ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Anh Chiến

ỨNG DỤNG HỌC TĂNG CƯỜNG SÂU CHO HOẠT ĐỘNG CỦA HỆ THỐNG ĐA ROBOT TRONG NHÀ MÁY

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Kỹ thuật Robot

HÀ NỘI - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Anh Chiến

ỨNG DỤNG HỌC TĂNG CƯỜNG SÂU CHO HOẠT ĐỘNG CỦA HỆ THỐNG ĐA ROBOT TRONG NHÀ MÁY

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Kỹ thuật Robot

Cán bộ hướng dẫn: TS. Phạm Duy Hưng

HÀ NỘI - 2024

Lời cảm ơn

Trước tiên, tôi xin gửi lời tri ân sâu sắc đến các thầy, cô đang công tác và giảng dạy tại Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Chính sự tận tâm trong giảng dạy, sự chỉ bảo nhiệt tình và những điều kiện thuận lợi mà các thầy, cô đã tạo ra đã giúp tôi có cơ hội học tập, nghiên cứu và hoàn thiện bản thân trong suốt bốn năm học vừa qua.

Đặc biệt, tôi xin gửi lời cảm ơn chân thành nhất tới TS. Phạm Duy Hưng, người đã tận tình hướng dẫn và đưa ra những ý kiến đóng góp quý báu, giúp tôi hoàn thành đồ án này. Sự chỉ bảo tận tụy của thầy đã giúp tôi vượt qua nhiều khó khăn và hoàn thiện công việc một cách tốt nhất.

Tôi cũng xin bày tỏ lòng biết ơn sâu sắc đến gia đình, những người luôn là chỗ dựa tinh thần vững chắc, bạn bè thân thiết và tập thể lớp K65R, những người đã luôn động viên, khích lệ tôi trong suốt quá trình học tập và thực hiện đồ án.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày ... tháng ... năm 2024

Sinh viên

Lê Anh Chiến

Lời cam đoan

Tôi xin cam đoan rằng tất cả các kết quả được trình bày trong đồ án này đều là sản phẩm do chính tôi thực hiện, dưới sự hướng dẫn tận tình của TS. Phạm Duy Hưng.

Toàn bộ các tài liệu tham khảo liên quan đến đồ án tốt nghiệp đã được tôi liệt kê đầy đủ và trích dẫn rõ ràng trong danh mục tài liệu tham khảo. Tôi khẳng định rằng đồ án này không sao chép bất kỳ tài liệu hay công trình nghiên cứu nào từ bất kỳ nguồn nào mà không được ghi rõ trong danh sách tài liệu tham khảo.

Nếu có bất kỳ nội dung nào trong đồ án không đúng với những điều đã nêu trên, tôi xin hoàn toàn chịu trách nhiệm theo quy định của Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội.

Hà Nội, ngày ... tháng ... năm 2024

Sinh viên

Lê Anh Chiến

Tóm tắt

Tóm tắt: Trong bối cảnh cách mạng công nghiệp 4.0, sự phát triển vượt bậc của công nghệ đã thúc đẩy việc ứng dụng các hệ thống tự động hóa trong sản xuất, đặc biệt là hệ thống đa robot trong nhà máy. Những hệ thống này không chỉ cải thiện hiệu suất sản xuất mà còn giảm thiểu chi phí và sự phụ thuộc vào nguồn nhân lực. Tuy nhiên, để tối ưu hóa hoạt động của các hệ thống đa robot, việc áp dụng các thuật toán phức tạp là điều cần thiết, trong đó học tăng cường sâu (Deep Reinforcement Learning - DRL) nổi lên như một công cụ mạnh mẽ và hiệu quả. Đồ án này tập trung nghiên cứu và phát triển một phương pháp tối ưu hóa phân nhiệm và điều phối robot dựa trên học tăng cường sâu, kết hợp với mạng nơ-ron đồ thị (Graph Neural Networks - GNN) và cơ chế chú ý (Attention Mechanism). Để hiện thực hóa mục tiêu, hệ thống nhà máy và hệ thống đa robot đã được mô hình hóa chi tiết. Các thuật toán quản lý trạng thái robot, phân nhiệm, và lên kế hoạch đường đi được thiết kế một cách toàn diện. Bên cạnh đó, đồ án đã đề xuất một chính sách phân nhiệm mới nhằm khắc phục các hạn chế của phương pháp RTAW và tiến hành so sánh hiệu suất với phương pháp RTAW cùng phương pháp đấu giá dựa trên khoảng cách. Kết quả thử nghiệm cho thấy phương pháp phân nhiệm đề xuất đạt hiệu suất tương đương với phương pháp RTAW và vượt trội hơn phương pháp đấu giá dựa trên khoảng cách. Đặc biệt, khi số lượng robot và nhiệm vụ tăng lên, phương pháp đề xuất đã cải thiện hiệu suất phân nhiệm so với RTAW từ 2% đến 5%, minh chứng tính hiệu quả và tiềm năng áp dụng trong thực tế của hệ thống.

Từ khóa: *Phân nhiệm đa robot, Học tăng cường sâu, Mạng nơ-ron đồ thị, Cơ chế chú ý*

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

Danh mục các từ viết tắt

STT	Từ viết tắt	Cụm từ đầy đủ	Cụm từ tiếng Việt
1	DRL	Deep Reinforcement Learnin	Học tăng cường sâu
2	RL	Reinforcement Learning	Học tăng cường
3	MDP	Markov Decision Process	Quá trình ra quyết định Markov
5	POMDP	Partially Observable Markov Decision Process	Quá trình quyết định Markov có thể quan sát được một phần
6	DNN	Deep Neural Network	Mạng nơ-ron sâu
8	DQN	Deep Q-Network	Mạng Q sâu
10	PPO	Proximal policy optimization	Tối ưu hóa chính sách gần
11	A2C	Advantage Actor-Critic	Phương pháp diễn viên-phê bình lợi thế
16	GNN	Graph Neural Network	Mạng nơ-ron đồ thị
17	GCN	Graph Convolutional Network	Mạng tích chập đồ thị
18	GAT	Graph Attention Network	Mạng chú ý đồ thị
21	RTAW	Reinforcement-based Task Assignment for Warehouse	Phân nhiệm trong kho dựa trên học tăng cường
22	FSM	Finite State Machine	Máy trạng thái hữu hạn

Chương 1

Giới thiệu chung

Chương này trình bày về bối cảnh và động lực của việc thực hiện đồ án. Sau đó, tổng quan các nghiên cứu liên quan được đưa ra. Phần cuối cùng là các nội dung thực hiện trong đồ án và bố cục của đồ án.

1.1 Bối cảnh và động lực

Trong thời đại công nghiệp 4.0, sự phát triển nhanh chóng của công nghệ đã tạo ra những thay đổi lớn trong nhiều lĩnh vực, đặc biệt là trong logistics và quản lý nhà máy. Các nhà máy tự động hóa đang dần trở thành xu hướng không thể thiếu, giúp nâng cao hiệu quả hoạt động, giảm chi phí và tăng độ chính xác trong quản lý hàng hóa. Một trong những công nghệ tiên tiến đang được ứng dụng rộng rãi tại các nhà máy là hệ thống đa robot.

Những robot này không chỉ giúp cải thiện hiệu suất trong việc di chuyển, sắp xếp và quản lý hàng hóa mà còn giúp giảm bớt sự phụ thuộc vào nhân lực và tối ưu hóa không gian nhà máy. Tuy nhiên, để hệ thống robot hoạt động hiệu quả, cần đến các thuật toán và chiến lược điều khiển phức tạp. Trong bối cảnh đó, học tăng cường sâu (Deep Reinforcement Learning - DRL) nổi lên như một công cụ mạnh mẽ giúp giải quyết các vấn đề này.

Học tăng cường sâu là sự kết hợp giữa học tăng cường và học sâu, cho phép hệ thống học hỏi từ môi trường và đưa ra các quyết định dựa trên kinh nghiệm thực tế. Điều này đặc biệt hữu ích khi điều khiển các hệ thống phức tạp như hệ thống đa robot, nơi các quyết định phải được thực hiện nhanh chóng và thích ứng với môi trường thay đổi liên tục.

Mục tiêu chính của nghiên cứu này xuất phát từ nhu cầu ngày càng cao về hiệu quả và độ chính xác trong quản lý nhà máy. Trong các nhà máy truyền thống, việc quản lý và vận hành thường gặp nhiều khó khăn do quy trình phức tạp và phụ thuộc nhiều vào lao động thủ công. Sử dụng hệ thống robot không chỉ giải quyết được những vấn đề này mà còn mở ra những cơ hội mới để tối ưu hóa hoạt động

nhà máy.

Học tăng cường sâu có tiềm năng lớn trong việc cải thiện hiệu suất của các hệ thống robot. Bằng cách học từ môi trường và liên tục tối ưu hóa hành động, các thuật toán học tăng cường sâu có thể giúp hệ thống robot tự động điều chỉnh và nâng cao hiệu quả vận hành. Điều này không chỉ giúp tiết kiệm chi phí mà còn nâng cao độ chính xác và hiệu quả trong quản lý hàng hóa.

Ngoài ra, việc nghiên cứu và phát triển ứng dụng học tăng cường sâu cho hệ thống robot trong nhà máy còn mang lại những đóng góp quan trọng cho lĩnh vực trí tuệ nhân tạo và robotics. Những kết quả thu được từ nghiên cứu này không chỉ ứng dụng được trong ngành logistics mà còn mở rộng ra nhiều lĩnh vực khác, từ sản xuất công nghiệp đến chăm sóc sức khỏe.

Vì vậy, nghiên cứu và phát triển "Ứng dụng học tăng cường sâu cho hoạt động của hệ thống robot trong nhà máy"không chỉ mang lại giá trị thực tiễn cao mà còn đóng góp đáng kể vào sự phát triển công nghệ và kinh tế.

1.2 Tổng quan các nghiên cứu liên quan

Hệ thống đa robot trong môi trường nhà máy đối mặt với hai thách thức cốt lõi là bài toán phân nhiệm và bài toán lên kế hoạch đường đi. Trong đó, phân nhiệm chịu trách nhiệm chỉ định tác vụ cho từng robot nhằm tối ưu hóa hiệu suất hệ thống, bao gồm các mục tiêu như giảm thiểu thời gian thực hiện, giảm thời gian không hoạt động, tăng số lượng nhiệm vụ hoàn thành, và đảm bảo độ tin cậy của quá trình phân nhiệm. Vì hiệu suất tối ưu là một khái niệm trừu tượng, các hệ thống thường sử dụng hàm tiện ích để định lượng chi phí và giá trị trong quy trình phân nhiệm. Ngược lại, bài toán lên kế hoạch đường đi tập trung vào việc xác định các tuyến đường tối ưu, đảm bảo tránh xung đột giữa các robot, cải thiện năng suất và duy trì sự ổn định toàn hệ thống.

Trong lĩnh vực phân nhiệm, các phương pháp phổ biến có thể chia thành hai nhóm chính. Nhóm thứ nhất là các phương pháp dựa trên thị trường (market-based), trong đó sử dụng các cơ chế đấu giá hoặc đàm phán để phân bổ nhiệm vụ cho robot. Mỗi robot hoặc nhóm robot tham gia đấu giá dựa trên một hàm chi phí thiết kế trước, nhằm tối ưu hóa hiệu quả toàn hệ thống [1, 2]. Nhóm thứ hai là các phương pháp tối ưu hóa, dựa trên các thuật toán heuristic, metaheuristic

hoặc các kỹ thuật tối ưu hóa khác để tìm kiếm các giải pháp hiệu quả, đặc biệt phù hợp với các hệ thống có quy mô lớn và yêu cầu tính toán cao [3]. Một phương pháp nổi bật trong nhóm này là RTAW áp dụng học tăng cường để ước lượng giá trị hành động trong môi trường kho hàng. Phương pháp này không chỉ tận dụng kinh nghiệm trong các vòng lặp trước để liên tục cải thiện chiến lược phân nhiệm, mà còn đặc biệt hiệu quả trong các môi trường động và phức tạp [4]. Ngoài ra, các phương pháp như đấu giá dựa trên khoảng cách mang lại lợi ích đơn giản nhưng hiệu quả trong môi trường có đặc tính ổn định, nơi khoảng cách đóng vai trò là tiêu chí quan trọng để quyết định phân nhiệm.

Bài toán lên kế hoạch đường đi cũng đã thu hút sự quan tâm lớn từ cộng đồng nghiên cứu, đặc biệt với sự phát triển của học tăng cường sâu. Các nghiên cứu trong lĩnh vực này có thể chia thành hai hướng chính bao gồm có giao tiếp và không giao tiếp. Hướng không giao tiếp tập trung vào việc các robot tự lập kế hoạch dựa trên quan sát cá nhân và sử dụng các mô hình được huấn luyện trước. Ví dụ, PRIMAL sử dụng thuật toán A* để lập kế hoạch cục bộ kết hợp với ma trận ràng buộc xung đột để giải quyết các vấn đề về đường đi [5]. Mapper, một phương pháp khác, áp dụng mô hình đào tạo và thực thi phân tán, giúp các robot mô phỏng và học hỏi từ môi trường động để tối ưu hóa quá trình lập kế hoạch [6]. Tuy nhiên, các phương pháp không giao tiếp thường gặp hạn chế trong các môi trường đông đúc và động, nơi sự phối hợp giữa các robot là yếu tố then chốt.

Đối với các phương pháp có giao tiếp, nhiều nghiên cứu đã tập trung vào việc phát triển giao thức truyền thông giữa các robot nhằm tăng cường sự phối hợp. CommNet là một ví dụ điển hình, với khả năng hợp nhất và truyền tải thông tin liên tục, giúp robot có được cái nhìn tổng quan về trạng thái hệ thống [7]. Trong khi đó, IC3Net sử dụng cơ chế mạng LSTM để kiểm soát giao tiếp trong các tình huống cạnh tranh giữa các robot, tối ưu hóa quyết định trong các hệ thống phức tạp [8]. Một phương pháp khác là DIAL, cho phép truyền thông tin hai chiều và hỗ trợ việc truyền ngược gradient toàn cục, từ đó cải thiện đáng kể hiệu quả của các hệ thống đa robot [9].

Mặc dù đã có nhiều tiến bộ trong cả hai hướng nghiên cứu này, việc áp dụng chúng vào môi trường nhà máy vẫn đặt ra nhiều thách thức. Các môi trường nhà máy thường có không gian cố định, số lượng robot hạn chế và yêu cầu cao về độ ổn định, tính toán, và hiệu quả vận hành. Điều này đòi hỏi các thuật toán phân

nhiệm và lập kế hoạch đường đi không chỉ cần đạt được hiệu suất cao mà còn phải đảm bảo khả năng triển khai thực tế, đáp ứng các yêu cầu sản xuất hàng ngày.

1.3 Mục tiêu và Phương pháp nghiên cứu

1.3.1 Mục tiêu nghiên cứu

Mục tiêu của đồ án này là phát triển một hệ thống đa robot mô phỏng trong môi trường nhà máy, bao gồm đầy đủ các lớp làm việc và khối chức năng tương tự như hệ thống thực tế. Đặc biệt, đồ án tập trung vào việc đề xuất xây dựng một phương pháp phân nhiệm mới, ứng dụng học tăng cường sâu kết hợp với mạng nơ-ron đồ thị (Graph Neural Network - GNN). Phương pháp phân nhiệm đề xuất được thiết kế nhằm đảm bảo tính linh hoạt và khả năng thích ứng cao với các thay đổi về số lượng robot, số lượng nhiệm vụ và số lượng điểm nhiệm vụ. Mục tiêu cuối cùng là tối ưu hóa chi phí thực hiện nhiệm vụ trong hệ thống đa robot, với chi phí được đo lường thông qua thời gian thực hiện nhiệm vụ trung bình.

Đối tượng nghiên cứu của đồ án là các nhà máy sử dụng hệ thống đa robot để thực hiện các nhiệm vụ trong không gian nhà kho được mô phỏng, với diện tích và bố trí kiến trúc linh hoạt. Hệ thống hoạt động dựa trên một máy chủ trung tâm, nơi thực hiện phân nhiệm, lập kế hoạch đường đi và phát lệnh di chuyển hoặc dừng cho các robot. Các robot trong hệ thống di chuyển theo lộ trình được chỉ định và hoàn thành nhiệm vụ đã phân công, từ đó đảm bảo sự hoạt động liên tục của hệ thống.

Trong nghiên cứu này, mô hình năng lượng của robot được bỏ qua. Vị trí và hướng của robot được tính toán trực tiếp mà không sử dụng các bộ lọc như trong các hệ thống thực tế. Hệ thống luôn có nhiệm vụ cần thực hiện tại mọi thời điểm, và các nhiệm vụ được giới hạn ở dạng hai điểm (điểm bắt đầu và điểm kết thúc). Các nhiệm vụ liên quan đến trạm sạc hoặc điểm chờ không được xem xét, vì giả định hệ thống luôn có nhiệm vụ cần thực hiện. Hệ thống đảm bảo rằng các robot luôn hoàn thành nhiệm vụ được giao, và kết nối giữa robot và máy chủ luôn ổn định, không có độ trễ. Các đoạn đường mà robot sử dụng để di chuyển trong môi trường chỉ theo một chiều để đảm bảo không xảy ra trường hợp hai robot đối đầu nhau trên cùng một đoạn đường, giảm đi những trường hợp va chạm phức tạp.

Các giải pháp được đề xuất trong đồ án sẽ được thử nghiệm và đánh giá thông qua mô phỏng máy tính hoặc một số mô hình thực nghiệm giả định. Hệ thống không được thử nghiệm trực tiếp trên các robot thực tế, mà chỉ tập trung vào các điều kiện giả lập nhằm kiểm tra tính khả thi và hiệu quả của các thuật toán đề xuất.

1.3.2 Phương pháp nghiên cứu

Đồ án sử dụng ngôn ngữ lập trình C++ để mô phỏng hệ thống đa robot trong môi trường nhà máy. Hệ thống được mô hình hóa với hai lớp chính, bao gồm lớp vật lý và lớp điều khiển. Lớp vật lý bao gồm hệ thống đa robot và môi trường nhà máy, trong khi lớp điều khiển bao gồm năm khối chức năng: khối quản lý trạng thái robot, khối quản lý nhiệm vụ, khối phân nhiệm, khối lập kế hoạch đường đi và khối điều khiển robot.

Thuật toán cho khối quản lý trạng thái robot được xây dựng dựa trên mô hình máy trạng thái hữu hạn (FSM). Khối lập kế hoạch đường đi được phát triển dựa trên thuật toán A*, một thuật toán phổ biến trong việc tìm kiếm đường đi tối ưu trong không gian 2D hoặc 3D. Đặc biệt, khối phân nhiệm được phát triển dựa trên học tăng cường sâu kết hợp với mạng nơ-ron đồ thị, sử dụng phương pháp phân nhiệm RTAW [4]. Kiến trúc của mạng nơ-ron trong khối phân nhiệm sử dụng mạng chú ý đồ thị (Graph Attention Network - GAT) kết hợp với cơ chế chú ý (Attention Mechanism), cho phép hệ thống học các chiến lược phân nhiệm tối ưu trong môi trường đa robot.

