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

BÁO CÁO TÌM HIỂU CÁC BIẾN THỂ CỦA THUẬT TOÁN ABC

STT	Họ và tên	Mã số sinh viên
1	Nguyễn Khắc Trí	20225769
2	Nguyễn Thiện Nam	20235790

Người hướng dẫn : Nguyễn Quang Đông

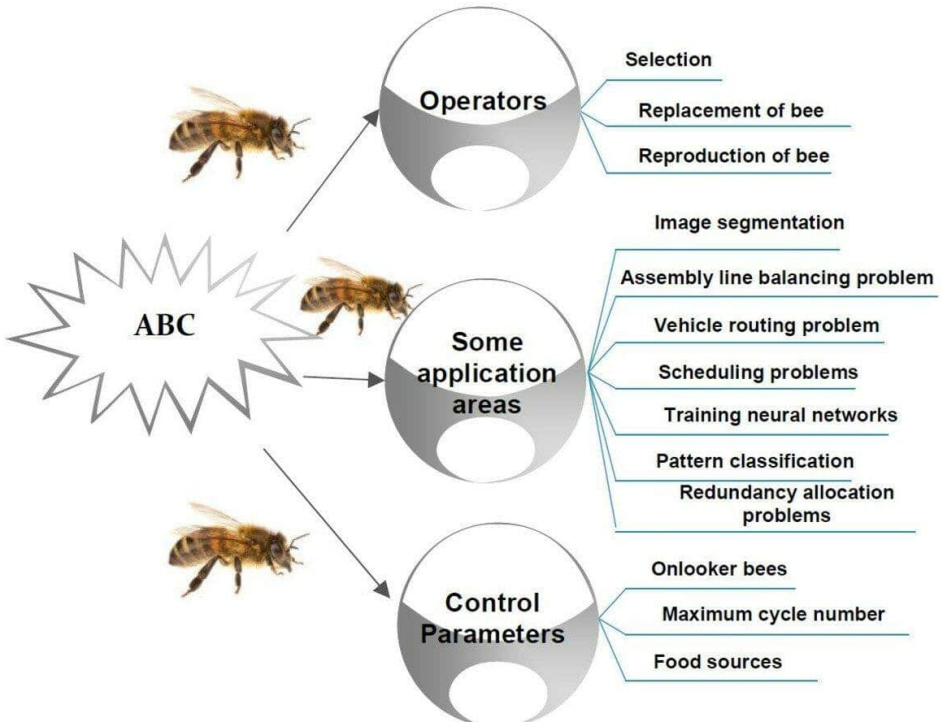
Mục lục

1 Giới thiệu tổng quan

Trong lĩnh vực trí tuệ nhân tạo và tối ưu hóa, các thuật toán dựa trên trí tuệ bầy đàn (Swarm Intelligence) đã chứng minh được hiệu quả vượt trội. Báo cáo này tập trung tìm hiểu sâu về Thuật toán Bầy ong Nhân tạo (Artificial Bee Colony - ABC) - một trong những thuật toán kinh điển nhất, đồng thời phân tích một biến thể hiện đại mang tên Adaptive Exploration ABC (AEABC).

Thuật toán ABC được phát triển bởi Dervis Karaboga và cộng sự từ đại học Erciyes (Thổ Nhĩ Kỳ) vào năm 2005. Thuật toán mô phỏng hành vi tìm kiếm thức ăn thông minh của đàn ong mật, bao gồm ba thành phần cốt lõi:

- **Ong Thợ (Employed bees):** Chịu trách nhiệm khai thác thông tin cục bộ quanh nguồn thức ăn hiện có.
- **Ong Quan sát (Onlooker bees):** Lựa chọn nguồn thức ăn dựa trên thông tin từ ong thợ (cơ chế chọn lọc tự nhiên).
- **Ong Trinh sát (Scout bees):** Tìm kiếm ngẫu nhiên các nguồn mới khi nguồn cũ cạn kiệt (cơ chế thoát khỏi cực trị địa phương).



Hình 1: Mô phỏng hành vi của bầy ong trong thuật toán ABC

Ưu điểm nổi bật của ABC là sự đơn giản, ít tham số và khả năng cân bằng giữa khám phá (exploration) và khai thác (exploitation). Tuy nhiên, đối với các bài toán quy mô lớn, ABC vẫn tồn tại nhược điểm về tốc độ hội tụ chậm. Đây chính là động lực để nghiên cứu biến thể AEABC được trình bày trong các phần sau.

2 Cơ sở lý thuyết và thuật toán ABC gốc

2.1 Khởi tạo (Initialization)

Quần thể ban đầu gồm SN giải pháp (nguồn thức ăn) được tạo ngẫu nhiên trong không gian tìm kiếm D-chiều. Giả sử x_i là giải pháp thứ i, thì thành phần thứ j của nó ($x_{i,j}$) được khởi tạo như sau:

$$x_{i,j} = x_{min,j} + \text{rand}(0,1) \cdot (x_{max,j} - x_{min,j}) \tag{1}$$

Trong đó:

- $i \in \{1, \ldots, SN\}$ và $j \in \{1, \ldots, D\}$.
- $x_{min,j}$ và $x_{max,j}$ là giới hạn dưới và giới hạn trên của tham số thứ j.

