CS112.P11.KHTN - Bài tập nhóm 7

Nhóm 3
Nguyễn Hữu Đặng Nguyên - 23521045
Trần Vạn Tấn - 23521407

Ngày 21 tháng 11 năm 2024

Mục lục

Bài toán 1: Set Cover

Ý tưởng

Bài toán Set Cover yêu cầu chọn một số ít các tập con sao cho tất cả các phần tử trong tập U đều được bao phủ bởi các tập con từ S. Mục tiêu là tối thiểu hóa số lượng tập con được chọn, tức là bài toán là bài toán cực tiểu.

Cách giải 1: Thuật toán Tham Lam (Greedy Algorithm)

Thuật toán Tham Lam (Greedy Algorithm) là một phương pháp đơn giản nhưng hiệu quả để giải bài toán Set Cover. Ý tưởng cơ bản của thuật toán này là chọn tập con bao phủ nhiều phần tử chưa được bao phủ nhất, sau đó tiếp tục chọn tập con tiếp theo cho đến khi tất cả các phần tử trong tập U được bao phủ.

Các bước giải quyết

1. **Khởi tạo**:
    - Tạo tập con $S' = \emptyset$ và tập các phần tử chưa được bao phủ $U_{\text{left}} = U$.
2. **Lặp qua các tập con**:
    - Trong khi U_{left} chưa rỗng:
        - Chọn tập con $S_i \in S$ sao cho $|S_i \cap U_{\text{left}}|$ là lớn nhất, tức là tập con này bao phủ nhiều phần tử chưa được bao phủ nhất.
        - Thêm tập con S_i vào S'.
        - Loại bỏ các phần tử của S_i khỏi U_{left}.

3. **Trả về kết quả**:

    - Sau khi tất cả các phần tử trong U đã được bao phủ, thuật toán dừng lại và trả về tập con S'.

Độ phức tạp

Độ phức tạp của thuật toán Tham Lam cho bài toán Set Cover là $O(mn)$, trong đó:

- m là số lượng tập con trong S.
- n là số lượng phần tử trong U.

Trong mỗi bước, thuật toán cần tìm tập con S_i sao cho số phần tử chưa được bao phủ là nhiều nhất, điều này đòi hỏi phải duyệt qua tất cả các tập con và tính toán số phần tử chưa được bao phủ.

Tính gần đúng

Thuật toán Tham Lam cho bài toán Set Cover là một thuật toán gần đúng. Nó không đảm bảo tìm được giải pháp tối ưu nhưng cung cấp một giải pháp khá gần đúng với chi phí tính toán hợp lý. Tuy nhiên, nó luôn cho ra một lời giải gần đúng với mức sai lệch tối đa là $\ln n$ lần so với giải pháp tối ưu.

Cách giải 2: Lập trình Tuyến tính (LP)

Bài toán Set Cover có thể được mô hình hóa như một bài toán lập trình tuyến tính (LP) bằng cách biến mỗi tập con S_i thành một biến quyết định x_i mà có giá trị trong khoảng $[0, 1]$. Mục tiêu là tối thiểu hóa tổng số tập con được chọn, trong khi vẫn đảm bảo rằng mỗi phần tử trong tập U được bao phủ ít nhất một lần.

Tóm tắt về Bài toán Lập trình Tuyến tính (LP)

Lập trình tuyến tính (LP) là một phương pháp tối ưu hóa trong đó mục tiêu là tối thiểu hóa hoặc tối đa hóa một hàm mục tiêu tuyến tính, đồng thời thỏa mãn các ràng buộc tuyến tính.

Mô hình hóa bài toán LP

Bài toán LP có thể được mô hình hóa như sau:

- **Biến quyết định**: Các biến mà chúng ta cần tối ưu hóa (ví dụ: $x_1, x_2, \ldots, x_n$).
- **Hàm mục tiêu**: Hàm mà chúng ta muốn tối thiểu hóa hoặc tối đa hóa. Ví dụ:

$$\text{Minimize or Maximize} \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

- **Ràng buộc**: Các điều kiện mà các biến phải thỏa mãn. Ví dụ:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$$

với $i = 1, 2, \ldots, m$.

