HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---




**Bùi Hữu Thành**

# NGHIÊN CỨU PHƯƠNG PHÁP PHÂN PHỐI TÁC VỤ CHO XE TỰ HÀNH TRÊN NỀN TẢNG ĐIỆN TOÁN BIÊN VÀ HỌC TĂNG CƯỜNG SÂU ĐA TÁC TỬ

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

**ĐỒ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT**

**(Theo định hướng ứng dụng)**

HÀ NỘI - 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---




**Bùi Hữu Thành**

# NGHIÊN CỨU PHƯƠNG PHÁP PHÂN PHỐI TÁC VỤ CHO XE TỰ HÀNH TRÊN NỀN TẢNG ĐIỆN TOÁN BIÊN VÀ HỌC TĂNG CƯỜNG SÂU ĐA TÁC TỬ

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

**ĐỒ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT**

**(Theo định hướng ứng dụng)**

**NGƯỜI HƯỚNG DẪN KHOA HỌC**

**TS. Đỗ Thị Liên**

HÀ NỘI - 2025

# LỜI CAM ĐOAN

Tôi xin cam đoan rằng đề tài nghiên cứu này là công trình do chính tôi thực hiện.

Toàn bộ số liệu, kết quả và nội dung trình bày trong đề án tốt nghiệp đều trung thực, phản ánh đúng quá trình nghiên cứu của tôi và chưa từng được công bố trong bất kỳ công trình nào khác.

Tôi hoàn toàn chịu trách nhiệm trước pháp luật và quy định của nhà trường về tính trung thực và nguyên bản của đề tài này.

Tác giả đề án tốt nghiệp

# LỜI CẢM ƠN

Trong suốt quá trình thực hiện đề tài tốt nghiệp, tôi đã nhận được sự hỗ trợ, chỉ dẫn và động viên quý báu từ nhiều cá nhân và tổ chức. Trước hết, tôi xin trân trọng cảm ơn quý thầy/cô trong khoa Đào tạo Sau đại học, khoa Công nghệ thông tin, Khoa Trí tuệ nhân tạo, đặc biệt là cô TS. Đỗ Thị Liên, người đã tận tình hướng dẫn, định hướng chuyên môn và luôn đồng hành cùng tôi trong suốt quá trình nghiên cứu và hoàn thành đề tài. Bên cạnh đó, tôi xin cảm ơn gia đình, bạn bè và đồng nghiệp đã luôn động viên, khích lệ và hỗ trợ tôi về tinh thần cũng như thời gian để tôi có thể hoàn thành tốt đề tài này.

Một lần nữa, tôi xin chân thành cảm ơn tất cả những sự giúp đỡ quý báu đó.

# MỤC LỤC

**LỜI CAM ĐOAN** ..... i
**LỜI CẢM ƠN** ..... ii
**MỤC LỤC** ..... iii
**DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT** ..... v
**DANH SÁCH BẢNG** ..... vi
**DANH SÁCH HÌNH VẼ** ..... vii
**MỞ ĐẦU** ..... 1
**Chương 1. Tổng quan về bài toán phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** 3
**1.1. Hệ thống xe tự hành** ..... 3
*1.1.1. Tổng quan về xe tự hành* ..... 3
*1.1.2. Bài toán phân phối nhiệm vụ cho xe tự hành trên nền tảng điện toán biên* ..... 4
**1.2. Điện toán biên và ưu điểm so với điện toán đám mây** ..... 6
**1.3. Mô hình kiến trúc hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** ..... 8
**1.4. Các hướng tiếp cận xây dựng hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** ..... 10
**1.5. Những thách thức và vấn đề còn tồn tại trong triển khai hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** ..... 15
**1.6. Kết luận chương 1** ..... 16
**Chương 2. Đề xuất phương pháp phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên và học tăng cường sâu đa tác tử** ..... 18
**2.1 Thiết kế kiến trúc hệ thống đề xuất** ..... 18
*2.1.1 Kiến trúc hệ thống* ..... 18
*2.1.2 Môi trường trong học tăng cường* ..... 19
*2.1.3 Các hành động và trạng thái* ..... 21
*2.2* **Mô hình toán học cho phân phối tác vụ trong kiến trúc đề xuất** ..... 23
*2.2.1 Phát biểu bài toán phân phối tác vụ* ..... 24
*2.2.2 Tối ưu hóa dựa trên mô hình học tăng cường sâu đa tác tử* ..... 31

**2.3 Xây dựng mô hình học tăng cường sâu đa tác tử cho bài toán tối ưu hóa phân phối tác vụ trong xe tự hành** ..... 36
*2.3.1 Kiến trúc mạng nơ-ron trong mô hình* ..... 36
*2.3.2 Quy trình huấn luyện mô hình* ..... 40
**2.4 Kết luận chương 2** ..... 43
**Chương 3. Kiểm nghiệm và đánh giá mô hình đề xuất trong tối ưu hóa phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** ..... 44
**3.1 Dữ liệu thực nghiệm** ..... 44
*3.1.1 Nhiệm vụ và tác vụ* ..... 44
*3.1.2 Xe tự hành và tác tử* ..... 46
*3.1.3 MEC Servers* ..... 47
*3.1.4 Mô phỏng mạng truyền thông* ..... 48
**3.2 Cài đặt thực nghiệm** ..... 50
**3.3 Đánh giá thực nghiệm** ..... 52
*3.3.1 Các chỉ số đánh giá* ..... 52
*3.3.2 Kết quả thực nghiệm* ..... 53
**3.4 Xây dựng thử nghiệm hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên** ..... 56
**3.5 Kết luận chương 3** ..... 57
**KẾT LUẬN CHUNG** ..... 58
**TÀI LIỆU THAM KHẢO** ..... 59
**PHỤ LỤC** ..... 61

# DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

| Viết tắt | Tiếng Anh | Tiếng Việt |
|---|---|---|
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| ML | Machine Learning | Học máy |
| MEC | Mobile Edge Computing | Điện toán biên di động |
| VEC | Vehicular Edge Computing | Điện toán biên cho phương tiện |
| V2V | Vehicle-to-Vehicle | Giao tiếp giữa các phương tiện |
| QoE | Quality of Experience | Chất lượng trải nghiệm |
| DRL | Deep Reinforcement Learning | Học tăng cường sâu |
| SA-DQN | Deep Q-Network with Simulated Annealing | Deep Q-Network kết hợp Simulated Annealing |
| MADRL | Multi-agent Deep Reinforcement Learning | Học tăng cường sâu đa tác tử |
| ARO | Artificial Rabbit Optimization | Thuật toán tối ưu hóa thỏ nhân tạo |
| GBO | Gradient-based Optimizer | Bộ tối ưu hóa dựa trên gradient |
| SWO | Spider Wasp Optimization | Thuật toán tối ưu hóa ong bắp cày nhện |
| DE | Differential Evolution | Thuật toán tiến hóa vi phân |
| ADMM | Alternating Direction Method of Multipliers | Phương pháp nhân tử theo hướng xen kẽ |
| CDN | Content Delivery Network | Mạng phân phối nội dung |
| AR | Augmented Reality | Thực tế tăng cường |
| FDSA | Flexible Network Architecture Dynamic Scheduling Algorithm | Thuật toán lập lịch động cho kiến trúc mạng linh hoạt |
| MDP | Markov Decision Process | Quy trình quyết định Markov |
| OSM | Open Street Map | Thư viện Open Street Map |
| IPO | Input Process Output | Mô hình Input – Process - Output |

# DANH SÁCH BẢNG

Bảng 1.1. Ưu điểm điện toán biên so với điện toán đám mây ..... 7
Bảng 1.2. Vai trò các lớp trong kiến trúc dịch vụ ba lớp cho mạng MEC ..... 9
Bảng 2.1. Ký hiệu và ý nghĩa các thành phần trong công thức mô hình bài toán ..... 23
Bảng 2.2. Bảng so sánh giữa Đơn tác tử và Đa tác tử ..... 31
Bảng 2.3 So sánh 3 thuật toán DQN, PPO, A2C ..... 37
Bảng 3.1. Phân bố xác xuất mức độ giao thông trên các đoạn đường ..... 49
Bảng 3.2. Cấu hình tham số mô phỏng môi trường ..... 49
Bảng 3.3. Tham số huấn luyện mô hình ..... 51
Bảng 3.4. Kết quả trung bình 100 lần mô phỏng các thuật toán đầu tiên ..... 54

# DANH SÁCH HÌNH VẼ

Hình 1.1. Hệ thống xe tự hành ..... 4
Hình 1.2. Mô hình IPO cho bài toàn phân phối nhiệm vụ cho xe tự hành ..... 5
Hình 1.3. Mô hình điện toán đám mây và điện toán biên ..... 6
Hình 1.4. Kiến trúc dịch vụ ba lớp trong mạng phương tiện hỗ trợ MEC ..... 8
Hình 2.1 Kiến trúc dịch vụ đề xuất ..... 18
Hình 2.2. Mô hình hóa môi trường học tăng cường sâu đa tác tử ..... 20
Hình 2.3. Sơ đồ kiến trúc mạng Double DQN ..... 39
Hình 2.4. Sơ đồ luồng huấn luyện mô hình cho các tác tử ..... 41
Hình 3.1. Mô phỏng một nhiệm vụ ..... 45
Hình 3.2. Mô hình hóa tập hợp tác vụ trên đường ..... 46
Hình 3.3. Kết quả huấn luyện mô hình MA-DDQN ..... 53
Hình 3.4. Kết quả tổng nhiệm vụ hoàn thành ..... 55
Hình 3.5. Kết quả tổng lợi nhuận đạt được ..... 55
Hình 3.6. Bản đồ mô phỏng khu vực Học viện Công nghệ Bưu Chính Viễn Thông ..... 56

# MỞ ĐẦU

Sự phát triển mạnh mẽ của công nghệ xe tự hành đang mở ra một kỷ nguyên mới cho giao thông thông minh, nơi các phương tiện có khả năng tự quan sát, phân tích và ra quyết định mà không cần sự can thiệp của con người. Để đảm bảo hoạt động an toàn và hiệu quả, các hệ thống xe tự hành phải xử lý một lượng lớn dữ liệu thời gian thực từ cảm biến, camera và các thiết bị hỗ trợ khác. Tuy nhiên, cách tiếp cận truyền thống dựa trên điện toán đám mây bộc lộ nhiều hạn chế về độ trễ, băng thông và khả năng mở rộng, gây ảnh hưởng trực tiếp đến khả năng phản hồi kịp thời của phương tiện trong các tình huống giao thông động và phức tạp.

Điện toán biên ra đời như một hướng tiếp cận tối ưu hơn, cho phép phân tán xử lý xuống gần vị trí phát sinh dữ liệu, từ đó giảm thiểu độ trễ và tiết kiệm tài nguyên mạng. Khi được áp dụng vào lĩnh vực giao thông thông minh, mô hình này trở thành một nền tảng quan trọng hỗ trợ xe tự hành xử lý tác vụ nhanh chóng và linh hoạt hơn. Tuy nhiên, do đặc thù của môi trường luôn biến động, đa tác vụ và phân tán, bài toán phân phối và lập lịch nhiệm vụ trở nên phức tạp hơn rất nhiều so với các hệ thống tập trung truyền thống. Trong bối cảnh đó, Học Tăng cường Sâu Đa Tác tử (MADRL) nổi lên như một phương pháp đầy tiềm năng để giải quyết các bài toán tối ưu phức tạp trong môi trường động. MADRL cho phép nhiều tác tử cùng học hỏi, thích nghi và phối hợp nhằm đưa ra các quyết định phân tải tối ưu theo thời gian thực. Nhờ khả năng học chính sách trong không gian trạng thái lớn và môi trường tương tác đa chiều, MADRL trở thành lựa chọn phù hợp để xây dựng một cơ chế phân phối tác vụ thông minh và linh hoạt cho hệ thống xe tự hành.