2.2 Giai đoạn Ong thợ (Employed Bee Phase)

Mỗi ong thợ tìm kiếm một giải pháp mới v_i trong vùng lân cận của vị trí hiện tại x_i theo công thức:

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j}) \tag{2}$$

Trong đó:

- $k \in \{1, \ldots, SN\}$ là chỉ số được chọn ngẫu nhiên khác với i ($k \neq i$).
- $\phi_{i,j}$ là một số ngẫu nhiên trong khoảng $[-1, 1]$, quyết định độ lớn bước nhảy.

Sau khi tạo ra v_i, thuật toán sẽ đánh giá độ thích nghi của nó. Nếu v_i tốt hơn x_i, ong thợ sẽ ghi nhớ vị trí mới và quên vị trí cũ (cơ chế lựa chọn tham lam - greedy selection).

2.3 Giai đoạn Ong quan sát (Onlooker Bee Phase)

Ong quan sát đánh giá thông tin từ ong thợ và chọn nguồn thức ăn dựa trên xác suất p_i. Xác suất chọn nguồn thức ăn thứ i thường được tính bằng phương pháp **bánh xe quay (Roulette Wheel Selection)**:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3}$$

Trong đó fit_i là giá trị độ thích nghi của giải pháp i. Sau khi chọn được nguồn thức ăn, ong quan sát cũng thực hiện tìm kiếm lân cận tương tự như ong thợ (sử dụng công thức tính $v_{i,j}$ ở trên).

2.4 Giai đoạn Ong trinh sát (Scout Bee Phase)

Nếu một nguồn thức ăn không được cải thiện sau một số lần thử nhất định (gọi là tham số 'limit'), nguồn đó được coi là đã cạn kiệt. Ong thợ tại vị trí đó trở thành ong trinh sát và tìm một nguồn thức ăn mới ngẫu nhiên:

$$x_{i,j} = x_{min,j} + \text{rand}(0,1) \cdot (x_{max,j} - x_{min,j}) \tag{4}$$

3 Triển khai thuật toán ABC (Implementation)

Dưới đây là mã nguồn triển khai thuật toán Bầy ong Nhân tạo (ABC) cơ bản sử dụng ngôn ngữ Python và thư viện NumPy để giải quyết bài toán tối ưu hóa hàm cầu (Sphere function).

```
import random
import numpy as np

# --- Thiết lạp các tham só ---
num_employed_bees = 50      # Só luọng ong thọ (Employed Bees)
num_onlooker_bees = 50      # Só luọng ong quan sát (Onlooker Bees)
max_iterations = 100        # Só vòng lạp tói đa
limit = 50                  # Giói hạn só làn không cái thiện truóc khi bó
    nguồn
problem_size = 5            # Kích thuóc bài toán (só chiều)

# --- Định nghĩa hàm thích nghi (Fitness Function) ---
def fitness_function(x):
    return np.sum(x**2)     # Bài toán tói uu hóa hàm càu (Sphere function)

# --- Khói tạo quàn thé ban đàu ---
food_sources = np.random.rand(num_employed_bees, problem_size) * 100
fitness_values = np.array([fitness_function(food) for food in food_sources])
no_improvement_counters = np.zeros(num_employed_bees) # Bọ đém só làn không
    cái thiện

# --- Vòng lạp chính ---
for iteration in range(max_iterations):
    # --- Giai đoạn Ong thọ (Employed bees) ---
    for i in range(num_employed_bees):
        # Chọn ngẫu nhiên mọt chiều (biến) đé thay đói
        dimension = random.randint(0, problem_size - 1)

        # Tạo ra mọt giái pháp úng viên mói (đọt biến/biến thé)
        mutant = np.copy(food_sources[i])
        # Súa đói chiều đuọc chọn mọt cách ngẫu nhiên
        mutant[dimension] += (random.random() - 0.5) * 2

        # Đánh giá đọ thích nghi cúa giái pháp mói
        mutant_fitness = fitness_function(mutant)

        # Lụa chọn tham lam (Greedy selection)
        if mutant_fitness < fitness_values[i]:
            food_sources[i] = mutant
            fitness_values[i] = mutant_fitness
            no_improvement_counters[i] = 0
        else:
            no_improvement_counters[i] += 1

    # --- Tính toán xác suát dụa trên giá trị thích nghi ---
    total_fitness = np.sum(fitness_values)
    if total_fitness == 0:
        probabilities = np.ones(num_employed_bees) / num_employed_bees
    else:
        # Công thúc tính xác suát (nghịch đáo vì tìm cục tiéu)
        probabilities = (1.0 / (1.0 + fitness_values)) / np.sum(1.0 / (1.0 +
    fitness_values))

    # --- Giai đoạn Ong quan sát (Onlooker bees) ---
```

```
    for j in range(num_onlooker_bees):
        # Chọn nguồn thức ăn dựa trên vòng quay roulette
        selected_food_source = np.random.choice(num_employed_bees, p=
    probabilities)

        dimension = random.randint(0, problem_size - 1)
        mutant = np.copy(food_sources[selected_food_source])
        mutant[dimension] += (random.random() - 0.5) * 2

        mutant_fitness = fitness_function(mutant)

        if mutant_fitness < fitness_values[selected_food_source]:
            food_sources[selected_food_source] = mutant
            fitness_values[selected_food_source] = mutant_fitness
            no_improvement_counters[selected_food_source] = 0
        else:
            no_improvement_counters[selected_food_source] += 1

    # --- Giai đoạn Ong trinh sát (Scout bees) ---
    for k in range(num_employed_bees):
        if no_improvement_counters[k] > limit:
            # Thay thé bàng mọt nguòn thúc ăn ngãu nhiên mói
            food_sources[k] = np.random.rand(problem_size) * 100
            fitness_values[k] = fitness_function(food_sources[k])
            no_improvement_counters[k] = 0

    # Hiến thị giải pháp tót nhát trong mõi vòng lạp
    best_fitness = np.min(fitness_values)
    print(f"Vòng lạp {iteration}, Dọ thích nghi tót nhát: {best_fitness}")

# --- Tìm giái pháp tót nhát tóng thé ---
overall_best_fitness = np.min(fitness_values)
overall_best_index = np.argmin(fitness_values)
overall_best_solution = food_sources[overall_best_index]

print("--- Két quá tói uu hóa ---")
print(f"Dọ thích nghi tót nhát: {overall_best_fitness}")
print(f"Giái pháp tót nhát: {overall_best_solution}")
```

