đầu nhờ vào việc khởi tạo lời giải ban đầu, từ đó chúng ta dùng giải thuật Hill Climbing để tiếp cận những vùng lân cận và leo đến vị trí cao nhất của vùng đó. Có rất nhiều cách để tạo lời giải ban đầu như là dùng ngẫu nhiên hoặc là dùng heuricstic.

Đối với việc tạo lời giải ban đầu bằng cách ngẫu nhiên, chúng ta đưa bài toán về dạng tạo ngẫu nhiên một mảng n phần tử không trùng nhau, cuối cùng chúng ta thêm thành phố cuối cùng là thành phố ban đầu của chúng ta.

Sơ đồ khối tạo tuyến đường ngẫu nhiên:



3.2 Hình ảnh mô tả thuật toán tạo tuyến đường ngẫu nhiên

1. Giá trị đầu vào là n độ dài tuyến đường cần tạo ngẫu nhiên.
2. Tạo mảng a có độ dài là n phần tử có giá trị phần tử tăng theo thứ tự. VD: [0, 1 , 2, 3, 4, 5] với n = 6.
3. Khởi tạo mảng kết quả result.
4. Kiểm tra mảng a còn phần tử hay không? Nếu có, chọn ngẫu nhiên một giá trị phần tử trong mảng a ra, đưa giá trị đó vào mảng kết quả result, cuối cùng xóa giá trị đó ra khỏi mảng a. Nếu không, thêm phần tử cuối cùng vào mảng với giá trị là result[0].
5. Kết thúc thuật toán.

Mô tả bằng code:

```
function randomPath(pathLength) {
	let array = Array.from({length: pathLength}).map((i, index) =>  index)
	let result = []
	for(let i = 0 ; i < pathLength ; i++) {
		let index = Math.floor(Math.random( ) * array.length)
		result.push(array[index])
		array = [
```

```
                    ...array.slice(0, index),
                    ...array.slice(index + 1)
                ]
        }
        result.push(result[0])
        return result
}
```

3.3 Mô tả code tạo tuyến đường ngẫu nhiên

Như vậy chúng ta đã hoàn toàn có thể tạo được hàm khởi tạo lời giải ban đầu cho bài toán.

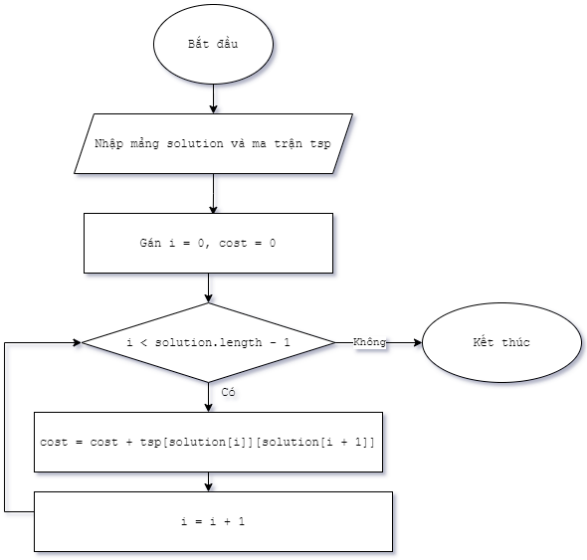
Chúng ta cần phải có thuật toán tính giá trị của lời giải đó để những bước sau có thể sử dụng. Như chúng ta đã thấy, cứ 2 thành phố sẽ có 1 giá trị trọng số giữa chúng, chúng ta nhóm các cặp thành phố, sau đó cộng tất cả các giá trị lại với nhau.

Ví dụ: chúng ta có tuyến đường [0, 1, 2, 0] thì cách tính thành phố sẽ là:

Thành phố 0 sang thành phố 1	d1
Thành phố 1 sang thành phố 2	d2
Thành phố 2 sang thành phố 0	d3

Với d1, d2, d3 là giá trị trọng số từ đỉnh a sang đỉnh b, giá trị đoạn đường $d(S) = d1 + d2 + d3$. Do đầu vào của bài toán là ma trận tsp (n cạnh x n cạnh) nên việc xác định giá trị d1, d2, d3 hoàn toàn dễ dàng. Nếu từ thành phố 0 sang thành phố 1 thì trong ma trận tsp (n cạnh x n cạnh) giá trị đó sẽ là tsp[0][1], nếu từ thành phố 1 sang thành phố 2 thì giá trị sẽ là tsp[1][2].

Sơ đồ khối tính giá trị tuyến đường:



3.4 Hình ảnh sơ đồ khối tính giá trị đường đi

1. Nhập vào giá trị tuyến đường đi solution và ma trận tsp.
2. Khởi tạo biến đếm i và giá trị tuyến đường cost là bằng 0.
3. Trong khi i < solution.length – 1, tức là chúng ta chạy đến giá trị kề cuối của solution. Nếu có, cộng vào giá trị đoạn đường vào cost, tăng biến đếm lên 1 . Nếu không kết thúc thuật toán và trả về kết quả.
4. Kết thúc thuật toán.

