|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

**TRƯỜNG ĐẠI HỌC THỦY LỢI**



PHẠM HOÀNG HƯNG

**Lập lịch đường đi cho robot di động tự trị trong môi trường tĩnh có chướng ngại vật dựa trên phép lai tối ưu hóa bầy đàn lai với giải thuật tiến hóa sai khác vi phân**

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2022

|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

PHẠM HOÀNG HƯNG

**Lập lịch đường đi cho robot di động tự trị trong môi trường tĩnh có chướng ngại vật dựa trên phép lai tối ưu hóa bầy đàn lai với giải thuật tiến hóa sai khác vi phân**

|  |  |
| --- | --- |
| Ngành : | Công nghệ thông tin |
| Mã số : |  |

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN | ThS.Trần Thị Cẩm Giang |
|  |  |

HÀ NỘI, NĂM 2021

|  |  |
| --- | --- |
| Logo-WRU | CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  **Độc lập - Tự do - Hạnh phúc**  ----------★----------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

**Họ tên sinh viên:** Phạm Hoàng Hưng **Hệ đào tạo**: Đại học chính quy

**Lớp**: 58TH2 **Ngành**: Công nghệ thông tin

**Khoa**: Công nghệ thông tin

1- TÊN ĐỀ TÀI:

**Lập lịch đường đi robot di động tự trị trong môi trường có chướng ngại vật dựa trên phép lại tối ưu hóa bầy đàn với giải thuật tiến hóa sai khác vi phân**

2- CÁC TÀI LIỆU CƠ BẢN:

|  |  |
| --- | --- |
| [1] | J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, Australia, 1995. |
| [2] | Parsopoulos et al, "Particle swarm optimization and intelligence: advances and applications," 2010. |
| [3] | Zhang et al,"Path Planning of Mobile Robot Based on Hybrid," 2018. |
| [4] | Deb and Kalyanmoy, "Multi-objective optimization," 2014. |
| [5] | Marler *et al*, "Survey of multi-objective optimization methods for engineering," in *Structural and Multidisciplinary Optimization volume 26, pages369–395(2004)*, 2004. |
| [6] | S. Zhongke et al, "Overview of multi-objective optimization methods," in *Journal of Systems Engineering and Electronics (Volume: 15, Issue: 2, June 2004)*, 2004. |
| [7] | Elarbi et al, "Multi-objective Optimization: Classical and Evolutionary Approaches," in *Recent Advances in Evolutionary Multi-objective Optimization*, 2016. |
| [8] | K. Price et al, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," 1997. |
| [9] | Feoktistov and Vitaliy, *Differential evolution*, Springer US,2006. |
| [10] | N.Sariff and N.Buniyamin, "An overview of autonomous mobile robot path planning algorithms," 2006. |
| [11] | J.Canny, *The Complexity of Robot Motion Planning*, The MIT Press: Cambridge, MA, USA,1988. |
| [12] | Zhang et al, "Path planning for the mobile robot: A review.", Symmetry 10.10 (2018): 450., 2018. |
| [13] | Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," in *IEEE Trans. Evolut. Comput. 10 (6) (2006) 658– 674*, 2006. |
| [14] | D. Chafekar et al, "Constrained multi-objective optimization using steady state genetic algorithms," in *Genetic and Evolutionary Computation — GECCO 2003 pp 813-824*, 2003. |
| [15] | Zhang et al, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," in *Neurocomputing 103 (2013): 172-185.2013*, 2013. |

3 - NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:

|  |  |
| --- | --- |
| **Nội dung các phần** | **Tỷ lệ** |
| Chương 1: Cơ sở lý thuyết   * Robot di động * Mô hình hóa môi trường * Thuật toán tối ưu hóa bầy đàn * Giải thuật sai khác vi phân |  |
| Chương 2: Bài toán lập kế hoạch đường đi cho robot   * Tổng quan bài toán lập kế hoạch đường đi cho robot * Phát biểu bài toán * Các nghiên cứu liên quan |  |
| Chương 3: Thuật toán tối ưu hóa bầy đàn kết hợp với giải thuật tiến hóa sai khác vi phân   * Thuật toán tối ưu hóa bầy đàn kết hợp đột biến của giải thuật tiến hóa sai khác vi phân * Cập nhật vị trí tốt nhất của cả quần thể * Đột biến các cá thể không khả thi |  |
| Chương 4: Kết quả thực nghiệm |  |

4. GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN

|  |  |
| --- | --- |
| **Phần** | **Họ tên giáo viên hướng dẫn** |
| Chương 1: Cơ sở lý thuyết   * Robot di động * Mô hình hóa môi trường * Thuật toán tối ưu hóa bầy đàn * Giải thuật tiến hóa sai khác vi phân | ThS.Trần Thị Cẩm Giang |
| Chương 2: Bài toán lập kế hoạch đường đi cho robot   * Tổng quan bài toán lập kế hoạch đường đi cho robot * Phát biểu bài toán * Các nghiên cứu liên quan | ThS.Trần Thị Cẩm Giang |
| Chương 3: Thuật toán tối ưu hóa bầy đàn kết hợp với giải thuật tiến hóa sai khác vi phân   * Thuật toán tối ưu hóa bầy đàn kết hợp đột biến của giải thuật tiến hóa sai khác vi phân * Cập nhật vị trí tốt nhất của cả quần thể * Đột biến các cá thể không khả thi | ThS.Trần Thị Cẩm Giang |
| Chương 4: Kết quả thực nghiệm | ThS.Trần Thị Cẩm Giang |

5. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày 14 tháng 09 năm 2021

|  |  |
| --- | --- |
| **Trưởng Bộ môn**  *(Ký và ghi rõ Họ tên)* | **Giáo viên hướng dẫn chính**  *(Ký và ghi rõ Họ tên)* |

ThS.Trần Thị Cẩm Giang

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày. . . . .tháng. . . . .năm 2021 **Chủ tịch Hội đồng**

*(Ký và ghi rõ Họ tên)*

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày 26 tháng 12 năm 2021

**Sinh viên làm Đồ án tốt nghiệp**

*(Ký và ghi rõ Họ tên)*

Hưng

Phạm Hoàng Hưng

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  **BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP** |

**TÊN ĐỀ TÀI: LẬP LỊCH ĐƯỜNG ĐI CHO ROBOT DI ĐỘNG TỰ TRỊ TRONG MÔI TRƯỜNG tĩnh CÓ CHƯỚNG NGẠI VẬT, DỰA TRÊN PHÉP LAI TỐI ƯU HÓA BẦY ĐÀN LAI VỚI GIẢI THUẬT TIẾN HÓA sai khác vi phân**

*Sinh viên thực hiện: Phạm Hoàng Hưng*

*Lớp: 58TH2*

*Giáo viên hướng dẫn: ThS. Trần Thị Cẩm Giang*

**TÓM TẮT ĐỀ TÀI**

Hiện nay cuộc cách mạng công nghiệp lần thứ 4 đã nở rộ trên toàn thế giới, kéo theo sự phát triển của khoa học công nghệ. Xu hướng của thế giới là tiến tới một thế giới thông minh nơi mà con người gần như không cần phải làm gì, mọi thứ đã có robot lo liệu. Và hiện tại robot đang được phát triển để làm những công việc khó hơn và tỉ mỉ hơn, cần độ chính xác cao và tuyệt đối cũng như là sứ mệnh khám phá, thám hiểm vũ trụ. Chính vì làm những công việc khó như vậy nên việc thiết kế để tối ưu hóa năng lượng cho robot là những băn khoăn và trăn trở của những nhà khoa học trong suốt nhiều năm. Và bài toán lập kế hoạch đường đi cho robot là một trong các bài toán tối ưu năng lượng cho Robot được đề cập nhiều. Dù được nghiên cứu nhiều và đã có rất nhiều phương pháp tìm đường đi được ứng dụng để giải quyết, nhưng do bài toán lập kế hoạch đường đi cho Robot thuộc lớp bài toán NP-khó. Nên bài toán lập kế hoạch đường đi cho robot vẫn chưa có phương pháp giải tối ưu cho tất cả các trường hợp của môi trường.

Quy hoạch đường đi của robot di động là bài toán quan trọng nhất trong việc điều hướng tự trị (bao gồm các vấn đề: nhận thức, cục bộ hoá, lập kế hoạch đường đi và kiểm soát chuyển động). Các phương pháp quy hoạch đường đi robot hiện tại: thuật toán cổ điển (classical algorithms) và phương pháp tiếp cận heuristic (heuristic approaches). Tuy nhiên, các phương pháp cổ điển có thể mắc một số nhược điểm như: tính toán nhiều hơn trong không gian làm việc lớn, mắc kẹt ở một số điểm cực tiểu cục bộ và không phù hợp với môi trường động hoặc không chắc chắn. Chính vì thế để giải quyết được bài toán này, chúng tôi đưa ra phép lai giữa giải thuật tối ưu hóa hóa bầy đàn với giải thuật tiến hóa sai khác vi phân, để giải quyết bài toán tìm ra đường đi tối ưu thỏa mãn ba mục tiêu: độ dài đường đi, mức độ trơn tru và mức độ an toàn trên đường đi.

**CÁC MỤC TIÊU CHÍNH**

* Hiểu rõ bài toán lập kế hoạch di chuyển cho robot di động tự trị trong môi trường tĩnh có chướng ngại vật.
* Các phương pháp mô hình hóa môi trường giải quyết bài toán lập kế hoạch đường đi cho robot
* Giải thuật tối ưu bầy đàn, giải thuật tiến hóa vi phân sai khác để tìm được đường đi tối ưu dựa trên tiêu chí: độ dài đường đi
* Cài đặt các giải thuật trên ngôn ngữ lập trình Java.
* Chạy mô phỏng, đưa ra các nhận xét về kết quả thu được.

**KẾT QUẢ DỰ KIẾN**

Cài đặt thành công phép lai giữa giải thuật tối ưu hóa bầy đàn và giải thuật tiến hóa vi phân sai khác bằng ngôn ngữ lập trình JAVA để tìm đường đi tối ưu cho robot di động thỏa mãn tiêu chí tối ưu đã nói trên.

**LỜI CAM ĐOAN**

Tôi xin cam đoan đây là Đồ án tốt nghiệp của bản thân tôi. Các kết quả trong Đồ án tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

|  |  |
| --- | --- |
|  | **Tác giả ĐATN**  **Phạm Hoàng Hưng** |

**LỜI CÁM ƠN**

Đầu tiên, em xin gửi lời cám ơn chân thành đến ThS.Trần Thị Cẩm Giang. Cám ơn cô đã dẫn dắt em đến với nghiên cứu khoa học. Sự hướng dẫn, chỉ dạy tận tình của cô đã giúp em có thêm nhiều kiến thức và kỹ năng cần thiết của một người làm nghiên cứu, giúp em vững tin hơn trong lĩnh vực mà mình đã chọn. Đồ án này chính này kết quả của sự nghiêm túc học tập và nghiên cứu của em dưới sự hướng dẫn nhiệt tình và cô.

Em cũng xin bày tỏ lòng kính trọng và biết ơn tới các thầy giáo, cô giáo khoa Công nghệ thông tin, những người đã luôn tận tâm truyền đạt tri thức và những kinh nghiệm sống, làm việc quý báu cho chúng em. Kiến thức, kinh nghiệm đó sẽ là hành trang vô giá theo em suốt cuộc đời và sự nghiệp sau này.

Xin cảm ơn bạn bè, những người đã luôn bên cạnh em trong những lúc vui buồn suốt năm năm đại học. Các bạn chính là một phần động lực để tôi bước qua quãng đời sinh viên đầy ý nghĩa này. Cảm ơn những anh chị em trên lab nghiên cứu đã nhiệt tình chia sẻ kinh nghiệm nghiên cứu, giúp em ngày càng hoàn thiện, mở rộng và nâng cao kiến thức. Xin kính chúc mọi người luôn vui vẻ, mạnh khỏe và thành công trong cuộc sống!

***MỤC LỤC***

[DANH MỤC HÌNH ẢNH v](#_heading=h.gjdgxs)

[DANH MỤC BẢNG BIỂU vi](#_heading=h.vx1227)

[DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ vii](#_heading=h.30j0zll)

[CHƯƠNG 1 CƠ SỞ LÝ THUYẾT 1](#_heading=h.1fob9te)

[1.1](#_heading=h.3fwokq0) Robot di động 1

[1.1.1 Robot di động là gì ? 1](#_heading=h.1v1yuxt)

[1.1.2 Ứng dụng của Robot di động 2](#_heading=h.3znysh7)

[1.1.3 Những thách thức cần phải vượt qua 4](#_heading=h.4f1mdlm)

[1.2 Mô hình hóa môi trường 10](#_heading=h.2u6wntf)

[1.2.1 Mô hình hóa môi trường sử dụng cung tròn 10](#_heading=h.19c6y18)

[1.2.2 Tính toán hàm mục tiêu 12](#_heading=h.3tbugp1)

[1.2.2.1 Độ dài đường đi 15](#_heading=h.28h4qwu)

[1.2.2.2 Mục tiêu của bài toán 15](#_heading=h.28h4qwu)

[1.3. Thuật toán tối ưu hóa bầy đàn 16](#_heading=h.nmf14n)

[1.3.1 Giới thiệu chung 15](#_heading=h.28h4qwu)

[1.3.2 Giải thuật tối ưu hóa bầy đàn 15](#_heading=h.28h4qwu)

[1.3.3 Ảnh hưởng của các tham số trong PSO 15](#_heading=h.28h4qwu)

[1.3.4 Ưu , nhược điểm và ứng dụng của thuật toán PSO 15](#_heading=h.28h4qwu)

[1.4. Giải thuật tiến hóa vi phân sai khác 16](#_heading=h.nmf14n)

[1.4.1 Giới thiệu chung 15](#_heading=h.28h4qwu)

[1.4.2 Đột biến 15](#_heading=h.28h4qwu)

[1.4.3 Lai ghép 15](#_heading=h.28h4qwu)

[1.3.4 Chọn lọc 15](#_heading=h.28h4qwu)

[1.4.4 Biến thể của giải thuật tiến hóa sai khác vi phân 15](#_heading=h.28h4qwu)

[CHƯƠNG 2 BÀI TOÁN LẬP KẾ HOẠCH ĐƯỜNG ĐI CHO ROBOT 22](#_heading=h.4d34og8)

[2.1](#_heading=h.2s8eyo1) Tổng quan bài toán lập kế hoạch đường đi cho robot 22

[2.1.1 Giới thiệu chung 22](#_heading=h.17dp8vu)

[2.1.2 Phân loại bài toán 23](#_heading=h.3rdcrjn)

[2.1.3 Định hướng giải bài toán 24](#_heading=h.lnxbz9)

[2.2](#_heading=h.44sinio) Phát biểu bài toán: 27

[2.3 Các nghiên cứu liên quan 28](#_heading=h.2jxsxqh)

[CHƯƠNG 3 THUẬT TOÁN TỐI ƯU HÓA BẦY ĐÀN KẾT HỢP GIẢI THUẬT TIẾN HÓA VI PHÂN SAI KHÁC 31](#_heading=h.z337ya)

[3.1 Thuật toán tối ưu hóa bầy đàn kết hợp với đột biến bằngthuật toán tiến hóa vi phân sai khác 35](#_heading=h.37m2jsg)

[3.1.1 Mã hóa cá thế 35](#_heading=h.1mrcu09)

[3.1.2 Cập nhật vị trí tốt nhất của một cá thể 35](#_heading=h.1y810tw)

[3.2 Cập nhật vị trí tốt nhất của cả quần thể. 37](#_heading=h.4i7ojhp)

[3.2.1 Hai tập lưu trữ các cá thể không bị trội khả thi Na và không khả thi Nb 37](#_heading=h.2xcytpi)

[3.2.2 Cập nhật vị trí tốt nhất của quần thể 38](#_heading=h.1ci93xb)

[3.3 Đột biến các cá thể không khả thi 37](#_heading=h.4i7ojhp)

[3.4 Mô tả thuật toán 40](#_heading=h.2bn6wsx)

[CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM 42](#_heading=h.3as4poj)

[4.1 Dữ liệu thực nghiệm 42](#_heading=h.1pxezwc)

[4.2 Cài đặt thực nghiệm 42](#_heading=h.49x2ik5)

[4.3 Thông số thực nghiệm 42](#_heading=h.46r0co2)

[4.4 Kết quả thực nghiệm 44](#_heading=h.2p2csry)

[4.4.1 Ảnh hưởng của hình dạng chướng ngại vật đến đường đi của robot 49](#_heading=h.1hmsyys)

