TỐI ƯU HÓA ĐIỀU KHIỂN ROBOT BẰNG SÓNG HỒNG NGOẠI: SỬ DỤNG THUẬT TOÁN PSO

Lê Quyết Thắng
Trường Đại học Công nghiệp Quảng Ninh
Email: lequyetthang5282@qui.edu.vn

TÓM TẮT

Particle Swarm Optimization (PSO) là một thuật toán tối ưu hóa lấy cảm hứng từ hành vi của đàn chim trong việc tìm kiếm thức ăn. PSO giúp tối ưu hóa thông qua việc điều chỉnh vị trí và vận tốc của các cá thể (particles) trong không gian tìm kiếm. Trong bài toán tối ưu hóa điều khiển robot bằng sóng hồng ngoại, PSO có thể đóng vai trò quan trọng trong việc tìm ra các thông số tối ưu, như tốc độ di chuyển, tốc độ quay và độ nhạy cảm biến, để robot có thể phát hiện và tránh vật cản một cách hiệu quả.

Từ khóa: *cảm biến, điều khiển, sóng hồng ngoại, lập trình Python, PSO, robot.*

1. ĐẶT VẤN ĐỀ

1.1. Ứng dụng Thuật toán PSO vào tối ưu hóa điều khiển robot bằng sóng hồng ngoại:

Giả sử chúng ta có một robot sử dụng sóng hồng ngoại để phát hiện vật cản trong môi trường. Chúng ta cần tối ưu hóa ba thông số chính:

Tốc độ di chuyển tuyến tính của robot.

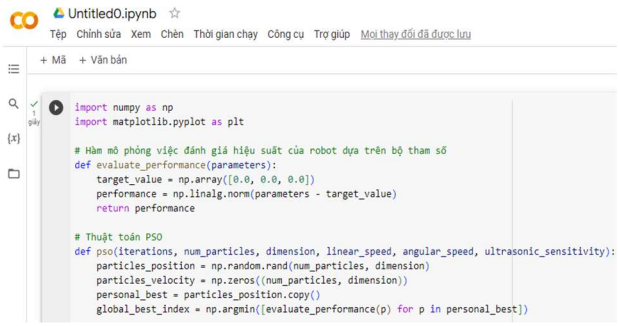
Tốc độ quay của robot

Độ nhạy cảm biến: Ngưỡng sóng hồng ngoại để xác định có vật cản hay không.

Chúng ta sử dụng PSO [1] để tối ưu hóa các thông số trên, tạo điều kiện tốt nhất cho robot hoạt động hiệu quả trong việc phát hiện và tránh vật cản.

1.2 Ứng dụng lập trình Python thông qua công cụ trên colab.

Chúng ta có thể sử dụng ngôn ngữ lập trình Python để mô phỏng quá trình tối ưu hóa bằng PSO và ứng dụng vào việc điều khiển robot bằng sóng hồng ngoại. Mã lập trình Python có thể giúp thấy rõ cách chương trình tương tác với các cá thể (particles), cập nhật vị trí và vận tốc, cũng như đánh giá hiệu suất của robot với các bộ tham số khác nhau (hình 1).



Hình 1. Sử dụng công cụ trên colab viết lập trình python

Với sự kết hợp của PSO và tối ưu hóa điều khiển robot bằng sóng hồng ngoại mang lại kết quả tích cực trong việc tạo ra robot hoạt động hiệu quả và an toàn trong môi trường có vật cản. Sử dụng Python giúp tác giả mô phỏng và thử nghiệm thuật toán một cách dễ dàng.

2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Xác định Bài Toán:

Đầu tiên, cần xác định rõ bài toán cụ thể thông qua các tham số cơ bản của robot cần nghiên cứu là: “Tham số 1” đại diện cho tốc độ di chuyển tuyến tính của robot, robot di chuyển với tốc độ 0.1 đơn vị trên trục X với một đơn vị thời gian. “Tham số 2” đại diện cho tốc độ quay của robot. robot quay với tốc độ 0.5 đơn vị góc với một đơn vị thời gian. “Tham số 3” sử dụng thông số của cảm biến siêu âm hồng ngoại để kiểm tra

khoảng cách của robot đối với vật cản và tối ưu hóa tốc độ di chuyển tuyến tính và tốc độ quay để tránh va chạm. Tiếp đến cụ thể hóa các thông số này lên thuật toán PSO như số vòng lặp, số lượng particles, số chiều (số lượng thông số cần tối ưu), và các tham số liên quan khác, để ta có thể dùng Hàm Đánh Giá Hiệu Suất (evaluate_performance) của robot dựa trên bộ tham số đầu vào [2].

