

BÁO CÁO

Bài toán A: Fucs và gánh nặng mưu sinh

Nhóm thực hiện: Nhóm 9

Ngày 15 tháng 12 năm 2025

1 Subtask 1: Tìm nghiệm tối ưu ($N \leq 20$)

1.1 Phương pháp thiết kế thuật toán

Phương pháp sử dụng: Quay lui (Backtracking) kết hợp Nhánh cận (Branch and Bound).

Nguyên lý và cách áp dụng:

- **Tiền xử lý (Sắp xếp):** Trước khi thực hiện quay lui, danh sách các tác vụ được sắp xếp tăng dần theo thời hạn hoàn thành (t_i) - quy tắc EDD (Earliest Due Date). Điều này giúp việc kiểm tra tính khả thi của một lịch trình trở nên đơn giản hơn: nếu một tập hợp tác vụ là khả thi, việc thực hiện chúng theo thứ tự EDD luôn là hợp lệ.
- **Quay lui:** Duyệt qua từng tác vụ theo thứ tự đã sắp xếp. Tại mỗi bước i , ta xét 2 trạng thái:
 1. Chọn tác vụ i : Nếu thời gian hiện tại cộng thời gian xử lý r_i nhỏ hơn hoặc bằng t_i .
 2. Không chọn tác vụ i .
- **Nhánh cận (Pruning):** Tại mỗi bước, ta tính tổng lợi nhuận hiện tại cộng với tổng lợi nhuận của tất cả các tác vụ còn lại (hậu tố). Nếu giá trị này không vượt qua lợi nhuận kỷ lục (best profit) đã tìm được trước đó, ta cắt bỏ nhánh này để tránh duyệt vô ích.

1.2 Tính phù hợp của phương pháp

- Với $N \leq 20$, không gian trạng thái tối đa là $2^{20} \approx 10^6$. Đây là con số nhỏ, hoàn toàn có thể duyệt hết trong thời gian giới hạn.
- Yêu cầu của Subtask 1 là tìm nghiệm tối ưu tuyệt đối. Quay lui vét cạn là phương pháp chắc chắn nhất để đảm bảo không bỏ sót nghiệm tốt nhất.
- Kỹ thuật nhánh cận giúp giảm đáng kể số lượng trạng thái phải duyệt trong thực tế so với lý thuyết $O(2^N)$.

1.3 Phân tích độ phức tạp

- **Độ phức tạp thời gian:** Trường hợp xấu nhất là $O(2^N)$. Tuy nhiên, nhờ sắp xếp EDD và nhánh cận, thời gian chạy thực tế nhanh hơn nhiều.
- **Độ phức tạp không gian:** $O(N)$ dùng cho ngăn xếp đệ quy (recursion stack) và mảng lưu vết đường đi.

2 Subtask 2: Tìm nghiệm gần đúng tốt nhất ($N \leq 500$)

2.1 Phương pháp thiết kế thuật toán

Phương pháp sử dụng: Heuristic tìm kiếm chùm (Beam Search) kết hợp Tối ưu hóa Pareto (Pareto Optimization/Dominance Pruning).

Nguyên lý và cách áp dụng:

- Bài toán này về bản chất có thể mô hình hóa bằng Quy hoạch động (Dynamic Programming), nhưng do miền giá trị thời gian quá lớn nên không thể dùng mảng DP thông thường.
- **Beam Search:** Thay vì mở rộng tất cả các trạng thái có thể (như BFS hay DP đầy đủ), tại mỗi bước xét tác vụ thứ i , thuật toán chỉ giữ lại K trạng thái tiềm năng nhất (gọi là Beam Width).
- **Trạng thái:** Mỗi trạng thái được biểu diễn bởi cặp (L_i _nhun, $Thi_gian_O_dng$).
- **Lọc Pareto (Dominance Pruning):** Giữa các trạng thái trong "chùm"(beam), nếu trạng thái A có lợi nhuận cao hơn trạng thái B nhưng thời gian sử dụng lại ít hơn (hoặc bằng), ta nói A trội hơn B. Trạng thái B sẽ bị loại bỏ ngay lập tức vì không có khả năng tạo ra kết quả tốt hơn A.

2.2 Tính phù hợp của phương pháp

- **Lý do chọn Beam Search:** Với $N = 500$, độ phức tạp mũ 2^N là bất khả thi. Các thuật toán Greedy đơn thuần (như chỉ chọn việc có lợi nhuận cao nhất) thường cho kết quả kém chính xác. Beam Search là sự cân bằng giữa vét cạn và tham lam, cho phép khám phá không gian lời giải rộng hơn Greedy nhưng vẫn kiểm soát được thời gian chạy.
- **Cấu hình tham số:**
 - **Tên thuật toán:** Beam Search + Pareto Pruning.
 - **Beam Width (K):** 3000.
 - **Lý do chọn tham số:** Với giới hạn thời gian 5 giây cho $N = 500$, việc duy trì 3000 trạng thái tốt nhất mỗi bước đảm bảo thuật toán chạy trong khoảng 1-2 giây (an toàn) đồng thời độ phủ của không gian tìm kiếm đủ lớn để tìm ra nghiệm rất gần hoặc bằng nghiệm tối ưu.

2.3 Phân tích độ phức tạp

- **Độ phức tạp thời gian:** $O(N \cdot K \log K)$.

Tại mỗi bước trong N bước, ta mở rộng các trạng thái, sau đó sắp xếp để lọc Pareto và chọn ra K trạng thái tốt nhất. Việc sắp xếp mất $K \log K$. Với $N = 500$, $K = 3000$, số phép tính khoảng $500 \cdot 3000 \cdot 11 \approx 1.6 \cdot 10^7$ phép tính, rất nhanh so với giới hạn 5 giây.

- **Độ phức tạp không gian:** $O(K)$. Ta chỉ cần lưu trữ danh sách K trạng thái của bước hiện tại và bước kế tiếp.