Mô hình học sâu được đào tạo bằng phương pháp Proximal Policy Optimization (PPO) [10], một phương pháp hiệu quả trong học tăng cường, kết hợp với thuật toán tối ưu Adam [11]. Đầu vào cho quá trình phân nhiệm là một tập nhiệm vụ ngẫu nhiên, được sinh ra bởi bộ sinh số ngẫu nhiên, giúp mô phỏng các tình huống phân nhiệm đa dạng và đảm bảo tính tổng quát của mô hình trong việc xử lý các nhiệm vụ trong môi trường nhà máy.

1.4 Bố cục của đồ án

Bố cục của đồ án được trình bày thành 4 chương. Mỗi chương trình bày và thảo luận các vấn đề khác nhau xoay quanh đồ án và được tóm tắt như sau:

Chương 1: Giới thiệu chung. Trình bày bối cảnh và động lực để thực hiện đồ án về hệ thống đa robot trong nhà máy. Bên cạnh đó, các nghiên cứu liên quan cho bài toán phân nhiệm và lên kế hoạch đường đi cho hệ thống đa robot được trình bày một cách tổng quan.

Chương 2: Cơ sở lý thuyết. Giới thiệu về các cơ sở lý thuyết được lấy làm cơ sở cho việc xây dựng chính sách, thuật toán cho đồ án bao gồm tổng quan về học tăng cường sâu, các thuật toán học tăng cường sâu điển hình, mạng nơ-ron đồ thị (graph neuron network), cơ chế chú ý (attention mechanism) .

Chương 3: Mô hình hóa hệ thống và xây dựng chính sách phân nhiệm. Mô hình hóa chi tiết hệ thống đa robot trong nhà kho. Trình bày chi tiết việc xây dựng thuật toán, chính sách cho các khối trong hệ thống đa robot trong nhà kho.

Chương 4: Kết quả Trình bày quá trình thí nghiệm và thu thập các kết quả của chính sách phân nhiệm cho hệ thống đa robot được đề xuất trong quá trình thực hiện. Đánh giá kết quả của chính sách phân nhiệm dựa trên các thước đo đã thiết kế từ trước.

Kết luận. Tổng kết về các công việc đã thực hiện trong đồ án và các kết quả đã đạt được. Đồng thời trình bày về những hướng phát triển trong tương lai cho đồ án.

Chương 2

Cơ sở lý thuyết

Chương này trình bày các cơ sở lý thuyết liên quan sử dụng trong đồ án. Nội dung các mục 2.1, 2.2, 2.3 được tham khảo từ nghiên cứu tổng quan về những thách thức và cơ hội khi kết hợp mạng nơ-ron đồ thị vào học tăng cường sâu của tác giả Sai Munikoti và các cộng sự trong [12]; mục 2.5 được tham khảo dựa trên hai nghiên cứu của tác giả Dzmitry Bahdanau cùng cộng sự trong [13] và Ashish Vaswani cùng cộng sự trong [14].

2.1 Tổng quan về học tăng cường sâu

Học tăng cường sâu (Deep Reinforcement Learning - DRL) là một phương pháp tiếp cận trong lĩnh vực học máy, kết hợp giữa học tăng cường (Reinforcement Learning - RL) và mạng nơ-ron sâu (Deep Neural Networks - DNN). Đây là một trong những lĩnh vực quan trọng của trí tuệ nhân tạo, nơi tác nhân (agent) học cách đưa ra các quyết định tối ưu trong các môi trường phức tạp thông qua quá trình tương tác liên tục với môi trường mà không cần biết trước tất cả các thông tin về môi trường đó.

Học tăng cường là một kỹ thuật học máy, trong đó tác nhân học cách tối đa hóa phần thưởng tích lũy bằng cách thực hiện các hành động dựa trên trạng thái hiện tại của môi trường. Mô hình ra quyết định Markov (Markov Decision Process - MDP) thường được sử dụng để mô tả quá trình này và được định nghĩa bởi $(\mathcal{S}, \mathcal{A}, p, r)$ trong đó $\mathcal{S}$ là không gian trạng thái, $\mathcal{A}$ là không gian hành động, p là xác suất chuyển trạng thái từ trạng thái $s_t \in \mathcal{S}$ tại thời điểm t sang trạng thái $s_{t+1} \in \mathcal{S}$ tại thời điểm $t+1$ và r là phần thưởng mà tác nhân thu được sau khi thực hiện hành động $a \in \mathcal{A}$. Mục tiêu của tác nhân là tìm ra chính sách tối ưu π^*, tức là cách lựa chọn hành động tốt nhất cho mỗi trạng thái nhằm tối đa hóa phần thưởng theo thời gian. Khi lựa chọn một hành động, tác nhân sẽ phải lựa chọn giữa thực hiện hành động dựa trên những kinh nghiệm quá khứ (khai thác - exploitation) và thu thập những kinh nghiệm mới (khám phá - exploration) để tạo

ra những hành động tốt hơn trong tương lai [15].

Bên cạnh MDP, mô hình ra quyết định Markov dựa trên quan sát một phần (POMDP) [15] được đưa ra để giải quyết các tình huống thực tế nơi mà các tác nhân không thể có đầy đủ sự hiểu viết về môi trường, chỉ có thể đưa ra lựa chọn hành động từ một phần hiểu biết về môi trường (có thể là rất nhỏ so với toàn bộ trạng thái của môi trường thực tế). POMDP được định nghĩa giống như MDP với các thành bổ sung như sau $(\mathcal{S}, a, \Omega, \mathcal{T}, p, \mathcal{O}, r, b_o)$. Các thành phần mới bao gồm $\mathcal{O}$ là quan sát một phần trạng thái của tác nhân, $\mathcal{T}$ biểu thị thời gian, $\mathcal{O}$ biểu diễn xác suất của $\mathcal{O}$ và b_o là phân phối xác suất của các trạng thái. Tại mỗi thời điểm, tác nhân thực hiện hành động thu được một quan sát $o \in \Omega$ phụ thuộc vào trạng thái mới của môi trường, hành động vừa thực hiện và xác suất $\mathcal{O}(o|s', a)$

Trong các bài toán có không gian trạng thái lớn và phức tạp, việc sử dụng bảng tra cứu để lưu trữ giá trị của các trạng thái và hành động là không khả thi. Để khắc phục điều này, DNN được sử dụng để xấp xỉ các hàm giá trị hoặc chính sách, giúp tác nhân có thể học được các quyết định tối ưu ngay cả khi không có đầy đủ thông tin về môi trường. Mạng nơ-ron sâu đảm bảo tính mở rộng và hiệu quả tính toán trong việc xử lý các bài toán phức tạp.

DRL mang lại nhiều ưu điểm vượt trội so với các phương pháp truyền thống và có khả năng xử lý hiệu quả các môi trường có không gian trạng thái lớn và phức tạp, cho phép các tác nhân học và đưa ra quyết định gần tối ưu trong các môi trường động đòi hỏi khả năng ra quyết định thời gian thực và không yêu cầu kiến thức trước về môi trường mà dựa trên các tương tác thực tế để học chính sách hành động tốt nhất. Học tăng cường sâu đã được áp dụng thành công trong nhiều lĩnh vực, bao gồm trò chơi, hệ thống khuyến nghị, điều khiển robot, và các hệ thống tự động phức tạp [12]. DRL đặc biệt hữu ích trong các môi trường có sự không chắc chắn và khó mô hình hóa một cách tường minh, như trong hệ thống đa robot hoạt động trong nhà máy.

2.2 Các thuật toán học tăng cường sâu điển hình

Các thuật toán học tăng cường sâu (DRL) có thể được phân loại theo nhiều cách khác nhau, tùy thuộc vào cơ chế hoạt động và phương pháp học của chúng. Các cách phân loại phổ biến hiện này bao gồm thuật toán không dựa trên mô

hình (model-free) với dựa trên mô hình (model based), thuật toán dựa trên giá trị (value based) với dựa trên chính sách (policy based), học ngoại tuyến (offline learning) với học trực tuyến (online learning). Trong phần này, đồ án sẽ trình bày ba loại thuật toán học tăng cường sâu bao gồm học tăng cường sâu dựa trên giá trị, học tăng cường sâu dựa trên chính sách và học tăng cường sâu diễn viên - nhà phê bình (actor-critic).

2.2.1 Học tăng cường sâu dựa trên giá trị

Phương pháp dựa trên giá trị có mục đích học giá trị của trạng thái s hoặc cặp trạng thái hành động (s, a) và sau đó hành động dựa trên giá trị đó. Hàm giá trị của trạng thái hành động $Q_\pi(s, a)$ được thể hiện trong phương trình 2.1, là giá trị kỳ vọng tại trạng thái s, thực hiện hành a theo một chính sách π. Deep Q learning (DQN) là một trong những thuật toán được sử dụng rộng rãi trong phương pháp này [16]. Q-learning cho phép các tác nhân chọn một hành động $a \in \mathcal{A}$ với hàm giá trị Q từ trạng thái $s \in \mathcal{S}$ dựa trên một mạng nơ-ron sâu ánh xạ từ không gian trạng thái-hành động sang giá trị Q. Các tham số của mạng nơ-ron trong DQN được cập nhật mỗi bước thời gian như trong phương trình 2.1, trong đó r là phần thưởng thu được và α là tốc độ học có giá trị từ 0 đến 1. DQN là một thuật toán học ngoại tuyến, trong đó một chính sách mục tiêu được sử dụng để thực hiện hành động tại trạng thái hiện tại $s_t$và một chính sách khác được sử dụng để chọn hành động tại trạng thái tiếp theo s_{t+1}.

$$\begin{aligned} Q_\pi(s,a) &= E_\pi(r_t|s_t,a_t) = E_\pi\left(\sum_{k=0}^{\infty}\gamma_k r_{t+k}|s_t,a_t\right) \\ &= (1-\alpha)Q(s,a) + \alpha\left[r + \gamma\max_{a'\in\mathcal{A}} Q(s_{t+1},a_{t+1})\right] \end{aligned} \tag{2.1}$$

Đặc trưng chính của quá trình đào tạo DQN là bộ đệm phát lại (relay buffer), lưu trữ thông tin (s_t, a_t, r_t, s_{t+1}) tại mỗi bước trong quá trình đào tạo. Trong DQN, mạng nơ-ron được đào tạo bằng cách sử dụng một lô nhỏ (minibatch) các mẫu ngẫu nhiên được lấy từ trong bộ đệm phát lại, đảm bảo về mặt lấy mẫu, phương sai thấp và phạm vi học lớn. Với mỗi mẫu, trạng thái (đầu vào) được truyền vào mạng nơ-ron hiện tại để tính toán đầu ra $\hat{Q}(s, a; \theta)$. Giá trị mục Q tương ứng với

phương trình tối ưu Bellman trong 2.1 và được tối thiểu hóa theo hàm mất mát [15]:

$$L(\theta) = E\left[r + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s, a; \theta)\right] \tag{2.2}$$

DQN có nhiều biến thể cải thiện thiết kế hiện tại bao gồm double DQN và dueling DQN. Toán tử max trong phương trình cập nhật lựa chọn và đánh giá một hành động đều sử dụng cùng một hàm Q. Kết quả là DQN ước tính quá mức hàm giá trị. Double DQN giải quyết vấn đề này bằng cách sử dụng hai mạng học sâu phân biệt, một cho lựa chọn hành động và một cho đánh giá hành động [17]. Tương tự, mạng dueling Q xấp xỉ hàm Q bằng cách tách rời hàm giá trị và hàm ưu thế (advantage function) [18].

2.2.2 Học tăng cường sâu dựa trên chính sách

Học tăng cường sâu dựa trên chính sách có mục đích là học chính sách một cách trực tiếp thông qua các kinh nghiệm trong quá trình đào tạo. Chính sách π_θ được cập nhật liên tục bằng cách tối thiểu hóa giá trị kỳ vọng dựa trên phương pháp gradient được biệt đến là lý thuyết gradient chính sách [19]. Các thuật toán dựa trên chính sách đặc biết phù hợp với những vấn đề, bài toán có không gian hành động lớn hoặc không gian hành động liên tục. Chính sách có thể được viết như sau:

$$\pi_(a|s, \theta) = P(a|s, \theta) \tag{2.3}$$

biểu diễn xác suất thực hiện hành động a tại trạng thái s với một chính sách có bộ tham số θ. Hàm mục tiêu của việc tối ưu chính sách được định nghĩa như sau:

$$J(\theta) = v_{\pi_\theta}(s_0) \tag{2.4}$$

trong đó, $v_{\pi_\theta}(s_0)$ là giá trị đúng của hàm giá trị với chính sách $\pi(a|s, \theta)$ và s_0 là trạng thái bắt đầu. Việc tối đa hóa $J(\theta$ đồng nghĩa với việc tối đa hóa $v_{\pi_\theta}(s_0)$ tương đương với $\nabla J(\theta) = \nabla v_{\pi_\theta}(s_0)$. Theo lý thuyết gradient chính sách, ta có

$$J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta), \tag{2.5}$$

trong đó $mu(s)$ là phân phối theo π, $q(s, a)$ là hàm giá trị của hành động và $\nabla v_{\pi_\theta}(s_0)$ là gradient của π được cho bởi s và θ. Vì vậy, lý thuyết gọi $J(\theta)$ tỷ lệ thuận với

tổng của hàm q nhân với gradient của các chính sách cho tất cả các hành động có thể có tại các trạng thái. Nhưng để tính được gradient này, chúng ta cần tính được $\nabla\pi(a|s,\theta)$ theo công thức như sau:

$$\nabla_\theta \pi_\theta(s,a) = \pi(s,a)\nabla_\theta \log \pi_\theta(s,a) \tag{2.6}$$

với $\nabla_\theta \log \pi_\theta(s,a)$ là hàm chấm điểm (scoring function). Quá trình cập nhật bộ tham số của chính sách mục tiêu được tiến hành theo cách sau:

$$\Delta\theta = \alpha\nabla_\theta J(\theta). \tag{2.7}$$

Có nhiều phương pháp khác nhau để định nghĩa chính sách như softmax, Gaussian, v.v. [15]. Ba phương pháp dựa trên chính sách được sử dụng rộng rãi bao gồm REINFORCE, Tối ưu chính sách dựa trên vùng tin tưởng (Trust region policy optimization - TRPO) và tối ưu hóa chính sách gần (Proximal policy optimization - PPO). Trong REINFORCE việc cập nhật tham số tại một bước thời gian nhất định chỉ liên quan đến hành động được thực hiện từ trạng thái hiện tại – quá trình cập nhật dựa vào giá trị hồi quy ước tính bằng phương pháp Monte-Carlo sử dụng các mẫu từ tập hợp episode. Vì nó dựa vào giá trị hồi quy kỳ vọng từ bước thời gian hiện tại, nên phương pháp này chỉ hoạt động cho các tác vụ có tính chu kỳ (episodic task). So với REINFORCE, TRPO [20] thêm các ràng buộc độ phân kỳ KL để tạo vùng tin cậy cho quá trình tối ưu hóa. Điều này đảm bảo rằng chính sách mới không quá khác biệt so với chính sách cũ, hoặc có thể nói rằng chính sách mới nằm trong vùng tin cậy của chính sách cũ. Ràng buộc này nằm trong không gian chính sách thay vì không gian tham số. Ràng buộc KL tạo ra thêm chi phí dưới dạng các ràng buộc cứng cho quá trình tối ưu hóa. PPO có cách tiếp cận đơn giản hơn dựa vào hàm mục tiêu thay thế được cắt để giảm độ lệch giữa chính sách mới và chính sách cũ. Về cơ bản, nó định nghĩa tỷ lệ xác suất giữa chính sách mới và chính sách cũ, và đảm bảo tỷ lệ này nằm trong một khoảng nhỏ quanh giá trị 1. Hàm cắt (clip function) giới hạn tỷ lệ chính sách trong khoảng từ $1-\epsilon$ đến $1+\epsilon$, với ϵ là một siêu tham số. Hàm mục tiêu của PPO lấy giá trị nhỏ nhất giữa giá trị ban đầu và giá trị đã được cắt. Nó tương đối đơn giản trong việc triển khai và về mặt thực nghiệm, hoạt động ngang tầm với TRPO. Hơn nữa, các thuật toán này (REINFORCE, TRPO, v.v.) tận dụng các ước lượng giá trị để cập nhật tham số. Còn có một hướng nghiên cứu khác giả định sự tồn tại của thông tin mô hình và

sử dụng các kỹ thuật từ lập trình động và lập trình động gần đúng để phát triển các kỹ thuật học tăng cường có hỗ trợ mô hình, như chính sách chỉ dành cho tác tử hoặc bộ điều khiển trực tiếp [21]

2.2.3 Học tăng cường sâu diễn viên - nhà phê bình

Cả các thuật toán dựa trên giá trị và dựa trên chính sách đều có một số hạn chế. Các thuật toán dựa trên giá trị không hiệu quả trong không gian hành động có chiều cao, trong khi các thuật toán dựa trên chính sách có độ phương sai cao trong ước tính gradient. Để khắc phục những hạn chế này, một phương pháp actor-critic đã được đề xuất, kết hợp cả hai cách tiếp cận [22]. Về cơ bản, tác tử được huấn luyện với hai bộ ước lượng. Đầu tiên là hàm actor, kiểm soát hành vi của tác tử bằng cách học chính sách tối ưu, tức là cung cấp hành động tốt nhất a_t cho bất kỳ trạng thái đầu vào s_t. Thứ hai là hàm critic, đánh giá hành động bằng cách tính toán hàm giá trị.

Một số biến thể phổ biến của các thuật toán trong danh mục này sẽ được thảo luận tiếp theo. Advantage actor-critic (A2C) bao gồm hai DNN – một cho actor và một cho critic [23]. Thuật ngữ "Advantage" tương ứng với sai số chênh lệch thời gian (TD) hay nói cách khác là sai số dự đoán. Sai số này là sự chênh lệch giữa giá trị TD mục tiêu (giá trị dự đoán của tất cả các phần thưởng tương lai từ trạng thái hiện tại s) và giá trị của trạng thái hiện tại được đánh giá từ mạng critic. Bên cạnh A2C, asynchronous advantage actor-critic (A3C) thực hiện nhiều tác tử song song trên nhiều phiên bản của môi trường thay vì sử dụng bộ đệm phát lại như trong A2C [23]. Mặc dù A3C tiết kiệm bộ nhớ, các bản cập nhật của nó không tối ưu vì các tác tử khác nhau làm việc với các phiên bản tham số mô hình khác nhau. Deep deterministic policy gradient (DDPG) là một mở rộng của deterministic policy gradient (DPG), được thiết kế cho không gian hành động liên tục [24]. DPG định nghĩa chính sách là hàm $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$.Ở đây, thay vì lấy tích phân trên các hành động như trong chính sách ngẫu nhiên, chúng ta chỉ cần cộng qua không gian trạng thái vì hành động là xác định. DDPG sử dụng một hàm actor có tham số hóa với một hàm critic có tham số hóa, hàm này ước tính hàm giá trị bằng cách sử dụng các mẫu. Theo cách này, DDPG có thể giải quyết độ phương sai lớn trong các gradient chính sách của các phương pháp chỉ dùng actor.