Listing 1: Mã nguồn Python mô phỏng thuật toán ABC

4 Các Biến thể của Thuật toán ABC (ABC Variants)

Mặc dù thuật toán ABC cơ bản có khả năng khám phá (exploration) tốt, nhưng nó đôi khi gặp hạn chế trong việc khai thác (exploitation) và tốc độ hội tụ chậm. Do đó, nhiều biến thể đã được phát triển để khắc phục những nhược điểm này.

4.1 Biến thể dựa trên Chiến lược Tìm kiếm (Search Strategy Variants)

Các biến thể này sửa đổi phương trình sinh giải pháp mới ($v_{i,j}$) để định hướng tìm kiếm tốt hơn.

Ví dụ: G-best Guided ABC (GABC)
Biến thể này sử dụng thông tin từ giải pháp tốt nhất toàn cục (x_{best}) để dẫn dắt các ong thợ và ong quan sát, giúp tăng tốc độ hội tụ. Phương trình tìm kiếm được sửa đổi như sau:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) + \psi_{i,j}(x_{best,j} - x_{i,j}) \quad (5)$$

Trong đó:

- $x_{best,j}$: Thành phần thứ j của giải pháp tốt nhất hiện tại.
- $\psi_{i,j}$: Một tham số ngẫu nhiên mới (thường trong khoảng $[0, 1.5]$), điều khiển mức độ ảnh hưởng của giải pháp tốt nhất.

4.2 Biến thể Thích nghi (Adaptive ABC)

Các tham số điều khiển (như limit hoặc kích thước đàn) hoặc chiến lược tìm kiếm được tự động điều chỉnh trong quá trình chạy thuật toán thay vì cố định. Ví dụ: *Adaptive Exploration ABC (AEABC)* giúp cân bằng động giữa quá trình tìm kiếm cục bộ và toàn cục.

4.3 Biến thể Đa mục tiêu (Multi-objective ABC - MOABC)

Được thiết kế để giải quyết các bài toán có nhiều mục tiêu mâu thuẫn nhau. MOABC sử dụng khái niệm *tối ưu Pareto* và thường tích hợp thêm một kho lưu trữ bên ngoài (external archive) để giữ các giải pháp không bị trội (non-dominated solutions).

4.4 Biến thể Lai ghép (Hybrid ABC)

Kết hợp ABC với các thuật toán khác như *Differential Evolution (DE)*, *Particle Swarm Optimization (PSO)* hoặc *Genetic Algorithm (GA)* để tận dụng ưu điểm của cả hai.

5 Ứng dụng của thuật toán ABC trong thực tiễn

Nhờ tính đơn giản trong cài đặt và khả năng vượt trội trong việc cân bằng giữa khám phá (exploration) và khai thác (exploitation), thuật toán ABC đã vượt ra khỏi các hàm toán học lý thuyết để giải quyết các bài toán phức tạp trong nhiều lĩnh vực công nghiệp và khoa học máy tính.

5.1 Kỹ thuật và Thiết kế Cơ khí (Engineering Design)

Trong lĩnh vực cơ khí, các bài toán thiết kế thường đi kèm với nhiều ràng buộc phi tuyến tính (về độ bền, độ võng, tần số rung). ABC được sử dụng để tìm ra bộ thông số thiết kế tối ưu nhằm giảm thiểu chi phí vật liệu hoặc trọng lượng nhưng vẫn đảm bảo các yếu tố an toàn kỹ thuật.

Nghiên cứu điển hình: Thiết kế Lò xo nén (Compression Spring Design)

- **Mục tiêu:** Tối thiểu hóa trọng lượng của lò xo.
- **Biến quyết định:** Đường kính dây (d), đường kính trung bình vòng cuộn (D), và số vòng dây hoạt động (N).
- **Ràng buộc:** Giới hạn về ứng suất cắt, tần số rung và độ võng dưới tải trọng.
- **Hiệu quả:** Nghiên cứu của *Karaboga và Akay (2009)* đã chứng minh ABC tìm được thiết kế nhẹ hơn so với các phương pháp truyền thống và các thuật toán di truyền (GA) trên cùng một tập ràng buộc.

