# **1.2.2. Cơ sở lý thuyết của thuật toán nhánh cận**

#### **1.2.2.1. Lý thuyết chung về thuật toán nhánh cận**

* Thuật toán nhánh cận (Branch and Bound) là một phương pháp giải quyết các bài toán tối ưu tổ hợp, trong đó tìm kiếm được tổ chức dưới dạng cây. Mỗi nhánh của cây đại diện cho một tập các quyết định và được cắt bớt khi phát hiện ra không thể tìm được lời giải tốt hơn trong nhánh đó. Ý tưởng chính của thuật toán nhánh cận là phân chia bài toán lớn thành các bài toán con nhỏ hơn và lần lượt khám phá các lời giải tiềm năng trong các nhánh này.
* Ứng dụng của Thuật Toán Nhánh Cận:

- Tối ưu hóa tổ hợp: Thuật toán nhánh cận được sử dụng rộng rãi trong việc giải quyết các bài toán tối ưu hóa tổ hợp như bài toán người bán hàng, bài toán cái túi và phân công công việc.

- Vấn đề thỏa mãn ràng buộc: Thuật toán có khả năng xử lý hiệu quả các vấn đề thỏa mãn ràng buộc bằng cách khám phá có hệ thống không gian tìm kiếm và cắt tỉa các nhánh dựa trên các ràng buộc.

- Phân bổ tài nguyên: Thuật toán được áp dụng trong các kịch bản như phân bổ tài nguyên, nơi cần phân phối tài nguyên một cách tối ưu giữa các nhu cầu cạnh tranh.

* Ưu điểm của Thuật Toán Nhánh Cận:

-Tính tối ưu: Thuật toán đảm bảo tính tối ưu trong các giải pháp cho những bài toán thỏa mãn các điều kiện nhất định, đảm bảo rằng giải pháp tốt nhất có thể được tìm thấy.

- Hiệu quả bộ nhớ: Thuật toán thường yêu cầu ít bộ nhớ hơn so với các phương pháp tìm kiếm toàn diện khác như brute force, đặc biệt đối với các bài toán có không gian tìm kiếm lớn.

- Tính linh hoạt: Thuật toán có thể điều chỉnh cho nhiều lĩnh vực vấn đề khác nhau và có thể đáp ứng các biểu diễn và ràng buộc vấn đề khác nhau.

- Song song hóa: Các thuật toán nhánh cận có thể được song song hóa hiệu quả, cho phép khám phá nhanh chóng không gian tìm kiếm bằng cách sử dụng nhiều bộ xử lý hoặc tài nguyên tính toán.

* Nhược điểm của Thuật Toán Nhánh Cận:

- Độ phức tạp Việc triển khai các thuật toán nhánh cận có thể rất phức tạp, đặc biệt đối với các bài toán có ràng buộc phức tạp và không gian tìm kiếm lớn.

- Phụ thuộc vào Heuristic: Hiệu quả của thuật toán nhánh cận phụ thuộc nặng nề vào chất lượng của hàm giới hạn (bounding function) và các heuristics được sử dụng để hướng dẫn tìm kiếm, điều này không phải lúc nào cũng có sẵn hoặc dễ dàng thiết kế.

- Khó khăn trong môi trường động: Thuật toán không phù hợp cho các môi trường động hoặc thay đổi, nơi mà các ràng buộc hoặc mục tiêu của bài toán có thể thay đổi thường xuyên, vì nó dựa vào một phiên bản bài toán cố định để khám phá không gian tìm kiếm.

#### **1.2.2.2. Nguyên lý hoạt động của thuật toán nhánh cận**

Thuật toán sử dụng cây tìm kiếm, mỗi nút đại diện cho một trạng thái phân công công việc, mỗi nhánh đại diện cho việc lựa chọn một công việc được giao cho một người cụ thể.

* **Nhánh (Branch):** Tại mỗi cấp độ của cây, bạn chọn một công việc và thử phân công nó cho một người khác nhau, tạo ra các nhánh con. Các nhánh con sẽ tiếp tục phân công những công việc còn lại cho đến khi tất cả các công việc đều được gán.
* **Cận (Bound):** Để giảm thiểu số lượng nhánh phải xét, tại mỗi nút, ta tính một giới hạn dưới cho tổng thời gian tối thiểu có thể đạt được. Nếu giới hạn này lớn hơn hoặc bằng tổng thời gian của nghiệm tốt nhất đã biết, ta có thể cắt bỏ nhánh đó mà không cần tiếp tục xét.