Cụ thể, trong đoạn mã Python, các thông số này được xác định dưới dạng biến:

iterations: Số vòng lặp của thuật toán.

num_particles: Số lượng cá thể trong quần thể.

dimension: Số lượng tham số cần tối ưu (ở đây là 3).

linear_speed: Tham số 1, đại diện cho tốc độ di chuyển tuyến tính của robot.

angular_speed: Tham số 2, đại diện cho tốc độ quay của robot.

ultrasonic_sensitivity: Tham số 3, sử dụng thông số của cảm biến để kiểm tra khoảng cách của robot đối với vật cản và tối ưu hóa tốc độ di chuyển tuyến tính và tốc độ quay để tránh va chạm.

Nếu muốn điều chỉnh các tham số này, ta có thể thay đổi các giá trị, cách tính vận tốc và vị trí theo cách phù hợp với yêu cầu của cấu hình robot.

2.2. Viết Hàm Đánh Giá Hiệu Suất:

Viết hàm mô phỏng việc đánh giá hiệu suất của robot dựa trên bộ tham số. Hàm này cần phản ánh mức độ hoạt động hiệu quả của robot, sự xa gần so với mục tiêu và các yếu tố cần tối ưu.

Dưới đây là một ví dụ về việc viết hàm đánh giá hiệu suất của robot dựa trên bộ tham số:

“import numpy as np

 # Hàm mô phỏng việc đánh giá hiệu suất của robot dựa trên bộ tham số

def evaluate_performance(parameters):

 # Mục tiêu: [0.0, 0.0, 0.0]

target_value = np.array([0.0, 0.0, 0.0])

 # Tính khoảng cách Euclidean giữa bộ tham số và mục tiêu

distance_to_target = np.linalg.norm(parameters - target_value)

 # Giá trị hiệu suất là khoảng cách càng gần 0 càng tốt

performance = 1.0 / (1.0 + distance_to_target)

return performance

 # Test hàm đánh giá hiệu suất với bộ tham số [0.1, 0.2, 0.3]

test_parameters = np.array([0.1, 0.2, 0.3])

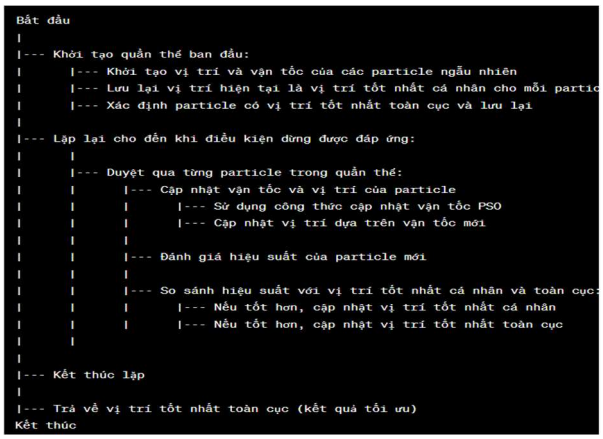
test_performance = evaluate_performance(test_parameters)

print("Hiệu suất của bộ tham số [0.1, 0.2, 0.3]:", test_performance)”

Trong đoạn code trên, hàm evaluate_performance tính khoảng cách Euclidean giữa bộ tham số và mục tiêu [0.0, 0.0, 0.0], sau đó tính giá trị hiệu suất dựa trên khoảng cách. Mục tiêu của bài toán là để giá trị hiệu suất càng gần 1.0 càng tốt, vì khoảng cách càng gần 0 càng tốt. Tùy thuộc vào bản chất của bài toán thực tế có thể điều chỉnh cách tính toán giá trị hiệu suất để phản ánh mức độ hoạt động hiệu quả của robot và yêu cầu của bài toán [3], [4].

