TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO HỆ HỖ TRỢ QUYẾT ĐỊNH

# PHÂN TÍCH TỐI ƯU

*Nhóm 8* Nguyễn Đức Vượng 20173603
Nguyễn Đức Hùng 20173520

Hà Nội, 2021

# Contents

# Lời nói đầu

Quy hoạch toán học là một họ công cụ được thiết kế để giúp giải quyết các vấn đề quản lý, trong đó người ra quyết định phải phân bổ các nguồn lực khan hiếm giữa các hoạt động để tối ưu hóa một mục tiêu có thể đo lường được.

Ví dụ, phân phối thời gian máy (tài nguyên) giữa các sản phẩm khác nhau (các hoạt động) là một vấn đề phân bổ điển hình. Các vấn đề phân bổ Quy hoạch tuyến tính (QHTT) thường có các đặc điểm sau:

- Có thể phân bổ một lượng hạn chế các nguồn lực kinh tế.
- Các nguồn lực được sử dụng để sản xuất các sản phẩm hoặc dịch vụ.
- Có hai hoặc nhiều cách để sử dụng các nguồn lực. Mỗi cái được gọi là giải pháp hoặc một chương trình.
- Mỗi hoạt động (sản xuất hoặc dịch vụ) trong đó các nguồn lực được sử dụng mang lại lợi nhuận dựa trên mục tiêu đã nêu.
- Việc phân bổ thường bị hạn chế bởi một số giới hạn và yêu cầu, được gọi là những ràng buộc.

Mô hình phân bổ QHTT dựa trên các giả định kinh tế sau:

- Có thể so sánh lợi nhuận từ các lần phân bổ khác nhau; nghĩa là chúng có thể được đo lường bằng một đơn vị chung (ví dụ: đô la, tiện ích).
- Lợi nhuận từ bất kỳ phân bổ nào là độc lập với các phân bổ khác.
- Tổng lợi nhuận là tổng lợi nhuận thu được từ các hoạt động khác nhau.
- Tất cả dữ liệu được nắm biết một cách chắc chắn.
- Các nguồn lực phải được sử dụng một cách tiết kiệm nhất.

Các vấn đề về phân bổ thường có một số lượng lớn các giải pháp khả thi. Dựa trên các giả thiết cơ bản, số lượng nghiệm có thể là vô hạn hoặc hữu hạn. Thông thường, các giải pháp khác nhau mang lại phần thưởng khác nhau. Trong số các giải pháp hiện có, ít nhất một giải pháp là tốt nhất, theo nghĩa là mức độ đạt

được mục tiêu đi kèm với nó là cao nhất (tức là tổng phần thưởng được tối đa hóa). Đây được gọi là một giải pháp tối ưu và nó có thể được tìm thấy bằng cách sử dụng một thuật toán đặc biệt.

# 1 Quy hoạch tuyến tính

Quy hoạch tuyến tính (QHTT) là công cụ nổi tiếng nhất trong họ các công cụ tối ưu hóa được gọi là quy hoạch toán học. Trong QHTT, tất cả các mối quan hệ giữa các biến là tuyến tính. Trong lĩnh vực kinh doanh và quản lý, những người ra quyết định thường phải đối mặt với vấn đề phân bổ nguồn lực hạn chế giữa các nhu cầu cạnh tranh để tối đa hóa sản lượng, doanh thu, mức độ dịch vụ, v.v. hoặc để giảm thiểu chi phí. Ví dụ, trong lập kế hoạch sản xuất, phối trộn nguyên liệu, quản lý danh mục đầu tư và lập kế hoạch cho nhân viên. QHTT cung cấp một công cụ hỗ trợ quyết định, toàn diện để tính toán một giải pháp tổng thể tốt nhất hoặc tối ưu.

Xét một ví dụ thực tế như sau:

Unimow sản xuất hai loại động cơ máy cắt cỏ James và Roker. Năng lực lao động, phay và tiện là những chỉ những ràng buộc về sản xuất. Để sản xuất mỗi James cần sử dụng ba giờ lao động, ba giờ phay và hai giờ tiện, mỗi Roker sử dụng hai giờ lao động, một giờ phay và bốn giờ tiện. James đóng góp cho đơn vị (trước khi phân bổ tổng chi phí cố định) là £ 22, Roker là £ 20. Trong một tháng bất kỳ, có 120.000 giờ nhân công có sẵn, thời gian phay 120.000 giờ và công suất tiện 150.000 giờ. Làm sao công ty có thể tối đa hóa lợi nhuận?

Trước tiên, cần mô hình hóa bài toán về dạng đại số, quá trình này gồm 4 bước

1. **Bước 1: Xác định vấn đề.** Ở đây, vấn đề cần giải quyết là tối đa hóa lợi nhuận.

2. **Bước 2: Xác định các biến.** Ở bước này, những người ra quyết định cần phải quyết định có những biến nào và ảnh hưởng như nào đến giải pháp. Trong ví dụ này chỉ có hai biến, số lượng James sản xuất và tương ứng số lượng Roker được sản xuất mỗi tháng, gọi là $J$ và $R$.

3. **Bước 3: Hình thành hàm mục tiêu.** Hàm mục tiêu là một phương trình biểu thị cách thức mà các biến quyết định ảnh hưởng đến giải pháp và thường được kết hợp với các từ tối đa hóa hoặc tối thiểu hóa. Hàm mục tiêu trong ví dụ này là

$$\max \ 22J + 20R$$

4. **Bước 4: Xác định các ràng buộc.** Giai đoạn cuối cùng là xác định tất cả các ràng buộc trong điều kiện về số lượng có thể được sản xuất. Trong ví dụ này,

$$\begin{array}{rrcrcl} \textbf{lao động} & 3J & + & 2R & \leq & 120 \\ \textbf{phay} & 3J & + & 1.5R & \leq & 120 \\ \textbf{tiện} & 2J & + & 4R & \leq & 150 \\ & & & J, R & \geq & 0 \end{array}$$

Có bốn phương pháp để giải quyết vấn đề:

1. **Mô hình "What if?"**, sử dụng phần mềm Microsoft Excel Solver. Mô hình này có thể rất mạnh mẽ và đơn giản để xây dựng nhưng dựa vào quyết định nhà sản xuất để đi đến câu trả lời chính xác. Bản thân mô hình không mang tính quy định trong việc cung cấp câu trả lời tốt nhất nhưng cho phép nhà quyết định xác định chiến lược nào là khả thi.