[4.4.1.1 Bộ dữ liệu 1 44](#_heading=h.2p2csry)

[4.4.1.2 Bộ dữ liệu 2 45](#_heading=h.3o7alnk)

[4.4.1.3 Bộ dữ liệu 3 46](#_heading=h.23ckvvd)

[4.4.1.4 Bộ dữ liệu 4 47](#_heading=h.ihv636)

[4.4.1.5 Bộ dữ liệu 5 48](#_heading=h.32hioqz)

[4.4.1.6 Nhận xét sự ảnh hưởng của hình dạng chướng ngại vật đến đường đi của robot 48](#_heading=h.32hioqz)

[4.4.2 Ảnh hưởng của mật độ chướng ngại vật đến đường đi của robot 49](#_heading=h.1hmsyys)

[4.4.2.1 Bộ dữ liệu 6 49](#_heading=h.1hmsyys)

[4.4.2.2 Bộ dữ liệu 7 44](#_heading=h.2p2csry)

[4.4.2.3 Bộ dữ liệu 8 45](#_heading=h.3o7alnk)

[4.4.2.3 Bộ dữ liệu 9 46](#_heading=h.23ckvvd)

[4.4.2.5 Bộ dữ liệu 10 47](#_heading=h.ihv636)

[4.4.2.6 Nhận xét sự ảnh hưởng của mật độ chướng ngại vật đến đường đi của robot 48](#_heading=h.32hioqz)

[4.4.3 Khả năng tìm đường đi của robot trong môi trường phòng thí nghiệm 49](#_heading=h.1hmsyys)

[4.4.3.1 Bộ dữ liệu room1 48](#_heading=h.32hioqz)

[4.4.3.2 Bộ dữ liệu room2 49](#_heading=h.1hmsyys)

[4.4.3.3 Bộ dữ liệu room3 49](#_heading=h.1hmsyys)

[KẾT LUẬN 51](#_heading=h.41mghml)

[TÀI LIỆU THAM KHẢO 53](#_heading=h.2grqrue)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.2 Biểu diễn tính toán đường đi sử dụng cung tròn . 5](#_heading=h.111kx3o)

[Hình 1.3 Sơ đồ giải thuật tiến hóa sai khác vi phân 13](#_heading=h.3dy6vkm)

[Hình 1.4 Hình minh họa quá trình lai ghép với vector 7 chiều (D=7) 15](#_heading=h.1t3h5sf)

[Hình 2.1 Phân loại bài toán lập kế hoạch đường đi cho robot 20](#_heading=h.26in1rg)

[Hình 2.2 Định hướng giải quyết bài toán lập kế hoạch đường đi cho robot trong môi trường tính đã biết trước 20](#_heading=h.35nkun2)

[Hình 2.3 Một số phương pháp mô hình hóa môi trường cổ điển [13] 22](#_heading=h.1ksv4uv)

[Hình 2.4 Một số phương pháp mô hình hóa môi trường phân rã ô 23](#_heading=h.2et92p0)

[Hình 3.1 Sơ đồ quy trình xác định tập hợp để cập nhật XP(t) 29](#_heading=h.tyjcwt)

[Hình 3.3 Sơ đồ thuật toán tối ưu hóa bầy đàn kết hợp giải thuật tiến hóa vi phân sai kh 32](#_heading=h.3l18frh)

[Hình 4.1 Đường đi robot trong bộ dữ liệu 1 sử dụng phương pháp cung tròn 34](#_heading=h.3whwml4)

[Hình 4.2 Đường đi robot trong bộ dữ liệu 2 sử dụng phương pháp cung tròn 35](#_heading=h.qsh70q)

[Hình 4.3 Đường đi robot trong bộ dữ liệu 3 sử dụng phương pháp cung tròn 37](#_heading=h.147n2zr)

[Hình 4.4 Đường đi robot trong bộ dữ liệu 4 sử dụng phương pháp cung tròn 38](#_heading=h.206ipza)

[Hình 4.5 Đường đi robot trong bộ dữ liệu 5 sử dụng phương pháp cung tròn 40](#_heading=h.23ckvvd)

[Hình 4.6 Đường đi robot trong bộ dữ liệu 6 sử dụng phương pháp cung tròn 41](#_heading=h.ihv636)

[Hình 4.7 Đường đi robot trong bộ dữ liệu 7 sử dụng phương pháp cung tròn 43](#_heading=h.ihv636)

[Hình 4.8 Đường đi robot trong bộ dữ liệu 8 sử dụng phương pháp cung tròn 44](#_heading=h.ihv636)

[Hình 4.9 Đường đi robot trong bộ dữ liệu 9 sử dụng phương pháp cung tròn 47](#_heading=h.ihv636)

[Hình 4.10 Đường đi robot trong bộ dữ liệu 10 sử dụng phương pháp cung tròn 49](#_heading=h.ihv636)

[Hình 4.11 Đường đi robot trong bộ dữ liệu room1 sử dụng phương pháp cung tròn 50](#_heading=h.ihv636)

[Hình 4.12 Đường đi robot trong bộ dữ liệu room2 sử dụng phương pháp cung tròn 52](#_heading=h.ihv636)

[Hình 4.13 Đường đi robot trong bộ dữ liệu room3 sử dụng phương pháp cung tròn 53](#_heading=h.ihv636)

# **DANH MỤC BẢNG BIỂU**

# [Bảng 2 Thông số cài đặt thực nghiệm sự ảnh hưởng của mật độ chướng ngại vật đến đường đi robot……………………..……….………………………………………….43](#_heading=h.4k668n3)

# **DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ**

|  |  |  |
| --- | --- | --- |
| **Viết tắt** | **Viết đầy đủ** | **Ý nghĩa** |
| PSO | Particle Swarm Optimization | Giải thuật tối ưu hóa bầy đàn |
| DE | Differential Evolution | Giải thuật tiến hóa vi phân sai khác |
| PSODE | Particle Swarm Optimization with Differential Evolution | Giải thuật tối ưu bầy đàn kết hợp với giải thuật tiến hóa vi phân sai khác |

# **CHƯƠNG 1 CƠ SỞ LÝ THUYẾT**

* 1. **Robot di động**

### ***Giới thiệu***

Trong giai đoạn hiện nay, máy tính có thể giải được rất nhiều bài toán khó mà trước kia chưa giải quyết được. Mặc dù vậy vẫn còn một số lớn các bài toán rất thú vị cần một thuật toán thích hợp để giải chúng. Trong đó, các bài toán sử dụng trí tuệ nhân tạo là các bài toán thường gặp trong các ứng dụng thực tiễn.

Trong đời sống còn có rất nhiều việc nặng nhọc và nguy hiểm mà con người không thể tự mình thực hiện được như khám phá , làm việc tại những nơi có địa hình phức tạp như đáy biển, hai cực của trái đất cũng như bên ngoài vũ trụ … . Khi đó các loại robot di động đặc biệt có khả năng di chuyển và tùy biến sẽ là nguồn lực quan trọng.

### ***Định nghĩa và ứng dụng của Robot di động***

***Định nghĩa :***

* Robot là một thiết bị có thể thực hiện những công việc một cách tự động bằng sự điều khiển của máy tính hoặc bằng những vi mạch lập trình. Robot có những đặc điểm sau đây : do con người sáng tạo ra , có khả năng nhận biết môi trường xung quanh , và tương tác với các vật thể trong môi trường, có khả năng đưa ra các lựa chọn dựa trên môi trường và được điều khiển một cách tự động theo những trình tự đã được lập trình trước, có thể điều khiển được bằng các lệnh để có thể thay đổi tùy theo yêu cầu của người sử dụng, có thể di chuyển quay hoặc tịnh tiến theo một hay nhiều chiều và khéo léo trong vận động.
* Mobile Robot là robot có khả năng tự di chuyển
* Mobile robot di chuyển trong môi trường của chúng, không cố định vào một vị trí thực. Robot di động có thể là "tự trị" (robot di động tự động) có nghĩa là chúng có khả năng điều hướng một môi trường không kiểm soát được mà không cần các thiết bị hướng dẫn vật lý hoặc cơ điện. Ngoài ra, robot di động có thể dựa vào các thiết bị hướng dẫn cho phép nó di chuyển tuyến đường định hướng được xác định trước trong không gian tương đối được kiểm soát (robot tự điều khiển). Nó khác với robot công nghiệp thường đặt gần cố định và hoạt động bằng các cánh tay.

***Ứng dụng của Mobile Robot*** :

* Mobile robot ứng dụng cho rất nhiều loại công việc khác nhau từ xây dựng đến nông nghiệp, từ đào mìn đến thăm dò dầu khí, xử lý môi trường, y tế, giải trí, vận chuyển,…
* Ngành lắp ráp tự động
* Kỹ thuật số
* Logictics

### ***Ưu , nhược điểm và những thách thức cần phải vượt qua***

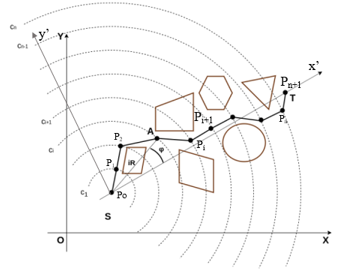
* **Ưu điểm** :
  + Mobile robot là loại robot có thể thực hiện các tác vụ ở các địa điểm khác nhau, không ở cố định một vị trí nào. Linh động là đặc tính của robot di động, có thể có được từ các bộ phận chuyển động như bánh xe, chân, tay, cánh quạt… Robot di động *“phải biết”* định vị và *“thu nhận”* được thông tin đầy đủ về môi trường xung quanh, sau đó mới có quyết định thực hiện hành động nào cho phù hợp.
  + Do đó, Mobile robot thường được tích hợp các cảm biến nhằm giúp cho chúng có thể nhận biết. Ngoài ra, robot di động còn có thể gắn kết với một hệ thống máy tính điều khiển và hệ thống cung cấp điện năng cho các chuyển động cũng như các cảm biến. Tùy vào tính chất công việc, các robot di động có thể phải mang theo nguồn điện, camera, micro, bộ cảm biến và các bộ xử lý. Tuy nhiên, do các robot di động đều có một tải trọng nhất định, nên khi thiết kế, cần tính toán trọng lượng các vật mang theo này ở mức vừa phải. Một đặc điểm quan trọng khác nữa là các robot di động cần phải có tính tự động một cách tương đối, nghĩa là phải có khả năng tự làm một hành động nào đó mà không cần có sự can thiệp của con người.
* **Nhược điểm** :
  + Khác với robot cố định, robot di động có những yêu cầu cao hơn, đòi hỏi đầu tư nhiều hơn. Trong khi robotcố định vận hành khá đơn giản, chỉ cần không gian cố định để thực các công việc lặp đi lặp lại, còn hệ thống robot di động hoạt động trong không gian mở, thay đổi liên tục và đôi khi rất phức tạp.
* **Những thách thức cần phải vượt qua** :
  + Bên cạnh các thành tựu nghiên cứu đã đạt được và nhu cầu sử dụng robot di động, còn rất nhiều những thách thức và khó khăn phải kể đến ví dụ như vấn đề thao tác của robot, làm sao robot có thể nhận biết được hành vi và đưa ra quyết định trong những tình huống bất ngờ…
  + Nói một cách đơn giản, robot cần phải trở nên khéo léo hơn rất nhiều trước khi chúng có thể sẵn sàng giúp con người thực hiện các tác vụ vật lý trong cuộc sống hàng ngày. Bước đầu tiên liên quan đến việc robot có trí thông minh để nhận thức và hiểu môi trường xung quanh. Tiếp theo, nó phải có khả năng đưa ra quyết định nhanh chóng trong những tình huống bất ngờ. Cuối cùng, robot nên khéo léo và nhanh nhẹn để thực hiện các hành động thích hợp. Tuy nhiên, 231các nhà thiết kế robot không thể lường trước mọi khả năng robot sẽ gặp trong môi trường thực tế. Do đó, robot cần có khả năng học hỏi trải nghiệm giống như con người….
  1. **Mô hình hóa môi trường**
     1. ***Mô hình hóa môi trường sử dụng cung tròn***

Để hoàn thành việc hoạch định đường đi hiệu quả, môi trường bản đồ của không gian làm việc trước tiên được chuyển đổi và lưu trữ thành biểu diễn tính toán để làm cho robot hiểu và nhìn thấy môi trường xung quanh.

Các phương pháp mô tả của bản đồ môi trường bao gồm: phân rã ô lưới, biểu đồ Maklink, biểu đồ Voronoi, phương pháp phân chia tương đương tuyến tính, ….oronoi, phương pháp phân chia tương đương tuyến tính, ….

Trong đồ án này để lập mô hình không gian làm việc cho robot ta sử dụng mô hình hóa môi trường cung tròn chia không gian làm việc của robot thành một tập các cung tròn có độ rộng bằng nhau với hai tiêu chí hiệu suất: chiều dài của đường dẫn, mức độ rủi ro của đường dẫn.

Trong hệ trục tọa độ Oxy như hình 3.1:

**

*Hình 1.1 Minh họa phương pháp mô hình hóa môi trường sử dụng cung tròn*

Các thực thể đa giác – các chướng ngại vật với số lượng là m. S và T lần lượt là điểm xuất phát và điểm đích đến của robot.

Trong đồ án phân rã môi trường bằng một tập các cung tròn 𝐶 = (𝑐1 , 𝑐2 , … , 𝑐𝑛) có tâm là điểm xuất phát 𝑆 của robot và có bán kính tương ứng lần lượt là 𝑅, 2𝑅, … , 𝑛𝑅 (𝑛𝑅 ≤ 𝑆𝑇). Một đường đi robot đi từ 𝑆 → 𝑇 ký hiệu là: 𝑃𝑎 = (𝑆, 𝑝1 , 𝑝2 , … , 𝑝𝑛, 𝑇) có thể được tạo thành bằng cách lấy ngẫu nhiên các điểm nằm trên các cung tròn thuộc tập 𝐶. Coi 𝑆 = 𝑝0 và 𝑇 = 𝑝𝑛+1 , như vậy, một đường đi của robot có thể được biểu diễn như sau: 𝑃𝑎 = (𝑝0 , 𝑝1 , 𝑝2 , … , 𝑝𝑛, 𝑝𝑛+1).

Một điểm A bất kỳ thuộc cung tròn 𝑐𝑖 có thể được xác định bởi góc 𝜑 tạo bởi  và bán kính 𝑖𝑅: 𝐴(𝜑, 𝑖𝑅). Riêng điểm 𝑝𝑛+1 tương ứng với điểm 𝑇 được biểu diễn là: (𝑆𝑇,0). Việc chuyển đổi giữa cách biểu thị điểm như trên với tọa độ thực của điểm đó trong hệ trục tọa độ 𝑂𝑥𝑦 được tính toán như sau:

xA = xs +iR *(1-1)*

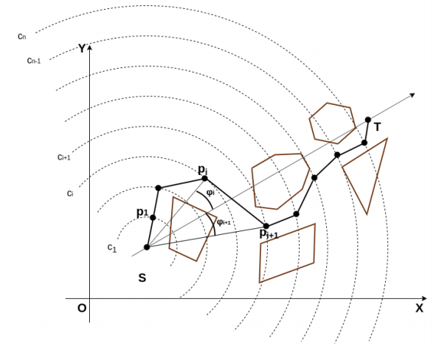
yA = ys +iR *(1-2)*

Đường đi khả thi là một đường đi an toàn cho robot di chuyển từ điểm xuất phát S đến điểm kết thúc T mà không va chạm với bất kỳ chướng ngại vật nào. Trong đồ án tính toán số lần va chạm của một đường đi với m chướng ngại vật trong môi trường để xác định một đường đi là khả thi hay không khả thi. Nếu một đoạn pipi+1 của đường đi Pa va chạm với một chướng ngại vật Ok thì giá trị va chạm: Cvik = 1, với i = 0, …, n; k = 1,…, m.

