Tìm đường đi Robot trong Môi trường động bằng Phát triển và Cải tiến Thuật toán Vết Dầu loang

Lê Anh Tuấn[1], Lê Văn Hưng[2], Phạm Văn Hải [3]
[1] Viện Công nghệ thông tin và truyền thông, Trường Đại học Bách Khoa Hà Nội
[2] Khoa Công nghệ thông tin, Trường Đại học Mỏ-Địa chất
[3] Bộ môn Hệ thống thông tin, Viện Công nghệ thông tin và truyền thông, Trường Đại học Bách Khoa Hà Nội
[1]tuanla.hust@gmail.com [3]haipv@soict.hust.edu.vn,

TÓM TẮT *: Trong bài báo này, chúng tôi trình bày một phương pháp tìm đường đi tối ưu từ điểm xuất phát đến đích cho robot trong môi trường có vật cản cố định và di động dựa trên thuật toán "Vết dầu loang". Hiện nay, các công trình nghiên cứu trên thế giới đã đưa ra nhiều thuật toán đường đi bao phủ nhằm giải quyết một số vấn đề trong điều khiển Robot lau nhà và lịch trình đường đi bao phủ tối ưu khoảng cách. Nghiên cứu tổng quan chỉ ra các trường hợp của di chuyển đường đi của robot như sau: Robot di chuyển trong môi trường tĩnh theo lịch trình định trước và di chuyển trong môi trường động khi gặp các chướng ngại vật di động. Chủ yếu những thuật toán truyền thống trong lớp bài toán đường đi bao phủ bao gồm vết dầu loang truyền thống tập trung giải quyết bài toán trong môi trường tĩnh. Nghiên cứu đã áp dụng cải tiến thuật toán vết dầu loang nhằm tìm ra hướng giải quyết cho bài toán robot di chuyển trong môi trường động, có nhiều vật cản khi cùng tham gia di chuyển. Đồng thời, nghiên cứu cũng chỉ ra sử dụng kỹ thuật "bán kính quét" nhằm xác định vị trí, và phát hiện vật cản để có xử lý tình huống thông minh "heuristics" vượt qua hoặc tránh chướng ngại vật.*

Từ khóa: Vết dầu loang, tìm đường đi tối ưu cho robot trong môi trường động, đường đi bao phủ.

1. GIỚI THIỆU

Vết dầu loang là phương phương pháp tìm kiếm theo chiều rộng được mô phỏng như hình ảnh vết dầu loang trên mặt nước. Trong thực tế, nếu đổ một giọt dầu trên bề mặt nước thì vết loang dầu đó sẽ loang ra theo không gian và thời gian. Ví dụ trên mặt nước, vết dầu sẽ loang theo đường tròn tức là những điểm cùng khoảng cách so với tâm vết loang thì vết dầu đến cùng lúc. Đây cũng là điều khá thú vị, bất kể điểm nào trên mặt nước có liên thông với với tâm vết dầu thì sẽ loang đến đó. Ứng dụng thuật toán vết dầu loang với các kỹ thuật cải tiến để giải quyết bài toán: tìm đường đi tối ưu trong môi trường tĩnh, xác định tính liên thông trong đồ thị,…

Xác định vị trí (localization) và xác định đường đi (path planning-PP) là hai thành phần quan trọng nhất của bài toán điều hướng (navigation). Xác định vị trí là tìm một vị trí trên bản đồ tương ứng với vị trí trong thế giới thực, còn bài toán PP là tìm một đường đi không đụng độ (collision-free) tối ưu từ vị trí bắt đầu đến vị trí đích. PP được xem là một bài toán lớn trong ngành khoa học về người máy (robotics). Trong [1] các tác giả đề xuất phân các phương pháp giải bài toán PP thành hai loại: (i) tìm đường đi toàn cục (GPP) và (ii) tìm đường đi cục bộ (LPP). Trong GPP, robot có toàn bộ thông tin về môi trường (vị trí của các vật cản và đích), còn trong LPP, robot không có hoặc có không đầy đủ thông tin về môi trường. Có nhiều kỹ thuật được sử dụng trong GPP như trường điện từ (potential field - PF), phân rã thành ô (cell decomposition - CD) và đồ thị trực quan (Visibility Graph – VG). Trong [2] các tác giả sử dụng cách phân loại sau cho các phương pháp PP: (1) Các phương pháp cổ điển: CD, PF, bản đồ đường đi (road map – RM) và (2) Các phương pháp heuristic: mạng nơ ron (Neural Network), giải thuật di truyền (Genetic Algorithm), tối ưu bầy đàn (Particle Swarm Optimization) và tối ưu đàn kiến (Ant Colony Optimization). Trong [3] các tác giả phân loại các thuật toán PP dựa vào tính đầy đủ của thông tin thành: (a) chính xác và (b) heuristic. Các thuật toán chính xác tìm thấy lời giải tối ưu nếu chúng tồn tại còn các thuật toán heuristic tìm kiếm các lời giải có chất lượng tốt trong khi tập trung vào việc giảm thời gian tìm kiếm.