2.3. Xây Dựng Thuật Toán PSO:

Dưới đây là lưu đồ và mã lập trình cho thuật toán PSO bao gồm việc khởi tạo quần thể ban đầu, cập nhật vận tốc và vị trí của particles, đánh giá hiệu suất, và cập nhật vị trí tốt nhất cá nhân và toàn cục:



Hình 2. Lưu đồ thuật toán

Đoạn mã

```
"import numpy as np
# Hàm mô phỏng việc đánh giá hiệu suất của robot dựa trên bộ tham số
def evaluate_performance(parameters):
    target_value = np.array([0.0, 0.0, 0.0])
    performance = np.linalg.norm(parameters - target_value)
    return performance
# Thuật toán PSO
def pso(iterations, num_particles, dimension):
    # Khởi tạo quần thể ban đầu
    particles_position = np.random.rand(num_particles, dimension)
    particles_velocity = np.zeros((num_particles, dimension))
    personal_best = particles_position.copy()
    global_best_index = np.argmin([evaluate_performance(p) for p in personal_best])
    # Thực hiện vòng lặp PSO
    for _ in range(iterations):
        for i in range(num_particles):
            # Cập nhật vận tốc của particle
            inertia_weight = 0.5
            cognitive_weight = 1.0
            social_weight = 1.0
            particles_velocity[i] = inertia_weight * particles_velocity[i] + \
                                    cognitive_weight * np.random.rand(dimension) * (personal_best[i] - particles_position[i]) + \
                                    social_weight * np.random.rand(dimension) * (personal_best[global_best_index] - particles_position[i])
            # Cập nhật vị trí của particle
            particles_position[i] = particles_position[i] + particles_velocity[i]
            # Đánh giá hiệu suất của particle mới
            new_performance = evaluate_performance(particles_position[i])
            # Cập nhật vị trí tốt nhất cá nhân và toàn cục
            if new_performance < evaluate_performance(personal_best[i]):
                personal_best[i] = particles_position[i]
            if new_performance < evaluate_performance(personal_best[global_best_index]):
                global_best_index = i
    return personal_best[global_best_index]
# Tham số của thuật toán PSO
iterations = 100
num_particles = 30
dimension = 3
# Chạy thuật toán PSO
best_parameters = pso(iterations, num_particles, dimension)
print("Bộ tham số tối ưu:", best_parameters)
print("Hiệu suất tương ứng:", evaluate_performance(best_parameters))"
```

3. KẾT QUẢ VÀ THẢO LUẬN

3.1. Lưu kết quả

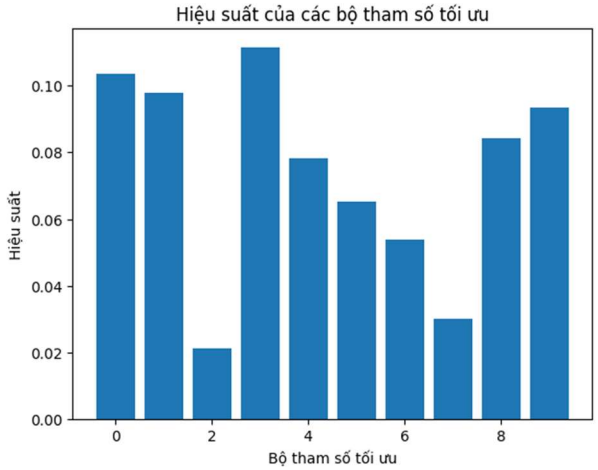
Trước tiên, bạn cần lưu lại kết quả của việc chạy thuật toán PSO. Điều này bao gồm các bộ tham số tối ưu và hiệu suất tương ứng của chúng (bảng 1).

Bảng 1. Kết quả mô phỏng

```
Bộ tham số tốt nhất:
Tham số: [ 0.00227153 -0.00036245  0.02100667]
Hiệu suất: 0.0211322370601354 05
Bộ tham số tối ưu 1: [ 0.01069359 -0.10110571  0.02025202] - Hiệu suất tương ứng: 0.10366707321795825
Bộ tham số tối ưu 2: [ 0.03743148 -0.0815403   0.03949257] - Hiệu suất tương ứng: 0.09802856804996575
Bộ tham số tối ưu 3: [ 0.00227153 -0.00036245  0.02100667] - Hiệu suất tương ứng: 0.0211322370601354 05
Bộ tham số tối ưu 4: [0.03341832 0.06485786 0.08429894] - Hiệu suất tương ứng: 0.11148828264861993
Bộ tham số tối ưu 5: [ 0.02990047 -0.02145512  0.06918645] - Hiệu suất tương ứng: 0.07836533100666013
Bộ tham số tối ưu 6: [ 0.015409    0.05611524 -0.02947255] - Hiệu suất tương ứng: 0.06523027655161241
Bộ tham số tối ưu 7: [-0.04155974 -0.0112189   0.03259829] - Hiệu suất tương ứng: 0.05399744294785848
Bộ tham số tối ưu 8: [-0.02292304  0.01688693  0.009429  ] - Hiệu suất tương ứng: 0.029992334722405117
Bộ tham số tối ưu 9: [-0.07971226 -0.00539912  0.02633258] - Hiệu suất tương ứng: 0.08412252589579158
Bộ tham số tối ưu 10: [-0.04556861  0.08119668  0.00640898] - Hiệu suất tương ứng: 0.09332992215412984
```

