ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

Học phần: Nhập môn Kỹ thuật truyền thông - 161323

Đề tài: Thuật toán Bầy ong Nhân tạo & Các biến thể
Phân tích bài toán tối ưu trong truyền thông

Nhóm sinh viên thực hiện

STT	Họ và tên	MSSV
1	Nguyễn Ngọc Dung	20235684
2	Triệu Trường Giang	20235700

Ngày 18 tháng 01 năm 2026

Mục lục

TÓM TẮT

Phần I: Thuật toán bầy ong nhân tạo và các biến thể.
Báo cáo trình bày thuật toán Bầy ong Nhân tạo, viết tắt là ABC, dựa trên hành vi kiếm mật của ong. Nội dung bao gồm các bước khởi tạo quần thể, pha ong thợ, pha ong quan sát, pha ong do thám, cách tính độ thích nghi, xử lý biên và mã giả triển khai. Trên nền ABC gốc, báo cáo giới thiệu các biến thể tiêu biểu gồm GABC, bổ sung thành phần dẫn hướng theo nghiệm tốt nhất toàn cục nhằm tăng khả năng khai thác và tăng tốc hội tụ; qABC, tập trung tìm kiếm quanh nghiệm tốt để giảm số phép đánh giá và rút ngắn thời gian hội tụ; và Chaos ABC hay CABC, thay biến ngẫu nhiên bằng chuỗi hỗn loạn logistic nhằm tăng đa dạng quần thể và giảm nguy cơ kẹt tại cực trị cục bộ. Các thuật toán được mô tả trong cùng một khung tối ưu đa biến và nhấn mạnh sự cân bằng giữa thăm dò và khai thác.

Phần II: Ứng dụng vào thiết kế multibeam beamforming cho JCAS và kết quả.
Báo cáo áp dụng ABC và các biến thể cho bài toán thiết kế multibeam beamforming trong hệ thống tích hợp truyền thông và cảm biến, viết tắt là JCAS, với mảng anten analog dùng chung. Hai beam con cho truyền thông và cảm biến được kết hợp thông qua hệ số chia công suất ρ trong khoảng từ 0 đến 1 và véc tơ pha $\boldsymbol{\phi}$ trong miền từ $-\pi$ đến π để cực đại hóa công suất thu chuẩn hóa. Hàm mục tiêu được xây dựng dưới dạng Rayleigh quotient dựa trên $\mathbf{H}^H\mathbf{H}$ khi biết đầy đủ ma trận kênh $\mathbf{H}$, hoặc tối ưu theo độ lợi mảng theo hướng phát trội khi chỉ biết AoD trội. Báo cáo cũng tổng hợp các phương án lượng tử hóa véc tơ beamforming phù hợp phần cứng analog gồm một bộ dịch pha, hai bộ dịch pha theo lượng tử riêng hoặc theo codebook, và IGSS Q tối ưu hệ số co giãn nhằm giảm suy hao lượng tử. Kết quả mô phỏng so sánh tốc độ hội tụ, độ ổn định và giá trị tối ưu cho thấy GABC thường hội tụ nhanh và đạt nghiệm tốt hơn, Chaos ABC tăng khả năng thăm dò và giảm nguy cơ mắc kẹt cục bộ, trong khi ABC gốc và qABC đóng vai trò chuẩn so sánh theo mức độ đơn giản và chi phí tính toán.

Phần I Thuật toán bầy ong nhân tạo (ABC algorithm)

1 Tổng quan về Thuật toán Bầy ong Nhân tạo

1.1 Giới thiệu thuật toán

Thuật toán Bầy ong Nhân tạo (ABC - Artificial Bee Colony) là thuật toán meta-heuristic để giải các bài toán gần đúng và các bài toán phức tạp. Theo đánh giá, ABC là thuật toán hiệu quả để giải quyết bài toán np-khó có nhiều điểm tối ưu cục bộ. Hiệu quả của thuật toán được đánh giá và kiểm tra trên các bài toán tối ưu hàm số thực. Kết quả thực nghiệm trên các hàm số thực cho ra kết quả nhanh và chính xác hơn

1.2 Ý tưởng thuật toán

Trong tự nhiên, ong mật có một cơ chế kiếm ăn rất hiệu quả. Một số ong thợ rời tổ để tìm kiếm các nguồn mật. Khi phát hiện một nguồn thức ăn dồi dào, chúng quay trở về tổ và thực hiện "điệu nhảy lắc" (waggle dance) để thông báo cho các ong khác về vị trí và chất lượng của nguồn thức ăn. Những ong quan sát tại tổ dựa trên thông tin này để quyết định có nên đến khai thác nguồn thức ăn đó hay không. Nếu một nguồn mật trở nên cạn kiệt, ong thợ sẽ từ bỏ và bắt đầu tìm kiếm nguồn mới ở một vị trí khác.

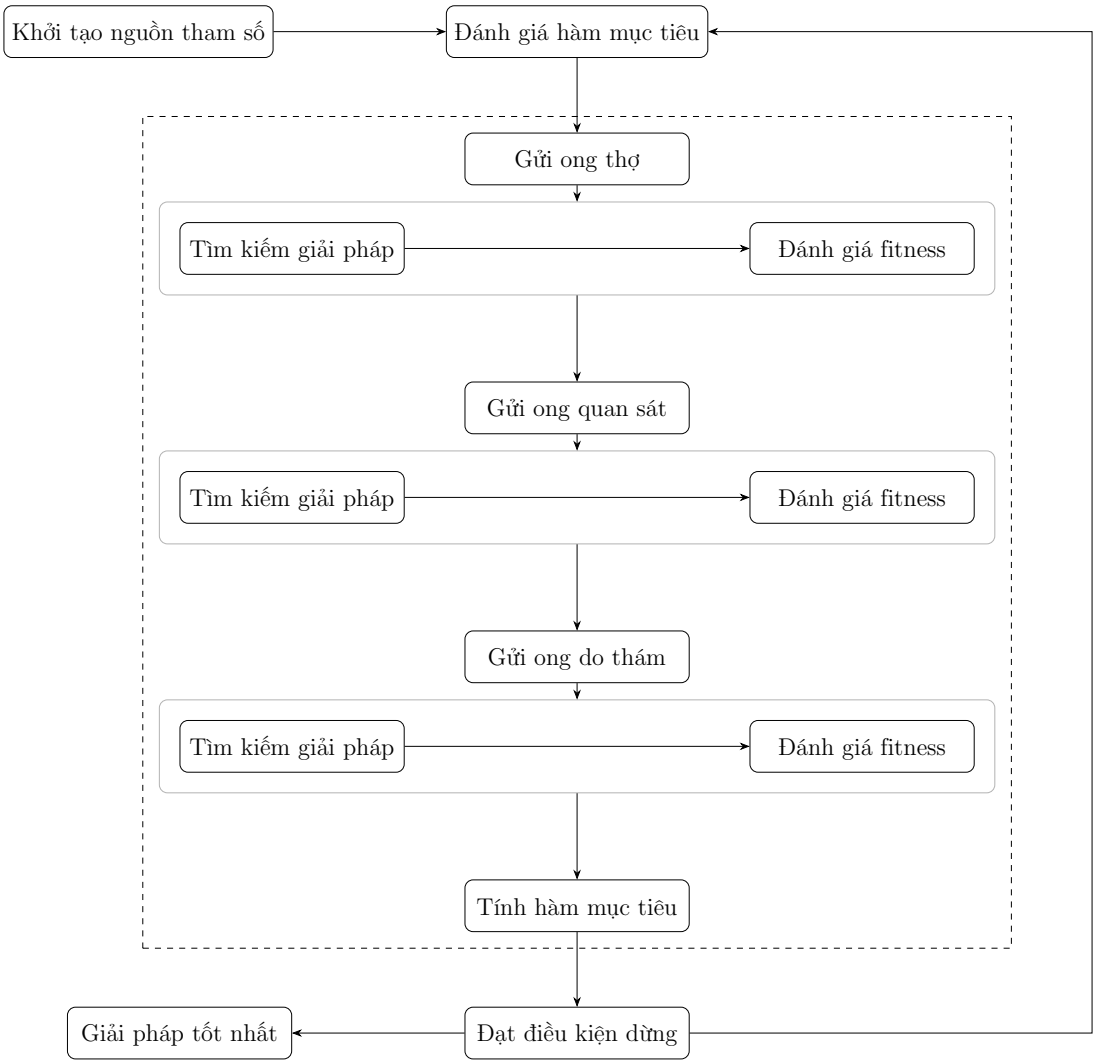
Cơ chế kiếm ăn thông minh này đã truyền cảm hứng cho thuật toán Bầy ong Nhân tạo (ABC). Trong mô hình toán học của ABC, mỗi nguồn thức ăn được xem như một nghiệm trong không gian tìm kiếm, còn các ong tương ứng với các tác nhân tìm kiếm:

- **Ong thợ (Employed Bees)**: Biểu diễn các nghiệm hiện tại và khai thác vùng lân cận để cải thiện nghiệm.
- **Ong quan sát (Onlooker Bees)**: Chọn nghiệm để khai thác tiếp dựa trên mức "chất lượng mật" (độ thích nghi).
- **Ong do thám (Scout Bees)**: Xuất hiện khi một nghiệm bị khai thác quá mức và không còn tiềm năng, và sẽ tìm kiếm một nghiệm hoàn toàn mới ở một vị trí ngẫu nhiên.

Nhờ sự luân phiên giữa hai quá trình: *khai thác cục bộ* (local exploitation) xung quanh nghiệm tốt và *khám phá toàn cục* (global exploration) thông qua ong trinh sát, thuật toán ABC có khả năng tìm kiếm đa dạng và tránh rơi vào các điểm tối ưu cục bộ. Đây là yếu tố giúp ABC trở thành một thuật toán meta-heuristic hiệu quả cho các bài toán tối ưu phức tạp.

1.3 Mô hình hoá thuật toán

Trong thuật toán ABC, quần thể có kích thước SN được chia thành hai nhóm bằng nhau: *ong thợ* (employed bees) và *ong quan sát* (onlooker bees). Mỗi ong thợ tương ứng với một nguồn thức ăn, tức là một nghiệm trong không gian tìm kiếm D chiều. Do đó, số lượng nguồn thức ăn bằng đúng số lượng ong thợ.



Hình 1: Lưu đồ thuật toán bầy ong ABC (artificial bee colony)

1.3.1 Giai đoạn 1: Khởi tạo nguồn thức ăn

Khởi tạo ngẫu nhiên SN nghiệm ban đầu theo công thức:

$$x_{ij} = min_j + rand(0,1)\,(max_j - min_j)$$

Trong đó:

- x_{ij} là giá trị tại chiều j của nguồn thức ăn thứ i,
- $rand(0,1)$ là số ngẫu nhiên trong khoảng $(0,1)$,
- min_j, max_j là giới hạn dưới và trên của chiều j.

Đánh giá nghiệm ban đầu

Sau khi khởi tạo ngẫu nhiên SN nguồn thức ăn, mỗi nghiệm được đánh giá bằng hàm mục tiêu và độ thích nghi (fitness). Nghiệm có fitness cao nhất được chọn làm nghiệm tốt nhất ban đầu X_{best} trước khi thuật toán chuyển sang pha ong thợ.

1.3.2 Giai đoạn 2: Pha của ong thợ (Employed Bees)

Mỗi ong thợ tìm cách cải thiện nghiệm mà nó đang phụ trách bằng cơ chế tìm kiếm lân cận.

Bước 1: Tại nguồn thức ăn i, chọn ngẫu nhiên một chiều $j \in \{1, \ldots, D\}$.

Bước 2: Chọn ngẫu nhiên một nguồn khác $k \neq i$.

Bước 3: Tạo nghiệm ứng viên theo:

$$v_{ij} = x_{ij} + \varphi\,(x_{ij} - x_{kj}),$$

với φ là số ngẫu nhiên trong khoảng $[-1, 1]$. Các chiều khác giữ nguyên: $v_{il} = x_{il}$ nếu $l \neq j$.

Bước 4: Tính giá trị hàm mục tiêu $f(v_i)$.

Bước 5: Chuyển đổi giá trị hàm mục tiêu sang độ thích nghi (fitness):

$$fitness(f(x)) = \begin{cases} \dfrac{1}{1+f(x)} & \text{nếu } f(x) \geq 0, \\ 1 + |f(x)| & \text{nếu } f(x) < 0. \end{cases}$$

Bước 6: Chọn giữa nghiệm cũ và nghiệm mới bằng chiến lược "greedy selection": nếu $f(v_i)$ tốt hơn $f(x_i)$ thì thay thế $x_i \leftarrow v_i$, ngược lại giữ nguyên và tăng bộ đếm số lần thất bại $trial_i = trial_i + 1$.

1.3.3 Giai đoạn 3: Pha của ong quan sát (Onlooker Bees)

Ong quan sát đưa ra quyết định dựa trên thông tin từ ong thợ.

Bước 1: Tính xác suất lựa chọn nguồn thức ăn dựa trên fitness:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}.$$

Bước 2: Mỗi ong quan sát chọn một nguồn thức ăn theo phân phối xác suất p_i và tạo nghiệm mới theo cùng cơ chế như ong thợ (Bước 1–5 của giai đoạn 2).

Bước 3: Cập nhật nghiệm tốt nhất tìm được trong giai đoạn này.

1.3.4 Giai đoạn 4: Pha của ong do thám (Scout Bees)

Nếu một nguồn thức ăn i không được cải thiện trong số lần vượt quá ngưỡng *limit* (tức là $trial_i > limit$), nguồn đó được xem là đã bị khai thác cạn kiệt.

Nguồn này sẽ bị loại bỏ và thay thế bằng một nghiệm mới sinh ngẫu nhiên theo công thức khởi tạo:

$$x_{ij} = min_j + rand(0, 1)\,(max_j - min_j).$$

1.3.5 Chu trình lặp

Ba giai đoạn: ong thợ, ong quan sát và ong trinh sát được lặp lại liên tục cho đến khi thoả mãn điều kiện dừng (số vòng lặp tối đa hoặc không còn cải thiện đáng kể). Nghiệm tốt nhất trong toàn bộ quá trình được xem là lời giải của bài toán tối ưu.

1.4 Mô tả bài toán

Bài toán tối ưu hóa được xem xét trong thuật toán ABC là tìm một nghiệm x^* trong không gian tìm kiếm D chiều sao cho giá trị của hàm mục tiêu $f(x)$ đạt cực tiểu (hoặc cực đại):

$$x^* = \arg\min_{x \in \Omega} f(x)$$

Trong mô hình ABC, mỗi nghiệm ứng viên được biểu diễn dưới dạng một *nguồn thức ăn* trong tự nhiên, và chất lượng của nghiệm tương ứng với lượng mật hoa thu được tại nguồn đó.

Ánh xạ hành vi sinh học sang bài toán tối ưu Để mô phỏng quá trình kiếm mật trong bài toán tối ưu, ABC ánh xạ các thành phần sinh học sang các khái niệm tối ưu như sau:

- **Nguồn thức ăn (Food source) ↔ Nghiệm ứng viên**: Mỗi nguồn thức ăn biểu diễn một nghiệm x_i trong không gian tìm kiếm. Chất lượng nguồn tương ứng với giá trị hàm mục tiêu.
- **Lượng mật hoa ↔ Độ thích nghi (Fitness)**: Nguồn có mật hoa nhiều hơn được xem là có chất lượng cao hơn; tương tự nghiệm có fitness tốt hơn sẽ được ưu tiên khai thác.
- **Ong thợ + Ong quan sát ↔ Khai thác (Exploitation)**: Thực hiện tìm kiếm lân cận nhằm cải thiện các nghiệm hiện có.
- **Ong do thám ↔ Khám phá (Exploration)**: Tạo nghiệm mới ngẫu nhiên khi nguồn hiện tại không còn được cải thiện.

Sự kết hợp giữa hai cơ chế *khai thác* và *khám phá* cho phép ABC duy trì sự cân bằng giữa tìm kiếm cục bộ và tìm kiếm toàn cục.

Cấu trúc đàn ong Dựa trên ánh xạ trên, ABC tổ chức đàn ong thành ba nhóm chính, mỗi nhóm đảm nhiệm một vai trò khác nhau trong quá trình tìm kiếm nghiệm tối ưu.

Ban đầu, thuật toán tạo ra một quần thể gồm SN nguồn thức ăn, trong đó mỗi nguồn x_i là một véc-tơ D chiều chứa các biến cần tối ưu. Đàn ong được chia thành ba nhóm: ong thợ (employed bees), ong quan sát (onlooker bees) và ong do thám (scout bees). Ba nhóm ong này phối hợp với nhau qua nhiều chu kỳ nhằm tìm ra nghiệm có chất lượng tốt nhất.