Mô tả bằng code:

```
function getDistanceFromSolution(tsp, solution) {
    let cost = 0;
    for (let i = 0; i < solution.length - 1; i++) {
        cost += tsp[solution[i]][solution[i + 1]];
    }
    return cost;
}
```

3.5 Hình ảnh mô tả code tính giá trị tuyến đường

3.2.1.2. Xác định điều kiện dừng của bài toán

Mấu chốt của bài toán là điều dừng, nếu không xác định điều kiện dừng đúng thì bài toán sẽ lặp vô tận. Ở đây chúng ta muốn tìm những lời giải hàng xóm của lời giải hiện tại, vậy thì bao nhiêu là đủ cho từng bài toán, cho từng trường hợp số thành phố khác nhau. Ý tưởng ở đây là chúng ta sẽ giới hạn mức tìm số lời giải hàng xóm của lời giải hiện tại bằng một con số, nếu như sau số lần tìm mà lời giải không thể cải thiện được hơn nữa thì chúng ta thực hiện dừng bài toán. Vì vậy mà chúng ta sẽ khởi tạo 2 biến $\boldsymbol{exceedTime}$(số lần tìm những lời giải hàng xóm xung quanh tại một lời giải hiện tại) và $\boldsymbol{countTotalStepAtCurrent}$(đếm số lần đã tìm lời giải hàng xóm của lời giải hiện tại). Nếu như $\boldsymbol{countTotalStepAtCurrent} > \boldsymbol{exceedTime}$ sẽ dừng bài toán còn nếu ngược lại thì sẽ thực hiện tìm lời giải hàng xóm và đánh giá kết quả đó là tốt hay là không tốt.

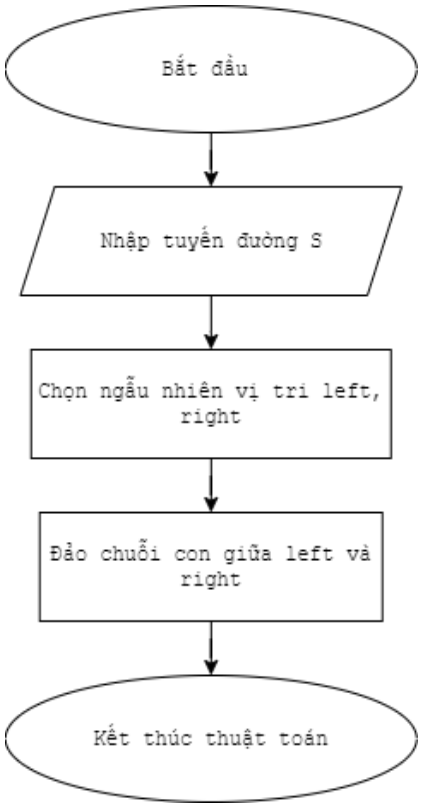
3.2.1.3. Khởi tạo tuyến đường S' và đánh giá với tuyến đường S

Sau khi xác định điều kiện dừng bài toán, chúng ta bước vào phần chính của giải thuật. Cách tạo S' lời giải hàng xóm từ tuyến đường S được cho ban đầu. S' được tạo từ hàm successor function và có rất nhiều cách để tạo:

1. Đảo (chọn 2 điểm, và chuỗi con giữa hai điểm này bị đảo)
2. Chèn (chọn 1 thành phố và chèn nó vào một vị trí ngẫu nhiên)

3. Dời chỗ (chọn một hành trình con rồi chèn nó vào một vị trí ngẫu nhiên)
4. Trao đổi qua lại (hoán vị 2 thành phố)

Chiến thuật toán đảo:

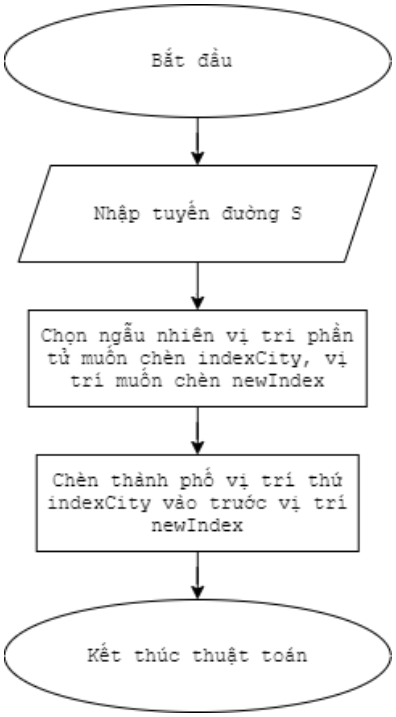



```
function generateNeighborSolutionUsingReserve(solution) {
	let left = 0
	let right = 0
	while(left == right || (right - left - 2) <= 1) {
		left = Math.floor(Math.random( ) * solution.length);
		right = Math.floor(Math.random( ) * solution.length);
		if(left > right) {
			t  = left
			left = right
			right = t
		}
	}

	let subArray = [
		...solution.slice(left + 1,right)
	]
	reverseArray = subArray.reverse( )
	let result  = [
```

```
            ...solution.slice(0, left + 1),
            ...reverseArray,
            ...solution.slice(right)
        ]
        return result
}
```