2.3 Mạng nơ-ron đồ thị

Học máy với dữ liệu có cấu trúc đồ thị, như các đồ thị tri thức, mạng sinh học và mạng xã hội gần đây đã thu hút nhiều sự chú ý trong nghiên cứu. Việc biểu diễn dữ liệu dưới dạng đồ thị mang lại nhiều lợi ích, chẳng hạn như mô hình hóa các mối quan hệ một cách có hệ thống, đơn giản hóa việc biểu diễn các vấn đề phức tạp, v.v. Tuy nhiên, việc diễn giải và đánh giá dữ liệu có cấu trúc đồ thị bằng cách sử dụng các phương pháp học truyền thống dựa trên DNN là một thách thức. Các thao tác toán học cơ bản như phép tích chập khó thực hiện trên đồ thị do cấu trúc không đồng đều của chúng, kích thước không đều của các đỉnh không có thứ tự và thành phần lân cận động. Mạng nơ-ron đồ thị (GNN) giải quyết những hạn chế này bằng cách mở rộng các kỹ thuật DNN cho dữ liệu có cấu trúc đồ thị [25]. Kiến trúc GNN có thể đồng thời mô hình hóa cả thông tin cấu trúc và thuộc tính của đỉnh. Chúng mang lại cải thiện hiệu suất đáng kể cho các tác vụ xử lý đồ thị như phân loại đỉnh, dự đoán liên kết, phát hiện cộng đồng và phân loại đồ thị [26]. Thông thường, các mô hình GNN bao gồm một cơ chế truyền thông điệp lan truyền thông tin đặc trưng của các đỉnh đến các đỉnh lân cận cho đến khi đạt được trạng thái cân bằng ổn định. Quá trình này có thể được biểu diễn toán học như sau:

$$h_u^{(l)} = f(h_v^{l-1}), v \in N(u) \cup u \tag{2.8}$$

trong đó, u và v là các đỉnh thuộc đồ thị, h là vec-tơ biểu diễn đặc trưng của đỉnh trong đồ thị (node embedding vector) và $N(u)$ là các hàng xóm của đỉnh u. Một vài thuật toán GNN được đề xuất để cải thiện kỹ thuật truyền bản tin (message passing) này. Trong phần này, một vài cách tiếp cận phổ biến, các ưu điểm và hạn chế cùng ứng dụng của mạng nơ-ron đồ thị sẽ được trình bày bên dưới.

2.3.1 Các cách tiếp cận phổ biến trong mạng nơ-ron đồ thị

Mạng tích chập đồ thị (Graph Convolutional Network - GCN)

Mạng tích chập đồ thị (Graph convolutional network - GCN) [27] là nỗ lực đầu tiên tích hợp các phép toán tích chập (tương tự như CNN) vào GNN. Ý tưởng cốt lõi đằng sau bất kỳ GNN nào là tạo ra biểu diễn Euclid duy nhất của các đỉnh/liên kết trong đồ thị. Theo cách truyền thống, các phương pháp phổ

(spectral methods) tạo ra các vector biểu diễn đỉnh bằng cách sử dụng phân rã riêng (eigen decomposition), nhưng chúng không hiệu quả về mặt tính toán và không có tính tổng quát hóa. GCN vượt qua những thách thức này với phép xấp xỉ mạnh mẽ của nó, trong đó phương trình cập nhật của vector biểu diễn đỉnh h_u tại một lớp l cụ thể được cho bởi:

$$h_u^{(l)} = g\left[\theta^{(l)} h_u^{(l-1)} A^*\right] = \left[\theta^{(l)} h_u^{(l-1)} D^{-\frac{1}{2}} A D^{\frac{1}{2}}\right] \tag{2.9}$$

trong đó, A là ma trận kế, D là ma trận bậc, θ là tham số có thể học được và g là hàm kích hoạt. $A^* = D^{-\frac{1}{2}} A D^{\frac{1}{2}}$ được chuẩn hóa theo cách này để điều chỉnh tỷ lệ các đặc trưng của đỉnh và đồng thời đảm bảo tính ổn định về mặt số học. Điều quan trọng cần lưu ý là GCN dựa vào toàn bộ đồ thị (tức là ma trận kề đầy đủ) để học biểu diễn đỉnh, điều này không hiệu quả vì số lượng đỉnh lân cận của một đỉnh có thể thay đổi từ một đến hàng nghìn hoặc thậm chí nhiều hơn và không thể tổng quát hóa cho các đồ thị có kích thước khác nhau.

GraphSAGE

GraphSAGE là một phương pháp nhúng đỉnh quy nạp (inductive) khai thác các thuộc tính của đỉnh để học một hàm nhúng (embedding function) [28]. Nó hỗ trợ việc học đồng thời cả cấu trúc tô pô (topological structure) và phân phối các đặc trưng của đỉnh trong một vùng lân cận giới hạn. Tiền đề cơ bản là huấn luyện một mạng nơ-ron có khả năng nhận biết các thuộc tính cấu trúc của vùng lân cận đỉnh, từ đó chỉ ra vai trò cục bộ của nó trong đồ thị cùng với vị trí toàn cục. Ban đầu, thuật toán lấy mẫu các đặc trưng đỉnh trong vùng lân cận cục bộ của từng đỉnh trong dữ liệu có cấu trúc đồ thị. Tiếp theo là việc học các ánh xạ hàm thích hợp để tổng hợp thông tin mà mỗi đỉnh nhận được khi nó lan truyền qua các lớp GNN. Phương pháp học quy nạp này có khả năng mở rộng trên các đồ thị có kích thước khác nhau cũng như các đồ thị con trong một đồ thị cho trước. Phép toán được thực hiện tại lớp nhúng đỉnh thứ l được cho bởi:

$$h_u^{(l)} = f^{(l)}\left(h_u^{l-1}, h_{N(u)}^{(l-1)}\right) = g\left[\theta_C^{(l)} h_u^{(l-1)} + \theta_A^{(l)} \tilde{A}\left(h_{N(u)}^{(l-1}\right)\right] \tag{2.10}$$

trong đó, $\tilde{A}$ biểu diễn phép toán tổng hợp, g làm hàm kích hoạt, $h_u^{(l)}$ là biến đỉnh của đỉnh u tại lớp thứ l, $N(u)$ là các hàng xóm của đỉnh u, θ_C và θ_A lần lượt là các tham số của phép kết hợp và tổng hợp trong GNN.

Mạng chú ý đồ thị (Graph Attention Network - GAT)

Mạng chú ý đồ thị giả định rằng đóng góp của các đỉnh lân cận đến đỉnh mục tiêu không được xác định trước như GCN cũng không giống nhau như GraphSage. GAT áp dụng các cơ chế chú ý để học các trọng số tương đối giữa hai đỉnh được kết nối [29]. Phép toán tích chập đồ thị theo GAT đơn đầu (single head GAT) được định nghĩa như sau:

$$\begin{aligned} h_u^{(l)} &= \left[\sum_{v \in N(u) \cup u} \alpha_{uv} \theta^{(l)} h_u^{(l-1)} \right] \\ \alpha_{uv}^l &= \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[\theta^{(l)} h_u^{(l-1)} || \theta^{(l)} h_v^{(l-1)}\right]\right)\right)}{\sum_{v \in N(u) \cup u} \exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[\theta^{(l)} h_u^{(l-1)} || \theta^{(l)} h_v^{(l-1)}\right]\right)\right)} \end{aligned} \tag{2.11}$$

với trọng số chú ý α_{uv} định lượng độ mạnh của liên kết giữa đỉnh u và đỉnh hàng xóm v của u, $\theta^{(l)}$ là ma trận trọng số áp dụng cho tất cả các đỉnh tại lớp thứ l, $\vec{a}$ là vec-tơ trọng số, ^{T} là toán tử chuyển vị và $||$ là phép ghép nối hai vec-tơ. Phương trình 2.11 được sử dụng khi lớp thứ l không quan tâm tới đặc trưng của cạnh và khi sử dụng đặc trưng cạnh thì trọng số chú ý được tính như sau:

$$\alpha_{uv}^l = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[\theta^{(l)} h_u^{(l-1)} || \theta^{(l)} h_v^{(l-1)} || \theta_e^{(l)} h_{e_{uv}}^{(l-1)}\right]\right)\right)}{\sum_{v \in N(u) \cup u} \exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[\theta^{(l)} h_u^{(l-1)} || \theta^{(l)} h_v^{(l-1)} || \theta_e^{(l)} h_{e_{uv}}^{(l-1)}\right]\right)\right)} \tag{2.12}$$

với $\theta_e^{(l)}$ là ma trận trọng số áp dụng cho tất cả các cạnh tại lớp thứ l và $h_{e_{uv}}^{(l-1)}$ là đặc trưng của cạnh giữa hai đỉnh u và v tại lớp thứ $l-1$.

Đối với trường hợp sử dụng cơ chế chú ý nhiều đầu (Multihead attention), phương trình cập nhật cho biểu diễn đỉnh được biểu diễn như sau:

$$h_u^{(l)} = ||_{k=1}^k \left[g\left(\sum_{v \in N(u) \cup u} \alpha_{uv}^k \theta^{(l),k} h_u^{(l-1)} \right) \right] \tag{2.13}$$

trong đó, $||$ biểu thị phép nối (concatenation), α_{uv}^k là các hệ số chú ý đã được chuẩn hóa được tính toán bởi cơ chế chú ý thứ k, và $\theta^{(l),k}$ là ma trận trọng số của phép biến đổi tuyến tính tương ứng. Cơ chế này tổng hợp có chọn lọc các đóng góp của vùng lân cận và triệt tiêu các chi tiết cấu trúc nhỏ.

2.3.2 Các ưu điểm và hạn chế của mạng nơ-ron đồ thị

Ưu điểm của GNN

GNN mang lại nhiều lợi ích nổi bật khi làm việc với dữ liệu dạng đồ thị, một số ưu điểm chính bao gồm:

- **Khả năng học trên cấu trúc không gian phức tạp**: GNN có thể trực tiếp làm việc với dữ liệu có cấu trúc không gian phức tạp, chẳng hạn như các mạng xã hội, các phân tử hóa học, hoặc đồ thị giao thông. Điều này cho phép GNN khai thác thông tin từ cả các đỉnh và các cạnh, giúp mô hình hiểu sâu hơn về mối quan hệ và tương tác trong dữ liệu.
- **Khả năng tổng quát hóa tốt**: Nhờ cơ chế lan truyền thông tin qua các lớp đồ thị (graph propagation), GNN có khả năng tổng quát hóa tốt từ dữ liệu huấn luyện sang dữ liệu mới. Điều này đặc biệt hữu ích trong các bài toán mà cấu trúc đồ thị có thể thay đổi hoặc chưa được biết trước.
- **Ứng dụng linh hoạt**: GNN có thể được áp dụng trong nhiều lĩnh vực khác nhau, từ xử lý ngôn ngữ tự nhiên, sinh học, đến phân tích mạng xã hội và hệ thống đề xuất, nhờ khả năng mô hình hóa dữ liệu có cấu trúc đồ thị một cách tự nhiên và hiệu quả.
- **Khả năng xử lý dữ liệu không đồng nhất**: GNN có thể làm việc với dữ liệu không đồng nhất, bao gồm cả các đỉnh và các cạnh có thuộc tính khác nhau, giúp mô hình linh hoạt trong việc học các mối quan hệ phức tạp giữa các phần tử trong đồ thị.

Nhược điểm của GNN

Bên cạnh những ưu điểm nổi bật, GNN cũng tồn tại một số nhược điểm cần cân nhắc:

- **Khó khăn trong việc huấn luyện**: GNN có thể gặp phải khó khăn trong việc huấn luyện, đặc biệt khi kích thước đồ thị lớn và có nhiều lớp lan truyền thông tin. Điều này có thể dẫn đến việc tiêu tốn nhiều tài nguyên tính toán và thời gian huấn luyện dài.

- **Vấn đề vanishing gradient**: Khi số lượng lớp GNN tăng lên, thông tin có thể bị mất mát hoặc bị phân tán quá mức qua nhiều bước lan truyền, dẫn đến hiện tượng vanishing gradient. Điều này khiến việc học sâu trở nên khó khăn và giảm hiệu suất mô hình.

- **Giới hạn khả năng mở rộng**: Đối với các đồ thị rất lớn (như mạng xã hội với hàng triệu đỉnh), việc huấn luyện và suy luận với GNN có thể trở nên không hiệu quả do yêu cầu bộ nhớ và tính toán tăng đột biến. Điều này đòi hỏi các phương pháp giảm thiểu như chia nhỏ đồ thị hoặc các kỹ thuật lấy mẫu (sampling).

- **Thiếu tiêu chuẩn hóa**: Hiện tại, không có một tiêu chuẩn rõ ràng nào cho việc lựa chọn và thiết kế các kiến trúc GNN phù hợp với từng loại bài toán hoặc dữ liệu. Điều này có thể làm phức tạp quá trình phát triển mô hình và yêu cầu thử nghiệm nhiều loại kiến trúc khác nhau.

2.4 Ứng dụng của Mạng Nơ-ron Đồ thị (GNN)

GNN có rất nhiều ứng dụng quan trọng trong các lĩnh vực khác nhau, từ khoa học máy tính đến khoa học tự nhiên. Một số ứng dụng tiêu biểu bao gồm:

- **Phân tích mạng xã hội (Social Network Analysis)**: GNN được sử dụng để phân tích các mạng xã hội, giúp dự đoán các kết nối mới, xác định các cộng đồng hoặc nhóm người dùng có đặc điểm chung, và phát hiện các ảnh hưởng hoặc thông tin lan truyền trong mạng lưới.

- **Hóa học và sinh học (Chemistry and Biology)**: Trong các lĩnh vực này, GNN có thể được sử dụng để phân tích cấu trúc phân tử, dự đoán tính chất hóa học của các hợp chất hoặc mô hình hóa các tương tác protein-protein. Bằng cách biểu diễn các phân tử dưới dạng đồ thị, GNN có khả năng học các mối quan hệ giữa các nguyên tử và các liên kết hóa học.

- **Hệ thống đề xuất (Recommendation Systems)**: GNN có thể được áp dụng trong các hệ thống đề xuất bằng cách mô hình hóa mối quan hệ giữa người dùng và sản phẩm dưới dạng đồ thị. Điều này giúp cải thiện độ chính

xác của việc đề xuất các sản phẩm hoặc nội dung mới dựa trên mối quan hệ và tương tác giữa các người dùng hoặc sản phẩm trong hệ thống.

- **Xử lý ngôn ngữ tự nhiên (Natural Language Processing)**: GNN có thể được sử dụng để mô hình hóa các mối quan hệ phức tạp giữa các từ trong một câu hoặc giữa các câu trong một văn bản thông qua biểu diễn đồ thị. Điều này giúp mô hình học tốt hơn các ngữ cảnh phức tạp và mối quan hệ giữa các thành phần trong ngôn ngữ tự nhiên.
- **Đồ thị tri thức (Knowledge Graphs)**: GNN có thể được sử dụng để suy luận và hoàn thiện các đồ thị tri thức, giúp phát hiện ra các mối quan hệ tiềm ẩn giữa các thực thể trong cơ sở dữ liệu. Điều này đặc biệt hữu ích trong các hệ thống truy vấn và tìm kiếm thông tin.

Nhờ vào khả năng linh hoạt trong việc mô hình hóa dữ liệu đồ thị, GNN đang ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, đóng vai trò quan trọng trong việc giải quyết các bài toán phức tạp với dữ liệu có cấu trúc.

2.5 Cơ chế chú ý

Cơ chế chú ý hay Attention Mechanism là một phương pháp tiên tiến và đã trở thành nền tảng trong nhiều mô hình học máy hiện đại, đặc biệt trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và thị giác máy tính (CV). Cơ chế này lần đầu tiên được giới thiệu trong bài báo của Bahdanau và các cộng sự [13] nhằm cải thiện hiệu quả của các mô hình dịch máy.

2.5.1 Khái niệm cơ bản của cơ chế chú ý

Mục tiêu chính của cơ chế chú ý là cho phép mô hình tập trung vào những phần quan trọng của đầu vào khi tạo ra đầu ra. Khác với các phương pháp truyền thống, trong đó mỗi bước của quá trình tính toán đều xem toàn bộ đầu vào với cùng trọng số, Attention Mechanism gán trọng số khác nhau cho các phần khác nhau của đầu vào, giúp mô hình chú ý nhiều hơn đến những phần quan trọng.

Cơ chế chú ý tính toán một tập các trọng số *attention weights* để biểu thị mức độ quan trọng của mỗi phần tử trong chuỗi đầu vào. Các trọng số này sau đó

được dùng để tính toán một biểu diễn trọng số (weighted representation) của đầu vào.

2.5.2 Mô hình toán học cơ bản của cơ chế chú ý

Giả sử chúng ta có một chuỗi các vector đầu vào $X = (x_1, x_2, ..., x_n)$, và cần tính một vector ngữ cảnh c dựa trên các vector này. Vector ngữ cảnh c được tính toán bằng cách tổng hợp có trọng số các vector đầu vào với các trọng số chú ý tương ứng:

$$c = \sum_{i=1}^{n} \alpha_i x_i$$

Trong đó, α_i là trọng số chú ý tương ứng với vector x_i, và $\sum_{i=1}^{n} \alpha_i = 1$. Các trọng số chú ý α_i được tính dựa trên một hàm điểm (scoring function) giữa vector truy vấn q và từng vector x_i:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)}$$

Với $e_i = \text{score}(q, x_i)$ là điểm số tính giữa truy vấn q và vector x_i. Hàm điểm có thể được định nghĩa theo nhiều cách khác nhau, chẳng hạn như hàm tích vô hướng (dot product), hàm tương tác tuyến tính (bilinear), hoặc mạng nơ-ron nhiều lớp (MLP).

2.5.3 Phân loại cơ chế chú ý

Self-Attention

Self-Attention, hay còn gọi là *chú ý tự bản thân*, cho phép mô hình học cách tập trung vào các phần khác nhau của chuỗi đầu vào với chính nó. Đối với con người, việc xác định một từ trong câu có mối quan hệ với các từ còn lại như thế nào là tương đối dễ dàng dựa trên ngữ cảnh. Tuy nhiên, đối với máy móc, việc này đòi hỏi khả năng hiểu sâu hơn về ngữ cảnh của cả chuỗi. Self-Attention giúp mô hình dịch máy có thể tính toán điểm chú ý giữa các từ trong chuỗi với chính nó, từ đó hiểu rõ hơn về mối quan hệ giữa chúng.

Dot-product Attention

Dot-product Attention là một cơ chế trong đó trọng số chú ý được tính bằng tích vô hướng (*dot product*) của vector truy vấn (*query*) và vector khóa (*key*). Cụ thể, điểm số chú ý giữa từ s_t trong chuỗi và từ h_i khác được tính như sau:

$$\text{score}(s_t, h_i) = s_t^\top h_i \tag{2.14}$$

Điểm số này sau đó được chuẩn hóa qua hàm softmax để tính trọng số chú ý cho mỗi từ trong chuỗi.

