BÁO CÁO GIẢI THUẬT CS112

Bài toán: A. Fucs và gánh nặng mưu sinh

Nhóm 10

Đoàn Hồng Bảo - 24520005
Lê Phạm Thành Nhân - 24520022

Ngày 16 tháng 12 năm 2025

Giới thiệu bài toán

Bài toán yêu cầu lựa chọn và sắp xếp một tập hợp các tác vụ N, mỗi tác vụ có thời gian thực hiện r_i, hạn chót t_i và lợi nhuận p_i. Mục tiêu là tối đa hóa tổng lợi nhuận sao cho các tác vụ được chọn hoàn thành trước deadline. Bài toán được chia làm 2 Subtask với quy mô dữ liệu khác nhau, yêu cầu các phương pháp tiếp cận riêng biệt.

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

1.1 1. Phương pháp thiết kế thuật toán

Phương pháp sử dụng: Vét cạn (Brute Force) kết hợp tham lam (Greedy) trên tập con.

Nguyên lý hoạt động: Vì N rất nhỏ ($N \leq 20$), ta có thể duyệt qua tất cả các tập con các tác vụ có thể chọn. Tuy nhiên, với một tập hợp các tác vụ đã chọn, thứ tự thực hiện tối ưu nhất để kiểm tra tính khả thi là sắp xếp theo hạn chót (Deadline) tăng dần (nguyên lý EDD - Earliest Due Date).

Các bước thực hiện:

1. Sử dụng một dãy bit (bitmask) từ 0 đến $2^N - 1$ để đại diện cho tất cả các tập hợp con của tác vụ. Nếu bit thứ i bật, tác vụ i được chọn.
2. Với mỗi tập con được chọn:
   - Sắp xếp các tác vụ trong tập con theo thứ tự t_i (deadline) tăng dần.
   - Duyệt qua các tác vụ đã sắp xếp, tính tổng thời gian tích lũy. Nếu tại bất kỳ thời điểm nào tổng thời gian vượt quá t_i của tác vụ hiện tại, tập con đó được coi là không hợp lệ.
3. Nếu tập con hợp lệ, tính tổng lợi nhuận $\sum p_i$. Cập nhật giá trị lợi nhuận lớn nhất (Max Profit) và lưu lại cấu hình.

1.2 2. Tính phù hợp của phương pháp

Phương pháp Vét cạn phù hợp cho Subtask 1 vì:

- **Đảm bảo tính tối ưu toàn cục:** Yêu cầu của Subtask 1 là tìm ra nghiệm tối ưu tuyệt đối. Vét cạn đảm bảo không bỏ sót bất kỳ trường hợp nào.
- **Kích thước dữ liệu nhỏ:** Với $N \leq 20$, số lượng tập con là $2^{20} \approx 10^6$. Máy tính hiện đại có thể xử lý số lượng phép tính này trong thời gian cho phép (5 giây).
- **Đơn giản và chính xác:** Tránh được các sai số có thể xảy ra khi dùng các thuật toán tham lam thuần túy hoặc heuristic.

1.3 3. Phân tích độ phức tạp

- **Độ phức tạp thời gian:** Có 2^N tập con. Với mỗi tập con kích thước tối đa N, ta cần sắp xếp mất $O(N \log N)$ và duyệt kiểm tra mất $O(N)$.

  $$T(N) = O(2^N \cdot N \log N)$$

  Với $N = 20$, $2^{20} \cdot 20 \cdot 4.3 \approx 9 \cdot 10^7$ phép tính, hoàn toàn khả thi trong 5s.
- **Độ phức tạp không gian:** $O(N)$ để lưu trữ mảng tác vụ và đệ quy (hoặc bitmask).

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

2.1 1. Phương pháp thiết kế thuật toán

Phương pháp sử dụng: Metaheuristic - Giải thuật Di truyền (Genetic Algorithm - GA).

Mô tả nguyên lý: Bài toán xếp lịch là một bài toán tối ưu hóa tổ hợp (Combinatorial Optimization). Với $N = 500$, không gian tìm kiếm là quá lớn (500!) để duyệt hết. Ta sử dụng GA để mô phỏng quá trình tiến hóa tự nhiên nhằm tìm kiếm lời giải tốt dần theo thời gian.