Các bài toán khác bao gồm:

- **Thiết kế dầm hàn (Welded Beam):** Tối ưu hóa kích thước mối hàn để chịu tải trọng P với chi phí thấp nhất (như đã phân tích ở phần trước).
- **Tối ưu hóa kết cấu giàn (Truss Optimization):** Xác định tiết diện thanh giàn để giảm trọng lượng công trình xây dựng.

5.2 Xử lý Ảnh và Thị giác máy tính (Image Processing)

Xử lý ảnh đòi hỏi tốc độ cao và độ chính xác trong môi trường dữ liệu nhiễu. ABC được ứng dụng để tối ưu hóa các tham số mà các phương pháp cổ điển thường bị mắc kẹt ở cực trị địa phương.

5.2.1 Phân ngưỡng ảnh đa cấp (Multilevel Image Thresholding)

Để tách đối tượng khỏi nền (ví dụ: tách khối u khỏi ảnh chụp MRI), phương pháp truyền thống như Otsu hoạt động tốt với 1 ngưỡng nhưng rất chậm khi cần tìm nhiều ngưỡng (multilevel).

- **Cách thức:** ABC coi các giá trị ngưỡng $(t_1, t_2, \ldots, t_k)$ là các nguồn thức ăn. Hàm mục tiêu là tối đa hóa phương sai giữa các lớp (between-class variance).
- **Ưu điểm:** ABC giảm đáng kể thời gian tính toán so với việc duyệt toàn bộ (exhaustive search) mà vẫn tìm được ngưỡng tối ưu để phân đoạn ảnh chính xác.

5.2.2 Phân cụm dữ liệu (Data Clustering)

Trong bài toán phân cụm, thuật toán K-means truyền thống rất nhạy cảm với việc khởi tạo tâm cụm ban đầu.

- **Ứng dụng:** ABC được sử dụng để tìm vị trí các tâm cụm (centroids) sao cho khoảng cách từ các điểm dữ liệu đến tâm là nhỏ nhất.
- **Kết quả:** Các biến thể như *ABC-Fuzzy C-means* cho độ chính xác phân loại cao hơn trên các tập dữ liệu chuẩn như Iris hay Wine data.

5.3 Huấn luyện Mạng Nơ-ron Nhân tạo (Artificial Neural Networks)

Mạng nơ-ron truyền thẳng (Feed-forward Neural Networks - FNN) thường được huấn luyện bằng thuật toán Lan truyền ngược (Backpropagation - BP). Tuy nhiên, BP dựa trên đạo hàm (Gradient Descent) nên dễ bị kẹt ở các điểm tối ưu cục bộ và hội tụ chậm.

- **Cơ chế của ABC:** Coi tập hợp các trọng số (weights) và độ lệch (biases) của mạng nơ-ron là một nguồn thức ăn (một vector giải pháp).
- **Hàm mục tiêu:** Tối thiểu hóa sai số trung bình bình phương (Mean Squared Error - MSE) trên tập huấn luyện.
- **Ưu thế:** ABC thực hiện tìm kiếm toàn cục, giúp mạng nơ-ron tránh bị kẹt ở các điểm lỗi cục bộ, đặc biệt hiệu quả cho các bài toán phân loại logic XOR hoặc dự báo chuỗi thời gian đơn giản.

5.4 Các bài toán Lập lịch (Scheduling Problems)

Đây là lớp bài toán NP-Hard điển hình, nơi không gian tìm kiếm tăng theo cấp số nhân với kích thước dữ liệu vào. ABC phiên bản rời rạc (Discrete ABC) được phát triển để giải quyết các vấn đề này.

Nghiên cứu điển hình: Lập lịch phân xưởng (Job-shop Scheduling)

- **Bài toán:** Sắp xếp n công việc trên m máy móc sao cho tổng thời gian hoàn thành (Makespan) là ngắn nhất.
- **Thách thức:** Phải tuân thủ thứ tự công đoạn và mỗi máy chỉ làm một việc tại một thời điểm.
- **Ứng dụng ABC:** Mỗi con ong đại diện cho một hoán vị của các công việc. ABC giúp tìm ra trình tự tối ưu nhanh hơn các phương pháp tìm kiếm heuristic cổ điển, giúp các nhà máy tối ưu hóa năng suất dây chuyền.

6 Biến thể cải tiến: Thuật toán Adaptive Exploration ABC - AEABC

Thuật toán AEABC đại diện cho một bước tiến đáng kể so với phiên bản gốc. Sự mới lạ của nó không nằm ở việc lai ghép với các siêu giải thuật khác, mà là ở việc tái thiết kế cơ chế nền tảng của ABC để sửa chữa lỗ hổng khái niệm và mô phỏng chính xác hơn hành vi thực tế của loài ong. AEABC hiệu chỉnh thiết kế ban đầu bằng cách tích hợp một tham số dựa trên khoảng cách, từ đó nâng cao đáng kể tính mạnh mẽ của thuật toán.

Sự đổi mới cốt lõi của AEABC là việc tích hợp một tham số dựa trên khoảng cách vào quá trình ra quyết định của ong. Điều này được thực hiện thông qua việc tính toán khoảng cách Euclidean giữa giải pháp hiện tại và một giải pháp được chọn ngẫu nhiên, sau đó chuyển đổi khoảng cách này thành một giá trị xác suất.