Trong các phương pháp cổ điển, hoặc một lời giải được tìm thấy hoặc một lời giải như vậy được chứng minh là không tồn tại. Điểm bất lợi chính của các phương pháp này là khối lượng tính toán lớn và không có khả năng làm việc trong môi trường không chắc chắn. Phương pháp CD phân chia không gian lớn thành các không gian nhỏ hơn để dễ dàng cho việc khảo sát và được sử dụng nhiều trong các bài toán PP. Ý tưởng của phương pháp này là làm giảm không gian tìm kiếm bằng cách sử dụng cách biểu diễn bằng các ô. Mục đích của nó là tìm một dãy các ô kề nhau (có biên chung), không đụng độ từ vị trí bắt đầu đến vị trí đích. Một dãy như vậy biểu diễn một đường đi từ ô bắt đầu đến ô đích. Nhược điểm của các phương pháp này sự bùng nổ tổ hợp (combinatorial explosion) về độ phức tạp tính toán [4]. Phương pháp PF thường được sử dụng để điều khiển chuyển động của robot trong các tình huống tránh vật cản [5]. Nói chung, PF có thể được hiểu bao gồm hai lực hút và đẩy, trong đó đích có lực hút còn các vật cản có lực đẩy đối với robot. Kết quả tổng hợp của các lực hút và đẩy sẽ dẫn robot đi đến đích. Trong phương pháp RM, một bản đồ đường đi được xây dựng với một tập các đường đi, sau đó các đường đi này sẽ được sử dụng cho việc di chuyển của robot [6]. Do đó, trong phương pháp này, bài toán PP có thể được xem như bài toán tìm đường đi ngắn nhất trong các đường đi có thể từ vị trí bắt đầu đến vị trí kết thúc sử dụng mạng bản đồ đường đi.

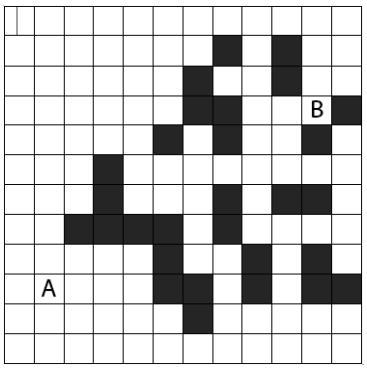
Bài toán điều hướng chịu ảnh hưởng bởi độ chính xác của bản đồ và kỹ thuật định vị. Bản chất động và không thể dự báo trước của các ứng dụng thực tế thường làm cho bài toán điều hướng trở thành một nhiệm vụ đầy thách thức. Trong [7] các tác giả đề xuất việc sử dụng các phương pháp heuristic dựa trên hành vi (behavioral-based) để xử lý các tình huống của môi trường thực tế. Trong các phương pháp này, một tập hành vi đơn giản được định nghĩa trước sẽ được thiết kế để giải quyết các tình huống phức tạp. Do các phương pháp này phụ thuộc vào trạng thái hiện thời và tập hành vi được thiết kế cho trạng thái đó, chúng đáng tin cậy trong môi trường động, tuy vậy, vẫn gặp khó khăn trong môi trường không chắc chắn. Ngược lại với các phương pháp cổ điển, các thuật toán heuristic không đảm bảo việc tìm được lời giải, nhưng chắc chắn rằng nếu chúng tìm được một lời giải thích hợp thì sẽ trong thời gian ngắn hơn so với các phương pháp cổ điển. Ngoài ra, chúng có thể tìm thấy một lời giải tồi (tối ưu cục bộ).

Trong bài báo này, chúng tôi kết hợp các kỹ thuật của các phương pháp PP cổ điển và heuristic để giải bài toán tìm đường đi của robot trong môi trường có cả vật cản tĩnh và vật cản động. Cụ thể, chúng tôi sẽ chia không gian di chuyển của robot thành các ô hình vuông có kích thước đủ bé. Các ô này được đánh dấu có vật cản hay không cho trường hợp các vật cản tĩnh. Sau đó, dựa trên nguyên lý vết dầu loang từ vị trí đích, một ma trận chi phí để đi đến đích sẽ được tính cho tất cả các ô có thể đi đến đích. Về bản chất, ma trận này có thể được xem như là một RM với chi phí để đi đến đích đã được tính. Trong trường hợp chỉ có các vật cản tĩnh như ban đầu, dựa trên ma trận chi phí nói trên, một đường đi tối ưu (với chi phí thấp nhất) sẽ được xác định một cách dễ dàng. Trong trường hợp có vật cản động, chúng tôi giả thiết rằng robot được trang bị cảm ứng để xác định được vị trí, hướng di chuyển và vị trí có khả năng đụng độ của vật cản với robot trong một bán kính thích hợp nào đó. Khi di chuyển trong môi trường có vật cản động, robot sẽ dựa vào khả năng phát hiện đụng độ và tính toán chi phí để có những hành vi thích hợp với mục đích tối ưu hóa chi phí đi đến đích. Tuy nhiên, cũng giống như các phương pháp heuristic khác, phương pháp này có khả năng không tìm được lời giải hoặc tìm được một lời giải không tối ưu trong môi trường động, không chắc chắn.