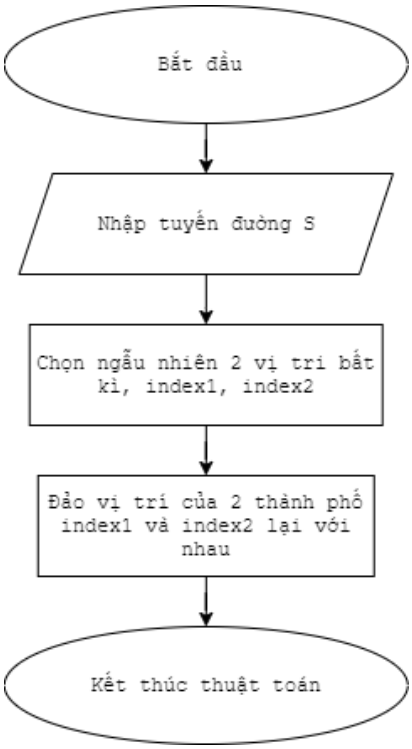
Chiến thuật chèn:




```
function generateNeighborSolutionUsingInsertCity(solution) {
        let indexCity = 0
        let newIndex = 0
        let result = []
        while (indexCity == newIndex) {
            indexCity = Math.floor(Math.random( ) * (solution.length - 2) ) + 1;
            newIndex = Math.floor(Math.random( ) * (solution.length - 2) ) + 1;
        }
        for(let i = 0 ; i < solution.length; i++) {
            if(i === indexCity) {
                continue
            } else if(i === newIndex) {
                result.push(solution[indexCity], solution[i])
            } else {
                result.push(solution[i])
            }
        }

        return result
}
```

Chiến thuật hoán vị:




```
function generateNeighborSolutionUsingPumatationTwoCity(solution) {
	let index1 = 0
	let index2 = 0
	while (index1 == index2) {
		index1 = Math.floor(Math.random() * (solution.length - 2) ) + 1;
		index2 = Math.floor(Math.random() * (solution.length - 2) ) + 1;
	}

	let result = [...solution]
	result[index1] = solution[index2]
	result[index2] = solution[index1]
	return result
}
```

Trên đây là 3 chiến thuật chúng ta áp dụng vào bài toán để sinh ra tuyến đường S'. Sau khi chúng ta xác định được S' bây giờ sẽ tới bước đánh giá chất lượng lời giải S'.

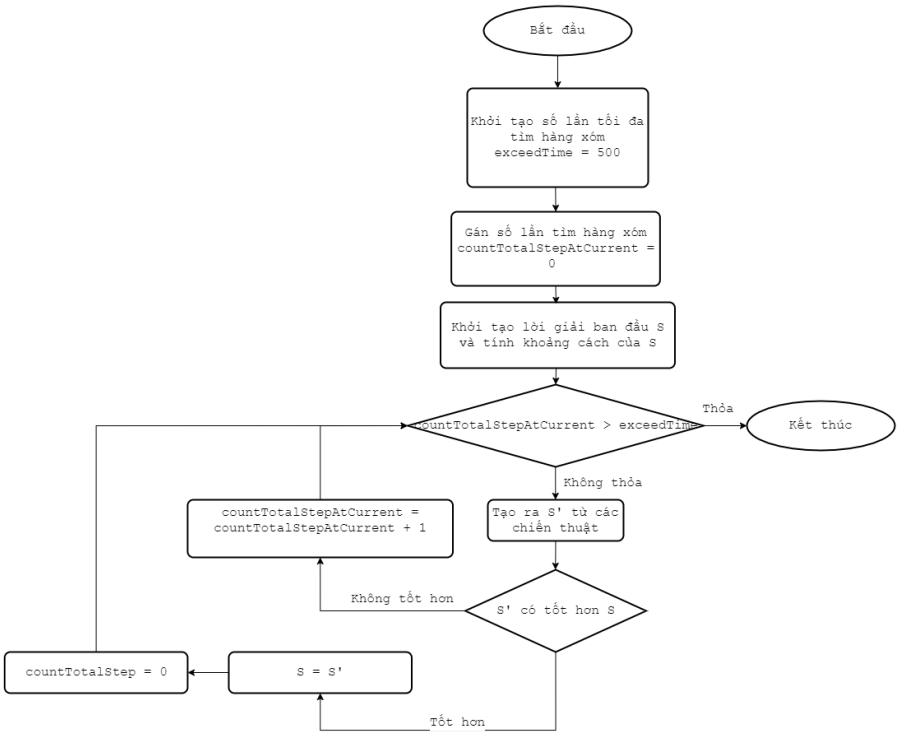
1. Tạo S'
2. Tính giá trị đường đi S'
3. So sánh giá trị cost của S' với giá trị cost của S
4. Nếu S' tốt hơn S thì sẽ thay S thành S' và $\boldsymbol{countTotalStepAtCurrent}$ sẽ bị gán về thành 0. Vì lúc này bài toán trở thành tìm tuyến đường tốt hơn của tuyến đường S' vừa mới tìm ra. Ngược lại, không cập nhật tuyến đường S, tăng giá trị