Scaled Dot-product Attention

Scaled Dot-product Attention là một biến thể của Dot-product Attention, trong đó tích vô hướng được chia tỷ lệ bằng cách chia cho căn bậc hai của kích thước vector khóa n. Điều này giúp ổn định quá trình học và tránh việc các giá trị tích vô hướng trở nên quá lớn khi kích thước vector tăng lên. Công thức được biểu diễn như sau:

$$\text{score}(s_t, h_i) = \frac{s_t^\top h_i}{\sqrt{n}} \tag{2.15}$$

Multi-head Attention

Multi-head Attention là một cơ chế mở rộng của Dot-product Attention, trong đó các vector truy vấn, khóa, và giá trị (*query*, *key*, *value*) được chia thành nhiều phần, hay còn gọi là "đầu chú ý"(*attention heads*). Sau đó, Dot-product Attention được tính toán độc lập trên từng đầu chú ý, và các kết quả này được kết hợp lại để cung cấp một biểu diễn chú ý tổng hợp. Điều này cho phép mô hình học được nhiều khía cạnh khác nhau của dữ liệu.

Self-Attention mở rộng

Self-Attention cũng là một trường hợp đặc biệt của cơ chế chú ý, trong đó chuỗi đầu vào được sử dụng làm cả truy vấn và khóa, giúp mô hình học được mối quan hệ giữa các từ trong cùng một chuỗi. Self-Attention đã trở thành nền tảng của nhiều mô hình hiện đại, điển hình là mô hình Transformer.

Structured Attention

Structured Attention là một loại cơ chế chú ý cho phép các trọng số chú ý được học thông qua các mô hình dự đoán có cấu trúc, chẳng hạn như Conditional Random Fields (CRFs). Điều này giúp mô hình có khả năng nắm bắt được các mối quan hệ phức tạp và cấu trúc trong dữ liệu, từ đó nâng cao hiệu quả của các bài toán dự đoán.

2.5.4 Ưu và nhược điểm của cơ chế chú ý

Cơ chế chú ý là một công cụ mạnh mẽ giúp cải thiện hiệu suất của các mô hình học sâu và mang lại nhiều lợi ích quan trọng. Một số ưu điểm chính của cơ chế chú ý bao gồm:

- **Cải thiện độ chính xác**: Bằng cách cho phép mô hình tập trung vào thông tin liên quan nhất, cơ chế chú ý giúp nâng cao độ chính xác của các dự đoán.
- **Tăng cường hiệu quả**: Cơ chế chú ý giúp mô hình hoạt động hiệu quả hơn bằng cách chỉ xử lý những dữ liệu quan trọng nhất, từ đó giảm thiểu tài nguyên tính toán và làm cho mô hình có khả năng mở rộng tốt hơn.
- **Cải thiện khả năng giải thích**: Các trọng số chú ý mà mô hình học được có thể cung cấp cái nhìn sâu sắc về những phần nào của dữ liệu là quan trọng nhất, giúp nâng cao tính giải thích của mô hình.

Tuy nhiên, cơ chế chú ý cũng có một số nhược điểm cần cân nhắc:

- **Khó khăn trong việc huấn luyện**: Cơ chế chú ý có thể gặp khó khăn khi huấn luyện, đặc biệt là với các bài toán lớn và phức tạp. Điều này là do việc học các trọng số chú ý từ dữ liệu đòi hỏi một lượng lớn dữ liệu và tài nguyên tính toán.
- **Dễ xảy ra overfitting**: Cơ chế chú ý có thể dễ bị overfitting, tức là mô hình hoạt động rất tốt trên dữ liệu huấn luyện nhưng không khái quát tốt trên dữ liệu mới. Mặc dù các kỹ thuật regularization có thể giảm thiểu vấn đề này, nhưng nó vẫn là một thách thức khi làm việc với các bài toán lớn và phức tạp.

- **Bias phơi nhiễm (Exposure Bias)**: Cơ chế chú ý có thể gặp phải vấn đề bias phơi nhiễm, xảy ra khi mô hình được huấn luyện để tạo ra chuỗi đầu ra theo từng bước, nhưng khi kiểm thử, nó cần tạo ra toàn bộ chuỗi cùng một lúc. Điều này có thể dẫn đến hiệu suất kém trên dữ liệu kiểm thử, vì mô hình có thể không tạo ra toàn bộ chuỗi đầu ra một cách chính xác.

Dù cơ chế chú ý mang lại nhiều lợi ích, việc hiểu rõ những hạn chế của nó cũng là điều cần thiết để đảm bảo áp dụng phù hợp và hiệu quả trong các bài toán cụ thể.

2.5.5 Ứng dụng của cơ chế chú ý

Cơ chế chú ý có rất nhiều ứng dụng trong các bài toán học sâu khác nhau. Một số ứng dụng chính bao gồm:

- **Xử lý ngôn ngữ tự nhiên (NLP)**: Các tác vụ như dịch máy, tóm tắt văn bản và trả lời câu hỏi là những lĩnh vực mà cơ chế chú ý có vai trò quan trọng. Trong các tác vụ này, cơ chế chú ý giúp mô hình hiểu được ý nghĩa của từ trong ngữ cảnh và tập trung vào thông tin quan trọng nhất.
- **Thị giác máy tính (Computer Vision)**: Các bài toán như phân loại hình ảnh và nhận diện đối tượng cũng ứng dụng cơ chế chú ý để giúp mô hình xác định các phần quan trọng của hình ảnh và tập trung vào các đối tượng cụ thể trong khung cảnh.
- **Nhận diện giọng nói (Speech Recognition)**: Trong các tác vụ như phiên âm các bản ghi âm hoặc nhận diện lệnh nói, cơ chế chú ý giúp mô hình tập trung vào các phần liên quan của tín hiệu âm thanh và xác định từ ngữ được nói.
- **Tạo nhạc (Music Generation)**: cơ chế chú ý cũng được áp dụng trong việc tạo ra các giai điệu hoặc hợp âm, giúp mô hình tập trung vào các yếu tố âm nhạc liên quan và tạo ra các tác phẩm âm nhạc mạch lạc và giàu cảm xúc.

Nhìn chung, cơ chế chú ý có rất nhiều tiềm năng ứng dụng trong một loạt các tác vụ học sâu và là công cụ quan trọng để cải thiện hiệu suất của các mô hình này.

Chương 3

Mô hình hóa hệ thống và xây dựng máy trạng thái, chính sách cho các khối trong hệ thống

Chương 3 trình bày kiến trúc tổng quan của hệ thống gồm bốn lớp chính: lớp vật lý, lớp truyền thông, lớp điều khiển và lớp ứng dụng, cùng với vai trò và chức năng của từng lớp. Tiếp theo, các thành phần trong lớp vật lý và lớp điều khiển được mô tả và mô hình hóa một cách chi tiết. Cuối cùng, thuật toán quản lý trạng thái robot và chính sách phân nhiệm được đề xuất cho khối phân nhiệm, dựa trên chính sách phân nhiệm RTAW [4] kết hợp với mạng nơ-ron đồ thị (GAT), được trình bày cụ thể.

3.1 Mô hình hóa hệ thống

3.1.1 Kiến trúc tổng quan hệ thống

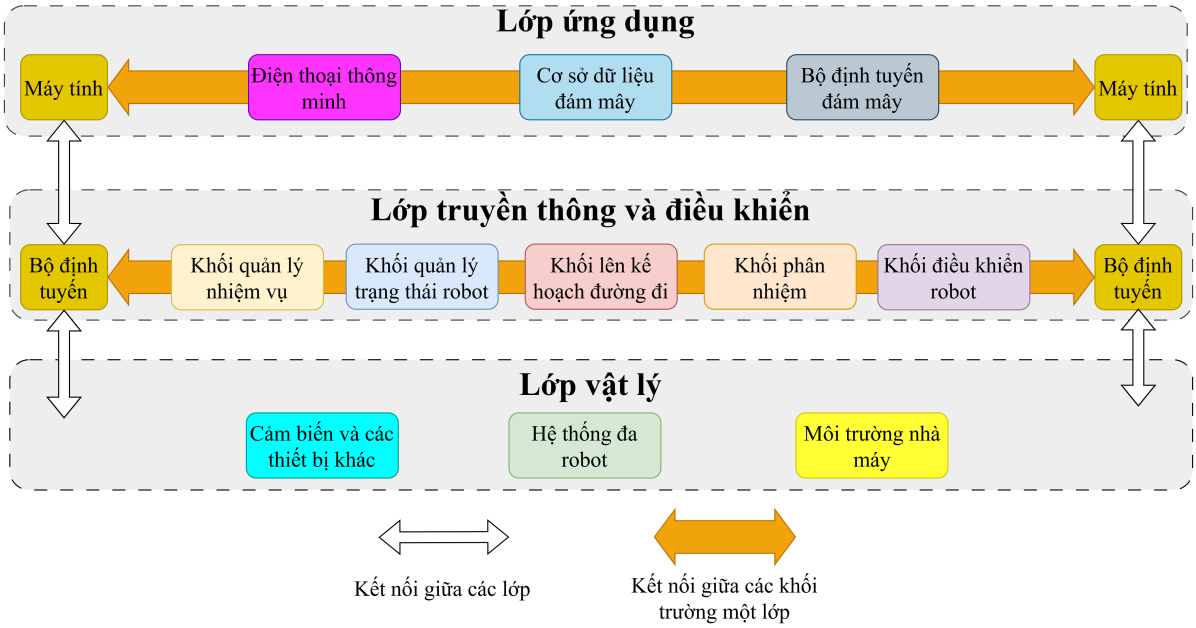
Hệ thống bao gồm bốn lớp chính: lớp vật lý (physical layer), lớp truyền thông (communication layer), lớp điều khiển (control layer), và lớp ứng dụng (application layer), như minh họa trong Hình 3.1. Lớp vật lý bao gồm các thiết bị phần cứng như cảm biến, robot, máy móc công nghiệp, thiết bị gia dụng và điện thoại thông minh. Lớp này chịu trách nhiệm thu thập dữ liệu từ môi trường thực thông qua các cảm biến, thực hiện các hành động qua thiết bị chấp hành và cung cấp dữ liệu đầu vào cho các lớp trên. Lớp truyền thông đảm bảo việc truyền tải dữ liệu ổn định và hiệu quả giữa lớp vật lý và các lớp khác, đồng thời duy trì tính bảo mật và độ tin cậy trong quá trình trao đổi thông tin. Lớp điều khiển xử lý và phân tích dữ liệu từ lớp vật lý để đưa ra quyết định điều khiển thiết bị, tích hợp các thuật toán và chiến lược nhằm đảm bảo hệ thống hoạt động đồng bộ và hiệu quả. Lớp ứng dụng là giao diện giữa hệ thống và người dùng, cung cấp các dịch vụ như quản lý nhà máy, điều khiển robot, giám sát từ xa hoặc tự động hóa. Lớp này hiển thị thông tin, kết quả phân tích và cho phép người dùng tương tác với hệ thống thông

qua các ứng dụng trực quan. Sự phối hợp giữa bốn lớp này giúp hệ thống thu thập và xử lý dữ liệu từ môi trường thực, đưa ra quyết định thông minh và cung cấp các dịch vụ hiệu quả trong nhiều lĩnh vực ứng dụng.

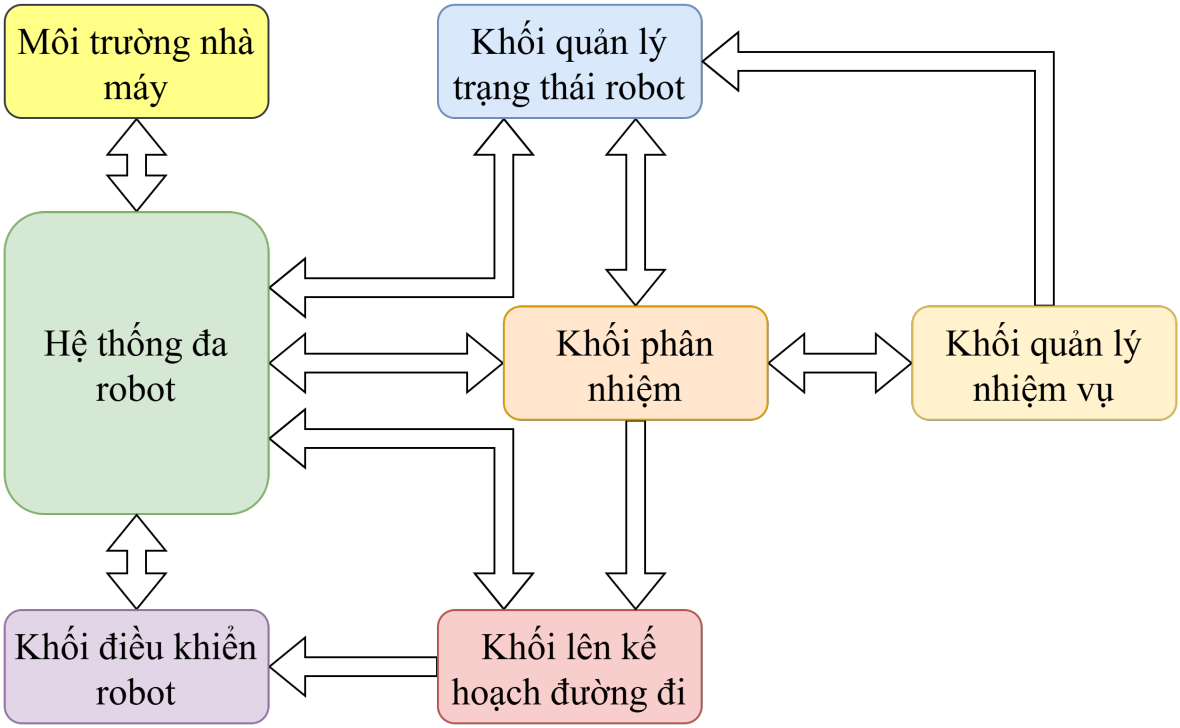
Trong phạm vi đề tài này, đồ án tập trung vào mô hình hóa lớp điều khiển và lớp vật lý, bỏ qua lớp truyền thông và lớp ứng dụng. Lớp vật lý được mô hình hóa với hai thành phần chính: môi trường nhà máy (factory environment) và hệ thống đa robot (multi-robot system). Lớp điều khiển bao gồm năm khối chính: khối quản lý nhiệm vụ (task management block), khối quản lý trạng thái robot (robot state management block), khối phân nhiệm (task allocation block), khối lên kế hoạch đường đi (path planning block) và khối điều khiển robot (robot control block). Hệ thống vận hành theo luồng dữ liệu và xử lý được minh họa trong Hình 3.2. Cụ thể, hệ thống đa robot tương tác trực tiếp với môi trường nhà máy để thực hiện các nhiệm vụ và thu thập dữ liệu về môi trường. Khối quản lý trạng thái robot tích hợp thông tin từ hệ thống đa robot, khối quản lý nhiệm vụ và khối phân nhiệm để xác định trạng thái hoạt động của từng robot, với thuật toán được trình bày trong Phần 3.2. Khối quản lý nhiệm vụ duy trì và cập nhật hàng đợi nhiệm vụ dựa trên phản hồi từ khối phân nhiệm, đảm bảo loại bỏ các nhiệm vụ đã được phân công và bổ sung nhiệm vụ mới. Khối phân nhiệm đóng vai trò then chốt trong tối ưu hóa hiệu suất hệ thống, phân tích dữ liệu từ khối quản lý nhiệm vụ, hệ thống đa robot và khối quản lý trạng thái để phân bổ nhiệm vụ một cách tối ưu, theo các chính sách trong Phần 3.3. Khối lên kế hoạch đường đi tính toán quỹ đạo tối ưu dựa trên dữ liệu từ khối phân nhiệm và hệ thống đa robot. Cuối cùng, khối điều khiển robot sử dụng dữ liệu từ khối lên kế hoạch đường đi và hệ thống đa robot để đưa ra quyết định điều khiển phù hợp, đảm bảo hiệu quả hoạt động của toàn hệ thống.

3.1.2 Mô hình hóa môi trường nhà máy

Môi trường nhà máy được mô hình hóa dưới dạng một đồ thị có hướng $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, như minh họa trong Hình 3.3 và Hình 3.4. Trong mô hình này, $\mathcal{V}$ là tập hợp các đỉnh, đại diện cho các vị trí trong không gian hai chiều (2D) của nhà máy, và $\mathcal{E}$ là tập hợp các cạnh có hướng, biểu diễn các tuyến đường di chuyển và hướng kết nối giữa các vị trí. Mỗi đỉnh $u \in \mathcal{V}$ được đặc trưng bởi một bộ ba giá trị (τ, x, y),



Hình 3.1: Kiến trúc tổng quan trong hệ thống đa robot trong nhà kho



Hình 3.2: Luồng dữ liệu giữa các khối trong hệ thống đa robot trong nhà máy

trong đó τ là loại của đỉnh, còn x và y là tọa độ không gian của đỉnh.

Các đỉnh trong đồ thị được phân loại thành năm loại chính: Điểm làm việc là nơi robot thực hiện các thao tác lấy hoặc trả hàng hóa, thường xuất hiện tại các vị trí sản xuất và được biểu diễn bằng các đỉnh màu xanh lá cây. Điểm kho là nơi robot lấy hàng để cung cấp cho các điểm làm việc hoặc nhận hàng từ các điểm làm việc để lưu trữ, được biểu diễn bằng các đỉnh màu xanh dương. Điểm chờ là nơi robot dừng lại khi không có nhiệm vụ, thường nằm ở vị trí chờ đợi, được biểu diễn bằng các đỉnh màu đỏ. Điểm sạc là các vị trí robot nạp năng lượng tại trạm sạc tự động, được biểu diễn bằng các đỉnh màu cam. Điểm trung chuyển đóng vai trò kết nối giữa các loại điểm khác, tạo thành một mạng lưới di chuyển khép kín trong không gian nhà máy và được biểu diễn bằng các đỉnh màu tím.

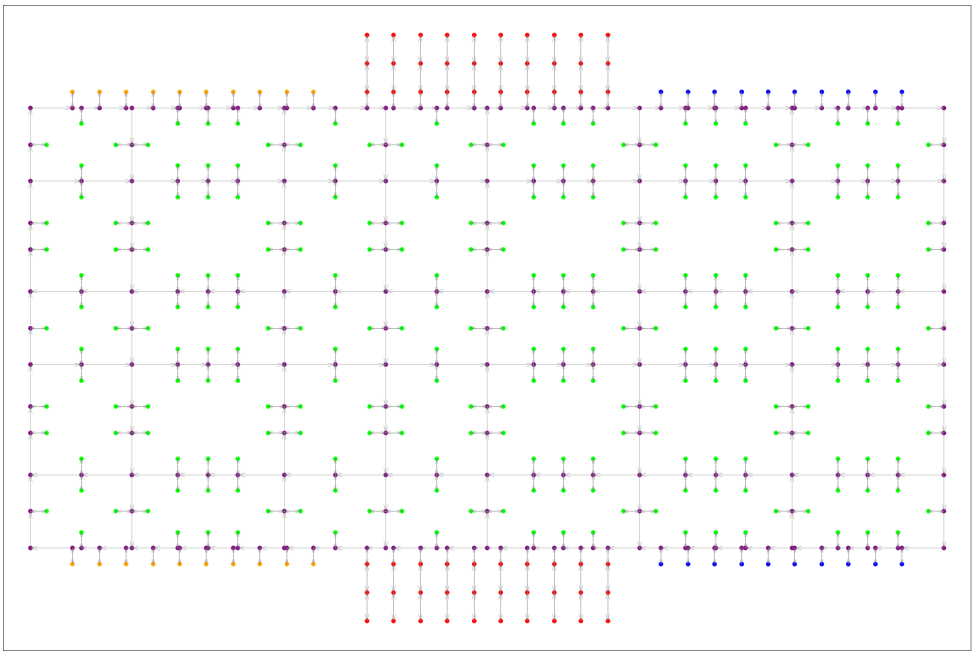
Mỗi cạnh $e \in \mathcal{E}$ trong đồ thị (biểu diễn bằng các mũi tên màu xám) được đặc trưng bởi ba tham số: góc φ giữa đường nối hai đỉnh và trục x, chiều dài d của cung (tính theo khoảng cách Euclid giữa hai đỉnh), và vận tốc tuyến tính tối đa ν mà robot có thể di chuyển trên cạnh đó.