Tổng số va chạm của đường đi robot với các chướng ngại vật:

Cv(Pa) = ik  *(1-3)*

* + 1. ***Tính toán hàm mục tiêu***
       1. ***Độ dài đường đi***



*Hình 1.2 Biểu diễn tính toán độ dài đường đi sử dụng cung tròn*

Độ dài đường đi của robot Pa :

*fL*(Pa) = (𝑝0 , 𝑝𝑛+1) *(1-4)*

Độ dài đoạn pipi+1 được tính theo góc 𝜑 ,𝑖+1 và 2 bán kính 𝑖𝑅, (𝑖 + 1)𝑅:

d(pipi+1) = R *(1-5)*

Độ dài đoạn 𝑝𝑛, 𝑝𝑛+1 :

d(pnpn+1) = *(1-6)*

* + - 1. ***Mục tiêu của bài toán***

Dựa vào các hàm mục tiêu được tính toán như trên , bài toán lập kế hoạch đường đi cho robot trong đồ án có thể được phát biểu là một bài toán tối ưu với một ràng buộc như sau:

𝑚𝑖𝑛(𝑃𝑎) = (𝑓𝐿 (𝑃𝑎)) ; rà𝑛𝑔 𝑏𝑢ộ𝑐: C𝑣(𝑃𝑎) = 0

* 1. **Thuật toán tối ưu hóa bầy đàn**
     1. ***Giới thiệu chung***

Thuật toán tối ưu bầy đàn (Particle Swarm Optimization – PSO) [1] được giới thiệu vào năm 1995 tại hội nghị của IEEE bởi James Kennedy và Russell C. Tác giả lấy cảm hứng từ việc nghiên cứu hành vi xã hội của các loài động vật. Các nghiên cứu này chỉ ra rằng, một vài loài động vật sống theo bầy đàn như chim, cá, … có khả năng chia sẻ thông tin giữa các cá thể trong bầy với nhau và khả năng trao đổi như vậy tạo nên một lợi thế lớn cho loài đó trong việc sinh tồn. Tác giả chỉ ra rằng thuật toán được bắt nguồn từ khái niệm trí thông minh bầy đàn thường quan sát được ở các nhóm động vật. PSO là một thuật toán được sử dụng rộng rãi để giải quyết các vấn đề tối ưu hóa trong không gian liên tục.

Để giải thích cho việc tại sao khái niệm trí thông minh bầy đàn lại truyền cảm hứng để xây dựng nên một thuật toán tối ưu hóa nhằm giải quyết các vấn đề toán học phức tạp, một cuộc thảo luận về hành vi của một đàn chim được đưa ra. Một đàn chim bay trên không trung từ nơi này đến nơi khác cần tìm một vị trí để hạ cánh và trong trường hợp này, việc xác định điểm hạ cánh là một vấn đề phức tạp bởi việc này phụ thuộc vào một số yếu tố như tối ưu số lượng thức ăn tìm được tại vị trí đó, giảm thiểu rủi ro tồn tại động vật thiên địch với loài chim,… Ta có thể hiểu được sự di chuyển của đàn chim ở đây như một chuyển động phối hợp: ban đầu, cả đàn chim bay đến các vị trí có thể là ngẫu nhiên, nhưng sau một khoảng thời gian, đàn chim bắt đầu tìm được các vị trí tối ưu hơn trong không gian. Đó là do một cá thể trong đàn chim tìm thấy được vị trí tốt: có nhiều thức ăn, ít thiên địch,… và tùy theo độ tối ưu của vị trí, cá thể đó có thể thông báo lại cho các cá thể ở vị trí lân cận, và lan truyền trong cả đàn chim. Đàn chim dựa vào các thông tin nhận được, điều chỉnh hướng bay cũng như vận tốc hướng đến vị trí có nhiều thức ăn nhất. Cơ chế lan truyền thông tin trong một quần thể như vậy chính là khái niệm trí thông minh bầy đàn. Thuật toán tối ưu bầy đàn PSO chính là mô phỏng lại cơ chế này để tìm ra giá trị tối ưu của một hàm mục tiêu 𝐹 trong không gian 𝑛 chiều.

* + 1. ***Giải thuật tối ưu hóa bầy đàn***

Tối ưu hóa bầy đàn được phát triển để tạo ra các đường dẫn khả thi bằng cách kết hợp các đường dẫn không khả thi bị chặn bởi các chướng ngại vật với con đường khả thi thông qua chiến lược đột biến cải tiến của tiến hóa khác biệt.

PSO được khởi tạo bởi một nhóm ngẫu nhiên các điểm, sau đó tìm kiếm giải pháp tối ưu bằng việc cập nhật các vị trí.

Giải thuật PSO trong không gian tìm kiếm A ⸦ Rn nhằm tối ưu hóa hàm mục tiêu

𝑓:𝐴 → 𝑌 ⊆ 𝑅n.

Một quần thể (swarm) được định nghĩa là một tập hợp (S= x*1*, x*2*,…, x*n*) của N cá thể [2]:

xi = (xi1, xi2, xi3, …, x) i= 1, 2, …, N

Các giá trị của xij (i=1, …,N; j=1, …, N) ban đầu được gán ngẫu nhiên. Với hàm mục tiêu f(x) được cho là giá trị tại mọi điểm trong không gian tìm kiếm A. Giá trị hàm mục tiêu xác định của mỗi cá thể:

𝑓𝑖 = (𝑥𝑖 ) ∈ Y

Các cá thể có khả năng dịch chuyển trong không gian tìm kiếm A bằng cách điều chỉnh vị trí của chúng bởi một đại lượng được gọi là vận tốc:

𝑣𝑖 = (𝑣𝑖1, 𝑣𝑖2 , …, 𝑣𝑖𝑛) T , 𝑖 = 1, 2, …, N

Vận tốc cá thể được thay đổi nhiều lần trong suốt quá trình tối ưu giúp cho cá thể có khả năng di chuyển đến bất cứ vùng nào trong không gian tìm kiếm. Với t là biến đếm số vòng lặp của thuật toán, thì xi(t) và vi(t) tương ứng là vị trí và vận tốc của cá thể *i* trong vòng lặp t.

Vận tốc của các thể được cập nhật dựa trên thông tin thu được từ vòng lặp trước của thuật toán. Các thông tin đó được lưu lại trong suốt quá trình tìm kiếm giá trị tối ưu. Vị trí này được ký hiệu: Pbest(t), với t là biến đếm vòng lặp của thuật toán.

Gbest là vị trí tốt nhất của cả quần thể. Được ký hiệu là: Gbest(t), với t là biến đếm vòng lặp của thuật toán.

🡪 Với tất cả các thông tin trên, vận tốc của cá thể và vị trí của nó được cập nhật theo công thức như sau:

𝑣𝑖(𝑡 + 1) = 𝑤\*𝑣𝑖𝑗(𝑡) + 𝑐1\*R1\*(Pbest𝑖𝑗(𝑡) − 𝑥𝑖𝑗(𝑡)) + 𝑐2\*𝑅2 \*(Gbest𝑖𝑗(𝑡) - 𝑥𝑖𝑗(𝑡)) *(1-7)*

x𝑖(𝑡 + 1) = 𝑥𝑖𝑗 (𝑡) + v𝑖𝑗 (𝑡 + 1) *(1-8)*

Trong đó:

𝑖 = 1, 2, … , 𝑁; 𝑗 = 1, 2, … , 𝑛;

t: biến đếm vòng lặp;

w: trọng số quán tính;

c1, c2: hệ số gia tốc;

R1, R2: biến ngẫu nhiên trong khoảng [0,1].

Mỗi vòng lặp sau khi cá thể được cập nhật vị trí và đánh giá giá trị hàm mục tiêu, giá trị tốt nhất cá thể từng đạt được cũng được cập nhật. Giá trị vị trí tốt nhất mới của cá thể i trong vòng lặp thứ t+1 được định nghĩa :

𝑥𝑖 (𝑡 + 1), 𝑖𝑓 (𝑥𝑖 (𝑡 + 1)) ≤ 𝑓(Pbest𝑖 (𝑡)),

Pbesti (t+1) = *(1-9)*

𝑝𝑖 (𝑡), 𝑖𝑓 (𝑥𝑖 (𝑡 + 1)) > 𝑓(Pbest𝑖 (𝑡)).

Sau khi cập nhật các giá trị vị trí trong tập hợp Pbest, việc xác định chỉ số i cho vị trí có giá trị hàm mục tiêu tốt nhất sẽ hoàn thành một vòng lặp của giải thuật PSO.

Cách thức hoạt động cụ thể của thuật toán PSO được mô tả như sau:



**Thuật toán PSO:**

19



**begin**

Đặt 𝑡 ← 0;

Khởi tạo quần thể ban đầu 𝑆, các cá thể với vị trí và vận tốc ngẫu nhiên;

4. Đánh giá độ phù hợp của mỗi cá thể;

5. **while** *chưa thỏa mãn điều kiện dừng* then

6. Cập nhật các giá trị vị trí cá thể trong tập 𝑆 theo công thức

*(1-1)* và *(1-2);*

7. Đánh giá giá trị hàm mục tiêu của 𝑆;

8. Cập nhật lại giá trị cho tập P;

9. Đặt 𝑡 ← 𝑡 + 1;

10. **end while**

11. **end**





* + 1. ***Ảnh hưởng của tham số trong tối ưu hóa bầy đàn ( PSO )***

Trong thuật toán PSO, các cá thể ban đầu thường được khởi tạo một cách ngẫu nhiên theo phân phối chuẩn trong không gian tìm kiếm 𝐴. Khởi tạo ngẫu nhiên như vậy giúp thuật toán đánh giá, xử lý các vùng trong không gian tìm kiếm 𝐴 đồng đều, phù hợp để áp dụng cho các bài toán không có thông tin về hình thức của không gian tìm kiếm cũng như hàm đánh giá mục tiêu.

*-* ***Giá trị vận tốc ở vòng lặp trước của một cá thể 𝑣𝑖(𝑡)***

Giá trị vận tốc ở vòng lặp trước của một cá thể 𝑣𝑖𝑗(𝑡), nằm ở vế phải của phương trình *(1.3-1)*, đưa ra định nghĩa về chuyển động quán tính của cá thể bởi giá trị này gây ảnh hưởng đến giá trị vận tốc ở vòng lặp sau của cá thể đó 𝑣𝑖𝑗(𝑡 + 1). Đặc tính này giúp tránh việc cá thể thiên vị các vị trí tốt nhất mà nó nắm được, nhằm giảm thiểu khả năng cá thể rơi vào cục bộ địa phương do cả hai thông số về giá trị vị trí tốt nhất 𝑝*best(𝑡)* và g*best(𝑡)* đều nằm trong khu vực này.

*-* ***Giá trị trọng số quán tính***

Việc sử dụng giá trị vận tốc ở vòng lặp trước đó vào trong phương trình cập nhật vận tốc mới của cá thể cũng gây ra sự bất lợi trong việc kiểm soát vận tốc. Khi có nhu cầu tìm kiếm tập trung tại một khu vực đầy hứa hẹn thì cần tránh việc cá thể dịch chuyển ra quá xa khỏi khu vực đó. Do đó, giá trị trọng số quán tính được đưa ra, gây ảnh hưởng lên giá trị vận tốc ở vòng lặp trước đó 𝑣𝑖(𝑡) của cá thể, giúp giảm thiểu độ nhiễu khiến cho cá thể đi xa khỏi khu vực hứa hẹn xung quanh các vị trí tốt nhất. Giá trị của trọng số quán tính 𝑤 sẽ được chọn sao cho ảnh hưởng của 𝑣𝑖(𝑡) giảm dần trong quá trình thực thi của thuật toán, tức là giá trị 𝑤 sẽ được đặt giảm dần theo thời gian. Cách đặt giá trị 𝑤 quen thuộc thường được áp dụng là khởi tạo 𝑤 có giá trị xấp xỉ 1 nhằm thúc đẩy việc các cá thể khám phá không gian tìm kiếm trong giai đoạn đầu và giảm dần tuyến tính giá trị 𝑤 về 0 để loại bỏ các dao động trong giai đoạn cuối của thuật toán. Nhìn chung, cơ chế giảm dần tuyến tính của 𝑤 được biểu diễn toán học như dưới đây:

(𝑡) = 𝑤𝑚𝑎𝑥 − (𝑤𝑚𝑎𝑥 − 𝑤𝑚𝑖𝑛) *(1-10)*

Trong đó:

𝑡 là biến đếm vòng lặp;

𝑤𝑚𝑎𝑥 và 𝑤𝑚𝑖𝑛 tương ứng là giá trị cận trên và cận dưới của 𝑤;

𝑇𝑚𝑎𝑥 là tổng số vòng lặp của thuật toán

*-* ***Giá trị tham số nhận thức* 𝑐1 *và tham số nhận thức 𝑐2***

Giá trị tham số nhận thức 𝑐1 và tham số xã hội *𝑐2* có thể gây ảnh hưởng đến khả năng tìm kiếm của giải thuật PSO bằng cách hướng vị trí mới của cá thể về gần các vị trí tốt nhất 𝑝best và 𝑔best cũng như việc thay đổi cường độ của việc tìm kiếm. Trong trường hợp cần nhiều sự thăm dò trong không gian tìm kiếm thì giá trị 𝑐1 và *𝑐2* lớn có khả năng đưa ra các vị trí mới ở các khu vực tương đối xa trong không gian tìm kiếm. Mặt khác, nếu có nhu cầu tìm kiếm cục bộ xung quanh các khu vực gần vị trí tốt nhất từng đạt được của cả quần thể thì các giá trị nhỏ sẽ phù hợp để lựa chọn gán cho hai tham số trên. Ngoài ra, khi lựa chọn gán giá trị 𝑐1 , *𝑐2* , việc đặt 𝑐1 > *𝑐2* sẽ khiến cá thể có thiên hướng di chuyển nhiều hơn về hướng vị trí 𝑝𝑖 , và ngược lại, nếu 𝑐1 < *𝑐2* cá thể sẽ có thiên hướng di chuyển nhiều hơn về hướng vị trí 𝑔best. Tính chất này của hai tham số sẽ có tác dụng trong các trường hợp bài toán có các thông tin cụ thể về hình thức của hàm đánh giá mục tiêu.

*-* ***Giá trị giới hạn không gian tìm kiếm***

Trong hầu hết các bài toán tối ưu, người ta thường chỉ xem xét đến các cá thể nằm trong không gian tìm kiếm 𝐴 của bài toán. Nhằm đảm bảo việc cập nhật các cá thể trong giải thuật PSO không khiến cá thể di chuyển ra ngoài không gian tìm kiếm, giới hạn giá trị cho từng chiều của bài toán được thiết lập. Nếu một cá thể sau khi cập nhật vị trí mới như phương trình (1-2) mà vị trí mới đó nằm ngoài giá trị giới hạn được đặt ra:

𝑀 = [𝑎1 , 𝑏1 ] × [𝑎2 , 𝑏2 ] × … × [𝑎𝑛, 𝑏𝑛 ]

với 𝑎𝑖 , 𝑏𝑖 ∈ 𝑹, 𝑖 = 1, 2, … , 𝑛

Thì vị trí mới đó sẽ được cập nhật lại như sau:

𝑎𝑗, 𝑖𝑓 𝑥𝑖(𝑡 + 1) < 𝑎𝑗

𝑥𝑖(𝑡 + 1) = *(1-11)*

𝑏𝑗, 𝑖𝑓 𝑥𝑖(𝑡 + 1) > 𝑏𝑗

với 𝑖 = 1, 2, … , 𝑁; 𝑗 = 1, 2, … , 𝑛;

* + 1. ***Ưu và nhược điểm của thuật toán***

**Ưu điểm :**

Tốc độ hội tụ nhanh

Các hàm đc tối ưu hóa vi phân

Thuật toán đơn giản và dễ thực hiện thông qua lập trình

**Nhược điểm :**

Đối với hàm có nhiều cực trị cục bộ có thể bị rơi vào cực trị cục bộ và không thể nhận được kết quả chính xác ( nguyên nhân là do sự hội tụ sớm).

PSO không sử dụng đầy đủ thông tin thu được trong quy trình tính toán (nguyên nhân là do thiếu sự hợp tác của các phương pháp tìm kiếm tốt).

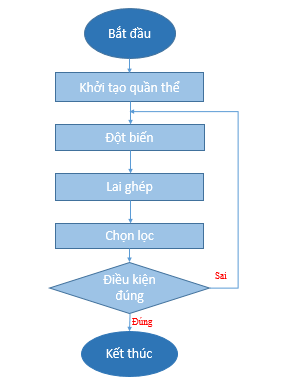
PSO cung cấp khả năng tìm kiếm toàn cầu đôi khi nó không thể đảm bảo sự hội tụ lên điều kiện toàn cầu.

Thuật toán PSO là một thuật toán tối ưu hóa sinh học meta-heuristic và không có nền tảng lý thuyết chặt chẽ cho đến nay. Được thiết kế thông qua việc đơn giản hóa và mô phỏng hiện tượng tìm kiếm của một số bầy đàn, nhưng nó không giải thích tại sao thuật toán này có hiệu quả theo nguyên tắc, cũng như không xác định phạm vi áp dụng của nó. Do đó, thuật toán PSO thường thích hợp cho một lớp các bài toán tối ưu hóa cao và không cần phải có các giải pháp quá chính xác.

