ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BỘ MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Báo cáo

Đề tài: Giải bài toán Set Cover và TSP bằng thuật toán gần đúng

---

Môn học: Phân tích và thiết kế thuật toán

Sinh viên thực hiện:

Nguyen Minh Huy(23520634)

Do Quang Luc(23520902)

Giáo viên hướng dẫn:

Nguyen Thanh Son

Ngày 21 tháng 11 năm 2024

Bài toán 1: Set Cover

Mô tả bài toán: - Cho tập $U = \{u_1, u_2, \dots, u_n\}$ và tập $S = \{S_1, S_2, \dots, S_m\}$, với mỗi $S_i \subseteq U$. - Mục tiêu: Chọn số lượng tập con ít nhất từ S sao cho mọi phần tử trong U xuất hiện ít nhất một lần trong các tập con được chọn.

—

Phương pháp 1: Thuật toán Greedy

Ý tưởng: - Chọn tập con S_i bao phủ nhiều phần tử chưa được chọn nhất. - Lặp lại cho đến khi toàn bộ U được bao phủ.

Mã giả C++:

```cpp
vector<set<int>> set_cover(set<int> U, vector<set<int>> S) {
    set<int> covered;
    vector<set<int>> result;

    while (covered != U) {
        set<int> best_set;
        int max_new_elements = 0;

        for (auto& Si : S) {
            set<int> new_elements;
            set_difference(Si.begin(), Si.end(), covered.begin(), covered.end(),
                           inserter(new_elements, new_elements.begin()));
            if (new_elements.size() > max_new_elements) {
                best_set = Si;
                max_new_elements = new_elements.size();
            }
        }

        covered.insert(best_set.begin(), best_set.end());
        result.push_back(best_set);
        S.erase(remove(S.begin(), S.end(), best_set), S.end());
    }

    return result;
```

```
}
```

—

Phương pháp 2: Làm tròn từ bài toán quy hoạch tuyến tính (LP Relaxation)

Ý tưởng: 1. Chuyển bài toán thành bài toán quy hoạch tuyến tính (LP):

$$\min \sum_{i=1}^{m} x_i, \quad \text{với } x_i \in [0,1], \forall i$$

2. Giải bài toán LP, sau đó làm tròn giá trị x_i để chọn tập con S_i.

—

Bài toán 2: Travelling Salesman Problem (TSP)

Mô tả bài toán: - Cho đồ thị $G = (V, E)$, với mỗi cạnh $(i, j) \in E$ có trọng số $c(i, j)$. - Mục tiêu: Tìm chu trình Hamilton có tổng trọng số nhỏ nhất.

—

Phương pháp 1: Thuật toán Greedy

Ý tưởng: - Bắt đầu từ một đỉnh bất kỳ. - Mỗi bước, chọn đỉnh gần nhất chưa được thăm. - Lặp lại cho đến khi quay lại đỉnh xuất phát.

Mã giả C++:

```cpp
vector<int> tsp_greedy(vector<vector<int>> G, int start) {
    int n = G.size();
    vector<bool> visited(n, false);
    vector<int> path;
    int current = start;

    path.push_back(current);
    visited[current] = true;

```

```cpp
    while (path.size() < n) {
        int next_city = -1;
        int min_distance = INT_MAX;

        for (int i = 0; i < n; ++i) {
            if (!visited[i] && G[current][i] < min_distance) {
                next_city = i;
                min_distance = G[current][i];
            }
        }

        visited[next_city] = true;
        path.push_back(next_city);
        current = next_city;
    }

    path.push_back(start);
    return path;
}
```

—

Phương pháp 2: Thuật toán Christofides

Ý tưởng: 1. Tìm cây khung nhỏ nhất (MST) từ đồ thị G. 2. Tìm matching trên tập các đỉnh bậc lẻ trong MST. 3. Kết hợp matching với MST để tạo đồ thị Euler. 4. Trích xuất chu trình Hamilton từ đồ thị Euler.

Mã giả C++:

```cpp
vector<int> christofides_tsp(vector<vector<int>> G) {
    auto MST = find_mst(G);
    auto odd_vertices = find_odd_vertices(MST);
    auto matching = find_perfect_matching(odd_vertices, G);
    auto eulerian_graph = combine_mst_matching(MST, matching);
    return extract_hamiltonian_cycle(eulerian_graph);
}
```

—

So sánh các phương pháp

Bài toán	**Phương pháp**	**Độ phức tạp**	**Ưu điểm / Nhược điểm**
Set Cover	Greedy	$O(n \cdot m)$	Đơn giản, nhanh nhưng không tối ưu
Set Cover	LP Relaxation	$O(n^3)$	Chính xác hơn, yêu cầu solver LP
TSP	Greedy	$O(n^2)$	Nhanh, dễ triển khai, không chính xác
TSP	Christofides	$O(n^3)$	Kết quả gần tối ưu, phức tạp hơn

—

Kết luận

- Đối với bài toán Set Cover, thuật toán Greedy thường được sử dụng do đơn giản và hiệu quả. - Đối với TSP, Christofides là phương pháp gần đúng tốt với sai số không quá 1.5 lần lời giải tối ưu. - Các thuật toán gần đúng mang lại hiệu quả thực tiễn cao, đặc biệt với bài toán lớn.