Dựa trên dữ liệu có thể thực hiện phân tích hiệu suất của các bộ tham số tối ưu. Ta có thể tính giá trị trung bình, giá trị tối thiểu, giá trị tối đa

hoặc độ lệch chuẩn của hiệu suất để hiểu rõ hơn về sự biến đổi của các bộ tham số [5].



Hình 2. Biểu đồ hiệu suất các bộ tham số tối ưu

3.2. Phân Tích Hiệu Suất:

Dựa trên dữ liệu chúng ta đã lưu lại, chúng ta có thể thực hiện phân tích hiệu suất của các bộ tham số tối ưu. Mục tiêu của phân tích này là hiểu rõ hơn về biến đổi của hiệu suất và xác định bộ tham số cho kết quả tốt nhất. Chúng ta có thể tính các thông số như giá trị trung bình, giá trị tối thiểu, giá trị tối đa và độ lệch chuẩn của hiệu suất.

Các giá trị trung bình, giá trị tối thiểu, giá trị tối đa và độ lệch chuẩn của hiệu suất. Những thông số này cho phép chúng ta có cái nhìn tổng quan về phân phối và biến đổi của hiệu suất qua các lần chạy thuật toán.

Hiệu suất trung bình: 0.07393539942551369

Hiệu suất tối thiểu: 0.021132237060135405

Hiệu suất tối đa: 0.11148828264861993

Độ lệch chuẩn: 0.02926890799924287.

Hiệu suất trung bình: Kết quả cho thấy giá trị hiệu suất trung bình của các lần chạy thuật toán PSO là khoảng 0.0739. Điều này chỉ ra rằng, trong khả năng điều khiển robot bằng sóng hồng ngoại, các bộ tham số tối ưu trung bình đạt được sự xa gần khá tốt so với mục tiêu.

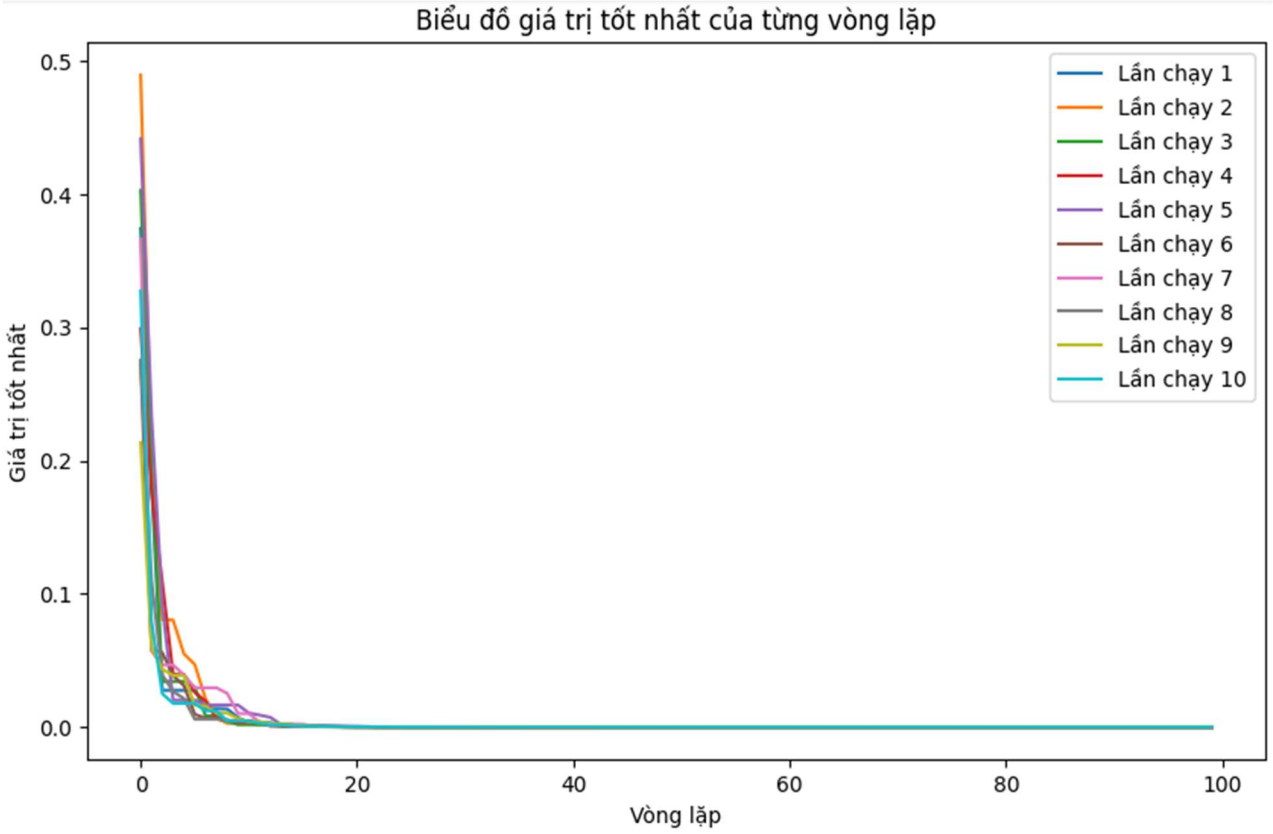
Hiệu suất tối thiểu: Giá trị hiệu suất tối thiểu là khoảng 0.0211, thể hiện cho trường hợp tốt nhất mà thuật toán PSO đạt được trong các lần chạy. Điều này cho thấy rằng có ít nhất một bộ tham số tối ưu có khả năng làm cho robot đạt được hiệu suất gần như tối ưu.