- **Công thức khoảng cách:** $d = \|x_{ij} - x_{kj}\|$
- **Công thức xác suất khoảng cách:** $P_d = e^{-(1/d)}$

Giá trị xác suất khoảng cách (P_d) này hoạt động như một ngưỡng động. Nếu một số ngẫu nhiên r (trong khoảng $[0, 1]$) lớn hơn P_d, giải pháp ứng cử viên sẽ được dịch chuyển, điều chỉnh quỹ đạo tìm kiếm. Bảng dưới đây giải thích các hệ quả của quy tắc này trong các kịch bản khoảng cách khác nhau:

Khoảng cách (d)	**Xác suất (P_d)**	**Mô tả Tác động**
Khoảng cách xa	P_d tiến gần đến 1	Xác suất dịch chuyển giải pháp là **thấp** (vì $r > P_d$ ít có khả năng xảy ra). Điều này ưu tiên **thăm dò (exploration)**, cho phép thuật toán duy trì quỹ đạo tìm kiếm rộng.
Khoảng cách gần	P_d tiến gần đến 0	Xác suất dịch chuyển giải pháp là **cao** (vì $r > P_d$ rất có khả năng xảy ra). Điều này ưu tiên **khai thác (exploitation)**, tăng cường tìm kiếm cục bộ trong một khu vực hứa hẹn.

Tham số xác suất dựa trên khoảng cách này chính là chìa khóa cho khả năng thích ứng của AEABC. Khi thăm dò các giải pháp ở xa nhau (d lớn, P_d cao), thuật toán có xác suất dịch chuyển thấp, cho phép nó duy trì một quỹ đạo tìm kiếm rộng và mang tính khám phá. Ngược lại, khi nó xác định được một vùng hứa hẹn với các giải pháp tập trung gần nhau (d nhỏ, P_d thấp), xác suất dịch chuyển cao sẽ tăng cường tìm kiếm cục bộ, thúc đẩy quá trình khai thác sâu trong khu vực đó. Cơ chế này cho phép AEABC tự động cân bằng giữa việc *khai thác* các giải pháp lân cận đầy hứa hẹn và việc *thăm dò* các vùng xa hơn trong không gian tìm kiếm, một yếu tố then chốt giúp nó đạt được hiệu suất vượt trội.

6.1 Bài toán tối ưu và phương pháp thử nghiệm

Để xác thực hiệu quả, tính mạnh mẽ và khả năng khái quát hóa của một thuật toán mới, việc kiểm tra nghiêm ngặt và toàn diện trên một loạt các bài toán đa dạng là cực kỳ quan trọng. Phương pháp này đảm bảo rằng hiệu suất vượt trội của thuật toán không chỉ giới hạn ở một loại vấn đề cụ thể mà có thể áp dụng cho nhiều thách thức khác nhau.

Lĩnh vực bài toán bao trùm mà AEABC được thiết kế để giải quyết là "tối ưu hóa toán học"(mathematical optimization). Để đánh giá hiệu suất của AEABC, một bộ gồm 25 hàm benchmark đã được sử dụng. Các hàm này được phân loại thành các nhóm khác nhau để kiểm tra các khía cạnh cụ thể về khả năng tìm kiếm của thuật toán:

- **Hàm benchmark đơn phương thức (Unimodal):** Các hàm này chỉ có một điểm tối ưu toàn cục duy nhất. Chúng được sử dụng để đánh giá khả năng khai thác và tốc độ hội tụ của thuật toán.
- **Hàm benchmark đa phương thức (Multimodal):** Các hàm này có nhiều điểm tối ưu cục bộ, tạo ra một thách thức lớn hơn. Chúng được dùng để kiểm tra khả năng của thuật toán trong việc tránh hội tụ sớm tại các điểm tối ưu cục bộ và tìm ra điểm tối ưu toàn cục.
- **Bài toán quy mô lớn (Large-scale):** Các bài toán có số lượng biến cao (ví dụ: $D = 1000$). Các bài toán này kiểm tra khả năng mở rộng và hiệu quả của thuật toán khi không gian tìm kiếm tăng lên theo cấp số nhân.
- **Bài toán quy mô nhỏ (Small-scale):** Các bài toán có số lượng biến thấp (ví dụ: $D = 2$). Chúng được sử dụng để phân tích chi tiết hành vi và độ chính xác của thuật toán trong các không gian tìm kiếm có thể quản lý được.

6.2 Triển khai thuật toán AEABC (Implementation)

Mã nguồn dưới đây thể hiện sự cải tiến từ thuật toán ABC gốc sang biến thể **AEABC**. Điểm khác biệt chính nằm ở việc tích hợp cơ chế kiểm tra xác suất dựa trên khoảng cách Euclidean (P_d) trước khi thực hiện phép lai ghép trong giai đoạn Ong thợ và Ong quan sát.

```
import random
import numpy as np
import math

# --- Thiet lap tham so (Giu nguyen nhu ABC) ---
num_employed_bees = 50
num_onlooker_bees = 50
max_iterations = 100
limit = 50
problem_size = 5