Bài toán 1: Set Cover

Mô tả bài toán: - Cho tập $U = \{u_1, u_2, \ldots, u_n\}$ và tập $S = \{S_1, S_2, \ldots, S_m\}$, với mỗi $S_i \subseteq U$. - Mục tiêu: Chọn số lượng tập con ít nhất từ S sao cho mọi phần tử trong U xuất hiện ít nhất một lần trong các tập con được chọn.

—

Phương pháp 1: Thuật toán Greedy

Ý tưởng: - Chọn tập con S_i bao phủ nhiều phần tử chưa được chọn nhất. - Lặp lại cho đến khi toàn bộ U được bao phủ.

Mã giả C++:

```cpp
vector<set<int>> set_cover(set<int> U, vector<set<int>> S) {
    set<int> covered;
    vector<set<int>> result;

    while (covered != U) {
        set<int> best_set;
        int max_new_elements = 0;

        for (auto& Si : S) {
            set<int> new_elements;
            set_difference(Si.begin(), Si.end(), covered.begin(), covered.end(),
                           inserter(new_elements, new_elements.begin()));
            if (new_elements.size() > max_new_elements) {
                best_set = Si;
                max_new_elements = new_elements.size();
            }
        }
```

```

        covered.insert(best_set.begin(), best_set.end());
        result.push_back(best_set);
        S.erase(remove(S.begin(), S.end(), best_set), S.end());
    }

    return result;
}
```

—

Phương pháp 2: Làm tròn từ bài toán quy hoạch tuyến tính (LP Relaxation)

Ý tưởng: 1. Chuyển bài toán Set Cover thành bài toán quy hoạch tuyến tính (LP):

$$\min \sum_{i=1}^{m} x_i, \quad \text{với } x_i \in [0,1], \forall i$$

và ràng buộc:

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

2. Giải bài toán quy hoạch tuyến tính để tìm nghiệm x_i. 3. Làm tròn giá trị x_i (ví dụ: chọn S_i nếu $x_i \geq 0.5$) để xác định tập con S'.

Mã giả C++:

```
// LP Relaxation for Set Cover (pseudo-code)
vector<int> lp_set_cover(set<int> U, vector<set<int>> S) {
    int m = S.size();
    vector<double> x(m, 0.0); // L Íi g i i t LP solver
    vector<int> result;

    // G i i b i to n LP v l y gi tr x_i
    x = solve_linear_program(U, S);

    // L m tr n c c gi tr x_i
    for (int i = 0; i < m; ++i) {
```

```
        if (x[i] >= 0.5) {  // L m tr n: c h n  t p con n ỉu x_i >= 0.5
            result.push_back(i);
        }
    }

    return result;
}
```

—

Bài toán 2: Travelling Salesman Problem (TSP)

Mô tả bài toán: - Cho đồ thị $G = (V, E)$, với mỗi cạnh $(i, j) \in E$ có trọng số $c(i, j)$. - Mục tiêu: Tìm chu trình Hamilton có tổng trọng số nhỏ nhất.

—

Phương pháp 1: Thuật toán Greedy

Ý tưởng: - Bắt đầu từ một đỉnh bất kỳ. - Mỗi bước, chọn đỉnh gần nhất chưa được thăm. - Lặp lại cho đến khi quay lại đỉnh xuất phát.

Mã giả C++:

```
vector<int> tsp_greedy(vector<vector<int>> G, int start) {
    int n = G.size();
    vector<bool> visited(n, false);
    vector<int> path;
    int current = start;

    path.push_back(current);
    visited[current] = true;

    while (path.size() < n) {
        int next_city = -1;
        int min_distance = INT_MAX;

        for (int i = 0; i < n; ++i) {
```

```
            if (!visited[i] && G[current][i] < min_distance) {
                next_city = i;
                min_distance = G[current][i];
            }
        }

        visited[next_city] = true;
        path.push_back(next_city);
        current = next_city;
    }

    path.push_back(start);
    return path;
}
```

—

So sánh các phương pháp

Bài toán	**Phương pháp**	**Độ phức tạp**	**Ưu điểm / Nhược điểm**
Set Cover	Greedy	$O(n \cdot m)$	Đơn giản, nhanh nhưng không tối ưu
Set Cover	LP Relaxation	$O(n^3)$	Chính xác hơn, yêu cầu solver LP
TSP	Greedy	$O(n^2)$	Nhanh, dễ triển khai, không chính xác

—

Kết luận

- Đối với bài toán Set Cover, thuật toán Greedy thường được sử dụng do đơn giản và hiệu quả. - Đối với TSP, thuật toán Greedy là một phương pháp gần đúng đơn giản và hiệu quả.