THUẬT TOÁN NHÁNH CẬN GIẢI MỘT SỐ BÀI TOÁN TỐI ƯU LIÊN QUAN ĐẾN CHU TRÌNH HAMILTON DỰA TRÊN BÀI TOÁN TSP

Ngô Thúy Ngân
Trường Đại học Thủ đô Hà Nội

Tóm tắt: *Bài toán Người du lịch (TSP) là một bài toán tối ưu tổ hợp quan trọng, yêu cầu tìm chu trình ngắn nhất đi qua tất cả các đỉnh đúng một lần. Do thuộc nhóm NP-khó, TSP đòi hỏi các phương pháp giải mạnh để tìm lời giải tối ưu. Thuật toán nhánh cận (Branch and Bound – BnB) là một cách tiếp cận hiệu quả cho bài toán này. BnB chia nhỏ không gian tìm kiếm, tính cận dưới cho từng nhánh và loại bỏ các nhánh không khả thi, giúp giảm số phép thử so với quay lui truyền thống. Bài viết trình bày chi tiết cách áp dụng BnB vào TSP, bao gồm mô tả thuật toán, ví dụ minh họa và đánh giá thực nghiệm. Kết quả cho thấy BnB hoạt động tốt với bài toán có quy mô nhỏ và vừa, nhưng thời gian xử lý tăng nhanh khi số lượng đỉnh lớn. Tác giả đề xuất kết hợp BnB với các phương pháp heuristic và tối ưu lưu trữ để nâng cao hiệu quả khi giải các bài toán TSP lớn trong thực tế.*

Từ khóa: *Bài toán người du lịch; bài toán tối ưu tổ hợp; chu trình Hamilton; đường Hamilton; NP - đầy đủ; NP - khó; thuật toán nhánh cận.*

Nhận bài ngày 10.04.2025; gửi phản biện, chỉnh sửa, duyệt đăng ngày 30.05.2025
Liên hệ tác giả: Ngô Thúy Ngân; email: ntngan@daihocthudo.edu.vn

1. ĐẶT VẤN ĐỀ

Trong lĩnh vực tối ưu tổ hợp, Chu trình Hamilton và bài toán Người du lịch (Traveling Salesman Problem - TSP) là hai bài toán quan trọng có nhiều ứng dụng thực tế trong khoa học máy tính, logistics, vận tải và lập lịch. Những bài toán này đòi hỏi việc tìm kiếm các hành trình tối ưu trong đồ thị có trọng số, đảm bảo các ràng buộc về lộ trình.

Chu trình Hamilton là một đường đi qua tất cả các đỉnh trong đồ thị đúng một lần và quay lại điểm xuất phát. Khi đồ thị có trọng số, bài toán tối ưu liên quan đến việc tìm chu trình Hamilton có tổng trọng số nhỏ nhất. Một trường hợp đặc biệt của bài toán này là bài toán Người du lịch, trong đó đồ thị được giả định là đầy đủ và trọng số thể hiện chi phí hoặc khoảng cách giữa các đỉnh.

Việc giải quyết bài toán TSP có độ phức tạp cao do số lượng hành trình tăng theo giai thừa của số đỉnh. Nhiều thuật toán đã được đề xuất để tìm lời giải tối ưu hoặc xấp xỉ, trong đó thuật toán nhánh cận (Branch and Bound - BnB) là một phương pháp mạnh mẽ giúp cắt giảm không gian tìm kiếm và cải thiện hiệu suất tính toán.

Bài báo này trình bày chi tiết phương pháp nhánh cận trong việc giải quyết bài toán TSP và ứng dụng vào việc tìm chu trình Hamilton tối ưu. Tác giả sẽ phân tích lý thuyết, mô tả thuật toán, minh họa bằng ví dụ cụ thể, và đánh giá hiệu quả của phương pháp này trong thực tế.

2. NỘI DUNG

2.1. Bài toán chu trình Halmiton

Chu trình Hamilton là một đường đi trong đồ thị vô hướng hoặc có hướng sao cho đi qua mỗi đỉnh đúng một lần và quay lại đỉnh xuất phát. Chu trình này là một tập hợp các cạnh tạo thành một chu trình đơn và bao gồm tất cả các đỉnh trong đồ thị.