1.1. Vấn đề di chuyển đường đi Robot

Trong không gian di chuyển được chia thành các ô hình vuông có các vật cản cố định và di động, hãy tìm đường đi nhanh nhất của robot từ ô xuất phát đến ô đích biết rằng robot chỉ có thể di chuyển từ một ô sang một ô khác có cạnh chung. Trong bài toán này robot được trang bị cảm ứng để phát hiện các vật cản xung quanh nó. Với cảm ứng này robot có thể phát hiện được vật cản trong phạm vi bán kính nào đó gọi là bán kính quét. Ngoài ra, robot còn có khả năng xác định hướng di chuyển và tốc độ của vật cản trong bán kính quét để xác định nó có khả năng va chạm với vật cản hay không và vị trí ô có khả năng xảy ra va chạm.

Ví dụ 1. Hình 1 mô tả bài toán tìm đường đi của robot từ ô A đến ô B trong không gian có kích thước hình chữ nhật MxN với các ô có vật cản được tô màu đen.



Hình 1: Bài toán tìm đường đi nhanh nhất

1.2. Lập luận nghiên cứu và phương pháp tiếp cận

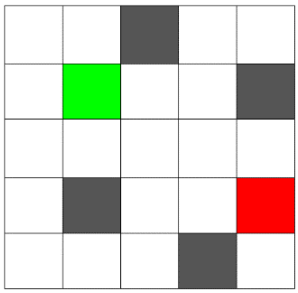
Giải pháp của bài toán dựa trên 2 giai đoạn như sau:

Giai đoạn 1: Dựa trên nguyên lý vết dầu loang để xây dựng ma trận chi phí đến đích:

Nguyên lý của dầu khi loang trên môi trường lỏng là loang đều ra các điểm xung quanh, những điểm ở gần được loang đến trước, những điểm ở ra được loang đến sau. Trong bài toán duyệt đồ thị có cấu trúc cây, nó tương đương với thuật toán tìm kiếm theo chiều rộng (Best First Search-BFS) với tâm của vết dầu chính là gốc của cây. Thuật toán BFS thường sử dụng cấu trúc hàng đợi để lưu các nút duyệt [8].

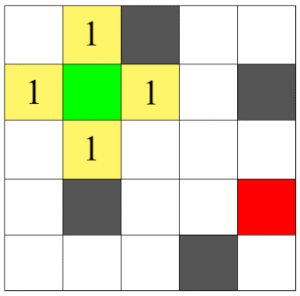
Ta xây dựng ma trận chi phí đến đích bằng cách bắt đầu loang từ vị trí đích. Cách làm này cho chúng ta biết được trước chi phí thời gian di chuyển từ một vị trí nào đó đến đích, do đó, cho phép ta chọn được đường đi tốt nhất từ vị trí đó đến đích nếu trên đường đi hiện tại xuất hiện vật cản.

Ví dụ 2: Ví dụ sau đây minh họa quá trình loang bắt đầu từ ô đích. Con số trong mỗi ô thể hiện chi phí (số lần dịch chuyển) từ ô đó đến đích. Hình 2 mô tả bài toán tìm đường đi với ô bắt đầu có màu xanh, ô đích có màu đỏ, các ô có vật cản có màu đen.



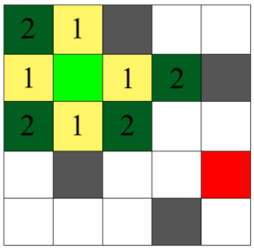
Hình 2: Bài toán tìm đường đi nhanh nhất

Hình 3 thể hiện bước loang đầu tiên từ ô màu xanh sang các ô màu vàng.



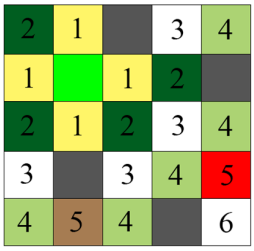
Hình 3: Sau bước loang đầu tiên

Hình 4 thể hiện kết quả sau hai bước loang.



Hình 4: Sau hai bước loang

Hình 5 thể hiện kết quả sau quá trình loang. Những ô cùng màu có cùng chi phí đến tâm.



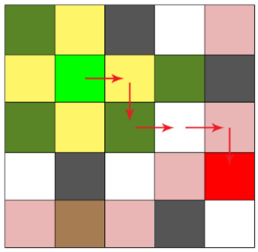
Hình 5: Kết quả của quá trình loang

Giai đoạn 2: Tìm đường đi từ ô xuất phát đến ô đích dựa trên ma trận chi phí

Trường hợp các vật cản là cố định, dựa trên ma trận chi phí ta có thể dễ dàng xác định được đường đi nhanh nhất từ ô xuất phát đến ô đích. Ô di chuyển tiếp theo sẽ được xác định như sau: Nếu robot đang đứng tại vị trí C(x,y) có chi phí là d(x,y) thì ô tiếp theo sẽ là $C_1(x_1, y_1)$ với:

$$\begin{cases} x_1 = x + Hx[i] \\ y_1 = y + Hy[i] \\ d(x_1, y_1) + 1 = d(x, y) \end{cases}, i = 0..3, \quad \begin{cases} Hx = \{0, 1, 0, -1\} \\ Hy = \{1, 0, -1, 0\} \end{cases}$$

Trong đó, *Hx, Hy* là độ lệch tọa độ của vị trí hiện tại so với vị trí mới. Quá trình di chuyển kết thúc khi $d(x_1, y_1) = 0$. Hình 6 thể hiện đường đi nhanh nhất từ ô xuất phát đến ô đích cho bài toán ở trong Ví dụ 2.