Xuất phát từ các yêu cầu thực tiễn và những tiến bộ của công nghệ, đề tài "Nghiên cứu phương pháp phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên và học tăng cường sâu đa tác tử." được thực hiện với mục tiêu xây dựng một mô hình lập lịch

có khả năng học thích nghi, phản hồi nhanh và tối ưu hóa hiệu suất xử lý trong môi trường điện toán biên. Đề án kỳ vọng đóng góp một hướng tiếp cận hiệu quả và khả thi cho bài toán phân phối tác vụ trong hệ thống hỗ trợ xe tự hành hiện đại.

# Chương 1. Tổng quan về bài toán phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên

Chương này trình bày tổng quan về các nền tảng công nghệ và xu hướng nghiên cứu liên quan đến việc phân phối tác vụ cho xe tự hành trong môi trường điện toán biên. Trong chương này, tôi sẽ giới thiệu khái quát về hệ thống xe tự hành và các hướng nghiên cứu hiện nay về tối ưu hóa xử lý tác vụ, đảm bảo độ trễ thấp và nâng cao độ tin cậy cho xe tự hành sẽ được phân tích nhằm làm rõ bối cảnh khoa học của bài toán. Đồng thời, tôi trình bày khái quát về bài toán phân phối nhiệm vụ cho xe tự hành trên nền tảng điện toán biên. Bên cạnh đó, chương này cũng trình bày vai trò của điện toán biên và những ưu điểm nổi bật của nó so với điện toán đám mây truyền thống trong việc hỗ trợ xử lý thời gian thực cho xe tự hành. Trên cơ sở đó, tôi mô tả mô hình kiến trúc hệ thống phân phối tác vụ trên nền tảng điện toán biên đang được áp dụng và các thách thức của bài toán.

## 1.1. Hệ thống xe tự hành

### *1.1.1. Tổng quan về xe tự hành*

Xe tự hành, hay còn gọi là xe kết nối và tự động hóa, là một công nghệ mang tính đột phá với khả năng nâng cao an toàn giao thông và tối ưu hóa hoạt động đô thị. Những phương tiện này có thể vận hành độc lập mà không cần sự can thiệp của con người, đồng thời kết nối với cơ sở hạ tầng và các phương tiện xung quanh để mở rộng khả năng nhận biết môi trường, giảm tai nạn và hạn chế ùn tắc *[1]*.

Xe tự hành có khả năng vận hành hoàn toàn độc lập, tự nhận biết môi trường và ra quyết định an toàn trong thời gian thực mà không cần tài xế. Nhờ tương tác với hạ tầng và các phương tiện khác, chúng mở rộng khả năng dự đoán giao thông, giảm tai nạn do lỗi con người và hạn chế ùn tắc. Công nghệ xe tự hành dựa trên sự kết hợp giữa trí tuệ nhân tạo và Internet vạn vật, cho phép phương tiện tự nhận thức và ra quyết định

trong môi trường giao thông phức tạp. Thông qua các mô hình giao tiếp V2V và V2I, cùng các công nghệ truyền thông như DSRC, LTE và 5G, xe có thể trao đổi dữ liệu độ trễ thấp để cập nhật và dự đoán tình huống giao thông *[2]*. Đồng thời, hệ thống tính toán trên xe xử lý dữ liệu cảm biến và ra quyết định thời gian thực, đảm bảo vận hành an toàn và hiệu quả.



**Hình 1.1. Hệ thống xe tự hành**

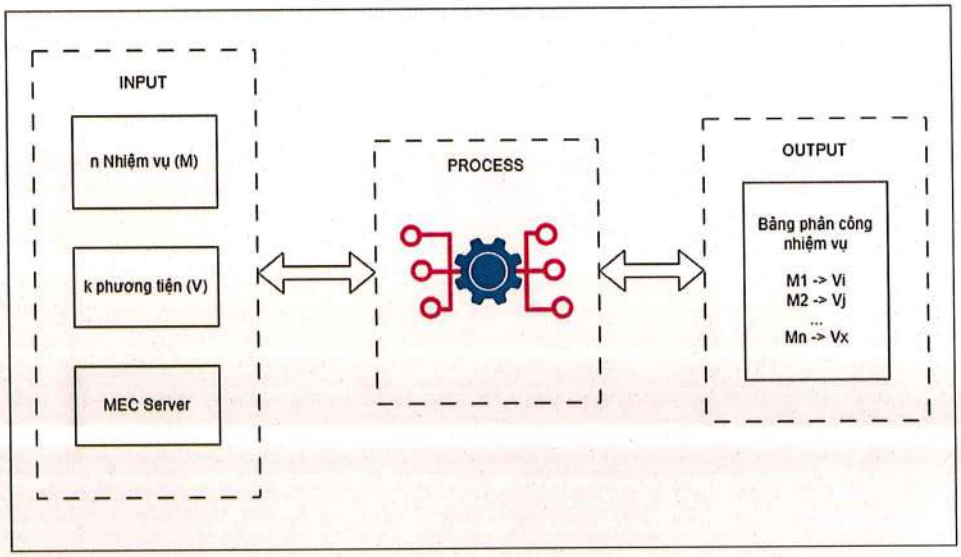
Các xe điện – nền tảng của xe tự hành – thường bị hạn chế về năng lượng và khả năng tính toán *[1]*. Khi các tác vụ phức tạp và yêu cầu thời gian thực tăng lên, dỡ tải tính toán sang hệ thống bên ngoài, đặc biệt là điện toán biên gần phương tiện, trở thành giải pháp cần thiết để giảm độ trễ và nâng cao hiệu suất. Tuy nhiên, trong môi trường giao thông động, việc quản lý dỡ tải và phân bổ tài nguyên rất phức tạp do sự thay đổi liên tục của phương tiện, mạng và yêu cầu xử lý, khiến bài toán tối ưu vẫn là một thách thức lớn *[4]*.

### *1.1.2. Bài toán phân phối nhiệm vụ cho xe tự hành trên nền tảng điện toán biên*

Nghiên cứu này tôi sẽ giải quyết bài toán lập lịch nhiệm vụ cho các xe tự hành. Hệ thống của tôi bao gồm tập hợp $V(\tau)$ chứa $k$ phương tiện, tập hợp chứa các MEC Server rải rác trên các tuyến đường để các phương tiện có thể dỡ tải tác vụ lên đó. Trong khoảng thời gian $\tau$ có n nhiệm vụ $M(\tau)$ cần được giải quyết. Các nhiệm vụ này sẽ có độ ưu tiên và một số nhiệm vụ phải được thực hiện trước các nhiệm vụ khác. Khi một xe

nhận một nhiệm vụ $M_i$ thì xe này cần thực hiện các tác vụ $T_i$ trên đường. Khi đó, các phương tiện sẽ cần cân nhắc thực hiện tác vụ $T_{i,j}$ tại xe hay dỡ tải tác vụ này lên các MEC Server xung quanh. Các phương tiện này sau khi hoàn thành các nhiệm vụ sẽ được nhận phần thưởng tương ứng với nhiệm vụ.

Trong bước xử lý, hệ thống tiếp nhận danh sách các nhiệm vụ cần thực hiện, danh sách các xe tự hành đang hoạt động trong khoảng thời gian quan sát, cùng thông tin về các MEC Server phân bố dọc tuyến đường. Dựa trên các thông tin đầu vào này, bộ lập lịch tiến hành xây dựng chiến lược phân phối nhiệm vụ sao cho mỗi nhiệm vụ được gán cho đúng một phương tiện và tuân thủ đầy đủ các ràng buộc về thứ tự phụ thuộc, độ ưu tiên và thời hạn hoàn thành nhiệm vụ.




**Hình 1.2. Mô hình IPO cho bài toán phân phối nhiệm vụ cho xe tự hành**

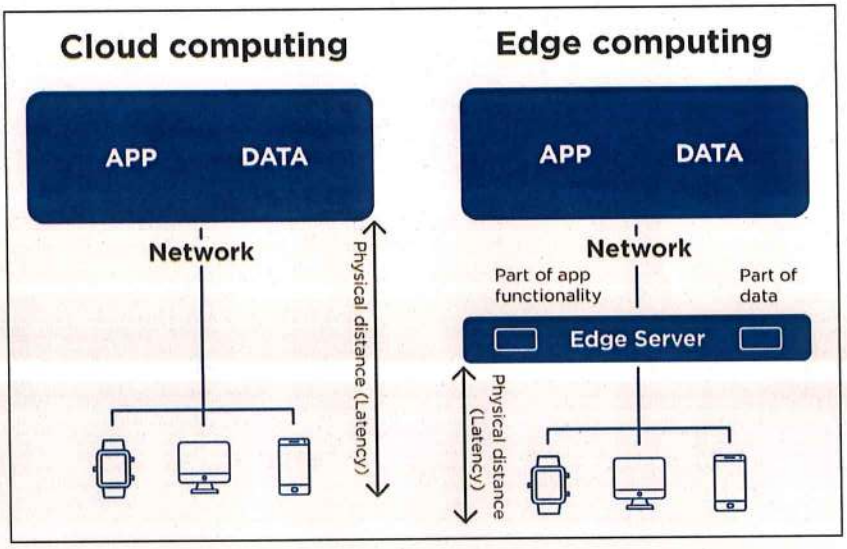
Đối với mỗi tác vụ thuộc một nhiệm vụ, phương tiện được chỉ định phải đưa ra quyết định tối ưu giữa việc xử lý cục bộ hoặc dỡ tải tác vụ lên một MEC Server lân cận. Quy trình này được thực hiện trong môi trường động, nơi trạng thái hệ thống liên tục thay đổi theo vị trí của xe, trạng thái tài nguyên MEC và số lượng nhiệm vụ tồn đọng.

Bộ xử lý trung tâm sẽ đánh giá chi phí xử lý, chi phí dỡ tải, độ trễ dự kiến và phần

thưởng tiềm năng để lựa chọn chiến lược hành động tối ưu. Kết quả của quá trình này là một phương án phân công nhiệm vụ cho các phương tiện và tập hợp quyết định xử lý/dỡ tải cho từng tác vụ, đảm bảo tối đa hóa số lượng nhiệm vụ hoàn thành đúng hạn và tối đa hóa tổng lợi nhuận thu được sau khi trừ chi phí xử lý và chi phí dỡ tải.

## 1.2. Điện toán biên và ưu điểm so với điện toán đám mây

Điện toán biên (Edge computing) và điện toán đám mây (Cloud computing) hoạt động phối hợp để mang lại giá trị lớn cho nhiều ngành công nghiệp. Khi kết hợp, hai nền tảng này có thể ngay lập tức hỗ trợ hoàn thành các tác vụ tốn nhiều tài nguyên, chẳng hạn như các hoạt động AI và ML quy mô lớn. Với sự phổ biến ngày càng tăng của công nghệ 5G và các tiến bộ khác gần đây, cả điện toán biên và điện toán đám mây đều được kỳ vọng sẽ tạo ra nhiều cơ hội mới trong các ngành công nghiệp hiện nay.