Cấu trúc giải thuật:

- **Mã hóa (Encoding):** Mỗi cá thể là một hoán vị của các chỉ số tác vụ $P = \{id_1, id_2, ..., id_N\}$.
- **Hàm thích nghi (Fitness Function):** Duyệt qua hoán vị P. Giữ một biến 'current_time'. Nếu 'current_time' + $r_i \leq t_i$, ta chọn tác vụ, cộng lợi nhuận và cập nhật 'current_time'. Nếu không, bỏ qua tác vụ. Fitness = Tổng lợi nhuận.
- **Khởi tạo quần thể:** Sinh ngẫu nhiên K hoán vị, trong đó có chèn 1 cá thể được tạo bởi thuật toán Greedy (xếp theo deadline tăng dần) để đảm bảo chất lượng ban đầu.
- **Toán tử lai ghép (Crossover):** Sử dụng lai ghép trật tự (Order Crossover - OX1) để giữ lại cấu trúc thứ tự tương đối của cha mẹ.
- **Toán tử đột biến (Mutation):** Swap Mutation (chọn ngẫu nhiên 2 vị trí trong hoán vị và đổi chỗ cho nhau).

2.2 2. Tính phù hợp của phương pháp

Tại sao chọn **Genetic Algorithm (GA)** cho Subtask 2?

- **Không gian tìm kiếm lớn:** $N = 500$ khiến các phương pháp chính xác (DP, Backtracking) bị quá tải bộ nhớ hoặc thời gian ($O(N \cdot \max(T))$ là không thể vì $T \leq 10^{12}$).
- **Tránh bẫy cục bộ:** Các thuật toán Greedy đơn giản (như xếp theo p_i giảm dần hoặc t_i tăng dần) thường bị kẹt ở nghiệm cục bộ kém. GA duy trì một quần thể đa dạng, giúp khám phá không gian lời giải tốt hơn.
- **Cơ chế tính điểm:** Subtask 2 chấm điểm dựa trên độ lệch so với Jury. GA có khả năng hội tụ về các nghiệm rất sát với tối ưu sau đủ số vòng lặp.

Cấu hình tham số:

- **Population Size (Kích thước quần thể):** 100. (Đủ lớn để duy trì sự đa dạng nhưng không quá chậm).
- **Generations (Số thế hệ):** 1000 - 2000 (Tùy thuộc vào thời gian chạy thực tế so với giới hạn 5s).
- **Mutation Rate (Tỷ lệ đột biến):** 0.1 (10%). Giúp tạo ra các biến dị nhỏ để thoát khỏi cực trị địa phương mà không phá vỡ cấu trúc tốt quá nhiều.

- **Elitism:** Giữ lại top 10% cá thể tốt nhất sang thế hệ sau không qua lai ghép để đảm bảo kết quả không bị thoái hóa.

2.3 3. Phân tích độ phức tạp

- **Độ phức tạp thời gian:** Gọi G là số thế hệ, M là kích thước quần thể. Trong mỗi thế hệ, ta thực hiện lai ghép, đột biến và tính Fitness (duyệt $O(N)$).

$$T(N) \approx O(G \cdot M \cdot N)$$

  Với $G = 2000, M = 100, N = 500 \rightarrow \approx 10^8$ phép tính, phù hợp với giới hạn 5s.

- **Độ phức tạp không gian:** $O(M \cdot N)$ để lưu trữ quần thể gồm M hoán vị độ dài N.

Kết luận

Nhóm 10 đã trình bày giải pháp toàn diện cho bài toán. Subtask 1 được giải quyết triệt để bằng thuật toán Vét cạn, đảm bảo điểm tối đa. Subtask 2 sử dụng Genetic Algorithm để tìm kiếm lời giải gần đúng chất lượng cao trong thời gian giới hạn, phù hợp với tính chất NP-hard của bài toán lập lịch trên quy mô lớn.