Hình 6: Kết quả của quá trình tìm đường đi

Trường hợp gặp vật cản xuất hiện trên đường đi, robot sẽ chọn một trong hai cách sau đây dựa trên việc so sánh chi phí thời gian giữa chúng: (a) Chờ vật cản đi qua và tiếp tục di chuyển theo lộ trình định trước, (b) Tránh vật cản bằng cách di chuyển theo lộ trình mới. Gọi Δc là chi phí thời gian chờ để tránh vật cản. Δc có đơn vị tính bằng chính thời gian cần để dịch chuyển từ một ô sang ô bên cạnh trong điều kiện bình thường. Gọi φ_1 là chi phí còn lại của trường hợp (a), φ_2 là chi phí còn lại của trường hợp (b), ta có:

$$\varphi_1 = d(x, y) + \Delta c$$

$$\varphi_2 = Min(d(x + Hx[i], y + Hy[i]) | \ i = 0..3)$$

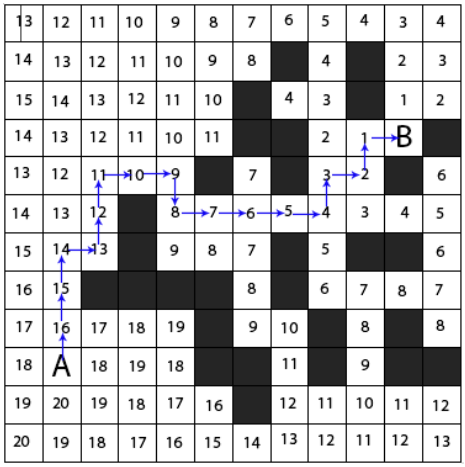
Nếu $\varphi_1 \leq \varphi_2$, robot sẽ chờ vật cản đi qua rồi đi theo lộ trình cũ. Chi phí thời gian di chuyển trong trường hợp này sẽ được cộng thêm thời gian chờ Δc. Nếu $\varphi_1 > \varphi_2$, robot sẽ đi theo lộ trình mới để tránh vật cản qua vị trí $(x + Hx[j], y + Hy[j])$ với $d(x + Hx[j], y + Hy[j]) = Min(d(x + Hx[i], y + Hy[i]) | \ i = 0..3)$ và chưa được đi qua.

Ví dụ 3: Giả sử sau quá trình loang ta xác định được ma trận chi phí cho bài toán tìm đường đi từ A đến B như trong Hình 7 sau:

13	12	11	10	9	8	7	6	5	4	3	4
14	13	12	11	10	9	8		4		2	3
15	14	13	12	11	10		4	3		1	2
14	13	12	11	10	11			2	1	B	
13	12	11	10	9		7		3	2		6
14	13	12		8	7	6	5	4	3	4	5
15	14	13		9	8	7		5			6
16	15					8		6	7	8	7
17	16	17	18	19		9	10		8		8
18	A	18	19	18			11		9		
19	20	19	18	17	16		12	11	10	11	12
20	19	18	17	16	15	14	13	12	11	12	13

Hình 7: Ma trận chi phí đến đích B

Sau khi có ma trận chi phí, robot sẽ xác định đường đi nhanh nhất (màu xanh) từ vị trí xuất phát A đến vị trí đích B như trong Hình 8.



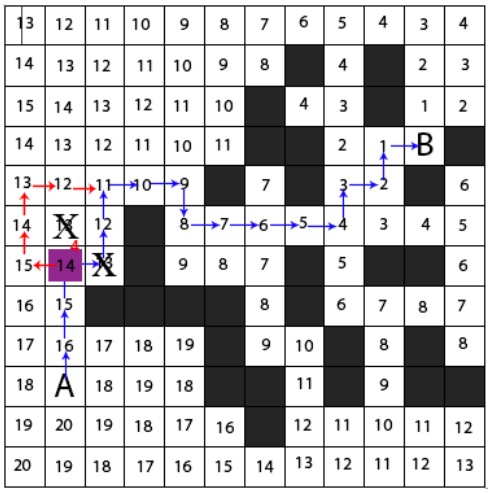
Hình 8: Đường đi nhanh nhất từ A đến B.