Bài toán Chu trình Hamilton yêu cầu xác định xem trong một đồ thị có tồn tại chu trình Hamilton hay không và nếu có, tìm chu trình Hamilton có tổng trọng số nhỏ nhất [1].

Điều kiện tồn tại:

Một đồ thị có thể không có chu trình Hamilton. Một số tiêu chí quan trọng giúp xác định sự tồn tại của chu trình Hamilton bao gồm:

- Định lý Dirac: Nếu một đồ thị vô hướng G có n đỉnh ($n \geq 3$) và mỗi đỉnh có bậc ít nhất n/2 thì G có ít nhất một chu trình Hamilton.
- Định lý Ore: Nếu với mọi cặp đỉnh không kề nhau u, v trong đồ thị G, tổng bậc của hai đỉnh này ít nhất là n ($\deg(u) + \deg(v) \geq n$), thì đồ thị G có chu trình Hamilton.
- Đồ thị đầy đủ: Một đồ thị đầy đủ với n đỉnh luôn có ít nhất một chu trình Hamilton.

Ví dụ:

Giả sử có một đồ thị với 5 đỉnh và các cạnh như sau:

V = {A, B, C, D, E}

E = {(A, B), (A, C), (A, D), (B, C), (B, D), (B, E), (C, D), (C, E), (D, E)}

Một chu trình Hamilton trong đồ thị này có thể là: A → B → E → C → D → A.

Nếu đồ thị là có trọng số, bài toán sẽ tìm chu trình Hamilton có tổng trọng số nhỏ nhất.

2.2. Bài toán người du lịch và thuật toán nhánh - cận

Bài toán Người du lịch là một trường hợp cụ thể của bài toán Chu trình Hamilton trong đó đồ thị G là đồ thị đầy đủ (mọi cặp đỉnh đều có cạnh nối), với mục tiêu tìm một chu trình Hamilton có tổng trọng số nhỏ nhất. Cụ thể:

- Xác định một hoán vị của n đỉnh ($p_1, p_2, ..., p_n$) sao cho tổng trọng số là nhỏ nhất.
- Ràng buộc: mỗi đỉnh chỉ được thăm đúng một lần.

Ví dụ:

Giả sử một nhân viên bán hàng cần đi qua 4 thành phố {A, B, C, D} và quay lại thành phố xuất phát. Khoảng cách giữa các thành phố được cho bởi ma trận trọng số:

	A	**B**	**C**	**D**
A	∞	10	15	20
B	10	∞	35	25
C	15	35	∞	30
D	20	25	30	∞

Mục tiêu là tìm chu trình tối ưu có tổng trọng số nhỏ nhất.

Giải quyết bài toán bằng thuật toán nhánh cận:

1. Bước khởi tạo: Xác định cận dưới ban đầu bằng cách lấy tổng trọng số hai cạnh nhỏ nhất từ mỗi thành phố, sau đó chia đôi.

2. Phân nhánh: Bắt đầu từ thành phố A, thử tất cả các khả năng đi đến các thành phố còn lại.
3. Tính toán cận dưới: Sau mỗi lần chọn một thành phố, cập nhật cận dưới để đánh giá xem có nên tiếp tục với nhánh đó hay không.
4. Tối ưu hóa: Nếu tìm được chu trình có tổng trọng số nhỏ hơn giá trị tối ưu hiện tại, cập nhật nghiệm tốt nhất.
5. Tiếp tục lặp cho đến khi không còn nhánh nào cần xét.

Ví dụ, nếu chúng ta chọn hành trình: A → B → D → C → A thì tổng trọng số là:

- A → B: 10
- B → D: 25
- D → C: 30
- C → A: 15
- Tổng: 10 + 25 + 30 + 15 = 80

So sánh với các hành trình khác để tìm hành trình có trọng số nhỏ nhất [2].