**Hình 1.3. Mô hình điện toán đám mây và điện toán biên**

Điện toán biên không phải là khái niệm mới, mà bắt nguồn từ các mạng phân phối nội dung (CDN) cuối những năm 1990, khi nội dung web và video được phục vụ từ các máy chủ gần người dùng. Đến đầu những năm 2000, mô hình này mở rộng sang lưu trữ và xử lý ứng dụng tại biên, hình thành các dịch vụ thương mại đầu tiên. Ngày nay, điện

toán biên đóng vai trò quan trọng trong xử lý dữ liệu thời gian thực và giảm độ trễ cho các ứng dụng hiện đại như xe tự hành, IoT, trợ lý giọng nói và quản lý giao thông.

Trong mô hình doanh nghiệp truyền thống, dữ liệu được gửi từ các điểm tạo về hệ thống tập trung để xử lý rồi trả kết quả lại, thông qua LAN, WAN hoặc internet. Khi số lượng thiết bị và dữ liệu tăng mạnh, mô hình này dễ gây quá tải mạng và làm chậm phản hồi. Điện toán biên giúp xử lý dữ liệu gần nguồn phát sinh, giảm tắc nghẽn mạng và nâng cao độ tin cậy của hệ thống.

Điện toán biên và điện toán đám mây có nhiều điểm tương đồng về khả năng tự động hóa, phân tích dữ liệu lớn, mô hình triển khai và hỗ trợ tuân thủ, đồng thời đều đóng vai trò quan trọng trong các ứng dụng tương lai như 5G và xe tự hành. Điện toán biên phù hợp cho xử lý thời gian thực, dữ liệu lớn khó truyền, kết nối hạn chế hoặc dữ liệu nhạy cảm; trong khi điện toán đám mây đảm nhiệm các tác vụ cần sức mạnh tính toán lớn, hợp nhất dữ liệu phân tán và khả năng mở rộng cao. Nhờ khả năng kết hợp linh hoạt, hai mô hình này bổ trợ cho nhau, giúp doanh nghiệp tối ưu hiệu quả xử lý dữ liệu trong nhiều kịch bản khác nhau.

Bảng dưới đây so sánh các ưu điểm chính của điện toán biên và điện toán đám mây, giúp làm rõ điểm mạnh và ứng dụng của từng mô hình.

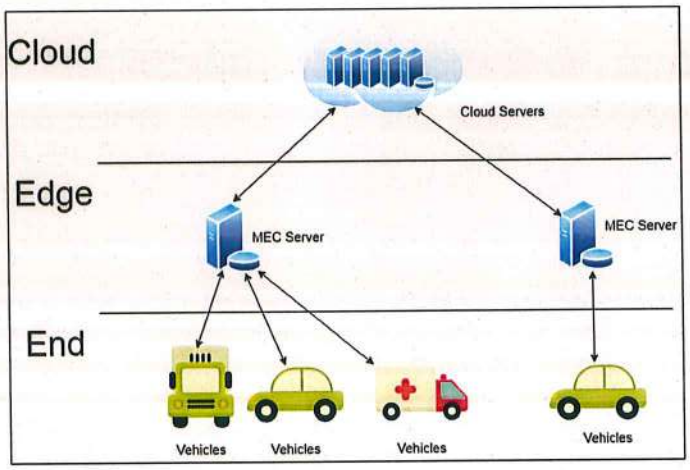
**Bảng 1.1. Ưu điểm điện toán biên so với điện toán đám mây**

| Tiêu chí | Điện toán biên | Điện toán đám mây |
|---|---|---|
| Độ trễ | Rất thấp vì xử lý ngay tại nguồn dữ liệu, phù hợp ứng dụng thời gian thực. | Cao hơn do phải gửi dữ liệu lên trung tâm dữ liệu, tùy thuộc khoảng cách và mạng. |
| Tải băng thông | Giảm đáng kể vì dữ liệu được xử lý/lọc tại biên trước khi gửi đi. | Có thể gây tắc nghẽn nếu lượng dữ liệu từ thiết bị lớn và truyền trực tiếp lên đám mây. |

| | | |
|---|---|---|
| Tính sẵn sàng | Hoạt động ổn định ngay cả khi mất kết nối internet; thiết bị vẫn xử lý cục bộ. | Phụ thuộc vào kết nối mạng; nếu mất mạng, nhiều dịch vụ đám mây dừng hoạt động. |
| Bảo mật và quyền riêng tư | Dữ liệu nhạy cảm được giữ tại biên, giảm rủi ro khi truyền qua mạng. | Dữ liệu tập trung tại đám mây, dễ quản lý nhưng tăng rủi ro nếu bị tấn công vào hệ thống trung tâm. |
| Khả năng mở rộng | Mở rộng phân tán bằng cách thêm các node biên, phù hợp với hệ thống lớn theo khu vực. | Mở rộng mạnh về tài nguyên tính toán và lưu trữ nhờ hạ tầng lớn của nhà cung cấp đám mây. |

Điện toán biên đưa xử lý đến gần nguồn dữ liệu, giúp giảm độ trễ, tăng tốc phản hồi và phù hợp với các ứng dụng thời gian thực như AR, robot hay xe tự hành. Việc xử lý tại biên giúp giảm tải băng thông, tăng tính sẵn sàng khi mất kết nối và nâng cao bảo mật nhờ giữ dữ liệu nhạy cảm tại chỗ. Về mở rộng, điện toán biên phát triển theo mô hình phân tán, trong khi đám mây mở rộng mạnh mẽ nhờ hạ tầng tập trung.

## 1.3. Mô hình kiến trúc hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên




**Hình 1.4. Kiến trúc dịch vụ ba lớp trong mạng phương tiện hỗ trợ MEC**

Kiến trúc dịch vụ ba lớp trong mạng phương tiện hỗ trợ MEC gồm tầng End, Edge

và Cloud. Tầng End bao gồm các xe tự hành hoặc thiết bị người dùng, nơi sinh ra dữ liệu và thực hiện các tác vụ nhẹ, yêu cầu phản hồi tức thời. Tầng Edge đặt các máy chủ MEC gần người dùng, chịu trách nhiệm xử lý các tác vụ có độ trễ thấp, tính toán cục bộ và giảm băng thông trước khi gửi dữ liệu lên cloud. Tầng Cloud đảm nhận lưu trữ dữ liệu lớn, xử lý các tác vụ phức tạp và huấn luyện mô hình, sau đó nạp kết quả xuống Edge và End để hỗ trợ hoạt động hiệu quả của hệ thống. Bảng dưới đây tóm tắt kiến trúc dịch vụ ba lớp trong mạng phương tiện hỗ trợ MEC

**Bảng 1.2. Vai trò các lớp trong kiến trúc dịch vụ ba lớp cho mạng MEC**

| Tầng | Vai trò | Chức năng chính |
|---|---|---|
| End (Xe tự hành) | Sinh dữ liệu và thực hiện tác vụ cục bộ | Xử lý tác vụ nhẹ, yêu cầu thời gian thực ngay tại thiết bị |
| Edge (MEC Server) | Xử lý gần người dùng, giảm độ trễ | Thực hiện tác vụ cục bộ, tối ưu hóa băng thông trước khi gửi dữ liệu lên cloud |
| Cloud | Lưu trữ, xử lý dữ liệu lớn và huấn luyện mô hình | Xử lý các tác vụ phức tạp, phân tích tổng hợp, nạp mô hình và thông tin xuống Edge/End |

Mô hình phân phối tác vụ cho xe tự hành trong triển khai thực tế có thể được tóm tắt thành bốn hướng chính.

- Xử lý cục bộ trên xe: Xe tự hành thực hiện toàn bộ tác vụ ngay trên thiết bị của mình. Ưu điểm là phản hồi tức thời và không phụ thuộc vào mạng, nhưng hạn chế về năng lực tính toán và năng lượng.
- Phân tải lên MEC server: Các tác vụ nặng hơn được gửi đến các máy chủ MEC gần xe để xử lý. Cách tiếp cận này giúp giảm độ trễ so với cloud và tận dụng tài nguyên biên, tuy nhiên vẫn cần kết nối mạng ổn định với MEC.
- Phân tải lên Cloud server: Tác vụ phức tạp hoặc yêu cầu lưu trữ lớn được gửi lên cloud để xử lý. Phương pháp này tận dụng khả năng tính toán gần như vô hạn và

xử lý dữ liệu quy mô lớn, nhưng lại có độ trễ cao và phụ thuộc vào kết nối Internet.

- Phân tải sang xe khác: Xe chia sẻ tác vụ với các xe xung quanh có tài nguyên rảnh. Phương pháp này tận dụng tài nguyên phân tán và giảm tải cho MEC hoặc cloud, nhưng đòi hỏi cơ chế phối hợp chặt chẽ và đảm bảo kết nối mạng giữa các xe.

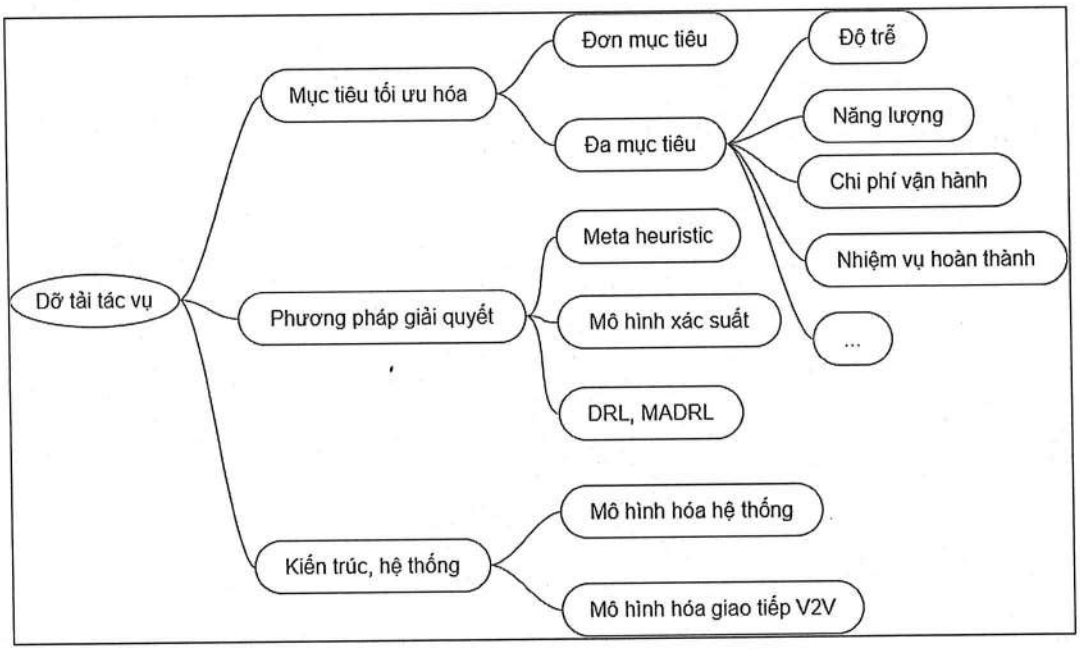
Trong hầu hết các triển khai thực tế, các xe tự hành thường ưu tiên phân tải tác vụ lên MEC server thay vì gửi trực tiếp lên cloud hay chia sẻ với các xe khác. Lý do chính là MEC server vừa đặt gần xe, giúp giảm đáng kể độ trễ so với cloud, vừa có đủ tài nguyên tính toán để xử lý các tác vụ nặng hơn khả năng của xe. Phương thức này cho phép các xe hoàn thành nhiệm vụ một cách hiệu quả, an toàn và ổn định, đồng thời giảm tải cho hệ thống cloud và hạn chế sự phức tạp trong cơ chế phối hợp V2V. Do vậy, MEC thường được xem là điểm trung tâm trong chiến lược phân phối tác vụ của xe tự hành.