- **Ong thợ:** Mỗi ong thợ gắn với một nguồn thức ăn và chịu trách nhiệm khảo sát vùng lân cận của nguồn đó. Từ vị trí hiện tại, ong thợ tạo ra một nghiệm mới và so sánh chất lượng của nghiệm mới với nghiệm cũ. Nếu nghiệm mới tốt hơn thì cập nhật vị trí; nếu không, giữ nguyên nghiệm hiện tại.

- **Ong quan sát:** Sau khi ong thợ trở về tổ, thông tin về chất lượng các nguồn thức ăn được chia sẻ trên "sàn nhảy". Ong quan sát lựa chọn nguồn thức ăn để khai thác dựa trên xác suất tỉ lệ thuận với độ thích nghi, và tiếp tục thực hiện tìm kiếm lân cận tại nguồn được chọn.
- **Ong do thám:** Mỗi nguồn thức ăn được theo dõi bởi một bộ đếm số lần không cải thiện. Nếu một nguồn bị khai thác quá nhiều lần mà không có tiến triển (vượt quá ngưỡng *limit*), nó được xem là đã "cạn kiệt". Khi đó, ong thợ tương ứng trở thành ong do thám và thay thế nguồn này bằng một nguồn hoàn toàn ngẫu nhiên.

Chu trình tìm kiếm Quá trình ong thợ → ong quan sát → ong do thám được lặp lại liên tục trong nhiều vòng lặp. Sau khi đạt điều kiện dừng (ví dụ: số vòng lặp tối đa hoặc không còn cải thiện đáng kể), thuật toán trả về nguồn thức ăn có lượng mật cao nhất trong toàn bộ quá trình tìm kiếm. Nguồn này tương ứng với nghiệm tối ưu x^* của bài toán.

1.5 Mã giả thuật toán ABC gốc

Algorithm 1 Artificial Bee Colony (ABC)

Khởi tạo ngẫu nhiên SN nguồn thức ăn x_i trong miền cho phép.
Đánh giá hàm mục tiêu $f(x_i)$ và tính fitness tương ứng.
for iter = 1 đến MAX_ITER **do**
  Employed Bee Phase:
  for mỗi nguồn thức ăn x_i **do**
    Chọn ngẫu nhiên $k \neq i$.
    Chọn ngẫu nhiên một chiều j.
    Sinh nghiệm mới:
      $v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$.
    Giữ các thành phần khác: $v_{il} = x_{il}$, $l \neq j$.
    Nếu $f(v_i)$ tốt hơn $f(x_i)$ thì thay $x_i \leftarrow v_i$, ngược lại tăng bộ đếm "bị bỏ rơi" của x_i.
  end for
  Onlooker Bee Phase:
  Tính xác suất chọn nguồn thức ăn p_i từ fitness của x_i.
  for mỗi onlooker **do**
    Chọn một nguồn x_i theo phân phối xác suất p_i.
    Sinh nghiệm mới v_i tương tự giai đoạn employed.
    Áp dụng quy tắc chọn nghiệm tốt hơn như trên.
  end for
  Scout Bee Phase:
  for mỗi nguồn thức ăn x_i **do**
    if số lần bị bỏ rơi của x_i vượt LIMIT **then**
      Khởi tạo lại x_i ngẫu nhiên trong miền tìm kiếm.
    end if
  end for
end for
Trả về nghiệm tốt nhất tìm được.

1.6 Triển khai thuật toán

Bước 1: Khởi tạo nguồn thức ăn

```python
import numpy as np

# ====================================
# GIAI ĐOẠN 1: KHỞI TẠO CÁC THAM SỐ
# ====================================

N = 6                   # Tổng số ong
D = 2                   # Số chiều của nghiệm
SN = N // 2             # Số nguồn thức ăn = số ong thợ = 3
LIMIT = (N * D) // 2    # LIMIT = 6
MAX_ITER = 50

bounds = np.array([
    [0.5, 1.5],         # min, max của x1
    [1.5, 2.5]          # min, max của x2
])

# Sinh ngẫu nhiên nguồn thức ăn ban đầu
food_sources = np.zeros((SN, D))
for i in range(SN):
    for j in range(D):
        min_j, max_j = bounds[j]
        food_sources[i, j] = min_j + np.random.rand() * (max_j - min_j)

trial = np.zeros(SN, dtype=int)
```

Hình 2: Đoạn mã: Khởi tạo các tham số của giải thuật ABC

Trong giai đoạn khởi tạo, thuật toán thiết lập các tham số điều khiển và sinh ra tập nguồn thức ăn ban đầu để bắt đầu quá trình tối ưu hoá. Các tham số được sử dụng trong bài toán như sau:

- **Số lượng ong trong bầy**: $N = 6$.
- **Số lượng nguồn thức ăn ban đầu**: Trong thuật toán ABC, mỗi ong thợ quản lý đúng một nguồn thức ăn. Vì số ong quan sát bằng số ong thợ nên:

$$SN = \frac{N}{2} = 3.$$

- **Số chiều của nghiệm**: $D = 2$, tương ứng với hai biến cần tối ưu hoá: x_1 và x_2.
- **Giới hạn số lần cải thiện thất bại (limit)**: LIMIT là số lần tối đa mà một nguồn thức ăn có thể không được cải thiện trước khi bị xem là "cạn kiệt" và được thay bằng một nguồn mới. Theo thiết kế chuẩn của thuật toán ABC, LIMIT được chọn theo công thức:

$$L = \frac{N \times D}{2} = 6.$$

- **Số vòng lặp tối đa của thuật toán**:

$$I_{\max} = 50.$$

- **Khởi tạo nguồn thức ăn ban đầu bằng công thức ngẫu nhiên**:

  Các nguồn thức ăn được khởi tạo ngẫu nhiên trong miền giới hạn của từng biến theo công thức:

$$x_{ij} = \min_j + \text{rand}(0,1)\,(\max_j - \min_j),$$

  với $\text{rand}(0,1)$ là số ngẫu nhiên phân bố đều trong khoảng $(0,1)$ và $(\min_j, \max_j)$ là cận dưới và cận trên của chiều j.

  Miền giá trị của bài toán:

$$0.5 \le x_1 \le 1.5, \qquad 1.5 \le x_2 \le 2.5.$$

- **Khởi tạo bộ đếm *trial***: Mỗi nguồn thức ăn được gắn một bộ đếm số lần không cải thiện, ban đầu:

$$C = [0,\ 0,\ 0].$$

Bước 2: Đánh giá nghiệm ban đầu

```python
# Hàm mục tiêu
def objective(x):
    return (x[0] - 1)**2 + (x[1] - 2)**2 + 3

# Fitness
def fitness_func(f):
    if f >= 0:
        # Nếu f không âm → dùng hàm nghịch đảo
        return 1 / (1 + f)
    else:
        # Nếu f âm → tăng fitness tuyến tính
        return 1 + abs(f)

# Đánh giá nghiệm ban đầu
fitness = np.array([fitness_func(objective(x)) for x in food_sources])
best_index = np.argmax(fitness)
X_best = food_sources[best_index].copy()

print("\n--- Giai đoạn 1: Khởi tạo nguồn thức ăn ---")
for i, fs in enumerate(food_sources):
    print(f"  X{i} = {fs}, fitness = {fitness[i]:.4f}")
print("  → Nghiệm tốt nhất ban đầu:", X_best)
```

Hình 3: Đoạn mã: Đánh giá nghiệm ban đầu

Sau khi khởi tạo ngẫu nhiên SN nguồn thức ăn theo công thức:

$$x_{ij} = \min_j + \text{rand}(0,1)\,(\max_j - \min_j),$$

Thuật toán tiến hành đánh giá chất lượng của từng nghiệm ban đầu dựa trên hàm mục tiêu:

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^2 + 3.$$

Với mỗi nghiệm x_i, ta tính giá trị hàm mục tiêu $f(x_i)$ và chuyển đổi sang độ thích nghi (fitness) theo công thức:

$$fitness(f(x)) = \begin{cases} \dfrac{1}{1+f(x)}, & \text{nếu } f(x) \ge 0, \\ 1 + |f(x)|, & \text{nếu } f(x) < 0. \end{cases}$$

Độ thích nghi càng lớn cho thấy nghiệm càng có chất lượng cao. Sau khi tính toán fitness cho toàn bộ SN nghiệm, ta thu được:

$$X_0 = (1.07114883,\ 1.91321764), \qquad fitness(X_0) = 0.2492$$

$$X_1 = (0.76298546,\ 2.35042331), \qquad fitness(X_1) = 0.2393$$

$$X_2 = (0.80818979,\ 1.89944192), \qquad fitness(X_2) = 0.2471$$

Nghiệm có độ thích nghi lớn nhất được chọn làm nghiệm tốt nhất ban đầu:

$$x_{\text{best}} = X_0 = (1.07114883,\ 1.91321764).$$

Nghiệm này được sử dụng làm điểm tham chiếu trong các pha tiếp theo của thuật toán ABC.

```
--- Giai đoạn 1: Khởi tạo nguồn thức ăn ---
  X0 = [1.07114883 1.91321764], fitness = 0.2492
  X1 = [0.76298546 2.35042331], fitness = 0.2393
  X2 = [0.80818979 1.89944192], fitness = 0.2471
  → Nghiệm tốt nhất ban đầu: [1.07114883 1.91321764]
```

Hình 4: Kết quả đoạn mã: Giai đoạn 1

Sau giai đoạn này, thuật toán đã có: tập nguồn thức ăn ban đầu sinh ngẫu nhiên, bộ đếm tương ứng và toàn bộ tham số điều khiển. Đây là cơ sở để bước sang giai đoạn tiếp theo.

Bước 3: Pha của ong thợ

```python
# =====================================
# HÀM SINH NGHIỆM MỚI (cho employed & onlooker)
# =====================================
def generate_new_solution(i):
    # Chọn ngẫu nhiên một chiều
    k = np.random.randint(D)
    # Chọn nguồn lân cận khác
    j = np.random.choice([a for a in range(SN) if a != i])
    # Tạo nhiễu phi ~ U(-1,1)
    phi = np.random.uniform(-1, 1)

    v = food_sources[i].copy()
    v[k] = v[k] + phi * (v[k] - food_sources[j][k])

    # Giữ nghiệm trong khoảng
    v = np.clip(v, bounds[:, 0], bounds[:, 1])
    return v
```

Hình 5: Đoạn mã: Hàm sinh nghiệm mới cho ong thợ và ong quan sát (ABC)

```python
# ====================================
# BẮT ĐẦU CHU TRÌNH LẶP CỦA THUẬT TOÁN ABC
# ====================================

for iteration in range(1, MAX_ITER + 1):

    print(f"\n================ VÒNG LẶP {iteration} ================")

    # ---------------------------------
    # Giai đoạn 2: Pha của ong thợ
    # ---------------------------------
    print("\n--- Giai đoạn 2: Ong thợ (Employed Bees) ---")

    for i in range(SN):
        v = generate_new_solution(i)
        f_new = objective(v)
        fit_new = fitness_func(f_new)

        f_old = objective(food_sources[i])
        fit_old = fitness_func(f_old)

        print(f"Ong thợ {i}:")
        print(f"  Nghiệm cũ X{i} = {food_sources[i]}, fitness = {fit_old:.4f}")
        print(f"  Nghiệm mới V{i} = {v}, fitness = {fit_new:.4f}")

        if fit_new > fit_old:
            food_sources[i] = v
            fitness[i] = fit_new
            trial[i] = 0
            print("  → Cập nhật nghiệm mới (tốt hơn)")
        else:
            trial[i] += 1
            print("  → Giữ nghiệm cũ (không cải thiện)")
```

Hình 6: Đoạn mã: Pha của Ong thợ

Tại vòng lặp đầu tiên, mỗi ong thợ sẽ cố gắng cải thiện nguồn thức ăn mà nó phụ trách bằng cách tạo ra một nghiệm mới trong vùng lân cận (neighbourhood search).

*** Ong thợ thứ nhất** (x_0)

1. Chọn ngẫu nhiên chỉ số chiều cần cập nhật. Giả sử chiều được chọn là $k = 1$ (tương ứng với $x_{0,1}$).

2. Chọn ngẫu nhiên một nguồn thức ăn khác để tạo nhiễu. Giả sử nguồn được chọn là $j = 1$ (tức x_1).

3. Sinh giá trị nhiễu ngẫu nhiên:

$$\Phi \in [-1, 1], \qquad \Phi = 0.2.$$

4. Tạo nghiệm ứng viên theo công thức:

$$v_{0,k} = x_{0,k} + \Phi(x_{0,k} - x_{1,k}).$$

Cụ thể:

$$v_{0,1} = 1.91321764 + 0.2 \times (1.91321764 - 2.35042331) = 1.92406466.$$

Các chiều khác giữ nguyên:

$$v_{0,0} = x_{0,0} = 1.07114883.$$

Do đó nghiệm mới là:

$$V_0 = (1.07114883,\ 1.92406466).$$

5. Tính giá trị hàm mục tiêu:

$$f(V_0) = (1.07114883 - 1)^2 + (1.92406466 - 2)^2 + 3.$$

6. Chuyển đổi sang fitness:

$$fit(V_0) = 0.2493.$$

Fitness của nghiệm hiện tại:

$$fit(x_0) = 0.2492.$$

So sánh: Vì $fit(V_0) > fit(x_0)$ nên nghiệm mới tốt hơn nghiệm cũ.
Do đó cập nhật:

$$x_0 \leftarrow V_0 = (1.07114883,\ 1.92406466).$$

Đặt bộ đếm của nguồn này về 0:

$$C(0) = 0.$$

```
================ VÒNG LẶP 1 ================

--- Giai đoạn 2: Ong thợ (Employed Bees) ---
Ong thợ 0:
  Nghiệm cũ X0 = [1.07114883 1.91321764], fitness = 0.2492
  Nghiệm mới V0 = [1.07114883 1.92406466], fitness = 0.2493
  → Cập nhật nghiệm mới (tốt hơn)
```

*** Ong thợ thứ hai (x_1) 1.** Chọn ngẫu nhiên chỉ số chiều cần cập nhật. Giả sử chiều được chọn là $k = 1$ (tức $x_{1,1}$).

2. Chọn ngẫu nhiên một nguồn thức ăn khác để tạo nhiễu. Giả sử nguồn được chọn là $j = 0$ (tức x_0).

3. Sinh giá trị nhiễu ngẫu nhiên:

$$\Phi \in [-1, 1], \qquad \Phi = 0.3.$$

4. Tạo nghiệm ứng viên theo công thức:

$$v_{1,k} = x_{1,k} + \Phi(x_{1,k} - x_{0,k}).$$

Cụ thể:

$$v_{1,1} = 2.35042331 + 0.3 \times (2.35042331 - 1.91321764) = 2.04691443.$$

Giữ nguyên chiều còn lại: $v_{1,0} = x_{1,0} = 0.76298546$.
Suy ra nghiệm mới: $V_1 = (0.76298546,\ 2.04691443)$.

5. Tính giá trị hàm mục tiêu:

$$f(V_1) = (0.76298546 - 1)^2 + (2.04691443 - 2)^2 + 3.$$

6. Chuyển đổi sang fitness:

$$fit(V_1) = 0.2464.$$

Fitness của nghiệm hiện tại: $fit(x_1) = 0.2393$.

So sánh: Vì $fit(V_1) > fit(x_1)$ nên nghiệm mới tốt hơn nghiệm cũ. Giữ lại nghiệm mới:

$$x_1 = (0.76298546,\ 2.04691443).$$

Đặt bộ đếm:

$$C(1) = 0.$$

```
Ong thợ 1:
  Nghiệm cũ X1 = [0.76298546 2.35042331], fitness = 0.2393
  Nghiệm mới V1 = [0.76298546 2.04691443], fitness = 0.2464
  → Cập nhật nghiệm mới (tốt hơn)
```

*** Ong thợ thứ ba (x_2) 1.** Chọn ngẫu nhiên chỉ số chiều cần cập nhật. Giả sử chiều được chọn là $k = 0$ (tức $x_{2,0}$).

2. Chọn ngẫu nhiên một nguồn thức ăn khác để tạo nhiễu. Giả sử nguồn được chọn là $j = 1$ (tức x_1).

3. Sinh giá trị nhiễu ngẫu nhiên:

$$\Phi \in [-1, 1], \qquad \Phi = 0.5.$$

4. Tạo nghiệm ứng viên theo công thức:

$$v_{2,k} = x_{2,k} + \Phi(x_{2,k} - x_{1,k}).$$

Cụ thể:

$$v_{2,0} = 0.80818979 + 0.5 \times (0.80818979 - 0.76298546) = 0.81159504.$$

Giữ nguyên chiều còn lại: $v_{2,1} = 1.89944192$.