Để đơn giản hóa quá trình lập kế hoạch đường đi, giảm thiểu va chạm và tập trung vào xây dựng phương pháp phân nhiệm, các đoạn đường giữa những điểm trung chuyển được thiết kế là các đường một chiều. Điều này đảm bảo rằng cạnh giữa hai đỉnh là điểm trung chuyển luôn có hướng cố định, đồng thời đảm bảo tính kết nối trong đồ thị, cho phép robot di chuyển từ bất kỳ đỉnh nào đến bất kỳ đỉnh khác.

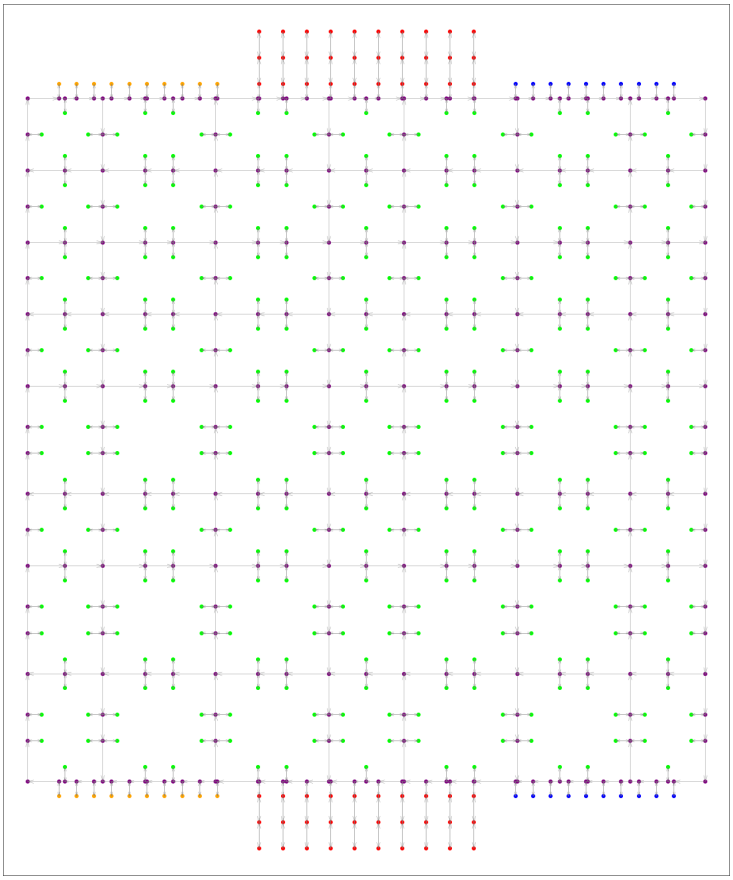
3.1.3 Mô hình hóa hệ thống đa robot

Hệ thống đa robot trong nhà máy được ký hiệu là $\mathcal{R} = (r_1, r_2, \ldots, r_N)$, trong đó N là số lượng robot. Mỗi robot r_i với $1 \leq i \leq N$ được mô hình hóa dưới dạng một hình chữ nhật có chiều dài l và chiều rộng w. Trạng thái của robot r_i tại thời điểm t được mô tả bằng bộ thông số $(x_{i,t}, y_{i,t}, \theta_{i,t}, v_{i,t}, \omega_{i,t}, e_{i,t})$, trong đó $x_{i,t}$ và $y_{i,t}$ là tọa độ vị trí của robot trong không gian 2D, $\theta_{i,t}$ là góc định hướng của robot so với trục x, $v_{i,t}$ là vận tốc tuyến tính, $\omega_{i,t}$ là vận tốc góc, và $e_{i,t}$ là năng lượng hiện tại của robot.

Trong phạm vi đồ án này, mô hình năng lượng của robot không được xây dựng, và quá trình tiêu hao năng lượng trong khi thực hiện nhiệm vụ được bỏ qua.



Hình 3.3: Môi trường thử nghiệm số 1



Hình 3.4: Môi trường thử nghiệm số 2

Đồng thời, vị trí và hướng của robot được tính toán trực tiếp thay vì sử dụng các bộ ước lượng vị trí như trong thực tế. Để mô phỏng sai số vị trí, nhiễu Gaussian $\mathcal{N}(\mu, \sigma^2)$ được thêm vào. Do đó, vị trí của robot luôn đảm bảo và năng lượng luôn được giả định ở trạng thái sẵn sàng nhận nhiệm vụ mới, nhằm đơn giản hóa mô hình và tập trung vào các khía cạnh khác của hệ thống.

Khi được khối phân nhiệm chỉ định nhiệm vụ và cung cấp lộ trình từ hệ thống lập kế hoạch đường đi, robot r_i cần tính toán các giá trị vận tốc tuyến tính $v_{i,t}$ và vận tốc góc $\omega_{i,t}$ để di chuyển theo đúng lộ trình đã định. Trong quá trình di chuyển, robot phải đảm bảo tránh va chạm với các robot khác trong hệ thống, dựa trên thông tin điều khiển từ khối điều khiển robot. Điều này yêu cầu sự phối hợp chặt chẽ giữa các thành phần trong hệ thống để đảm bảo hoạt động an toàn và hiệu quả trong môi trường nhà máy.

3.1.4 Mô hình hóa khối quản lý nhiệm vụ

Các nhiệm vụ trong hệ thống được quản lý thông qua một hàng đợi $\mathcal{M} = (m_1, m_2, \ldots, m_M)$, trong đó M là số lượng tối đa các nhiệm vụ có thể tồn tại trong hàng đợi tại một thời điểm nhất định. Mỗi nhiệm vụ m_j, với $1 \leq j \leq M$, được mô tả bằng một tập hợp các điểm $(p_1, p_2, \ldots, p_K)$ với K là số lượng điểm nhiệm vụ trong nhiệm vụ. Các điểm này, gọi là task points hoặc điểm nhiệm vụ, là các đỉnh trong đồ thị $\mathcal{G}$, tương ứng với các vị trí mà robot cần đến để thực hiện nhiệm vụ, chẳng hạn như các điểm lấy hàng hoặc trả hàng.

Trong phạm vi đồ án này, chính sách phân nhiệm được đề xuất dựa trên và so sánh với chính sách phân nhiệm RTAW [4]. Do đó, mỗi nhiệm vụ trong hàng đợi $\mathcal{M}$ chỉ bao gồm hai điểm chính: điểm bắt đầu và điểm kết thúc của nhiệm vụ. Khi một nhiệm vụ từ hàng đợi $\mathcal{M}$ được khối phân nhiệm chỉ định cho một robot, nhiệm vụ đó sẽ ngay lập tức được chuyển vào hàng đợi nhiệm vụ của robot tương ứng.

Quá trình thêm nhiệm vụ mới cần đảm bảo rằng nhiệm vụ này không trùng lặp với bất kỳ nhiệm vụ nào đã tồn tại trong hàng đợi $\mathcal{M}$ hoặc với các nhiệm vụ đang được thực hiện bởi bất kỳ robot nào trong tập hợp robot $\mathcal{R}$. Với giả thiết rằng hệ thống tại mọi thời điểm luôn có nhiệm vụ được giao, cùng với việc bỏ qua các nhiệm vụ liên quan đến việc di chuyển đến điểm chờ hoặc trạm sạc (theo mô

hình hệ thống đa robot được trình bày ở phần 3.1.3), khối quản lý nhiệm vụ chỉ tạo ra các nhiệm vụ gồm hai điểm. Các điểm này là giữa các vị trí làm việc với nhau hoặc giữa kho với các vị trí làm việc.

3.1.5 Mô hình hóa khối quản lý trạng thái robot

Trong phạm vi của đồ án, trạng thái hoạt động của robot được mô hình hóa thành bảy trạng thái rời rạc, được định nghĩa như sau.

Trạng thái **sẵn sàng (Ready)** là trạng thái trong đó robot duy trì mức năng lượng đủ để vận hành và sẵn sàng tiếp nhận nhiệm vụ mới từ khối phân nhiệm. Chỉ trong trạng thái này, khối phân nhiệm mới được phép chỉ định nhiệm vụ cho robot. Trạng thái **chờ (Waiting)** xảy ra khi robot không được phân công nhiệm vụ nhưng vẫn duy trì mức năng lượng hoạt động, đồng thời cần di chuyển đến vị trí chờ được chỉ định.

Khi robot nhận được nhiệm vụ từ khối phân nhiệm, robot chuyển sang trạng thái **lên kế hoạch (Planning)**, nơi khối lên kế hoạch đường đi sẽ tính toán lộ trình từ vị trí hiện tại của robot đến điểm thực hiện nhiệm vụ. Sau khi hoàn thành việc lập kế hoạch và di chuyển đến điểm đích, robot chuyển sang trạng thái **thực thi (Executing)**, thực hiện các thao tác cụ thể theo yêu cầu nhiệm vụ như lấy hàng, trả hàng hoặc tiếp cận trạm sạc ... Khi các thao tác tại điểm nhiệm vụ đã được hoàn thành, robot sẽ chuyển sang trạng thái **hoàn thành (Completed)**.

Trạng thái **năng lượng thấp (Low Battery)** được kích hoạt khi mức năng lượng của robot giảm xuống dưới ngưỡng an toàn, không đủ điều kiện nhận nhiệm vụ mới. Trong trạng thái này, robot cần di chuyển đến trạm sạc. Sau khi đến trạm sạc, robot bước vào trạng thái **sạc (Charging)**, nơi nó nạp lại năng lượng và không tiếp nhận nhiệm vụ mới cho đến khi năng lượng của robot đủ để nhận nhiệm vụ mới.

Việc xác định và quản lý trạng thái của robot được thực hiện thông qua mô hình máy trạng thái hữu hạn (Finite State Machine - FSM). Mô hình FSM cho phép hệ thống định nghĩa rõ ràng các trạng thái và quy tắc chuyển đổi giữa chúng. Chi tiết về cơ chế chuyển đổi trạng thái và các điều kiện kích hoạt sẽ được trình bày trong phần 3.2.

3.1.6 Mô hình hóa khối phân nhiệm

Bài toán phân nhiệm yêu cầu phân phối các nhiệm vụ trong hàng đợi $\mathcal{M}$ cho các robot trong tập $\mathcal{R}$, với các yêu cầu cơ bản: mỗi robot chỉ được chỉ định một nhiệm vụ từ khối quản lý nhiệm vụ $\mathcal{M}$ nếu có nhiệm vụ khả dụng; không có sự trùng lặp nhiệm vụ giữa các robot; và mỗi robot phải hoàn thành nhiệm vụ hiện tại trước khi nhận nhiệm vụ mới.

Hàm mục tiêu trong bài toán phân nhiệm được biểu diễn dưới dạng chi phí trung bình khi thực hiện nhiệm vụ của tất cả các robot trong hệ thống, như sau:

$$\min \frac{1}{\sum_{i=1}^{N} P_i} \sum_{i=1}^{N} \sum_{j=1}^{P_i} C_T(r_i, m_j) \tag{3.1}$$

Trong đó, $C_T(r_i, m_j)$ là chi phí khi robot r_i hoàn thành nhiệm vụ m_j, N là số lượng robot, và P_i là số lượng nhiệm vụ mà robot r_i đã hoàn thành. Hàm chi phí $C_T(r_i, m_j)$ có thể bao gồm thời gian và năng lượng mà robot r_i sử dụng để hoàn thành nhiệm vụ m_j.

Trong phạm vi đồ án này, chúng tôi chỉ xem xét chi phí liên quan đến thời gian mà robot hoàn thành nhiệm vụ, vì vậy hàm mục tiêu sẽ được điều chỉnh thành:

$$\min \frac{1}{\sum_{i=1}^{N} P_i} \sum_{i=1}^{N} \sum_{j=1}^{P_i} T(r_i, m_j) \tag{3.2}$$

Trong đó, $T(r_i, m_j)$ là thời gian mà robot r_i hoàn thành nhiệm vụ m_j.

3.1.7 Mô hình hóa khối lên kế hoạch đường đi

Bài toán lên kế hoạch đường đi trong hệ thống đa robot được định nghĩa với tập robot $\mathcal{R}$ và hàng đợi nhiệm vụ $\mathcal{M}$. Đối với mỗi nhiệm vụ m_j được gán cho một robot r_i, mục tiêu là xác định lộ trình tối ưu từ vị trí hiện tại của robot đến các điểm nhiệm vụ trong nhiệm vụ, đồng thời tránh va chạm với các robot khác.

Hàm mục tiêu của bài toán lên kế hoạch đường đi được biểu diễn dưới dạng chi phí di chuyển trung bình khi thực hiện nhiệm vụ của tất cả robot trong hệ thống, như sau:

$$\min \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{P_i} K_j} \sum_{i=1}^{N}\sum_{j=1}^{P_i}\sum_{k=1}^{K_j} C_P(r_i, m_j, p_k) \tag{3.3}$$

Trong đó, $C_P(r_i, m_j, p_k)$ là hàm chi phí khi robot r_i di chuyển đến điểm p_k trong quá trình thực hiện nhiệm vụ m_j, N là số lượng robot, P_i là số lượng nhiệm vụ mà robot r_i đã hoàn thành, và K_j là số lượng điểm nhiệm vụ trong nhiệm vụ m_j. Tương tự như với khối phân nhiệm, hàm mục tiêu của khối lên kế hoạch đường đi cũng có thể bao gồm tối ưu hóa thời gian và năng lượng mà robot r_i sử dụng trong quá trình di chuyển để hoàn thành nhiệm vụ m_j.

Trong đồ án này, thuật toán A* (A-star) được sử dụng để giải quyết bài toán. Hàm heuristic $h(v)$ được xác định là khoảng cách Manhattan từ đỉnh v đến đích g_j, tức là $h(v) = d(v, g_j)$, trong đó $d(v, g_j)$ là khoảng cách Manhattan. Thuật toán A* có độ phức tạp thời gian $O(|V| \log |V|)$ và độ phức tạp không gian $O(|V|)$, với $|V|$ là số lượng đỉnh trong đồ thị.

3.1.8 Mô hình hóa khối điều khiển robot

Sau khi nhận nhiệm vụ và lộ trình từ các khối phân nhiệm và lên kế hoạch đường đi, mỗi robot $r_i \in \mathcal{R}$ được điều khiển bởi khối điều khiển robot. Khối này sinh ra hai loại lệnh điều khiển cơ bản:

$$c(r_i, t) = \begin{cases} \text{PAUSE}, & \text{khi phát hiện khả năng va chạm} \\ \text{NORMAL}, & \text{khi đường đi an toàn} \end{cases}$$

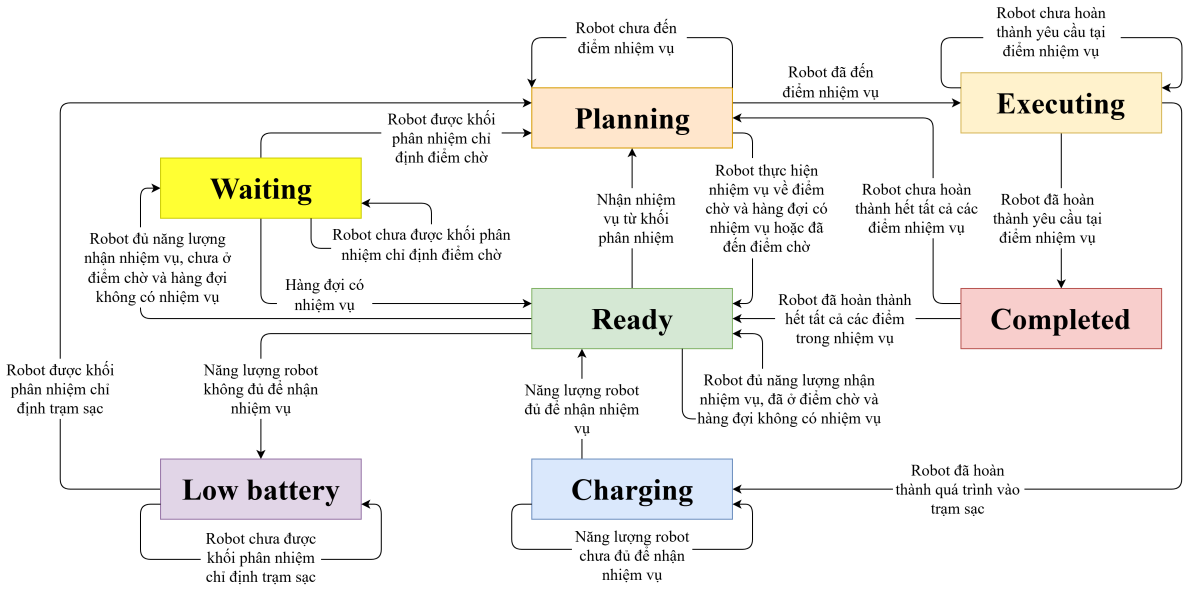
Lệnh PAUSE yêu cầu robot r_i dừng lại trong hai trường hợp. Trường hợp thứ nhất là khi robot gặp các robot khác tại các điểm giao nhau, được xác định là các đỉnh $v \in \mathcal{V}$ có bậc $\deg(v) \geq 3$, tương ứng với các ngã ba hoặc ngã tư trong môi trường thực tế. Trong trường hợp này, khối điều khiển sử dụng thời gian đã chờ tại các điểm giao nhau cùng với khoảng cách tới điểm giao để quyết định robot nào sẽ dừng lại hoặc tiếp tục di chuyển. Trường hợp thứ hai là khi khoảng cách giữa robot r_i và robot phía trước không đảm bảo an toàn, tức là $d(r_i, r_j) < d_{\text{safe}}$, trong đó d_{safe} là khoảng cách an toàn tối thiểu giữa hai robot.

Lệnh NORMAL cho phép robot di chuyển với vận tốc theo điều kiện $v(r_i, t) \leq v_{\max}(e_{ij})$, với R_i là đường đi mà robot nhận được từ khối lên kế hoạch đường đi và

$v_{\max}(e_{ij})$ là vận tốc tối đa cho phép trên cạnh e_{ij} của đồ thị $\mathcal{G}$. Tại mỗi thời điểm, hệ thống điều khiển thực hiện giám sát liên tục để đảm bảo tính an toàn cho toàn bộ hệ thống đa robot.