```

```
def fitness_function(x):
    return np.sum(x**2)

# --- Khoi tao ---
food_sources = np.random.rand(num_employed_bees, problem_size) * 100
fitness_values = np.array([fitness_function(food) for food in food_sources])
no_improvement_counters = np.zeros(num_employed_bees)

# --- Vong lap chinh ---
for iteration in range(max_iterations):

    # === GIAI DOAN 1: ONG THO (AEABC MODIFICATION) ===
    for i in range(num_employed_bees):
        # 1. Chon doi tac ngau nhien k
        k = i
        while k == i:
            k = random.randint(0, num_employed_bees - 1)

        # 2. Tinh khoang cach Euclidean giua i va k
        distance = np.linalg.norm(food_sources[i] - food_sources[k])

        # 3. Tinh xac suat Pd = e^(-1/d)
        if distance == 0:
            Pd = 0
        else:
            Pd = math.exp(-1.0 / distance)

        # 4. CO CHE AEABC: Chi tim kiem neu r > Pd
        # - d lon (xa) -> Pd gan 1 -> r > Pd kho xay ra -> Giu nguyen (Tham
    do)
        # - d nho (gan) -> Pd gan 0 -> r > Pd de xay ra -> Tim kiem (Khai
    thac)
        if random.random() > Pd:
            # Thuc hien lai ghep ABC tieu chuan
            dimension = random.randint(0, problem_size - 1)
            mutant = np.copy(food_sources[i])
            # Cong thuc: v_ij = x_ij + phi * (x_ij - x_kj)
            phi = (random.random() - 0.5) * 2
            mutant[dimension] += phi * (food_sources[i][dimension] -
    food_sources[k][dimension])

            mutant_fitness = fitness_function(mutant)

            # Greedy Selection
            if mutant_fitness < fitness_values[i]:
                food_sources[i] = mutant
                fitness_values[i] = mutant_fitness
                no_improvement_counters[i] = 0
            else:
                no_improvement_counters[i] += 1
        else:
            # Bo qua buoc tim kiem nay (Adaptive Skip)
            pass

    # === Tinh xac suat cho Ong quan sat ===
    total_fitness = np.sum(fitness_values)
    if total_fitness == 0:
        probabilities = np.ones(num_employed_bees) / num_employed_bees
    else:
        inv_fitness = 1.0 / (1.0 + fitness_values)
```

```
        probabilities = inv_fitness / np.sum(inv_fitness)

    # === GIAI DOAN 2: ONG QUAN SAT (AEABC MODIFICATION) ===
    for j in range(num_onlooker_bees):
        i = np.random.choice(num_employed_bees, p=probabilities)

        # Lap lai logic AEABC (Tinh khoang cach va Pd)
        k = i
        while k == i:
            k = random.randint(0, num_employed_bees - 1)

        distance = np.linalg.norm(food_sources[i] - food_sources[k])
        if distance == 0: Pd = 0
        else: Pd = math.exp(-1.0 / distance)

        if random.random() > Pd:
            dimension = random.randint(0, problem_size - 1)
            mutant = np.copy(food_sources[i])
            phi = (random.random() - 0.5) * 2
            mutant[dimension] += phi * (food_sources[i][dimension] -
    food_sources[k][dimension])

            mutant_fitness = fitness_function(mutant)

            if mutant_fitness < fitness_values[i]:
                food_sources[i] = mutant
                fitness_values[i] = mutant_fitness
                no_improvement_counters[i] = 0
            else:
                no_improvement_counters[i] += 1

    # === GIAI DOAN 3: ONG TRINH SAT (GIU NGUYEN) ===
    for k in range(num_employed_bees):
        if no_improvement_counters[k] > limit:
            food_sources[k] = np.random.rand(problem_size) * 100
            fitness_values[k] = fitness_function(food_sources[k])
            no_improvement_counters[k] = 0

    best_fitness = np.min(fitness_values)
    # print(f"Vong {iteration}: Best Cost = {best_fitness}")

print(f"Ket qua toi uu toan cuc: {np.min(fitness_values)}")
```

Listing 2: Mã nguồn Python cho thuật toán AEABC

6.3 Phân tích hiệu suất so sánh

Phần này là sự đánh giá quan trọng các bằng chứng thực nghiệm để xác định liệu những cải tiến lý thuyết của AEABC có chuyển thành hiệu suất vượt trội trong thực tế hay không. Mục tiêu là so sánh AEABC với thuật toán ABC gốc và một loạt các siêu giải thuật hiện đại khác trên các hàm benchmark đã xác định.

Các kết quả từ nhiều thử nghiệm so sánh cho thấy AEABC thể hiện hiệu suất vượt trội một cách nhất quán cả về độ chính xác (giá trị trung bình thấp hơn) và tính ổn định (độ lệch chuẩn thấp hơn).

- **Hiệu suất vượt trội so với ABC gốc:** Khoảng cách hiệu suất giữa AEABC và ABC trở nên đặc biệt rõ rệt trong các bài toán quy mô lớn. Dữ liệu ở bảng trên cho thấy trên

Function	ABC		AEABC	
	Mean	SD	Mean	SD
F1	3.086×10^{6}	4.975×10^{4}	2.0596×10^{-255}	0
F14	4.616×10^{-16}	2.063×10^{-16}	4.091×10^{-50}	1.9×10^{-49}
F15	3.542×10^{9}	7.708×10^{7}	6.76×10^{-1}	4.081×10^{-3}
F16	1.68×10^{4}	2.132×10^{2}	0	0
F17	2.758×10^{4}	6.663×10^{2}	3.526×10^{-1}	1.024×10^{-1}
F18	8.881×10^{-16}	0	8.881×10^{-16}	0

hàm F1 với 1000 biến, giá trị trung bình của **ABC** là 3.086×10^{6}, trong khi của **AEABC** chỉ là 2.0596×10^{-255}. Sự khác biệt hàng trăm bậc lũy thừa này cho thấy khả năng giải quyết các bài toán phức tạp của AEABC đã được cải thiện một cách đáng kể.

