ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO TÌM HIỂU BÀI TOÁN TỐI ƯU

Giảng viên: Trịnh Văn Chiến
Người hướng dẫn: Nguyễn Quang Đông

Thành viên	MSSV
Đào Thái Hoàng	20235720
Trần Thu Phương	20235811

Hà Nội, 2025

Mục lục

1 CƠ SỞ LÝ THUYẾT

1.1 Bài toán tối ưu

1.1.1 Bài toán tối ưu là gì?

Bài toán tối ưu là quá trình tìm kiếm giá trị cực trị (cực tiểu hoặc cực đại) của một hàm số nhất định trên một tập hợp các lời giải khả thi (không gian lời giải).

Ví dụ cơ bản: Tìm giá trị nhỏ nhất của hàm số $f(x) = x^2 + 1$ trên tập $D = [-1, 1]$. Lời giải cho bài toán này là tại $x = 0$ **hàm số** $f(x)$ **đạt giá trị cực tiểu bằng 1**.

1.1.2 Phát biểu tổng quát

Cho một ánh xạ $f : D \to \mathbb{R}^n$ với $D \subseteq \mathbb{R}^n$. Mục tiêu là tìm $x \in D$ sao cho $f(x)$ đạt cực tiểu (hoặc cực đại):

$$f(x) \to \min$$

Tập các giá trị x thỏa mãn điều kiện này được gọi là tập lời giải tối ưu: $argmin\{x \in D | f(x)\}$.

1.1.3 Các thành phần cơ bản

- $f(x)$ được gọi là **hàm mục tiêu (objective function)**: Đây là hàm số mà chúng ta muốn tìm cực trị.
- D là **không gian lời giải (solution space)** hay **miền ràng buộc (constraints)**: Đây là tập hợp tất cả các giá trị x hợp lệ mà lời giải phải thuộc về. Không gian này thường được định nghĩa bởi một hoặc nhiều hàm ràng buộc $g(x)$ (ví dụ: $g(x) \leq 0$ hoặc $g(x) = 0$).
- **Trạng thái (hay lời giải):** Là một giá trị x cụ thể thuộc không gian D. Lời giải tối ưu là trạng thái $x_0 \in D$ sao cho $f(x_0)$ là cực trị.

1.1.4 Ví dụ trong Kỹ thuật Truyền thông

Trong tài liệu "Nhập môn Kỹ thuật Truyền thông" (IT4593), các bài toán tối ưu xuất hiện liên tục:

1. **Tối ưu hóa hệ thống:** Hệ thống truyền thông được thiết kế để "tối ưu hóa về độ tin cậy, độ trễ và hiệu suất phổ" (Lời nói đầu, trang 2). Đây là một bài toán tối ưu phức tạp.
2. **Tối đa hóa SNR:** Bộ lọc phối hợp (Matched Filter) được thiết kế để "tối đa hóa tỷ lệ tín hiệu so với nhiễu (SNR) tại thời điểm lấy mẫu" (Chương 6, trang 78).
   - *Hàm mục tiêu:* $SNR(t)$.
   - *Mục tiêu:* $\max(SNR(t))$.
3. **Tối thiểu hóa xác suất lỗi (MAP/ML):** Lý thuyết ra quyết định (Chương 5) tìm cách tối thiểu hóa xác suất lỗi.
   - **Tiêu chuẩn MAP (Maximum a Posteriori):** Tìm lời giải s_R để *cực đại hóa* xác suất hậu nghiệm: $s_R = \arg\max_{s_i} Pr(s_T = s_i | r = \rho)$.
   - **Tiêu chuẩn ML (Maximum Likelihood):** Tìm lời giải s_R để *cực đại hóa* hàm khả năng: $s_R = \arg\max_{s_i} f_r(\rho | s_T = s_i)$.
   - Trong kênh AWGN, tiêu chuẩn ML tương đương với việc *tối thiểu hóa* khoảng cách Euclid: $s_R = \arg\min_{s_i} d_E^2(\rho - s_i)$ (Chương 5, trang 72). Đây là một bài toán tối ưu cổ điển.