countTotalStepAtCurrent lên 1, xác nhận vừa tìm được một lời giải hàng xóm của lời giải S.

5. Xét lại điều kiện dừng của bài toán

3.2.1.4. Tổng hợp tất cả các bước

Sau khi qua tất cả 3 bước của giải thuật Hill Climbing chúng ta sẽ được lời giải tổng quát như sơ đồ dưới đây.



3.6 Hình ảnh mô tả tổng quát các quá trình thuật toán Hill Climbing

3.2.2. Giải bài toán ví dụ với giải thuật Hill Climbing

Bây giờ chúng ta sẽ trình bài bài toán trên cùng với cách giải thông qua một ví dụ nhỏ với ma trận TSP (5x5). Đây là một trong 20 bộ test được đề cập trong Chương 4.

Ma trận khoảng cách của bài toán:

	0	**1**	**2**	**3**	**4**
0	0.0	3.0	4.0	2.0	7.0
1	3.0	0.0	4.0	6.0	3.0
2	4.0	4.0	0.0	5.0	8.0
3	2.0	6.0	5.0	0.0	6.0
4	7.0	3.0	8.0	6.0	0.0

Lời giải:

<table>
<tr><th>Vòng Lặp Thứ</th><th>Lời Giải</th><th>Kết Quả Tại Vòng Lặp Thứ I</th></tr>
<tr><td>1</td><td>Bước 1: Chúng ta sẽ khởi tạo giá trị cho bài toán
exceedTime = 0,
countTotalStepAtCurrent = 0
Bước 2: Tạo một lời giải bất kì là:
Lời giải S
<table><tr><td>4</td><td>0</td><td>3</td><td>1</td><td>2</td><td>4</td></tr></table>
Chi phí đoạn đường: 27
Bước 3: countTotalStepAtCurrent > exceedtime là false
Bước 4: Chúng ta sẽ chọn chiến lược đảo, chọn 2 vị trí bất kì của kết quả S đảo chuỗi con giữa 2 đầu mút.
Lời giải S’
Với vị trí 1 và 5 ta được:
<table><tr><td>4</td><td>0</td><td>2</td><td>1</td><td>3</td><td>4</td></tr></table>Chi phí đoạn đường: 27
Bước 5: Bây giờ chúng ta sẽ đánh giá S’ tốt hơn S hay không. Do S có khoảng cách hiện tại là 27, sau khi tính S’ chúng ta nhận được khoảng cách vẫn là 27 với kết quả này không làm cải thiện được kết quả. Vậy nên S vẫn giữ nguyên. Tăng countTotalStepAtCurrent lên 1 và quay lại bước 3.</td><td>Đoạn đường
<table><tr><td>4</td><td>0</td><td>3</td><td>1</td><td>2</td><td>4</td></tr></table>
Chi phí của tuyến đường này là 27</td></tr>
<tr><td>2</td><td>Bước 3: countTotalStepAtCurrent > exceedtime là false</td><td>Đoạn đường</td></tr>
</table>