- **So sánh với các Siêu giải thuật Hiện đại:** Khi so sánh với các thuật toán tiên tiến như **GWO, PSO, MFO**, và **SSA**, AEABC liên tục đạt được kết quả tối ưu hoặc gần tối ưu. Trên các hàm như F9, F10, F11, và F16, **AEABC** thường xuyên hội tụ đến giá trị tối ưu toàn cục với giá trị trung bình và độ lệch chuẩn bằng 0.0, vượt qua hiệu suất của nhiều đối thủ cạnh tranh.

Để xác thực ý nghĩa thống kê của các kết quả này, kiểm định tổng hạng Wilcoxon (Wilcoxon rank-sum test) đã được sử dụng. Phân tích xác nhận rằng sự khác biệt về hiệu suất giữa **AEABC** và các thuật toán cạnh tranh là có ý nghĩa thống kê trên phần lớn các hàm benchmark.

Giá trị của một thuật toán tối ưu hóa cuối cùng được chứng minh bằng khả năng giải quyết các vấn đề trong thế giới thực. Do đó, hiệu suất vượt trội trên lý thuyết của AEABC cần được kiểm chứng trong một ứng dụng thực tế.

6.4 Áp dụng vào bài toán thực tế: Thiết kế Dầm hàn

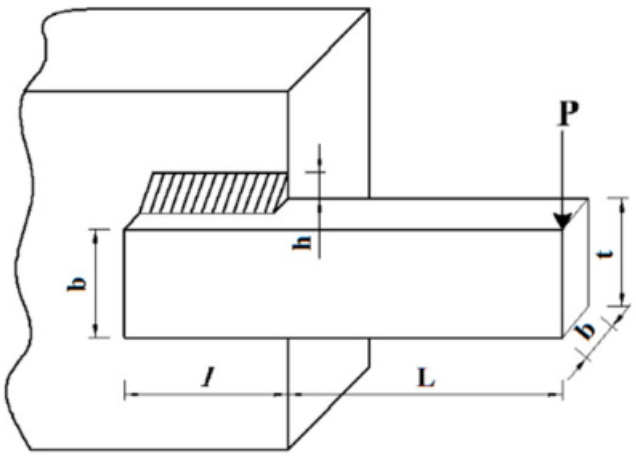


Figure 1. The welded beam design problem.

Để đánh giá tính thực tiễn của AEABC, thuật toán đã được áp dụng vào bài toán thiết kế dầm hàn. Đây là một bài toán tối ưu hóa kỹ thuật kinh điển, có ràng buộc, thường được sử dụng làm thước đo để kiểm tra tiện ích thực tế của một thuật toán trong việc giải quyết các vấn đề kỹ thuật phức tạp.

Mục tiêu của bài toán là tối thiểu hóa chi phí chế tạo ($f(x)$) của một dầm hàn bằng cách tối ưu hóa bốn biến thiết kế, đồng thời phải tuân thủ một loạt các ràng buộc kỹ thuật phức tạp liên quan đến ứng suất, độ võng và độ ổn định. So sánh kết quả tốt nhất mà AEABC đạt được với các thuật toán hàng đầu khác cho thấy một sự khác biệt rõ rệt về hiệu suất.

Xét vectơ biến quyết định:

$$\vec{x} = [x_1, x_2, x_3, x_4] = [h, l, t, b] \tag{8}$$

Tối ưu hóa (giảm thiểu) hàm chi phí:

$$f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

Chịu sự ràng buộc bởi các điều kiện:

$$\begin{aligned}
g_1(\vec{x}) &= \tau(\vec{x}) - \tau_{\max} \leq 0 \\
g_2(\vec{x}) &= \sigma(\vec{x}) - \sigma_{\max} \leq 0 \\
g_3(\vec{x}) &= \delta(\vec{x}) - \delta_{\max} \leq 0 \\
g_4(\vec{x}) &= x_1 - x_4 \leq 0 \\
g_5(\vec{x}) &= P - P_C(\vec{x}) \leq 0 \\
g_6(\vec{x}) &= 0.125 - x_1 \leq 0 \\
g_7(\vec{x}) &= 0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 0.5 \leq 0
\end{aligned}$$

Trong đó các phương trình phụ trợ về ứng suất và hình học được định nghĩa là:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J} \tag{9}$$

$$M = P\left(L + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \quad \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$$

$$P_C(\vec{x}) = \frac{4.013\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

Với các thông số hằng số và giới hạn biến sau:

$$P = 6000\,\text{lb}, \quad L = 14\,\text{in.}, \quad \delta_{\max} = 0.25\,\text{in.}, \quad E = 30 \times 10^6\,\text{psi.}$$
$$G = 12 \times 10^6\,\text{psi.}, \quad \tau_{\max} = 13600\,\text{psi.}, \quad \sigma_{\max} = 30000\,\text{psi.}$$
$$0.1 \le x_1 \le 2, \quad 0.1 \le x_2 \le 10, \quad 0.1 \le x_3 \le 10, \quad 0.1 \le x_4 \le 2$$