Ta coi gốc tọa độ của không gian di chuyển là góc trên-trái. Giả sử robot đang đến vị trí màu tím có tọa độ (7,2) với d(7,2) = 14 thì phát hiện có vật cản đang di chuyển về các vị trí (6,2) và (7,3) và sẽ đụng độ tại vị trí (7,3). Giả sử chi phí chờ để vật cản đi qua là Δc = 4, ta có:

$$\varphi_1 = d(x,y) + \Delta c = 14 + 4 = 18$$

$$\varphi_2 = Min(d(x + Hx[i], y + Hy[i]) | i = 1..3) = d(7,1) = 15$$

Như vậy $\varphi_1 > \varphi_2$ nên robot sẽ chọn đi theo lộ trình mới (có màu đỏ) đi qua ô có vị trí (7,1) như trong Hình 9. Trong trường hợp này chi phí của đường đi sẽ được cộng thêm $\varphi_2 - d(x,y) + 1$.



Hình 9: Trường hợp gặp vật cản

2. THUẬT TOÁN

Thuật toán sử dụng cấu trúc hàng đợi (Q) để duyệt theo chiều rộng. Giá trị d(x,y) là chi phí di chuyển từ vị trí (x,y) đến vị trí đích (x_B, y_B). Hàm check(x,y) cho giá trị *đúng* nếu ô (x,y) nằm trong không gian di chuyển và chưa được tính chi phí. Hàm next(x,y) trả về tọa độ của ô tiếp theo của lộ trình hiện tại. Hàm collision (x,y) cho biết tại vị trí (x,y) có đụng độ hay không. Hàm change(x,y) trả về tọa độ (x',y') của ô sẽ đi sang nếu thay đổi đường đi, nghĩa là ô (x',y') ở bên cạnh ô (x,y), không có khả năng đụng độ, chưa được đi qua và có chi phí d(x',y') bé nhất. Hàm save(x,y,x',y') lưu đường đi từ ô (x,y) sang ô (x',y'). Biến cost lưu chi phí của đường đi đến ô hiện tại.

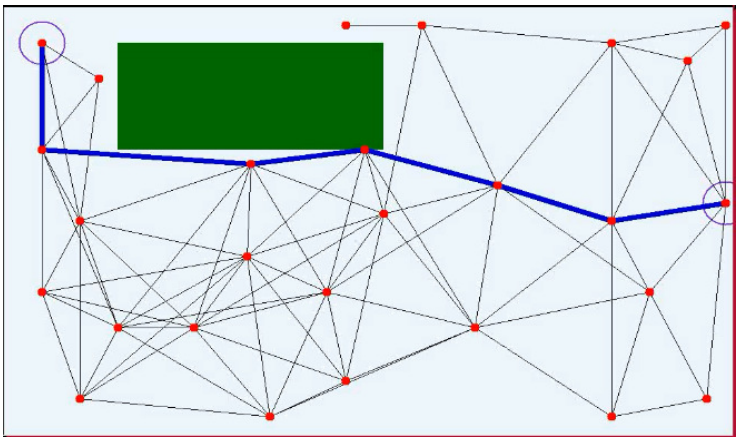
1) /* Khởi tạo ma trận chi phí */
2) for(i=0;i<M;i++)
3) for(j=0;j<N;j++) d(i,j) = ∞ /* ∞ là một giá trị đủ lớn, đánh dấu trường hợp không có đường đi */
4) /* Tính ma trận chi phí */
5) x = x_B, y = y_B /* bắt đầu từ đích */

```
6)  Q←(x,y)
7)  While(Q ≠ Φ)
8)      for( i = 0;i < 4;i ++)
9)              x' = x + Hx[i];
10)             y' = y + Hy[i];
11)             if(check(x',y'))
12)                     Q←(x',y');
13)                     d(x',y') = d(x,y) + 1;
14)             endif;
15)     endfor;
16)     (x,y) ← Q;
17) endWhile;
18) /* Xác định đường đi và chi phí */
19)  x = x_A, y = y_A; /* bắt đầu từ vị trí xuất phát */
20) cost = 0;
21) if(d(x,y) = ∞) /* kiểm tra có đường đi hay không */
22)     print("Path not found");
23) else
24)     while(x ≠ x_B || y ≠ y_B)
25)         (x1,y1) ← next(x,y);
26)         if(collision(x1,y1))
27)             φ1 = d(x,y) + Δc;
28)             φ2 = Min(d(x + Hx[i], y + Hy[i]));
29)             if(φ1 > φ2) /* Đi theo đường mới*/
30)                 (x1,y1) ← change(x,y);
31)                 cost = cost + φ2 − d(x,y) + 1;
32)             else
33)                 cost = cost + Δc;
34)             endIf;
35)         else
36)             cost = cost + 1;
37)         endIf
38)         save(x,y,x1,y1); /*Lưu đường đi từ (x,y) sang (x1,y1) */
39)         (x,y) ← (x1,y1); /* tọa độ (x1,y1) trở thành vị trí hiện tại */
40)     endWhile;
41) endIf;
```