1.2 Các khái niệm cơ bản về cực tiểu

Để hiểu rõ hơn về lời giải của bài toán tối ưu, ta phân biệt các loại cực tiểu sau:

Khái niệm	Định nghĩa	Phát biểu Toán học
Cực tiểu toàn cục (Global minimum)	Điểm tại đó $f(x)$ đạt giá trị nhỏ nhất trên toàn bộ tập D.	x_0 là cực tiểu toàn cục khi: $f(x_0) \leq f(x) \forall x \in D$.
Cực tiểu toàn cục chặt (Strict global minimum)	Điểm tại đó $f(x)$ đạt giá trị nhỏ nhất và giá trị đó là duy nhất.	x_0 là cực tiểu toàn cục chặt khi: $f(x_0) < f(x) \forall x \in D \setminus \{x_0\}$.
Cực tiểu địa phương (Local minimum)	Là điểm mà $f(x)$ đạt giá trị nhỏ nhất trong một lân cận nào đó của nó.	x_0 là cực tiểu địa phương khi tồn tại $\epsilon > 0$ sao cho: $f(x_0) \leq f(x) \forall x \in [x_0 - \epsilon, x_0 + \epsilon] \cap D$.
Cực tiểu địa phương chặt (Strict local optimum)	Là điểm cực tiểu địa phương và giá trị đó là duy nhất trong lân cận.	$f(x_0) < f(x) \forall x \in ([x_0 - \epsilon, x_0 + \epsilon] \setminus \{x_0\}) \cap D$.

Đánh giá tính "chặt" (cho hàm liên tục):

- Nếu $f'(x_0) = 0$ và $f''(x_0) > 0$, thì x_0 là một điểm cực tiểu địa phương chặt.

1.3 Phân loại các bài toán tối ưu

Tùy thuộc vào đặc điểm của hàm mục tiêu $f(x)$ và không gian lời giải D, bài toán tối ưu được phân loại thành các dạng sau:

- **Quy hoạch tuyến tính (Linear Programming):** Nếu hàm mục tiêu $f(x)$ và các hàm ràng buộc $g(x)$ đều là tuyến tính, và không gian lời giải D là một tập lồi.
- **Quy hoạch phi tuyến (Non-linear Programming):** Nếu hàm mục tiêu hoặc ít nhất một hàm ràng buộc **không phải là tuyến tính**.
- **Quy hoạch nguyên (Integer Programming):** Nếu không gian lời giải D bị ràng buộc phải là các giá trị **nguyên rời rạc**.
- **Quy hoạch đa mục tiêu (Multi-objective Programming):** Nếu có nhiều hơn một hàm mục tiêu cần tối ưu đồng thời.
- **Quy hoạch ngẫu nhiên (Stochastic Programming):** Khi hàm mục tiêu hoặc hàm ràng buộc chứa các hệ số không chắc chắn, tuân theo một phân phối xác suất nào đó.
- **Quy hoạch tham số (Parametric Programming):** Khi hàm mục tiêu hoặc hàm ràng buộc chứa các tham số chưa xác định.

1.3.1 Bài toán tối ưu nguyên (Tối ưu rời rạc/Tổ hợp)

- **Đặc điểm:** Tập lời giải D là các giá trị rời rạc (đếm được).
- **Phát biểu toán học:** $I = (U, P, f, extr)$.
  - U: Không gian lời giải (hữu hạn hoặc đếm được).
  - P: Các ràng buộc (constraints).

- f: Hàm mục tiêu cần tối ưu.
- $extr$: Cực trị (thường là max hoặc min).

- **Ví dụ thường gặp:** Bài toán người du lịch (TSP), Bài toán cái túi (Knapsack Problem - KP), Bài toán phân công (Assignment Problem - AP), Bài toán cây khung nhỏ nhất (MST).