* + 1. ***Ứng dụng của thuật toán***
* Ứng dụng thuật toán PSO cho bài toán tối ưu hóa trong quản lý xây dựng.
* Ứng dụng thuật toán PSO nhằm phân bố tối ưu công suất trong lưới điện phân phối.
* Ứng dụng thuật toán PSO vào hệ thống gợi ý.
  1. **Giải thuật tiến hóa vi phân sai khác**
     1. **Giới thiệu chung**

Giải thuật tiến hóa sai khác vi phân (Differential evolution - DE) được Rainer Storn và Kenneth Price giới thiệu lần đầu vào năm 1997 [8].Giải thuật DE là một giải thuật đơn giản nhưng hiệu quả được sử dụng trong các bài toán tối ưu.

Giải thuật DE thuộc lớp các thuật toán tiến hóa (Evolutionary algorithms - EAs) do cơ chế của thuật toán cũng là quần thể tiến hóa dựa trên các toán tử đột biến, lai ghép, chọn lọc để tạo ra một tập hợp các cá thể có giá trị hàm mục tiêu tối ưu.



*Hình 1.3 Sơ đồ giải thuật tiến hóa sai khác vi phân*

Giải thuật DE là một phương pháp tìm kiếm ngẫu nhiên, song song, sử dụng 𝑁𝑃 vector tham số 𝐷 chiều.

Với 𝑥𝑖,, 𝑖 = 1, 2, … , 𝑁𝑃 là một vector cá thể trong quần thể của thuật toán tại thế hệ thứ 𝐺. Số lượng cá thể 𝑁𝑃 trong quần thể không thay đổi trong quá trình tối ưu của bài toán. Các vector của quần thể khởi tạo được lựa chọn ngẫu nhiên và đảm bảo bao quát được toàn bộ không gian tìm kiếm. Thuật toán quy định sử dụng phân phối đều liên tục cho tất cả các lựa chọn ngẫu nhiên trừ những trường hợp được quy định khác. Trong trường hợp hàm tối ưu có sẵn một tập giải pháp ban đầu thì quần thể khởi tạo có thể được tạo ra bằng cách cộng thêm một độ lệch được tạo ngẫu nhiên theo phân phối chuẩn vào giải pháp ban đầu 𝑥𝑛𝑜𝑚,0 đó.

Thuật toán DE tạo ra một vector tham số mới trong thế hệ mới bằng cách cộng thêm trọng số chênh lệch giữa hai vector trong quần thể vào một vector thứ ba. Phép toán này được gọi là đột biến. Các tham số của vector bị đột biến được trộn lẫn với các tham số của một vector định trước khác – vector đích, để tạo thành một vector được gọi là vector thử nghiệm. Việc trộn lẫn các tham số thường được gọi là lai ghép trong lớp các thuật toán sử dụng chiến lược tiến hóa. Nếu vector thử nghiệm mới được tạo ra có giá trị hàm mục tiêu tốt hơn giá trị hàm mục tiêu của vector đích, thì vector thử nghiệm đó sẽ thay thế vector đích trong thế hệ kế tiếp. Việc so sánh và lựa chọn thay thế như vậy được gọi là chọn lọc. Trong mỗi thế hệ, tất cả các vector cá thể trong quần thể đều phải đóng vai trò làm vector đích để đảm bảo có 𝑁𝑃 lần lai ghép và chọn lọc được diễn ra trong một thế hệ.

* + 1. **Đột biến**

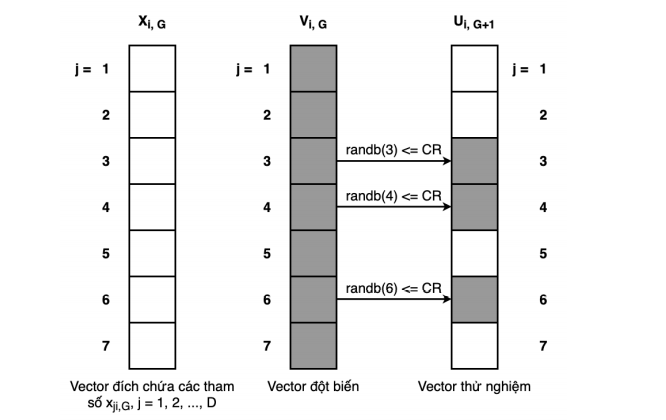
Với mỗi vector đích 𝑥𝑖,, 𝑖 = 1, 2, … , 𝑁𝑃 một vector đột biến được tạo ra như sau:

𝑣𝑖,+1 = 𝑥𝑟1,𝐺 + 𝐹 ∙ (𝑥𝑟2,𝐺 − 𝑥𝑟3,𝐺) *(1-12)*

Với r1, r2, r3 ∈ {1, 2, … , 𝑁𝑃} là các chỉ số nguyên, ngẫu nhiên và khác nhau, 𝐹 > 0.

Ngoài ra, các chỉ số nguyên 𝑟1 , 𝑟2 , 𝑟3 cũng phải lựa chọn sao cho khác với chỉ số 𝑖, do đó số lượng các vector cá thể trong quần thể NP phải đảm bảo 𝑁𝑃 ≥ 4. Hệ số 𝐹 là một số thực và là hằng số, 𝐹 ∈ [0, 2], 𝐹 có tác dụng điều chỉnh hệ số khuếch đại của sai khác vi phân (𝑥𝑟2, − 𝑥𝑟3,𝐺).

* + 1. **Lai ghép**



*Hình 1.4 Hình minh họa quá trình lai ghép với vector 7 chiều (D=7)*

Một vector thử nghiệm: 𝑢𝑖,+1 = (𝑢1𝑖,𝐺+1 , 𝑢2𝑖,𝐺+1 , … , 𝑢𝐷𝑖,𝐺+1) *(1-13)*

được tạo ra như sau:

𝑣𝑗𝑖,+1 , 𝑖𝑓 (𝑟𝑎𝑛𝑑𝑏(𝑗) ≤ 𝐶𝑅) ℎ𝑜ặ𝑐 𝑗 = 𝑟𝑛𝑏𝑟(𝑖)

𝑢𝑗𝑖,+1 =  *(1-14)*

𝑥𝑗𝑖,, 𝑖𝑓 (𝑟𝑎𝑛𝑑𝑏(𝑗) > 𝐶𝑅) ℎ𝑜ặ𝑐 𝑗 ≠ 𝑟𝑛𝑏𝑟(𝑖)

với: 𝑟𝑎𝑛𝑑𝑏(𝑗) là hệ số đánh giá thứ 𝑗 được tạo ra bởi bộ sinh số ngẫu nhiên theo phân phối chuẩn với kết quả sinh thuộc khoảng [0, 1];

𝑟𝑛𝑏(𝑖) là chỉ số được chọn ngẫu nhiên ∈ 1, 2, … ,𝐷 nhằm mục đích đảm bảo 𝑢𝑖,𝐺+1 nhận ít nhất một tham số từ 𝑣𝑖,𝐺+1 .

* + 1. **Chọn lọc**

Để xác định xem vector thử nghiệm mới tạo ra có nên trở thành cá thể trong thế hệ mới của quần thể hay không, vector thử nghiệm 𝑢𝑖,𝐺+1 được so sánh với vector đích 𝑥𝑖,𝐺 theo tiêu chí tham lam. Nếu vector 𝑢𝑖,+1 có giá trị hàm mục tiêu tốt hơn giá trị hàm mục tiêu của vector 𝑥𝑖,𝐺 thì 𝑢𝑖,𝐺+1 được chọn làm 𝑥𝑖,𝐺+1 . Nếu không, giá trị 𝑥𝑖, được giữ lại [9].

Cấu trúc cơ bản của thuật toán DE :

1. **Begin** 

2. Khởi tạo quần thể ban đầu với NP cá thể được sinh ngẫu nhiên; 

3. Đánh giá giá trị hàm mục tiêu của từng cá thể trong quần thể;

4. **while** G < GEN **then** 

5. **while** i < NP **then**

6. Chọn ngẫu nhiên 𝑟1 , 𝑟2 , 𝑟3 ∈ [1, … , 𝑁𝑃], 𝑟1 ≠ 𝑟2 ≠ 𝑟3 ≠ 𝑖;

7. Tạo ra vector thử nghiệm 𝑢𝑖,+1 dựa theo công thức *(1-6)*và *(1-8)*;

8. Đánh giá giá trị hàm mục tiêu cho 𝑢𝑖,+1 , so sánh và cập nhật giá trị cho 𝑥𝑖,𝐺+1 ;

9. Đặt i← 𝑖 + 1;

10. **end**

11. Đặ𝑡 𝑔 ← 𝑔 + 1;

12. **end**

13. **end**

* + 1. **Các biến thể của thuật toán tiến hóa vi phân sai khác (DE)**

Cấu trúc thuật toán như trên không phải là biến thể duy nhất của thuật toán DE được chứng minh là hữu ích trong các bài toán tối ưu. Để phân biệt các biến thể khác nhau của thuật toán, ký hiệu 𝐷𝐸/𝑥/𝑦/𝑧 được đưa ra:

Với:

* 𝑥 xác định vị trí của vector cá thể sẽ bị đột biến, 𝑥 hiện nay có thể là “rand” (một vector cá thể được lựa chọn ngẫu nhiên) hoặc “best” (cá thể có giá trị hàm mục tiêu tốt nhất trong cả quần thể).
* 𝑦 là số lượng vector sai khác được sử dụng, với một vector sai khác vi phân là một cặp (𝑥𝑟2,𝐺 − 𝑥𝑟3,𝐺).
* z thể hiện sơ đồ của việc lai ghép, với biến thể được trình bày như ở phần trên thì giá trị của 𝑧 là “bin” (independent binominal experiments – thí nghiệm nhị thức độc lập)
* Sử dụng ký hiệu này, thuật toán DE cơ bản được trình bày như trên được viết ra như sau: 𝐷𝐸/𝑟𝑎𝑛𝑑/1/𝑏𝑖𝑛. Ngoài ra, một biến thể của DE được sử dụng nhiều mang đến hiệu quả lớn của Price (1996) *là* 𝐷𝐸/𝑏𝑒𝑠𝑡/2/𝑏𝑖𝑛 với :

𝑣𝑖,+1 = 𝑥𝑏𝑒𝑠𝑡,𝐺 + 𝐹 ∙ (𝑥𝑟1,𝐺 + 𝑥𝑟2,𝐺 − 𝑥𝑟3,𝐺 − 𝑥𝑟4,𝐺) *(1-15)*

* Việc sử dụng 2 vector sai khác vi phân trong biến thể này có khả năng giúp nâng cao sự đa dạng của quần thể trong trường hợp số lượng cá thể trong quần thể đủ lớn.

# **CHƯƠNG 2 BÀI TOÁN LẬP KẾ HOẠCH ĐƯỜNG ĐI CHO ROBOT**

* 1. **Tổng quan bài toán lập kế hoạch đường đi cho robot**

### ***2.1.1 Giới thiệu chung***

Robot di động hiện nay đã và đang được ứng dụng rộng rãi trong rất nhiều lĩnh vực nhằm mục đích thay thế sức lao động của con người, đặc biệt là trong các ngành có môi trường làm việc nguy hiểm, độc hại như nghiên cứu hàng không vũ trụ, công nghiệp hạt nhân, khai khoáng, tìm kiếm cứu trợ thảm họa… Đối với robot di động tự trị, khả năng tìm ra một đường đi an toàn trong không gian là một yêu cầu thiết yếu cho sự thành công của một hệ thống robot. Do đó, vấn đề điều hướng cho robot di chuyển tự trị vẫn luôn là một vấn đề được rất nhiều các nhà nghiên cứu quan tâm.

Điều hướng là quá trình robot lập kế hoạch và điều khiển một tuyến đường đi của mình. Đây là nhiệm vụ đòi hỏi sự chính xác cao để giúp robot có thể di chuyển an toàn từ vị trí này đến vị trí khác mà không gặp nguy hiểm hay va chạm với chướng ngại vật [10]. Việc điều hướng có ba vấn đề cơ bản là định vị, lập kế hoạch đường đi và điều khiển chuyển động.

Trong ba vấn đề này, vấn đề lập kế hoạch đường đi giúp robot lựa chọn và xác định một tuyến đường di chuyển phù hợp trong không gian khiến cho robot có thể an toàn hoàn thành nhiệm vụ đề ra, do đó có thể khẳng định, đây là vấn đề quan trọng nhất của cả quá trình điều hướng. Một kế hoạch đường đi tốt phải có khả năng tìm ra một đường đi tối ưu về nhiều mặt như người dùng mong muốn (độ dài đường đi, thời gian di chuyển,…), giúp robot di chuyển từ vị trí bắt đầu đến vị trí kết thúc mà không va chạm với chướng ngại vật trong không gian gây nguy hiểm cho hoạt động của robot. Ngoài ra, mở rộng vấn đề đối với các môi trường động, chức năng lập kế hoạch đường đi của robot còn phải giải quyết, cảm biến được các thay đổi của môi trường, giảm thiểu tác động của các vật tác động lên robot, tìm ra đường đi tối ưu trong thời gian phù hợp để đáp ứng thường xuyên cho robot. Với những khó khăn trong việc tìm ra đường đi tối ưu chính xác, bài toán lập kế hoạch đường đi cho robot đã được xác định thuộc lớp bài toán tối ưu khó (NP-hard) [11] .

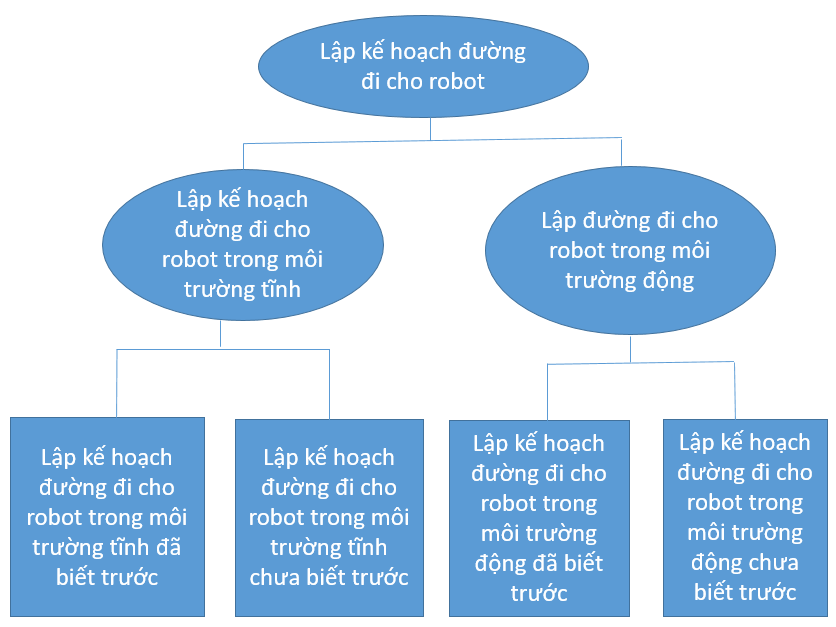
### ***2.1.2 Phân loại bài toán***

Dựa vào môi trường mà robot đang hoạt động, bài toán lập kế hoạch đường đi cho robot được chia thành hai loại:

* Lập kế hoạch đường đi cho robot trong môi trường tĩnh.
* Lập kế hoạch đường đi cho robot trong môi trường động.

Hai loại bài toán như trên còn có thể phân loại nhỏ hơn dựa trên thông tin robot biết được về toàn bộ môi trường di chuyển:

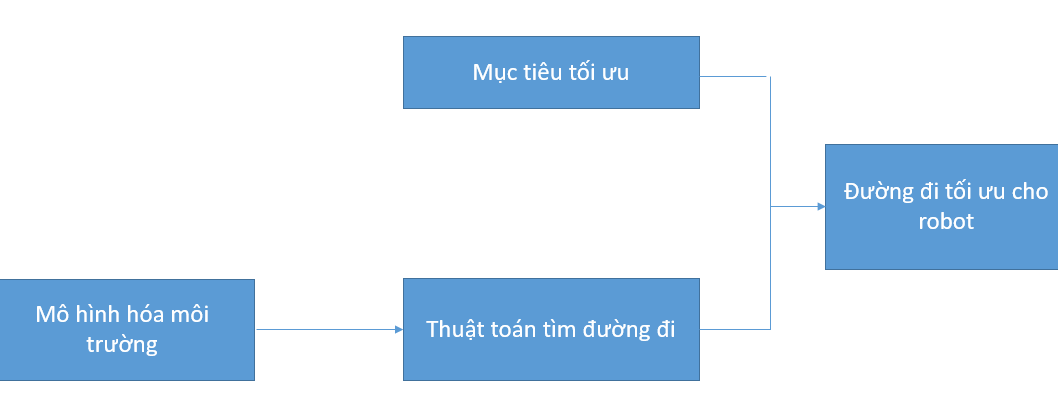
* Lập kế hoạch đường đi cho robot trong môi trường đã biết trước (robot đã biết vị trí của các chướng ngại vật trước khi di chuyển). Đường đi của robot có thể là đường đi tối ưu toàn cục do toàn bộ thông tin đã được biết trước đó.
* Lập kế hoạch đường đi cho robot trong môi trường chưa biết trước( robot chỉ nằm được một phần thông tin của môi trường, hoặc chỉ nắm được các thông tin không chắc chắn). Trường hợp này yêu cầu robot phải trang thiết bị cảm biến để thu thập các thông tin cục bộ về vị trí, hình dạng, kích thước của chướng ngại vật, từ đó sử dụng các thông tin thu thập được để đưa ra đường đi tối ưu.