3. KẾT QUẢ THỰC NGHIỆM

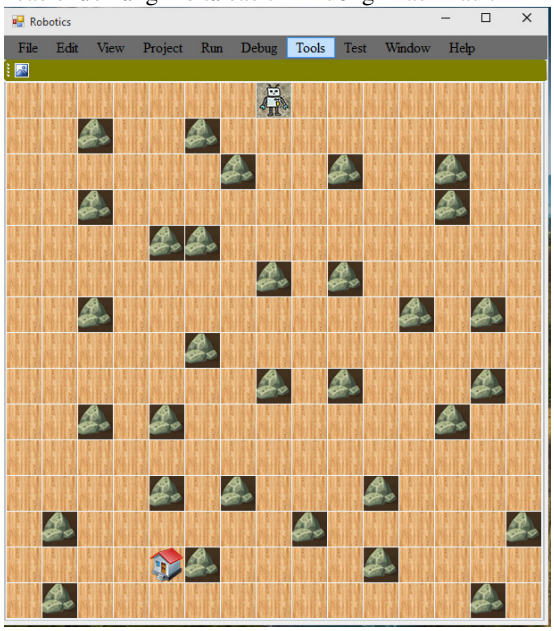
3.1. Kết quả thực nghiệm

Đường đi của Robot được biểu diễn dưới dạng cấu trúc dưới dạng đồ thị như hình 11. Để xác định lịch trình đường đi của Robot tìm đường đi tối ưu từ vị trí xuất phát đến vị trí đích, Robot cần tìm đường đi tối ưu tránh các chướng ngại vật trên đường đi. Thực nghiệm tiến hành với vài trăm nodes đồ thị và lịch trình đường đi khác nhau. Trong quá trình di chuyển Robot gặp chướng ngại vật di động Robot có khả năng tránh các chướng ngại vật di động và tìm đường đi tối ưu từ điểm xuất phát đến điểm đích.



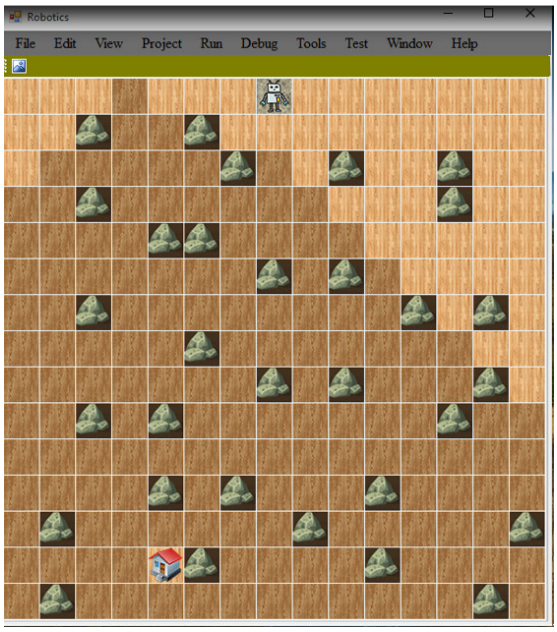
Hình 11: Kết quả tìm kiếm đường đi cho robot.

Chương trình cài đặt C++ với các chức năng mô tả các tình huống khác nhau . Hình



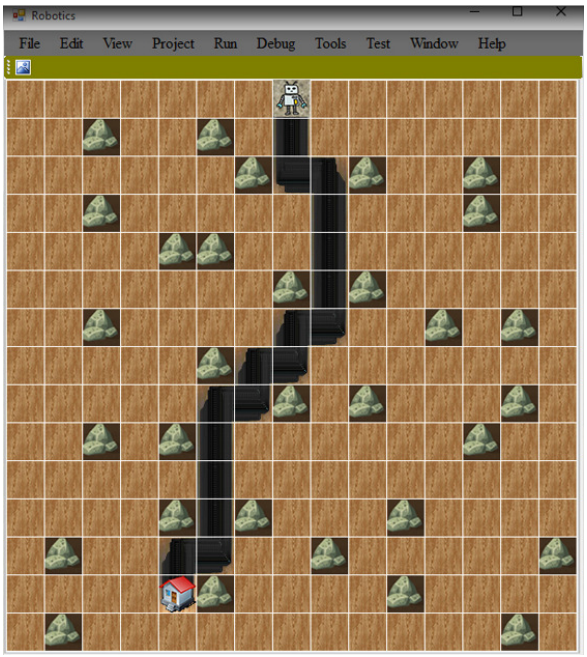
Hình 12 – Robot xuất phát trên đường đi gặp các vật cản di động

Chương trình mô phỏng vết dầu loang để mô tả lịch trình đường đi mô tả như hình 13.



Hình 13 – Vết dầu loang xác định từ điểm đích

Sau khi mô tả vết dầu loang xong thì robot sẽ tiến hành xác định lộ trình và bắt đầu di chuyển hướng tới đích. Đường đi di chuyển của Robot ngắn nhất giữa điểm đầu và đích, tránh các chướng ngại vật mô tả như hình 14.



Hình 14- Lộ trình Robot

3.2. Đánh giá chương trình và kết quả thảo luận

- Tính đúng đắn của thuật toán trong thực nghiệm

Với thực nghiệm vết dầu loang ta có bảng chi phí này là nếu d(x,y) ≠ ∞, thì d(x,y) chính là chi phí nhỏ nhất để di chuyển đến B. Nếu d(x,y)=∞ thì không tồn tại đường đi từ (x,y) đến B. Trước khi chạy chương trình, quá trình tìm kiếm chúng ta sẽ kiểm tra nếu d(x,y)=∞ thì thông báo không tồn tại đường đi, ngược lại thì tồn tại đường đi và bắt đầu tìm kiếm.