- **Ví dụ trong Kỹ thuật Truyền thông:** Việc "Gán nhãn Gray" (Chương 8, trang 109) là một bài toán tối ưu tổ hợp. Mục tiêu là tìm một phép gán nhãn (ánh xạ) $e : \mathbb{H}_k \leftrightarrow \mathcal{M}$ sao cho tổng số bit lỗi khi xảy ra nhầm lẫn giữa các ký hiệu lân cận là *cực tiểu*.

1.3.2 Bài toán tối ưu liên tục

- **Đặc điểm:** Tập lời giải D là các giá trị liên tục (thường là một tập con của $\mathbb{R}^n$).

- **Phát biểu toán học:** $\min_{x \in \mathbb{R}^n} f(x)$ với $g_i(x) \leq 0$.

- **Ví dụ trong Kỹ thuật Truyền thông:**
  - Thiết kế bộ lọc: Tìm các hệ số của bộ lọc (các giá trị thực, liên tục) để tối đa hóa SNR.
  - Đồng bộ thời gian: Tìm thời điểm lấy mẫu t_0 (một giá trị liên tục) để tối đa hóa đầu ra của bộ lọc phối hợp (Chương 6, 9).

1.3.3 Bài toán tối ưu đa mục tiêu

- **Đặc điểm:** Khi có nhiều hơn một (thường là mâu thuẫn nhau) hàm mục tiêu cần tối ưu. Ví dụ: $F(x) = (f_1(x), f_2(x), ..., f_k(x)) \rightarrow \min$. Thường không thể tìm được một điểm x duy nhất mà tất cả các hàm $f_i(x)$ cùng đạt giá trị nhỏ nhất.

- **Khái niệm Tối ưu Pareto:**
  - Một điểm x_1 **được gọi là trội hơn (dominates)** x_2 (ký hiệu $x_1 \prec x_2$) nếu:
    1. x_1 không tồi hơn x_2 ở mọi mục tiêu: $f_i(x_1) \leq f_i(x_2) \forall i \in \{1, ..., k\}$.
    2. x_1 tốt hơn x_2 ở ít nhất một mục tiêu: $\exists j \in \{1, ..., k\} : f_j(x_1) < f_j(x_2)$.
  - **Biên Pareto (Pareto Front):** Là tập hợp tất cả các điểm x "không bị trội" (non-dominated), tức là không có bất kỳ điểm x' nào khác trong D mà $x' \prec x$. Đây là tập các lời giải "tốt nhất" có thể, thể hiện sự đánh đổi giữa các mục tiêu.

- **Ví dụ trong Kỹ thuật Truyền thông:**
  - Tài liệu IT4593 (Lời nói đầu, trang 2) mô tả hệ thống truyền thông là sự cân bằng của nhiều mục tiêu: **tối ưu độ tin cậy** (tức là min(lỗi)), **tối ưu độ trễ** (min(trễ)), và **tối ưu hiệu suất phổ** (max(hiệu suất)).
  - Chương 10 (trang 170) nêu rõ bài toán "Cân bằng giữa hiệu quả sử dụng phổ và xác suất lỗi". Tăng hiệu quả sử dụng phổ (ví dụ: dùng 64-PAM thay vì 2-PAM) thường sẽ làm tăng xác suất lỗi. Người thiết kế phải chọn một điểm trên Biên Pareto (một sự đánh đổi) phù hợp với yêu cầu.