*Hình 2.1 Phân loại bài toán lập kế hoạch đường đi cho robot [12]*

### **2.1.3 Định hướng giải bài toán**

Định hướng chung để giải quyết một bài toán lập kế hoạch đường đi cho robot trong môi trường đã biết trước thường được thực hiện theo quy trình sau:



*Hình 2.2 Định hướng giải quyết bài toán lập kế hoạch đường đi cho robot trong môi trường tĩnh đã biết trước [12]*

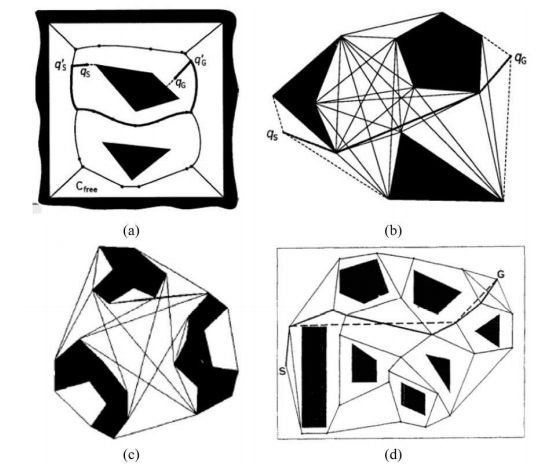
Để giải quyết bài toán lập kế hoạch đường đi cho robot trong môi trường đã biết trước bao gồm 3 phần:

Lựa chọn phương pháp mô hình hóa môi trường, lựa chọn thuật toán tìm đường, nhận thức tính toán các mục tiêu tối ưu của bài toán và từ đó đưa ra được đường đi tối ưu. Mô hình hóa môi trường là việc dựa trên các thông tin đã biết trước về môi trường làm việc của robot trong không gian thực tế để xây dựng lên một bản đồ, hay một đồ thị di chuyển cho robot.

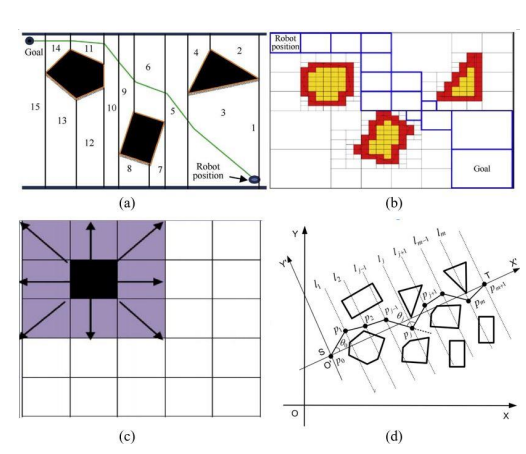
Phương pháp mô hình hóa môi trường cần phải phù hợp với thuật toán ứng dụng để giải quyết bài toán do hai đường đi tối ưu hai thành phần này có liên quan mật thiết đến nhau và ảnh hưởng rất lớn đến kết quả đường đi tối ưu tìm được. Ban đầu, khi các thuật toán xấp xỉ chưa được ứng dụng nhiều vào bài toán lập kế hoạch đường đi, các phương pháp mô hình hóa hay được sử dụng là sơ đồ Voronoi( hình 2.3.a), đồ thị tầm nhìn (đồ thị tầm nhìn (Visibility graph – hình 2.3.b), đồ thị Tangent (hình 2.3.c), đồ thị MAKLINK (hình 2.3.d),… hay các đồ thị đỉnh khác được xây dựng dựa trên tọa độ các đỉnh của chướng ngại vật, cũng như tọa độ của điểm xuất phát và điểm mục tiêu. Từ các môi trường được mô hình hóa như trên, người ta ứng dụng các giải thuật tìm đường trong đồ thị để tìm ra đường đi cho robot.

Tuy nhiên, với các phương pháp này, các thuật toán áp dụng vào thường chỉ tìm được đường đi chứ không tìm được đường đi tối ưu nhất về mặt độ dài. Ngoài ra, việc sử dụng các đỉnh của chướng ngại vật để tạo nên đồ thị cũng khiến cho độ an toàn của đường đi đối với robot là thấp, bởi trong thực tế, robot là một vật có kích thước tương đối chứ không chỉ là một chấm như trên đồ thị, việc đi quá gần các đỉnh sẽ tăng khả năng robot bị va chạm với chướng ngại vật. Do đó, các phương pháp mô hình hóa môi trường trên không được sử dụng nhiều nữa. Thay vào đó, người ta thường sử dụng các mô hình hóa chia không gian làm việc của robot thành một số lượng ô cơ bản (cell), phương pháp này được gọi là phương pháp mô hình hóa môi trường phân rã ô (cell decomposition method). Các phương pháp phân rã ô thường gặp: phân rã ô chính xác (hình 2.4.a), phân rã ô thích ứng (hình 2.4.b), phân rã ô xấp xỉ (hình 2.4.c), xoay trục tọa độ phân rã ô trục dọc (hình 2.4.d),… Ứng dụng vào việc mô hình hóa môi trường phân rã theo ô như trên các giải thuật xấp xỉ giúp các nghiên cứu có thể tìm ra được các đường đi có độ dài tối ưu hơn nhiều so với các phương pháp cổ điển.

Ngoài ra còn có thể tối ưu thêm nhiều mục tiêu khác giúp bài toán đáp ứng được tốt hơn các yêu cầu của robot trong thực tế.



*Hình 2.3 Một số phương pháp mô hình hóa môi trường cổ điển [12]*



*Hình 2.4 Một số phương pháp mô hình hóa môi trường phân rã ô*

Ngoài hai thành phần mô hình hóa môi trường và thuật toán nêu trên, các mục tiêu tối ưu bài toán yêu cầu cũng có ảnh hưởng đến kết quả của bài toán tìm ra đường đi tối ưu. Có ba mục tiêu thường hay được các nhà nghiên cứu về bài toán lập kế hoạch đường đi cho robot xét đến: độ dài , độ trơn tru và độ an toàn của đường đi.

* 1. **Phát biểu bài toán:**

Bài toán lập kế hoạch đường đi cho robot trong môi trường tĩnh đã biết trước được phát biểu như sau:

Trong một môi trường 2D cho robot di chuyển, có m chướng ngại vật (O = (O1, O2,…., Om)), tọa độ vị trí các chướng ngại vật đã biết trước và không thay đổi trong cả quá trình xử lý bài toán. Robot được xem như là một điểm. Vị trí bắt đầu và vị trí mục tiêu của robot được ký hiệu lần lượt là 𝑆 và 𝑇. Mục tiêu của bài toán là tìm ra một hay nhiều đường đi 𝑃𝑎 = (𝑝0 , 𝑝1 , … , 𝑝𝑛, 𝑝𝑛+1), (𝑆 = 𝑝0 , 𝑇 = 𝑝𝑛+1) sao cho:

* Đường đi Pa không va chạm với chướng ngại vật trong môi trường.
* Tối ưu mục tiêu:

Độ dài đường đi Pa:

*fL*(Pa) = (𝑝0 , 𝑝𝑛+1) *(2-1)*

## **2.3 Các nghiên cứu liên quan**

Lập kế hoạch đường đi cho robot cổ điển có thể dễ mắc bẫy một số cực tiểu cục bộ và cần nhiều thời gian tính toán hơn trong không gian làm việc lớn. Các thuật toán heuristic đã được đề xuất để khắc phục nhược điểm của phương pháp cổ điển trong lĩnh vực robot di động.

Mạng nơron nhân tạo đã được sử dụng để nhận ra đường dẫn lập kế hoạch cho robot di động trong các môi trường khác nhau. Trong Q-learning và lập kế hoạch mạng nơron được sử dụng để giải quyết việc tránh chướng ngại vật và lập kế hoạch quỹ đạo cho thiết bị robot di động. Robot và chướng ngại vật trong môi trường trước hết là được mô hình hóa bằng ngôn ngữ mô hình thực tế ảo, va chạm sau khi phát hiện sẽ tạo quỹ đạo được thực hiện bởi mạng nơron Pos-net, có đầu vào và đầu ra là vị trí hiện tại của robot, thời gian lấy mẫu và ma trận Q-value và ba vector phần tử chứa vị trí của robot và thời gian lấy mẫu riêng biệt.

Thuật toán di truyền là một tối ưu hóa heuristic dựa trên các hoạt động chọn lọc tự nhiên, trao đổi chéo và đột biến được áp dụng đến miền lập kế hoạch đường dẫn robot. Một gen được sửa đổi thuật toán với thước đo đa dạng được trình bày để tối ưu hóa đường dẫn dựa trên đường cong Bezier. Người ta bố trí hai máy quay Kinect trên nóc không gian di chuyển của robot nhằm mục đích mô hình hóa lại môi trường của robot và sau đó sử dụng thuật toán di truyền với lai ghép ngẫu nhiên và tám cách đột biến khác nhau cho đường đi của robot để tìm ra đường đi ngắn nhất với ít sai số và độ lệch nhất [14].

Tối ưu hóa đàn kiến là một loại thuật toán tiến hóa tự nhiên toàn cầu lấy cảm hứng từ các hành vi kiếm ăn của đàn kiến. Liu và cộng sự đã đề xuất một kế hoạch đường đi toàn cầu của một robot di động trong mô hình môi trường lưới dựa trên thuật toán đàn kiến kết hợp với trường tiềm năng nhân tạo và tối ưu hóa hình học cục bộ. Các lực trường thế từ các chướng ngại vật và điểm mục tiêu được dùng để điều hướng cho đường dẫn khuếch tán pheromone. Sau đó, các đường dẫn được tạo ra sẽ được tinh chỉnh và tối ưu lại bằng hình học cục bộ để loại bỏ các đường chéo nhau, đường dẫn tròn và đường răng cưa. Một công thức heuristic mới được đề xuất để cập nhật xác suất lựa chọn của vị trí tiếp theo dựa trên khoảng cách trọng số. Và bản cập nhật pheromone cũng được đổi mới.

Thuật toán tối ưu hóa bầy đàn (PSO) là một thuật toán thuật toán phổ biến đã được áp dụng cho nhiều ứng dụng khác nhau, đặc biệt là trong lĩnh vực robot di động. Vấn đề đầu tiên: tác giả tập trung đến việc tìm đường đi cho robot trong môi trường có nhiều nguồn nguy hiểm và sử dụng thuật toán PSO để tối ưu hai mục tiêu cho đường đi đó. Sau khi mô hình hóa môi trường bằng cách xoay trục tọa độ phân rã ô trục dọc chia môi trường bằng nhiều đoạn bằng nhau, tác giả xây dựng hai công thức đánh giá độ dài và mức độ rủi ro cho đường đi nhằm mục tiêu tối ưu hai tiêu chí này. Sau đó, tác giả đưa ra một giải thuật PSO có kết hợp thêm việc lưu trữ riêng hai tập đường đi khả thi và đường đi không khả thi, cũng như việc dùng phương pháp lấy mẫu ngẫu nhiên để chọn ra giá trị tối ưu nhất của cả quần thể nhằm cải thiện độ đa dạng của quần thể và cân bằng giữa việc khai thác và khám phá cho các cá thể trong quần thể. Tiếp đến tác giả cải tiến giải thuật tốt hơn so với nghiên cứu trên đó là: thuật toán PSO tối ưu mục tiêu về độ dài.

Tuy nhiên hầu hết các phương pháp nêu trên lập kế hoạch đường đi cho robot được xây dựng dưới dạng một bài toán tối ưu hóa mục tiêu duy nhất hoặc chuyển đổi đa mục tiêu thành một mục tiêu bằng phương pháp cộng trọng số. Về cơ bản là một phương pháp tối ưu hóa mục tiêu duy nhất và có một số vấn đề không thể tránh khỏi, cụ thể là do thiếu phát hiện các vùng không lồi trong biên Pareto. Các nghiên cứu sau đã giúp giải quyết được các nhược điểm này bằng việc đưa ra thuật toán tối ưu bầy đàn đa mục tiêu có cải tiến và kết hợp cả thêm việc đột biến bằng thuật toán tiến hóa sai khác vi phân. Bên cạnh đó phương pháp phương pháp mô hình hóa môi trường sử dụng trong hai nghiên cứu này có một nhược điểm rất lớn là không bao quát được hết toàn bộ không gian làm việc của robot. Việc robot chỉ có thể di chuyển theo một chiều từ điểm xuất phát đến điểm mục tiêu, trong không gian bị giới hạn bởi hai đường đi qua hai điểm xuất phát và điểm mục tiêu và vuông góc với đường nối hai điểm đó khiến không gian tìm kiếm bị thu hẹp lại và có khả năng làm ảnh hưởng đến độ hội tụ của thuật toán. Ngoài ra, cách mô hình hóa môi trường bị phụ thuộc vào hai điểm sẽ gây khó khăn trong việc phát triển thuật toán để mở rộng hơn cho các bài toán tìm đường đi cho robot, mong muốn đi đến nhiều mục tiêu khác nhau. Chính vì thế, đồ án đề xuất một phương pháp mô hình hóa môi trường mới vẫn phù hợp với thuật toán PSO tối ưu mục tiêu về độ dài nhưng có khả năng bao quát không gian làm việc của robot tốt hơn để giải quyết vấn đề này.

# **CHƯƠNG 3 THUẬT TOÁN TỐI ƯU HÓA BẦY ĐÀN KẾT HỢP GIẢI THUẬT TIẾN HÓA VI PHÂN SAI KHÁC**

## **3.1 Thuật toán tối ưu hóa bầy đàn kết hợp đột biến bằng thuật toán tiến hóa vi phân sai khác**

Đồ án kết hợp giữa tối ưu hóa bầy đàn lai với giải thuật tiến hóa vi phân sai khác ( Particle Swarm Optimization with Differential Evolution ) để giải bài toán lập kế hoạch đường đi cho robot với hàm mục tiêu được đưa ra ở phần trên

### ***3.1.1 Mã hóa cá thể***

### Một đường đi robot được ký hiệu: Pa = (po, p1, p2, …, pn, pn+1) như đã trình bày trên với pi(𝜑𝑖 ,𝑖R), các giá trị tọa độ các điểm pi trong đường đi robot được xác định duy nhất bởi các giá trị góc 𝜑𝑖 do các giá trị iR đã được cố định trước đó dựa trên dữ liệu về kích thước môi trường và kích thước các chướng ngại vật trong đó. Một cá thể trong thuật toán sẽ được mã hóa bởi một vector n chiều:

𝑋𝑖 = (𝜑1 ,2 , … ,𝜑𝑛 )

### ***3.1.2 Cập nhật vị trí tốt nhất của một cá thể***

Đồ án tập trung nghiên cứu bài toán tối ưu hóa mục tiêu có ràng buộc, nên cần định nghĩa lại việc so sánh trội giữa hai giải pháp bất kỳ của bài toán.

Ký hiệu Xi(t) và XPi(t) là vị trí và vị trí tốt nhất trong lần lặp t, đồ án xét so sánh trội giữa Xi(t+1) và XPi(t) để xác định XPi(t+1).

Nếu cả Xi(t+1) và XPi(t) đều là đường đi khả thi tức là:

Cv(X*i* (t +1)) = 0

Cv (XP*i* (t)) = 0,

Đồ án thực hiện so sánh trội cho hai vị trí với mục tiêu: độ dài của một đường dẫn, được sử dụng để xây dựng sự thống trị Pareto, cập nhật vị trí riêng tốt nhất của robot.

* X*i*(t +1) được chọn làm vị trí tối ưu cục bộ mới của robot nếu:

*f* ( (X*i* (t + 1))≺F *f* (XP*i* (t));

với ≺F là ký hiệu so sánh trội hơn cho các đường đi khả thi.