2. **Phương pháp hình học** sử dụng đồ thị, nhưng phương pháp này chỉ có thể được sử dụng để giải quyết các bài toán đơn giản chỉ với hai biến và hạn chế số lượng ràng buộc.

3. **Thuật toán Đơn hình** hoặc một trong những biến thể của nó, một công cụ mạnh mẽ, nhưng một đòi hỏi một khối lượng lớn các phép tính. Bất kỳ số lượng biến nào có thể là đã xử lý nhưng số lượng phép tính nhanh chóng trở nên không thể thực hiện được.

4. Sử dụng các gói phần mềm chuyên dụng, thường không thân thiện với người dùng, tốn kém và khó diễn giải. Nhưng sẽ không giới hạn về số lượng các biến có thể được xử lý.

# 2 Quy hoạch nguyên

Một quy hoạch tuyến tính nguyên là một bài toán quy hoạch tuyến tính mà trong đó một phần hay toàn bộ các biến số mục tiêu có điều kiện nguyên. Trong nhiều trường hợp, quy hoạch tuyến tính nguyên được nói tắt là quy hoạch nguyên. Bài toán quy hoạch nguyên với toàn bộ các biến mục tiêu là nguyên được gọi là bài toán quy hoạch nguyên hoàn toàn, ngược lại thì được gọi là bài toán quy hoạch nguyên bộ phận.

Một mô hình quy hoạch nguyên có dạng

$$\begin{array}{rrcrcccrcl} \min & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & & \\ & a_{1,1}x_1 & + & a_{1,2}x_2 & + & \cdots & + & a_{1,n}x_n & \leq & b_1 \\ & \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ & a_{m,1}x_1 & + & a_{m,2}x_2 & + & \cdots & + & a_{m,n}x_n & \leq & b_m \\ & & & & & & & x_1, x_2, \ldots, x_n & \in & \mathbb{Z}_+ \end{array}$$

Trong đó, các số $b_1, b_2, \ldots, b_m$ là các số không âm.

Trong việc hỗ trợ đưa ra quyết định, bài toán quy hoạch nguyên thường được dùng trong việc phân chia tài nguyên vào các công việc khác nhau. Trong đó các biến $x_1, x_2, \ldots, x_n$ là các biến quyết định, các bất phương trình $a$

$$a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,n}x_n \leq b_i$$

là các ràng buộc thể hiện những điều kiện thực tế không điều khiển được như công nghệ, tài chính, nhân lực, etc. Một cách lý thuyết, những bài toán quy hoạch nguyên dưới dạng này luôn có nghiệm. Bài toán quy hoạch nguyên có một số phương pháp giải như nhánh cận, hình học...

**Ví dụ 1.** Một công ty ở có dự định xây thêm một vài phân xưởng mới ở hai thành phố A và B để gia tăng sản xuất. Ước tính mỗi xưởng ở thành phố A sẽ đem lại lợi nhuận 70 000VNĐ mỗi ngày, ở thành phố B là 60 000VNĐ mỗi ngày. Xây dựng một xưởng ở thành phố A sẽ tốn 60 000 000 VNĐ và ở thành phố B sẽ tốn 50 000 000VNĐ. Tổng số tiền cho xây dựng thêm là 300 000 000VNĐ. Vậy ở mỗi thành phố nên xây dựng thêm bao nhiêu phân xưởng?

Các biến quyết định ở đây là số xưởng cần xây, do đó đặt số xưởng sẽ xây

dựng ở thành phố A là $x_1$, ở thành phố B là $x_2$. Quy đơn vị tiền mặt sang nghìn VNĐ. Số tiền lợi nhuận cho một ngày sẽ là

$$70x_1 + 60x_2.$$

Ta mong muốn lợi nhuận càng cao càng tốt, do đó $\max 70x_1 + 60x_2$ sẽ là hàm mục tiêu của chúng ta. Với điều kiện chi phí xây dựng ở mỗi thành phố và tổng số tiền, ta có ràng buộc:

$$60\,000x_1 + 50\,000x_2 \leq 300\,000.$$

Điều kiện này có thể rút gọn lại là:

$$6x_1 + 5x_2 \leq 30.$$

Trong đó theo lẽ thường, các số xưởng được xây–các biến $x_1$ và $x_2$ phải là số nguyên, ta có thêm ràng buộc $x_1, x_2$ nguyên. Vậy quy hoạch nguyên mô hình hóa bài toán này là:

$$\begin{array}{rrrrl} \max & 70x_1 & + & 60x_2 & \\ & 6x_1 & + & 5x_2 & \leq 30\ . \\ & & & x_1, x_2 & \text{nguyên} \end{array}$$

Sau khi giải bài toán này, chúng ta sẽ thu được số phân xưởng cần xây dựng để sinh lời lớn nhất, thỏa mãn với những ràng buộc về điều kiện tài chính hiện tại.

# 3 Tối ưu tổ hợp

Tối ưu tổ hợp là một nhánh của tối ưu. Những bài toán tối ưu tổ hợp nằm dưới dạng: tìm một lời giải tối ưu trong tập hữu hạn những lời giải. "Lời giải" ở đây được hiểu theo nghĩa rất chung, mỗi bài toán sẽ có một định nghĩa các lời giải riêng. Một số ví dụ về tối ưu tổi hợp là:

- Bài toán người đi du lịch: cho một danh sách các thành phố và khoảng cách của mỗi cặp trong số các thành phố đó; một người muốn đi qua các thành phố này, mỗi thành phố đi một lần và quay lại thành phố ban đầu; vậy người đó nên đi qua các thành phố theo thứ tự nào để đoạn đường đi qua là ngắn nhất? Các lời giải ở đây là thứ tự đi qua mỗi thành phố.