Khi chạy chương trình thực nghiệm, với mỗi bước đi ta sẽ quét vết dầu loang quanh khu vực và xác định vật cản.

+ Nếu không tồn tại vật cản thì robot sẽ đi theo lộ trình ban đầu và lộ trình này cũng chính là lộ trình nhanh nhất.

+ Nếu tồn tại vật cản nhưng không xảy ra đụng độ thì lộ trình vẫn được xác định như ban đầu và cũng chính là lộ trình nhanh nhất.

+ Nếu tồn tại vật cản và xảy ra đụng độ thì chúng ta có 2 sự lựa chọn, với mỗi sự lựa chọn đều tình được chi phí đụng độ.

$\varphi_1 = d(x,y) + \Delta c$

$\varphi_2 = Min\big(d(x + Hx[i], y + Hy[i])\big), i = 0..3$

Khi chọn các giá trị đó so sánh với nhau, và lấy giá trị chi phí nhỏ nhất để thực hiện lộ trình. Do vậy trong trường hợp này cũng là lộ trình tối ưu nhất.

Δc là chi phí đụng độ hay là thời gian mà robot phải chờ đợi vật cản. Thời gian Δc có thể được ước tính, nhưng trong chương trình đã mặc định Δc = 6(s). Thời gian Robot có thể hiệu chỉnh tùy thuộc vào các nodes di chuyển trong đồ thị mà Δc nhận các giá trị thích hợp.

Như vậy, kết quả thực nghiệm đã cho thấy giải quyết vấn đề tối ưu về lộ trình, đảm bảo tìm được một lộ trình nhanh nhất, với ít chi phí nhất, và không xảy ra đụng độ với bất kỳ vật cản nào trong quá trình di chuyển.

- Tính tối ưu của thuật toán

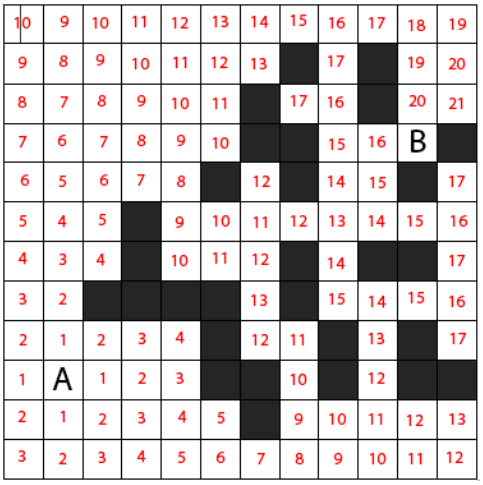
Trong quá trình thực nghiệm, nhóm nghiên cứu tiến hành so sánh giữa thuật toán vết dầu loang truyền thống và vết dầu loang cải tiến trong bài nghiên cứu này đối với trường hợp không gặp vật cản. Thời gian chi phí đường đi của thuật toán cải tiến nhanh hơn thuật toán vết dầu loang truyền thống.

Trong trường hợp gặp các chướng ngại vật, kết quả thực nghiệm cho thấy thời gian thực hiện xét vết dầu loang truyền thống và thuật toán đề xuất gần tương đồng như nhau, nghĩa là số nút phải duyệt của vết dầu loang truyền thống(xuất phát từ A) chính bằng số nút phía duyệt của vết dầu loang cải tiến(xuất phát từ B). Nếu ta gọi số nút phải duyệt của vết dầu loang truyền thống là NumTT và số nút phải duyệt của vết dầu loang cải tiến là NumHust thì: NumTT = NumHust(3).

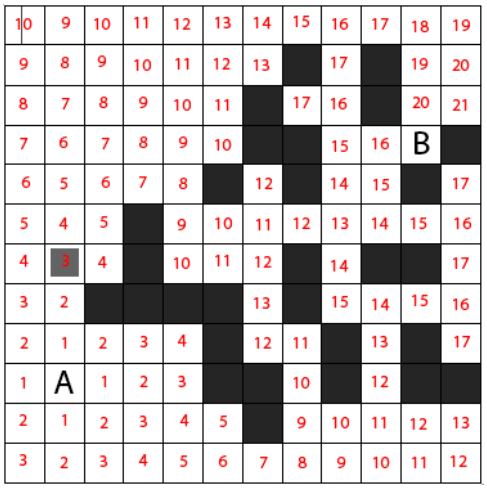
Chúng ta suy ra được: thời gian thực hiện quá trình vết dầu loang trong 2 thuật toán là như nhau:

$T_{Numtt} = T_{NumHust}$ (4).

+ Thời gian tính toán xử lý khi gặp vật cản mô tả như hình 15 và 16:



Hình 15: Bảng chi phí xuất phát từ A của vết dầu loang truyền thống.



Hình 16: Quá trình xảy ra đụng độ của vết dầu loang truyền thống.