<table>
<tr><td></td><td>Bước 4:
Với vị trí 0, 4 ta được:
Đoạn S:
<table><tr><td>4</td><td>0</td><td>3</td><td>1</td><td>2</td><td>4</td></tr></table>Khởi tạo S’:
<table><tr><td>4</td><td>1</td><td>3</td><td>0</td><td>2</td><td>4</td></tr></table>Chi phí đoạn đường: 23
Bước 5: Đánh giá S’ có tốt hơn S. Là có 27 > 23. Như vậy thì chúng ta sẽ gán S thành S’, countTotalStepAtCurrent = 0</td><td><table><tr><td>4</td><td>1</td><td>3</td><td>0</td><td>2</td><td>4</td></tr></table>
Chi phí của tuyến đường này là 23</td></tr>
<tr><td>3</td><td>Bước 3: countTotalStepAtCurrent > exceedtime là false
Bước 4:
Với vị trí 1, 5 ta được:
Đoạn S:
<table><tr><td>4</td><td>1</td><td>3</td><td>0</td><td>2</td><td>4</td></tr></table>Khởi tạo S’:
<table><tr><td>4</td><td>1</td><td>2</td><td>0</td><td>3</td><td>4</td></tr></table>Chi phí đoạn đường: 19
Bước 5: Đánh giá S’ có tốt hơn S. Là có 23 > 23. Như vậy thì chúng ta sẽ gán S thành S’, countTotalStepAtCurrent = 0</td><td>Đoạn đường
<table><tr><td>4</td><td>1</td><td>2</td><td>0</td><td>3</td><td>4</td></tr></table>
Chi phí của tuyến đường 19</td></tr>
<tr><td>4</td><td>Bước 3: countTotalStepAtCurrent > exceedtime là false
Bước 4:
Với vị trí 0, 5 ta được:
Đoạn S:
<table><tr><td>4</td><td>1</td><td>2</td><td>0</td><td>3</td><td>4</td></tr></table>Khởi tạo S’:</td><td>Đoạn đường
<table><tr><td>4</td><td>1</td><td>2</td><td>0</td><td>3</td><td>4</td></tr></table>
Chi phí của tuyến đường 19</td></tr>
</table>

	<table><tr><td>**4**</td><td>**3**</td><td>**0**</td><td>**2**</td><td>**1**</td><td>**4**</td></tr></table> Chi phí đoạn đường: 19 **Bước 5:** Đánh giá S' có tốt hơn S. Là có 19 > 19 là không như vậy vẫn giữ nguyên S, tăng countTotalStepAtCurrent = countTotalStepAtCurrent + 1	
53	Sau 49 vòng lặp tìm tất cả lời giải xung quanh S. <table><tr><td>**4**</td><td>**1**</td><td>**2**</td><td>**0**</td><td>**3**</td><td>**4**</td></tr></table> Chúng ta không tìm thấy được lời giải nào cải thiện được kết quả. Vậy nên, vòng lặp sẽ dừng lại và kết quả cuối cùng chính là S	Vậy kết quả cuối cùng của bài toán chính là: <table><tr><td>**4**</td><td>**1**</td><td>**2**</td><td>**0**</td><td>**3**</td><td>**4**</td></tr></table> Chi phí của đoạn đường là: 19

3.3. Kết luận chương 3

Như vậy với chương 3 chúng ta đã hiểu được rõ cách hoạt động của thuật toán Hill Climbing. Từng các thuật toán nhỏ bên trong và cách xử lý nó. Hill Climbing là một thuật toán rất dễ tiếp cận nên vì vậy không quá khó để cài đặt.

CHƯƠNG 4 THỰC NGHIỆM ĐÁNH GIÁ

4.1.Môi trường thực nghiệm

4.1.1. Về máy tính

	Bộ phận	Mô tả
1	Loại máy	HP Pavilion 15 -cs1xxx
2	CPU	Intel core i7 9nth 8 xung
3	Hệ điều hành	Windows 10 Pro
4	Ram	8GB
5	Ổ đĩa cứng	200 SSD + 1TB HDD

4.1.2. Về ngôn ngữ

Ngôn ngữ lập trình Javascipt và NodeJS phiên bản 10x trở lên

	Thư viện	Phiên bản	Mô tả
1	fs	0.0.1-security	- Đọc, viết, xóa file - Kiểm tra folder, tạo folder
2	line-by-line	0.1.6	- Đọc từng dòng trong file
3	lodash	4.17.21	- Các hàm thư viện toán học và xử lý thông dụng nhanh chóng

4.2.Hệ thống thực nghiệm

Bộ test được lấy từ trang http://people.sc.fsu.edu và trang http://elib.zib.de. Gồm 20 bộ dữ liệu có kết quả có sẵn và sử dụng thuật toán Hill Climbing để khảo sát 20 bộ.

4.3.Kết quả thực nghiệm

Stt	Tên File	Chiều Dài	Kết Quả	Kết Quả TN
1	five_d	5	19	19
2	p01_d	15	291	291
3	gr17_d	17	2085	2085
4	fri26_d	26	937	953
5	att48	48	33522	34758
6	eil51	51	426	443
7	st70	70	675	722
8	eil76	76	538	585
9	pr76	76	108159	118773
10	kroA100	100	21282	22747
11	kroC100	100	20749	22587
12	kroD100	100	21294	23165
13	eil101	101	629	698
14	lin105	105	14379	15774
15	ch130	130	6110	6860
16	ch150	150	6528	7614
17	xql662	662	2513	3569
18	pbm436	436	1443	1878
19	pbn423	423	1365	1777
20	pbk411	411	1343	1745

4.4.Đánh giá chất lượng lời giải và kết quả

Kết quả thực nghiệm được chạy chương trình 30 lần với mỗi chiến thuật. Ở đây chúng có 3 chiến thuật. Như vậy thì một bộ dữ liệu sẽ được chạy 90 lần. Ở một thời điểm cụ thể, lời giải hiện tại sẽ tìm lời giải mới và sẽ tìm 500 lời giải hàng xóm xung quanh của nó để trao dồi giá trị lời giải tốt hơn.