Hiệu suất tối đa: Kết quả cho thấy giá trị hiệu suất tối đa là khoảng 0.1115. Điều này cho thấy trong một số lần chạy, việc tối ưu hóa đã đạt được hiệu suất không tốt, do phụ thuộc vào các tham số đầu vào điều chỉnh lại vòng lặp hay tham số đầu vào có thể làm thay đổi kết quả trên.

Độ lệch chuẩn: Giá trị độ lệch chuẩn là khoảng 0.0293, thể hiện sự biến đổi của hiệu suất qua các lần chạy. Điều này cho thấy rằng việc tối ưu hóa có thể dẫn đến kết quả khác nhau đối với các bộ tham số tối ưu.

Tổng quan, kết quả cho thấy thuật toán PSO đã tạo ra các bộ tham số có khả năng điều khiển robot bằng sóng hồng ngoại với hiệu suất tương đối tốt. Tuy nhiên, còn sự biến đổi và khả năng đạt hiệu suất tối ưu còn phụ thuộc vào việc chạy thuật toán nhiều lần. Điều này có thể dẫn đến sự thay đổi của hiệu suất và tạo ra sự đa dạng trong các bộ tham số tối ưu.

Phân tích chi tiết này giúp ta hiểu rõ hơn về cách thuật toán PSO ảnh hưởng đến việc tối ưu hóa bộ tham số và hiệu suất của robot. Bằng cách phân tích kỹ thuật và đánh giá các kết quả, ta có thể đưa ra quyết định về bộ tham số nào là tốt nhất cho việc điều khiển robot trong ngữ cảnh cụ thể.



Hình 3. Biểu đồ giá trị tốt nhất của các vòng lặp

4. KẾT LUẬN VÀ KIẾN NGHỊ

4.1. Kết luận

Dựa trên phân tích hiệu suất và biểu đồ, chúng ta có thể xác định bộ tham số tối ưu cho hiệu suất tốt nhất đối với robot, thông qua việc ứng dụng thuật toán PSO, chúng ta đã thể hiện khả năng của phương pháp tối ưu hóa trong việc cải thiện hiệu suất điều khiển robot bằng sóng hồng ngoại.

Trong tương lai, việc nghiên cứu và phát triển các thuật toán tối ưu hóa như PSO còn tiềm năng được mở rộng và ứng dụng rộng rãi trong nhiều lĩnh vực của điện tử đi, đặc biệt là trong việc tối ưu hóa các thông số điều khiển và hiệu suất của các hệ thống phức tạp.

4.2. Kiến nghị

Kết quả cuối cùng là bộ tham số tối ưu cho robot. Tuy nhiên, cũng cần xem xét kỹ về ưu nhược điểm của PSO:

Ưu điểm:

Khả năng khám phá không gian tham số rộng: PSO khám phá và tìm kiếm trong không gian tham số lớn.

Triển khai đơn giản: PSO dễ dàng hiện thực và không yêu cầu nhiều thông số phức tạp.