Các bước giải quyết

1. **Mô hình hóa bài toán dưới dạng LP**:

   - Đặt một biến x_i cho mỗi tập con $S_i \in S$, với $x_i = 1$ nếu tập con S_i được chọn và $x_i = 0$ nếu không.
   - Mục tiêu là tối thiểu hóa tổng số các biến x_i đã chọn, tức là:

$$\min \sum_{i=1}^{m} x_i$$

   - Đảm bảo mỗi phần tử $u_j \in U$ phải được bao phủ ít nhất một lần, nghĩa là tổng các biến x_i cho tất cả các tập con S_i chứa u_j phải lớn hơn hoặc bằng 1. Điều này có thể được biểu diễn như sau:

$$\sum_{i:u_j \in S_i} x_i \geq 1, \quad \forall u_j \in U$$

   - Giới hạn: Các biến x_i phải nằm trong khoảng $[0, 1]$:

$$x_i \in [0, 1], \quad \forall i$$

2. **Giải bài toán LP**: Bài toán này có thể được giải bằng các phương pháp giải LP như thuật toán Simplex hoặc các phương pháp giải LP nội bộ. Giải thuật LP sẽ tìm ra giá trị tối ưu của các biến x_i sao cho hàm mục tiêu đạt được giá trị tối thiểu và các ràng buộc được thỏa mãn.

3. **Làm tròn các giá trị x_i**:

   - Sau khi giải xong bài toán LP, các giá trị của biến x_i có thể nằm trong khoảng $[0, 1]$, vì vậy cần phải làm tròn các giá trị này để có quyết định cuối cùng về việc chọn các tập con.
   - Làm tròn các giá trị của x_i theo cách sau:
     - Nếu $x_i \geq 0.5$, chọn tập con S_i (tức là $x_i = 1$).
     - Nếu $x_i < 0.5$, bỏ qua tập con S_i (tức là $x_i = 0$).

4. **Trả về tập con S'**:

   - Các tập con S_i có $x_i = 1$ được chọn vào tập S'.
   - Trả về tập S' chứa các tập con được chọn.

Độ phức tạp của Thuật toán LP

Độ phức tạp của bài toán LP chủ yếu phụ thuộc vào số lượng biến và ràng buộc trong mô hình LP.
Giả sử: - m là số lượng tập con trong S.
- n là số lượng phần tử trong U.
Độ phức tạp của việc giải bài toán LP với các phương pháp giải LP chuẩn như **Simplex** là $O(m^3)$ hoặc $O(m^2n)$, tùy vào phương pháp và cấu trúc của bài toán. Tuy nhiên, với các phương pháp nội bộ, độ phức tạp có thể thấp hơn, có thể đạt được $O(m^2n)$.

Tính gần đúng của Thuật toán LP

Tính gần đúng của thuật toán LP dựa trên **bộ làm tròn** (rounding) và **tính toán độ chính xác** của bài toán. Bài toán LP Relaxation cho bài toán Set Cover thường có độ gần đúng là $O(\ln n)$, tức là giải pháp của thuật toán sẽ không vượt quá chi phí tối ưu gấp $\ln n$ lần, trong đó n là số phần tử trong U.

Bài toán 2: TSP (Travelling Salesman Problem)

Ý tưởng

Bài toán TSP yêu cầu tìm một chu trình Hamiltonian sao cho tổng chi phí (hoặc tổng quãng đường) đi qua tất cả các đỉnh là nhỏ nhất. Mục tiêu là tối thiểu hóa tổng chi phí, do đó bài toán là bài toán cực tiểu.

Cách giải 1: Thuật toán Tham Lam (Nearest Neighbor)

Thuật toán Tham Lam (Nearest Neighbor) là một phương pháp đơn giản để giải bài toán TSP. Ý tưởng cơ bản là bắt đầu từ một đỉnh bất kỳ, và tại mỗi bước, chọn đỉnh gần nhất chưa được thăm. Thuật toán này sẽ tiếp tục cho đến khi tất cả các đỉnh đều được thăm và tạo thành một chu trình Hamiltonian.

Các bước giải quyết

1. **Chọn đỉnh khởi đầu**:
   - Chọn một đỉnh bất kỳ v_{start} làm điểm xuất phát của chu trình.
2. **Khởi tạo danh sách các đỉnh đã thăm**:
   - Tạo một danh sách các đỉnh đã thăm, bắt đầu với v_{start}.
3. **Lặp qua các đỉnh chưa thăm**:
   - Tại mỗi bước, chọn đỉnh v_{nearest} chưa thăm sao cho khoảng cách từ đỉnh hiện tại v_{current} đến v_{nearest} là nhỏ nhất.

- Thêm v_{nearest} vào danh sách các đỉnh đã thăm và cập nhật v_{current} thành v_{nearest}.

4. **Kết thúc chu trình**:
   - Sau khi thăm tất cả các đỉnh, quay lại đỉnh ban đầu v_{start} để hoàn thành chu trình.
5. **Trả về chu trình**:
   - Trả về chu trình Hamiltonian được tạo thành từ các đỉnh đã thăm.

Độ phức tạp

Độ phức tạp của thuật toán Tham Lam (Nearest Neighbor) là $O(n^2)$, trong đó n là số lượng đỉnh trong đồ thị. Điều này là do thuật toán phải tính toán khoảng cách từ đỉnh hiện tại đến tất cả các đỉnh chưa thăm để chọn đỉnh gần nhất. Với mỗi đỉnh, cần phải duyệt qua các đỉnh còn lại, dẫn đến độ phức tạp tổng cộng là $O(n^2)$.