Suy ra nghiệm mới: $V_2 = (0.81159504,\ 1.89944192)$. **5.** Tính giá trị hàm mục tiêu:

$$f(V_2) = (0.81159504 - 1)^2 + (1.89944192 - 2)^2 + 3.$$

6. Chuyển đổi sang fitness:

$$fit(V_2) = 0.2472.$$

Fitness nghiệm hiện tại: $fit(x_2) = 0.2471$.

So sánh: Do $fit(V_2) > fit(x_2)$ nên nghiệm mới tốt hơn nghiệm cũ. Giữ lại nghiệm mới:

$$x_2 = (0.81159504,\ 1.89944192).$$

Đặt bộ đếm:

$$C(2) = 0.$$

```
Ong thợ 2:
  Nghiệm cũ X2 = [0.80818979 1.89944192], fitness = 0.2471
  Nghiệm mới V2 = [0.81159504 1.89944192], fitness = 0.2472
  → Cập nhật nghiệm mới (tốt hơn)
```

Bước 4: Pha ong quan sát

```
# ---------------------------------
# Giai đoạn 3: Ong quan sát
# ---------------------------------
print("\n--- Giai đoạn 3: Ong quan sát (Onlooker Bees) ---")

prob = fitness / np.sum(fitness)
for _ in range(SN):
    i = np.random.choice(range(SN), p=prob)
    v = generate_new_solution(i)

    f_new = objective(v)
    fit_new = fitness_func(f_new)
    f_old = objective(food_sources[i])
    fit_old = fitness_func(f_old)

    print(f"Ong quan sát chọn nguồn {i}:")
    print(f"  V{i} = {v}, fitness = {fit_new:.4f}")

    if fit_new > fit_old:
        food_sources[i] = v
        fitness[i] = fit_new
        trial[i] = 0
        print("  → Cập nhật nghiệm mới")
    else:
        trial[i] += 1
        print("  → Không cập nhật")
```

Hình 7: Đoạn mã: Pha của Ong quan sát

Tính toán xác suất lựa chọn các giải pháp dựa trên tất cả các giải pháp của ong thợ:

$$P_i = \frac{fit_i}{\sum_{j=0}^{2} fit_j}$$

Xác suất để các ong quan sát chọn giải pháp là:

$$P_0 = \frac{fit(X_0)}{\sum_{j=0}^{2} fit(X_j)} = \frac{0.2493}{0.2493 + 0.2464 + 0.2472} = 0.3360$$

$$P_1 = \frac{fit(X_1)}{\sum_{j=0}^{2} fit(X_j)} = \frac{0.2464}{0.2493 + 0.2464 + 0.2472} = 0.3317$$

$$P_2 = \frac{fit(X_2)}{\sum_{j=0}^{2} fit(X_j)} = \frac{0.2472}{0.2493 + 0.2464 + 0.2472} = 0.3328$$

Các ong quan sát sinh giải pháp mới

*** Ong quan sát thứ nhất:** Ong quan sát thứ nhất sẽ chọn ngẫu nhiên theo xác suất giải pháp $P_i\,(i = 0, 2)$. Giả sử chọn được $x_1 = (0.76298546,\ 2.04691443)$.

- Chọn ngẫu nhiên chỉ số chiều cần cập nhật: $k = 0$.
- Chọn ngẫu nhiên hàng xóm: $j = 0$.
- Chọn giá trị ngẫu nhiên: $\phi = -0.3$.

Khi đó nghiệm mới:

$$v_{1,0} = x_{1,0} + \phi(x_{1,0} - x_{0,0}) = 0.76298546 - 0.3(0.76298546 - 1.07114883) = 0.74093495$$

$$v_{1,1} = x_{1,1} = 2.04691443$$

Do đó nghiệm mới:

$$V_1 = (0.74093495,\ 2.04691443)$$

Tính giá trị hàm mục tiêu:

$$f(V_1) = (0.74093495 - 1)^2 + (2.04691443 - 2)^2 + 3 = 3.0696$$

Độ thích nghi:

$$fit(V_1) = \frac{1}{1 + 3.0696} = 0.2457$$

Thực hiện kiểm tra:

$$fit(V_1) = 0.2457 < fit(x_1) = 0.2464$$

Do đó **không cập nhật** nghiệm.

$$x_1 = (0.76298546,\ 2.04691443)$$

Bộ đếm:

$$C(1) = 1$$

```
--- Giai đoạn 3: Ong quan sát (Onlooker Bees) ---
Ong quan sát chọn nguồn 1:
  V1 = [0.74093495 2.04691443], fitness = 0.2457
  → Không cập nhật
```

*** Ong quan sát thứ hai:** Giả sử ong quan sát thứ hai cũng chọn $x_1 = (0.76298546, 2.04691443)$.

- Chọn chiều cập nhật: $k = 0$.
- Chọn hàng xóm: $j = 2$.
- Chọn $\phi = 0.2$.

$$v_{1,0} = 0.76298546 + 0.2(0.76298546 - 0.81159504) = 0.76757082$$

$$v_{1,1} = 2.04691443$$

$$V_1 = (0.76757082,\ 2.04691443)$$

$$f(V_1) = (0.76757082 - 1)^2 + (2.04691443 - 2)^2 + 3 = 3.0562$$

$$fit(V_1) = \frac{1}{1 + 3.0562} = 0.2465$$

$$fit(V_1) = 0.2465 > fit(x_1) = 0.2464$$

Vì vậy ta **cập nhật nghiệm mới**:

$$x_1 = (0.76757082,\ 2.04691443)$$

Bộ đếm:

$$C(1) = 0$$

```
Ong quan sát chọn nguồn 1:
  V1 = [0.76757082 2.04691443], fitness = 0.2465
  → Cập nhật nghiệm mới
```

*** Ong quan sát thứ ba:** Giả sử chọn được $x_0 = (1.07114883,\ 1.92406466)$.

- Chọn chiều cập nhật: $k = 1$.
- Chọn hàng xóm: $j = 1$.
- Chọn $\phi = -0.3$.

$$v_{0,1} = 1.92406466 - 0.3(1.92406466 - 2.04691443) = 1.90558396$$

$$v_{0,0} = 1.07114883$$

$$V_0 = (1.07114883,\ 1.90558396)$$

$$f(V_0) = (1.07114883 - 1)^2 + (1.90558396 - 2)^2 + 3 = 3.0136$$

$$fit(V_0) = \frac{1}{1 + 3.0136} = 0.2491$$

$$fit(V_0) = 0.2491 < fit(x_0) = 0.2493$$

Do đó nghiệm **không được cập nhật**:

$$x_0 = (1.07114883,\ 1.92406466)$$

Bộ đếm:

$$C(0) = 1$$

```
Ong quan sát chọn nguồn 0:
  V0 = [1.07114883 1.90558396], fitness = 0.2491
  → Không cập nhật
```

Bước 5: Pha ong do thám

```
# ---------------------------------
# Giai đoạn 4: Ong do thám
# ---------------------------------
print("\n--- Giai đoạn 4: Ong do thám (Scout Bees) ---")

for i in range(SN):
    if trial[i] > LIMIT:
        print(f"  Nguồn {i} bị cạn kiệt → sinh nguồn mới")
        for j in range(D):
            min_j, max_j = bounds[j]
            food_sources[i][j] = min_j + np.random.rand() * (max_j - min_j)
        trial[i] = 0
        fitness[i] = fitness_func(objective(food_sources[i]))
```

Hình 8: Pha Ong do thám

Duyệt toàn bộ các giải pháp để lấy ra bộ đếm tương ứng với các giải pháp đó. Nhận thấy:

$$C(0) = 0 \ < \ L = 6;$$
$$C(1) = 2 \ < \ L = 6;$$
$$C(2) = 1 \ < \ L = 6;$$

Kết luận Nên các giải pháp hiện tại không cần phải thay thế là:

$$x_0 = (1.07114883,\ 1.92406466), \qquad fit(x_0) = 0.2493$$
$$x_1 = (0.76298546,\ 2.04691443), \qquad fit(x_1) = 0.2393$$
$$x_2 = (0.80818979,\ 1.89944192), \qquad fit(x_2) = 0.2471$$

```
# ---------------------------------
# Cập nhật nghiệm tốt nhất
# ---------------------------------
best_index = np.argmax(fitness)
X_best = food_sources[best_index].copy()
print("\n→ Nghiệm tốt nhất hiện tại:", X_best)
print("→ Fitness:", fitness[best_index])
```

Hình 9: Cập nhật lại nghiệm tốt nhất sau mỗi lần lặp

Do đó nghiệm tốt nhất là:

$$x_{\text{best}} = (1.07114883,\ 1.92406466)$$

Sau khi thực hiện 50 vòng lặp, ta thu được các giá trị hội tụ ở $(0.99967539,\ 2.00014272)$.

1.7 Ưu điểm và nhược điểm của giải thuật bầy ong nhân tạo

1.7.1 Ưu điểm

- ABC duy trì nhiều bước thăm dò nên tiến độ tối ưu hơn các thuật toán PSO và DE
- ABC có khả năng thoát bẫy cục bộ tốt do có cơ chế ong do thám tự động bỏ nghiệm kém và tìm vùng mới
- ABC thường cho nghiệm cuối ổn định và ít dao động qua nhiều lần chạy
- ABC có tính bền vững hơn trong các bài toán nhiễu hoặc nhiều cực trị

1.7.2 Nhược điểm

- Tốc độ hội tụ của ABC chậm hơn PSA và DE do phải trải qua ba pha : ong thợ -> ong quan sát -> ong do thám
- Phụ thuộc vào tham số limit, nếu limit quá nhỏ dẫn đến thuật toán không được khai thác tốt.
- Yêu cầu nhiều lần đánh giá hàm mục tiêu khiến giảm hiệu quả trên các bài toán đơn giản hoặc có kích thước lớn

2 Các biến thể của Thuật toán Bầy ong Nhân tạo - ABC

2.1 Thuật toán Gbest-guided ABC (GABC)

2.1.1 Ý tưởng thay đổi và mục tiêu cải thiện

Mục tiêu của GABC là **tăng khả năng khai thác cục bộ và tốc độ hội tụ** của ABC gốc. Quan sát rằng ABC gốc chỉ sử dụng thông tin từ nghiệm hiện tại x_i và một nghiệm ngẫu nhiên x_k, nên hướng dịch chuyển mang tính ngẫu nhiên cao và thiếu định hướng rõ rệt, ngay cả khi trong quần thể đã có một nghiệm rất tốt.

Ý tưởng chính của GABC là:

Thêm một lực "kéo" các nghiệm về gần nghiệm tốt nhất toàn đàn g.

Nhờ vậy, thay vì di chuyển ngẫu nhiên, các nghiệm sẽ có xu hướng tiến gần hơn tới vùng đang được đánh giá là tốt nhất, rút ngắn thời gian tìm kiếm.

2.1.2 Thay đổi so với ABC gốc

GABC chỉ sửa **giai đoạn employed (và theo đó là onlooker)**. Công thức cập nhật trở thành:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(g_j - x_{ij}),$$

trong đó:

- g là nghiệm tốt nhất hiện tại trong quần thể,
- ψ_{ij} là hệ số ngẫu nhiên (ví dụ trong $[0, 1]$) điều khiển mức độ "kéo" về nghiệm tốt nhất. Hiện nay đề xuất mở rộng phạm vi lên đến 1.5.

Ở đây:

- thành phần $\phi_{ij}(x_{ij} - x_{kj})$ giữ vai trò như ABC gốc, giúp duy trì tính đa dạng và khám phá,
- thành phần $\psi_{ij}(g_j - x_{ij})$ bổ sung xu hướng hội tụ về nghiệm tốt nhất.

2.1.3 Ưu điểm và hạn chế của GABC

Ưu điểm

- Tăng khả năng khai thác cục bộ quanh nghiệm tốt nhất.
- Tốc độ hội tụ nhanh hơn ABC gốc (ít vòng lặp hơn để đạt nghiệm tốt).
- Phù hợp với các bài toán tối ưu liên tục, hàm mục tiêu trơn và ít cực trị cục bộ.

Hạn chế

- Nếu hệ số ψ_{ij} quá lớn, thuật toán có thể hội tụ quá sớm, dễ bị kẹt vào cực trị cục bộ.
- Tính đa dạng quần thể có thể giảm dần theo thời gian nếu không có cơ chế bù trừ.

2.1.4 Mã giả thuật toán GABC

Algorithm 2 Gbest-guided ABC (GABC)

Khởi tạo quần thể nguồn thức ăn x_i giống ABC gốc.
Đánh giá $f(x_i)$ và xác định nghiệm tốt nhất g.
for iter = 1 đến MAX_ITER **do**
  Cập nhật lại nghiệm tốt nhất g trong quần thể.
  Employed Bee Phase:
  for mỗi nguồn thức ăn x_i **do**
    Chọn $k \neq i$ và một chiều j.
    Sinh nghiệm mới:
      $v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(g_j - x_{ij})$.
    Giữ các thành phần khác: $v_{il} = x_{il}$, $l \neq j$.
    Nếu $f(v_i)$ tốt hơn $f(x_i)$ thì $x_i \leftarrow v_i$, ngược lại tăng bộ đếm bỏ rơi.
  end for
  Onlooker Bee Phase:
  Tính xác suất p_i từ fitness của x_i.
  for mỗi onlooker **do**
    Chọn nguồn x_i theo xác suất p_i.
    Chọn $k \neq i$ và chiều j.
    Sinh nghiệm mới v_i bằng công thức GABC như trên.
    Cập nhật x_i nếu v_i tốt hơn.
  end for
  Scout Bee Phase: giống ABC gốc.
end for
Trả về nghiệm tốt nhất g.

2.2 Thuật toán Quick ABC (qABC)

2.2.1 Ý tưởng thay đổi và mục tiêu cải thiện

Mục tiêu của qABC là **tăng tốc độ hội tụ** và **giảm số lần đánh giá hàm mục tiêu** so với ABC gốc. Trong ABC truyền thống, việc tạo nghiệm mới dựa trên chênh lệch giữa hai nguồn thức ăn ngẫu nhiên khiến hướng tìm kiếm còn hạn chế và tốc độ cải thiện nghiệm chậm.

Ý tưởng cốt lõi của qABC là:

> **Tập trung tìm kiếm xung quanh nghiệm tốt nhất hiện tại để rút ngắn quá trình hội tụ và giảm số nghiệm cần cập nhật.**

Cơ chế này cho phép thuật toán khai thác tốt hơn vùng quanh nghiệm tốt, rút ngắn thời gian tối ưu và giảm số phép đánh giá hàm không cần thiết.

2.2.2 Thay đổi so với ABC gốc

Trong qABC, cách tạo nghiệm mới được điều chỉnh theo hướng dẫn của nghiệm tốt nhất. Thay vì dùng:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}),$$

qABC sử dụng chiến lược *best–guided*:

$$v_{ij} = x_{ij} + \phi_{ij}(x_j^{best} - x_{ij}),$$

trong đó:

- x^{best} là nghiệm tốt nhất hiện tại,
- ϕ_{ij} là số ngẫu nhiên trong khoảng $[-1, 1]$,
- cập nhật diễn ra chủ yếu quanh nghiệm tốt nhằm tăng tốc hội tụ.

Ngoài ra, qABC **rút gọn pha Onlooker**, chỉ thực hiện với một số ít nghiệm có xác suất cao, nhằm giảm số lần cập nhật và giảm chi phí tính toán.

2.2.3 Ưu điểm và hạn chế của qABC

Ưu điểm

- Tốc độ hội tụ nhanh hơn ABC và GABC nhờ tìm kiếm có định hướng.
- Giảm số lượng đánh giá hàm mục tiêu.
- Phù hợp với bài toán đơn giản hoặc yêu cầu tốc độ.
- Giữ cấu trúc ABC gốc, dễ cài đặt.

Hạn chế

- Khả năng khám phá toàn cục kém hơn ABC gốc và QABC.
- Dễ rơi vào cực trị cục bộ do phụ thuộc mạnh vào nghiệm tốt nhất.
- Không phù hợp với bài toán nhiều cực trị phức tạp.