* Nếu X*i*(t +1) và XP*i* (t) không chi phối lẫn nhau, vị trí tối ưu cục bộ mới của cá thể được chọn ngẫu nhiên từ X*i*(t +1) và XP*i* (t).

Cả hai giải pháp không khả thi, nghĩa là

Cv (Xi (t +1)) ≠ 0

Cv (XPi (t)) ≠ 0,

Đồ án thực hiện so sánh trội cho hai vị trí với hai mục tiêu: độ dài đường đi và số lần đường đi va chạm với chướng ngại vật được sử dụng để xây dựng sự thống trị Pareto hai mục tiêu để cập nhật vị trí riêng tốt nhất của robot.

* Xi (t +1) được chọn là vị trí tối ưu cục bộ nếu *f*(Xi (t + 1)) ≺IF *f* (XPi (t)),
* Xi (t +1) và XPi (t) không thống trị lẫn nhau, vị trí tối ưu cục bộ mới được chọn ngẫu nhiên từ Xi (t +1) và XPi (t).

Nếu 1 giải pháp là khả thi, giải pháp còn lại là không khả thi nghĩa là

( Xi(t +1)) = 0

Cv(XPi(t)) ≠ 0,

* Xi(t + 1) được sử dụng là vị trí riêng tốt nhất của robot. Nếu không thì, giữ vị trí riêng tốt nhất không thay đổi.

## **3.2 Cập nhật vị trí tốt nhất của cả quần thể.**

### ***3.2.1 Hai tập lưu trữ các cá thể không bị trội khả thi Na và không khả thi N’a***

***3.2.1.1 Khái quát về hai tập lưu trữ cá thể không bị trội***

Giải thuật PSO với mỗi vòng lặp của thuật toán một cá thể được cập nhật vị trí bằng cách di chuyển hướng về phía vị trí tốt nhất của bản thân đã tìm được và vị trí tốt nhất của cả quần thể. Đối với bài toán này mục tiêu là độ dài đường đi thì vị trí tốt nhất của cả quần thể là một tập hợp Pareto bởi vậy giải thuật PSO giải quyết vấn đề lựa chọn vị trí tốt nhất của cả một quần thể bằng cách lưu lại một tập hợp riêng chứa hữu hạn các giải pháp không bị trội. Ngoài ra, việc lưu lại tập hợp riêng như vậy còn giúp hỗ trợ cho việc thêm toán tử đột biến cho thuật toán, cải thiện khả năng khám phá của thuật toán và giảm thiểu việc giải thuật mắc kẹt tại tối ưu cục bộ.

Tập hợp Na lưu các cá thể không bị trội tức là khả thi, tập N’a lưu các cá thể không khả thi. Số lượng cá thể trong Na và N’a là hữu hạn không vượt quá một số cho trước. Việc lưu trữ và sử dụng cả tập hợp các cá thể không bị trội không khả thi N’a trong việc cập nhật vị trí tốt nhất của cá thể đã được chứng minh rằng có thể giúp tạo thành cầu nối để giải thuật khám phá các vùng khả thi bị cô lập trong không gian tìm kiếm [13], [14]

* + - 1. ***Cập nhật hai tập lưu trữ không bị trội***

Khi đã cập nhật vị trí tốt nhất của từng cá thể trong vòng lặp t thuật toán sẽ so sánh XPi(t) với các cá thể lưu trữ trong Na (N’a):

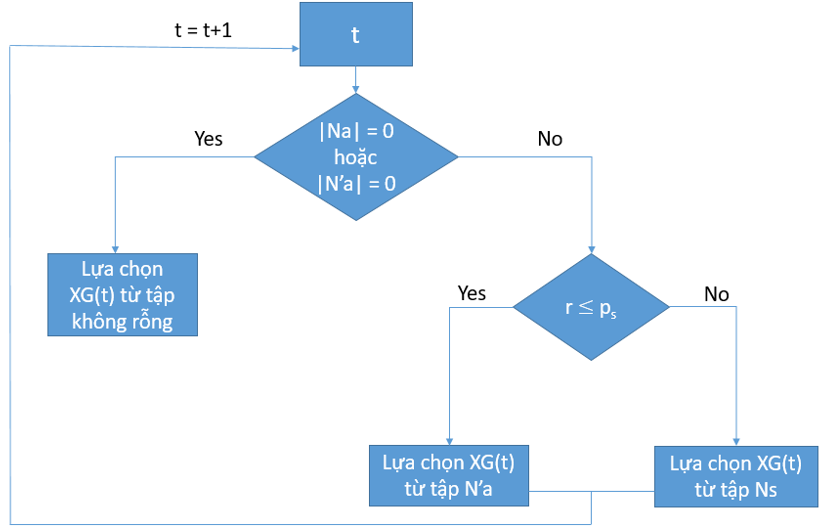
* Nếu tập 𝑁𝑎 (𝑁*’a*) rỗng, 𝑋𝑃(𝑡) được thêm trực tiếp vào tập hợp.
* Nếu 𝑋𝑃(𝑡) bị trội bởi một cá thể nào đó trong 𝑁𝑎 (𝑁*’a*), loại bỏ.
* Nếu 𝑋𝑃(𝑡) không bị trội bởi các cá thể trong 𝑁𝑎 (𝑁*’a*), thêm 𝑋𝑃𝑖(𝑡) vào 𝑁𝑎 (𝑁*’a*).
* Nếu các cá thể trong 𝑁𝑎 (𝑁’*a*) bị trội bởi 𝑋𝑃(𝑡), loại bỏ các cá thể đó.
* Nếu sau khi thêm 𝑋𝑃𝑖(𝑡) vào 𝑁𝑎 (𝑁’a) mà số lượng cá thể trong 𝑁𝑎 (𝑁’*a*) lớn hơn quy định, so sánh khoảng cách quy tụ (Crowding distance) của các cá thể trong 𝑁𝑎 (𝑁’*a*) để loại bỏ cá thể dư thừa.

### ***Cập nhật vị trí tốt nhất của quần thể***

Với XG(t) là vị trí tốt nhất của cả quần thể trong vòng lặp t, việc lựa chọn cập nhật XG(t) từ tập Na và Nb được cân bằng dựa trên một xác suất 𝑝𝑠 để đảm bảo cân bằng trong việc khám phá và khai thác không gian tìm kiếm. Với 𝑡 là vòng lặp hiện tại và 𝑇𝑚𝑎𝑥 là số vòng lặp của cả thuật toán, xác suất 𝑝𝑠 được xác định như sau:

ps = 0.5 x (1- ) *(3-1)*

Khi xác định được tập hợp để lựa chọn XG(t) dựa trên xác suất ps và một số ngẫu nhiên r (𝑟 ∈ [0,1]), thuật toán chọn ra XG(t) bằng cách lấy ngẫu nhiên 2 cá thể ra từ tập hợp để so sánh, cá thể có khoảng cách quy tụ lớn hơn sẽ được chọn làm XG(t).



*Hình 3.1 Sơ đồ quy trình xác định tập hợp để cập nhật XP(t)*

Trên sơ đồ có thể dễ dàng nhận thấy ở các vòng lặp ban đầu của giải thuật, vị trí tốt nhất của quần thể có thể được lựa chọn từ tập N’a với xác suất cao hơn, giúp làm tăng độ đa dạng của quần thể, tăng khả năng cá thể tìm kiếm ở các vùng khả thi chưa biết tới. Nhưng khi dần về đến các vòng lặp cuối, giá trị xác suất 𝑝𝑠 ngày càng giảm dần, cá thể khai thác các vị trí tốt đã biết trong quần thể nhiều hơn, và cũng giúp đảm bảo giá trị cuối cùng thuật toán trả về là một đường đi khả thi.

### **Đột biến các cá thể không khả thi dựa trên tiến hóa vi phân sai khác ( DE )**

Một cá thể không khả thi Xi, việc đột biến bằng DE có thể giúp cải thiện cá thể đó và tăng độ đa dạng của thuật toán. Xác suất đột biến 𝑝(𝑖) của cá thể 𝑋𝑖 được tính toán dựa trên độ va chạm của cá thể với môi trường, đảm bảo các cá thể va chạm nhiều có tỉ lệ đột biến cao hơn:

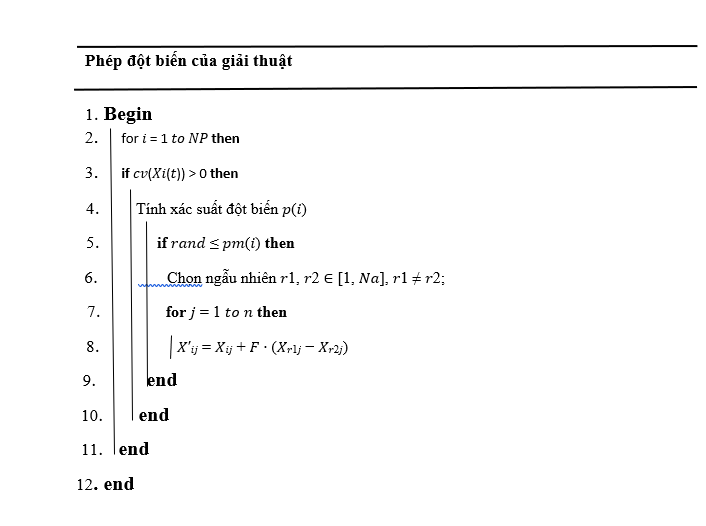
pm(i) = *(3-2)*

Nếu một số ngẫu nhiên 𝑟𝑎𝑛𝑑 ∊ [0,1], 𝑟𝑎𝑛𝑑 ≤ 𝑝𝑚(𝑖), thì ta lấy ngẫu nhiên 2 đường đi khả thi từ tập 𝑁𝑎: 𝑋𝑟1 và 𝑋𝑟2 xem như là 2 vector sai khác vi phân của thuật toán DE, và đường đi mới 𝑋′𝑖 sẽ được tạo ra bằng cách kết hợp 2 vector vi phân với đường đi không khả thi cũ 𝑋𝑖 .

Công thức cập nhật 𝐷𝐸/𝑟𝑎𝑛𝑑/1 được biểu diễn như dưới đây:

𝑋′𝑖 = 𝑋𝑖 + 𝐹 ∙ (𝑋𝑟1 − 𝑋𝑟2) *(3-3)*

với F là hệ số sai khác



## **Mô tả thuật toán**

Thuật toán để giải bài toán lập kế hoạch đường đi cho robot với phương pháp mô hình hóa môi trường sử dụng cung tròn được mô tả cụ thể theo sơ đồ trong hình 3.5.

Giải thuật được bắt đầu sau khi các thông số của thuật toán được thiết lập: bán kính 𝑅, số lượng cung tròn trong tập 𝐶: 𝑛 = 𝑟𝑜𝑢𝑛𝑑𝑑𝑜𝑤𝑛 ( ), số vòng lặp 𝑇𝑚𝑎𝑥, số cá thể trong quần thể 𝑁𝑃, kích thước của hai tập 𝑁𝑎 và 𝑁′𝑎.

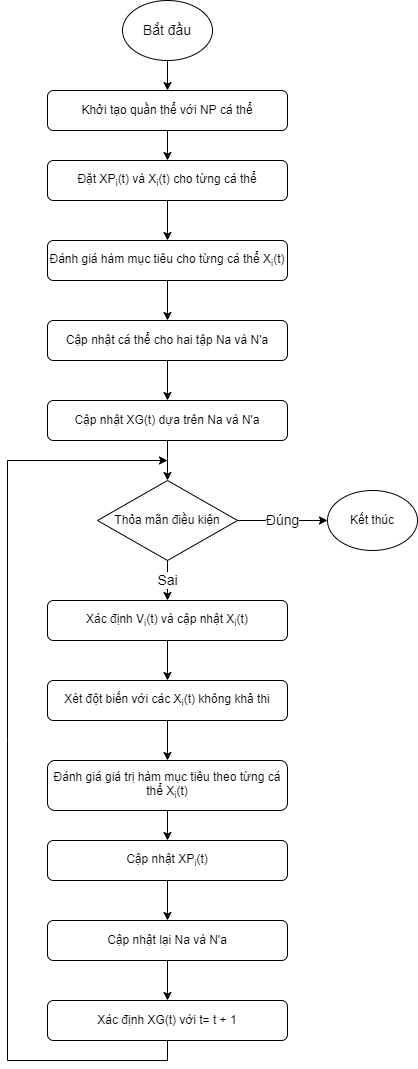
Ngoài ra còn có các thông số liên quan đến thuật toán PSO: 𝑤𝑀𝑎𝑥, 𝑤𝑀𝑖𝑛, 𝑟1 , 𝑟2 , 𝑐1 , 𝑐2 , 𝑉𝑚𝑎𝑥, 𝑉𝑚𝑖𝑛, 𝑋𝑚𝑎𝑥, 𝑋𝑚𝑖𝑛 …

Việc cập nhật vị trí của từng cá thể 𝑋(𝑡) được thực hiện theo công thức như dưới đây:

𝑉𝑖(𝑡 + 1) = 𝑤𝑉𝑖𝑗(𝑡) + 𝑐1𝑟1 (𝑋𝑃𝑖𝑗(𝑡) − 𝑋𝑖𝑗(𝑡)) + 𝑐2𝑟2 (𝑋𝐺𝑖𝑗(𝑡) − 𝑋𝑖𝑗(𝑡)) *(3-4)*

𝑋𝑖(𝑡 + 1) = 𝑋𝑖𝑗(𝑡) + 𝑉𝑖𝑗(𝑡 + 1) *(3-5)*

với 𝑗 = 1 … n



*Hình 3.2 Sơ đồ thuật toán tối ưu hóa bầy đàn kếp hợp giải thuật tiến hóa vi phân sai khác*

# **CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM**

## **4.1 Dữ liệu thực nghiệm**

Dữ liệu được sử dụng trong quá trình thực nghiệm có đặc điểm chung sau đây :

* Sử dụng môi trường có kích thước ( 500, 150 , 15)
* Lưu thông tin về tọa độ của các chướng ngại vật có trong môi trường
* Lưu thông tin về điểm xuất phát và điểm mục tiêu của robot

Quá trình thực nghiệm được chạy trên máy tính HP 14-bs 1xx có thông số :

* Bộ vi xử lý : Intel(R) Core(TM)i5 – 8250U
* Hệ điều hành Windows 10 Enterprise
* Dung lượng RAM: 8GB

## **4.2 Cài đặt thực nghiệm**

Để thực hiện tìm đường đi của robot dựa trên thuật toán tối ưu hóa bầy đàn lai giải thuật tiến hóa vi phân sai khác trong môi trường tĩnh có chướng ngại vật , ta thực hiện ba thực nghiệm sau :

* Thực nghiệm 1 : Sự ảnh hưởng của hình dạng chướng ngại vật lên đường đi của robot ( 5 bộ dữ liệu )
* Thực nghiệm 2 : Sự ảnh hưởng của mật độ chướng ngại vật lên đường đi của robot ( 5 bộ dữ liệu )
* Thực nghiệm 3 : Khả năng tìm đường đi của robot trong môi trường phòng thí nghiệm kín ( 3 bộ dữ liệu )

## **4.3 Kết quả thực nghiệm**

Sau đây là kết quả chạy thực nghiệm trên 10 bộ dữ liệu khác nhau có mật độ chướng ngại vật tăng dần từ 1 đến 10 và 3 bộ dữ liệu chướng ngại vật room1, room2, room3

Kết quả chạy có sự xuất hiện của 2 đường :

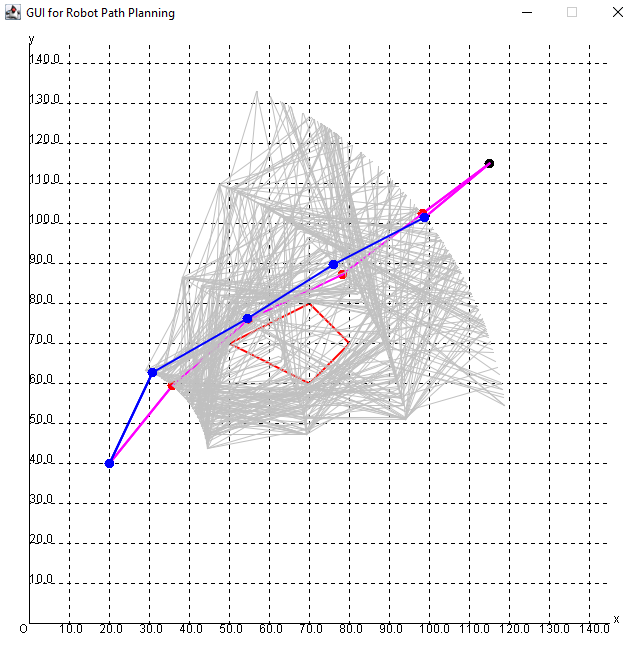
* Đường màu BLUE là đường đi của Robot dựa trên thuật toán PSO
* Đường màu MAGENTA là đường đi của Robot dựa trên thuật toán PSO lai với DE