- Bài toán cái túi: một người cần cho một số lượng đồ tư trang vào một túi hành lý, mỗi vật có một trọng lượng nào đó, nên mang đồ vật như thế nào để vừa thỏa mãn sử dụng cá nhân, vừa không quá sức chứa của túi hành lý. Các lời giải ở đây là cách thức người đó xếp đồ vào hành lý (e.g. xếp những thứ gì, số lượng bao nhiêu).

Mỗi bài toán tổ hợp sẽ có một bài toán phụ đi kèm gọi là *bài toán quyết định*: cho một lời giải $y$ của bài toán tổ hợp, hỏi lời giải đó có chấp nhận được không. Thông thường, những bài toán tối ưu tổ hợp có tập lời giải rất lớn, do đó việc lặp qua từng phần tử để đánh giá và lựa chọn hoặc không khả thi, hoặc rất tốn kém và không hiệu quả.

Tối ưu tổ hợp có nhiều ứng dụng trong thực tế. Một dạng bài toán có thể được áp dụng cho nhiều vấn đề khác nhau với cùng dạng đó. Phần này liệt kê một số dạng tối ưu tổ hợp mà hữu dụng trong việc hỗ trợ quyết định.

## 3.1 Bài toán vận tải

Bài toán vận tài là bài toán nghiên cứu tối ưu việc vận chuyển và phân bố tài nguyên. Bài toán vận tải có thể được mô hình hóa như một bài toán nguyên. Ở dạng đơn giản nhất, bài toán vận tải được mô tả như sau:

- Một công ty cần vận chuyển một loại hàng từ $m$ điểm phát tới $n$ điểm thu

– $m$ điểm phát có lượng cung tương ứng là $a_1, a_2, \ldots, a_m$, $n$ điểm thu có lượng cầu tương ứng là $b_1, b_2, \ldots, b_n$

– Hàng hóa được phân phối từ mỗi điểm phát đến mỗi điểm thu tùy ý.

– Một điểm thu có thể nhận hàng tại một điểm phát bất kì.

– Các điểm thu phải thu đủ số hàng.

– Các điểm phát phải phát đủ số hàng.

– Hàng đã được nhận rồi thì không thể được trả lại.

Ký hiệu $x_{i,j}$ là lượng hàng vận chuyển và $c_{i,j}$ là cước phí vận chuyển một đơn vị hàng từ điểm phát $i$ tới điểm thu $j$. Bài toán vận tải sẽ được mô hình hóa như sau:

$$\begin{aligned} \min f(x) &= \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} \\ \text{v.đ.k } \sum_{j=1}^{n} x_{ij} &= a_i \quad i = \overline{1,m} \\ \sum_{i=1}^{m} x_{ij} &= b_j \quad j = \overline{1,n} \\ x_{ij} &\geq 0 \quad i = \overline{1,m}, j = \overline{1,n}. \end{aligned}$$

Nghiệm của bài toán sẽ là một ma trận phân phát, cho biết những cơ sở sản xuất nào nên phân phát hàng tới điểm thu nào, và với trữ lượng bao nhiêu. Ngoài việc sử dụng các thuật toán quy hoạch tuyến tính ra, thuật toán chuyên biệt như thuật toán thế vị có thể được sử dụng để tìm nghiệm nhanh chóng hơn.

Điều kiện để bài toán vận tải có nghiệm là tổng trữ lượng ở điểm phát và tổng nhu cầu ở điểm thu phải bằng nhau. Trong thực tế điều này không phải lúc nào cũng thỏa mãn. Giả sử trong trường hợp tổng lượng hàng cần vận chuyển bé hơn nhu cầu, chúng ta xử lý bằng cách tạo ra một điểm phát "giả" và giải bài toán như bình thường. Trong trường hợp này, khi giải ra nghiệm, điểm thu mà nhận hàng từ điểm phát "ảo" sẽ bị thiếu hàng so với nhu cầu thực tế.

Ngoài dạng cơ bản ra, bài toán vận tải còn có những dạng biến thể khác, phục vụ cho những bài toán khác mà không phải phân phối hàng (e.g. bài toán phân công công việc), hoặc cho các điều kiện khác trong thực tế, ví dụ như bài toán vận tải dạng max–chúng ta không tối ưu chi phí, mà là lợi nhuận, bài toán vận tải có ô cấm–một số điểm phát hàng không thể vận chuyển tới một số điểm thu nào đó...

## 3.2 Bài toán luồng cực đại

Bài toán luồng cực đại trên đồ thị là bài toán mà trong đó, chúng ta có một mạng lưới mà tài nguyên lưu thông từ một điểm nguồn tới một điểm đích. Trên mạng lưới này, đường thông qua từ một nút này tới một nút khác gọi là cung, khả năng thông qua của mỗi cung này đều đã được biết. Một luồng trên mạng có thể hiểu là một cách cấu hình lượng tài nguyên đi qua trên mỗi cung. Ta cần tìm một luồng trên mạng lưới này sao cho khả năng tài nguyên thông qua mạng này là lớn nhất.

Một cách hình thức, bài toán luồng cực đại được mô hình hóa như sau: Cho $N$ là một mạng lưới với $s$, $t$ lần lượt là điểm phát và điểm thu, $V$ là tập các điểm và $E$ là tập các cung. Xét điểm $u, v$ bất kì thuộc $V$. Mỗi cung $e = (u, v) \in E$ trên mạng lưới có giới hạn lưu lượng là số thực $c_{u,v}$. Một luồng trên mạng là một hàm $f(e)$ đo giá trị của luồng qua cung đó. Ta xét một luồng cực đại hay không căn cứ vào giá trị của luồng, được tính bởi.

$$|f| = \sum_{v\colon (s,v)\in E} f((s,v)),$$

tức tổng lưu lượng từ điểm phát tới điểm thu. Vì lưu lượng trên mỗi cung không thể vượt quá giới hạn, ta có ràng buộc

$$f((u,v)) \le c_{u,v}, \qquad \text{với mọi } (u,v) \in E.$$

Vì mỗi điểm trung gian không thể tồn lại bất kỳ tài nguyên nào, ta có thêm ràng buộc với mọi điểm $v$ trong $V$ mà không phải điểm thu hay điểm phát:

$$\forall v \in V \setminus \{s,t\} : \sum_{u\colon (u,v)\in E} f((u,v)) = \sum_{u\colon (v,u)\in E} f((v,u))$$

Vì bài toán luồng cực đại cũng là một quy hoạch tuyến tính, ta cũng có thể giải bằng các công cụ quy hoạch tuyến tính. Tuy nhiên có các phương pháp khác được thiết kế riêng cho bài toán như phương pháp đường tăng luồng, phương pháp Ford–Fulkerson...