## 1.4. Các hướng tiếp cận xây dựng hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên

Bài toán phân phối tác vụ cho xe tự hành, hay còn gọi là bài toán dỡ tải tính toán, là một bài toán tối ưu hóa phức tạp nhằm xác định nơi và cách thức xử lý các tác vụ tính toán chuyên sâu do phương tiện tạo ra *[3], [4]*. Trong quá trình vận hành, xe tự hành liên tục thu thập và xử lý lượng lớn dữ liệu từ cảm biến, camera và hệ thống điều khiển, dẫn đến nhu cầu tính toán cao vượt quá khả năng xử lý nội tại của xe. Vì vậy, việc quyết định tác vụ nào nên được xử lý trực tiếp trên xe và tác vụ nào cần chuyển sang các nút tính toán biên hoặc đám mây trở thành yếu tố then chốt để đảm bảo hiệu suất, độ trễ thấp và an toàn hệ thống. Đây là một thách thức lớn do môi trường giao thông luôn thay đổi, yêu cầu thuật toán phân phối tác vụ phải thích ứng linh hoạt và tối ưu trong mọi tình huống.

Bài toán phân tải tác vụ cho xe tự hành hiện đang trở thành một chủ đề nghiên cứu trọng tâm trong lĩnh vực mạng và tính toán biên, khi nhu cầu xử lý dữ liệu thời gian

thực và ra quyết định nhanh trở nên ngày càng quan trọng.




**Hình 1.5. Tổng quan các hướng nghiên cứu về phân phối tác vụ hiện nay**

Các nghiên cứu trong lĩnh vực này tập trung vào ba hướng chính: mục tiêu tối ưu hóa như độ trễ, năng lượng, độ tin cậy và hiệu quả tài nguyên; phương pháp giải quyết với các thuật toán và mô hình phân tải thích nghi trong môi trường động và kiến trúc và điều kiện hệ thống, bao gồm MEC, MEC–Cloud, hợp tác V2V và các đặc thù về mạng, phương tiện. Ba hướng này kết hợp tạo nền tảng cho các giải pháp phân tải tác vụ hiệu quả trong hệ thống xe tự hành.

Các nghiên cứu hiện đại về phân tải tác vụ cho xe tự hành cho thấy xu hướng dịch chuyển mạnh từ việc tối ưu hóa đơn mục tiêu sang các bài toán tối ưu đa mục tiêu phức tạp hơn, phản ánh đúng tính đa chiều của yêu cầu trong môi trường vận hành thực tế. Trọng tâm trước hết là giảm thiểu độ trễ xử lý và tiêu thụ năng lượng, vốn là hai tiêu chí then chốt trong hầu hết các mô hình tối ưu *[1], [5], [6]*. Bên cạnh đó, các mục tiêu tối ưu hóa khác bao gồm giảm tỷ lệ thất bại của tác vụ *[1]*, tối đa hóa số lượng tác vụ được

tiếp nhận có tính đến mức độ ưu tiên cũng như các yêu cầu về chất lượng dịch vụ (QoS) như thời hạn hoàn thành và yêu cầu bảo mật *[7]*. Ngoài ra, một số hướng tiếp cận chú trọng đến tối ưu hóa hiệu quả sử dụng tài nguyên mạng *[4]* và khả năng trích xuất đặc trưng dữ liệu có tính khái quát cao, nhằm ứng phó với các tình huống dữ liệu có sự dịch chuyển miền hoặc ngữ nghĩa, mà không cần truy cập vào dữ liệu kiểm thử trong giai đoạn huấn luyện *[8]*. Trong các hệ thống kết hợp máy bay không người lái (UAV), các nghiên cứu còn hướng tới tối ưu hóa tổng thể việc lập lịch tác vụ và triển khai tác vụ trên UAV *[5]*, nhằm đảm bảo tính linh hoạt, khả năng chịu lỗi, và hiệu quả tác vụ trong môi trường động và phân tán cao

Trong hướng nghiên cứu về kiến trúc và điều kiện hệ thống, các công trình hiện đại tập trung xử lý những đặc thù phức tạp và năng động của môi trường mạng phương tiện, nơi việc dỡ tải tác vụ bị chi phối bởi nhiều yếu tố hệ thống liên quan. Một trong những chủ đề quan trọng là bài toán phụ thuộc tác vụ, trong đó các mô hình dỡ tải mới được đề xuất nhằm phản ánh đúng sự phụ thuộc phức tạp giữa các nhiệm vụ, điều mà nhiều nghiên cứu trước đây thường giản lược. Bên cạnh đó, các giải pháp phối hợp tài nguyên để tối ưu hóa triển khai và lập lịch tác vụ cho người dùng di động đang nhận được sự quan tâm lớn *[1], [2]*. Điều này cũng dẫn đến bài toán tối ưu khai thác tài nguyên tính toán nhàn rỗi từ các phương tiện dịch vụ thông qua V2V nhằm tăng tính linh hoạt dỡ tải. Các kiến trúc phân cấp End–Edge–Cloud cũng được tích hợp để xây dựng cơ chế dỡ tải phù hợp, cho phép chuyển các tác vụ phức tạp lên đám mây xử lý nhằm nâng cao hiệu năng và khả năng mở rộng của hệ thống *[1], [2], [4]*. Ngoài ra, vấn đề bảo mật ngày càng được chú trọng, với xu hướng ứng dụng Blockchain để tăng cường quyền riêng tư và tính toàn vẹn dữ liệu trong môi trường VEC *[4]*. Cuối cùng, nhiều nghiên cứu hướng tới cải thiện khả năng thích ứng của hệ thống trong môi trường động, thông qua các phương pháp lập lịch nhạy cảm với thời hạn, mô hình hóa các yếu tố biến động như mật độ phương tiện và tính dị thể của máy chủ MEC *[3]*, cũng như xem xét việc ứng dụng kiến trúc Open RAN trong hệ thống giao thông thông minh để tối ưu hóa việc tổ chức

và phân tải tác vụ *[10]*.

Các nghiên cứu gần đây về phân phối tác vụ trong môi trường điện toán biên cho xe tự hành chủ yếu khai thác sức mạnh của Học tăng cường sâu (DRL) và các thuật toán tối ưu hóa phức tạp để đối phó với sự năng động và không chắc chắn của hệ thống. Trong số đó, các biến thể như SA-DQN được đề xuất nhằm tối ưu hóa đồng thời tỷ lệ thất bại và tiêu thụ năng lượng trong điều kiện có sự phụ thuộc tác vụ và tính di động của phương tiện *[1]*; MADRL được ứng dụng trong các mô hình phi tập trung để tăng hiệu quả phối hợp giữa các tác tử và cải thiện khả năng mở rộng *[4]*. Bên cạnh đó, nhiều thuật toán DRL như DQN, DDPG, MADDPG, MAAC, MAPPO, COMA, QMIX hay SAC cũng được triển khai nhằm tối ưu hóa độ trễ, phân bổ tài nguyên và tải tính toán trong các kiến trúc tập trung, phi tập trung hoặc phân cấp *[2]*. Song song với đó, các kỹ thuật tối ưu hóa như Probabilistic Computation Offloading (PCO) dựa trên ADMM được dùng để tìm xác suất phân bổ tối ưu nhằm giảm độ trễ *[3]*; Mixed Integer Linear Programming (MILP) được áp dụng để tối đa hóa số tác vụ được tiếp nhận dưới các ràng buộc QoS, lưu trữ và bảo mật *[7]*; các thuật toán tối ưu hóa đa mục tiêu lai như DVCG-MWOA giúp giảm thiểu độ trễ và năng lượng trong hệ UAV-MEC *[6]*; các thuật toán siêu việt hai lớp như GBO, GNDO, SWO, DE kết hợp với thuật toán tham lam ở lớp dưới được sử dụng cho bài toán lập lịch và triển khai UAV *[5]*; và cuối cùng, thuật toán FDSA được phát triển để giảm thời gian lập lịch và tối ưu hóa ánh xạ tác vụ trong các cụm UAV thông qua phân lớp linh hoạt và cơ chế đánh giá năng lực xử lý *[9]*.

**Bảng 1.4. Ưu điểm, hạn chế của các hướng nghiên cứu gần đây**

| Tiêu chí | SA-DQN | MADRL | PCO | MILP |
|---|---|---|---|---|
| Mục tiêu tối ưu hóa | Tỷ lệ thất bại dỡ tải và tổng năng lượng tiêu thụ | Tổng năng lượng tiêu thụ và tỷ lệ hoàn thành tác vụ | Độ trễ dịch vụ | Số lượng tác vụ được đáp ứng |
| Ưu điểm | Tốc độ hội tụ | Khả năng mở rộng | Có thể độc lập | Cung cấp giải |

| | | | | |
|---|---|---|---|---|
| | nhanh hơn so với các thuật toán RL truyền thống | tốt hơn trong các mạng lưới lớn; hiệu quả với môi trường phương tiện thay đổi | đưa ra quyết định lập lịch trực tuyến; khả năng mở rộng tốt | pháp tối ưu trong điều kiện lý tưởng |
| Hạn chế | Nhạy cảm với tham số của SA; độ phức tạp tính toán cao; dễ bị mắc kẹt ở cực tiểu cục bộ; chỉ xét đến các tác vụ độc lập | Độ phức tạp tính toán cao | Chi phí lưu trữ và truyền thông cao; không phù hợp với môi trường có nhiều thay đổi | Giả định các tác vụ độc lập, tuyến tính hóa bài toán dựa trên giả định các tác vụ hoàn thành đúng hạn, không phù hợp với bài toán thực tế |

Trong bài toán phân phối tác vụ cho các xe tự hành trong môi trường điện toán biên phương tiện, tôi đã chọn Học tăng cường sâu đa tác tử (MADRL) bởi khả năng xử lý hiệu quả tính động, tính phân tán và sự phức tạp vốn có của hệ thống. Khác với các phương pháp tối ưu hóa truyền thống hoặc DRL tập trung, MADRL cho phép nhiều tác tử học cách ra quyết định một cách tự chủ và phối hợp, từ đó nâng cao khả năng mở rộng khi số lượng xe tăng và giảm sự phụ thuộc vào một bộ điều khiển trung tâm. Cơ chế đào tạo tập trung nhưng thực thi phân tán giúp hệ thống duy trì được sự phối hợp toàn cục trong quá trình học, đồng thời thích ứng linh hoạt với môi trường xe cộ có tính di động cao và không tĩnh. Bên cạnh đó, MADRL đã chứng minh hiệu quả vượt trội trong việc tối ưu hóa các chỉ số như giảm độ trễ, giảm tiêu thụ năng lượng, và cải thiện tỷ lệ hoàn thành tác vụ, đặc biệt đối với các tác vụ nhạy cảm với thời gian. Những ưu điểm này là cơ sở để tôi sử dụng MADRL để giải quyết bài toán phân phối tác vụ cho xe tự hành trong hệ thống VEC quy mô lớn và đầy biến động trong đề án này.

