**1. Cơ sở lý thuyết của thuật toán nhánh cận**  
**1.1. 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.
* Giải thuật nhánh cận bao gồm ba thành phần chính:

♣ Một hàm đánh giá, giúp ước lượng giá trị tốt nhất có thể đạt được từ một lời giải chưa hoàn chỉnh.  
♣ Một phương pháp phân chia (branching), dùng để chia bài toán thành các bài toán con.  
♣ Một cơ chế cận, giúp loại bỏ các nhánh không cần thiết khi biết rằng chúng không thể dẫn tới lời giải tối ưu.

* Thuật toán nhánh cận hoạt động theo cách:

♣ Tạo cây tìm kiếm từ bài toán gốc, trong đó mỗi nút đại diện cho một trạng thái của bài toán.  
♣ Tại mỗi nút, nếu phát hiện ra rằng không thể tìm được lời giải tốt hơn từ nhánh này, thuật toán sẽ cắt bỏ toàn bộ nhánh (cận).  
♣ Tiếp tục duyệt qua các nhánh khác cho đến khi tìm được lời giải tối ưu hoặc không còn nhánh nào có thể duyệt.

**1.2. Thuật toán nhánh cận trong bài toán phân công công việc (Job Assignment Problem)**

* Thuật toán nhánh cận có thể áp dụng trong bài toán phân công công việc để tìm giải pháp tối ưu khi bài toán phức tạp hơn các điều kiện tham lam có thể giải quyết.
* **Ước lượng chi phí**  
  ♣ Tại mỗi bước, thuật toán sẽ sử dụng một hàm ước lượng để tính toán chi phí tối thiểu có thể đạt được từ trạng thái hiện tại của bài toán. Hàm ước lượng này sẽ giúp quyết định xem có nên tiếp tục khám phá các nhánh con hay cắt bỏ chúng.
* **Cấu trúc phân nhánh**  
  ♣ Bài toán được phân chia thành các nhánh, trong đó mỗi nhánh đại diện cho một phân công tạm thời của công việc cho người thực hiện. Mỗi lần phân công mới sẽ tạo ra một bài toán con, và bài toán này sẽ tiếp tục được phân nhánh nếu cần thiết.
* **Cận dưới và cắt nhánh**  
  ♣ Nếu tại một bước, chi phí ước lượng của một nhánh con cao hơn lời giải hiện có, thuật toán sẽ cắt bỏ toàn bộ nhánh con đó (cắt nhánh) và không tiếp tục tìm kiếm trong nhánh này. Điều này giúp giảm bớt số lượng trường hợp cần phải kiểm tra, tăng hiệu quả của thuật toán.
* **Lời giải tối ưu**  
  ♣ Thuật toán sẽ duyệt qua toàn bộ các khả năng có thể và sử dụng cơ chế cận để cắt bớt các nhánh không tiềm năng, từ đó dẫn đến lời giải tối ưu toàn cục cho bài toán phân công công việc.

**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.

**3. Các bước giải thuật**

**Bước 1: Khởi tạo**

* **Tạo trạng thái gốc**: Ở trạng thái này, chưa có công việc nào được phân công. Đây là gốc của cây tìm kiếm.

**Bước 2: Phát triển nhánh**

* Tại mỗi cấp độ của cây tìm kiếm, chọn một công việc và thử phân công công việc đó cho một người.
* Sau khi phân công một công việc cho một người, ta tạo ra các nhánh con tương ứng, trong đó mỗi nhánh đại diện cho việc phân công công việc tiếp theo cho các người còn lại.

**Bước 3: Tính toán tổng thời gian và cận dưới**

* **Tính tổng thời gian tạm thời**: Sau mỗi bước phân công, tính tổng thời gian của các công việc đã được phân công.
* **Tính cận dưới** 
  + Để tối ưu hóa quá trình tìm kiếm, tính giới hạn cận dưới cho thời gian tối thiểu có thể đạt được.
  + Cận dưới có thể được tính bằng cách cộng tổng thời gian tạm thời với thời gian nhỏ nhất có thể cho các công việc chưa được phân công.
  + Cận dưới giúp loại bỏ những nhánh không khả thi.

**Bước 4: Cắt nhánh**

* Sau khi tính được tổng thời gian và cận dưới cho một nhánh:
  + So sánh cận dưới với giá trị của nghiệm tốt nhất hiện tại (nếu có).
  + Nếu cận dưới của nhánh đó lớn hơn hoặc bằng tổng thời gian của nghiệm tốt nhất hiện tại, cắt nhánh đó và không mở rộng thêm nhánh con từ nhánh này.
  + Nếu tổng thời gian tạm thời nhỏ hơn nghiệm tốt nhất, tiếp tục phân nhánh để tìm các phương án tốt hơn.

**Bước 5: Cập nhật nghiệm tối ưu**

Khi một trạng thái phân công đầy đủ được tìm thấy (tất cả công việc đều được phân cho một người):

* + So sánh tổng thời gian của trạng thái này với nghiệm tốt nhất hiện tại.
  + Nếu tổng thời gian của trạng thái này nhỏ hơn, cập nhật nghiệm tốt nhất bằng trạng thái hiện tại.

**Bước 6: Tiếp tục tìm kiếm hoặc dừng lại**

* Lặp lại quá trình phát triển nhánh, tính cận dưới và cắt nhánh cho đến khi: Tất cả các nhánh khả thi đều được xét hoặc cắt bỏ.
* Khi không còn nhánh khả thi nào để mở rộng, nghiệm tối ưu sẽ là nghiệm cuối cùng được cập nhật.