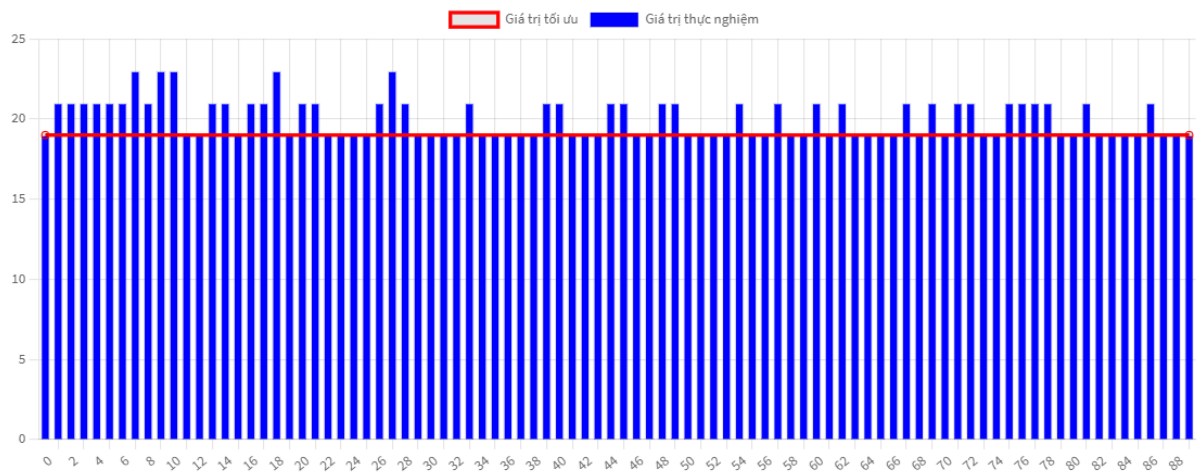
Lời giải của từng bài toán có sự khác biệt, cụ thể là với những bài toán với thành nhỏ thì Hill Climbing giải bài toán với kết quả chính xác cao. Trong khi với những bài toán thành phố cao hơn thì Hill Climbing cho kết quả chênh lệch khá nhiều so với đáp án

gốc. Như vậy thì kết quả sẽ bị giảm độ chính xác tỉ lệ thuận với số thành phố mà tăng cao.

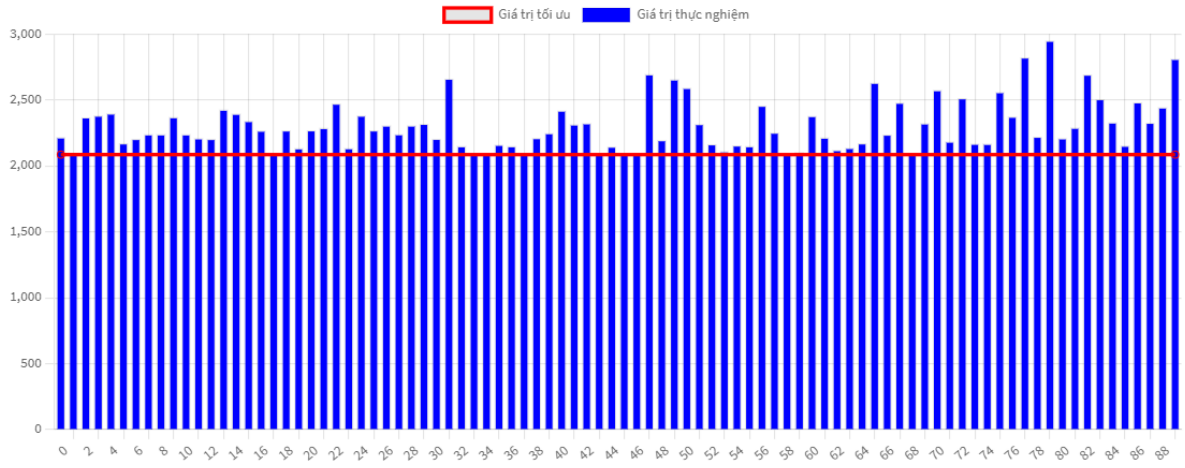
4.5.Kết luận và đóng góp

Chương 4 hoàn thành khi chúng ta viết được hoàn chỉnh chương trình Hill Climbing để giải bài toán TSP từ việc chọn lọc dữ liệu bộ test, xây dựng các chức năng cần thiết để chạy chương trình, kịch bản thực nghiệm rồi từ đó đánh giá hiệu quả của thuật toán Hill Climbing. Kết quả bài toán sẽ được đánh giá trên các tiêu chí số lần thực hiện và giá trị tối ưu.