2.2.4 Mã giả thuật toán qABC

Algorithm 3 Quick ABC (qABC)

Khởi tạo quần thể nguồn thức ăn x_i như ABC gốc.
Đánh giá $f(x_i)$ và tìm nghiệm tốt nhất ban đầu x^{best}.
for iter = 1 đến MAX_ITER **do**
  Employed Bee Phase (Best-guided):
  for mỗi nguồn thức ăn x_i **do**
    Chọn ngẫu nhiên một chiều j.
    Sinh $\phi_{ij} \sim U[-1, 1]$.
    Cập nhật nghiệm mới theo hướng nghiệm tốt nhất:

$$v_{ij} = x_{ij} + \phi_{ij}(x_j^{best} - x_{ij})$$

    Giữ các chiều khác: $v_{il} = x_{il},\ l \neq j$.
    Nếu $f(v_i)$ tốt hơn $f(x_i)$ thì cập nhật $x_i \leftarrow v_i$.
  end for
  Cập nhật nghiệm tốt nhất x^{best}.
  Onlooker Bee Phase rút gọn:
  Tính xác suất chọn nguồn thức ăn từ fitness.
  for một số onlooker được chọn **do**
    Chọn một nguồn thức ăn x_i theo xác suất.
    Sinh $\phi_{ij} \sim U[-1, 1]$ và cập nhật giống Employed Phase.
    Nếu cải thiện, thay thế nghiệm hiện tại.
  end for
  Scout Bee Phase: giữ nguyên như ABC gốc.
end for
Trả về nghiệm tốt nhất x^{best} tìm được.

2.3 Thuật toán Chaotic ABC (CABC)

2.3.1 Ý tưởng thay đổi và mục tiêu cải thiện

CABC được đề xuất với mục tiêu **tăng tính đa dạng quần thể và khả năng khám phá không gian nghiệm**, từ đó giảm hiện tượng hội tụ sớm vào cực trị cục bộ.

Thay vì sử dụng các số ngẫu nhiên độc lập như trong ABC gốc, CABC dùng **chuỗi hỗn loạn** sinh bởi *logistic map*:

$$z_{t+1} = \mu z_t(1 - z_t), \quad \mu \approx 4,\ z_t \in (0, 1).$$

Chuỗi này có tính:

- phân bố đều trên $(0, 1)$ hơn so với random đơn thuần,
- nhạy cảm với điều kiện ban đầu (small change in $z_0 \to$ big change in sequence),
- không tuần hoàn trong phạm vi quan sát.

Nhờ đó, nghiệm được sinh ra "rải đều" hơn trong không gian tìm kiếm, giúp tăng khả năng khám phá những vùng mới.

2.3.2 Thay đổi so với ABC gốc

CABC thay đổi ở ba vị trí chính:

1. **Khởi tạo quần thể:** sử dụng z_t để sinh toạ độ:

   $$x_{ij} = x_j^{\min} + z_t\big(x_j^{\max} - x_j^{\min}\big),$$

   giúp các nghiệm ban đầu phân bố trải đều hơn trong miền tìm kiếm.

2. **Giai đoạn employed / onlooker:** dùng z_t thay cho ϕ_{ij}:

   $$v_{ij} = x_{ij} + z_t(x_{ij} - x_{kj}),$$

   để điều chỉnh bước nhảy theo chuỗi hỗn loạn thay vì random độc lập.

3. **Scout**: khi khởi tạo lại nguồn bị bỏ rơi cũng dùng logistic map thay vì random thuần túy, đảm bảo nguồn mới vẫn mang tính hỗn loạn.

2.3.3 Ưu điểm và hạn chế của CABC

Ưu điểm

- Giữ được tính đa dạng quần thể tốt hơn, đặc biệt khi số vòng lặp lớn.
- Tăng khả năng khám phá những vùng mới trong không gian nghiệm.
- Giảm khả năng bị hội tụ sớm vào một cực trị cục bộ.

Hạn chế

- Đối với các bài toán đơn giản, chỉ có một cực trị toàn cục (hàm lồi), hiệu quả của hỗn loạn không thể hiện rõ so với ABC gốc.
- Phụ thuộc vào lựa chọn tham số μ và giá trị khởi tạo z_0.

2.3.4 Mã giả thuật toán CABC

Algorithm 4 Chaotic ABC (CABC)

Chọn giá trị ban đầu $z_0 \in (0,1)$ và tham số $\mu = 4$.
Khởi tạo quần thể bằng hỗn loạn:
for mỗi nguồn thức ăn x_i **do**
  for mỗi chiều j **do**
    Cập nhật $z_{t+1} = \mu z_t(1 - z_t)$.
    Đặt $x_{ij} = x_j^{\min} + z_{t+1}\big(x_j^{\max} - x_j^{\min}\big)$.
  end for
end for
Đánh giá $f(x_i)$ và tính fitness.
for iter = 1 đến MAX_ITER **do**
  Employed Bee Phase:
  for mỗi nguồn thức ăn x_i **do**
    Chọn $k \neq i$ và một chiều j.
    Cập nhật $z_{t+1} = \mu z_t(1 - z_t)$.
    Sinh nghiệm mới:
      $v_{ij} = x_{ij} + z_{t+1}(x_{ij} - x_{kj})$.
    Giữ $v_{il} = x_{il}$ với $l \neq j$.
    Nếu $f(v_i)$ tốt hơn $f(x_i)$ thì $x_i \leftarrow v_i$, ngược lại tăng bộ đếm bỏ rơi.
  end for
  Onlooker Bee Phase:
  Tính xác suất p_i từ fitness.
  for mỗi onlooker **do**
    Chọn nguồn x_i theo p_i.
    Chọn $k \neq i$, chiều j.
    Cập nhật $z_{t+1} = \mu z_t(1 - z_t)$.
    Sinh nghiệm mới v_i bằng công thức CABC.
    Cập nhật x_i nếu v_i tốt hơn.
  end for
  Scout Bee Phase:
  for mỗi nguồn x_i **do**
    if số lần bị bỏ rơi của x_i vượt LIMIT **then**
      for mỗi chiều j **do**
        Cập nhật $z_{t+1} = \mu z_t(1 - z_t)$.
        Đặt $x_{ij} = x_j^{\min} + z_{t+1}\big(x_j^{\max} - x_j^{\min}\big)$.
      end for
    end if
  end for
end for
Trả về nghiệm tốt nhất tìm được.

3 Nhận xét và so sánh tổng hợp

Trong phần này, ta tóm tắt lại các đặc điểm chính của ba thuật toán **ABC gốc**, **GABC** và **CABC** dưới góc nhìn: cơ chế tìm kiếm, điểm mạnh, hạn chế và dạng bài toán phù hợp. Mục tiêu là giúp nhìn nhanh được vai trò của từng thuật toán trong thực hành tối ưu hoá.

3.1 Áp dụng vào bài toán thực tế

chúng em áp dụng thuật toán Bầy ong nhân tạo (ABC) để giải bài toán tối ưu không ràng buộc sau:

$$\min_{\mathbf{x}\in D} f(\mathbf{x}), \qquad \mathbf{x} = (x_1, x_2)$$

Hàm mục tiêu được cho bởi:

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^2 + 3.$$

Miền tìm kiếm là hình chữ nhật hai chiều:

$$D = \left\{ (x_1, x_2) \in \mathbb{R}^2 \;\middle|\; 0.5 \le x_1 \le 1.5,\ 1.5 \le x_2 \le 2.5 \right\}.$$

Nếu không xét ràng buộc, hàm số đạt cực tiểu tại:

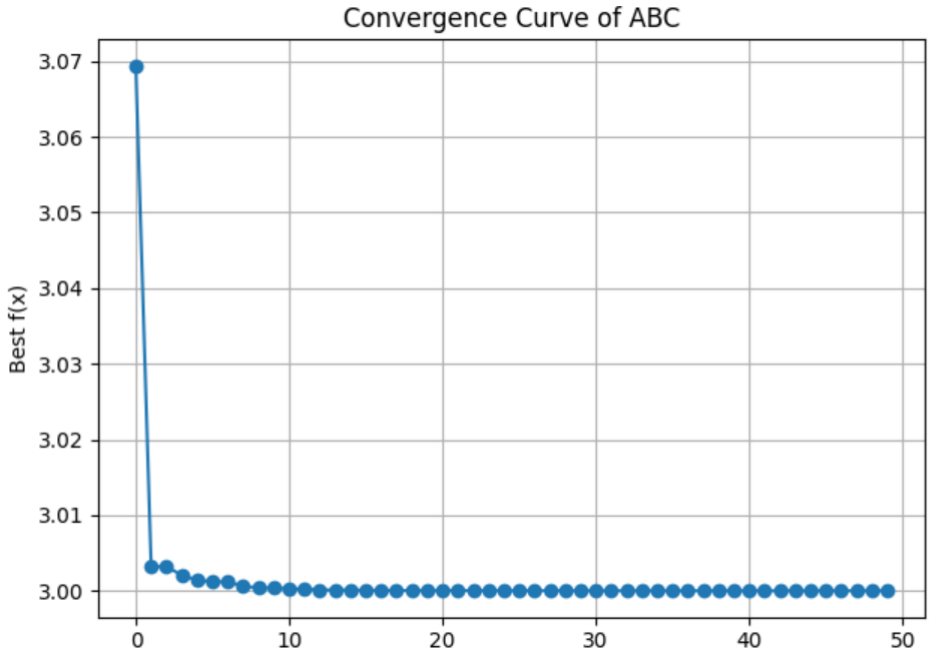
$$x_1^* = 1, \qquad x_2^* = 2$$

với giá trị nhỏ nhất:

$$f^* = 3.$$

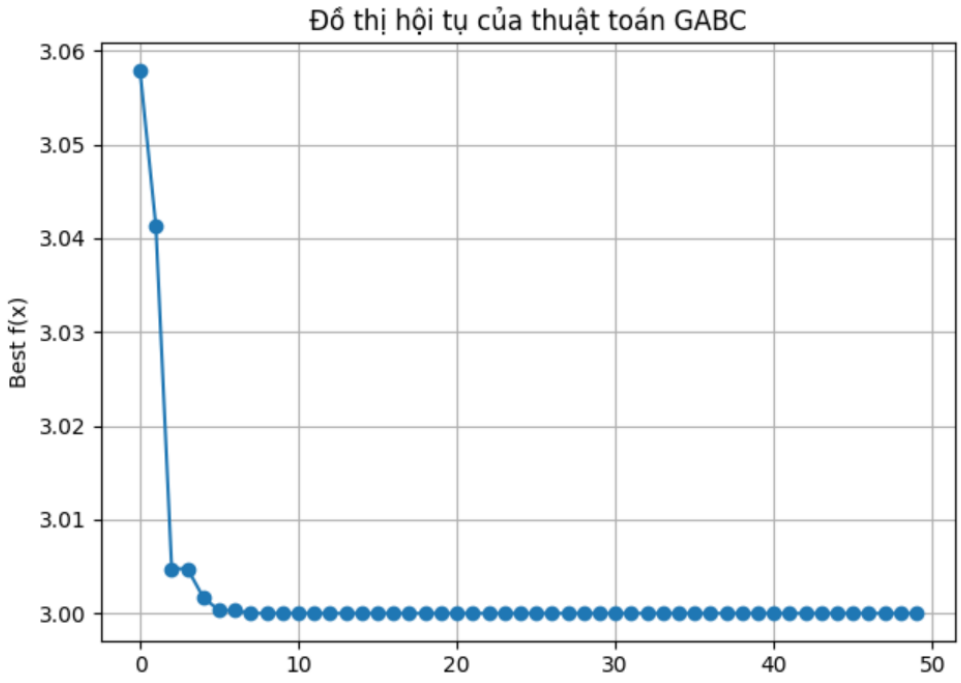
ABC:

```
→ Nghiệm tốt nhất hiện tại: [1.00387219 2.00001082]
→ Fitness: 0.24999906287990076
```



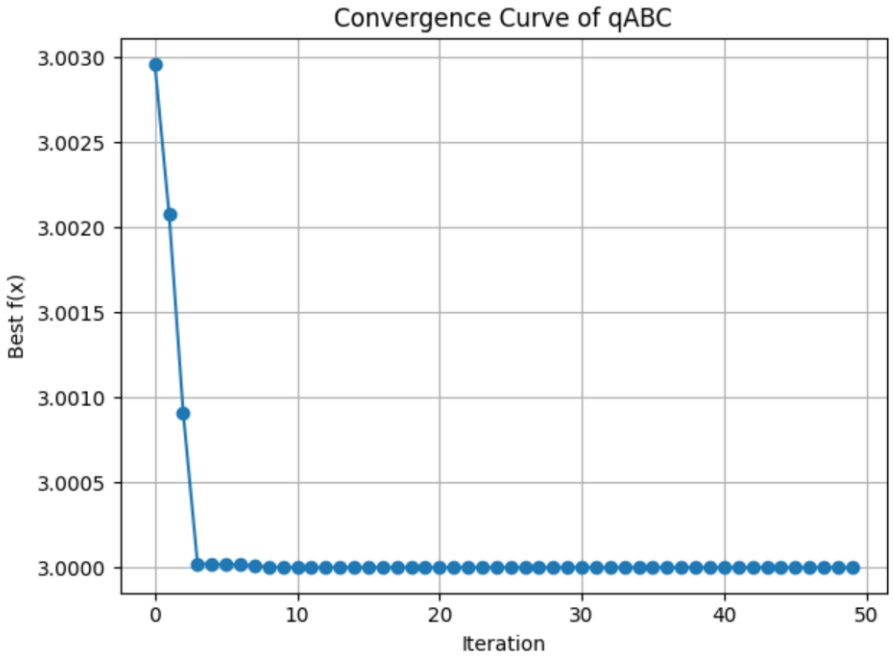
GABC:

```
=============== VÒNG LẶP 50 ===============

--- Ong thợ (Employed Bees) ---
Ong thợ 0: X0=[1.00003287 1.99321452], V0=[1.01011902 1.99321452]
  → Giữ nghiệm cũ
Ong thợ 1: X1=[0.58418211 2.30455135], V1=[0.58418211 2.07973741]
  → Cập nhật
Ong thợ 2: X2=[1.01209821 2.00218109], V2=[1.01209821 2.06702696]
  → Giữ nghiệm cũ

--- Ong quan sát (Onlooker) ---
Ong quan sát chọn 1: V=[0.58418211 2.00175833]
  → Cập nhật
Ong quan sát chọn 2: V=[1.01209821 2.00117278]
  → Cập nhật
Ong quan sát chọn 1: V=[0.52414239 2.00175833]

--- Ong do thám (Scout) ---

** Gbest hiện tại: [1.00003287 1.99321452]  | Fitness: 0.24999712229452428
```



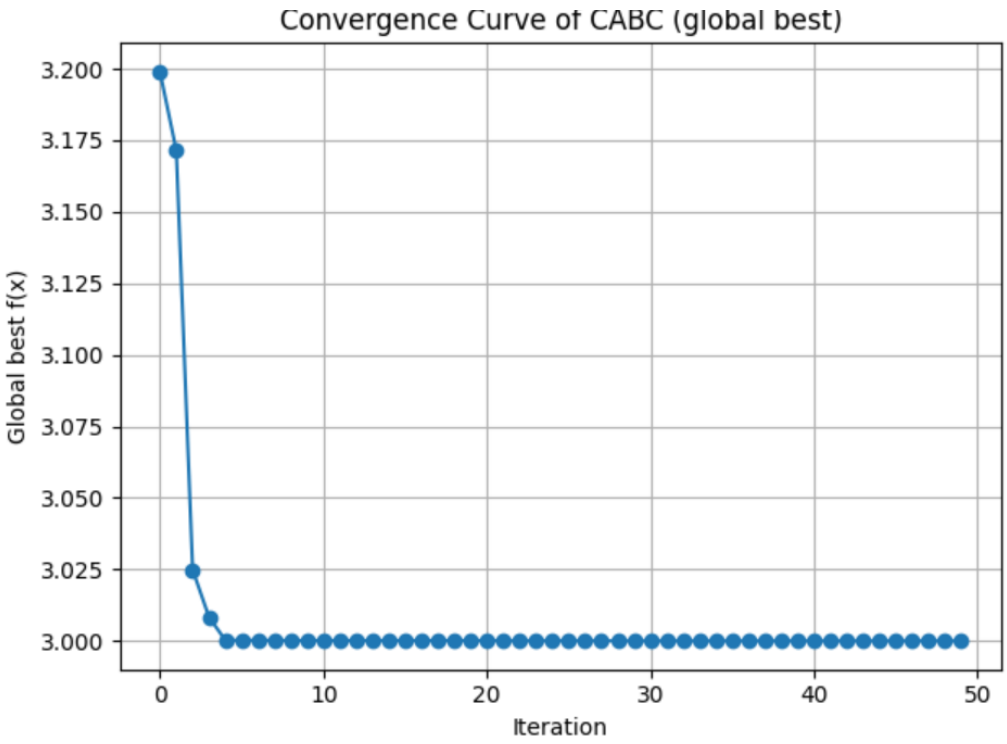
QABC:

```
=== KẾT QUẢ CUỐI CÙNG (qABC) ===
Gbest ≈ [1.0000061 2.0005936]
f(Gbest) ≈ 3.000000352399686
Thời gian chạy qABC: 0.038986 giây
```



CABC:

```
=============== VÒNG LẶP 50 ===============

--- Ong thợ ---
Ong thợ 0: X0=[0.99999291 1.99999957], V0=[0.99999291 1.73585264]
Ong thợ 1: X1=[0.50657848 1.52614082], V1=[0.59925195 1.52614082]
  → Cập nhật
Ong thợ 2: X2=[1.34544792 1.7824692 ], V2=[1.34544792 1.65212291]

--- Ong quan sát ---
Ong quan sát chọn 0: V=[0.86940612 1.99999957]
Ong quan sát chọn 1: V=[0.59925195 1.5       ]
Ong quan sát chọn 2: V=[1.5        1.7824692]

--- Ong do thám ---

→ Gbest: [0.99999291 1.99999957] | Fitness: 0.24999999999685085
```



Sự khác nhau về tốc độ hội tụ

- **ABC gốc**: Đồ thị hội tụ cho thấy thuật toán giảm giá trị hàm mục tiêu tương đối nhanh trong 3–5 vòng lặp đầu, sau đó tiến dần về giá trị tối ưu nhưng với tốc độ chậm hơn. ABC có khuynh hướng hội tụ ổn định nhưng không quá nhanh vì thiếu cơ chế định hướng trực tiếp về nghiệm tốt nhất.