Trong thực tế, bài toán luồng cực đại có rất nhiều ứng dụng. Đôi lúc, chúng ta có nhiều điểm phát và nhiều điểm thu, khi đó chỉ cần thêm một điểm phát và điểm thu giả để thay thế cho tất cả những điểm phát, điểm thu là ta lại có bài toán gốc. Những cung nối với điểm thu/phát giả sẽ không có giới hạn lưu lượng. Ngoài ra cũng có thể có những biến thể khác sao cho phù hợp với bài toán thực tế (e.g. lập lịch cho các chuyến bay của các hãng hàng không).

## 3.3 Bài toán cắt vật liệu

Bài toán cắt vật liệu liên quan tới việc cắt các tấm vật liệu với kích cỡ nào đó, sao cho đủ yêu cầu và tối thiểu số vật liệu bị lãng phí. Đây là một bài toán xuất phát từ nhu cầu trong công nghiệp.

**Ví dụ 2.** Nhà sản xuất có các thanh vật liệu dài 2m, được yêu cầu sản xuất ra 1000 thanh vật liệu loại 0.9m,1200 thanh loại 0.5m, 1500 thanh loại ba 0.4m. Vấn đề đặt ra là cần một cách cắt sao cho tổng số vật liệu là nhỏ nhất.

Đặt $J$ là tập các mẫu cắt $j$, $Y_j$ là số lần cắt theo mẫu cắt $i$, $a_{i,j}$ là số lần cắt vật liệu $i$ theo mẫu $j$, bài toán này có thể được mô hình hóa như một bài toán quy hoạch nguyên:

$$\begin{aligned}
&\min \sum_j Y_j \\
&\sum_j a_{1,j} Y_j \geq 1000 \\
&\sum_j a_{2,j} Y \geq 1200 \\
&\sum_j a_{3,j} Y_j \geq 1500 \\
&Y_j \in \mathbb{Z}, Y_j \geq 0 \text{ với } j \in J
\end{aligned}$$

Tuy nhiên, khi giải những bài toán cắt vật bằng các phương pháp của quy hoạch nguyên, ta gặp một vấn đề, đó là tập mẫu cắt có thể rất lớn, khiến cho kích cỡ của bàn toán trở nên khổng lồ. Với những bài toán cắt vật liệu, có một phương pháp hiệu quả hơn, đó là phương pháp sinh cột. Trong phương pháp sinh cột, chúng ta sẽ bắt đầu giải bài toán với một tập con của tập mẫu cắt $J$. Ví dụ torng bài toán trên, tập mẫu cắt khởi tạo có thể là:

- Mẫu 1: cắt hai thanh vật liệu 0.9m
- Mẫu 2: cắt bốn thanh vật liệu 0.5m
- Mẫu 3: cắt năm thanh vật liệu 0.4m

Trong quá trình giải theo thuật toán, những mẫu cắt mới cần thiết sẽ được đưa vào, do đó cách tiếp cận này giải được bài toán mà không cần xét tới toàn bộ tập mẫu cắt. Kích cỡ của bài toán sẽ tăng dần lên sau mỗi bước sinh cột (sinh mẫu). Trong trường hợp tốt nhất, thuật toán sẽ tìm được nghiệm tối ưu ở bước đầu tiên, chúng ta có cách cắt tối ưu với ba mẫu đã chọn ban đầu.

Khi giải theo thuật toán sinh cột, tùy mỗi cách chọn mẫu đầu vào, thuật toán có thể tạo ra những phương án tối ưu khác nhau. Mặc dù mức sử dụng vật

liệu trên lý thuyết là giống nhau, những phương án tối ưu này sẽ tạo ra các dư thừa vật liệu khác nhau. Lượng dư thừa của vật liệu $i$–khi có nghiệm của bài toán–có thể dễ dàng tính bằng công thức $\sum_j Y_j a_{i,j} - n_j$. Nhà quản lý khi ứng dụng bài toán cắt vật liệu có thể cân nhắc chọn phương án phù hợp với chiến lược của mình.

# 4 Tối ưu đa mục tiêu

Tối ưu đa mục tiêu (MOP) là quy trình để tối ưu hóa hai hoặc nhiều mục tiêu đồng thời với một số ràng buộc nhất định. Về mặt toán học, bài toán MOO là bài toán tìm một vectơ của các biến quyết định thỏa mãn một số ràng buộc và tối ưu hóa một hàm vectơ có các phần tử đại diện cho các hàm mục tiêu được cực tiểu hoặc cực đại đồng thời. Mô hình bài toán: tìm vector biến

$$x = (x_1, \ldots, x_n) \in \mathbb{R}^n$$

sao cho tối đa/tối thiểu hàm mục tiêu

$$f(x) = (f_1(x), \ldots, f_m(x))^T$$

thỏa mãn

$$x_i^{lb} \leq x_i \leq x_i^{ub},\ i = 1, \ldots, n$$

$J$ ràng buộc dấu

$$g_j(x) \leq 0,\ j = 1, \ldots, J$$

$K$ ràng buộc đẳng thức

$$h_k(x) = 0,\ k = 1, \ldots, K$$

Giải một bài toán MOP là tìm ra một tập nghiệm được gọi là tập Pareto. Sau đó, chọn một nghiệm từ tập Pareto. Không giống như một bài toán tối ưu hóa một mục tiêu, việc giải quyết một bài toán MOP bao gồm ba bước quan trọng và hoàn toàn khác nhau:

1. Hình thành một mô hình toán học;
2. Tối ưu hóa;
3. Ra quyết định.

Trong bước tối ưu hóa, tập Pareto được xác định; Tuy nhiên, trong bước ra quyết định, nghiệm tốt nhất được chọn từ tập Pareto dựa trên nhu cầu của người ra quyết định.