#### **1.2.2.3. Các thành phần và mô hình của thuật toán nhánh cận**

Phương pháp nhánh và cận là một dạng cải tiến của phương pháp quay lui, được áp dụng để tìm nghiệm của bài toán tối ưu. Giả sử nghiệm của bài toán có thể biểu diễn dưới dạng một vector ( x1, x2, …., xn) mỗi thành phần xi ( i = 1,2, . . , n) được chọn ra từ tập Si. Mỗi nghiệm của bài toán X = ( x1, x2 ,…., xn) được xác định “ độ tốt” bằng một hàm f (X) và mục tiêu cần tìm nghiệm có giá trị f (X) đạt giá trị nhỏ nhất (hoặc đạt giá trị lớn nhất). Tư tưởng của phương pháp nhánh và cận như sau: Giả sử, đã xây dựng được k thành phần ( x1, x2, …., xk) của nghiệm và khi mở rộng nghiệm ( x1, x2, …., xk+1) , nếu biết rằng tất cả các nghiệm mở rộng của nó ( x1, x2, …., xk+1, …..) nếu không tốt bằng nghiệm tốt nhất đã biết ở thời điểm đó, thì ta không cần mở rộng từ ( x1, x2, …., xk) nữa. Như vậy, với phương pháp nhánh và cận, ta không phải duyệt toàn bộ các phương án để tìm ra nghiệm tốt nhất mà bằng cách đánh giá các nghiệm mở rộng, ta có thể cắt bỏ đi những phương án (nhánh) không cần thiết, do đó việc tìm nghiệm tối ưu sẽ nhanh hơn. Cái khó nhất trong việc áp dụng phương pháp nhánh và cận là đánh giá được các nghiệm mở rộng, nếu đánh giá được tốt sẽ giúp bỏ qua được nhiều phương án không cần thiết, khi đó thuật toán nhánh cận sẽ chạy nhanh hơn nhiều so với thuật toán vét cạn.

Thuật toán nhánh cận có thể mô tả bằng mô hình đệ quy sau:

procedure BranchBound(i);// xây dựng thành phần thứ i

begin

<Đánh giá các nghiệm mở rộng>;

if(các nghiệm mở rộng ñều không tốt hơn BestSolution) then exit;

<Xác định Si>;

for xi Si do begin

<ghi nhận thành phần thứ i>;

if (tìm thấy nghiệm) then < cập nhật BestSolution>

else BranchBound(i+1);

<loại thành phần i>;

end;

end;

A diagram of a flowchart

Description automatically generated

**Ví dụ trong giải bài toán người du lịch**

**Bài toán:** Cho n thành phố đánh số từ 1 đến n và các tuyến đường giao thông hai chiều giữa chúng, mạng lưới giao thông này được cho bởi mảng C[1..n,1..n], ở đây Cij = Cji là chi phí phí đoạn đường trực tiếp từ thành phố i đến thành phố j. Một người du lịch xuất phát từ thành phố 1, muốn đi thăm tất cả các thành phố còn lại mỗi thành phố đúng 1 lần và cuối cùng quay lại thành phố 1. Hãy chỉ ra cho người đó hành trình với chi phí ít nhất. Bài toán được gọi là bài toán người du lịch hay bài toán người chào hàng (Travelling Salesman Problem - TSP)

Dữ liệu vào trong file “TSP.INP” có dạng:

- Dòng đầu chứa số n (1<n<=20), là số thành phố

- n dòng tiếp theo, mỗi dòng n số mô tả mảng C

Kết quả ra file “TSP.OUT” có dạng:

- Dòng đầu là chi phí ít nhất

- Dòng thứ hai mô tả hành trình

A screenshot of a computer

Description automatically generated

1) Hành trình cần tìm có dạng (x1 = 1, x2, ..., xn, xn+1 = 1), ở đây giữa xi và xi+1: hai thành phố liên tiếp trong hành trình phải có đường đi trực tiếp; trừ thành phố 1, không thành phố nào được lặp lại hai lần, có nghĩa là dãy (x1, x2, ..., xn) lập thành một hoán vị của (1, 2, ..., n).

2) Duyệt quay lui: x2 có thể chọn một trong các thành phố mà x1 có đường đi trực tiếp tới, với mỗi cách thử chọn x2 như vậy thì x3 có thể chọn một trong các thành phố mà x2 có đường đi tới (ngoài x1). Tổng quát: xi có thể chọn 1 trong các thành phố chưa đi qua mà từ xi-1 có đường đi trực tiếp tới.(2 ≤ i ≤ n).

3) Nhánh cận: Khởi tạo cấu hình BestSolution có chi phí = +∞. Với mỗi bước thử chọn xi xem chi phí đường đi cho tới lúc đó có nhỏ hơn chi phí của cấu hình BestSolution không? nếu không nhỏ hơn thì thử giá trị khác ngay bởi có đi tiếp cũng chỉ tốn thêm. Khi thử được một giá trị xn ta kiểm tra xem xn có đường đi trực tiếp về 1 không ? Nếu có đánh giá chi phí đi từ thành phố 1 đến thành phố xn cộng với chi phí từ xn đi trực tiếp về 1, nếu nhỏ hơn chi phí của đường đi BestSolution thì cập nhật lại BestSolution bằng cách đi mới.