Hướng giải	**Phương pháp điển hình**	**Đặc điểm**
Giải chính xác (Exact)	Thuật toán **Vét cạn** (Brute-force)	Duyệt toàn bộ không gian lời giải D. Chỉ khả thi với D rất nhỏ.
	Thuật toán **Nhánh cận** (Branch and Bound)	Phương pháp duyệt có chiến lược, loại bỏ các nhánh tìm kiếm không tiềm năng.
	Quy hoạch động (Dynamic Programming)	Giải bài toán con và lưu kết quả để sử dụng lại.
	Giải tích (ví dụ: Gradient Descent)	Dùng đạo hàm để tìm cực tiểu (cho bài toán liên tục, phi tuyến).
	Tiêu chuẩn ML/MAP (trong truyền thông)	Như đã đề cập ở (Chương 5, IT4593), đây là các phương pháp giải *chính xác* cho bài toán *tìm tín hiệu có khả năng cao nhất* trong một tập hữu hạn.
Giải gần đúng (Heuristic/Approximate)	Thuật toán **Tham lam** (Greedy)	Lựa chọn phương án có vẻ là tối ưu nhất ở trạng thái hiện tại, có thể rơi vào bẫy cực tiểu địa phương.
	Thuật toán **Mô phỏng luyện kim** (Simulated Annealing)	Lựa chọn dựa trên xác suất; các phương án tồi có thể được chấp nhận với xác suất giảm dần, nhằm thoát bẫy cục bộ.
	Tính toán Tiến hóa (Evolutionary Computing)	(Xem chi tiết bên dưới) Sử dụng quần thể lời giải và các toán tử di truyền để tìm kiếm.

1.4 Các phương pháp giải bài toán tối ưu

Để giải bài toán tối ưu, có hai hướng tiếp cận chính: **giải chính xác** (tìm lời giải tối ưu toàn cục) và **giải gần đúng** (tìm lời giải đủ tốt trong thời gian chấp nhận được).

Giải gần đúng (Heuristic) được sử dụng vì nhiều bài toán tối ưu (đặc biệt là tối ưu tổ hợp) thuộc lớp NP-hard, nghĩa là không thể tìm ra lời giải chính xác trong thời gian đa thức.

2 GIỚI THIỆU VỀ TÍNH TOÁN TIẾN HÓA

Tính toán tiến hóa (Evolutionary Computing - EC) là một kỹ thuật tìm kiếm heuristic, lấy cảm hứng từ quá trình tiến hóa và chọn lọc tự nhiên trong sinh học. EC sử dụng một "quần thể" các lời giải tiềm năng và áp dụng các "toán tử di truyền" để cải thiện quần thể qua nhiều "thế hệ".

Hai toán tử cốt lõi trong EC là **toán tử lai ghép (crossover operator)** và **toán tử đột biến (mutation operator)**.

2.1 Thuật toán tiến hóa (Evolution Algorithm – EA)

- EA là một thuật toán thuộc loại **heuristic** (giải gần đúng).
- EA là giải pháp hiệu quả để tiết kiệm chi phí tính toán cho các bài toán tối ưu phức tạp (NP-hard). Thay vì duyệt toàn bộ không gian lời giải (tốn kém), EA thực hiện một cuộc tìm kiếm thông minh, có hướng dẫn để hội tụ về một lời giải "đủ tốt".
- Ví dụ: Google Maps sử dụng các thuật toán heuristic (như A*) để tìm đường đi. Đường đi gợi ý có thể chưa chắc là ngắn nhất tuyệt đối về mặt lý thuyết, nhưng là lời giải rất tốt và được tìm thấy gần như tức thời.

2.1.1 Ý tưởng của EA

Bắt đầu từ một quần thể các lời giải (cha mẹ), EA tạo ra thế hệ con thông qua các phép toán **lai ghép** (kết hợp các đặc điểm tốt của cha mẹ) và sử dụng phép toán **đột biến** (thêm vào các đặc điểm mới) để duy trì sự đa dạng của quần thể. Sau đó, quá trình **chọn lọc** sẽ giữ lại các cá thể "thích nghi" nhất (có giá trị hàm mục tiêu tốt nhất) để tạo ra thế hệ kế tiếp.

