Practice Algorithm Design

Nhóm 9

December 4, 2025

Abstract

Báo cáo này trình bày một phương pháp tiếp cận tối ưu để giải quyết bài toán tìm đường đi ngắn nhất trên một đồ thị đầy đủ bị ràng buộc bởi hàm chi phí phi tuyến tính. Cấu trúc chi phí phụ thuộc vào thuộc tính $\max(0, a_{u,k} - a_{v,k})$ và chi phí đầu vào c_v, dẫn đến cấu trúc đồ thị tiềm năng $O(N^2)$ cạnh. Giải pháp đề xuất tích hợp thuật toán **Dijkstra** với **Quy hoạch động (DP) Tách Trạng thái** và **Kỹ thuật Tối ưu hóa Quét Tuyến (Sweep-line Optimization)**. Phương pháp này đã được chứng minh là giảm đáng kể độ phức tạp tính toán, đạt được hiệu suất tối ưu $\mathbf{O(N \cdot M \log(N \cdot M))}$, vượt trội so với các thuật toán dựa trên duyệt vét cạn truyền thống.

---

1 Mô tả Bài Toán và Công thức Chi Phí

Bài toán đặt ra là tìm chi phí tích lũy cực tiểu để thiết lập một chuỗi thắng cuộc liên tiếp $P_1 \to P_2 \to \cdots \to P_n$. Chi phí chuyển đổi trạng thái từ đỉnh u sang đỉnh v được định nghĩa như sau:

$$\text{Cost}(u \to v) = d[u] + c_v + \min_{1 \le k \le m} \{\Delta_k(u, v)\}$$

Trong đó:

- $d[u]$: Chi phí tích lũy cực tiểu để đạt đến u.
- c_v: Chi phí đầu vào (thuê) cho đỉnh v.
- $\Delta_k(u, v) = \max(0, a_{u,k} - a_{v,k})$: Chi phí nâng cấp thuộc tính k, là hàm phi tuyến tính.

Do sự phụ thuộc vào $\min_k$ và Δ_k, việc tính toán $d[v]$ dựa trên tất cả u đã xử lý đòi hỏi một thuật toán có khả năng tối ưu hóa truy vấn.

2 Phương Pháp Thiết Kế Thuật Toán

Giải pháp được xây dựng trên nguyên lý của Thuật toán Đường đi Ngắn nhất Đơn nguồn (Single-Source Shortest Path - SSSP), cụ thể là Dijkstra, nhưng được mở rộng bằng hai kỹ thuật chính.

2.1 Kỹ thuật Tách Trạng thái (State Separation DP)

Để giải quyết bản chất phức tạp của hàm chi phí, ta phân tách trạng thái của đỉnh i thành $M + 1$ trạng thái DP: $\mathbf{dp}[i][k]$.

- $k = 0$ (**Trạng thái Tổng hợp**): Chi phí $dp[i][0]$ đại diện cho chi phí đã bao gồm c_i. Trạng thái này là đầu vào cho Priority Queue (PQ) của Dijkstra và được sử dụng để bắt đầu một chuỗi thách đấu mới ($u \to v$).

- $k \in [1, m]$ (**Trạng thái Tối ưu hóa thuộc tính**): Chi phí $dp[i][k]$ **chưa** bao gồm chi phí c_i. Trạng thái này được tối ưu hóa riêng biệt theo thuộc tính k, phục vụ cho việc truyền giá trị tối thiểu trong quá trình Quét Tuyến.

Mô hình này cho phép kiểm soát chặt chẽ thời điểm áp dụng chi phí đầu vào c_i ($dp[i][k] \to dp[i][0]$) và đảm bảo tính chính xác của chi phí nâng cấp.

2.2 Tối Ưu Hóa Quét Tuyến (Sweep-line Optimization)

Kỹ thuật này được áp dụng để giảm số lượng cạnh ảo được xét từ $O(N^2)$ xuống $O(N)$ trong mỗi lần duyệt thuộc tính.

1. **Tiền xử lý:** Đối với mỗi thuộc tính k, tập hợp các Pokémon được sắp xếp theo $a_{i,k}$ tăng dần và lưu trữ trong danh sách $\mathbf{vt}[\mathbf{k}]$.
2. **Cập nhật Hiệu quả (Cạnh Động):** Khi trạng thái tối ưu $\mathbf{dp}[u][k]$ được rút ra từ PQ:
   - **Truyền chi phí (Trường hợp $a_{v,k} \geq a_{u,k}$):** Ta duyệt ngược (sweep backward) danh sách $vt[k]$. Các đỉnh v có $a_{v,k} \geq a_{u,k}$ nhận được cập nhật chi phí tối ưu $dp[u][k]$ với trọng số 0. Mỗi cặp (v, k) được **loại bỏ vĩnh viễn** (`pop_back`) khỏi tập xét sau khi cập nhật, đảm bảo tính tuyến tính hóa của quá trình.
   - **Xử lý Nâng cấp (Trường hợp $a_{v,k} < a_{u,k}$):** Đỉnh v_{last} còn lại trong $vt[k]$ là đỉnh tối ưu nhất để áp dụng công thức nâng cấp, do đó chỉ một cạnh đến v_{last} được xét với trọng số $a_{u,k} - a_{v_{last},k}$.

Quá trình này đảm bảo rằng tổng số lần loại bỏ (thao tác trên các cạnh) là $O(N \cdot M)$ trong suốt thời gian chạy của thuật toán.

3 Phân Tích Hiệu Năng Tính Toán

3.1 Độ phức tạp Thời gian ($T(n, m)$)

- **Chi phí Sắp xếp:** $O(M \cdot N \log N)$ cho việc tiền xử lý M danh sách thuộc tính.
- **Chi phí Dijkstra:** Tổng số trạng thái (đỉnh ảo) $V' = O(N \cdot M)$. Tổng số cạnh ảo E' được thêm vào PQ, bao gồm cạnh chuyển trạng thái $k \to 0$ và cạnh Quét Tuyến, là $O(N \cdot M)$.

Áp dụng công thức độ phức tạp cho Dijkstra trên đồ thị E' cạnh và V' đỉnh:

$$T(n, m) = O(M \cdot N \log N) + O(E' \log V')$$

$$\mathbf{T(n, m) = O(N \cdot M \log(N \cdot M))}$$

Độ phức tạp này được coi là **tối ưu** cho bài toán này, đặc biệt trong các môi trường kiểm thử với ràng buộc thời gian chặt chẽ.

3.2 Độ phức tạp Không gian ($S(n, m)$)

- **Lưu trữ Dữ liệu Chính:** $O(N \cdot M)$ cho các ma trận a, dp, và danh sách vt đã sắp xếp.
- **Cấu trúc Phụ trợ:** $O(N \cdot M)$ cho hàng đợi ưu tiên (PQ), do số lượng trạng thái là $O(N \cdot M)$.

Độ phức tạp không gian là $\mathbf{S(n, m) = O(N \cdot M)}$.

4 Kết Luận

Phương pháp tích hợp Dijkstra, DP Tách Trạng thái và Tối ưu hóa Quét Tuyến đã cung cấp một giải pháp mạnh mẽ và hiệu quả cho bài toán tìm đường đi ngắn nhất bị ràng buộc phức tạp. Việc tuyến tính hóa các truy vấn min và kiểm soát chặt chẽ việc cập nhật chi phí thông qua *pop_back* trên danh sách đã sắp xếp là yếu tố then chốt, giúp đạt được độ phức tạp thời gian cực kỳ cạnh tranh $\mathbf{O(N \cdot M \log(N \cdot M))}$.

account codeforces: $\text{Just}_M emories$