- **GABC**: Thuật toán hội tụ nhanh hơn rõ rệt so với ABC gốc. Ngay từ các vòng đầu, nghiệm đã được kéo mạnh về vùng chứa nghiệm tối ưu nhờ thành phần dẫn hướng từ nghiệm tốt nhất toàn đàn (gbest). Đồ thị cho thấy giá trị $f(x)$ giảm nhanh và đạt gần tối ưu chỉ sau vài vòng lặp.

- **CABC**: Mặc dù hội tụ đến giá trị tối ưu, quá trình hội tụ diễn ra chậm hơn ABC và GABC ở một vài giai đoạn. Do đặc tính hỗn loạn, nghiệm có thể nhảy qua nhiều vùng khác nhau trong không gian tìm kiếm, tạo ra các dao động nhỏ trên đồ thị. Tuy nhiên, CABC vẫn thành công tránh hội tụ sớm và dần đạt nghiệm tốt về cuối quá trình lặp.

- **qABC**: Tốc độ hội tụ nhanh hơn ABC gốc nhờ tìm kiếm có định hướng quanh nghiệm tốt nhất. Tuy nhiên, do vẫn giữ một phần tính ngẫu nhiên để duy trì đa dạng, tốc độ hội tụ thường chậm hơn GABC. Đồ thị hội tụ cho thấy qABC tiến nhanh về vùng nghiệm tốt ở các vòng đầu, sau đó ổn định dần mà không có dao động mạnh.

Mức độ ổn định của nghiệm

- **ABC gốc**: Tương đối ổn định, sau khi tìm được vùng tốt thì dao động nhỏ và ít thay đổi.

- **GABC**: Ổn định nhất trong ba thuật toán. Một khi nghiệm đã gần tối ưu, thuật toán tập trung khai thác quanh nghiệm tốt nhất và không dao động nhiều.
- **CABC**: Dao động nhiều nhất. Hỗn loạn (chaos) giúp thuật toán tránh rơi vào cực trị cục bộ nhưng cũng làm cho nghiệm không ổn định trong một số vòng lặp.
- **qABC**: Ổn định hơn ABC và CABC nhờ cơ chế "best-guided search" hướng nghiệm về lời giải tốt nhất. Tuy nhiên, độ ổn định vẫn thấp hơn GABC do qABC vẫn duy trì một mức độ thăm dò nhất định. Khi thuật toán tiến gần nghiệm tối ưu, dao động giảm rõ rệt và nghiệm hội tụ ổn định.

Khả năng khám phá không gian nghiệm

- **ABC gốc**: Cân bằng giữa khám phá và khai thác. Tuy nhiên đôi khi gặp hiện tượng hội tụ sớm.
- **GABC**: Khả năng khám phá giảm do bị kéo mạnh về nghiệm tốt nhất. Dễ hội tụ nhanh nhưng đôi khi có nguy cơ mắc kẹt nếu bài toán có nhiều cực trị cục bộ.
- **CABC**: Khám phá mạnh nhất trong ba thuật toán nhờ nhiễu hỗn loạn logistic. Ít khi rơi vào cực trị cục bộ.
- **qABC**: Khả năng khám phá tốt hơn ABC gốc nhờ cập nhật dựa trên nghiệm tốt nhất kết hợp nhiễu ngẫu nhiên. qABC giúp quần thể tiếp tục tìm kiếm quanh vùng lân cận có triển vọng mà vẫn tránh bị kéo quá mạnh về một điểm như GABC. Tuy nhiên, khả năng khám phá không mạnh bằng CABC (chaotic) và phụ thuộc vào mức độ rút gọn của pha Onlooker.

3.2 So sánh tổng quan bốn thuật toán

Thuật toán	Ý tưởng chính	Điểm mạnh	Hạn chế / Bài toán phù hợp
ABC gốc	Mô phỏng đàn ong tìm nguồn thức ăn. Cập nhật nghiệm dựa trên chênh lệch giữa hai nghiệm bất kỳ và các bước nhảy ngẫu nhiên.	Cấu trúc đơn giản, dễ cài đặt; cân bằng tương đối giữa khám phá và khai thác; dùng làm baseline để so sánh với các biến thể khác.	Khai thác cục bộ còn yếu, hội tụ có thể chậm; dễ kẹt ở cực trị cục bộ trong bài toán phức tạp. Phù hợp cho bài toán đơn giản, hoặc dùng làm chuẩn so sánh.
GABC	Thêm thành phần hướng nghiệm về *nghiệm tốt nhất toàn đàn* g trong bước cập nhật, tạo lực "kéo" về vùng tốt.	Khai thác cục bộ mạnh; hội tụ nhanh hơn ABC gốc; cho nghiệm ổn định và chính xác hơn trong bài toán liên tục, hàm mục tiêu trơn.	Nếu tham số hướng về g quá lớn, dễ hội tụ sớm vào cực trị cục bộ; tính đa dạng có thể giảm dần. Phù hợp cho bài toán lồi, ít cực trị và cần hội tụ nhanh.
CABC	Thay số ngẫu nhiên bằng *chuỗi hỗn loạn* logistic trong khởi tạo, cập nhật và tái khởi tạo nguồn thức ăn, giúp nghiệm phân bố đều hơn.	Đa dạng quần thể cao; khả năng khám phá mạnh; giảm nguy cơ kẹt tại cực trị cục bộ; tốt cho bài toán nhiều cực trị.	Hiệu quả vượt trội không thể hiện rõ trên các bài toán đơn giản, chỉ có một cực trị; phụ thuộc vào lựa chọn tham số hỗn loạn. Phù hợp cho bài toán phi tuyến, đa đỉnh, mô hình thực tế phức tạp.
qABC	Rút gọn quá trình cập nhật; onlooker bees dùng cơ chế "best-guided search" dựa trên nghiệm tốt nhất để cải thiện tốc độ hội tụ.	Hội tụ nhanh hơn ABC gốc; số lần đánh giá hàm ít hơn; ổn định hơn CABC; phù hợp với bài toán yêu cầu tốc độ và độ chính xác vừa phải.	Khả năng khám phá không mạnh bằng CABC; dễ hội tụ sớm nếu nghiệm tốt nhất không đủ đa dạng. Thích hợp cho bài toán liên tục có số cực trị vừa phải.

Bảng 1: So sánh tổng quan giữa ABC gốc, GABC, CABC và qABC

3.3 Kết luận

Từ các phân tích và mô tả chi tiết có thể rút ra một số nhận xét chính:

- **ABC gốc** đóng vai trò là nền tảng: mô hình hóa trực quan hành vi đàn ong, dễ triển khai và mở rộng. Tuy nhiên, cơ chế cập nhật chỉ dựa trên chênh lệch ngẫu nhiên giữa hai nghiệm khiến thuật toán có thể hội tụ chậm và đôi khi thiếu định hướng.
- **GABC** cải thiện rõ rệt **khả năng khai thác** bằng cách tận dụng thông tin của nghiệm tốt nhất toàn đàn. Thành phần hướng về g giúp thuật toán hội tụ nhanh hơn và đạt nghiệm chất lượng tốt trong các bài toán tối ưu liên tục. Đây là lựa chọn phù hợp khi bài toán không quá nhiều cực trị cục bộ và mục tiêu là tìm nghiệm tốt trong thời gian ngắn.
- **CABC** tập trung tăng **khả năng khám phá** nhờ sử dụng chuỗi hỗn loạn. Việc thay random bằng logistic map giúp quần thể duy trì được mức đa dạng cao hơn, giảm khả năng kẹt tại một vùng nghiệm duy nhất. Thuật toán này đặc biệt hữu ích cho những bài toán khó, nhiều cực trị, hoặc khi mô hình truyền thông có nhiều tham số và cấu hình cần khảo sát.
- **qABC** trong phiên bản *Quick ABC* tập trung vào **tăng tốc hội tụ** thông qua cơ chế cập nhật có định hướng quanh nghiệm tốt nhất hiện tại. Bằng cách mô hình lại hành vi của onlooker bees và ưu tiên các nghiệm có chất lượng cao, qABC giảm số lần đánh giá hàm mục tiêu và rút ngắn quá trình tối ưu. Thuật toán phù hợp cho các bài toán liên tục với số cực trị vừa phải, hoặc khi yêu cầu đạt nghiệm tốt trong thời gian tính toán hạn chế.

Tóm lại, không có một biến thể nào là "tốt nhất trong mọi trường hợp":

- ABC gốc phù hợp làm chuẩn so sánh và cho các bài toán đơn giản.
- GABC là lựa chọn tốt khi ưu tiên tốc độ hội tụ và độ chính xác trên bài toán liên tục.
- CABC phù hợp khi cần khám phá không gian nghiệm rộng và đối mặt với nhiều cực trị cục bộ.
- qABC phù hợp cho các bài toán yêu cầu hội tụ nhanh, số lượng phép tính hạn chế, hoặc khi muốn giảm chi phí tính toán mà vẫn duy trì chất lượng nghiệm.

Việc lựa chọn thuật toán phù hợp cần dựa trên đặc trưng cụ thể của bài toán truyền thông và yêu cầu về độ chính xác, thời gian tính toán trong từng ứng dụng.

4 Mã nguồn

Toàn bộ mã nguồn dùng để thực hiện các thí nghiệm và vẽ đồ thị trong báo cáo được lưu trữ tại Google Colab :

Colab Notebook - ABC, GABC, qABC, CABC

Phần II Bài toán multibeam (beamforming) và áp dụng ABC để tối ưu

Ký hiệu và quy ước

- Chữ thường: vô hướng (scalar), ví dụ b_l, ρ.
- Chữ đậm thường: véc-tơ, ví dụ $\mathbf{w}_t, \mathbf{a}(\theta)$.
- Chữ đậm hoa: ma trận, ví dụ $\mathbf{H}$.
- $(\cdot)^T$: chuyển vị; $(\cdot)^H$: chuyển vị liên hợp; $(\cdot)^*$: liên hợp.
- $\|\cdot\|$: chuẩn Euclid (l_2).
- $\jmath$: đơn vị ảo.
- Miền pha: mặc định $\phi_i \in [-\pi, \pi]$ (pha liên tục), hoặc lượng tử theo số bit b.

1 Giới thiệu

1.1 Bối cảnh

Trong các hệ thống thông tin vô tuyến thế hệ mới, xu hướng tích hợp nhiều chức năng trên cùng một hạ tầng phần cứng ngày càng trở nên phổ biến. Một trong những hướng nghiên cứu quan trọng là tích hợp truyền thông và cảm biến, còn được gọi là Joint Communication and Sensing (JCAS).

Thay vì sử dụng hai hệ thống riêng biệt cho truyền dữ liệu và radar/sensing, JCAS cho phép dùng chung dàn anten và chuỗi RF, giúp giảm chi phí phần cứng và tăng hiệu quả sử dụng tài nguyên phổ tần. Tuy nhiên, việc thiết kế beamforming trong hệ thống JCAS trở nên phức tạp do yêu cầu khác nhau giữa truyền thông và sensing: truyền thông ưu tiên beam hẹp hướng người dùng, trong khi sensing thường yêu cầu beam rộng hoặc đa tia để quét không gian.

1.2 Động lực và mục tiêu nghiên cứu

Trong khi truyền thông cần chùm sóng hẹp, ổn định và hướng chính xác tới người dùng, sensing lại yêu cầu chùm sóng rộng hoặc nhiều tia để quét không gian. Việc đáp ứng đồng thời hai yêu cầu này trong điều kiện phần cứng anten analog (chủ yếu điều chỉnh pha, biên độ gần như cố định và pha có thể bị lượng tử hóa) đặt ra một bài toán tối ưu phi tuyến khó giải.

Mục tiêu của bài báo/bài tập là:

- Thiết kế véc-tơ beamforming đa tia phục vụ đồng thời truyền thông và sensing.
- Tối ưu các tham số kết hợp beam nhằm cực đại hóa công suất tín hiệu thu.

- Mô tả phương pháp lượng tử hóa véc-tơ beamforming phù hợp phần cứng analog (1 phase shifter, 2 phase shifters, IGSS-Q).
- Áp dụng metaheuristic (ABC và biến thể) để tìm nghiệm gần tối ưu cho bài toán không lồi, nhiều cực trị cục bộ.

2 Phát biểu bài toán tối ưu

2.1 Mô hình hệ thống và tín hiệu

2.1.1 Mảng anten tuyến tính đều (ULA) và véc-tơ hướng

Xét ULA gồm M phần tử, khoảng cách giữa hai phần tử kề nhau bằng nửa bước sóng. Véc-tơ hướng ứng với góc θ được viết:

$$\mathbf{a}(\theta) = \left[1, e^{\mathrm{J}\pi \sin(\theta)}, \ldots, e^{\mathrm{J}\pi(M-1)\sin(\theta)}\right]^T.$$

2.1.2 Mô hình kênh đa đường

Giả sử có L thành phần đa đường, mỗi đường có AoD $\theta_{t,l}$, AoA $\theta_{r,l}$, độ trễ τ_l và Doppler $f_{D,l}$. Kênh có thể biểu diễn dạng tổng:

$$\mathbf{H}(t) = \sum_{l=1}^{L} b_l \, \delta(t - \tau_l) \, e^{\mathrm{J}2\pi f_{D,l} t} \, \mathbf{a}(\theta_{t,l}) \mathbf{a}^T(\theta_{r,l}),$$

trong đó $b_l \in \mathbb{C}$ là hệ số suy hao (biên độ + pha) của đường truyền thứ l.

Trong phần còn lại của báo cáo, ta xét dạng ma trận kênh hiệu dụng $\mathbf{H}$ (bỏ qua biến thời gian để đơn giản hóa mô hình tối ưu).

2.1.3 Tín hiệu thu sau beamforming và SNR

Ký hiệu:

- $\mathbf{H} \in \mathbb{C}^{M \times M}$: ma trận kênh truyền.
- $\mathbf{w}_t \in \mathbb{C}^{M \times 1}$: véc-tơ beam phát.
- $\mathbf{w}_r \in \mathbb{C}^{M \times 1}$: véc-tơ beam thu.
- $s(t)$: tín hiệu phát; $\mathbf{z}(t)$: nhiễu Gauss trắng (AWGN) có phương sai σ_n^2.

Tín hiệu thu sau beamforming:

$$y(t) = \mathbf{w}_r^H \mathbf{H} \mathbf{w}_t \, s(t) + \mathbf{w}_r^H \mathbf{z}(t).$$

Nếu công suất tín hiệu phát là σ_s^2 và công suất nhiễu là σ_n^2, SNR là:

$$\gamma = \frac{|\mathbf{w}_r^H \mathbf{H} \mathbf{w}_t|^2}{\|\mathbf{w}_r\|^2} \cdot \frac{\sigma_s^2}{\sigma_n^2}.$$

Trong bài toán tối ưu, thành phần phụ thuộc trực tiếp vào beam là:

$$\frac{|\mathbf{w}_r^H \mathbf{H} \mathbf{w}_t|^2}{\|\mathbf{w}_r\|^2}.$$

2.2 Sinh hai beam con bằng Two-step ILS (Iterative Least Squares)

Để tạo hai beam con $\mathbf{w}_{t,c}$ (communication) và $\mathbf{w}_{t,s}$ (sensing), bài báo sử dụng thuật toán Two-step ILS. Ý tưởng là khớp đáp ứng mảng thực tế với đáp ứng mong muốn theo nghĩa bình phương tối thiểu.