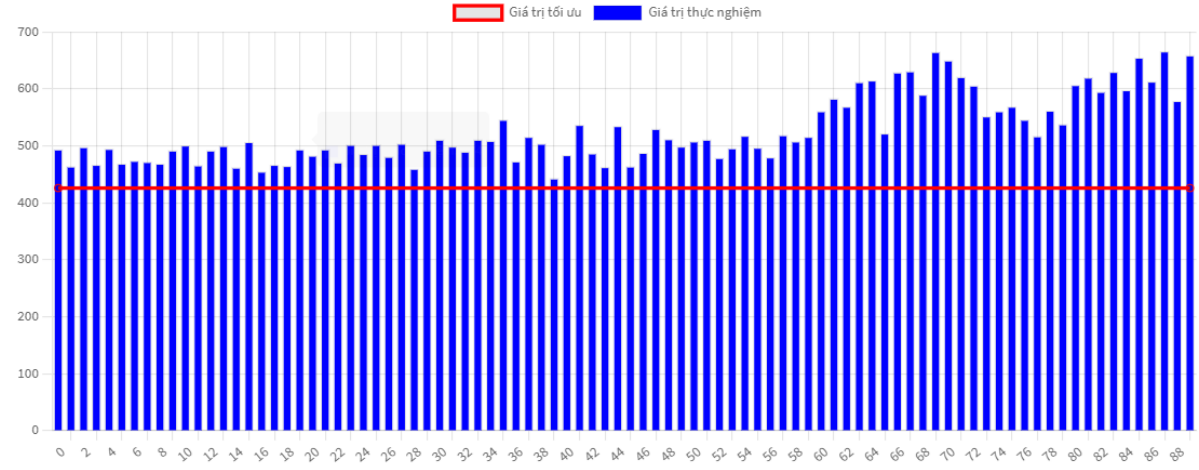
Trong quá trình làm báo cáo, chúng ta đã viết lại chương trình tiến hành mô phỏng thuật toán Hill Climbing để giải quyết các vấn đề TSP từ việc hiểu rõ bản chất và nguyên lý hoạt động. Ở đây, chúng ta sử dụng ngôn ngữ Javascript và NodeJS để thực hiện tạo chương trình, chúng ta đọc dữ liệu từ các bộ test trên mạng sau đó chuyển nó sang dạng ma trận nxn. Rồi dùng thuật toán Hill Climbing để hoàn thiện chương trình, chúng ta chạy các bộ test với mỗi bộ 90 lần đồng thời có các chiến thuật kèm bên trong bao gồm: Chèn, Thêm, Đảo. Cải tiến của thuật toán ở đây là chúng ta dùng Shotgun Hill Climbing thay vì chỉ dùng Hill Climbing để giải bài toán. Sau đó kết quả chương trình được lưu lại và thực hiện đánh giá sau khi dùng thuật toán Hill Climbing để giải quyết. Như vậy, chúng ta đã khái quát cũng như là thực nghiệm lại hoàn chỉnh thuật toán Hill Climbing.



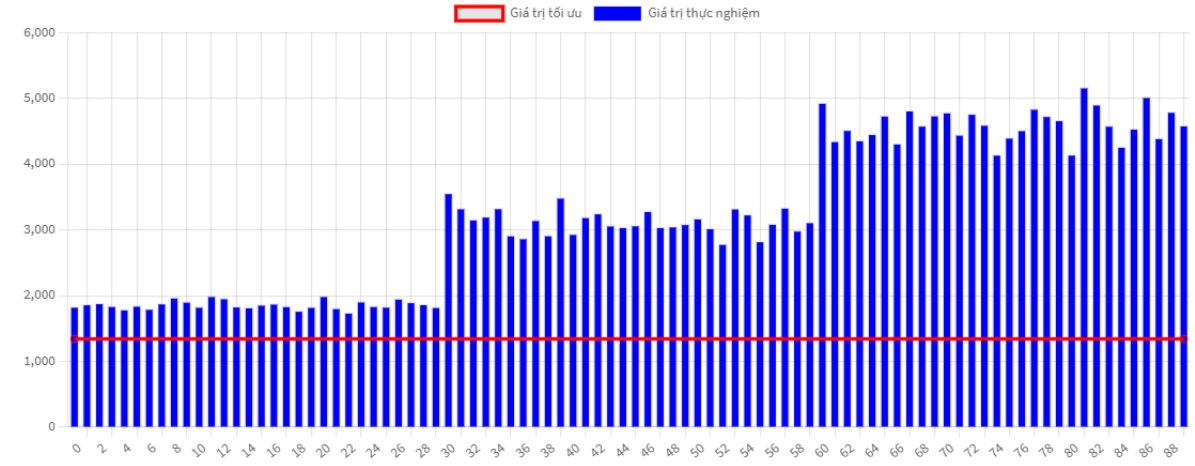
3.1 Hình ảnh bộ five_d



3.2 Hình ảnh bộ gr17_d



3.3 Hình ảnh bộ eil51



3.4 Hình ảnh bộ pkb411

Như chúng ta thấy một vài kết quả ở đây thì giá trị của kết quả từ Hill Climbing sẽ bị cao hơn dần so với kết quả tối ưu của bộ test ở thời điểm hiện tại.