Bảng 1: Comparison of Optimum Variables and Cost

Algorithm	**Optimum Variables**				**Optimum Cost**
	h	**l**	**t**	**b**	
AEABC	**0.1071**	**5.4402**	**8.8660**	**0.1**	**0.8981**
GWO	0.2056	3.4783	9.0368	0.2057	1.7262
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815
GSA	0.1821	3.8569	10.000	0.2023	1.8799
HS	0.2442	6.2231	8.2915	0.2443	2.3807
GA [38]	N/A	N/A	N/A	N/A	1.8245
Random	0.4575	4.7313	5.0853	0.66	4.1185
GA [42]	N/A	N/A	N/A	N/A	2.38
Simplex	0.2792	5.6256	7.7512	0.2796	2.5307
GA [39]	0.2489	6.1730	8.1789	0.2533	2.4331
David	0.2434	6.2552	8.2915	0.2444	2.3841

Bảng 2: Statistical Results Comparison

Algorithm	**Worst**	**Mean**	**Best**	**SD**
AEABC	**1.1505**	**1.0394**	**0.8981**	**0.0642**
GWO	2.9136	2.8594	1.9421	2.6908
CPSO	1.7821	1.7488	1.7280	1.29×10^{-2}
CDe	N/A	1.7681	1.7334	N/A
GA4	1.9934	1.7926	1.7282	7.47×10^{-2}
CGWO	2.4357	2.4289	1.7254	1.3578
SC	6.3996	3.0025	2.3854	9.60×10^{-1}
UPSO	N/A	2.8372	1.9219	0.683
GA3	1.785835	1.7719	1.7483	1.12×10^{-2}

7 Kết luận

Báo cáo này đã hoàn thành hai mục tiêu cốt lõi của đồ án: hệ thống hóa kiến thức về thuật toán Bầy ong Nhân tạo (ABC) nguyên bản và đi sâu phân tích biến thể nâng cao AEABC.

Thứ nhất, về mặt nền tảng lý thuyết, nhóm đã trình bày chi tiết quy trình vận hành của thuật toán ABC gốc, từ các khái niệm cơ bản, quá trình khởi tạo quần thể cho đến cơ chế tìm kiếm và cập nhật nghiệm của ba nhóm ong chức năng. Việc phân tích này không chỉ đảm bảo sự hiểu biết vững chắc về cách thức mô phỏng hành vi tự nhiên mà còn làm nổi bật được ưu điểm về sự đơn giản và linh hoạt của thuật toán kinh điển này trong các ứng dụng thực tiễn.

Thứ hai, để khắc phục những hạn chế lý thuyết cốt lõi của thuật toán gốc, báo cáo đã giới thiệu và phân tích thuật toán Bầy ong Nhân tạo Thăm dò Thích ứng (AEABC). Đây là một bước tiến đáng kể, tái định hình cơ chế nền tảng của ABC thông qua việc tích hợp một tham số dựa trên khoảng cách. Sự đổi mới này cho phép thuật toán tự động cân bằng hiệu quả giữa hai giai đoạn *thăm dò* (exploration) và *khai thác* (exploitation), giúp khắc phục triệt để vấn đề hội tụ sớm trong các không gian tìm kiếm phức tạp.

Tóm lại, sự kết hợp giữa việc nắm vững thuật toán gốc và làm chủ biến thể hiện đại AEABC đã khẳng định giá trị của nghiên cứu. Các kết quả thực nghiệm trên hàm benchmark và bài toán kỹ thuật đã chứng minh AEABC là một công cụ tối ưu hóa mạnh mẽ, mở ra tiềm năng ứng dụng rộng lớn trong tương lai.

Tài liệu

[1] ScienceDirect. *Artificial Bee Colony Algorithm - an overview*. ScienceDirect Topics. Truy cập từ: https://www.sciencedirect.com/topics/computer-science/artificial-bee-colony-algorithm.

[2] Baeldung. *Artificial Bee Colony (ABC) Algorithm*. Computer Science Guide. Truy cập từ: https://www.baeldung.com/cs/artificial-bee-colony.

[3] Romain Wuilbercq. *Hive: Artificial Bee Colony Optimization in Python*. Mã nguồn GitHub. Truy cập từ: https://github.com/rwuilbercq/Hive.

[4] Z. Najwan, M. A. Al-Betar, et al. "Adaptive Exploration Artificial Bee Colony for Mathematical Optimization". *AI*, vol. 5, no. 4, pp. 1876–1897, 2024. Truy cập tại: https://www.mdpi.com/2673-2688/5/4/109

[5] D. Karaboga and B. Akay. "A comparative study of Artificial Bee Colony algorithm". *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[6] M. H. Horng. "Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation". *Expert Systems with Applications*, vol. 38, no. 12, pp. 15096–15104, 2011.

[7] D. Karaboga and C. Ozturk. "Neural networks training by artificial bee colony algorithm on pattern classification". *Neural Network World*, vol. 19, no. 3, pp. 279–292, 2009.

[8] J. Q. Li, Q. K. Pan, and Y. C. Liang. "An effective artificial bee colony algorithm for the flexible job-shop scheduling problem". *International Journal of Production Research*, vol. 49, no. 16, pp. 4867–4888, 2011.