## 1.5. Những thách thức và vấn đề còn tồn tại trong triển khai hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên

Các nghiên cứu trước đây về phân phối tác vụ trong môi trường điện toán biên cho xe tự hành (VEC) vẫn còn tồn tại nhiều hạn chế cả về kiến trúc và kỹ thuật. Các phương pháp tập trung thường gặp khó khăn về khả năng mở rộng và bùng nổ không gian hành động trong môi trường xe cộ năng động, trong khi các phương pháp phi tập trung truyền thống lại thiếu cơ chế phối hợp hiệu quả giữa các tác tử [4]. Để giải quyết vấn đề này, các nghiên cứu gần đây chuyển hướng sang học tăng cường sâu đa tác tử (MADRL) nhằm tận dụng tính phi tập trung và tích hợp cơ chế hợp tác [4]. Bên cạnh đó, nhiều nghiên cứu trước đây chỉ tập trung vào tối ưu một mục tiêu như độ trễ hoặc tiêu thụ năng lượng, trong khi các phương pháp mới hướng đến tối ưu hóa đa mục tiêu, bao gồm cả tỷ lệ thất bại và hiệu quả năng lượng [1]. Ngoài ra, nhiều mô hình truyền thống giả định môi trường tĩnh và các tác vụ độc lập, không phản ánh đúng tính di động của phương tiện và sự phụ thuộc giữa các tác vụ. Các phương pháp mới đã kết hợp DAGs và dữ liệu di động vào mô hình để tăng tính thực tiễn [1]. Hơn nữa, việc giảm tải trước đây chỉ giới hạn ở các trạm RSU hoặc thiết bị OBU, trong khi các nghiên cứu hiện đại đã mở rộng phạm vi đến các phương tiện hợp tác (SeVs) và máy chủ đám mây [1]. Cuối cùng, vấn đề dịch chuyển phân phối trong truyền thông định hướng tác vụ vốn bị bỏ qua, nay được giải quyết bằng cách áp dụng các phương pháp như Information Bottleneck (IB) và Invariant Risk Minimization (IRM), giúp trích xuất đặc trưng có khả năng khái quát hóa mà không cần thông tin kiểm thử [8].

Dù các phương pháp tối ưu phân phối tác vụ hiện tại đã đạt được nhiều thành tựu đáng kể, phần lớn các nghiên cứu vẫn chưa xem xét đầy đủ yếu tố như thời hạn hoàn thành không cố định. Trong môi trường xe tự hành, đây là loại tác vụ phổ biến và có ảnh hưởng lớn đến chiến lược phân phối. Ví dụ, một tác vụ thu thập dữ liệu môi trường (như đo chất lượng không khí, phân tích tín hiệu giao thông) vẫn mang lại giá trị nếu hoàn

thành muộn hơn dự kiến, nhưng sẽ có giá trị cao hơn nếu xử lý kịp thời. Tương tự, việc cập nhật bản đồ thời gian thực hoặc chia sẻ cảnh báo tắc đường không yêu cầu xử lý tức thời nhưng có giá trị giảm dần theo thời gian. Việc không tính đến đặc tính linh hoạt về thời hạn hoàn thành có thể dẫn đến phân bổ tài nguyên không hiệu quả, gây lãng phí hoặc làm giảm chất lượng dịch vụ toàn hệ thống. Do đó, cần thiết có những mô hình tối ưu hóa mới trong phân phối tác vụ cho xe tự hành, có khả năng đánh giá giá trị thời gian của tác vụ để đưa ra quyết định phân phối hợp lý và thích ứng hơn trong môi trường động và phân tán như VEC.

## 1.6. Kết luận chương 1

Trong chương này, tôi đã trình bày tổng quan về xe tự hành và vai trò của điện toán biên trong việc hỗ trợ xử lý dữ liệu thời gian thực cho các hệ thống giao thông thông minh. Tiếp đến, tôi đã trình bày một cách khái quát về bài toán phân phối nhiệm vụ cho xe tự hành trên nền tảng điện toán biên. Chương này cũng đã trình bày rõ vai trò của điện toán biên và điện toán đám mây trong việc hỗ trợ xử lý tác vụ cho hệ thống xe tự hành. Trong phần này các hướng nghiên cứu chính giải quyết bài toán, bao gồm nghiên cứu về các mục tiêu tối ưu, nghiên cứu về phương pháp giải quyết và nghiên cứu về kiến trúc hệ thống. Các phương pháp giải quyết bài toán gồm tối ưu hóa truyền thống, các thuật toán phân tán/heuristic và các phương pháp học tăng cường sâu. Mỗi hướng có ưu nhược điểm riêng, từ khả năng tìm nghiệm tối ưu đến khả năng thích ứng trong môi trường động. Điều này cho thấy cần có giải pháp cân bằng giữa hiệu quả, chi phí và tính linh hoạt. Các nghiên cứu trước về phân phối tác vụ trong VEC dù đa dạng nhưng vẫn còn hạn chế về khả năng phối hợp, tính động của môi trường, mô hình phụ thuộc tác vụ và chưa xem xét đầy đủ đặc tính thời hạn linh hoạt của nhiệm vụ. Do đó, cần những mô hình tối ưu hóa mới có thể đánh giá giá trị theo thời gian của tác vụ và thích ứng tốt hơn trong môi trường xe tự hành năng động và phân tán. Trong đề án này, tôi tập trung giải quyết bài toán lập lịch nhiệm vụ cho các xe tự hành dưới các ràng buộc về thứ tự phụ

thuộc giữa nhiệm vụ, thời hạn hoàn thành và quyết định xử lý cục bộ hay dỡ tải lên MEC. Mục tiêu là tối đa hóa số nhiệm vụ hoàn thành và lợi nhuận thu được trong một môi trường động và phức tạp. Trong chương tiếp theo, tôi sẽ đề xuất mô hình toán học cho bài toán phân phối nhiệm vụ cho xe tự hành, xây dựng phương pháp lập lịch phù hợp và trình bày cách mô phỏng môi trường để đánh giá hiệu quả giải pháp.

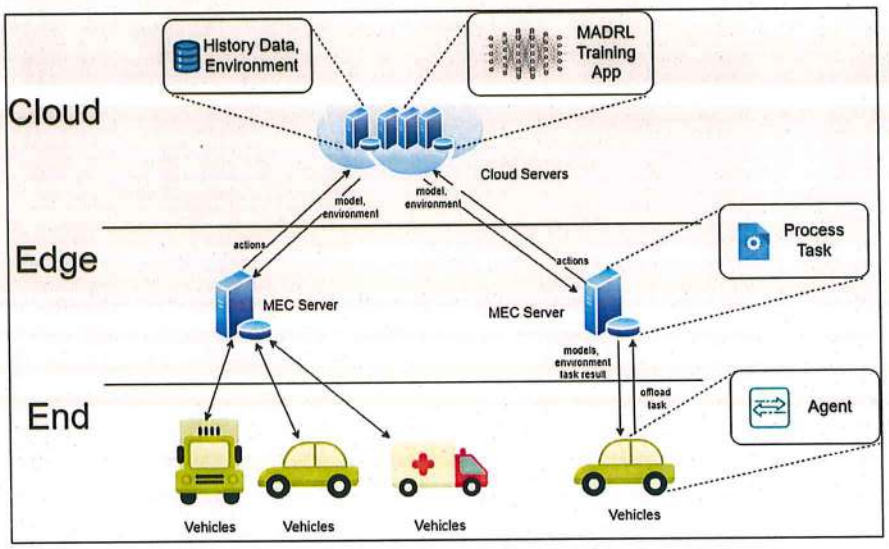
# Chương 2. Đề xuất phương pháp phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên và học tăng cường sâu đa tác tử

Trong chương này, tôi sẽ trình bày phương pháp đề xuất nhằm giải quyết bài toán phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên bằng cách ứng dụng học tăng cường sâu đa tác tử. Trước hết, tôi sẽ mô tả kiến trúc hệ thống đề xuất và mô hình hóa bài toán tối ưu hóa. Tiếp theo là môi trường học tăng cường với các đối tượng như phương tiện, MEC Server, bản đồ giao thông, nhiệm vụ, tác vụ,... Đồng thời, môi trường cũng đề cập đến các hành động của các tác tử, các trạng thái trong môi trường và đặc biệt là hàm phần thưởng để đánh giá hiệu quả của các chiến lược học tăng cường. Cuối chương tôi sẽ trình bày kiến trúc mạng nơ-ron, quy trình huấn luyện mô hình MADRL và cách tích hợp chúng vào cơ chế lập lịch để tối ưu hiệu năng phân phối tác vụ.

## 2.1 Thiết kế kiến trúc hệ thống đề xuất

### *2.1.1 Kiến trúc hệ thống*

Kiến trúc đề xuất cho hệ thống phân phối tác vụ của xe tự hành được thiết kế theo mô hình ba lớp, bao gồm Cloud, Edge và End.




**Hình 2.1 Kiến trúc dịch vụ đề xuất**

Mục tiêu chính của kiến trúc này là tối ưu hóa việc xử lý tác vụ với các yêu cầu về độ trễ khác nhau, từ các tác vụ không yêu cầu thời gian thực đến các tác vụ cần phản hồi tức thời. Mô hình ba lớp này cho phép phân tán thông minh tài nguyên tính toán và tăng cường khả năng đáp ứng nhanh cho các xe tự hành.

Lớp Cloud chịu trách nhiệm triển khai các ứng dụng non-RealTime, tức là những tác vụ không yêu cầu phản hồi tức thời, với độ trễ xử lý trên 1 giây. Tại đây, các chức năng chính bao gồm quản lý tổng thể hệ thống, phân tích dữ liệu lớn, huấn luyện mô hình trí tuệ nhân tạo và thực hiện các tác vụ không đòi hỏi phản hồi ngay lập tức. Cloud kết nối với lớp Edge để trao đổi thông tin cấu hình và chính sách điều khiển mạng, từ đó hỗ trợ điều phối và tối ưu hóa các tác vụ tại các lớp biên và End.

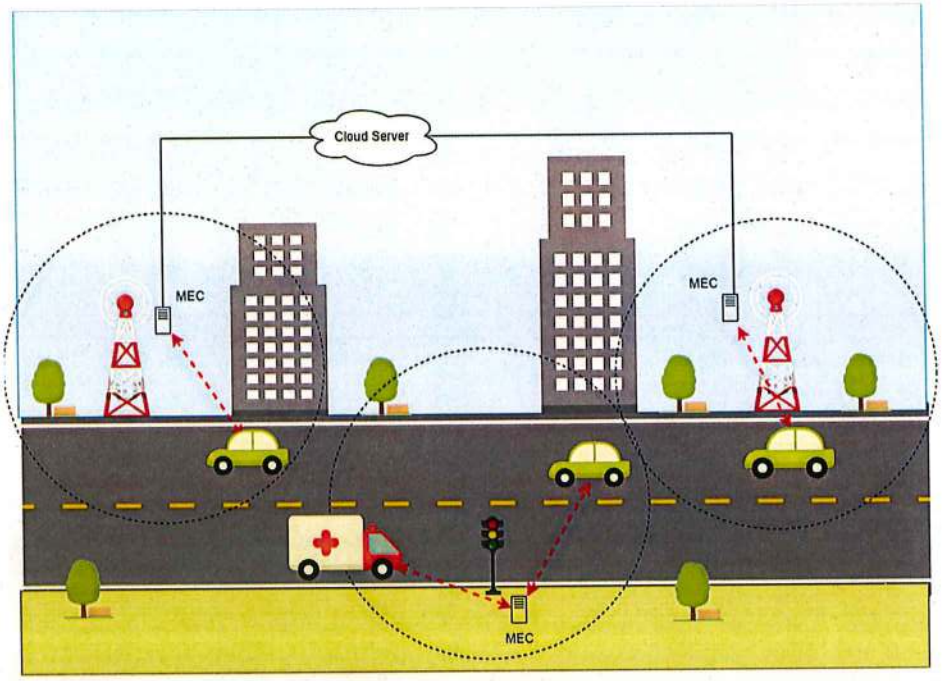
Lớp Edge được triển khai gần với người dùng hơn và bao gồm các MEC servers, chịu trách nhiệm xử lý các tác vụ Near-RealTime với độ trễ từ 10ms đến 1 giây. Near-RealTime tại Edge triển khai các ứng dụng thực hiện điều khiển mạng gần thời gian thực, tối ưu phân phối tác vụ và quản lý tài nguyên biên. Nhờ đó, Edge giảm tải cho Cloud, đồng thời tăng khả năng phản hồi nhanh đối với các yêu cầu từ xe tự hành.