Theo tiếp cận trên không gian quyết định (decision space), việc giải bài toán (MOP) được xem như việc xác định toàn bộ hay một phần của tập nghiệm hữu hiệu $X_E$ hoặc tập nghiệm hữu hiệu yếu $X_{WE}$. Đây là một việc khó, vì ngay cả trong trường hợp đơn giản nhất của (MOP) là bài toán quy hoạch đa mục tiêu tuyến tính (LMOP), tập nghiệm hữu hiệu $X_E$ và tập nghiệm hữu hiệu yếu $X_{WE}$ đã là các tập không lồi với cấu trúc rất phức tạp.

Với hy vọng giảm khối lượng tính toán, các thuật toán theo hướng tiếp cận trên không gian ảnh hay không gian giá trị (outcome space) được thiết kế để xác định toàn bộ hay một phần của tập ảnh hữu hiệu $Y_E = f(X_E)$ hoặc tập ảnh hữu hiệu yếu $Y_{WE} = f(X_{WE})$. Lý do chính cho hướng tiếp cận này là: i) Các bài toán tối ưu đa mục tiêu nảy sinh trong thực tế thường có số hàm mục tiêu $p$ nhỏ hơn rất nhiều so với số biến $n$; ii) Tập ảnh hữu hiệu $Y_E$ có cấu trúc đơn giản hơn $X_E$; iii) Trong quá trình đưa ra quyết định, người ta thường lựa chọn phương án dựa trên giá trị hữu hiệu hơn là dựa trên nghiệm hữu hiệu.

Trong tối ưu đa mục tiêu, việc nghiên cứu để giải bài toán quy hoạch đa mục tiêu tuyến tính (LMOP) có thể xem gần như hoàn chỉnh. Rất nhiều thuật toán đã được đề xuất theo cả hai hướng tiếp cận trên không gian quyết định và không gian ảnh để giải bài toán này bằng nhiều phương pháp khác nhau như phương pháp đơn hình đa mục tiêu, phương pháp tham số, phương pháp vô hướng hóa, phương pháp nón pháp tuyến, phương pháp xấp xỉ ngoài hoặc kết hợp của các phương pháp đó. Với bài toán quy hoạch đa mục tiêu lồi và không lồi, đã có một số thuật toán được đề xuất. Hầu hết các thuật toán theo tiếp cận trên không gian quyết định được thiết kế dựa trên các phương pháp trọng số, phương pháp $\epsilon-$ràng buộc, phương pháp hàm lợi ích, phương pháp lexicographic, phương pháp Tchebycheff,... để sinh một phần tập nghiệm hữu hiệu hay hữu hiệu yếu của bài toán. Theo tiếp cận trên không gian ảnh, các thuật toán thường sử dụng kỹ thuật xấp xỉ ngoài để xây dựng một dãy các tập xấp xỉ tập ảnh, trong đó ta có thể dễ dàng xác định được tập hữu hiệu các tập xấp xỉ này. Với cách tiếp cận này, một mặt, thuật toán sinh ra một phần của tập ảnh hữu hiệu của bài toán, mặt khác, nó sinh ra tập xấp xỉ của tập ảnh hữu hiệu chứa toàn bộ tập ảnh hữu hiệu.

Khi đã tìm được tập nghiệm Pareto, việc xác định một nghiệm tối ưu dựa trên các yêu cầu chủ quan của DM cũng vô cùng quan trọng. Đó gọi là bài toán tối ưu trên tập nghiệm hữu hiệu, có mô hình như sau:

$$\min\ h(x) \text{ v.đ.k x}\in X_E$$

trong đó $h(x)$ là một hàm số thực xác định và $X_E$ là tập nghiệm của bài toán quy hoạch đa mục tiêu (MOP).

Ví dụ, hãy xem xét một công ty tạo ra lợi nhuận. Ngoài việc kiếm tiền, công

ty muốn tăng trưởng, phát triển sản phẩm và nhân viên của mình, đảm bảo việc làm cho người lao động và phục vụ cộng đồng. Các nhà quản lý muốn làm hài lòng các cổ đông, đồng thời được hưởng mức lương và tài khoản chi phí cao, và nhân viên muốn tăng lương và phúc lợi của họ. Khi phải đưa ra một quyết định - ví dụ, về một dự án đầu tư - một số mục tiêu này bổ sung cho nhau, trong khi những mục tiêu khác thì khác. Kearns đã mô tả cách thức quy trình phân cấp phân tích (AHP), kết hợp với quy hoạch nguyên để giải quyết nhiều mục tiêu trong việc đánh giá đầu tư trong lĩnh vực IT.

Trong thực tế sẽ tồn tại những khó khăn nhất định có thể phát sinh khi tối ưu nhiều mục tiêu:

- Thông thường rất khó để có được một mô hình rõ ràng về các mục tiêu của tổ chức.
- Người ra quyết định có thể thay đổi tầm quan trọng được giao cho các mục tiêu cụ thể theo thời gian hoặc cho các tình huống quyết định khác nhau.
- Các mục tiêu chính và mục tiêu phụ được nhìn nhận khác nhau ở các cấp độ khác nhau của tổ chức và trong các phòng ban khác nhau.
- Các mục tiêu thay đổi để đáp ứng với những thay đổi trong tổ chức và môi trường của nó.
- Mối quan hệ giữa các lựa chọn thay thế và vai trò của chúng trong việc xác định mục tiêu có thể khó định lượng.
- Các vấn đề phức tạp được giải quyết bởi các nhóm người ra quyết định, mỗi người trong số họ có một hướng giải quyết khác nhau.
- Những người tham gia đánh giá tầm quan trọng (ưu tiên) của các mục tiêu khác nhau một cách khác nhau.

