**GIẢI THUẬT TỐI ƯU ĐÀN KIẾN ĐỂ LẬP LỊCH ĐƯỜNG ĐI CHO ROBOT DI ĐỘNG TRONG MÔI TRƯỜNG TĨNH CÓ CHƯỚNG NGẠI VẬT**

Ngô Việt Hoàng

*Trường Đại học Thủy lợi, email: giangttc@tlu.edu.vn*

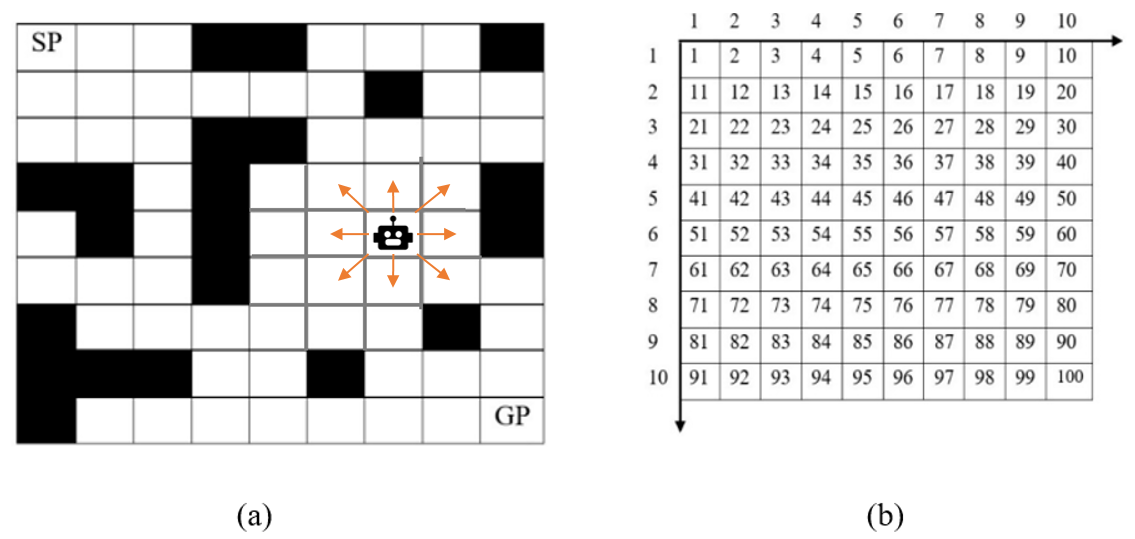
1. GIỚI THIỆU

Robot di động đang dần thay thế con người trong nhiều lĩnh vực khác nhau như công nghiệp, nông nghiệp, quân đội, cứu hộ,... Lập đường đi cho robot di động trở thành một trong những bài toán được nhiều nhà nghiên cứu quan tâm, nhằm tìm ra một hoặc nhiều đường đi khả thi, ngắn và an toàn để tối ưu năng lượng của robot khi di chuyển. Thuật toán tối ưu hóa đàn kiến (**A**nt **C**olony **O**ptimization Algorithm, viết tắt **ACO**) là một trong số những thuật toán được ứng dụng rộng rãi để lập đường đi cho robot di động do cơ chế phản hồi tích cực và hiệu quả trong tính toán phân tán. Thuật toán ACO vẫn có những nhược điểm như đường đi tìm được có những đoạn đường dư thừa, tốc độ hội tụ chậm và dễ rơi vào trạng thái tối ưu hóa cục bộ [2]. Bài báo này đề xuất cải tiến giải thuật tối ưu hóa đàn kiến (**I**mproved **A**nt **C**olony **O**ptimization Algorithm, viết tắt **IACO**) bằng cách kết hợp giải thuật ACO với giải thuật tìm kiếm theo chiều rộng (**B**readth **F**irst **S**earch, viết tắt **BFS**) và một số giải thuật khác nhằm khắc phục những nhược điểm của giải thuật ACO. Thông qua nhiều thực nghiệm, kết quả cho thấy rằng thuật toán IACO của chúng tôi tốt hơn giải thuật ACO về thời gian chạy cũng như độ dài đường đi.