Tính gần đúng

Thuật toán Tham Lam (Nearest Neighbor) cung cấp một lời giải gần đúng cho bài toán TSP, nhưng không đảm bảo tối ưu. Độ gần đúng của thuật toán này là **O(n)**, có nghĩa là chi phí chu trình tìm được sẽ không vượt quá một hằng số nhân với chi phí chu trình tối ưu.

Cách giải 2: Thuật toán 2-Approximation với MST (Minimum Spanning Tree)

Thuật toán 2-Approximation cho bài toán TSP sử dụng cây khung nhỏ nhất (MST) để tạo ra một chu trình gần đúng. Thuật toán này dựa trên một số lý thuyết trong lý thuyết đồ thị, trong đó cây khung nhỏ nhất giúp chúng ta tìm ra một chu trình gần nhất trong đồ thị mà không cần phải kiểm tra tất cả các chu trình có thể có.

Các bước giải quyết

1. **Xây dựng cây khung nhỏ nhất (MST)**: Sử dụng các thuật toán như **Kruskal** hoặc **Prim** để xây dựng cây khung nhỏ nhất từ đồ thị $G = (V, E)$.
2. **Duyệt cây theo thứ tự Euler**:
   - Sau khi có cây khung nhỏ nhất, chúng ta duyệt cây theo thứ tự Euler (thực hiện duyệt theo các cạnh của cây khung). Thứ tự Euler trong cây này sẽ tạo ra một chuỗi các đỉnh mà chúng ta có thể đi qua.
   - Trong quá trình duyệt cây, có thể gặp các đỉnh bị lặp lại. Điều này là do mỗi đỉnh có thể xuất hiện trong cả hai lần khi duyệt theo các cạnh của cây.
3. **Loại bỏ đỉnh lặp**:

- Khi duyệt cây theo thứ tự Euler, các đỉnh có thể bị lặp lại (do mỗi đỉnh có thể được thăm nhiều lần trong quá trình duyệt). Ta sẽ loại bỏ các đỉnh lặp này để đảm bảo rằng mỗi đỉnh chỉ xuất hiện một lần trong chu trình cuối cùng.

4. **Thêm đỉnh bắt đầu vào cuối path**:

   - Sau khi đã duyệt xong cây và loại bỏ các đỉnh lặp, ta sẽ thêm đỉnh bắt đầu vào cuối chuỗi, để tạo thành một chu trình Hamiltonian.

5. **Trả về chu trình**:

   - Chu trình cuối cùng được tạo thành từ các đỉnh đã thăm, và ta sẽ trả về chu trình này là kết quả gần đúng cho bài toán TSP.

Độ phức tạp và Tính gần đúng

- **Độ phức tạp**: Thuật toán này có độ phức tạp $O(E \log V)$, trong đó V là số lượng đỉnh và E là số lượng cạnh của đồ thị. Độ phức tạp này chủ yếu đến từ việc xây dựng cây khung nhỏ nhất.

- **Tính gần đúng**: Thuật toán này là một thuật toán gần đúng 2-Approximation, tức là tổng chi phí của chu trình tìm được sẽ không vượt quá gấp 2 lần chi phí của chu trình tối ưu. Điều này có nghĩa là chúng ta sẽ không tìm ra chu trình tối ưu, nhưng kết quả sẽ luôn tốt với mức sai lệch tối đa là 2 lần chi phí tối ưu.

Cận trên và cận dưới

Bài toán Set Cover

- Cận dưới: Số lượng tối thiểu các tập con cần chọn không thể nhỏ hơn số phần tử trong U chia cho số phần tử lớn nhất mà một tập con S_i có thể bao phủ.

- Cận trên: Số lượng tối đa tập con là $|S|$, vì mỗi tập con có thể được chọn một lần.

Bài toán TSP

- Cận dưới: Tổng chi phí tối thiểu phải lớn hơn hoặc bằng với cây khung nhỏ nhất (MST) của đồ thị.

- Cận trên: Tổng chi phí tối đa có thể đạt được khi đi qua tất cả các đỉnh theo cách ngẫu nhiên mà không có tối ưu nào.

Loại bài toán

Set Cover

Bài toán Set Cover là bài toán cực tiểu vì mục tiêu là tối thiểu hóa số lượng các tập con được chọn sao cho tất cả các phần tử trong U được bao phủ.

TSP

Bài toán TSP là bài toán cực tiểu vì mục tiêu là tối thiểu hóa tổng chi phí (hoặc quãng đường) của chu trình Hamiltonian.