Lớp End bao gồm các xe tự hành và cơ sở hạ tầng mạng trực tiếp phục vụ xe tự hành, nơi các tác vụ cần phản hồi tức thời được thực hiện với độ trễ dưới 10ms. Các tác vụ tại lớp này bao gồm thu thập dữ liệu cảm biến, xử lý tín hiệu và duy trì thông tin truyền thông xe-mạng. Các MEC Server trong Edge được phân bố thuận tiện, đảm bảo việc xử lý tác vụ vừa nhanh vừa hiệu quả, đồng thời hỗ trợ cơ chế dỡ tải hợp tác giữa các lớp.

### *2.1.2 Môi trường trong học tăng cường*

Môi trường trong bài toán phân tải tác vụ cho nhóm xe tự hành được xây dựng dựa trên bốn thành phần cơ bản bao quát các yếu tố quyết định hoạt động của hệ thống. Thứ nhất là các nhiệm vụ với giá trị lợi ích và giới hạn thời gian hoàn thành khác nhau.

Kế đến là các xe tự hành đóng vai trò agent đưa ra quyết định xử lý cục bộ hoặc gửi tác vụ lên MEC server. Tiếp theo là các MEC server có năng lực tính toán hạn chế đảm nhận xử lý các tác vụ nhận được. Cuối cùng là các Cloud server chủ yếu phục vụ huấn luyện mô hình và phân phối chính sách tối ưu xuống hệ thống.




**Hình 2.2. Mô hình hóa môi trường học tăng cường sâu đa tác tử**

Sự phối hợp giữa bốn thành phần này hình thành môi trường vận hành thực tế, nơi các agent có thể học và ra quyết định tối ưu trong giới hạn thời gian và nguồn lực có sẵn.

- *Nhiệm vụ*: mỗi nhiệm vụ trong hệ thống đều có một giá trị lợi ích xác định và một giới hạn thời gian hoàn thành cụ thể, phản ánh mức độ quan trọng và lợi ích mà nhiệm vụ mang lại cho hệ thống khi được hoàn thành đúng hạn. Các nhiệm vụ có thể khác nhau về độ phức tạp và yêu cầu tài nguyên tính toán, tạo ra thách thức trong việc phân phối và xử lý nhằm tối ưu tổng lợi ích trong giới hạn thời gian. Đồng thời, các nhiệm vụ có sự phụ thuộc lẫn nhau: một nhiệm vụ chỉ có thể được

thực hiện sau khi tất cả các nhiệm vụ mà nó phụ thuộc đã hoàn thành, đòi hỏi cơ chế điều phối thông minh để đảm bảo tuần tự và hiệu quả trong xử lý.

- *Xe tự hành:* Các xe tự hành đóng vai trò là agent trong môi trường, chịu trách nhiệm quyết định lựa chọn nhiệm vụ để thực hiện. Đồng thời, mỗi xe tự hành quyết định giữa việc xử lý tác vụ trực tiếp trên xe (xử lý cục bộ) hoặc gửi tác vụ lên các MEC server để được xử lý. Quyết định này dựa trên năng lực tính toán của xe, năng lực tính toán của MEC Server và khối lượng công việc của các tác vụ. Trong mô hình này, các xe không gửi tác vụ trực tiếp lên Cloud cũng như không phân chia tác vụ cho các xe khác, tập trung tối ưu hóa sự phối hợp với MEC server.
- *MEC server:* Các MEC server là các nút xử lý biên có khả năng tính toán hạn chế, đóng vai trò nhận và xử lý các tác vụ được gửi từ xe tự hành. MEC server giúp giảm độ trễ so với việc xử lý tập trung trên Cloud. Đồng thời, với năng lực tính toán vượt trội so với xe tự hành, việc dỡ tải tác vụ từ xe lên MEC Server giúp xử lý trong thời gian ngắn để ra quyết định kịp thời và gần với thời gian thực. Để giảm độ phức tạp của bài toán, tôi giả định rằng các máy chủ MEC đều có năng lực tính toán không giới hạn, do đó có khả năng tiếp nhận và xử lý toàn bộ các yêu cầu từ các xe tự hành mà không phát sinh các vấn đề liên quan đến hiệu năng hay độ trễ xử lý.
- *Cloud server:* Trong mô hình này, các Cloud server tồn tại nhưng không được sử dụng để xử lý trực tiếp các tác vụ từ xe. Cloud chủ yếu phục vụ cho việc huấn luyện mô hình học tăng cường sâu và quản lý dữ liệu tổng thể, sau đó phân phối các chính sách tối ưu xuống xe và MEC server. Điều này đảm bảo rằng các agent trên xe có thể ra quyết định hiệu quả trong thời gian thực mà không phụ thuộc vào kết nối trực tiếp với Cloud.

### *2.1.3 Các hành động và trạng thái*

Trong môi trường mô phỏng, có 5 xe tự hành bắt đầu ở 5 vị trí khác nhau trên bản đồ. Khi bắt đầu, có 25 nhiệm vụ cần hoàn thành, mỗi nhiệm vụ có giá trị lợi ích và giới hạn thời gian riêng, đồng thời có thể phụ thuộc lẫn nhau. Hành động đầu tiên của mỗi xe là chọn nhiệm vụ trong 25 nhiệm vụ để thực hiện. Mỗi xe lần lượt lựa chọn nhiệm vụ theo chiến lược tối ưu sao cho tổng số lượng nhiệm vụ hoàn thành là lớn nhất. Sau khi chọn nhiệm vụ, xe sẽ tìm đường đi ngắn nhất tới vị trí nhiệm vụ, kết hợp với các điều kiện giao thông được mô phỏng trên bản đồ. Trên đường đi, các xe sẽ gặp phải các tác vụ nhỏ phát sinh (ví dụ: kiểm tra tình trạng cảm biến, thu thập dữ liệu môi trường, cập nhật bản đồ địa phương hoặc xử lý cảnh báo giao thông). Khi xử lý các tác vụ này, mỗi xe có hai lựa chọn hành động: tự xử lý tại xe hoặc dỡ tải lên MEC server để xử lý. Lựa chọn này cần cân nhắc chi phí, năng lực tính toán tại xe, độ trễ và lợi ích tổng thể. Như vậy, trong mô hình này, mỗi xe có hai cấp độ hành động: (1) lựa chọn nhiệm vụ để thực hiện và (2) quyết định phương án xử lý các tác vụ trên đường, giữa xử lý cục bộ và dỡ tải lên MEC. Cấu trúc này tạo nên môi trường phức tạp nhưng thực tế, giúp huấn luyện các tác tử phối hợp tối ưu trong việc chọn nhiệm vụ và ra quyết định xử lý tác vụ.

Trong môi trường mô hình phân tải tác vụ cho xe tự hành, trạng thái được mô tả dựa trên bốn thành phần chính. Thứ nhất là thông tin về các xe, bao gồm vị trí của các xe, khả năng tính toán trên xe để quyết định dỡ tải tác vụ. Thứ hai là trạng thái các nhiệm vụ, với giá trị lợi ích, thời hạn hoàn thành, kích thước và các phụ thuộc lẫn nhau. Thứ ba là trạng thái của MEC server bao gồm vị trí trên bản đồ và khả năng tính toán tại biên. Cuối cùng, toàn bộ thông tin này được kết hợp để tạo thành trạng thái tổng quan của môi trường, giúp xe tự hành lựa chọn nhiệm vụ và quyết định dỡ tải tác vụ lên MEC Server.

- *Trạng thái xe*: Mỗi xe lưu trữ thông tin về vị trí hiện tại trên bản đồ, năng lực xử lý cục bộ và danh sách các nhiệm vụ đã chọn
- *Trạng thái nhiệm vụ*: Bao gồm danh sách 25 nhiệm vụ, với thông tin chi tiết về giá trị lợi ích, thời hạn hoàn thành và các phụ thuộc giữa các nhiệm vụ.

- *Trạng thái MEC server:* Năng lực xử lý của MEC Server giúp xe tự hành quyết định dỡ tải tác vụ lên MEC Server hay xử lý tại xe.
- *Trạng thái giao thông và bản đồ*: Bao gồm tình trạng bận rộn của các đoạn đường, các tác vụ nhỏ phát sinh trên lộ trình (kiểm tra cảm biến, thu thập dữ liệu môi trường...)

## 2.2 Mô hình toán học cho phân phối tác vụ trong kiến trúc đề xuất

Với mô hình hệ thống đề xuất ở trên, ta giả sử trong khoảng thời gian $\tau$ tập hợp $V(\tau)$ chứa $k$ phương tiện khả dụng và danh sách n nhiệm vụ $M(\tau)$ cần được giải quyết. Trong thực tế, số lượng phương tiện khả dụng và nhiệm vụ cần xử lý có thể thay đổi theo thời gian. Tuy nhiên, để thuận tiện cho việc mô phỏng và huấn luyện thuật toán, ta có thể quy chuẩn tập hợp $V(\tau)$ chứa $k$ phương tiện và danh sách nhiệm vụ $M(\tau)$ về một danh sách cố định với số lượng xác định trước. Việc này giúp chuẩn hóa đầu vào cho mô hình, đảm bảo rằng mỗi lượt huấn luyện có cùng cấu trúc trạng thái và hành động, từ đó thuật toán học tăng cường có thể học và so sánh hiệu quả giữa các hành động khác nhau một cách nhất quán. Đồng thời, việc cố định này cũng giúp đơn giản hóa việc đánh giá tổng lợi ích hệ thống, dễ dàng kiểm tra các kịch bản phân tải tác vụ và so sánh kết quả giữa các chiến lược khác nhau.

Trước khi trình bày chi tiết phát biểu bài toán và cách xây dựng hàm phần thưởng cho thuật toán học tăng cường, tôi liệt kê các ký hiệu được sử dụng trong công thức cùng giải thích ý nghĩa của từng ký hiệu để đảm bảo tính rõ ràng và thống nhất trong mô hình.