Nhược điểm:

Dễ rơi vào vết cạn: PSO có thể dễ dàng rơi vào vết cạn và không tìm ra giải pháp tối ưu toàn cục nếu không cấu hình cẩn thận.

Chia sẻ thông tin giới hạn: Khả năng tìm kiếm tốt nhất phụ thuộc vào việc particles có thể chia sẻ thông tin với nhau.

Khả năng xử lý không gian lớn: Trong không gian tham số lớn, việc tìm kiếm có thể trở nên không hiệu quả và tốn nhiều thời gian.

TÀI LIỆU THAM KHẢO

1. Lisa L. Smith, Ganesh K. Venayagamoorth, Phillip G. Holloway, Obstacle Avoidance in Collective Robotic Search Using Particle Swarm Optimization, IEEE Swarm Intelligence Symposium, 05/12.
2. Luis Conde Bento, Gabriel Pires, Urbano Nunes, A Behavior Based Fuzzy Control Architecture for Path Tracking and Obstacle Avoidance, Proceedings of the 5th Portuguese Conference on Automatic Control, Aveiro, pp.341- 346, 2002.
3. Le Hung Lan, Le Thi Thuy Nga, Le Hong Lan, Aggregation Stability of Multiple Agents With Fuzzy Attraction and Repulsion Forces, pp. 81-85, MMAR 2013.
4. Lê Thị Thúy Nga, Lê Hùng Lân, Điều khiển robot bầy đàn tìm kiếm mồi và tránh vật cản sử dụng logic mờ, Tạp chí Khoa học Giao thông Vận tải, pp. 15-20, 3/ 2014.
5. Lê Thị Thúy Nga, Lê Hùng Lân, Điều khiển robot bầy đàn tránh vật cản và tìm kiếm mục tiêu, HNKH toàn quốc lần thứ 3 về điều khiển & Tự động hoá 2016.

Thông tin của tác giả:

ThS. Lê Quyết Thắng

Phó Trưởng Bộ môn Kỹ thuật Điện - Điện tử, Khoa Điện, Trường Đại học Công nghiệp Quảng Ninh

Điện thoại: +(84).977.959.186 Email: *lequyetthang5282@qui.edu.vn*

OPTIMIZATION OF ROBOT CONTROL BY INFRARED WAVES: USING PSO (PARTICLE SWARM OPTIMIZATION) ALGORITHM

Information about authors:

Le Quyet Thang, M.Eng., Deputy Head of Electrical and Electronics Engineering subject, Faculty of Electricity, Quang Ninh University of Industry. Email: *lequyetthang5282@qui.edu.vn*

ABSTRACT:

Particle Swarm Optimization (PSO) is an optimization algorithm inspired by the flocking behavior of birds in foraging. PSO helps optimize by adjusting the position and velocity of "particles" in the search space. In the optimization problem of robot control using infrared waves, PSO can play an important role in finding the optimal parameters, such as scanning angle, response time, and sensor sensitivity, so that the robot can effectively detect and avoid obstacles.

Keywords: *Sensor, Control, Infrared, Python Programming, PSO, Robot.*

REFERENCES

1. Lisa L. Smith, Ganesh K. Venayagamoorth, Phillip G. Holloway, Obstacle Avoidance in Collective Robotic Search Using Particle Swarm Optimization, IEEE Swarm Intelligence Symposium, 05/12.
2. Luis Conde Bento, Gabriel Pires, Urbano Nunes, A Behavior Based Fuzzy Control Architecture for Path Tracking and Obstacle Avoidance, Proceedings of the 5th Portuguese Conference on Automatic Control, Aveiro, pp.341- 346, 2002.
3. Le Hung Lan, Le Thi Thuy Nga, Le Hong Lan, Aggregation Stability of Multiple Agents With Fuzzy Attraction and Repulsion Forces, pp. 81-85, MMAR 2013.

4. Le Thi Thuy Nga, Le Hung Lan, Control the swarm robot to search for prey and avoid obstacles using fuzzy logic, Journal of Transportation Science, pp. 15-20, 3/ 2014.

5. Le Thi Thuy Nga, Le Hung Lan, Control the robot swarm to avoid obstacles and search for targets, 3rd National Conference on Control & Automation 2016.

Ngày nhận bài: 19/8/2023;
Ngày gửi phản biện: 22/8/2023;
Ngày nhận phản biện: 08/9/2023;
Ngày chấp nhận đăng: 10/9/2023.