2. KIẾN THỨC NỀN TẢNG

**2.1. Phương pháp phân rã ô lưới**

Bài báo này sử dụng phương pháp phân rã môi trường ô lưới (Cell decomposition approach) để mô hình hóa môi trường. Mỗi ô lưới chứa một thông tin nhị phân cho biết các ô bị cản trở bằng chướng ngại vật là “1” và các ô trống là “0” như trong *Hình 1.a*. Mỗi ô được xác định bởi một số được gọi là “địa chỉ” [1]. Địa chỉ của ô có thể được xác định bằng phương pháp: tọa độ lưới 2D (r, c), trong đó gốc tọa độ gốc là ô trên cùng bên trái của lưới với vị trí đầu tiên có tọa độ là (1,1) và mỗi ô được đánh trọng số như *Hình 1.b.*



**Hình 1.** *Phương pháp phân rã ô lưới*

Trong mô hình này, Robot có thể di chuyển đến 8 ô lân cận xung quanh ô robot đang đứng.

**2.2. Giải thuật tối ưu đàn kiến**

Giải thuật được đề xuất bởi Marco Dorigo vào năm 1991. Ý tưởng của thuật toán dựa hành vi đi lang thang và đánh dấu đường đi bằng pheromone của loài kiến khi tìm kiếm nguồn thức ăn. Đoạn đường nào có nhiều pheromone hơn thì đoạn đường đó thu hút nhiều con kiến đi qua hơn. Những con kiến đi sau sẽ lựa chọn những đoạn đường có nồng độ pheromone lớn nhất để đi. Theo thời gian, lượng pheromone trên đường đi sẽ bay hơi khiến nồng độ pheromone giảm đi dẫn đến chỉ còn lại những đoạn đường ngắn có nồng độ pheromone lớn. Những đoạn đường đó sẽ tiếp tục thu hút nhiều kiến hơn [2].

Hàm đánh giá của mỗi đoạn đường được tính như sau:

* Công thức xác suất chọn của mỗi đoạn đường của kiến k:

* Công thức cập nhật pheromone:

Trong đó:

𝜏i,j : lượng pheromone trên cạnh *i*,*j*

ηi,j : hàm heuristic ()

α: tham số kiểm soát ảnh hưởng của 𝜏i,j

β: tham số kiểm soát ảnh hưởng của ηi,j

allowedi: các đỉnh trong tập làng giềng của i chưa được đi qua.

di,j : khoảng cách giữa hai đỉnh *i*,*j*

𝜌: tốc độ bay hơi của pheromone.

*Lk*: độ dài quãng đường di chuyển của kiến k.

*Q*: tham số điều chỉnh lượng pheromone.

**2.3. Hàm đánh giá**

Độ dài đường đi Pa ngắn nhất:

*Length(Pa) =*

**3. GIẢI THUẬT ĐỀ XUẤT IACO**

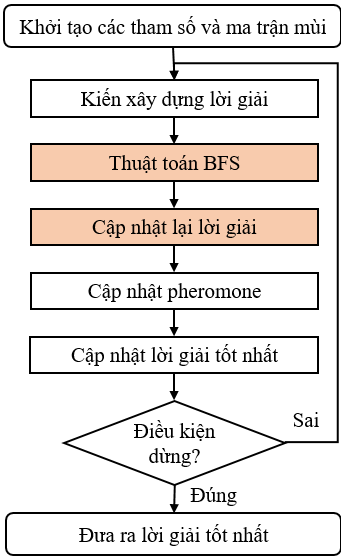
**Đầu vào:**

* Điểm xuất phát và điểm đích
* Tập m chướng ngại vật là các ô được xác định tọa độ

**Đầu ra:**

* Đường đi Pa = ( P0, P1,…, Pn) mà robot cần tìm và độ dài đường đi đó.

**Lưu đồ giải thuật cải tiến (***Hình 2***):**



**Hình 2**. *Lưu đồ giải thuật cải tiến IACO: lai ghép giữa thuật toán ACO và BFS*

**Công thức xác suất kích thích:**

**Công thức xác suất chọn:**

Trong đó:

Nobs: số lượng chướng ngại vật xung quanh

: tích vô hướng của vector từ *i* đến *j* và vector từ *i* đến điểm kết thúc

**Những điểm cải tiến:**

* Đề xuất kết hợp sử dụng giải thuật BFS để loại bỏ những đoạn đường không cần thiết để đường đi tìm được ngắn hơn.
* Để tránh đi vào những ngõ cụt và tăng tốc độ hội tụ của giải thuật ta sử dụng xác suất kích thích [2].
* Khắc phục việc những con kiến chọn những đoạn đường tối ưu cục bộ mà bỏ qua những điểm tối ưu toàn cục và tăng tốc độ hội tụ [3].