**Bảng 2.1. Ký hiệu và ý nghĩa các thành phần trong công thức mô hình bài toán**

| Ký hiệu | Ý nghĩa |
|---|---|
| $M(\tau)$ | Danh sách các nhiệm vụ cần giải quyết |
| $M_i$ | Nhiệm vụ thứ $i$ trong $M(\tau)$ |
| $K_i$ | Danh sách các tác vụ cần thực hiện trong nhiệm vụ $M_i$ |

| | |
|---|---|
| $t_{i,j}$ | Tác vụ thứ $j$ trong danh sách các tác vụ $K_i$ |
| $N(\tau)$ | Tập nghiệm cho việc phân công n nhiệm vụ trong $M(\tau)$ cho k phương tiện |
| $\text{profit}_i$ | Lợi nhuận cơ bản của nhiệm vụ $i$ |
| $T_i$ | Thời hạn (deadline) của nhiệm vụ $i$ |
| $t_i$ | Thời điểm thực tế nhiệm vụ $i$được hoàn thành |
| $\mathbf{1}_{\{\text{complete}_i\}}$ | Chỉ số thể hiện nhiệm vụ $i$ được hoàn thành hay không. Chỉ số này bằng 1 nếu hoàn thành và 0 nếu ngược lại |
| $\mathbf{1}_{\{\text{offload}_{i_j}\}}$ | Chỉ số thể hiện tác vụ thứ $j$ trong nhiệm vụ thứ $i$ được dỡ tải lên MEC Server hay không |
| $\text{route}_i$ | Danh sách các tuyến đường khi xe thực hiện nhiệm vụ $i$ |
| $\text{distance}_\ell$ | Khoảng cách của đoạn đường $\ell$ |
| $\text{average_speed}_\ell$ | Tốc độ trung bình trên đoạn đường $\ell$ |
| $\text{Dep}(i)$ | Tập các nhiệm vụ phụ thuộc vào nhiệm vụ $i$ |
| $r_1, r_2, r_3, r_4$ | Các hệ số dương sử dụng cho hàm tính phần thưởng |
| $R_i$ | Tổng thưởng cho nhiệm vụ $i$ |
| $R_{total}$ | Tổng thưởng cho toàn bộ hệ thống |

### *2.2.1 Phát biểu bài toán phân phối tác vụ*

Một nhiệm vụ có thể được biểu diễn bởi một bộ 5 thành phần như sau

$$M_i \triangleq \langle s_i, e_i, T_i, P_i, M_i^- \rangle \tag{2.1}$$

Trong đó $M_i$ là nhiệm vụ thứ $i$ trong danh sách $M(\tau)$; $s_i$ là điểm bắt đầu nhận nhiệm vụ, $e_i$ là điểm kết thúc nhiệm vụ; $T_i$ là thời hạn hoàn thành của nhiệm vụ; $P_i$ là lợi nhuận cơ bản của phương tiện khi hoàn thành nhiệm vụ và $M_i^- \in M(\tau)$ là tập hợp các nhiệm vụ tiền nhiệm mà phải hoàn thành trước $M_i$. Các nhiệm vụ có $M_i^- = \emptyset$ có thể thực hiện ngay lập tức mà không cần chờ đợi.

Trong quá trình thực hiện các nhiệm vụ trên đường, các phương tiện phải xử lý nhiều tác vụ nhỏ phát sinh, bao gồm tự động hóa các chức năng điều khiển, quản lý hệ thống giải trí, cũng như các chức năng bổ trợ khác nhằm nâng cao trải nghiệm người dùng,... Những tác vụ này tạo ra nhu cầu lựa chọn chiến lược xử lý giữa việc thực hiện trực tiếp trên xe hoặc dỡ tải các tác vụ lên MEC server. Để đơn giản hóa mô hình, các tác vụ trên đường không được sinh ngẫu nhiên trong quá trình di chuyển của xe, mà được tạo sẵn trước. Khi các phương tiện đi qua các vị trí này, chúng bắt buộc phải thực hiện các tác vụ đã định, giúp việc mô phỏng và huấn luyện agent trở nên ổn định và có kiểm soát hơn.

Gọi tập $K_i$ là danh sách các tác vụ cần thực hiện trong nhiệm vụ $M_i$. Khi đó một tác vụ $t_{i,j} \in K_i$ sẽ phụ thuộc vào các yếu tố

$$t_{i,j} \triangleq \langle d_j, c_j \rangle \tag{2.2}$$

Trong đó $d_j$ và $c_j$ lần lượt là khối lượng dữ liệu cần truyền tải (Data size) và số chu kỳ CPU cần thiết để xử lý tác vụ (Computation cycles) trong tác vụ thứ j của nhiệm vụ $M_i$.

*Trạng thái giao thông*

Trong mô hình mô phỏng, điều kiện giao thông được chia thành năm loại và giữ cố định trong suốt khoảng thời gian τ.

- Lưu thông tự do: khi các phương tiện di chuyển với tốc độ tối đa mà không gặp chướng ngại vật.
- Lưu thông ổn định: khi phương tiện di chuyển đều đặn nhưng tốc độ giảm nhẹ do mật độ xe tăng hoặc phải nhường đường cho xe khác, mặc dù không xảy ra tắc nghẽn.
- Lưu thông chậm: khi mật độ phương tiện tăng đáng kể, buộc các xe thường

xuyên giảm tốc hoặc dừng ngắn, nhưng giao thông vẫn tiếp tục di chuyển.

- Lưu thông tắc nghẽn: phương tiện di chuyển rất chậm hoặc dừng hoàn toàn, gây ra chậm trễ lớn ngay cả trên quãng đường ngắn.
- Tắc nghẽn nghiêm trọng: trạng thái giao thông gần như tê liệt, các phương tiện đứng yên hoặc di chuyển rất chậm, thường xuất hiện vào giờ cao điểm hoặc do tai nạn.

Mỗi đoạn đường được gán một hệ số trạng thái giao thông $bs_i$ biểu thị mức độ tắc nghẽn (busy state). Dựa trên hệ số này, tôi sử dụng thuật toán Dijkstra để tìm tuyến đường ngắn nhất cho việc thực hiện nhiệm vụ và điều hướng phương tiện.

*Nghiệm của bài toán phân công nhiệm vụ*

Gọi $N(\tau)$ là một tập nghiệm cho việc phân công n nhiệm vụ trong $M(\tau)$ cho k phương tiện trong $V(\tau)$. Ta sẽ biểu diễn $N(\tau)$ dưới dạng mảng có n phần tử, mỗi phần tử này chứa thông tin của phương tiện (vid) và thứ tự thực hiện nhiệm vụ (ord) đó.

$$N_i(\tau) = [\text{vid, ord}] \tag{2.3}$$

Trong ví dụ dưới dây, ta có thể thấy nhiệm vụ đầu tiên (index = 0) sẽ do xe 1 xử lý, thứ tự ưu tiên là 0 nên xe 1 sẽ xử lý nhiệm vụ này đầu tiên. Tiếp theo, nhiệm vụ thứ 2 (index = 1) và nhiệm vụ thứ n (index = n – 1) cùng được xe 3 xử lý. Các nhiệm vụ này được xử lý theo thứ tự của order nên nhiệm vụ thứ n sẽ được xe 3 xử lý trước nhiệm vụ thứ 2.

$$N(\tau) = \begin{bmatrix} [1,0] \\ [3,2] \\ \vdots \\ [3,1] \end{bmatrix} \tag{2.4}$$

Đến đây, ta cần đảm bảo nghiệm của bài toán phân công nhiệm vụ cần thỏa mãn các điều kiện sau đây.

- Mỗi nhiệm vụ chỉ được gán cho duy nhất một phương tiện. Điều này tương đương số lượng phần tử trong $N_i(\tau)$ đúng bằng số lượng nhiệm vụ cần thực hiện

$$|N(\tau)| = |M(\tau)| = n \tag{2.5}$$

- Danh sách nhiệm vụ của một phương tiện không có hai nhiệm vụ nào có cùng thứ tự ưu tiên

$$N_i(\tau)[\text{ord}] \neq N_j(\tau)[\text{ord}] \; \forall \, N_i(\tau), N_j(\tau) \in N(\tau), i \neq j, \; N_i(\tau)[\text{vid}] = N_j(\tau)[\text{vid}] \tag{2.6}$$

- Trong danh sách các nhiệm vụ, các nhiệm vụ tiền nhiệm phải có thứ tự ưu tiên nhỏ hơn các nhiệm vụ phụ thuộc vào nó

$$N_i(\tau)[\text{ord}] < N_j(\tau)[\text{ord}] \forall \, N_i(\tau), N_j(\tau) \in N(\tau), M_i \in M_j^- \tag{2.7}$$

*Chiến lược dỡ tải tác vụ*

Chiến lược dỡ tải tác vụ của phương tiện được xây dựng dựa trên việc cân nhắc giữa khả năng xử lý cục bộ và điều kiện của MEC server. Khi một tác vụ xuất hiện trên lộ trình, phương tiện phải quyết định liệu sẽ tự xử lý bằng tài nguyên tính toán sẵn có hay gửi tác vụ lên MEC server để rút ngắn thời gian xử lý. Quyết định này phụ thuộc vào nhiều yếu tố như kích thước dữ liệu cần truyền, số chu kỳ xử lý cần thiết, băng thông hiện tại. Nếu Phương tiện sẽ lựa chọn dỡ tải lên MEC Server nếu thời gian dỡ tải dự kiến nhỏ hơn thời gian xử lý tác vụ tại thiết bị. Chiến lược này giúp phương tiện tối ưu hóa lợi ích tổng thể bằng cách linh hoạt lựa chọn giữa xử lý cục bộ và dỡ tải lên MEC. Khi phương tiện dỡ tải tác vụ j lên các MEC server, tổng thời gian xử lý tác vụ $\text{t_off}_j$ được tính theo công thức

$$\text{t_off}_j = \frac{d_j}{R_{up}} + q_{MEC} + \frac{c_j}{f_{MEC}} + \frac{s_j}{R_{down}} + \tau_{ov} \tag{2.8}$$

Trong đó $d_j$ là kích thước dữ liệu (bits) cần đẩy lên MEC Server, $s_j$ là kích thước

dữ liệu (bits) MEC Server trả về; $R_{up}$ là tốc độ uplink (bits/s) thực tế, $R_{down}$ là tốc độ downlink (bits/s) thực tế; $q_{MEC}$ là thời gian chờ trên MEC Server trước khi tác vụ được xử lý; $c_j$ là số chu kỳ CPU cần để xử lý tác vụ (cycles), $f_{MEC}$ là tần số xử lý hiệu dụng của CPU MEC (cyles/s); $\tau_{ov}$ là tổng thời gian xử lý các việc khác như mã hóa/giải mã, thiết lập kết nối.