3.2 Xây dựng thuật toán cho khối quản lý trạng thái robot

Trong quá trình thiết kế hệ thống robot tự động hoạt động trong môi trường nhà máy, một thành phần then chốt là khối quản lý trạng thái robot, chịu trách nhiệm điều khiển các trạng thái vận hành từ khi robot nhận nhiệm vụ, lập kế hoạch di chuyển, đến khi hoàn thành nhiệm vụ hoặc cần sạc lại năng lượng. Để quản lý các trạng thái phức tạp này, một máy trạng thái hữu hạn (Finite State Machine - FSM) đã được áp dụng nhằm đảm bảo các quá trình chuyển trạng thái diễn ra hợp lý và có tổ chức. Hình 3.5 mô tả chi tiết mô hình FSM mà hệ thống sử dụng, cung cấp cái nhìn rõ ràng về cơ chế hoạt động của robot trong quá trình vận hành.



Hình 3.5: Sơ đồ máy trạng thái hữu hạn của khối quản lý trạng thái robot

Khi robot khởi động thành công với mức năng lượng đủ để thực hiện nhiệm vụ, trạng thái của robot chuyển từ *START* sang *Ready*. Trạng thái *Ready* cho phép robot nhận nhiệm vụ mới từ hàng đợi. Nếu hàng đợi không có nhiệm vụ và robot đã ở điểm chờ, robot sẽ duy trì trạng thái *Ready*. Khi có nhiệm vụ mới, robot được chỉ định nhiệm vụ từ khối phân nhiệm và chuyển sang trạng thái *Planning*. Tại

đây, khối lên kế hoạch sẽ liên tục gửi lộ trình để robot di chuyển đến điểm nhiệm vụ. Khi robot đến điểm nhiệm vụ, trạng thái chuyển từ *Planning* sang *Executing*, nơi robot thực hiện các yêu cầu tại điểm nhiệm vụ. Sau khi hoàn thành yêu cầu tại một điểm, nếu còn các điểm nhiệm vụ khác, robot sẽ quay lại trạng thái *Planning* để lập kế hoạch đến điểm tiếp theo. Nếu đã hoàn thành toàn bộ nhiệm vụ, robot chuyển về trạng thái *Ready*.

Nếu trong trạng thái *Ready* mà hàng đợi không có nhiệm vụ mới, robot sẽ chuyển sang trạng thái *Waiting* và chờ được chỉ định điểm chờ. Tại đây, robot có thể chuyển về trạng thái *Ready* nếu hàng đợi xuất hiện nhiệm vụ mới. Trong trường hợp được chỉ định điểm chờ, robot chuyển sang trạng thái *Planning* để nhận lộ trình đến điểm chờ và trở về *Ready* khi đã đến nơi. Nếu trên đường đến điểm chờ mà xuất hiện nhiệm vụ mới, robot sẽ chuyển trực tiếp về trạng thái *Ready*.

Khi năng lượng của robot không đủ để nhận nhiệm vụ, trạng thái chuyển từ *Ready* sang *Low battery*, và robot chờ được phân công điểm sạc. Sau khi được chỉ định, robot chuyển sang trạng thái *Planning* để nhận lộ trình đến trạm sạc. Khi đến trạm, trạng thái đổi sang *Executing*, và robot thực hiện các thao tác cần thiết để tiếp cận trạm sạc. Sau đó, robot chuyển sang trạng thái *Charging*, nơi robot được sạc đến khi năng lượng đủ để thực hiện nhiệm vụ mới, và trạng thái trở về *Ready*.

Thuật toán quản lý trạng thái robot hoạt động dựa trên cơ chế phản hồi các điều kiện theo thời gian thực, chẳng hạn như nhận nhiệm vụ, hoàn thành công việc, hoặc phát hiện mức năng lượng thấp. Quá trình chuyển đổi trạng thái được kích hoạt ngay khi các điều kiện được thỏa mãn, đảm bảo rằng robot hoạt động hiệu quả và liên tục trong môi trường phức tạp. Cơ chế này không chỉ hỗ trợ tối ưu hóa hiệu suất mà còn đảm bảo an toàn và dễ dàng mở rộng trong các ứng dụng công nghiệp.

3.3 Xây dựng chính sách phân nhiệm cho hệ thống đa robot

Trong phần này, chi tiết chính sách phân nhiệm của hệ thống đa robot được trình bày, với mục tiêu thực hiện các nhiệm vụ từ hàng đợi, đồng thời bỏ qua chính sách phân nhiệm liên quan đến trạm sạc và điểm chờ. Chính sách phân nhiệm được phát triển dựa trên phương pháp phân nhiệm RTAW [4], kết hợp với mạng nơ-ron

đồ thị. Nội dung của phần này được chia thành hai phần. Phần 3.3.1 trình bày phương pháp RTAW và các giả thuyết liên quan. Phần 3.3.2 mô tả chi tiết chính sách phân nhiệm đề xuất, bao gồm các điều chỉnh và cải tiến nhằm tối ưu hóa hiệu quả thực hiện nhiệm vụ của các robot, đồng thời tăng cường tính linh hoạt cho quá trình phân nhiệm trong thực tế.

3.3.1 Phương pháp phân nhiệm RTAW

Phương pháp RTAW (Reinforcement-based Task Assignment for Warehouse) áp dụng học tăng cường để phân nhiệm vụ cho các robot trong môi trường nhà kho, với mục tiêu tối ưu hóa quy trình phân nhiệm và giảm thiểu thời gian di chuyển không thực hiện nhiệm vụ của robot (Total Travel Delay - TTD). Phương pháp này mô hình hóa quá trình phân nhiệm thành một bài toán Quyết định Markov (MDP) và sử dụng mạng nơ-ron với cơ chế attention để quản lý đồng thời nhiều robot và nhiệm vụ. Tại mỗi thời điểm, khối phân nhiệm chỉ phân nhiệm cho duy nhất một robot ở trạng thái *Ready*, các robot còn lại được phân nhiệm vào thời điểm sau.

Không gian trạng thái $\mathcal{S}$ biểu diễn trạng thái của hệ thống, trong đó mỗi trạng thái $s \in \mathcal{S}$ được xác định như sau:

$$s = \{(x_i, y_i, \eta_i) \mid 0 \leq i \leq N\} \cup \{(o_j, d_j, k_j, l_j) \mid 0 \leq j \leq M\} \cup \{x_{\text{sel}}, y_{\text{sel}}\}$$

với: x_i, y_i là vị trí của robot r_i, η_i là thời gian còn lại để hoàn thành nhiệm vụ hiện tại của robot r_i, o_j, d_j là điểm bắt đầu và điểm kết thúc của nhiệm vụ m_j trong hàng đợi $\mathcal{M}$, k_j là khoảng cách (tính theo thuật toán A*) từ robot được chọn đến điểm bắt đầu của nhiệm vụ m_j, l_j là khoảng cách (tính theo thuật toán A*) từ điểm bắt đầu đến điểm kết thúc của nhiệm vụ m_j, và $\{x_{\text{sel}}, y_{\text{sel}}\}$ là vị trí của robot được chọn để phân nhiệm. Thời gian còn lại để hoàn thành nhiệm vụ của robot r_i được ước tính theo công thức $\eta_i = \eta_{i,1} + \eta_{i,2} + \eta_{i,3}$, trong đó $\eta_{i,1}$ là tổng thời gian ước tính thực hiện đường đi từ vị trí hiện tại đến điểm bắt đầu và điểm kết thúc của nhiệm vụ hiện tại (bằng tổng độ dài giữa hai điểm liên tiếp thuộc đường đi chia cho tổng vận tốc tối đa robot có thể di chuyển giữa hai điểm đó), $\eta_{i,2}$ là tổng thời gian thực hiện yêu cầu tại mỗi điểm nhiệm vụ (giả định 3 s/yêu cầu), và $\eta_{i,3}$ là thời gian cộng thêm trong quá trình robot thực hiện nhiệm vụ, được tính bằng bội số của tổng $\eta_{i,1} + \eta_{i,2}$ với một hệ số ngẫu nhiên theo phân phối Gauss $\mathcal{N}(0.25, 0.25)$.

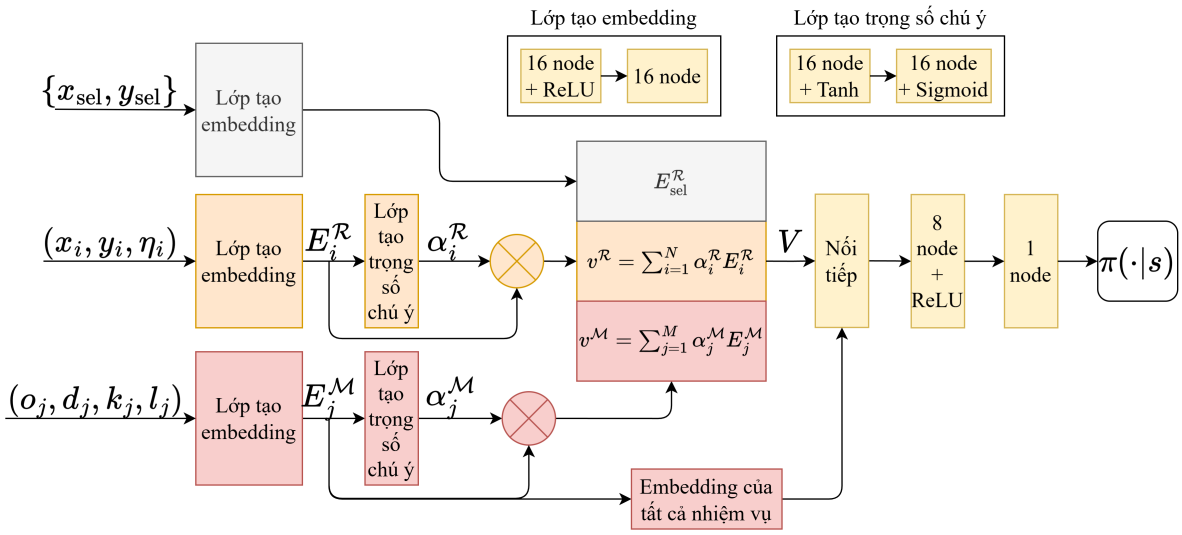
Trạng thái s_t được ghi lại khi một robot ở trạng thái *Ready*, và trạng thái s_{t+1} khi robot khác chuyển sang trạng thái *Ready*.

Hành động $a \in \mathcal{A}$ là việc chọn một nhiệm vụ j từ hàng đợi $\mathcal{M}$ để chỉ định cho robot đang sẵn sàng được chọn. Chính sách π xác định xác suất chọn mỗi nhiệm vụ cho robot được chọn. Phần thưởng r trong phương pháp RTAW được thiết kế dựa trên TTD: khi một robot được chỉ định nhiệm vụ (thực hiện hành động a), phần thưởng là giá trị âm của thời gian robot di chuyển đến điểm bắt đầu của nhiệm vụ. Bằng cách tối thiểu hóa TTD, hệ thống tối ưu hóa thời gian hoàn thành nhiệm vụ và tăng cường hiệu quả phân nhiệm, như đã chứng minh trong [4].

Kiến trúc mạng nơ-ron của hệ thống được thiết kế sử dụng cơ chế attention, giúp mạng có khả năng xử lý hiệu quả các đầu vào với kích thước thay đổi như trong Hình 3.6.. Đối với mỗi robot r_i, đặc trưng $\mathcal{F}_i^{\mathcal{R}} = \{x_i, y_i, \eta_i\}$ được truyền qua một lớp mạng với 16 node và hàm kích hoạt ReLU, sau đó qua một lớp mạng khác cũng gồm 16 node để tạo ra embedding tương ứng, ký hiệu là $E_i^{\mathcal{R}}$. Tương tự, embedding của robot được chọn $E_{sel}^{\mathcal{R}}$ được tính toán từ đặc trưng $\{x_{\text{sel}}, y_{\text{sel}}\}$. Đối với mỗi nhiệm vụ $m_j \in \mathcal{M}$, đặc trưng $\mathcal{F}_j^{\mathcal{M}} = \{o_j, d_j, k_j, l_j\}$ được truyền qua một lớp mạng với 16 node sử dụng hàm kích hoạt ReLU, tiếp theo là một lớp mạng khác cũng gồm 16 node để tạo ra embedding tương ứng, ký hiệu là $E_j^{\mathcal{M}}$. Embedding của robot và nhiệm vụ sau đó được đưa vào một lớp mạng với 16 node sử dụng hàm kích hoạt Tanh, tiếp theo là một lớp mạng 1 node với hàm kích hoạt sigmoid để tính toán các trọng số attention $\alpha_i^{\mathcal{R}}$ và $\alpha_j^{\mathcal{M}}$. Các trọng số này được dùng để tổng hợp các vector toàn cục:

$$v^{\mathcal{R}} = \sum_{i=1}^{N} \alpha_i^{\mathcal{R}} E_i^{\mathcal{R}}, \quad v^{\mathcal{M}} = \sum_{j=1}^{M} \alpha_j^{\mathcal{M}} E_j^{\mathcal{M}}.$$

Sau đó, các vector $v^{\mathcal{R}}$, $v^{\mathcal{M}}$, và $E_{sel}^{\mathcal{R}}$ được ghép nối để tạo thành vector tổng hợp $V = (v^{\mathcal{R}}, v^{\mathcal{M}}, E_{sel}^{\mathcal{R}})$. Vector V cùng với các embedding của nhiệm vụ được đưa qua một lớp mạng với 8 node sử dụng hàm kích hoạt ReLU, và cuối cùng là một lớp mạng 1 node sử dụng hàm kích hoạt softmax để tính xác suất lựa chọn cho từng nhiệm vụ.



Hình 3.6: Kiến trúc mạng nơ-ron của phương pháp RTAW cho phân nhiệm đa robot

3.3.2 Phương pháp phân nhiệm đề xuất

Phương pháp phân nhiệm RTAW [4] đã chứng minh tính linh hoạt trong hệ thống phân nhiệm đa robot, đặc biệt khi số lượng robot và nhiệm vụ trong hàng đợi thay đổi. Tuy nhiên, phương pháp này tồn tại hai hạn chế. Thứ nhất, việc dự đoán thời gian còn lại để robot hoàn thành nhiệm vụ dựa trên một bộ số cố định từ phân phối Gaussian đòi hỏi tinh chỉnh phù hợp với từng môi trường nhà máy, làm giảm tính ứng dụng thực tế, đặc biệt khi không có dữ liệu trước để hiệu chỉnh. Thứ hai, RTAW được thiết kế chủ yếu cho các nhiệm vụ gồm hai điểm, trong khi thực tế, các nhiệm vụ trong môi trường nhà máy thường phức tạp hơn, bao gồm một, hai, ba, hoặc nhiều điểm tùy thuộc vào yêu cầu của từng nhà máy.

Để khắc phục những hạn chế này, một phương pháp phân nhiệm mới được đề xuất với cách biểu diễn không gian trạng thái $\mathcal{S}$ và mô hình chính sách phân nhiệm với những cải tiến mới. Phương pháp này kết hợp kiến trúc mạng nơ-ron dựa trên cơ chế chú ý của RTAW với mạng nơ-ron đồ thị (Graph Attention Network - GAT) để tăng tính linh hoạt trong các môi trường thực tế. Mặc dù các hàm phần thưởng và không gian hành động của hệ thống vẫn giữ nguyên như trong RTAW, cách biểu diễn trạng thái và kiến trúc mạng nơ-ron mang lại hiệu suất tương tự và trong một vài trường hợp có hiệu suất cao hơn.

Không gian trạng thái $\mathcal{S}$ bao gồm mỗi trạng thái $s \in \mathcal{S}$ được biểu diễn dưới dạng một đồ thị $\mathcal{G}_a = \{\mathcal{V}_a, \mathcal{E}_a\}$, được xây dựng dựa trên đồ thị của hệ thống $\mathcal{G}$ đã

trình bày trong phần 3.1. Trong đồ thị này, tập hợp $\mathcal{V}_a$ chứa các vector đặc trưng của các đỉnh, còn tập hợp $\mathcal{E}_a$ chứa các vector đặc trưng của các cạnh.

Mỗi đỉnh $u_a = (\tau_a, x, y) \in \mathcal{V}_a$ được định nghĩa bởi ba yếu tố chính. Thứ nhất, loại của đỉnh τ_a, được biểu diễn dưới dạng một số, xác định chức năng của đỉnh trong đồ thị. Thứ hai, tọa độ x, y của đỉnh trong không gian 2D, xác định vị trí của đỉnh trên bản đồ. Các đỉnh được phân loại thành 6 loại khác nhau, bao gồm: đỉnh đại diện cho các điểm nhiệm vụ trong hàng đợi (loại 1), đỉnh đại diện cho các điểm nhiệm vụ đã được giao (loại 2), đỉnh đại diện cho các robot đã được phân nhiệm (loại 3), đỉnh đại diện cho các robot chưa được phân nhiệm (loại 4), đỉnh đại diện cho robot được chọn để phân nhiệm (loại 5), và các đỉnh còn lại (loại 6).

Mỗi cạnh $e_a = (\varphi_a, d_a, \nu_a) \in \mathcal{E}_a$ được đặc trưng bởi ba tham số. Góc φ_a xác định hướng của đường nối giữa hai đỉnh so với trục x. Khoảng cách d_a được tính bằng thuật toán A* giữa hai đỉnh, đảm bảo rằng khoảng cách được đo theo đường đi ngắn nhất khả thi trong môi trường. Cuối cùng, vận tốc tuyến tính trung bình ν_a phản ánh vận tốc tối đa trung bình trên đường đi này.

Để xây dựng đồ thị $\mathcal{G}_a$, ngoài các cạnh ban đầu được định nghĩa trong $\mathcal{G}$, một số quy tắc bổ sung được áp dụng nhằm tạo các cạnh mới. Các cạnh có hướng được thêm vào từ các đỉnh loại 5 đến các đỉnh loại 1. Đồng thời, các cạnh được thêm từ đỉnh loại 2 đến các đỉnh loại 3 (điểm nhiệm vụ thuộc nhiệm vụ mà robot đó đang thực hiện). Các điểm loại 1 và loại 2 cũng được liên kết với nhau nếu chúng cùng thuộc một nhiệm vụ. Những cạnh này giúp biểu diễn mối liên hệ giữa robot và nhiệm vụ mà chúng thực hiện. Ngoài ra, các đỉnh loại 6, đại diện cho các điểm thuộc trên đường đi được tính bằng thuật toán A*, cũng được liên kết với các đỉnh nhiệm vụ. Các đỉnh loại 6 được nối với các đỉnh loại 1 khi chúng thuộc đường đi từ robot được chọn đến các điểm nhiệm vụ trong hàng đợi. Các đỉnh loại 6 cũng nối đến các đỉnh loại 3 nếu chúng thuộc đường đi từ robot đến các điểm nhiệm vụ hiện tại của robot. Cuối cùng, để hoàn thiện đồ thị, các cạnh được thêm từ các đỉnh loại 3 và 4 đến đỉnh loại 5 tạo ra sự trao đổi thông tin giữa robot được chọn để phân nhiệm với các robot khác trong hệ thống. Việc áp dụng các quy tắc trên đảm bảo rằng trạng thái s được mô hình hóa một cách toàn diện, không chỉ biểu diễn mối quan hệ giữa robot và nhiệm vụ mà còn mô tả đầy đủ không gian nhiệm vụ và mối quan hệ giữa các thành phần trong hệ thống, cũng như giúp khối phân nhiệm có thể dự đoán thời gian hoàn thành nhiệm vụ của các robot và đưa ra lựa

chọn phân nhiệm tối ưu.