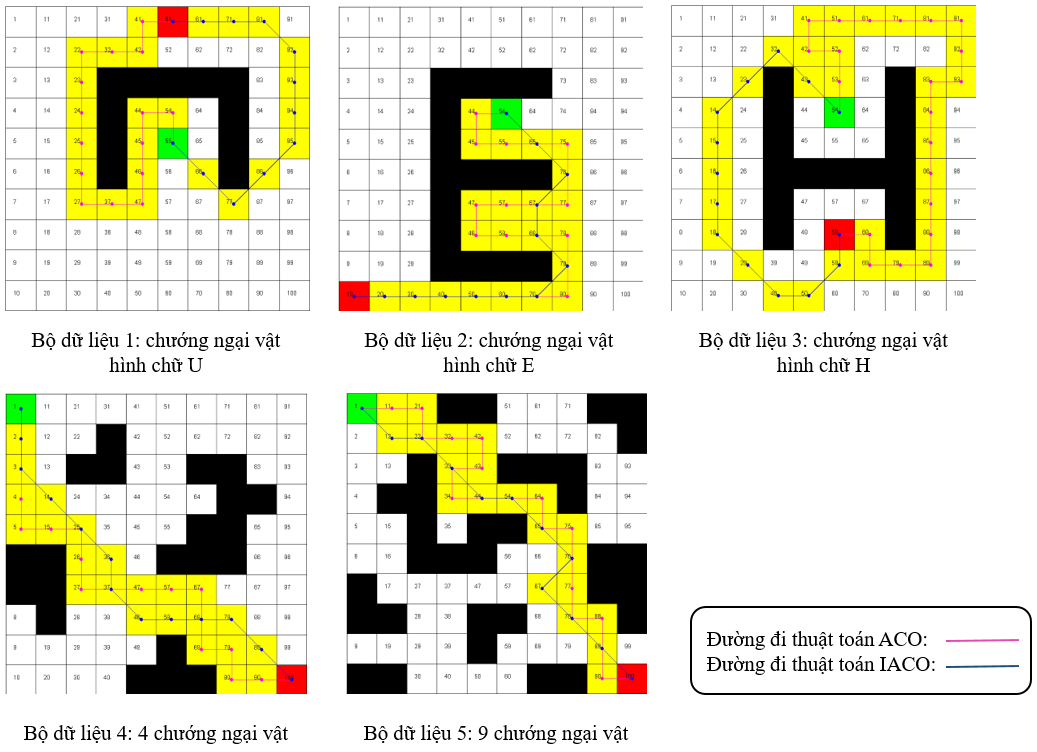
**4. KẾT QUẢ THỰC NGHIỆM**

Bài báo thực hiện 3 thực nghiệm sau để phân tích hiệu quả của giải thuật ACO và IACO dựa vào độ đo thời gian chạy trung bình của giải thuật (Ttb), độ dài trung bình (Ttb) và tỉ lệ thành công trung bình (Tltb):

* **Thực nghiệm 1:** Phân tích sự ảnh hưởng của số lượng và hình dạng chướng ngại vật.
* **Thực nghiệm 2:** Phân tích và đánh giá kết quả của IACO với các bộ dữ liệu mô phỏng không gian trong phòng thí nghiệm

**4.1. Ảnh hưởng của số lượng và hình dạng của chướng ngại vật**

Có 5 bộ dữ liệu khác nhau được sử dụng trong thực nghiệm này. Mỗi môi trường của trong bộ thí nghiệm có số lượng và hình dạng chướng ngại vật khác nhau nhằm đánh giá khả năng tìm đường của robot.