Nhưng đừng quá lo lắng, đây là một giải thuật heuristic nên việc kết quả ra như vậy thì không có gì đáng quan ngại. Chúng ta có rất nhiều thuật toán khác để cải thiện điều đó.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1.Kết luận

TSP là một trong những bài toán thuộc dạng NP Hard và lời giải của nó vẫn chưa có lời giải chính xác, chỉ có thể tìm lời giải gần đúng. Như vậy, với những cách giải hiện tại chỉ cần thỏa mãn đường đi từ đỉnh bất kì sau đó quay về thành phố ban đầu thì coi như là đã đáp ứng điều kiện tiên quyết rồi. Một trong những thuật toán có thể giúp chúng ta giải quyết các bài toán TSP mà chúng ta để cập trên đây là Hill Climbing, với ý nghĩa và vai trò của nó thì Hill Climbing trở thành một bài toán giải quyết được vấn đề của TSP. Hill Climbing sẽ cho ra kết quả gần đúng với những bộ test TSP nhiều thành phố và kết quả đúng với những bộ test TSP ít thành phố hơn.

Kết quả về mặt lý thuyết:

- Lý thuyết đồ thị, nghiên cứu tìm hiểu nội dung, lịch sử, mô tả về bài toán người du lịch.
- Nghiên cứu tìm hiểu thuật toán Hill Climbing, các phiên bản thuật toán Hill Climbing.
- Sử dụng thuật toán Shotgun Hill Climbing để nâng cao chất lượng lời giải

Kết quả thực nghiệm:

- Báo cáo đã áp dụng thuật toán Hill Climbing để giải quyết bài toán người du lịch.
- Mô hình giải quyết bài toán đơn giản, dễ cài đặt và thích hợp, không cần đòi hỏi quá nhiều về phần cứng.
- Lập trình đơn giản, ngắn gọn, kết quả chính xác, áp dụng được cho nhiều bộ dữ liệu lớn.
- Thực nghiệm tìm đường đi tối ưu nhất của bài toán người du lịch có thể áp dụng cho nhiều nguồn dữ liệu khác nhau: dữ liệu ngẫu nhiên, dữ liệu từ tập tin khoảng cách giữa các điểm, dữ liệu thử nghiệm chuẩn TSPLIB. Kết quả thử 20 nghiệm giải bài toán người du lịch bằng thuật toán Hill Climbing cho thấy kết quả khá gần so với kết quả tối ưu nhất được tìm thấy cho đến thời điểm hiện tại.

5.2.Hướng phát triển

- Thực hiện so sánh kết quả của bài toán Hill Climbing với các giải thuật khác, để xem chất lượng lời giải cũng như là thời gian giải quyết. Như chúng ta đã biết ngoài thuật toán Hill Climbing còn có thuật toán GA, thuật toán local search, thuật toán Ant Colony, Thuật toán Bee. Tất cả thuật toán đều có điểm mạnh riêng và giải thuật khác nhau nên sẽ tạo ra chất lượng lời giải khác nhau. Như vậy chúng ta cần khái quát tất cả lời giải một cách cụ thể để thấy được chi tiết nhất.
- Tiếp tục tìm hiểu các thuật toán có liên quan để giải quyết bài toán TSP. Sau đó vận dụng để nâng cao chất lượng lời giải với thuật toán Hill Climbing. Chúng ta vận dụng những điểm hay điểm mạnh trong những thuật toán kia để trao dồi những điểm ở thuật toán Hill Climbing chưa có.

TÀI LIỆU THAM KHẢO

[1]. Corinne Brucato, (2013). “The Traveling Salesman Problem”, University of Pittsburgh.

[2]. Arash Mazid, (2017). “Meta-Heuristic Approaches for Solving Travelling Salesman Problem”, International Journal of Advanced Research in Computer Science.

[3]. Jakub Štencek. May (2013). “Traveling salesman problem”, JAMK University of Applied sciences

[4]. Bektas T, (Jun 2006). “The multiple traveling salesman problem: an overview of formulations and solution procedures”, JAMK University of Applied sciences.

[5]. https://www.section.io/engineering-education/understanding-hill-climbing-in-ai/

[6]. https://blog.routific.com/travelling-salesman-problem

[7]. https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

[8]. https://towardsdatascience.com/how-to-implement-the-hill-climbing-algorithm-in-python-1c65c29469de

[9]. https://www.phamduytung.com/blog/2019-02-08-getting-started-with-randomized-optimization-in-python/