### ***4.3.1. Ảnh hưởng của hình dạng chướng ngại vật đến đường đi của robot***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tham số thực nghiệm | Bộ dữ liệu 1 | Bộ dữ liệu 2 | Bộ dữ liệu 3 | Bộ dữ liệu 4 | Bộ dữ liệu 5 |
| Kích thước quần thể NP | 100 | 100 | 100 | 100 | 100 |
| Số lần lặp | 10 | 10 | 10 | 10 | 10 |
| Kích thước hai tập không bị trội Na và N’a | 10 | 10 | 10 | 10 | 10 |
| Chướng ngại vật và hình dạng | 1 chướng ngại vật (tứ giác lồi) | 2 chướng ngại vật (1 tứ giác và 1 tam giác) | 3 chướng ngại vật (  3 tứ giác) | 4 chướng ngại vật ( 3 tứ giác và 1 tam giác) | 5 chướng ngại vật ( 5 đa giác ) |
| Số lần chạy | 30 | 30 | 30 | 30 | 30 |

*Bảng 1. Thông số cài đặt thực nghiệm sự ảnh hưởng của hình dạng chướng ngại vật đến đường đi của robot*

### ***4.3.1.1 Bộ dữ liệu 1***



*Hình 4.1 Đường đi robot tìm đường trong bộ dữ liệu 1 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 1 chướng ngại vật lồi cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (30.879, 62.509) (54.5, 76.19) (76.138, 89.735) (98.986, 101.328)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (35.899, 59.293) (54.778, 75.924) (78.418, 87.035) (98.251, 102.264)

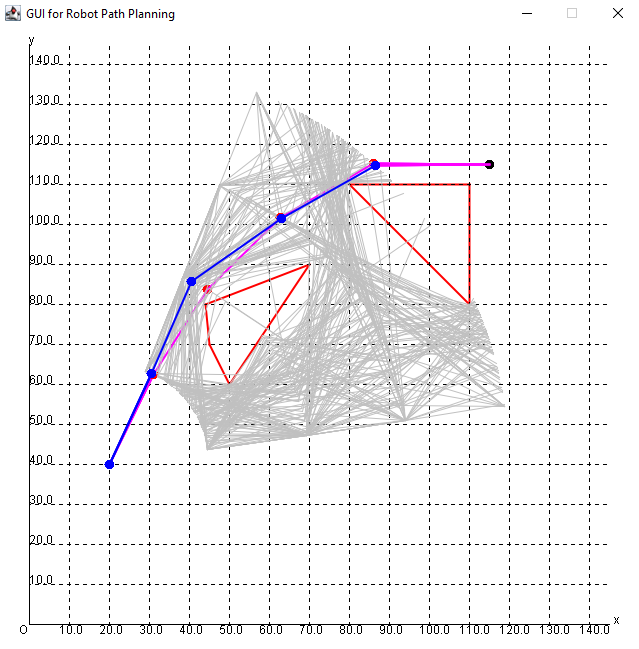
Chiều dài đường đi của PSO là: 124.5020384400685

Chiều dài đường đi thuật toán PSO lai DE là : 122.32708012474185

Thời gian chạy của thuật toán PSO là : 1266ms

Thời gian chạy của thuật toán PSO lai DE là : 1322ms

### ***4.3.1.2 Bộ dữ liệu 2***



*Hình 4.2 Đường đi robot tìm đường trong bộ dữ liệu 2 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 2 chướng ngại vật là 1 đa giác và 1 tam giác cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (30.593, 62.645) (40.659, 85.533) (63.095, 101.382) (86.695, 114.51)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (31.037, 62.432) (44.508, 83.581) (62.819, 101.576) (86.06, 115.074)

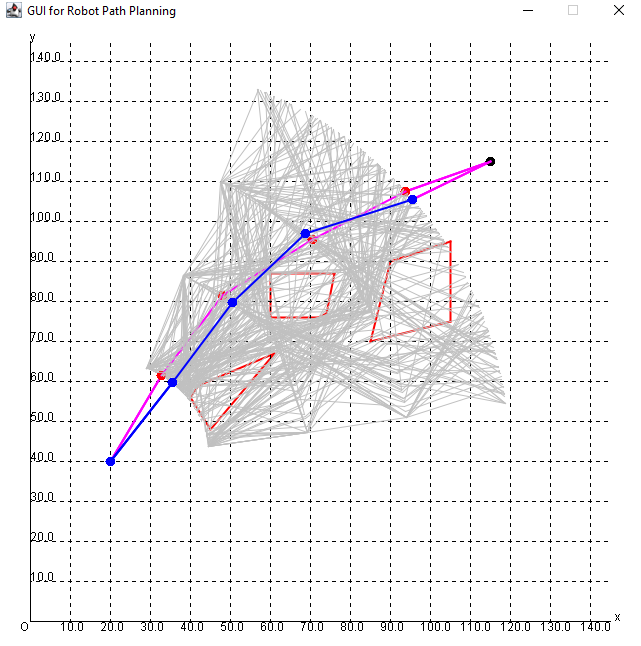
Chiều dài đường đi của PSO là: 132.7881203477861

Chiều dài đường đi thuật toán PSO lai DE là : 131.5643248819359

Thời gian chạy của thuật toán PSO là : 1244ms

Thời gian chạy của thuật toán PSO lai DE là : 1327ms

### ***4.3.1.3 Bộ dữ liệu 3***



*Hình 4.3 Đường đi robot tìm đường trong bộ dữ liệu 3 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 3 chướng ngại vật là những đa giác lồi cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (29.863, 62.972) (52.231, 78.225) (67.371, 98.147) (93.421, 107.892)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (30.999, 62.451) (47.073, 82.037) (69.481, 96.361) (95.673, 105.373)

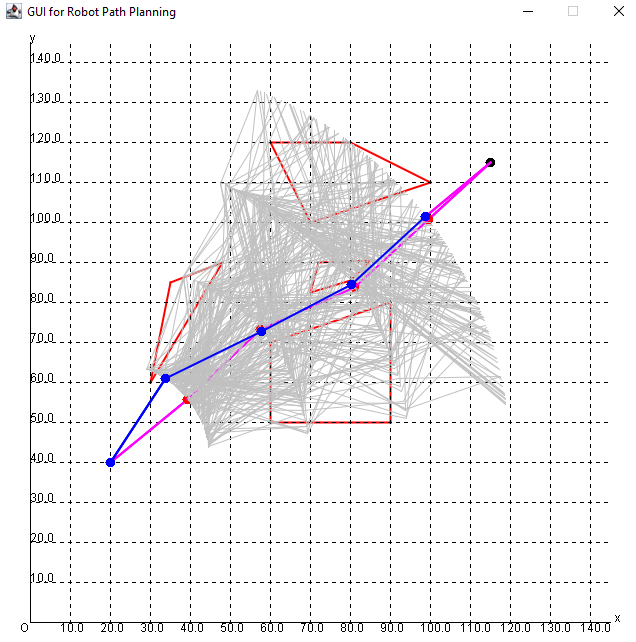
Chiều dài đường đi của PSO là: 127.6278888600193

Chiều dài đường đi thuật toán PSO lai DE là : 126.22398569131843

Thời gian chạy của thuật toán PSO là : 1346ms

Thời gian chạy của thuật toán PSO lai DE là : 1381ms

### ***4.3.1.4 Bộ dữ liệu 4***



*Hình 4.4 Đường đi robot tìm đường trong bộ dữ liệu 4 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 4 chướng ngại vật cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (33.816, 60.835) (57.845, 72.676) (80.382, 84.486) (98.875, 101.471)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (39.453, 55.703) (57.552, 73.013) (81.043, 83.574) (99.544, 100.603)

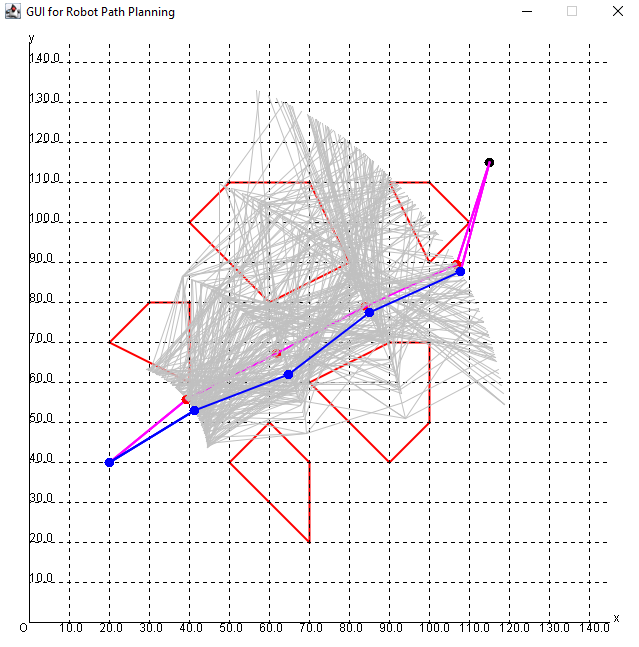
Chiều dài đường đi của PSO là: 123.389881639917

Chiều dài đường đi thuật toán PSO lai DE là : 122.0675426360603

Thời gian chạy của thuật toán PSO là : 1370ms

Thời gian chạy của thuật toán PSO lai DE là : 1397ms

### ***4.3.1.5 Bộ dữ liệu 5***



*Hình 4.5 Đường đi robot tìm đường trong bộ dữ liệu 5 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 5 chướng ngại vật là những đa giác lồi cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (41.473, 52.802) (64.957, 61.883) (85.073, 77.289) (107.923, 87.639)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (39.434, 55.726) (61.935, 67.23) (84.142, 78.868) (106.978, 89.345)

Chiều dài đường đi của PSO là: 128.86187644792727

Chiều dài đường đi thuật toán PSO lai DE là : 127.34739079998964

Thời gian chạy của thuật toán PSO là : 1398ms

Thời gian chạy của thuật toán PSO lai DE là : 1432ms

### ***4.3.1.6 Nhận xét sự ảnh hưởng của hình dạng chướng ngại vật đến đường đi của robot***

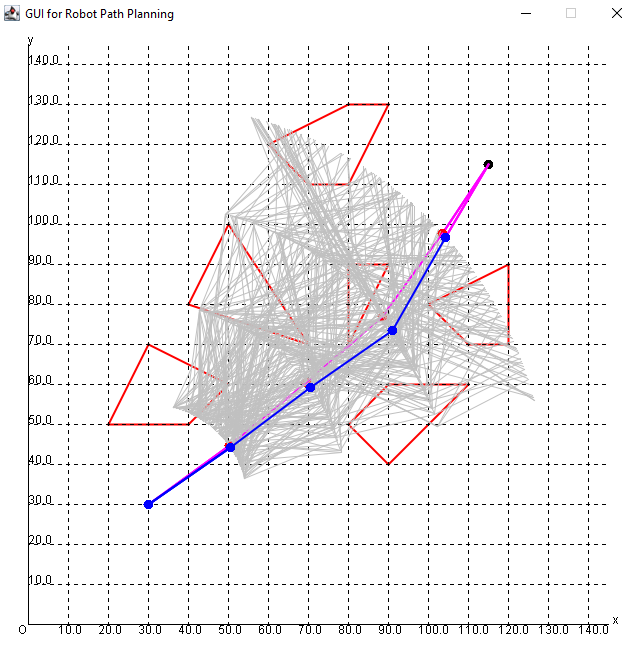
* Với những chướng ngại vật là tam giác và tứ giác thì đường đi của robot tốt hơn về cả quãng đường và thời gian

### **4.3.2 Ảnh hưởng của mật độ chướng ngại vật đến đường đi của robot**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tham số thực nghiệm | Bộ dữ liệu 6 | Bộ dữ liệu 7 | Bộ dữ liệu 8 | Bộ dữ liệu 9 | Bộ dữ liệu 10 |
| Kích thược quần thể | 100 | 100 | 100 | 100 | 100 |
| Số lần lặp | 10 | 10 | 10 | 10 | 10 |
| Kích thước hai tập không bị trội Na và N’a | 10 | 10 | 10 | 10 | 10 |
| Mật độ chướng ngại vật | 6 chướng ngại vật | 7 chướng ngại vật | 8 chướng ngại vật | 9 chướng ngại vật | 10 chướng ngại vật |
| Số lần chạy | 30 | 30 | 30 | 30 | 30 |

*Bảng 2. Thông số cài đặt thực nghiệm sự ảnh hưởng của mật độ chướng ngại vật đến đường đi của robot*

### ***4.3.2.1 Bộ dữ liệu 6***



*Hình 4.6 Đường đi robot tìm đường trong bộ dữ liệu 6 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 6 chướng ngại vật cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (30, 30) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (50.592, 44.176) (70.626, 59.146) (91.239, 73.298) (104.465, 96.745)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (50.251, 44.659) (69.678, 60.424) (88.515, 76.915) (103.7, 97.589)

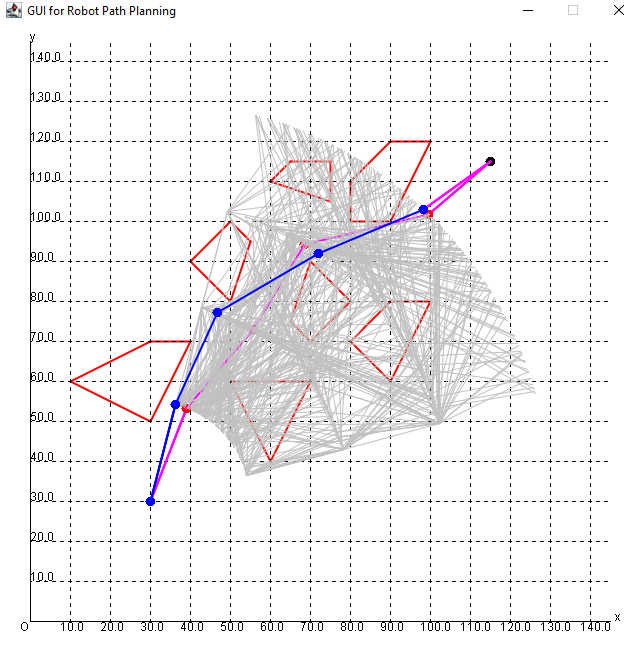
Chiều dài đường đi của PSO là: 123.00935420286724

Chiều dài đường đi thuật toán PSO lai DE là : 121.4624600888346

Thời gian chạy của thuật toán PSO là : 1263ms

Thời gian chạy của thuật toán PSO lai DE là : 1316ms

### ***4.3.2.2 Bộ dữ liệu 7***



*Hình 4.7 Đường đi robot tìm đường trong bộ dữ liệu 7 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 7 chướng ngại vật là những đa giác cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (30, 30) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (36.47, 54.148) (46.888, 77.062) (72.206, 91.997) (98.449, 102.903)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (39.224, 53.236) (55.876, 72.783) (68.746, 94.216) (99.749, 101.659)

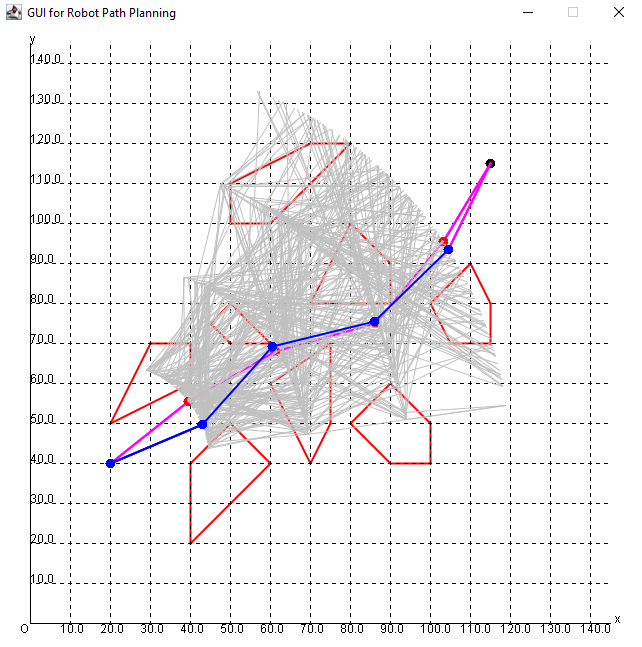
Chiều dài đường đi của PSO là: 128.4848448241107