# 5 Vét cạn

Tìm kiếm vét cạn (complete search) là một phương pháp thông thường để giải hầu hết bất kỳ bài toán tối ưu nào. Ý tưởng là duyệt hết tất cả các lời giải có thể có của bài toán bằng cách sử dụng vét cạn (brute force), và sau đó lựa chọn một giải pháp tốt nhất. Tìm kiếm vét cạn là một kỹ thuật tốt nếu có đủ thời gian để đi qua hết tất cả các lời giải, vì việc tìm kiếm thường dễ để thực thi và nó luôn cho ra lời giải chính xác.

## 5.1 Sinh tập con

Có 2 phương pháp thông thường để tạo tập hợp con: chúng ta có thể thực hiện một tìm kiếm đệ quy hoặc khai thác cách biểu diễn bit của số nguyên.

## 5.2 Sinh các hoán vị

2 cách tiếp cận: chúng ta có thể dùng đệ quy hoặc duyệt qua các hoán vị lặp.

## 5.3 Quay lui

Thuật toán quay lui (backtracking) bắt đầu với một lời giải rỗng và mở rộng lời giải từng bước. Việc tìm kiếm đệ quy duyệt qua các cách khác nhau là cách để xây dựng nên lời giải.

**Ví dụ 3.** xem xét bài toán tính số cách đặt n con hậu lên bàn cờ n x n sao cho không có con cậu nào tấn công lẫn nhau.

Bài toán có thể được giải bằng cách sử dụng quay lui bởi việc đặt con hậu lên bàn cờ theo từng hàng. Chính xác hơn, một con hậu sẽ được đặt lên mỗi hàng sao cho không có con hậu nào tấn công bất kỳ con hậu nào được đặt trước đó. Một lời giải được tìm thấy khi tất cả n quân hậu được đặt trên bàn cờ.

## 5.4 Tìm kiếm nhánh cận

húng ta thường có thể tối ưu quay lui bằng cách cắt tỉa cây tìm kiếm. Ý tưởng là thêm 1 chút thông minh vào thuật toán để nó nhận ra càng sớm càng tốt nếu một phần lời giải không thể được mở rộng đến lời giải hoàn chỉnh. Việc tối ưu có ảnh hưởng rất lớn đến hiệu quả của việc tìm kiếm.

Chúng ta cùng xem xét bài toán tính số lượng con đường trên một lưới n x n từ góc trên bên trái đến góc dưới bên phải sao cho con đường viến thăm mỗi ô vuông duy nhất 1 lần. Ví dụ, trên lưới 7x7. Chúng ta tập trung vào trường hợp 7x7, bởi vì mức độ khó của nó là phù hợp với nhu cầu của chúng ta. Chúng ta bắt đầu với một thuật toán quay lui đơn giản, và sau đó tối ưu nó từng bước bằng cách sử dụng quan sát, việc tìm kiếm có thể được cắt tỉa.

### 5.4.1 Thuật toán cơ bản

Phiên bản đầu tiên của thuật toán không chứa bất kỳ tối ưu nào. Đơn giản, chúng ta sử dụng quay lui để tạo ra tất cả các con đường từ góc trên bên trái đến góc dưới bên phải và đếm số lượng các con đường như vậy.

### 5.4.2 Tối ưu 1

Trong bất kỳ lời giải nào, ở bước đầu tiên chúng ta di chuyển xuống hoặc qua phải. Luôn có 2 con đường đối xứng qua đường chéo của lưới sau bước đầu tiên. Do đó, chúng ta có thể quyết định rằng chúng ta bước đầu tiên chúng ta luôn di chuyển xuống dưới (hoặc phải), và cuối cùng nhân với 2 thì ra số lượng lời giải.

### 5.4.3 Tối ưu 2

Nếu con đường dẫn đến ô dưới cùng bên phải trước khi nó viếng thăm tất cả các ô vuông khác của lưới, thì đó rõ ràng không thể mang lại một lời giải hoàn chỉnh. Sử dụng quan sát này, chúng ta kết thúc việc tìm kiếm ngay lập tức nếu chúng ta tìm đến ô dưới cùng bên phải quá sớm.

### 5.4.4 Tối ưu 3

Nếu con đường chạm vào bước tường và chỉ có thể xoay qua bên trái hoặc bên phải thì lưới bị chia thành 2 phần chứa các ô chưa được đi tới. Trong trường hợp này, chúng ta không thể viếng thăm tất cả các ô, vì vậy chúng ta kết thúc việc tìm kiếm.

### 5.4.5 Tối ưu 4

Ý tưởng ở tối ưu 3 có thể được tổng quát: Nếu con đường không thể tiếp tục tiến tới mà chỉ có thể rẽ trái hoặc phải, thì lưới sẽ bị chia thành 2 phần mà cả hai phần có chứa những ô chưa được viếng thăm. Rõ ràng chúng ta không thể viếng thăm tất cả các ô được, vì thế chúng ta có thể kết thúc sớm việc tìm kiếm.

## 5.5 Chia để trị

Chia để trị là một kỹ thuật mà không gian tìm kiếm được chia thành 2 phần về kích thước. Một tìm kiếm riêng biệt được thực hiện cho cả 2 phần, và kết quả cuối cùng của việc tìm kiếm được kết hợp lại.

Kỹ thuật này có thể được sử dụng nếu có một cách hiệu quả để kết hợp các kết quả tìm kiếm lại. Trong tình huống này, việc tìm kiếm song song có thể yêu cầu ít thời gian hơn trong một không gian tìm kiếm lớn. Thường, chúng ta có thể chuyển nhân tử $2^n$ thành $2^{n/2}$ khi sử dụng kỹ thuật chia để trị.

## 5.6 Kết luận

Tìm kiếm vét cạn không phải là một giải pháp tối ưu để giải quyết một bài toán, nhưng nó luôn là 1 cách tiếp cận dễ để cho ta 1 giải pháp ban đầu đúng đắn. Từ lời giải này ta có thể áp dụng các kỹ thuật như tìm kiếm nhánh cận hoặc chia để trị để tối ưu dần.