2.2.1 Đáp ứng mong muốn và bài toán LS

Ký hiệu đáp ứng mong muốn:

$$\mathbf{v} = \mathbf{D}_v \mathbf{p}_v,$$

trong đó $\mathbf{D}_v$ là (ma trận) biên độ mong muốn và $\mathbf{p}_v$ là véc-tơ pha mong muốn. Gọi

$$\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_K]^T$$

với K là số hướng (directions) được lấy mẫu để phân tích.

Một phát biểu tối ưu điển hình theo pha:

$$\mathbf{p}_{v,\text{opt}} = \arg\min_{\mathbf{p}_v} \left\| (\mathbf{A}\mathbf{A}^\dagger - \mathbf{I}) \mathbf{D}_v \mathbf{p}_v \right\|_2^2,$$

trong đó $(\cdot)^\dagger$ là giả nghịch đảo Moore–Penrose.

2.2.2 Các bước Two-step ILS

Thuật toán lặp với $n = 1, 2, \ldots$:

- Bước 1 (cập nhật pha):
$$\mathbf{p}_{v,n} = \mathbf{w}_{n-1}^H \mathbf{v} \mathbf{D}_v^{-1}.$$
- Bước 2 (chiếu lên đường tròn đơn vị): chiếu từng phần tử của $\mathbf{p}_{v,n}$ về phần tử gần nhất trên đường tròn đơn vị để được $\mathbf{p}'_{v,n}$.
- Bước 3 (cập nhật beam):
$$\mathbf{w}_n = (\mathbf{v}\mathbf{v}^H)^{-1} \mathbf{v} \mathbf{D}_v \mathbf{p}'_{v,n}.$$

Sau khi hội tụ, thu được beam con phục vụ từng mục tiêu (communication/sensing) bằng cách chọn đáp ứng mong muốn tương ứng (ví dụ búp hẹp cho communication và búp quét cho sensing).

2.3 Kết hợp multibeam (đa biến)

Giả sử đã có sẵn hai beam con:

- $\mathbf{w}_{t,c}$: beam cho truyền thông (communication).
- $\mathbf{w}_{t,s}$: beam cho sensing.

Kết hợp đa tia theo từng phần tử (đa biến):

$$\mathbf{w}_t(\rho, \boldsymbol{\phi}) = \sqrt{\rho}\, \mathbf{w}_{t,c} + \sqrt{1-\rho} \begin{bmatrix} e^{\mathrm{j}\phi_1} w_{t,s,1} \\ e^{\mathrm{j}\phi_2} w_{t,s,2} \\ \vdots \\ e^{\mathrm{j}\phi_M} w_{t,s,M} \end{bmatrix}.$$

Giải thích các thành phần:

- $\rho \in [0, 1]$: hệ số chia công suất giữa hai beam; ρ càng lớn thì ưu tiên truyền thông càng mạnh.
- $\phi_i \in [-\pi, \pi]$: pha kết hợp tại phần tử anten thứ i.
- $e^{\mathrm{J}\phi_i}$: hệ số xoay pha phức (phasor) áp lên thành phần sensing tại phần tử anten thứ i.

Đầu vào bài toán gồm $\mathbf{H}$, $\mathbf{w}_{t,c}$, $\mathbf{w}_{t,s}$; đầu ra là bộ tham số tối ưu $(\rho^\star, \boldsymbol{\phi}^\star)$.

2.4 Biến cần tìm và ý nghĩa

Các biến tối ưu:

- ρ: quyết định mức độ ưu tiên giữa truyền thông và sensing.
- ϕ_i: điều chỉnh pha của beam sensing tại từng phần tử anten, ảnh hưởng trực tiếp đến sự cộng hưởng/triệt tiêu tín hiệu và do đó ảnh hưởng công suất thu.

2.5 Ý nghĩa hệ số cân bằng ρ và đánh đổi giữa truyền thông và sensing

Trong mô hình multibeam, $\rho \in [0, 1]$ quyết định cách chia công suất giữa hai thành phần trong vector phát $\mathbf{w}_t(\rho, \boldsymbol{\phi})$. Cụ thể, phần beam truyền thông được nhân với $\sqrt{\rho}$, còn phần beam sensing được nhân với $\sqrt{1-\rho}$. Vì vậy, khi thay đổi ρ thì mức độ ưu tiên giữa hai nhiệm vụ cũng thay đổi theo.

Khi ρ tiến gần 1, $\mathbf{w}_t$ sẽ gần với $\mathbf{w}_{t,c}$ nên hệ thống ưu tiên tăng hiệu năng truyền thông như độ lợi theo kênh và SNR tại máy thu. Ngược lại, khi ρ tiến gần 0, $\mathbf{w}_t$ sẽ gần với $\mathbf{w}_{t,s}$ nên hệ thống ưu tiên tạo búp sóng phù hợp cho sensing, ví dụ tăng độ lợi theo vùng hoặc hướng quét mong muốn.

Với mỗi giá trị ρ, ta có thể tối ưu các pha kết hợp $\boldsymbol{\phi}$ để đạt giá trị hàm mục tiêu tốt nhất:

$$f^\star(\rho) = \max_{\boldsymbol{\phi} \in [-\pi, \pi]^M} f(\rho, \boldsymbol{\phi}). \tag{1}$$

Đường cong $f^\star(\rho)$ phản ánh sự đánh đổi giữa truyền thông và sensing: tăng ρ sẽ giúp thiên về truyền thông, giảm ρ sẽ giúp thiên về sensing. Do đó, việc quan sát $\rho^\star \approx 0.5$ trong thực nghiệm là hợp lý trong trường hợp mong muốn cân bằng hai nhiệm vụ, tức là hai thành phần đóng góp tương đương và thuật toán tìm $\boldsymbol{\phi}$ để hai beam kết hợp hiệu quả mà không làm một nhiệm vụ lấn át nhiệm vụ còn lại.

3 Hàm mục tiêu

3.1 Hàm mục tiêu khi biết đầy đủ ma trận kênh H (Known Channel Matrix)

Với beam thu theo phương pháp Maximal Ratio Combining (MRC), chọn:

$$\mathbf{w}_r = (\mathbf{H}\mathbf{w}_t)^*.$$

Thay vào biểu thức SNR và rút gọn, ta thu được thước đo công suất chuẩn hóa (dạng Rayleigh quotient):

$$f(\rho, \boldsymbol{\phi}) = \frac{\mathbf{w}_t^H(\rho, \boldsymbol{\phi})\, \mathbf{H}^H \mathbf{H}\, \mathbf{w}_t(\rho, \boldsymbol{\phi})}{\|\mathbf{w}_t(\rho, \boldsymbol{\phi})\|^2}.$$

Giải thích:

- $\mathbf{H}^H\mathbf{H}$: ma trận "năng lượng kênh" (channel energy matrix).
- Tử số $\mathbf{w}_t^H\mathbf{H}^H\mathbf{H}\mathbf{w}_t$: năng lượng tín hiệu đi qua kênh khi dùng beam $\mathbf{w}_t$.
- Mẫu số $\|\mathbf{w}_t\|^2$: chuẩn hóa công suất phát để so sánh công bằng giữa các nghiệm.
- Hàm mục tiêu có tính chu kỳ 2π theo từng biến pha ϕ_i.

3.2 Hàm mục tiêu khi chỉ biết AoD trội (Known dominating AoD)

Trong thực tế, việc ước lượng đầy đủ ma trận kênh $\mathbf{H}$ có thể khó. Một kịch bản thay thế là máy thu (hoặc khối tối ưu) chỉ biết hướng phát trội (dominating AoD) ký hiệu θ_t. Khi đó, thay vì tối ưu trực tiếp năng lượng tại máy thu thông qua $\mathbf{H}^H\mathbf{H}$, ta tối ưu năng lượng theo hướng AoD trội bằng đáp ứng mảng:

$$\tilde{f}(\rho, \boldsymbol{\phi}) = \frac{\left\|\mathbf{a}^T(\theta_t)\mathbf{w}_t(\rho, \boldsymbol{\phi})\right\|^2}{\|\mathbf{w}_t(\rho, \boldsymbol{\phi})\|^2}.$$

Ý nghĩa:

- $\mathbf{a}^T(\theta_t)\mathbf{w}_t$ đo "độ khớp" (beamforming gain) của beam phát theo hướng AoD trội.
- Khi chỉ biết AoD trội, tối ưu $\tilde{f}$ giúp tăng công suất phát theo hướng quan trọng nhất của kênh.
- Hàm $\tilde{f}$ cũng có tính chu kỳ 2π theo từng ϕ_i.

3.3 Bài toán tối ưu

Tùy thông tin kênh, bài toán có thể là:

(1) Known H:

$$(\rho^\star, \boldsymbol{\phi}^\star) = \arg\max_{\rho, \boldsymbol{\phi}} f(\rho, \boldsymbol{\phi}).$$

(2) Known dominating AoD θ_t:

$$(\rho^\star, \tilde{\boldsymbol{\phi}}^\star) = \arg\max_{\rho, \boldsymbol{\phi}} \tilde{f}(\rho, \boldsymbol{\phi}).$$

4 Ràng buộc tối ưu

$$0 \le \rho \le 1, \tag{2}$$
$$-\pi \le \phi_i \le \pi, \quad i = 1, \dots, M, \tag{3}$$
$$\|\mathbf{w}_t(\rho, \boldsymbol{\phi})\|^2 = 1. \tag{4}$$

Trong triển khai thực tế, có thể thêm ràng buộc lượng tử hóa pha:

$$\phi_i \in \left\{0, \Delta, 2\Delta, \dots, (2^b - 1)\Delta\right\}, \qquad \Delta = \frac{2\pi}{2^b}.$$

Trong báo cáo này, để minh họa ABC, ta xét pha liên tục trong $[-\pi, \pi]$ (sau đó mở rộng cho lượng tử hóa ở mục sau).

5 Ví dụ minh họa phép tính cho bài toán đa biến

Xét một dàn anten với $M = 3$ phần tử. Cho trước:

$$\mathbf{w}_{t,c} = \frac{1}{\sqrt{3}}[1, 1, 1]^T, \qquad \mathbf{w}_{t,s} = \frac{1}{\sqrt{3}}[1, e^{\jmath\pi/4}, e^{\jmath\pi/2}]^T,$$

$$\mathbf{H} = \operatorname{diag}\left(1,\ 0.8e^{\jmath\pi/4},\ 0.5e^{\jmath\pi/2}\right).$$

Véc-tơ biến tối ưu:

$$\mathbf{x} = (\rho, \phi_1, \phi_2, \phi_3),$$

tức bài toán có $M + 1 = 4$ biến tối ưu.

Bước 1: Tính $\mathbf{H}^H\mathbf{H}$

Vì $\mathbf{H}$ là ma trận chéo:

$$\mathbf{H}^H\mathbf{H} = \operatorname{diag}\left(|1|^2,\ |0.8e^{\jmath\pi/4}|^2,\ |0.5e^{\jmath\pi/2}|^2\right) = \operatorname{diag}(1,\ 0.64,\ 0.25).$$

Bước 2: Xây dựng $\mathbf{w}_t(\rho, \boldsymbol{\phi})$

$$\mathbf{w}_t(\rho, \boldsymbol{\phi}) = \sqrt{\rho}\,\mathbf{w}_{t,c} + \sqrt{1-\rho}\begin{bmatrix} e^{\jmath\phi_1} w_{t,s,1} \\ e^{\jmath\phi_2} w_{t,s,2} \\ e^{\jmath\phi_3} w_{t,s,3} \end{bmatrix}.$$

Thay $\mathbf{w}_{t,c}$ và $\mathbf{w}_{t,s}$:

$$\mathbf{w}_t(\rho, \boldsymbol{\phi}) = \frac{1}{\sqrt{3}}\begin{bmatrix} \sqrt{\rho} + \sqrt{1-\rho}\, e^{\jmath\phi_1} \\ \sqrt{\rho} + \sqrt{1-\rho}\, e^{\jmath(\phi_2 + \pi/4)} \\ \sqrt{\rho} + \sqrt{1-\rho}\, e^{\jmath(\phi_3 + \pi/2)} \end{bmatrix}.$$

Bước 3: Chọn một nghiệm cụ thể để minh họa

Xét:

$$\rho = 0.5, \qquad \boldsymbol{\phi} = [0, 0, 0]^T.$$

Với $\sqrt{\rho} = \sqrt{1-\rho} = \frac{1}{\sqrt{2}}$:

$$\mathbf{w}_t = \frac{1}{\sqrt{2}}\mathbf{w}_{t,c} + \frac{1}{\sqrt{2}}\mathbf{w}_{t,s} = \frac{1}{\sqrt{6}}\Big([1,1,1]^T + [1, e^{\jmath\pi/4}, e^{\jmath\pi/2}]^T\Big),$$

$$\mathbf{w}_t = \frac{1}{\sqrt{6}}\left[2,\ 1+e^{\jmath\pi/4},\ 1+e^{\jmath\pi/2}\right]^T.$$

Bước 4: Tính $\|\mathbf{w}_t\|^2$ và chuẩn hóa

$$\|\mathbf{w}_t\|^2 = \frac{1}{6}\left(|2|^2 + |1+e^{\jmath\pi/4}|^2 + |1+e^{\jmath\pi/2}|^2\right).$$

Ta có:

$$|2|^2 = 4, \qquad |1+e^{\jmath\pi/2}|^2 = |1+\jmath|^2 = 2.$$

Vì $e^{\jmath\pi/4} = \frac{\sqrt{2}}{2} + \jmath\frac{\sqrt{2}}{2}$:

$$1 + e^{\jmath\pi/4} = \left(1 + \frac{\sqrt{2}}{2}\right) + \jmath\frac{\sqrt{2}}{2},$$

$$|1+e^{\jmath\pi/4}|^2 = \left(1+\frac{\sqrt{2}}{2}\right)^2 + \left(\frac{\sqrt{2}}{2}\right)^2 \approx 3.4142.$$

Suy ra:

$$\|\mathbf{w}_t\|^2 \approx \frac{1}{6}(4 + 3.4142 + 2) = 1.5690.$$

Đặt $\hat{\mathbf{w}}_t = \mathbf{w}_t / \|\mathbf{w}_t\|$ để đảm bảo $\|\hat{\mathbf{w}}_t\|^2 = 1$.

Bước 5: Tính $f(\rho, \boldsymbol{\phi})$

Vì $\|\hat{\mathbf{w}}_t\|^2 = 1$:

$$f(\rho, \boldsymbol{\phi}) = \hat{\mathbf{w}}_t^H (\mathbf{H}^H \mathbf{H}) \hat{\mathbf{w}}_t.$$

Do $\mathbf{H}^H\mathbf{H}$ là ma trận chéo:

$$f = 1 \cdot |\hat{w}_{t,1}|^2 + 0.64 \cdot |\hat{w}_{t,2}|^2 + 0.25 \cdot |\hat{w}_{t,3}|^2.$$

Từ các giá trị đã tính (tương ứng với nghiệm đang xét), ta thu được xấp xỉ:

$$f \approx 0.7101.$$

Nhận xét

Giá trị trên tương ứng với nghiệm cụ thể $(\rho, \phi_1, \phi_2, \phi_3) = (0.5, 0, 0, 0)$. Trong bài toán tổng quát, việc thay đổi từng pha ϕ_i sẽ làm thay đổi mức độ cộng hưởng và triệt tiêu tín hiệu, từ đó làm thay đổi đáng kể giá trị hàm mục tiêu. Do đó, cần thuật toán tối ưu đa biến để tìm $(\rho^\star, \boldsymbol{\phi}^\star)$.

6 Lượng tử hóa véc-tơ beamforming trong mảng anten analog

6.1 Một bộ dịch pha (Single phase shifter)

Trong mảng analog, mỗi phần tử thường điều khiển được pha rời rạc với b bit. Mỗi phần tử $w_i = |w_i|e^{\mathrm{J}\psi_i}$ được ánh xạ pha về tập rời rạc $\mathcal{B}$:

$$\hat{\beta}_i = \arg\min_{\hat{\beta}\in\mathcal{B}} \left|\mathrm{mod}_{2\pi}(\psi_i - \hat{\beta})\right|, \quad \mathcal{B} = \{0, \Delta, 2\Delta, \ldots, (2^b-1)\Delta\},\ \Delta = \frac{2\pi}{2^b}.$$

Khi chỉ lượng tử hóa pha, các điểm lượng tử có cùng biên độ (constellation đồng biên độ), dẫn đến sai lệch biên độ so với $|w_i|$ mong muốn.