Chiều dài đường đi thuật toán PSO lai DE là : 127.82462722394787

Thời gian chạy của thuật toán PSO là : 1496ms

Thời gian chạy của thuật toán PSO lai DE là : 1576ms

### ***4.3.2.3 Bộ dữ liệu 8***



*Hình 4.8 Đường đi robot tìm đường trong bộ dữ liệu 8 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 8 chướng ngại vật là những đa giác cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (20, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (43.123, 49.505) (60.556, 69.244) (86.09, 75.455) (104.633, 93.267)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (39.718, 55.369) (61.495, 67.895) (86.361, 74.947) (103.319, 95.299)

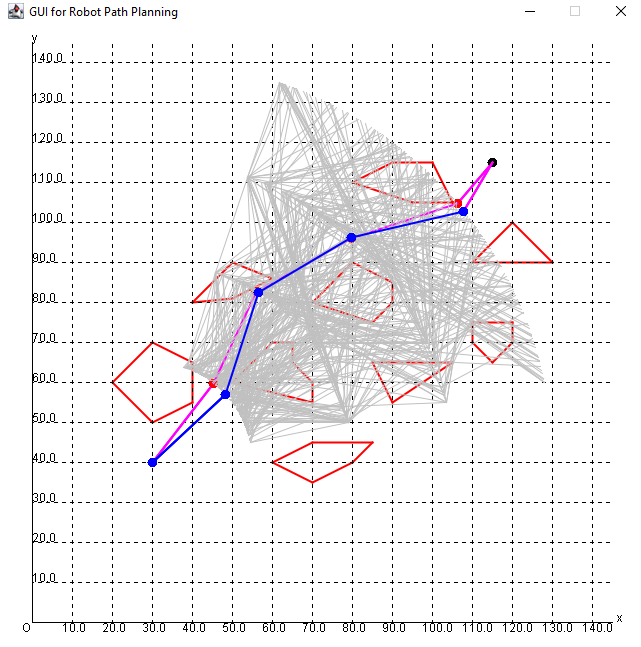
Chiều dài đường đi của PSO là: 127.40447750728356

Chiều dài đường đi thuật toán PSO lai DE là : 125.36372495012517

Thời gian chạy của thuật toán PSO là : 1511ms

Thời gian chạy của thuật toán PSO lai DE là : 1629ms

### ***4.3.2.4 Bộ dữ liệu 9***



*Hình 4.9 Đường đi robot tìm đường trong bộ dữ liệu 9 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 9 chướng ngại vật là những tam giác , tứ giác và đa giác cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (30, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (48.366, 56.962) (56.592, 82.343) (79.882, 96.007) (107.921, 102.676)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (45.377, 59.711) (56.13, 82.629) (79.931, 95.963) (106.279, 104.665)

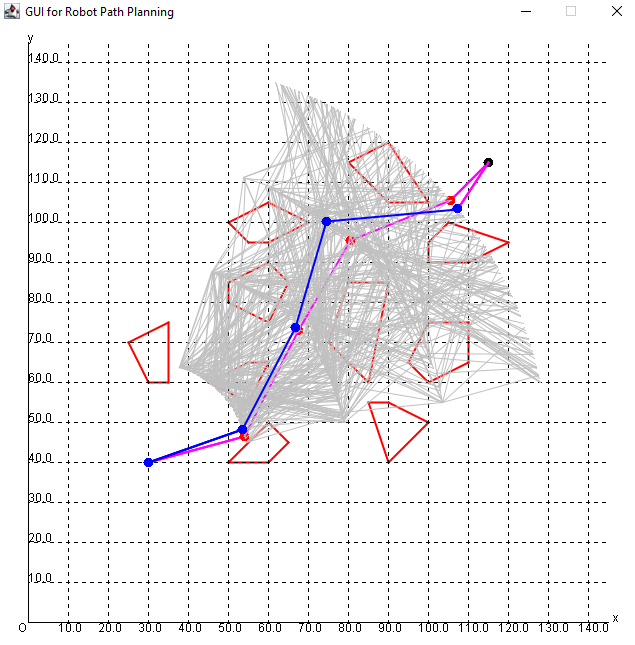
Chiều dài đường đi của PSO là: 121.71731835851287

Chiều dài đường đi thuật toán PSO lai DE là : 118.86716765670914

Thời gian chạy của thuật toán PSO là : 1391ms

Thời gian chạy của thuật toán PSO lai DE là : 1443ms

### ***4.3.2.5 Bộ dữ liệu 10***



*Hình 4.10 Đường đi robot tìm đường trong bộ dữ liệu 10 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm 10 chướng ngại vật là những tứ giác, đa giác cố định, điểm xuất và điểm mục tiêu của robot tương ứng là: (30, 40) và (115, 115).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (53.625, 48.176) (66.937, 73.699) (74.667, 100.248) (107.456, 103.25)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (54.186, 46.329) (67.676, 72.871) (80.685, 95.282) (105.665, 105.382)

Chiều dài đường đi của PSO là: 128.32610266686882

Chiều dài đường đi thuật toán PSO lai DE là : 121.03451209475278

Thời gian chạy của thuật toán PSO là : 1403ms

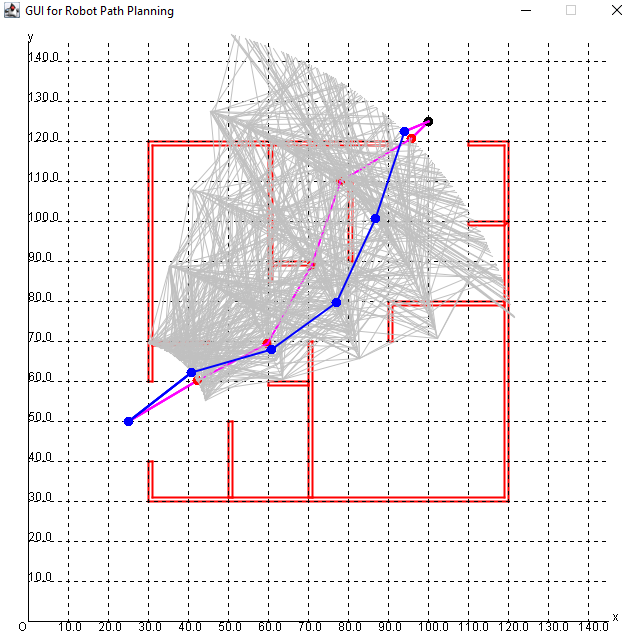
Thời gian chạy của thuật toán PSO lai DE là : 1464ms

### ***4.3.2.6 Nhận xét sự ảnh hưởng của mật độ chướng ngại vật đến đường đi của robot***

* Với mật độ chướng ngại vật càng tăng thì thời gian và quãng đường đi của robot cũng khác nhau
* Đường đi và thời gian sẽ tăng lên khi mật độ chướng ngại vật tăng

### **4.3.3 Khả năng tìm đường đi của robot trong môi trường phòng thí nghiệm**

### ***4.3.3.1 Bộ dữ liệu room1***



*Hình 4.11 Đường đi robot tìm đường trong bộ dữ liệu room1 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm những chướng ngại vật như hình trên , điểm xuất và điểm mục tiêu của robot tương ứng là: (25, 50) và (100, 125).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (40.915, 62.112) (60.802, 67.838) (77.12, 79.724) (86.894, 100.687) (94.006, 122.375)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (42.289, 60.054) (59.972, 69.416) (70.782, 88.781) (78.192, 109.755) (95.767, 120.654)

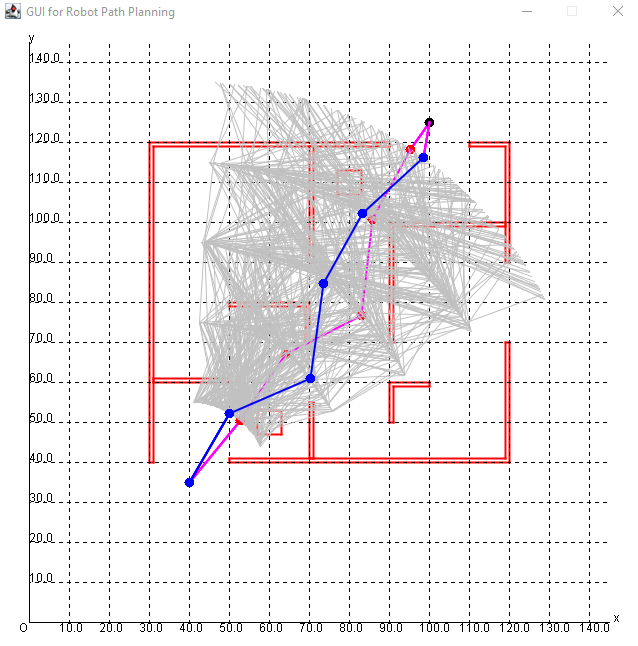
Chiều dài đường đi của PSO là: 113.38016311141176

Chiều dài đường đi thuật toán PSO lai DE là : 111.17749993552377

Thời gian chạy của thuật toán PSO là : 1508ms

Thời gian chạy của thuật toán PSO lai DE là : 1597ms

### ***4.3.3.2 Bộ dữ liệu room2***



*Hình 4.12 Đường đi robot tìm đường trong bộ dữ liệu room2 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm những chướng ngại vật như hình trên , điểm xuất và điểm mục tiêu của robot tương ứng là: (40, 35) và (100, 125).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (50.218, 52.193) (70.432, 60.96) (73.559, 84.737) (83.469, 102.16) (98.538, 116.076)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (52.882, 50.299) (64.127, 66.904) (83.233, 76.604) (85.824, 100.576) (95.467, 118.207)

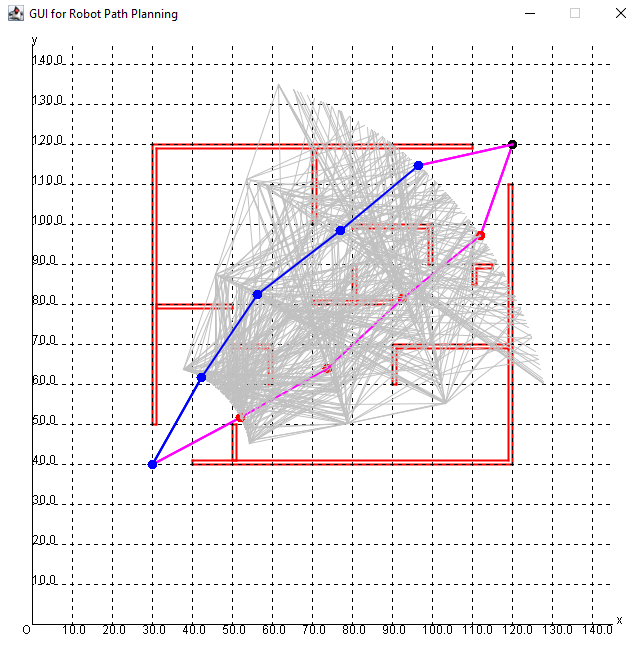
Chiều dài đường đi của PSO là: 115.61334717307842

Chiều dài đường đi thuật toán PSO lai DE là : 113.85574319629298

Thời gian chạy của thuật toán PSO là : 1447ms

Thời gian chạy của thuật toán PSO lai DE là : 1555ms

### ***4.3.3.3 Bộ dữ liệu room3***



*Hình 4.13 Đường đi robot tìm đường trong bộ dữ liệu room3 với phương pháp sử dụng cung tròn*

Bộ dữ liệu bao gồm những chướng ngại vật như hình trên , điểm xuất và điểm mục tiêu của robot tương ứng là: (30, 40) và (120, 120).

Toạ độ đường đi của robot dựa trên thuật toán PSO là : (42.389, 61.714) (56.464, 82.422) (77.048, 98.408) (96.631, 114.567)

Toạ độ đường đi của robot dựa trên thuật toán PSO lai DE là : (52.153, 51.586) (73.911, 63.913) (92.583, 81.332) (112.044, 97.174)

Chiều dài đường đi của PSO là: 125.48219968848913

Chiều dài đường đi thuật toán PSO lai DE là : 124.80935000954135

Thời gian chạy của thuật toán PSO là : 1454ms

Thời gian chạy của thuật toán PSO lai DE là : 1526ms

# **KẾT LUẬN**

1. **Về lý thuyết, đồ án đã trình bày được những nội dung sau:**

* Giới thiệu chung về Robot, Robot di động, những ứng dụng và những thách thức cần phải vượt qua
* Tổng quan các phương pháp mô hình hóa môi trường, mô hình hóa môi trường sử dụng cung tròn
* Tổng quan về giải thuật tối ưu bầy đàn và ảnh hưởng của các tham số trong giải thuật đối với thuật toán.
* Tổng quan về giải thuật tiến hóa sai khác vi phân, các phép toán của thuật toán: đột biến, chọn lọc, lai ghép.
* Tổng quan về bài toán lập kế hoạch đường đi cho robot.

1. **Về mặt thực nghiệm, đồ án đã thu được các kết quả sau:**

* Cài đặt giải thuật tối ưu hóa bầy đàn và giải thuật tối ưu hóa bầy đàn lai giải thuật tiến hóa vi phân sai khác , áp dụng với phương pháp mô hình hóa môi trường sử dụng cung tròn.
* Chạy thực nghiệm phương pháp mô hình hóa môi trường sử dụng cụng tròn đã đề xuất trên 10 bộ dữ liệu môi trường ( số lượng chướng ngại vật tăng dần từ 1 đến 10 ) và 3 bộ dữ liệu room1, room2, room3

1. **Kết quả thực nghiệm cho thấy:**

* Giải thuật tối ưu hóa bầy đàn lai giải thuật tiến hóa vi phân sai khác có đường đi trung bình tốt hơn so với giải thuật tối ưu hóa bầy đàn trong môi trường mô hình hóa sử dụng cung tròn
* Nhưng bên cạnh đó thời gian chạy của thuật toán tối ưu hóa bầy đàn luôn luôn nhanh hơn thuật toán tối ưu hóa bầy đàn lai giải thuật vi phân sai khác
* Phương pháp mô hình hóa này rất tốt cho những bài toán tìm đường đi cho robot có nhiều điểm đích cần tới vì phương pháp này không phụ thuộc vào tọa độ điểm đích
* Qua nhiều lần chạy thử nghiệm , thấy mật độ chướng ngại vật ảnh hưởng rất nhiều đến tốc độ chạy của thuật toán và với mật độ chướng ngại vật dày thì có sảy ra va chạm giữa đường đi và chướng ngại vật

# **TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | J.Kennedy and R.Eberhart, "Particle swarm optimization", in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, Australia, 1995. |
| [2] | Parsopoulos et al, "Particle swarm optimization and intelligence: advances and applications," 2010. |
| [3] | Zhang et al ,"Path Planning of Mobile Robot Based on Hybrid," 2018. |
| [4] | Deb and Kalyanmoy, "Multi-objective optimization," 2014. |
| [5] | Marler et al, "Survey of multi-objective optimization methods for engineering," in *Structural and Multidisciplinary Optimization volume 26, pages369–395(2004)*, 2004. |
| [6] | S.Zhongke et al, "Overview of multi-objective optimization methods," in *Journal of Systems Engineering and Electronics ( Volume: 15, Issue: 2, June 2004)*, 2004. |
| [7] | Elarbi et al, "Multi-objective Optimization: Classical and Evolutionary Approaches," in *Recent Advances in Evolutionary Multi-objective Optimization*, 2016. |
| [8] | K. Price et al, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," 1997. |
| [9] | Feoktistov and Vitaliy, *Differential evolution*, Springer US, 2006. |
| [10] | N.Sariff and N.Buniyamin, "An overview of autonomous mobile robot path planning algorithms," 2006. |
| [11] | J.Canny, *The Complexity of Robot Motion Planning*, The MIT Press: Cambridge, MA, USA,1988. |
| [12] | Zhang et al, "Path planning for the mobile robot: A review.", Symmetry 10.10 (2018): 450, 2018. |
| [13] | Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," in *IEEE Trans. Evolut. Comput. 10 (6) (2006) 658– 674*, 2006. |
| [14] | D. Chafekar et al, "Constrained multi-objective optimization using steady state genetic algorithms," in *Genetic and Evolutionary Computation — GECCO 2003 pp 813-824*, 2003. |
| [15] | Zhang et al, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," in *Neurocomputing 103 (2013): 172-185.2013*, 2013. |