# 6 Độ phức tạp

Bài toán quyết định là một trong những đối tượng trọng tâm nghiên cứu của lý thuyết độ phức tạp tính toán.

## 6.1 Đánh giá thời gian thực hiện thuật giải:

### 6.1.1 Tính độc lập:

Thế nào là một thuật giải nhanh. Có thể lập chương trình, chạy máy rồi bấm giờ. Tuy nhiên tốc độ thực hiện một chương trình phụ thuộc vào ngôn ngữ lập trình, chương trình dịch, hệ điều hành, phần cứng của máy... Mặt khác, phải lập trình mới đo được thời gian thực hiện của thuật giải.

Cần đánh giá thời thực hiện sao cho:

- Không phụ thuộc máy, ngôn ngữ lập trình, chương trình biên dịch.
- Không cần phải triển khai chương trình thực hiện thuật giải.
- Chỉ dựa vào bản thân thuật giải.

### 6.1.2 Các phép toán sơ cấp:

Trước hết ta cần thống nhất những thao tác nào được coi là một phép tính.

Đây là khái niện phép toán sơ cấp. Các phép toán sơ cấp là những phép toán mà thời gian thực hiện nó đủ ngắn, hay nói đúng hơn là không vượt quá một hằng số nào đó. Các phép toán sau đây có thể coi là sơ cấp:

- Các phép tính số học.
- Các phép tính logic.
- Các phép chuyển chỗ, gán...

### 6.1.3 Kích thước dữ liệu đầu vào:

Cho một thuật giải ta hoàn toàn ước lượng được tổng số các phép toán sơ cấp cần thiết để thực hiện thuật giải đó. Một điều hiển nhiên là tổng số phép toán sơ cấp để giải một bài toán phụ thuộc vào kích thước của bài toán. Dùng cùng một thuật toán, tính một định thức cấp 5 rõ ràng cần ít phép tính hơn định thức

cấp 10. Tổng số mục dữ liệu đầu vào là đặc trưng cho kích thước của bài toán. Người ta thường dùng một số nguyên dương $n$ để thể hiện kích thước này.

Như vậy, một thuật giải $T$ áp dụng để giải bài toán có kích thước n sẽ cần một tổng số $T(n)$ các phép toán sơ cấp. $T(n)$T(n) là một hàm của tham số $n$.

Hàm số $T(n)$ là đặc trưng cho hiệu quả của thuật giải $T$.

### 6.1.4 Tình trạng dữ liệu đầu vào:

Không chỉ có số lượng dữ liệu đầu vào quyết định thời gian thực hiện giải thuật mà tình trạng dữ liệu cũng ảnh hưởng đến việc thuật giải thực hiện nhanh hay chậm. Xét bài toán sắp xếp một dãy số. Rõ ràng là nếu dãy đã có sẵn thứ tự mong muốn hoặc gần thư thế thì công việc phải làm ít hơn trường hợp một dãy bất kỳ.

## 6.2 Ký hiệu O lớn

Định nghĩa: Giả sử $f(n)$, $g(n)$ là 2 hàm số không âm, đồng biến theo $n$. Ta nói "$f(n)$ là O lớn của $g(n)$" và viết: $f(n) = O(g(n))$ khi và chỉ khi tồn tại hằng số $C$ để $f(n) \leq C.g(n)$ kể từ $n \geq n_0$ nào đó.

Ta nói $f(n)$ có cấp lớn không vượt quá $g(n)$ (dễ hiểu là $f(n)$ có tăng tới đâu đi nữa cũng không thể vượt quá tốc độ tăng của $g(n)$).

Ví dụ: $f(n) = 2n^2 + 3n + 5$.

$f(n) \leq 2n^2 + 3n^2 + 5n^2 = 12n^2 \ \forall n \geq 1$. Ta viết $f(n) = O(n2)$

Viết $T(n) = O(g(n))$ nghĩa là tốc độ tăng của $T(n)$ khi tiến đến vô cùng không vượt quá tốc độ tăng của $g(n)$. Khi $n$ lớn, $g(n)$ cho ta hình dung được mức lớn của $T(n).g(n)$ là "thước đo" độ lớn của $T(n)$.

## 6.3 Ký hiệu $\Omega$ lớn và $\Theta$ lớn

Tương tự như với bậc big-O, nếu như tìm được các hằng số $C, k1, k2$ đều dương và không phụ thuộc vào $n$, sao cho với $n$ đủ lớn, các hàm $R(n), f(n)$ và $h(n)$ đều dương và

$$R(n) \geq C \cdot f(n)$$

$$k_1 \cdot h(n) \leq R(n) \leq k_2 \cdot h(n)$$

thì ta nói thuật toán có độ phức tạp cỡ lớn hơn $\Omega(n)$, và đúng bằng cỡ $\Theta(h(n))$.

Như vậy nếu xét một cách chặt chẽ, kí hiệu $\Theta$ mới biểu thị độ phức tạp của thuật toán một cách chặt chẽ. Do đó 2 ký hiệu thường được sử dụng trong đánh giá độ phức tạp của thuật toán là Big-O và Big-$\Theta$. Nhưng chúng ta thường sử dụng Big-O hơn.

# 7 Thuật toán xấp xỉ

Thuật toán xấp xỉ là những thuật toán với hiệu năng tốt hơn các thuật toán thông thường, để tìm nghiệm *gần đúng* cho các bài toán tối ưu. Những thuật toán có một đảm bảo nào đó, chứng minh được, về sai số giữa nghiệm gần tối ưu và nghiệm tối ưu thực sự. Những thuật toán này được thiết kế với phỏng đoán rằng $P \neq NP$, i.e. có một số lớp bài toán mà không thể giải được với thời gian đa thức.