6.2 Hai bộ dịch pha (Two phase shifters) để khớp biên độ

Để giảm sai lệch biên độ, có thể dùng hai bộ dịch pha và xấp xỉ:

$$w_i = |w_i|e^{\mathrm{J}\psi_i} \approx e^{\mathrm{J}\beta_1^{(i)}} + e^{\mathrm{J}\beta_2^{(i)}}.$$

Có ba phương án:

6.2.1 (1) Lượng tử riêng từng bộ dịch pha

$$\hat{\beta}_1^{(i)} = \arg\min_{\hat{\beta}_1\in\mathcal{B}_1} |\mathrm{mod}_{2\pi}(\beta_1^{(i)} - \hat{\beta}_1)|, \quad \hat{\beta}_2^{(i)} = \arg\min_{\hat{\beta}_2\in\mathcal{B}_2} |\mathrm{mod}_{2\pi}(\beta_2^{(i)} - \hat{\beta}_2)|.$$

6.2.2 (2) Lượng tử kết hợp bằng codebook

Tạo codebook $\mathcal{C}$ từ các cặp pha lượng tử:

$$\hat{c}_k = e^{\mathrm{J}\hat{\beta}_1} + e^{\mathrm{J}\hat{\beta}_2}, \quad \hat{c}_k \in \mathcal{C}.$$

Chọn phần tử gần nhất:

$$\hat{w}_i = \arg\min_{\hat{c}_k\in\mathcal{C}} |w_i - \hat{c}_k|^2.$$

Trong thực tế, codebook thường được chuẩn hóa để đảm bảo công suất trung bình phù hợp (ví dụ $E[|\hat{c}_k|^2] = 1/M$).

6.2.3 (3) Lượng tử với hệ số co giãn tối ưu (IGSS-Q)

Xét hệ số scale $\nu > 0$ để tối ưu sai số lượng tử:

$$\nu_{\mathrm{opt}} = \arg\min_{\nu}\ \|\nu\mathbf{w}_t - \hat{\mathbf{q}}(\nu)\|_2^2,$$

với mỗi phần tử

$$\hat{q}_i(\nu) = \arg\min_{\hat{c}_k\in\mathcal{C}} |\nu w_i - \hat{c}_k|^2.$$

Cuối cùng chuẩn hóa:

$$\hat{\mathbf{w}}_t = \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|}.$$

IGSS-Q thường dùng biến thể golden-section search để tìm nhanh ν_{opt}.

7 Thuật toán metaheuristic áp dụng

7.1 Biểu diễn nghiệm, hàm thích nghi và xử lý biên

Một nghiệm ứng viên:

$$\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M) \in \Omega, \qquad \Omega = [0,1] \times [-\pi, \pi]^M.$$

Khi đánh giá mục tiêu, ta tạo $\mathbf{w}_t(\rho, \boldsymbol{\phi})$, sau đó chuẩn hóa để đảm bảo ràng buộc công suất:

$$\mathbf{w}_t \leftarrow \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}.$$

Fitness: do bài toán là cực đại hóa, có thể chọn:

$$fit(\mathbf{x}) = \begin{cases} \dfrac{1}{1 + \dfrac{1}{f(\mathbf{x})}}, & f(\mathbf{x}) > 0, \\ 0, & \text{ngược lại.} \end{cases}$$

Ghi chú: cũng có thể dùng trực tiếp $fit(\mathbf{x}) = f(\mathbf{x})$ nếu luôn dương và ổn định số.

Xử lý biên (Bound/Wrap):

- Kẹp $\rho \leftarrow \min(\max(\rho, 0), 1)$.
- Với mỗi ϕ_i: nếu $\phi_i > \pi$ thì $\phi_i \leftarrow \phi_i - 2\pi$; nếu $\phi_i < -\pi$ thì $\phi_i \leftarrow \phi_i + 2\pi$.

7.2 Mã giả chi tiết: Thuật toán ABC

Thuật toán ABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Thuật toán ABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Algorithm 5 Thuật toán ABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Require: Số ong SN, số vòng lặp MAX_ITER, ngưỡng $LIMIT$, miền tìm kiếm Ω, hàm mục tiêu $f(\mathbf{x})$
Ensure: Nghiệm tối ưu $\mathbf{x}^\star$ và giá trị $f(\mathbf{x}^\star)$
Khởi tạo SN nghiệm ban đầu $\mathbf{x}_i \sim U(\Omega)$
Tính $f_i = f(\mathbf{x}_i)$, $fit_i = Fitness(f_i)$, $trial_i \leftarrow 0$
$\mathbf{x}^\star \leftarrow \arg\max_i f_i$
for $t = 1$ to MAX_ITER **do**
  Pha ong thợ (Employed bees)
  for $i = 1$ to SN **do**
    Chọn ngẫu nhiên $k \neq i$
    $\mathbf{v} \leftarrow \mathbf{x}_i + \boldsymbol{\varphi} \odot (\mathbf{x}_i - \mathbf{x}_k)$
    $\boldsymbol{\varphi} \sim U[-1, 1]$
    $\mathbf{v} \leftarrow Bound(\mathbf{v}, \Omega)$
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $fit_i \leftarrow Fitness(f(\mathbf{v}))$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong quan sát (Onlooker bees)
  Tính xác suất chọn $p_i = \dfrac{fit_i}{\sum_{m=1}^{SN} fit_m}$
  for $n = 1$ to SN **do**
    Chọn nghiệm $\mathbf{x}_i$ theo roulette-wheel dựa trên p_i
    Chọn $k \neq i$ và sinh nghiệm $\mathbf{v}$ tương tự pha ong thợ
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $fit_i \leftarrow Fitness(f(\mathbf{v}))$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong do thám (Scout bees)
  for $i = 1$ to SN **do**
    if $trial_i \geq LIMIT$ **then**
      Sinh lại $\mathbf{x}_i \sim U(\Omega)$; $trial_i \leftarrow 0$
      $fit_i \leftarrow Fitness(f(\mathbf{x}_i))$
    end if
  end for
  Cập nhật nghiệm tốt nhất $\mathbf{x}^\star \leftarrow \arg\max_i f(\mathbf{x}_i)$
end for
Return $\mathbf{x}^\star, f(\mathbf{x}^\star)$

7.3 Mã giả chi tiết: GABC (Global-best guided ABC)

GABC cải tiến ABC bằng cách bổ sung thành phần dẫn hướng theo nghiệm tốt nhất toàn cục $\mathbf{x}^\star$. Công thức cập nhật nghiệm lân cận được viết như sau:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(x_j^\star - x_{ij}),$$

trong đó $\varphi_{ij} \sim U[-1, 1]$, $\psi_{ij} \sim U[0, C]$ và $C > 0$.

Thuật toán GABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Algorithm 6 Thuật toán ABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Require: Số ong SN, số vòng lặp MAX_ITER, ngưỡng $LIMIT$, miền tìm kiếm Ω, hàm mục tiêu $f(\mathbf{x})$
Ensure: Nghiệm tối ưu $\mathbf{x}^\star$ và giá trị $f(\mathbf{x}^\star)$
Khởi tạo SN nghiệm ban đầu $\mathbf{x}_i \sim U(\Omega)$
Tính $f_i = f(\mathbf{x}_i)$, $fit_i = Fitness(f_i)$, $trial_i \leftarrow 0$
$\mathbf{x}^\star \leftarrow \arg\max_i f_i$
for $t = 1$ to MAX_ITER **do**
  Pha ong thợ (Employed bees)
  for $i = 1$ to SN **do**
    Chọn ngẫu nhiên $k \neq i$
    $\mathbf{v} \leftarrow \mathbf{x}_i + \boldsymbol{\varphi} \odot (\mathbf{x}_i - \mathbf{x}_k)$
    $\boldsymbol{\varphi} \sim U[-1, 1]$
    $\mathbf{v} \leftarrow Bound(\mathbf{v}, \Omega)$
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $fit_i \leftarrow Fitness(f(\mathbf{v}))$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong quan sát (Onlooker bees)
  Tính xác suất chọn $p_i = \dfrac{fit_i}{\sum_{m=1}^{SN} fit_m}$
  for $n = 1$ to SN **do**
    Chọn nghiệm $\mathbf{x}_i$ theo roulette-wheel dựa trên p_i
    Chọn $k \neq i$ và sinh nghiệm $\mathbf{v}$ tương tự pha ong thợ
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $fit_i \leftarrow Fitness(f(\mathbf{v}))$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong do thám (Scout bees)
  for $i = 1$ to SN **do**
    if $trial_i \geq LIMIT$ **then**
      Sinh lại $\mathbf{x}_i \sim U(\Omega)$; $trial_i \leftarrow 0$
      $fit_i \leftarrow Fitness(f(\mathbf{x}_i))$
    end if
  end for
  Cập nhật nghiệm tốt nhất $\mathbf{x}^\star \leftarrow \arg\max_i f(\mathbf{x}_i)$
end for
Return $\mathbf{x}^\star, f(\mathbf{x}^\star)$

7.4 Mã giả chi tiết: Chaos-ABC (Chaotic ABC)

Chaos-ABC thay thế các biến ngẫu nhiên thuần tuý trong ABC bằng chuỗi hỗn loạn nhằm tăng tính đa dạng của quần thể và giảm nguy cơ kẹt tại cực trị cục bộ. Một lựa

chọn phổ biến là ánh xạ logistic:

$$u_{t+1} = \mu u_t(1-u_t), \qquad \mu = 4,\ u_t \in (0,1).$$

Từ u_t, các tham số được sinh như sau:

$$\varphi_{ij} = 2u_t - 1 \in [-1,1], \qquad \psi_{ij} = Cu_t \in [0,C].$$

Thuật toán CABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Algorithm 7 Thuật toán CABC cho tối ưu đa biến $\mathbf{x} = (\rho, \phi_1, \ldots, \phi_M)$

Require: Số ong SN, số vòng lặp MAX_ITER, ngưỡng $LIMIT$, miền Ω, tham số $\mu = 4$, giá trị khởi tạo $u_0 \in (0,1)$
Ensure: Nghiệm tối ưu $\mathbf{x}^\star$ và $f(\mathbf{x}^\star)$

Khởi tạo quần thể $\mathbf{x}_i \sim U(\Omega)$, tính f_i, fit_i, $trial_i \leftarrow 0$
Đặt $u \leftarrow u_0$
$\mathbf{x}^\star \leftarrow \arg\max_i f_i$
for $t = 1$ to MAX_ITER **do**
  Pha ong thợ (Employed bees)
  for $i = 1$ to SN **do**
    Chọn ngẫu nhiên $k \neq i$
    Cập nhật chuỗi hỗn loạn $u \leftarrow \mu u(1-u)$
    Sinh nghiệm lân cận $\mathbf{v} \leftarrow \mathbf{x}_i + (2u-1) \odot (\mathbf{x}_i - \mathbf{x}_k)$
    $\mathbf{v} \leftarrow Bound(\mathbf{v}, \Omega)$
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong quan sát (Onlooker bees)
  Tính xác suất chọn $p_i \propto fit_i$
  for $n = 1$ to SN **do**
    Chọn nghiệm $\mathbf{x}_i$ theo roulette-wheel
    Cập nhật $u \leftarrow \mu u(1-u)$ và sinh nghiệm $\mathbf{v}$ tương tự pha ong thợ
    if $Fitness(f(\mathbf{v})) > fit_i$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{v}$; $trial_i \leftarrow 0$
    else
      $trial_i \leftarrow trial_i + 1$
    end if
  end for
  Pha ong do thám (Scout bees)
  for $i = 1$ to SN **do**
    if $trial_i \geq LIMIT$ **then**
      Cập nhật $u \leftarrow \mu u(1-u)$
      Sinh lại $\mathbf{x}_i \sim U(\Omega)$ (có thể dùng u để lấy mẫu); $trial_i \leftarrow 0$
    end if
  end for
  Cập nhật nghiệm tốt nhất $\mathbf{x}^\star \leftarrow \arg\max_i f(\mathbf{x}_i)$
end for
Return $\mathbf{x}^\star$, $f(\mathbf{x}^\star)$

8 Áp dụng và so sánh các thuật toán vào bài toán thực tế

8.1 Mô tả bài toán tối ưu thiết kế beamforming multibeam

8.1.1 Bối cảnh

Xét một hệ thống đa anten với M phần tử anten phát. Ta muốn thiết kế một vector beamforming phát (transmit beam) $\mathbf{w}_t \in \mathbb{C}^M$ theo mô hình *multibeam*, tức là kết hợp giữa:

- beam phục vụ truyền thông $\mathbf{w}_c \in \mathbb{C}^M$,
- beam phục vụ sensing $\mathbf{w}_s \in \mathbb{C}^M$.

Kênh truyền được mô tả bởi ma trận $\mathbf{H} \in \mathbb{C}^{M\times M}$ (trong ví dụ minh hoạ có thể lấy $\mathbf{H}$ là ma trận chéo).

8.1.2 Biến tối ưu và mô hình multibeam

Ta tối ưu theo biến quyết định:

$$\mathbf{x} = [\rho,\ \phi_1,\ \phi_2,\ \ldots,\ \phi_M]^T,$$

trong đó:

$$\rho \in [0,1], \qquad \phi_i \in [-\pi,\pi],\ i = 1,\ldots,M.$$

Vector beamforming phát được xây dựng theo dạng:

$$\mathbf{w}_t(\mathbf{x}) = \sqrt{\rho}\,\mathbf{w}_c + \sqrt{1-\rho}\,\big(\exp(j\boldsymbol{\phi}) \odot \mathbf{w}_s\big), \tag{5}$$

với:

$$\boldsymbol{\phi} = [\phi_1,\ldots,\phi_M]^T, \quad \exp(j\boldsymbol{\phi}) = [e^{j\phi_1},\ldots,e^{j\phi_M}]^T,$$

và $\odot$ là phép nhân theo từng phần tử (Hadamard product).

8.1.3 Hàm mục tiêu

Mục tiêu là **tối đa hóa** năng lượng tín hiệu sau kênh khi dùng $\mathbf{w}_t(\mathbf{x})$, đồng thời chuẩn hóa theo công suất phát để so sánh công bằng. Hàm mục tiêu được chọn dạng Rayleigh quotient:

$$\max_{\mathbf{x}}\ f(\mathbf{x}) = \frac{\mathbf{w}_t(\mathbf{x})^H\big(\mathbf{H}^H\mathbf{H}\big)\mathbf{w}_t(\mathbf{x})}{\|\mathbf{w}_t(\mathbf{x})\|_2^2}, \tag{6}$$

trong đó $(\cdot)^H$ là chuyển vị liên hợp và $\|\cdot\|_2$ là chuẩn Euclid.