2.3. Thuật toán nhánh cận

Thuật toán nhánh cận hoạt động dựa trên việc cắt giảm không gian tìm kiếm bằng cách ước lượng cận dưới và phân nhánh theo các lựa chọn tốt nhất. Các bước chính của thuật toán như sau:

1. **Khởi tạo cận dưới**: Xác định cận dưới ban đầu bằng cách tính tổng trọng số hai cạnh nhỏ nhất từ mỗi đỉnh, sau đó chia đôi để có giá trị ước lượng.
2. **Phân nhánh**: Bắt đầu từ một thành phố (ví dụ A), tạo ra các nhánh bằng cách chọn thành phố tiếp theo có trọng số nhỏ nhất.
3. **Tính toán cận dưới cho mỗi nhánh**:
   - Nếu cận dưới của một nhánh lớn hơn nghiệm tối ưu tạm thời, loại bỏ nhánh đó để giảm không gian tìm kiếm.
   - Nếu nhánh đạt đến một nghiệm đầy đủ (đã đi qua tất cả các thành phố và quay lại điểm xuất phát), kiểm tra xem tổng trọng số có nhỏ hơn nghiệm tốt nhất hiện tại không.
4. **Cập nhật nghiệm tốt nhất**: Nếu tìm thấy một chu trình có tổng trọng số nhỏ hơn nghiệm tối ưu hiện tại, cập nhật giá trị tối ưu.
5. **Tiếp tục mở rộng nhánh hợp lệ** cho đến khi không còn nhánh nào cần xét [3].

3. ĐÁNH GIÁ

Thuật toán nhánh cận là một phương pháp hiệu quả để giải bài toán Chu trình Hamilton và bài toán Người du lịch, đặc biệt khi số lượng đỉnh không quá lớn. Dưới đây là một số đánh giá về hiệu suất và ứng dụng của thuật toán:

3.1. Ưu điểm

- **Tìm lời giải tối ưu**: Không giống như các phương pháp xấp xỉ như thuật toán tham lam hay heuristic, thuật toán nhánh cận đảm bảo tìm được lời giải tối ưu nếu có đủ tài nguyên tính toán.

- **Cắt giảm không gian tìm kiếm**: Sử dụng cận dưới giúp thuật toán loại bỏ những nhánh không tiềm năng, giảm số lần kiểm tra so với phương pháp quay lui (backtracking) thuần túy.
- **Linh hoạt**: Có thể áp dụng cho nhiều bài toán tối ưu khác ngoài TSP, như lập lịch, thiết kế mạng, và quản lý chuỗi cung ứng [4].

3.2. Hạn chế

- **Độ phức tạp cao**: Khi số lượng thành phố (đỉnh) tăng lên, không gian tìm kiếm vẫn tăng theo cấp số nhân, làm cho thuật toán trở nên chậm khi n lớn.
- **Bộ nhớ lớn**: Thuật toán yêu cầu lưu trữ nhiều thông tin về các nhánh và giá trị cận dưới, có thể gây ra vấn đề về bộ nhớ khi áp dụng cho đồ thị lớn.
- **Nhạy cảm với cách chọn cận dưới**: Hiệu suất của thuật toán phụ thuộc nhiều vào cách tính cận dưới, nếu cận dưới không chặt chẽ, không gian tìm kiếm có thể không được giảm đáng kể.

3.3. Ứng dụng thực tế

Thuật toán nhánh cận có nhiều ứng dụng quan trọng trong thực tế, bao gồm:

- **Lập lịch công việc**: Giúp tối ưu hóa thời gian và tài nguyên khi sắp xếp các công việc có trình tự nhất định.
- **Quản lý chuỗi cung ứng**: Tìm lộ trình giao hàng ngắn nhất để giảm chi phí vận chuyển.
- **Thiết kế vi mạch**: Tối ưu hóa đường kết nối trong các thiết bị điện tử để giảm độ trễ tín hiệu.
- **Quản lý giao thông**: Áp dụng trong việc điều phối phương tiện công cộng và quy hoạch đường đi tối ưu.

3.4. Hướng phát triển

Để cải thiện hiệu suất của thuật toán nhánh cận, có thể xem xét một số hướng phát triển sau:

- **Kết hợp với heuristic**: Kết hợp với thuật toán di truyền, mô phỏng tôi luyện (Simulated Annealing) hoặc các phương pháp heuristic khác để tìm cận dưới tốt hơn.
- **Tối ưu hóa lưu trữ**: Giảm bộ nhớ bằng cách sử dụng cấu trúc dữ liệu hiệu quả hơn, như bảng băm hay danh sách ưu tiên.
- **Tận dụng tính song song**: Triển khai thuật toán trên kiến trúc tính toán song song hoặc phân tán để tăng tốc độ xử lý.