2.1.2 Đoạn mã giả của Thuật toán Tiến hóa (EA)

```
begin
    Khởi tạo quần thể P gồm N cá thể;
    Đánh giá độ thích nghi của các cá thể trong P;

    while (chưa đạt điều kiện dừng) do
        P_parents = Chọn ra các cá thể cha mẹ từ P;
        P_offspring = Tạo ra các cá thể con bằng toán tử lai ghép
                      và đột biến từ P_parents;
        Đánh giá độ thích nghi của các cá thể con trong P_offspring;
        P = Chọn ra các cá thể sống sót từ (P   P_offspring)
            để tạo quần thể mới;
    end;
end
```

Giải thích các bước:

1. **Khởi tạo quần thể:** Tạo ngẫu nhiên N cá thể (lời giải) trong không gian D.
2. **Đánh giá (Evaluation):** Tính giá trị hàm mục tiêu $f(x)$ cho mỗi cá thể. Giá trị này gọi là "độ thích nghi" (fitness).
3. **Chọn cha mẹ (Selection):** Chọn các cá thể từ P để sinh sản. Các cá thể có độ thích nghi cao hơn sẽ có cơ hội được chọn cao hơn (ví dụ: chọn lọc Tournament, Roulette Wheel).
4. **Lai ghép (Crossover):** Tạo cá thể con bằng cách kết hợp thông tin di truyền (các phần của lời giải) từ hai hay nhiều cha mẹ.
5. **Đột biến (Mutation):** Thay đổi ngẫu nhiên một phần nhỏ của cá thể con. Bước này giúp duy trì sự đa dạng và tránh hội tụ sớm về cực tiểu địa phương.
6. **Chọn lọc sống sót (Survival Selection):** Chọn N cá thể tốt nhất từ cả quần thể cha mẹ (P) và con (P_offspring) để tạo ra thế hệ P tiếp theo.
7. **Điều kiện dừng:** Có thể là một trong các điều kiện sau:
    - Đạt được một giá trị mục tiêu (fitness) đủ tốt.
    - Số thế hệ (vòng lặp) đạt giới hạn.
    - Độ đa dạng của quần thể giảm xuống dưới một ngưỡng (quần thể đã hội tụ).

2.2 Mô hình hóa bài toán (Mã hóa)

Để áp dụng EA, ta phải mã hóa lời giải của bài toán thành một cấu trúc dữ liệu mà EA có thể thao tác (gọi là "nhiễm sắc thể" hay "kiểu gen").

- **Với bài toán TSP:** Cá thể có thể mô hình hóa là một **dãy hoán vị** của các thành phố (ví dụ: `[3, 1, 4, 2]`).
- **Với bài toán cái túi (KP):** Cá thể có thể mô hình hóa là một **dãy bit** (ví dụ: `[1, 0, 1, 1]`, bit 1 nghĩa là "chọn" vật phẩm, 0 là "không chọn").

2.2.1 Kiểu gen và Kiểu hình

- **Kiểu gen (Genotype):** Là lời giải đã được mã hóa mà thuật toán làm việc trực tiếp trên đó (ví dụ: chuỗi bit `0100`). Các toán tử lai ghép và đột biến được thực hiện trên kiểu gen.
- **Kiểu hình (Phenotype):** Là lời giải tường minh của bài toán (ví dụ: số `4`). Cần một hàm giải mã (decode) để chuyển từ kiểu gen sang kiểu hình để đánh giá độ thích nghi.

2.3 Phân nhánh Thuật toán Tiến hóa (EA)

Tùy theo cách mô hình hóa (mã hóa) cá thể, EA được phân thành nhiều nhánh khác nhau:

- **Thuật toán Di truyền (Genetic Algorithm – GA):** Kiểu gen thường là chuỗi ký tự (chuỗi bit, chuỗi số nguyên).
- **Chiến lược Tiến hóa (Evolution Strategy – ES):** Kiểu gen là vector số thực. Thường dùng cho bài toán tối ưu liên tục.

- **Lập trình Di truyền (Genetic Programing – GP):** Kiểu gen là một cây cú pháp (đại diện cho một chương trình máy tính hoặc một biểu thức toán học).

- **Tiến hóa Sai phân (Differential Evolution – DE):** Sử dụng vector số thực, nhưng dùng phép toán "sai phân" (hiệu của các vector) để tạo ra cá thể mới.

- **Tiến hóa Đa nhiệm (Evolutionary Multi-tasking – EM):** Giải quyết nhiều bài toán tối ưu cùng một lúc.