8.1.4 Ràng buộc

Bài toán tối ưu đầy đủ:

$$\max_{\rho,\boldsymbol{\phi}} \quad \frac{\mathbf{w}_t(\rho,\boldsymbol{\phi})^H\big(\mathbf{H}^H\mathbf{H}\big)\mathbf{w}_t(\rho,\boldsymbol{\phi})}{\|\mathbf{w}_t(\rho,\boldsymbol{\phi})\|_2^2} \tag{7}$$

$$\text{s.t.} \quad 0 \le \rho \le 1, \tag{8}$$

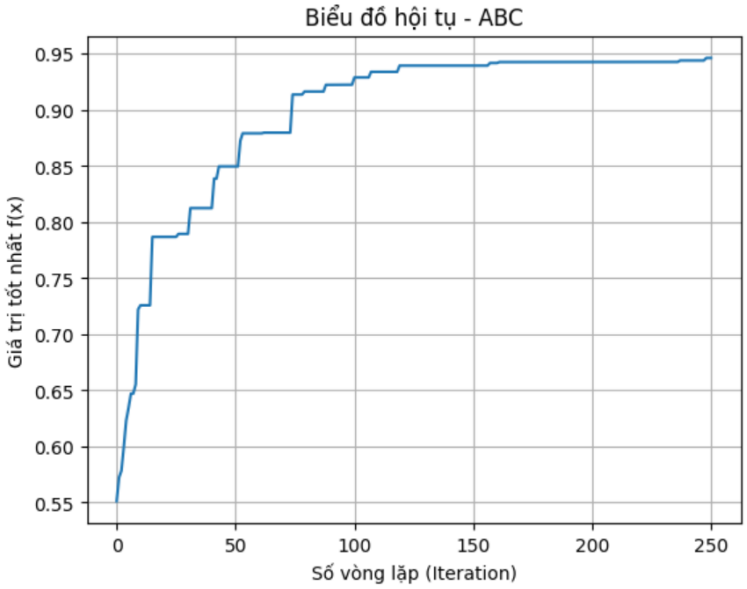
$$-\pi \le \phi_i \le \pi, \quad i = 1,\ldots,M. \tag{9}$$

8.2 Kết quả

8.2.1 ABC

```
============================================================
== ABC ==
Thời gian chạy (s): 1.764045
Best f(x):          0.946185846073

Vector nghiệm tối ưu x* = [rho, phi1..phiM]:
[ 0.499701 -0.832389 -3.071416  2.489666 -0.232888 -0.631184  0.488766
 -0.402177 -1.000035 -2.265303  2.224181 -2.06919   3.009919]

Rho*:
0.49970128629603905

Vector pha tối ưu phi*:
[-0.832389 -3.071416  2.489666 -0.232888 -0.631184  0.488766 -0.402177
 -1.000035 -2.265303  2.224181 -2.06919   3.009919]

Vector beam tối ưu wt* (complex):
[-6.037551e-05+0.005014j  3.177608e-04-0.013393j -1.013830e-04+0.002898j
  1.607745e-04+0.010741j  8.893158e-03-0.060002j  2.315579e-05-0.007696j
  4.374251e-05-0.008224j  3.348756e-03+0.037487j  3.779620e-01+0.107008j
  2.219732e-03+0.030835j  6.188471e-03-0.05037j   2.468033e-04+0.012266j]

Check lại f(x*):    0.946185846073
```

Hình 10: Kết quả thuật toán ABC

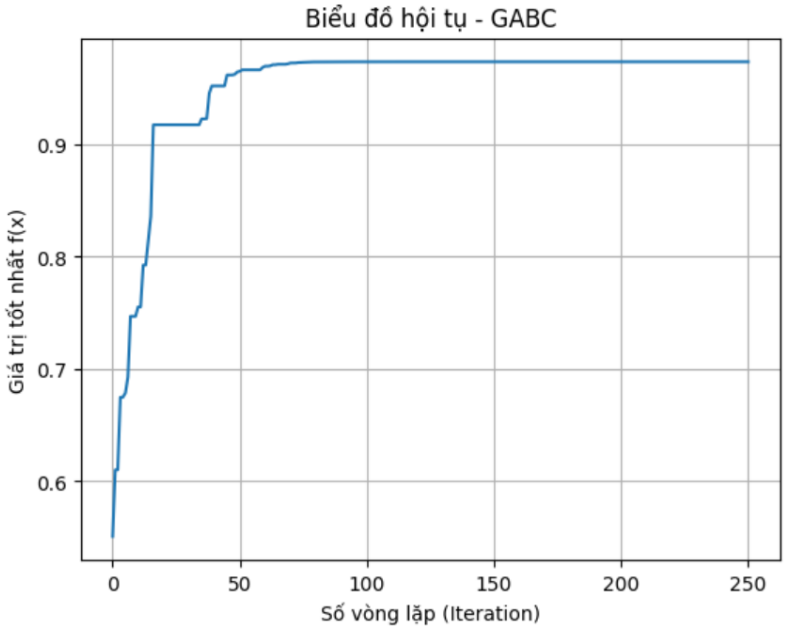


Hình 11: Biểu đồ hội tụ của thuật toán ABC

8.2.2 gABC

```
============================================================
== GABC ==
Thời gian chạy (s): 1.633370
Best f(x):          0.973457596707

Vector nghiệm tối ưu x* = [rho, phi1..phiM]:
[ 0.5      -0.80783  -3.137055  2.50386  -0.180258 -0.929447  0.451063
 -0.442466 -0.815394  2.971197  2.375775 -2.318452  3.070027]

Rho*:
0.4999999970648135

Vector pha tối ưu phi*:
[-0.80783  -3.137055  2.50386  -0.180258 -0.929447  0.451063 -0.442466
 -0.815394  2.971197  2.375775 -2.318452  3.070027]

Vector beam tối ưu wt* (complex):
[-1.198285e-09+1.936323e-09j -1.198285e-09-3.149611e-09j
 -1.198285e-09-8.194175e-10j -1.198285e-09+1.619859e-09j
 -1.198285e-09+4.404430e-10j -1.198285e-09+3.128173e-09j
 -1.198285e-09-2.409707e-10j -1.198285e-09-3.659299e-09j
  3.837414e-01-9.697584e-02j -1.198285e-09+2.792877e-09j
 -1.198285e-09-3.577505e-09j -1.198285e-09-1.082305e-09j]

Check lại f(x*):    0.973457596707
```

Hình 12: Kết quả thuật toán gABC

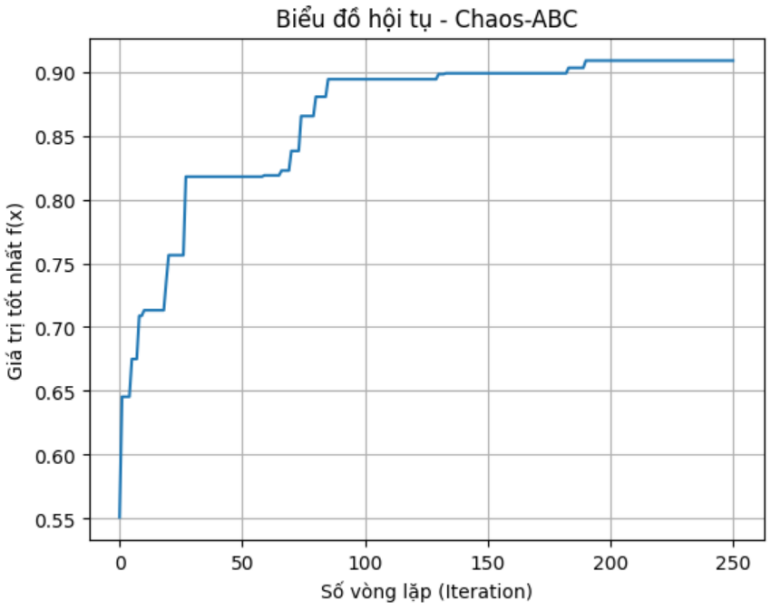


Hình 13: Biểu đồ hội tụ của thuật toán gABC

8.2.3 Chaos-ABC

```
============================================================
== Chaos-ABC ==
Thời gian chạy (s): 1.527389
Best f(x):          0.908813196024

Vector nghiệm tối ưu x* = [rho, phi1..phiM]:
[ 0.499592 -1.453592 -2.996168  2.535866 -0.103039 -1.248108  0.455474
 -0.907848 -1.077661 -2.547852  2.320828 -2.274546  2.920397]

Rho*:
0.49959168975103235

Vector pha tối ưu phi*:
[-1.453592 -2.996168  2.535866 -0.103039 -1.248108  0.455474 -0.907848
 -1.077661 -2.547852  2.320828 -2.274546  2.920397]

Vector beam tối ưu wt* (complex):
[ 4.095221e-02+0.122893j  1.856645e-03-0.028675j -6.210302e-05-0.006535j
  4.418351e-04-0.015753j  1.011399e-02+0.063977j -1.647061e-04-0.000901j
  2.155075e-02+0.091641j  6.816231e-03+0.052945j  4.009001e-01+0.054287j
  1.415041e-04+0.011215j  3.010781e-05-0.008963j  2.115073e-03+0.030442j]

Check lại f(x*):    0.908813196024
```

Hình 14: Kết quả thuật toán CABC



Hình 15: Biểu đồ hội tụ của thuật toán CABC

8.3 So sánh tổng hợp các thuật toán ABC

Tiêu chí	**ABC**	**GABC**	**Chaos-ABC**
Giá trị $f_{\max}$ đạt được	0.9462	**0.9735**	0.9088
Thời gian chạy (s)	1.7640	1.6334	**1.5274**
Số vòng lặp hội tụ	∼150	∼**50**	∼120
Tốc độ hội tụ	Chậm	**Nhanh**	Trung bình
Độ ổn định nghiệm	Trung bình	**Cao**	Trung bình
Khả năng thăm dò (exploration)	Trung bình	Trung bình	**Tốt**
Khả năng khai thác (exploitation)	Thấp	**Cao**	Trung bình
Giá trị ρ^* tối ưu	≈ 0.5	≈ 0.5	≈ 0.5
Đặc điểm nổi bật	Cấu trúc đơn giản	Dẫn hướng nghiệm toàn cục	Tránh cực trị cục bộ

Bảng 2: So sánh tổng hợp kết quả thực nghiệm giữa ABC, GABC và Chaos-ABC

Nhận xét: Từ bảng so sánh tổng hợp có thể rút ra một số nhận xét quan trọng như sau. Thứ nhất, thuật toán GABC cho kết quả tốt nhất về mặt giá trị hàm mục tiêu với $f_{\max} = 0.9735$, cao hơn đáng kể so với ABC gốc và Chaos-ABC. Đồng thời, GABC đạt tốc độ hội tụ nhanh nhất với số vòng lặp hội tụ chỉ khoảng 50 vòng, cho thấy hiệu quả rõ rệt của cơ chế dẫn hướng nghiệm về nghiệm tốt nhất toàn cục. Điều này chứng tỏ GABC có khả năng khai thác (exploitation) rất mạnh và phù hợp với bài toán thiết kế beamforming liên tục.

Thứ hai, thuật toán Chaos-ABC thể hiện khả năng thăm dò (exploration) tốt hơn so với ABC gốc, đặc biệt ở giai đoạn đầu của quá trình tối ưu, nhờ việc sử dụng chuỗi hỗn loạn logistic thay cho số ngẫu nhiên thuần. Tuy nhiên, do thiếu cơ chế dẫn hướng mạnh về nghiệm toàn cục, Chaos-ABC có tốc độ hội tụ chậm hơn GABC và giá trị tối ưu đạt được thấp hơn, với $f_{\max} = 0.9088$. Điều này cho thấy Chaos-ABC phù hợp hơn với các bài toán có nhiều cực trị cục bộ, nơi khả năng khám phá đóng vai trò quan trọng.

Cuối cùng, thuật toán ABC gốc có cấu trúc đơn giản và dễ cài đặt, nhưng thể hiện rõ hạn chế về tốc độ hội tụ và khả năng khai thác cục bộ. Giá trị hàm mục tiêu đạt được thấp hơn so với hai biến thể cải tiến, và số vòng lặp cần thiết để hội tụ là lớn nhất. Do đó, ABC gốc chủ yếu phù hợp để làm thuật toán nền (baseline) nhằm so sánh hiệu năng với các phương pháp cải tiến.

Nhìn chung, đối với bài toán thiết kế beamforming đa chùm trong hệ ISAC đang xét, GABC là lựa chọn phù hợp nhất khi ưu tiên hiệu năng và tốc độ hội tụ, trong khi Chaos-ABC đóng vai trò bổ trợ tốt trong các kịch bản có không gian nghiệm phức tạp và nhiều cực trị cục bộ.

8.4 Nhận xét định tính

- ABC dễ cài đặt, thường hội tụ ổn định với bài toán vừa và nhỏ.
- GABC có xu hướng hội tụ nhanh hơn nhờ dẫn hướng theo nghiệm tốt nhất toàn cục, nhưng cần chọn C phù hợp để tránh mất đa dạng.
- Chaos-ABC tăng đa dạng nhờ logistic map nên thường giúp thoát kẹt cực trị cục bộ tốt hơn trong không gian tìm kiếm lớn.

9 Mã nguồn

Toàn bộ mã nguồn dùng để thực hiện các thí nghiệm và vẽ đồ thị trong báo cáo được lưu trữ tại Google Colab :

Colab Notebook - multibeam (beamforming) và sử dụng ABC, GABC, CABC

Phần III Kết luận

Phần I: Thuật toán bầy ong nhân tạo và các biến thể

Báo cáo đã hệ thống hóa thuật toán Bầy ong Nhân tạo, viết tắt là ABC, như một phương pháp tối ưu metaheuristic dựa trên cơ chế tìm kiếm thức ăn của đàn ong. Các thành phần quan trọng gồm cách khởi tạo quần thể nghiệm, cơ chế sinh nghiệm lân cận trong pha ong thợ và pha ong quan sát, quy tắc chọn lọc tham lam để giữ nghiệm tốt hơn, công thức đổi hàm mục tiêu sang độ thích nghi, cùng với xử lý biên và điều kiện dừng. Trên cơ sở đó, báo cáo mô tả rõ vai trò cân bằng giữa thăm dò và khai thác, trong đó cơ chế ong do thám giúp làm mới nghiệm khi một nguồn thức ăn không còn được cải thiện và hỗ trợ thoát khỏi cực trị cục bộ.

Bên cạnh ABC gốc, báo cáo trình bày ba biến thể tiêu biểu và nêu nhận xét định tính từ các kết quả minh họa. Thuật toán GABC bổ sung thành phần dẫn hướng theo nghiệm tốt nhất toàn cục nên tăng khả năng khai thác và thường hội tụ nhanh hơn trong các bài toán liên tục. Thuật toán qABC tập trung tìm kiếm quanh nghiệm tốt và rút gọn quá trình cập nhật nên có thể giảm chi phí tính toán, tuy nhiên dễ hội tụ sớm nếu không gian nghiệm nhiều cực trị. Thuật toán Chaos-ABC sử dụng chuỗi hỗn loạn logistic để thay cho biến ngẫu nhiên trong khởi tạo và cập nhật, nhờ đó tăng đa dạng quần thể và cải thiện khả năng thăm dò, phù hợp hơn khi bài toán có nhiều cực trị cục bộ. Từ các phân tích trên, có thể rút ra rằng ABC gốc phù hợp làm chuẩn so sánh và triển khai đơn giản, trong khi lựa chọn biến thể nên dựa trên mức độ phức tạp của hàm mục tiêu và yêu cầu về tốc độ hội tụ.

Phần II: Bài toán multibeam và áp dụng ABC để tối ưu

Báo cáo đã trình bày bài toán tối ưu beamforming đa tia trong hệ thống JCAS dưới dạng bài toán tối ưu phi tuyến đa biến, giải thích rõ ý nghĩa các biến và các thành phần trong công thức hàm mục tiêu. Bằng việc chọn beam thu theo nguyên lý MRC, trường hợp biết đầy đủ ma trận kênh $\mathbf{H}$ được đưa về cực đại hóa dạng Rayleigh quotient theo $\mathbf{w}_t(\rho, \boldsymbol{\phi})$. Trong trường hợp chỉ biết AoD trội, báo cáo bổ sung mục tiêu thay thế dựa trên độ lợi mảng theo $\mathbf{a}(\theta_t)$ để giảm yêu cầu CSI. Báo cáo cũng nêu cơ chế sinh beam con bằng thuật toán Two-step ILS và thảo luận các ràng buộc phần cứng analog thông qua lượng tử hóa pha với một bộ dịch pha, hai bộ dịch pha theo lượng tử riêng hoặc theo codebook và IGSS-Q tối ưu hệ số co giãn. Các thuật toán ABC và các biến thể gồm GABC và Chaos-ABC là lựa chọn phù hợp cho bài toán không lồi, nhiều cực trị cục bộ và số biến lớn; trong đó GABC thường hội tụ nhanh hơn, còn Chaos-ABC tăng khả năng thoát cực trị cục bộ. Cuối cùng, phần lượng tử hóa với hai bộ dịch pha và IGSS-Q được kỳ vọng giảm suy hao tốt hơn so với lượng tử hóa pha đơn giản.

Phần IV Tài liệu tham khảo

- D. Karaboga, *Artificial Bee Colony Algorithm*. Scholarpedia.
  Link: http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm
- E. Cuevas et al., *GABC: A Modified Artificial Bee Colony Algorithm Based on Genetic Operators*. Applied Sciences, MDPI, 2020.
  Link: https://www.mdpi.com/2076-3417/10/10/3352
- M. Arnaout et al., *CABC: Chaotic Artificial Bee Colony Algorithm*. Scientific Reports, Nature, 2023.
  Link: https://www.nature.com/articles/s41598-023-44770-8
- B. Cico, *Quantum-inspired Artificial Bee Colony Algorithm*. Epoka University.
  Link: https://www.dspace.epoka.edu.al/bitstream/handle/1/849/paper_43.pdf?sequence=1
- Slide/Bài báo được giao trong học phần về tối ưu và lượng tử hóa multibeam beamforming vector cho JCAS.
- Z. Shi and Z. Feng, "A new array pattern synthesis algorithm using the two-step least-squares method," *IEEE Signal Processing Letters*, 2005.
- J. A. Zhang et al., "Multibeam for Joint Communication and Sensing Using Steerable Analog Antenna Arrays," 2018.
- D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005.