Trong trường hợp áp dụng thuật toán vết dầu loang truyền thống cho quá trình xử lý đụng độ thì cần phải xử lý:

Từ vị trí (7,2) xác định đi tiếp đường nào để tối ưu như nào? Đây là nhược điểm của thuật toán vết dầu loàn truyền thống. Từ vị trí (7,2) (màu xám) chúng ta vẫn có thể rẽ theo 4 nhánh, nhưng nhánh nào là nhánh tối ưu, nhánh đó có thực sự đến được đích hay sẽ bị bước vào ngõ cụt? Đây là những vấn đề mà thuật toán truyền thống mắc phải.

Nếu như từ vị trí (2,7) ta lại sử dụng vết dầu loang cho chính điểm đó thì chi phí cho mỗi lần xét là $O(n^2)$, do đó việc chi phí thời gian lớn.

Đối với thuật toán vết dầu loang cải tiến thì gặp trường hợp nêu trên thời gian xử lý coi như là bằng 0. Vì nó không phải duyệt bất cứ nút nào ngoài chính 4 nút lân cận nút đó (có thể bỏ qua nút vừa đi để còn lại 3 nút).

Nếu gọi thời gian xử lý khi gặp sự kiện của thuật toán vết dầu loang truyền thống là $T_{CostTT} = O(n^2)$, còn thời gian xử lý của thuật toán vết dầu loang cải tiến là $T_{CostHust} \approx 0$(quá nhỏ so với tốc độ xử lý của máy tính cỡ $1/10^6$).

Ta thấy: $T_{CostHust} \ll T_{CostTT} = O(n^2)$ (5)

Từ (4) và (5) ta có:

$T_{CostHust} + T_{NumHust} \ll T_{CostTT} + T_{NumTT}$ (6)

Do đó thuật toán vết dầu loang HUST ngoài việc ứng dụng để giải quyết bài toán môi trường động, vật cản di chuyển mà vết dầu loang truyên thống chưa giải quyết được thì nó còn cải thiện tốc độ xử lý rất lớn.

4. KẾT LUẬN

Kết quả thực nghiệm cho thấy được thuật toán đã giải quyết được bài toán tìm kiếm đường đi của Robot với rất nhiều ràng buộc xử lý các tình huống phức tạp mà Robot gặp vật cản . So với thuật toán vết dầu loang truyền thống thì nghiên cứu này đã giải quyết tốt vấn đề xử lý vật cản, tối ưu đường đi và tối ưu thời gian xử lý tìm kiếm của thuật toán.

Nghiên cứu này đã giải quyết được vấn đề đề tìm kiếm đường đi cho robot trong môi trường biến động. Thuật toán vết dầu loang cải tiến này đảm bảo tìm ra được một lộ trình tối ưu và không xảy ra bất cứ đụng độ nào trong quá trình di chuyển. Trong thời gian tới, nhóm nghiên cứu sẽ tiến hành thực nghiệm áp dụng thuật toán vào thực tế và cài đặt trên các thiết bị ví dụ như Robot lau nhà, Robot cứu hộ, … với các thiết bị như bộ định vị GPS nhằm xác định bản đồ, các Camera, hoặc cảm biến để quét vật cản. Các thiết bị này đều là những thiết bị chuyên dụng, dễ dàng sử dụng và cài đặt. Chúng tôi mong rằng thuật toán này sẽ sớm được đưa vào thực tế nhằm phát triển xã hội và nâng cao cuộc sống của con người.

5. LỜI CẢM ƠN.

Công trình nghiên cứu theo nội dung nghiên cứu thuộc đề tài mã số B2015-01-91 về I Clause Robot tìm đường đi tối ưu bao phủ.

6. TÀI LIỆU THAM KHẢO.

[1] Yuan-Qing Qin, De-Bao Sun, Ning Li, Yi-Gang Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator". In Proceedings of the Third International Conference on Machine learning and Cybernetics, Shanghai, pp. 2473-2478, 2004.

[2] Ellips Masehian, Davoud Sedighizadeh, "Classic and heuristic approaches in robot motion planning - a chronological review". In Proceedings of world academy of science, engineering and technology, vol. 23, pp. 101-106, 2007.

[3] Ahmed Elshamli, Hussein A. Abdullah, Shawki Areibi, "Genetic algorithm for dynamic path planning". In Proceeding of IEEE CCECE 2004, pp. 677-680, 2004.

[4] Milos Seda, "Roadmap method vs. cell decomposition in robot motion planning". In Proceedings of 6th WSEAS international conf on signal processing, Robotics and automation, Grees, pp. 127-132, 2007.

[5] Jean-Claude Latombe, "Robot motion planning". In Kluwer International Series in Engineering and Computer Science, London, 1991.

[6] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza, "Introduction to autonomous mobile robots". In MIT-Press, pp. 1-12, 2004.

[7] Gordon Cheng , Alexander Zelinsky, "A physically grounded search in a behavior based robot". In Proc. Eighth Australian Joint Conference on Artificial Intelligence, pp. 547-554, 1995.

[8] Lê Minh Hoàng, Phương pháp tìm kiếm theo chiều rộng – Giải thuật và lập trình, Trường Đại Học Sư Phạm Hà Nội, 1999-2002.