Kiến trúc mạng nơ-ron đề xuất được minh họa trong Hình 3.7. Đầu tiên, đồ thị $\mathcal{G}_a$ được truyền qua một lớp GAT (Graph Attention Network) với số lượng đặc trưng đầu ra là 8, số đầu (head) là 2, và sử dụng kết hợp đặc trưng của cạnh cùng với hàm kích hoạt Tanh. Tiếp theo, đồ thị được xử lý qua một lớp GAT thứ hai với cùng số lượng đặc trưng đầu ra và số đầu, nhưng không sử dụng đặc trưng của cạnh. Kết quả thu được là tập hợp đặc trưng của tất cả các đỉnh trong đồ thị, ký hiệu là $E_{\mathcal{V}_a}$. Từ $E_{\mathcal{V}_a}$, các embedding liên quan được trích xuất. Cụ thể, embedding của các điểm nhiệm vụ trong nhiệm vụ $m_j \in \mathcal{M}$ được ký hiệu lần lượt là $E_j^1, \ldots, E_j^k$, với $1 \leq k \leq K$, trong đó K là số điểm nhiệm vụ trong nhiệm vụ m_j. Embedding của robot r_i và embedding của robot được chọn lần lượt được ký hiệu là $E_i^{\mathcal{R}}$ và $E_{sel}^{\mathcal{R}}$.

Embedding của các điểm nhiệm vụ được truyền qua một lớp mạng nơ-ron một node với hàm kích hoạt Sigmoid để tính toán trọng số chú ý $\alpha_j^1, \ldots, \alpha_j^k$. Sau đó, embedding của nhiệm vụ được tính theo công thức:

$$E_j^{\mathcal{M}} = \sum_{k=0}^{K} \alpha_j^k E_j^k.$$

Tiếp theo, embedding của robot và embedding của nhiệm vụ được đưa qua một lớp mạng nơ-ron một node với hàm kích hoạt Sigmoid để tính toán trọng số chú ý tương ứng $\alpha_i^{\mathcal{R}}$ và $\alpha_j^{\mathcal{M}}$. Dựa trên các trọng số này, các vector toàn cục được tổng hợp như sau:

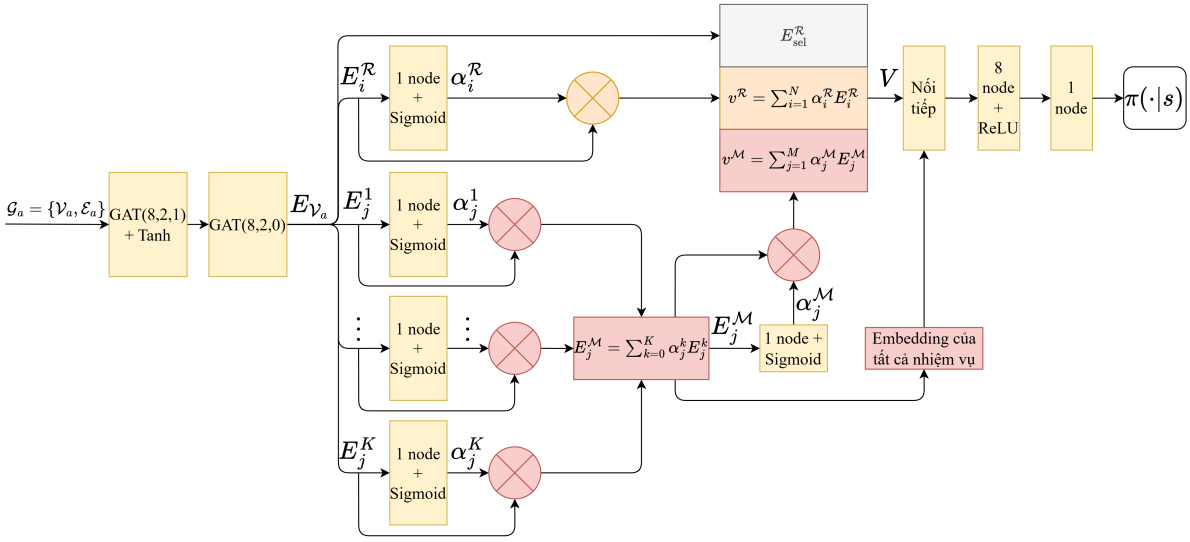
$$v^{\mathcal{R}} = \sum_{i=1}^{N} \alpha_i^{\mathcal{R}} E_i^{\mathcal{R}}, \quad v^{\mathcal{M}} = \sum_{j=1}^{M} \alpha_j^{\mathcal{M}} E_j^{\mathcal{M}}.$$

Sau đó, các vector $v^{\mathcal{R}}$, $v^{\mathcal{M}}$, và $E_{sel}^{\mathcal{R}}$ được ghép nối để tạo thành vector tổng hợp V:

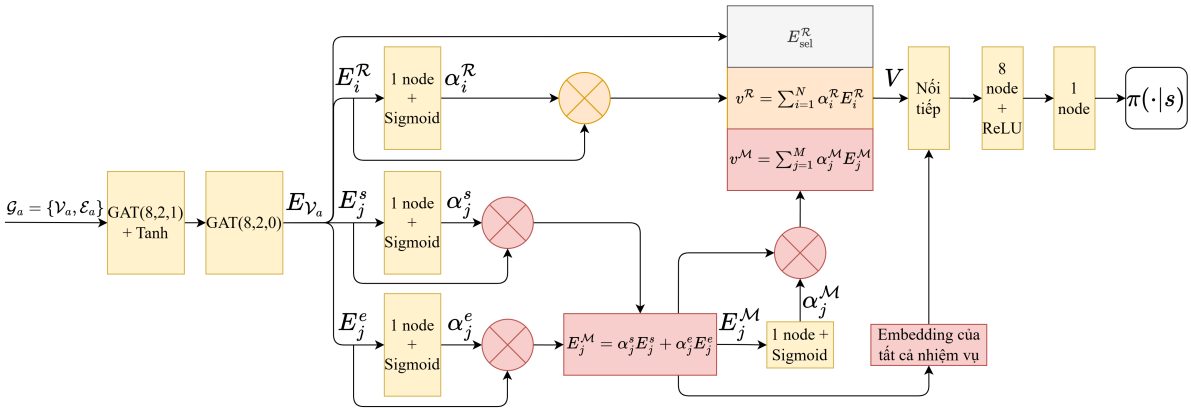
$$V = (v^{\mathcal{R}}, v^{\mathcal{M}}, E_{sel}^{\mathcal{R}}).$$

Vector V cùng với các embedding của nhiệm vụ được đưa qua một lớp mạng nơ-ron với 8 node, sử dụng hàm kích hoạt ReLU, và cuối cùng qua một lớp mạng một node sử dụng hàm kích hoạt Softmax để tính xác suất lựa chọn cho từng nhiệm vụ. Với kiến trúc này, việc đáp ứng linh hoạt cho số lượng điểm nhiệm vụ, số lượng nhiệm vụ và số lượng robot thay đổi luôn được đảm bảo cho quá trình vận hành hệ thống trong thực tế.

Để đánh giá hiệu suất của phương pháp phân nhiệm đề xuất so với phương pháp phân nhiệm RTAW, mô hình trong đồ án này được giới hạn trong các nhiệm vụ có hai điểm nhiệm vụ trong quá trình huấn luyện. Kiến trúc mạng nơ-ron cho trường hợp này được minh họa trong Hình 3.8.



Hình 3.7: Kiến trúc mạng nơ-ron của phương pháp đề xuất cho phân nhiệm đa robot với nhiệm vụ có số lượng điểm nhiệm vụ không có định.



Hình 3.8: Kiến trúc mạng nơ-ron của phương pháp đề xuất cho phân nhiệm đa robot với nhiệm vụ chỉ có hai điểm nhiệm vụ.

Chương 4

Kết quả

Trong phần này, đồ án trình bày chi tiết mục tiêu thí nghiệm, các thiết lập thí nghiệm và cuối cùng đưa ra các kết quả so sánh giữa thuật toán phân nhiệm đề xuất với các thuật toán phân nhiệm RTAW và tham lam dựa trên khoảng cách đến điểm nhiệm vụ.

4.1 Mục tiêu thí nghiệm

Các thí nghiệm trong đồ án được thiết kế nhằm kiểm chứng tính hiệu quả của phương pháp phân nhiệm đề xuất trong hệ thống đa robot tại nhà máy. Để đánh giá, thời gian trung bình mà robot hoàn thành một nhiệm vụ được sử dụng làm thước đo chính. Kết quả thu được từ phương pháp phân nhiệm đề xuất được so sánh với hai phương pháp khác, bao gồm phương pháp phân nhiệm RTAW và phương pháp đấu giá dựa trên khoảng cách đến điểm nhiệm vụ. Các thí nghiệm này không chỉ cung cấp đánh giá toàn diện về hiệu suất mà còn làm rõ các ưu điểm và hạn chế của phương pháp mới trong bối cảnh thực tế.

4.2 Thiết lập thí nghiệm

Phần này mô tả chi tiết các thiết lập và quy trình triển khai, huấn luyện, và đánh giá mô hình cho bài toán phân nhiệm đa robot trong môi trường nhà máy sản xuất. Hai bố cục nhà máy được sử dụng trong quá trình huấn luyện và đánh giá cho cả phương pháp phân nhiệm RTAW và phương pháp đề xuất, như mô tả trong Hình 3.3 và Hình 3.4. Các bố cục nhà máy này giúp kiểm tra khả năng của các phương pháp trong các tình huống môi trường khác nhau, như sự thay đổi trong số lượng robot, nhiệm vụ, và độ phức tạp của các nhiệm vụ. Bằng cách so sánh kết quả giữa phương pháp RTAW và phương pháp đề xuất, các đánh giá về hiệu quả và khả năng mở rộng của các phương pháp có thể được thực hiện khi triển khai trong các hệ thống thực tế.

4.2.1 Thiết lập cho quá trình huấn luyện mô hình

Quá trình huấn luyện mô hình được thực hiện với một tập hợp cố định gồm 32 robot và 16 nhiệm vụ trong hàng đợi $\mathcal{M}$. Để tăng tốc độ thu thập dữ liệu và đảm bảo tính đa dạng trong quá trình học, hệ thống triển khai đồng thời 8 môi trường song song. Mô hình được tối ưu hóa dựa trên thuật toán Proximal Policy Optimization (PPO) [10], kết hợp với thuật toán Adam [11] để đạt hiệu quả tối ưu hóa và tốc độ hội tụ. Vị trí ban đầu của robot được khởi tạo ngẫu nhiên trên bản đồ, với điều kiện ràng buộc rằng mỗi vị trí chỉ được chiếm bởi một robot duy nhất. Điều này giúp tránh xung đột ở trạng thái khởi tạo và tăng cường tính đa dạng cho dữ liệu đầu vào. Minh họa môi trường huấn luyện được trình bày trong Hình 3.3 và 3.4. Toàn bộ quá trình huấn luyện sử dụng bộ sinh số ngẫu nhiên cố định (seed) nhằm đảm bảo khả năng tái lập kết quả.

Hệ thống phần cứng sử dụng trong quá trình huấn luyện bao gồm một máy tính với bộ xử lý AMD Ryzen 7 4800H, 16GB RAM, và card đồ họa GTX 1650, hoạt động trên hệ điều hành Ubuntu 20.04. Việc triển khai mô hình sử dụng thư viện học sâu PyTorch trên nền tảng ngôn ngữ lập trình C++. Các siêu tham số được sử dụng trong quá trình huấn luyện được trình bày chi tiết trong Bảng 4.1. Hệ số học (learning rate) được đặt ở mức 3×10^{-4}, đảm bảo sự cân bằng giữa tốc độ hội tụ và tính ổn định trong quá trình tối ưu hóa. Hệ số γ và λ lần lượt được thiết lập là 0.99 và 0.95, tối ưu cho việc dự đoán giá trị dài hạn và điều chỉnh phạm vi hồi tưởng giá trị. Các giá trị này được tham khảo từ nghiên cứu trước đây [10, 11] và được tinh chỉnh qua thử nghiệm thực tế.

4.2.2 Thiết lập cho quá trình đánh giá mô hình

Sau khi hoàn tất quá trình huấn luyện, đồ án tiến hành đánh giá hiệu suất của phương pháp phân nhiệm đề xuất bằng cách so sánh với phương pháp phân nhiệm RTAW và phương pháp đấu giá dựa trên khoảng cách. Tiêu chí đánh giá chính là thời gian trung bình để một robot hoàn thành một nhiệm vụ. Trong phương pháp đấu giá, nhiệm vụ được lựa chọn dựa trên khoảng cách gần nhất từ robot được chọn đến điểm bắt đầu của các nhiệm vụ trong hàng đợi. Khoảng cách này được tính toán bằng thuật toán A*.

Quá trình đánh giá sử dụng ba nhóm số lượng robot: 10, 50 và 100 robot. Số

Bảng 4.1: Các siêu tham số sử dụng trong quá trình huấn luyện

Siêu tham số	**Giá trị**
Hệ số học (Learning rate)	3×10^{-4}
Kích thước mini batch	32
Số lượng epoch	16
Gamma (γ)	0.99
Lambda (λ)	0.95
Hệ số entropy (Entropy coefficient)	0.01
Hệ số giá trị (Value coefficient)	0.0002
Hệ số chính sách (Policy coefficient)	1.0
Hệ số cắt (Clip coefficient)	0.2
Mục tiêu KL (KL target)	0.01

lượng nhiệm vụ được thiết lập ở hai mức: 100 nhiệm vụ và 500 nhiệm vụ. Trong cả hai trường hợp, số lượng nhiệm vụ trong hàng đợi $\mathcal{M}$ được cố định ở mức 16 nhiệm vụ, giống như trong quá trình huấn luyện.

Với mỗi cấu hình đánh giá bao gồm phương pháp phân nhiệm, số lượng robot, và số lượng nhiệm vụ, thực hiện 10 lần chạy thử nghiệm. Để đảm bảo tính công bằng giữa các phương pháp, cả ba phương pháp phân nhiệm đều sử dụng chung 10 bộ sinh số ngẫu nhiên. Kết quả thu được từ quá trình đánh giá sẽ được trình bày chi tiết trong các phần sau.

4.3 Kết quả và thảo luận

4.3.1 Quá trình thực hiện đánh giá

Quá trình thực hiện đánh giá hiệu suất phân nhiệm được minh họa trong 4.1. Đầu tiên, vị trí của mỗi robot và nhiệm vụ trong hàng đợi được khởi tạo ngẫu nhiên với số lượng nhiệm vụ đã hoàn thành ban đầu là 0, và trạng thái của robot được đặt là *Ready*, như trong Hình 4.1(a). Các hình chữ nhật thể hiện vị trí của robot, với màu sắc biểu thị trạng thái: màu xanh lá là trạng thái *Ready*, màu vàng là trạng thái *Planning*, và màu xanh dương là trạng thái *Executing*. Số thứ tự trên mỗi robot biểu thị vị trí của chúng trong danh sách.

Sau khi khởi tạo, các robot lần lượt nhận nhiệm vụ từ khối phân nhiệm, chuyển trạng thái từ *Ready* sang *Planning* (màu xanh lá chuyển sang màu vàng), và nhận đường đi từ khối lập kế hoạch đường đi từ vị trí hiện tại đến vị trí nhiệm vụ. Các đường di chuyển được thể hiện bằng các đường màu nổi bật trong các Hình 4.1(b), 4.1(c), 4.1(d). Sau khi đến điểm nhiệm vụ và thực hiện nhiệm vụ, robot chuyển trạng thái từ *Planning* sang *Executing* (màu xanh dương trong Hình 4.1(c).

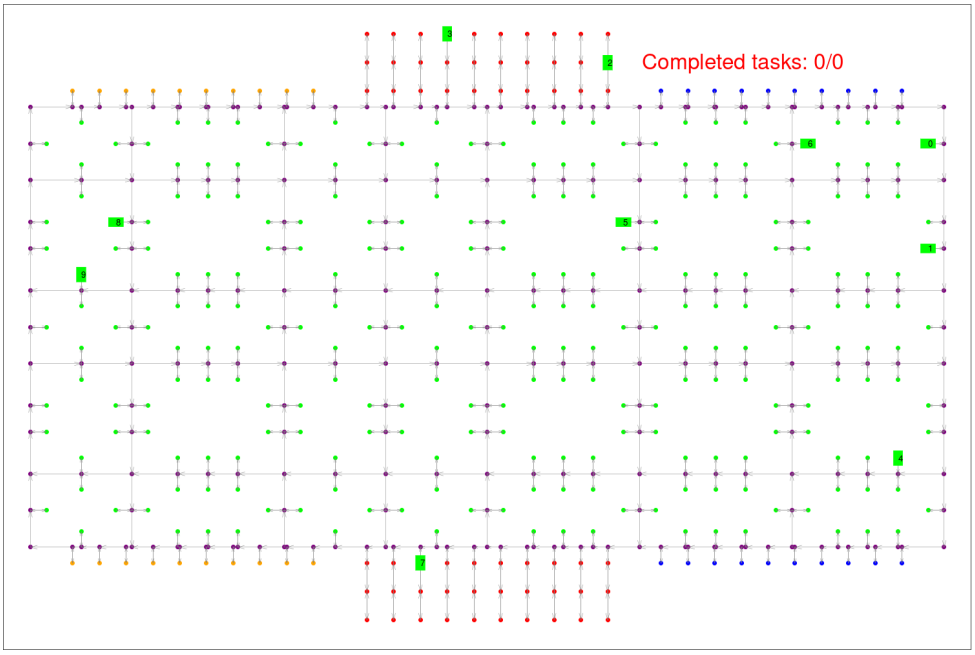
Khi hệ thống hoàn thành nhiệm vụ cuối cùng (ví dụ, nhiệm vụ thứ 100 trong 100 nhiệm vụ), thời gian thực hiện nhiệm vụ trung bình được ghi nhận như trong 4.1(d). Sau đó, vị trí của mỗi robot và nhiệm vụ trong hàng đợi được khởi tạo lại với bộ sinh số ngẫu nhiên mới, và trạng thái của các robot được đặt về *Ready*. Quá trình được lặp lại trong 10 lần thử nghiệm với các bộ sinh số ngẫu nhiên khác nhau. Video kết quả được đăng tại [1].