4. KẾT LUẬN

Thuật toán nhánh cận (Branch and Bound) là một phương pháp mạnh mẽ để giải quyết bài toán Người bán hàng (Traveling Salesman Problem - TSP), giúp giảm đáng kể số lượng trường hợp cần kiểm tra bằng cách loại bỏ sớm những nhánh không có tiềm năng dẫn đến lời giải tối ưu. Cụ thể, thuật toán hoạt động bằng cách chia bài toán thành các nhánh nhỏ hơn (nhánh - branching) và sử dụng một giới hạn dưới (cận - bound) để đánh giá tiềm năng của từng nhánh. Nếu một nhánh có giá trị cận lớn hơn giá trị của lời giải tốt nhất hiện tại, nhánh đó sẽ bị loại bỏ ngay lập tức, giúp tiết kiệm đáng kể tài nguyên tính toán.

Mặc dù nhánh cận rất hiệu quả trong việc thu hẹp không gian tìm kiếm, nhưng khi số lượng thành phố tăng lên, số lượng nhánh cần xét vẫn có thể tăng theo cấp số nhân, khiến

thuật toán gặp khó khăn trong việc xử lý các bài toán có quy mô lớn. Vì vậy, trong thực tế, thuật toán này thường được kết hợp với các kỹ thuật tối ưu hóa khác để cải thiện hiệu suất.

Ứng dụng của thuật toán nhánh cận rất rộng rãi trong nhiều lĩnh vực thực tế. Trong tối ưu hóa lộ trình giao thông, thuật toán này giúp tìm ra tuyến đường ngắn nhất cho các phương tiện vận chuyển, giảm chi phí nhiên liệu và thời gian di chuyển. Trong lập lịch sản xuất, nó hỗ trợ sắp xếp trình tự công việc sao cho hiệu quả, giúp giảm thời gian chờ và tối ưu hóa năng suất. Ngoài ra, thuật toán còn được áp dụng trong thiết kế vi mạch để tối ưu hóa việc bố trí các linh kiện nhằm giảm độ trễ tín hiệu, cũng như trong lĩnh vực logistics để tối ưu hóa quá trình phân phối hàng hóa, giúp giảm chi phí vận hành và tăng hiệu suất chuỗi cung ứng.

TÀI LIỆU THAM KHẢO

1. Arora, S., & Barak, B. (2010). *Computational complexity: A modern approach*. New York, USA: Cambridge University Press.

2. Korte, B., & Vygen, J. (2011). *Combinatorial optimization: Theory and algorithms* (4th ed.). Berlin, Germany: Springer.

3. Cook, W. J. (2011). *In pursuit of the traveling salesman: Mathematics at the limits of computation*. Princeton, USA: Princeton University Press.

4. Cheikhrouhou, O., & Khoufi, I. (2021). A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy. *Artificial Intelligence Review*, 54, 5137–5186.

COMPOSITE BRANCH ALGORITHM TO SOLV SOME PROBLEMS OPTIMIZATION MATH INVOLVING HAMILTON CYCLES BASED ON THE TSP PROBLEM

Abstract: *The Traveling Sparrow Problem (TSP) is an important combinatorial optimization problem that requires finding the shortest path that visits all vertices exactly once. Because it belongs to the NP-hard group, TSP requires robust methods to find the optimal solution. The Branch and Bound (BnB) algorithm is an effective approach to this problem. BnB divides the search space, computes a lower bound for each branch, and eliminates infeasible branches, which reduces the number of trials compared to traditional backtracking. This paper presents in detail how to apply BnB to TSP, including algorithm description, illustrative examples, and experimental evaluation. The results show that BnB works well for small and medium-sized problems, but the processing time increases rapidly when the number of vertices is large. The author proposes to combine BnB with heuristic methods and storage optimization to improve the efficiency when solving large TSP problems in practice.*

Keywords: *Branch-bound algorithm; Combinatorial optimization problem; Hamiltonian cycle; NP-C (Non-deterministic Polynomial Complete; NP-Hard; Traveling Salesman Problem (TSP)*