****

**Hình 3**. *Kết quả thực nghiệm 1*

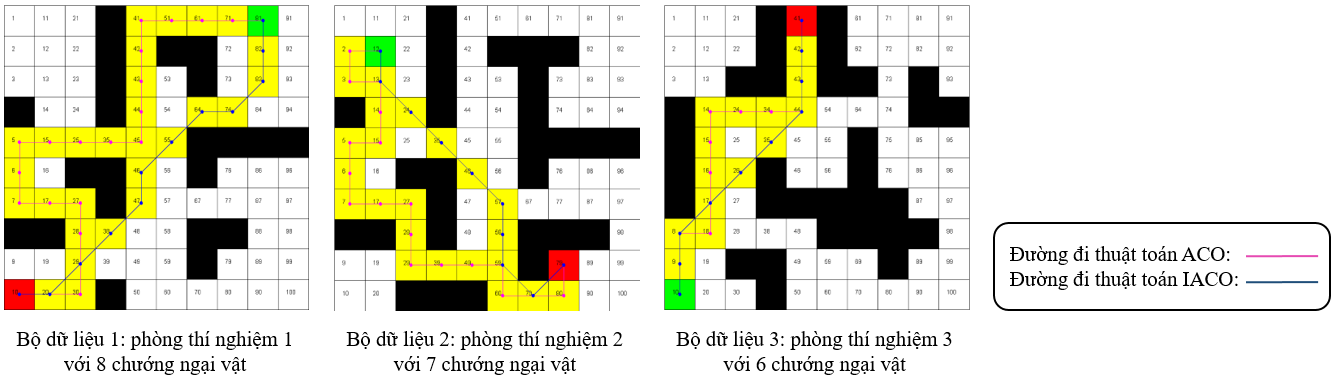
Kết quả từ *Hình 3* và *Bảng 1* cho thấy IACO có đường đi và thời gian chạy trung bình ngắn hơn so với ACO. Với những bộ dữ liệu có ít chướng ngại vậy làm tăng phạm vi tìm kiếm đường đi khiến số lượng đoạn đường cần xét tăng lên nhưng IACO vẫn nhanh hơn so với ACO.

**Bảng 1.** So sánh giải thuật ACO và IACO trong thực nghiệm 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bộ dữ liệu | Giải thuật | Ttb (ms) | Ltb tìm được | Tltb |
| Bộ dữ liệu 1 | ACO | 14.7 | 17.1 | 100% |
| **IACO** | **5.5** | **13.1** | **100%** |
| Bộ dữ liệu 2 | ACO | 13.2 | 20.3 | 100% |
| **IACO** | **5.7** | **14.1** | **100%** |
| Bộ dữ liệu 3 | ACO | 14.8 | 20 | 100% |
| **IACO** | **8.3** | **16.1** | **100%** |
| Bộ dữ liệu 4 | ACO | 15.1 | 19.2 | 100% |
| **IACO** | **7.8** | **14.5** | **100%** |
| Bộ dữ liệu 5 | ACO | 14.9 | 20 | 100% |
| **IACO** | **11.4** | **14.9** | **100%** |

**4.2. Đánh giá kết quả của giải thuật IACO với các bộ dữ liệu mô phỏng không gian trong phòng thí nghiệm**

Bao gồm 3 bộ dữ liệu mô phỏng không gian trong môi trường thí nghiệm nhằm kiểm nghiệm khả năng chạy trong thực tế của thuật toán



**Hình 4**. *Kết quả thực nghiệm 2*

Từ *Hình 4* và *Bảng 2*, ta thấy thuật toán IACO có thể chạy tốt, ổn định trong mỗi bộ dữ liệu mô phỏng một không gian trong phòng thí nghiệm. Đường đi mà giải thuật IACO tìm ra tối ưu hơn, độ dài ngắn hơn và tốn ít thời gian thực hiện hơn so với ACO.

**Bảng 2.** *So sánh giải thuật ACO và IACO trong thực nghiệm 2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bộ dữ liệu | Giải thuật | Ttb (ms) | Ltb tìm được | Tltb |
| Bộ dữ liệu 1 (chiếm 21% môi trường) | ACO | 16.6 | 21.9 | 100% |
| **IACO** | **8.8** | **13.5** | **100%** |
| Bộ dữ liệu 2 (chiếm 28% môi trường) | ACO | 13.6 | 18.3 | 100% |
| **IACO** | **10.6** | **11.5** | **100%** |
| Bộ dữ liệu 3 (chiếm 31% môi trường) | ACO | 14.1 | 16.6 | 100% |
| **IACO** | **7.3** | **10.7** | **100%** |

**5. TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | F.H.Ajeil, et al. (2020). Grid-based mobile robot path planning using aging - based ant colony optimization algorithm in static and dynamic environments. |
| [2] | Khaled Akka and Farid Khaber (2018). Mobile robot path planning using an improved ant colony optimization. |
| [3] | D.R.Parhi and J.K.Pothal (2011). Intelligent navigation of multiple mobile robots using an ant colony optimization technique in a highly cluttered environment. |