Một số cách thiết kế thuật toán xấp xỉ bao gồm:

1. Thuật toán tham lam là dạng thuật toán mà chọn điểm tối ưu địa phương ở mỗi bước. E.g. trong bài toán cái túi, phương pháp tham lam là lấy những vật có giá trị cao và trọng lượng thấp đưa vào trước
2. Tìm kiếm địa phương: lặp qua các phương án từ một phương án ban đầu cho tới khi không tìm được phương án nào tốt hơn.
3. Quy hoạch động: chia bài toán thành những vấn đề nhỏ hơn, và giải một cách đệ quy.
4. Giải bài toán nguyên ở dạng nới lỏng để thu được nghiệm không nguyên, sau đó làm tròn một cách hợp lý.
5. Lấy mẫu ngẫu nhiên hoặc sử dụng tính ngẫu nhiên.

Chúng ta có thể chứng minh được một vài tính chất về thuật toán xấp xỉ. Một thuật toán xấp xỉ được gọi là thuật toán $\rho$–xấp xỉ nếu như với một đầu vào $x$ thì tỉ lệ $^{\text{giá trị}}/_{\text{chi phí}}$ của việc xấp xỉ sẽ bé hơn (hoặc không lớn hơn) $\rho$. Trong đó $\rho$ được gọi là *giá trị đảm bảo hiệu năng tương đối.* Ngoài ra ta có *giá trị đảm bảo hiệu năng* $R(x, y)$, với phương án $y$ của đầu vào $x$, $R(x, y) \geq 1$, dấu bằng xảy ra khi mà nghiệm xấp xỉ chính là nghiệm tối ưu. Một số bài toán được chứng minh lý thuyết rằng không có thuật toán xấp xỉ $\rho$ với một số giá trị nhất định. Ví dụ với bài toán người đi du lịch, nếu ta có một thuật toán xấp xỉ, giá trị $\rho$ của nó sẽ nằm trong khoảng 1.008196 và 1.5.

Không phải thuật toán xấp xỉ nào cũng dùng được trong thực tế. Ngoài những vấn đề như việc cái đặt phức tạp, thuật toán chỉ cải thiện không thực tiễn... còn có cả vấn đề về sự chính xác, nghiệm xấp xỉ không có độ chính xác về số mà bài toán quyết định yêu cầu. Trong thực tế, những thuật toán xấp xỉ thường được sử dụng là những thuật toán xấp xỉ dạng PTAS.

# 8 Meta Heuristics

Một heuristic là một dạng tìm kiếm để tìm nghiệm gần đúng của bài toán. Ở mỗi bước lặp, một heuristic sẽ cân nhắc giá trị của các lựa chọn để quyết định nên tiến hành theo lựa chọn nào. Các heuristic đánh đổi sự chính xác và đầy đủ cho thời gian chạy. Một meta heuristic là một heuristic bậc cao, có thể tìm, sinh ra, hoặc lựa chọn giữa các heuristic. Meta heuristic có thể được dùng để tìm nghiệm gần đúng, với rất ít giả định và đủ tốt cho các bài toán tối ưu, do đó được sử dụng rộng rãi để giải quyết nhiều bài toán.

Khác với thuật toán xấp xỉ và các thuật toán tìm đúng, heuristic/meta heuristic nói chung không có đảm bảo về mặt lý thuyết. Thậm chí với những meta heuristic sử dụng yếu tố ngẫu nhiên, kết quả sẽ phụ thuộc vào những số ngẫu nhiên được sinh ra trong quá trình giải. Tuy nhiên, vì việc sử dụng meta heuristic đưa ra kết quả khả quan trong nhiều trường hợp, nó vẫn được ứng dụng rộng rãi.

Một họ meta heuristic rất phổ biến đó là thuật toán di truyền. Để sử dụng thuật toán di truyền, đầu vào của bài toán cần được mã hóa ở dạng nào đó. Sau đó, ta cần định nghĩa một hàm đánh giá chất lượng, một cách để trao đổi chéo, một cách để đột biến. Quá trình chạy thuật toán có thể tổng quát như sau:

1. Chọn và mã hóa một số đầu vào $x_1, x_2, \ldots, x_n$, ta gọi mỗi $x_i$ là một cá thể, và tập các $x_i$ là quần thể.

2. Đánh giá mỗi cá thể sử dụng hàm đánh giá $f$, hàm này thường sử dụng hàm mục tiêu của bài toán.

3. Tính độ thích nghi của cá thể $x_i$ vào với công thức

$$\frac{f\left(x_i\right)}{\sum_{k=1}^{n} f\left(x_k\right)}.$$

4. Chọn một cách ngẫu nhiên hai hay nhiều cá thể giữa các cá thể dựa vào tỉ lệ thích nghi. Tiến hành trao đổi chéo để sinh ra một các thể mới.

5. Chọn ngẫu nhiên một cá thể để đột biến, sinh ra cá thể mới.

6. Thêm các cá thể mới vào trong quần thể, loại bỏ những cá thể $x_i$ mà có $f\left(x_i\right)$ không cao.

7. Lặp lại từ bước 3 cho tới khi quần thể không đổi qua các bước lặp, hoặc đã qua một số bước lặp nào đó mà không đạt được điều kiện như trên.

Đây chỉ là phiên bản đơn giản nhất của giải thuật di truyền. Tùy vào bài toán, cách cài đặt có thể đưa ra các biến thể, cấu hình khác nhau của thuật toán (e.g. ta có thể giới hạn cho bước đột biến xảy ra với một số cá thể có ràng buộc nào đó như $f(x_i)$ thấp, hoặc cho bước đột biến chỉ xảy ra với một điều kiện nào đó).

**Ví dụ 4.** Áp dụng thuật toán di truyền với bài toán người đi du lịch, ta có thể:

1. Mã hóa đầu vào là một dãy số, thể hiện thứ tự đi qua các thành phố.
2. Hàm đánh giá $f$ chính là hàm mục tiêu.
3. Việc trao đổi chéo giữa $x_i$ và $x_j$ được định nghĩa bằng cách dùng $x_j$ như một hoán vị của $x_i$ (hoặc ngược lại)
4. Việc đột biến của $x_i$ được định nghĩa bằng cách lấy ra hai số thứ tự trong $x_i$ và đổi thứ tự của chúng cho nhau