4.3.2 Kết quả đánh giá

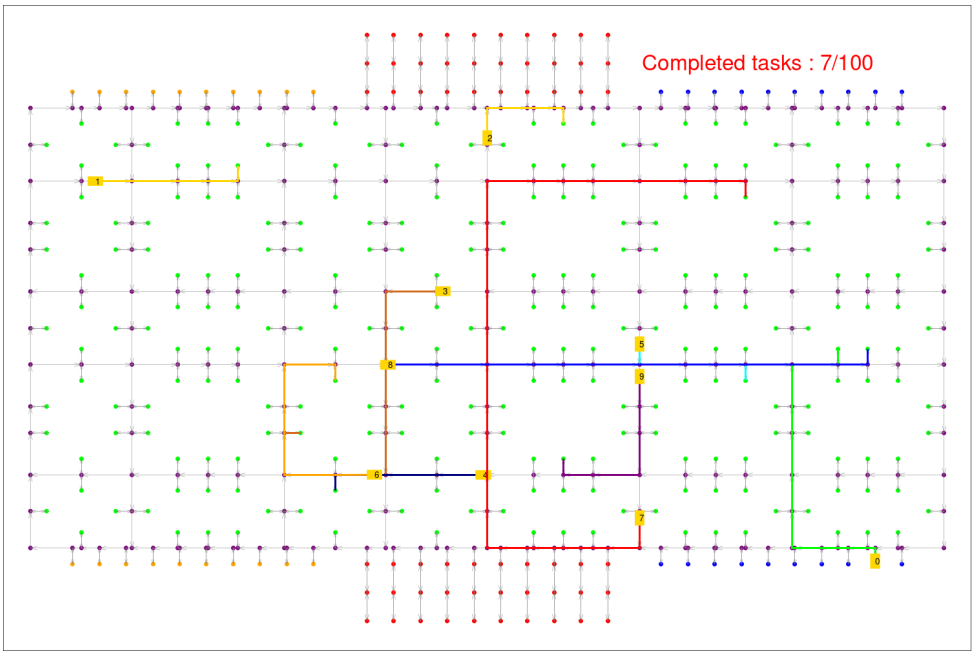
Kết quả thí nghiệm trên hai môi trường thử nghiệm được minh họa trong Hình 4.2 và 4.3. Trong các hình này, các phương pháp phân nhiệm được ký hiệu rõ ràng, với hai số trong dấu ngoặc lần lượt biểu thị số lượng robot và số lượng nhiệm vụ được thử nghiệm. Cụ thể, **A*** là phương pháp phân nhiệm dựa trên đấu giá khoảng cách, **RTAW** đại diện cho thuật toán RTAW, và **Our** là phương pháp phân nhiệm được đề xuất. Mỗi phương pháp được đánh giá dựa trên thời gian thực hiện nhiệm vụ trung bình, một chỉ số quan trọng để đo lường hiệu quả và khả năng mở rộng của các giải pháp trong môi trường robot đa tác vụ.

Kết quả cho thấy phương pháp A* có thời gian thực hiện nhiệm vụ trung bình cao nhất trong tất cả các trường hợp thử nghiệm, đặc biệt khi số lượng robot và nhiệm vụ tăng lên. Điều này phản ánh hạn chế rõ ràng của A* trong môi trường có quy mô lớn, khi mà chi phí tính toán trở thành vấn đề nghiêm trọng. Trong khi đó, phương pháp RTAW đạt hiệu suất khá tốt nhưng lại thể hiện sự dao động lớn về thời gian thực hiện, được minh họa qua độ dài của các hộp trong đồ thị. Đáng chú ý, khi quy mô hệ thống tăng, thời gian thực hiện của RTAW cũng tăng nhanh hơn, cho thấy nó gặp phải hạn chế về tính ổn định và khả năng mở rộng trong các

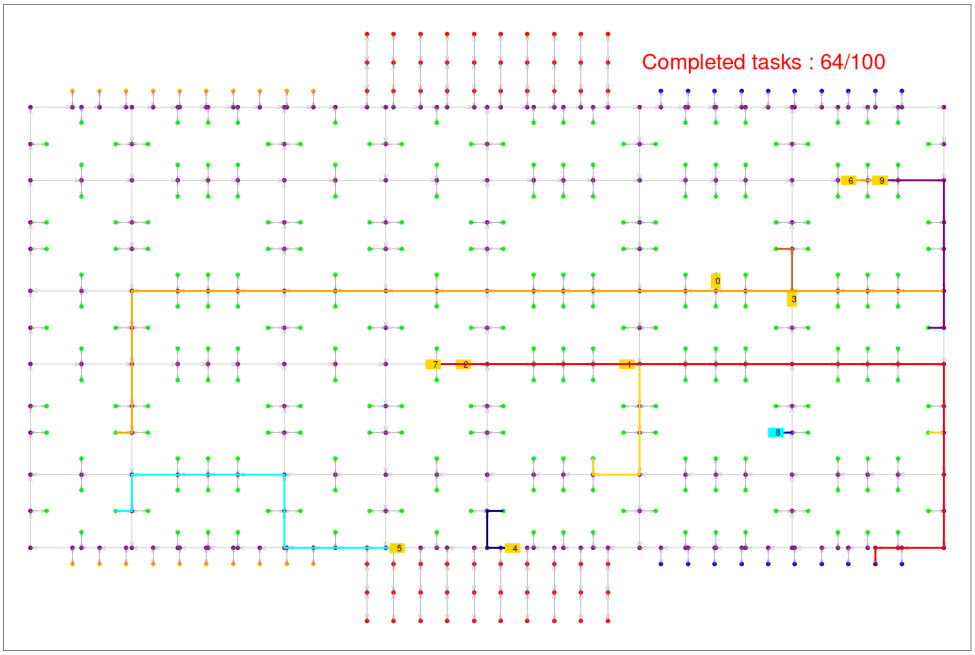
[1] https://s.net.vn/Bfq9



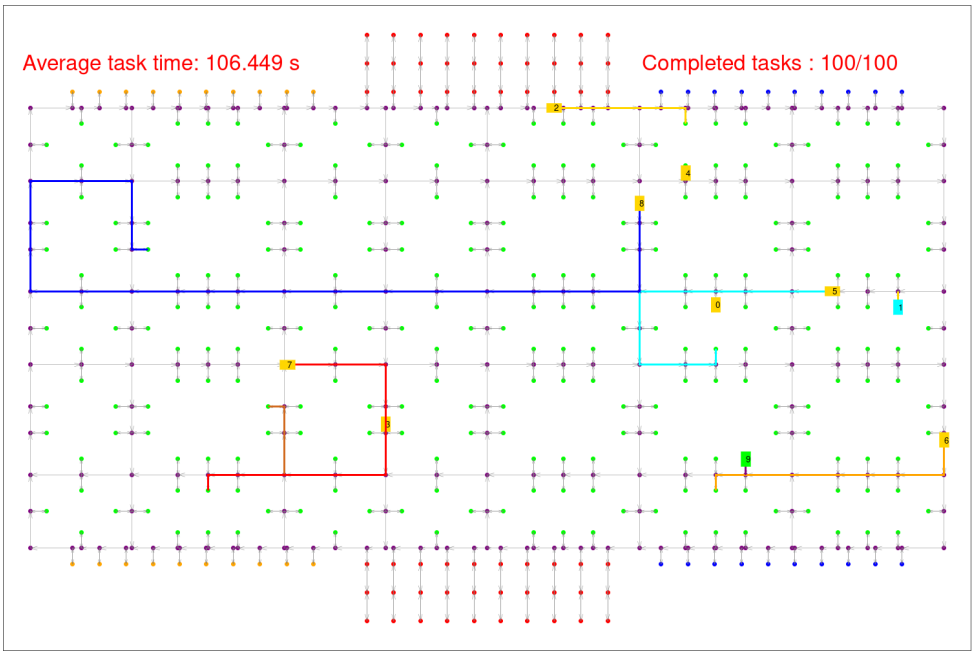
((a)) Khởi tạo hệ thống



((b)) Quá trình robot di chuyển đến điểm nhiệm vụ



((c)) Quá trình robot thực hiện yêu cầu tại điểm nhiệm vụ



((d)) Đưa ra thời gian thực hiện nhiệm vụ trung bình

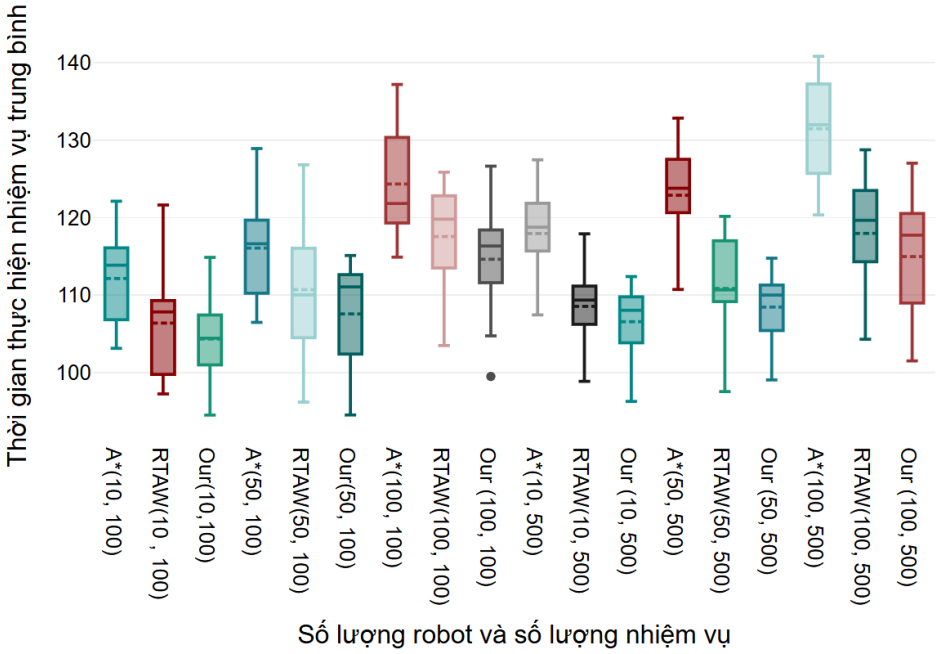
Hình 4.1: Quá trình thực hiện đánh giá hiệu suất mô hình phân nhiệm

bài toán lớn.

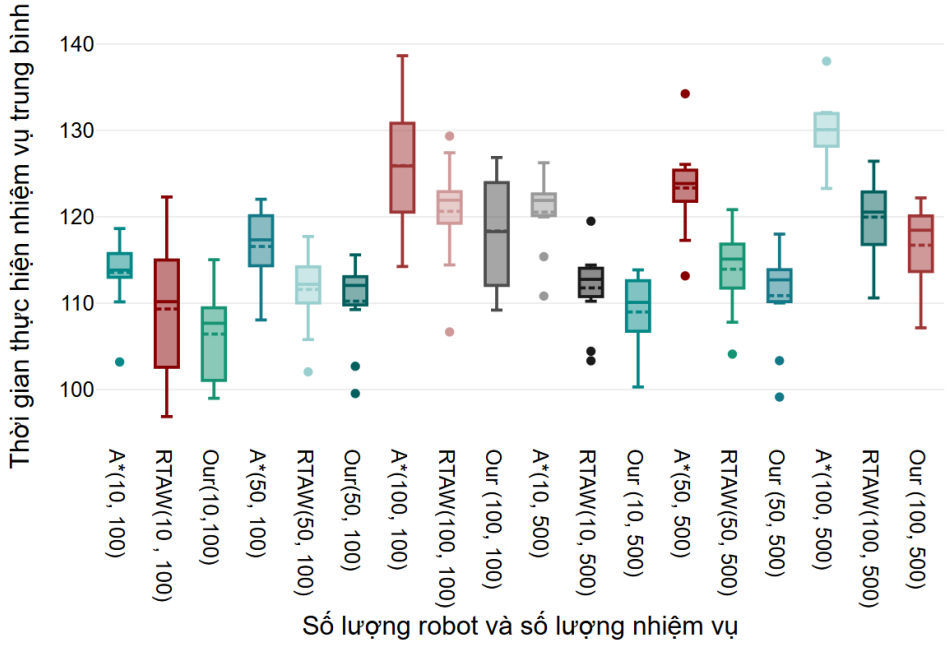
Ngược lại, phương pháp phân nhiệm đề xuất không chỉ đạt được thời gian thực hiện trung bình tương đường với RTAW mà còn duy trì sự ổn định cao hơn so với hai phương pháp còn lại được thể hiện rõ qua kích thước hộp nhỏ hơn trong hầu hết các trường hợp. So với RTAW, phương pháp của chúng tôi cải thiện thời gian thực hiện trung bình từ 2% đến 5%, với sự chênh lệch rõ rệt hơn khi hệ thống đạt quy mô lớn nhất (100 robot và 500 nhiệm vụ). Kết quả này khẳng định rằng phương pháp phân nhiệm đề xuất không chỉ khắc phục được hạn chế về sự ổn định của RTAW mà còn mang lại hiệu suất cao hơn trong việc ứng dụng thực tế vào môi trường nhà máy.

Tóm lại, phân tích từ các đồ thị đã chứng minh rằng phương pháp phân nhiệm đề xuất có khả năng tối ưu hóa hiệu quả, đặc biệt trong các môi trường phức tạp và có quy mô lớn. Bên cạnh việc cải thiện thời gian thực hiện, phương pháp của chúng tôi còn đảm bảo tính ổn định cao, một yếu tố quan trọng trong các hệ thống robot hiện đại. Những kết quả này mở ra triển vọng lớn cho việc áp dụng phương pháp trong các hệ thống robot đa tác vụ, nơi sự linh hoạt và hiệu quả là yếu tố then chốt.

Tuy nhiên, phương pháp đề xuất vẫn tồn tại một số hạn chế cần cải thiện trước khi triển khai thực tế. Mô hình hiện tại chưa xem xét tiêu hao năng lượng của robot hoặc tác động của khối lượng hàng hóa mà robot phải mang theo. Hệ thống giả định rằng robot luôn hoàn thành nhiệm vụ và không gặp lỗi vận hành, điều này chưa phản ánh hết các trường hợp thực tế. Việc truyền thông tin giữa robot và bộ điều khiển được giả định là hoàn hảo, không tính đến các trường hợp mất kết nối hoặc độ trễ. Ngoài ra, mô hình hiện tại giả định tất cả robot đều giống nhau và có vận tốc tương tự, trong khi trong thực tế, hệ thống thường bao gồm các robot không đồng nhất với các chức năng và đặc điểm khác nhau. Những hạn chế trên sẽ là định hướng cho các nghiên cứu trong tương lai nhằm cải thiện tính thực tế và hiệu quả của hệ thống.



Hình 4.2: Kết quả thí nghiệm trong môi trường thử nghiệm 1



Hình 4.3: Kết quả thí nghiệm trong môi trường thử nghiệm 2

Kết luận

Trong đồ án, mô phỏng hệ thống đa robot trong môi trường nhà máy đã được xây dựng, đồng thời áp dụng thành công thuật toán phân nhiệm dựa trên học tăng cường sâu kết hợp với mạng nơ-ron đồ thị vào bài toán phân nhiệm. Kết quả thu được từ mô hình cho thấy tính khả thi của phương pháp, mở ra cơ hội ứng dụng thực tế. Hệ thống được thiết kế bao gồm hai lớp làm việc quan trọng: lớp điều khiển với 5 khối chức năng và lớp vật lý với hệ thống đa robot cùng môi trường nhà máy.

Các thành phần của hệ thống, bao gồm hệ thống đa robot, môi trường nhà máy, khối quản lý trạng thái robot, khối quản lý nhiệm vụ, khối phân nhiệm, khối lên kế hoạch đường đi và khối điều khiển robot, đã được mô hình hóa chi tiết. Máy trạng thái hữu hạn được xây dựng cho khối quản lý trạng thái robot nhằm đảm bảo tính đúng đắn và khả năng áp dụng thực tế. Một phương pháp phân nhiệm mới đã được đề xuất, đảm bảo tính linh hoạt trong việc áp dụng cho nhiều tình huống khác nhau, bất kể số lượng robot, nhiệm vụ, hay điểm nhiệm vụ. Phương pháp này giúp hệ thống thích ứng hiệu quả với các thay đổi trong môi trường vận hành, đồng thời tối ưu hóa quá trình phân nhiệm.

Những hạn chế trong phương pháp phân nhiệm đề xuất đã gợi mở các hướng nghiên cứu và phát triển tiếp theo nhằm cải thiện mô hình. Việc tích hợp tiêu hao năng lượng và khối lượng hàng hóa vào quá trình ra quyết định phân nhiệm là một yếu tố quan trọng để tăng tính chính xác và thực tiễn. Phát triển các thuật toán phân nhiệm có khả năng ứng phó với lỗi hệ thống và sự gián đoạn trong truyền thông sẽ nâng cao độ ổn định trong điều kiện thực tế. Bên cạnh đó, mở rộng mô hình để xử lý hiệu quả các hệ thống đa robot không đồng nhất với các loại robot có chức năng khác nhau sẽ tăng khả năng áp dụng trong các môi trường phức tạp hơn.

Ngoài việc tiếp tục hoàn thiện phương pháp phân nhiệm, việc phát triển thêm phương pháp lên kế hoạch đường đi mới dựa trên học tăng cường sâu là một hướng đi cần thiết. Phương pháp này sẽ giúp hệ thống tối ưu hóa tốt hơn trong nhiều loại kiến trúc nhà máy khác nhau, cải thiện khả năng lên kế hoạch đường đi cho nhiều loại robot khác nhau, và đảm bảo tính tối ưu toàn hệ thống. Điều này sẽ tránh các xung đột tiềm năng, vượt trội hơn so với việc chỉ sử dụng thuật toán A*.

Tài liệu tham khảo

[1] Robert Zlot and Anthony Stentz. Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.

[2] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[3] Mohamed Badreldin, Ahmed Hussein, and Alaa Khamis. A comparative study between optimization and market-based approaches to multi-robot task allocation. *Advances in Artificial Intelligence*, 2013(1):256524, 2013.

[4] Aakriti Agrawal, Amrit Singh Bedi, and Dinesh Manocha. Rtaw: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1393–1399. IEEE, 2023.

[5] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[6] Zuxin Liu, Baiming Chen, Hongyi Zhou, Guru Koushik, Martial Hebert, and Ding Zhao. Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11748–11754. IEEE, 2020.

[7] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

[8] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

[9] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Sai Munikoti, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar, and Balasubramaniam Natarajan. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *IEEE transactions on neural networks and learning systems*, 2023.

[13] Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[14] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[15] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[17] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[18] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[19] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[20] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[21] Constantin-Valentin Pal and Florin Leon. Brief survey of model-based reinforcement learning techniques. In *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 92–97. IEEE, 2020.

[22] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6):1291–1307, 2012.

[23] Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

[24] TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[26] Kaushalya Madhawa and Tsuyoshi Murata. Active learning for node classification: An evaluation. *Entropy*, 22(10):1164, 2020.

[27] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[28] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[30] Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2151–2159, 2017.

[31] Zhilan Lou, Wanchen Jie, and Shuzhu Zhang. Multi-objective optimization for order assignment in food delivery industry with human factor considerations. *Sustainability*, 12(19):7955, 2020.

[32] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.

[33] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[34] Benjamin Riviere, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. *IEEE robotics and automation letters*, 5(3):4249–4256, 2020.

[35] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.