Để đơn giản hóa quá trình mô phỏng và giảm chi phí tính toán, trong đề án này tôi chỉ xét hai thành phần thời gian chính khi dỡ tải tác vụ lên MEC server: thời gian truyền dữ liệu lên và thời gian xử lý trên MEC. Các độ trễ khác được gom lại thành một hằng số cố định. Cách tiếp cận này giúp giữ lại các yếu tố ảnh hưởng quan trọng nhất đến quyết định dỡ tải của xe tự hành, đồng thời tránh làm mô hình trở nên quá phức tạp hoặc tiêu tốn tài nguyên tính toán không cần thiết. Từ đó tôi sẽ sử dụng công thức mới để tính tổng thời gian offload tác vụ lên server với một hằng số $\tau_{fixed}$ biểu diễn tổng các độ trễ khác

$$\mathrm{t_off}_j = \frac{d_j}{R_{up}} + \frac{c_j}{f_{MEC}} + \tau_{fixed} \tag{2.9}$$

Trong trường hợp phương tiện không dỡ tải tác vụ lên MEC Server, tổng thời gian xử lý tác vụ trên xe $\mathrm{t_veh}_j$ được biểu diễn bằng công thức dựa trên số chu kỳ CPU cần để xử lý tác vụ $c_j$ và $f_{VEH}$ là tần số xử lý hiệu dụng của CPU của phương tiện.

$$\mathrm{t_veh}_j = \frac{c_j}{f_{VEH}} \tag{2.10}$$

Để đơn giản việc lựa chọn phương pháp xử lý tác vụ sẽ được quyết định bởi tổng thời gian xử lý ít hơn của mỗi phương pháp. Ta có công thức tính tổng thời gian thực tế

$$\mathrm{t_proc}_j = \min\left(\mathrm{t_off}_j, \mathrm{t_veh}_j\right) \tag{2.11}$$

*Thời gian hoàn thành nhiệm vụ*

Thời hạn hoàn thành một nhiệm vụ $M_i$ đơn lẻ được tính bằng tổng thời gian hoàn thành các tác vụ khi thực hiện nhiệm vụ

$$\delta_i = \sum\nolimits_{j|t_{i,j}\in K_i} \text{t_proc}_j \tag{2.12}$$

Trong thực tế, một nhiệm vụ phụ thuộc rất nhiều và các nhiệm vụ tiền nhiệm và các nhiệm vụ có thứ tự thực hiện ưu tiên hơn trong cùng phương tiện. Do vậy, thời gian tổng cộng để một phương tiện hoàn thành một nhiệm vụ tối thiểu là

$$\delta_i(\tau) \geq \delta_i + \sum\nolimits_{i'\epsilon\kappa_1} \delta_{i'}(\tau) + \sum\nolimits_{i'\epsilon\kappa_2} \left[\sum\nolimits_{i''\epsilon\kappa_3} \delta_{i''}(\tau) + \delta_{i'}\right] \tag{2.13a}$$

Trong đó các tập $\kappa_1, \kappa_2, \kappa_3$ lần lượt là các nhiệm vụ cùng phương tiện nhưng có thứ tự thực hiện trước nhiệm vụ đang xét, các nhiệm vụ tiên quyết của nhiệm vụ đang xét do các phương tiện khác thực hiện và các nhiệm vụ cần hoàn thành trước các nhiệm vụ tiên quyết trên các phương tiện khác

$$\kappa_1 \triangleq \{M_{i'}\}\, \forall i' \neq i, N_i(\tau)[\text{vid}] = N_{i'}(\tau)[\text{vid}], N_i(\tau)[\text{ord}] < N_{i'}(\tau)[\text{ord}] \tag{2.14}$$

$$\kappa_2 \triangleq \{M_{i'}\}\, \forall i' \neq i, N_i(\tau)[\text{vid}] \neq N_{i'}(\tau)[\text{vid}], M_{i'} \in M_i^- \tag{2.15}$$

$$\kappa_3 \triangleq \{M_{i'}\}\, \forall i'' \neq i, N_i(\tau)[\text{vid}] \neq N_{i''}(\tau)[\text{vid}], N_{i''}(\tau)[\text{ord}] < N_{i'}(\tau)[\text{ord}], M_{i'} \in \kappa_2 \tag{2.16}$$

Từ các định nghĩa trên, để nhiệm vụ có thể hoàn thành được trong thời gian $T$, ta có thêm điều kiện sau

$$\delta_i + \sum\nolimits_{i'\epsilon\kappa_1} \delta_{i'}(\tau) + \sum\nolimits_{i'\epsilon\kappa_2} \left[\sum\nolimits_{i''\epsilon\kappa_3} \delta_{i''}(\tau) + \delta_{i'}\right] \leq \delta_i(\tau) \leq T \tag{2.13b}$$

*Chi phí nhiệm vụ và lợi nhuận còn lại*

Khi phương tiện thực hiện một nhiệm vụ, một phần chi phí vận hành sẽ được trừ trực tiếp vào tổng lợi nhuận thu được sau khi nhiệm vụ hoàn thành. Chi phí này có thể

bao gồm chi phí tính toán, chi phí truyền dữ liệu hoặc các chi phí tài nguyên khác phát sinh trong quá trình xử lý tác vụ. Việc khấu trừ này giúp phản ánh chính xác hơn giá trị thực mà phương tiện thu được, đồng thời khuyến khích hệ thống tối ưu hóa việc phân phối tác vụ sao cho chi phí xử lý là thấp nhất và lợi nhuận ròng đạt mức cao nhất. Điều này cũng tạo cơ sở để thuật toán ra quyết định lựa chọn chiến lược thực thi tác vụ tại thiết bị hay dỡ tải lên MEC Server nhằm tối ưu hóa hiệu quả tổng thể. Chi phí cho nhiệm vụ $M_i$ có thể được tính theo công thức dựa trên với $c_k$ là chi phí đơn vị được tính bởi MEC Server k

$$C_i = c_k \sum_{j|t_{i,j} \in K_i} \mathbf{1}_{\{\text{t_off}_{i,j} < \text{t_veh}_{i,j}\}} \text{t_off}_{i,j} \tag{2.17}$$

Lợi nhuận còn lại $R_i$ sau khi phương tiện hoàn thành nhiệm vụ trừ đi chi phí phát sinh được tính lại theo công thức

$$R_i = P_i - C_i \geq 0 \tag{2.18}$$

*Phát biểu bài toán tối ưu hóa*

Cuối cùng, bài toán tối ưu hóa việc gán nhiệm vụ và dỡ tải tác vụ cho xe tự hành với mục tiêu tối đa hóa số lượng nhiệm vụ hoàn thành trước thời gian $T$ có thể được mô hình hóa như sau

$$\mathcal{P}_1 : \max_{N(\tau)} \sum_{M_i \in M(\tau)} \mathbf{1}_{\{\delta_i(\tau) < T\}} \tag{2.19a}$$

$$s.t. (2.5), (2.6), (2.7), (2.13b), (2.18) \tag{2.19b}$$

Hay có thể phát biểu ngắn gọn thành bài toán tối đa hóa số lượng nhiệm vụ được hoàn thành trong thời gian $\tau$ thỏa mãn các điều kiện trong các công thức $(2.5), (2.6), (2.7), (2.13b), (2.18b)$.

Bài toán này thường được tiếp cận bằng nhiều phương pháp như quy hoạch tổ

hợp, thuật toán xấp xỉ, meta-heuristic (tiến hóa, bầy đàn, tìm kiếm cục bộ), hoặc các kỹ thuật mô phỏng *[7], [10]*. Tuy nhiên, những phương pháp này thường gặp giới hạn khi áp dụng trong môi trường động, phân tán và có không gian trạng thái lớn, vì chúng thiếu khả năng thích nghi theo thời gian và khó xử lý tính không ổn định của hệ thống thực tế *[3], [7]*. Học tăng cường sâu được áp dụng rộng rãi để xử lý tính phức tạp và tính động của các bài toán NP-hard trong các hệ thống mạng quy mô lớn, nơi các phương pháp lặp truyền thống thường bị giới hạn bởi độ phức tạp thời gian *[3]*. Phương pháp này hứa hẹn cho phép tác tử đưa ra quyết định thích ứng theo thời gian thực và đạt khả năng mở rộng tốt hơn so với các thuật toán meta-heuristic vốn đòi hỏi chi phí tính toán đáng kể.

### *2.2.2 Tối ưu hóa dựa trên mô hình học tăng cường sâu đa tác tử*

Trong các bài toán tối ưu hóa phức tạp như phân phối tác vụ trong hệ thống xe tự hành trên nền tảng điện toán biên, việc lựa chọn mô hình học tăng cường phù hợp đóng vai trò then chốt quyết định hiệu quả xử lý. Các mô hình tác tử đơn lẻ truyền thống thường được áp dụng vì cấu trúc đơn giản và dễ huấn luyện. Tuy nhiên, khi quy mô hệ thống mở rộng và mức độ tương tác giữa các phương tiện trở nên phức tạp hơn, mô hình này bộc lộ nhiều hạn chế nghiêm trọng về khả năng mở rộng, hành động tập trung và chi phí tính toán. Điều này dẫn đến sự nổi lên của mô hình đa tác tử, nơi nhiều tác tử hợp tác hoặc cạnh tranh trong một môi trường động để đạt được hiệu suất tối ưu hơn. Bảng dưới đây tổng hợp các đặc điểm nổi bật nhất giữa hai mô hình đơn tác tử và đa tác tử *[4], [5]*.

**Bảng 2.2. Bảng so sánh giữa Đơn tác tử và Đa tác tử**

| | Đơn tác tử | Đa Tác tử |
|---|---|---|
| Cấu trúc Mạng | Tập trung: Một tác tử đưa ra tất cả các quyết định | Phân tán hoặc Phân cấp: Nhiều tác tử tương tác và cùng đưa ra quyết định |
| Quy mô Hệ | khả năng mở rộng kém khi số | Khả năng mở rộng cao hơn |

| | | |
|---|---|---|
| thống | lượng phương tiện tăng lên | |
| Nhận thức Môi trường | Có khả năng truy cập vào trạng thái toàn cục | Thường dựa vào quan sát cục bộ hoặc quan sát một phần |
| Hành động | Một tác tử tạo ra một véc-tơ hành động hợp nhất | Mỗi tác tử tự chủ thực hiện hành động riêng lẻ |
| Thách thức Lớn | Nút thắt cổ chai về tính toán và tính phi thực tế khi không gian hành động tăng | Vấn đề Phi tĩnh, phối hợp và chi phí tính toán cao do sự tương tác phức tạp |
| Mục tiêu Tối ưu hóa | Thường nhằm mục tiêu tối ưu hóa toàn cục | Tối ưu hóa tiện ích tổng hợp hoặc mục tiêu cá nhân |
| Kiến trúc DRL tiêu biểu | DQN, PPO, A2C | MADDPG, COMA, VDN, MADQN, MAAC |

Trong các hệ thống điện toán biên cho phương tiện, tính phân tán, tính động cao và sự tương tác phức tạp giữa các phương tiện khiến mô hình tác tử đơn lẻ không còn phù hợp. Ngược lại, học tăng cường sâu đa tác tử nổi lên như một giải pháp hiệu quả nhờ khả năng học chính sách hợp tác hoặc cạnh tranh giữa nhiều tác tử đồng thời, giúp mô phỏng sát thực tế hành vi của các phương tiện trong mạng. Thứ nhất, MADRL hỗ trợ cơ chế phối hợp giữa nhiều tác tử, cho phép các phương tiện tự chủ đưa ra quyết định tối ưu trong môi trường chung. Mặc dù việc thực thi được phân tán, những thuật toán như MA-DDQN vẫn sử dụng một số thành phần tập trung như bộ nhớ kinh nghiệm chung để cải thiện khả năng hội tụ và chất lượng chính sách. Thứ hai, khung huấn luyện CTDE (Centralized Training, Decentralized Execution) là một đặc trưng nổi bật của MADRL. Trong giai đoạn huấn luyện, hệ thống có quyền truy cập trạng thái toàn cục để giảm thiểu tác động của tính phi tĩnh. Tuy nhiên, khi triển khai thực tế, mỗi tác tử lại ra quyết định dựa trên quan sát cục bộ, giúp duy trì tính mở rộng và giảm chi phí trao đổi thông tin *[5]*. Thứ ba, MADRL mang lại hiệu suất vượt trội trong nhiều nghiên cứu, thể hiện qua

khả năng giảm tiêu thụ năng lượng, giảm độ trễ, và tăng tỷ lệ hoàn thành nhiệm vụ so với các thuật toán điều phối truyền thống *[4]*. Điều này chứng minh MADRL phù hợp với các bài toán tối ưu NP-hard có tính động và yêu cầu phối hợp trong VEC. Từ các ý trên, MADRL là lựa chọn ưu tiên cho bài toán phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên đã được đề xuất.

Với việc sử dụng Học Tăng cường Đa Tác tử làm hướng tiếp cận chính cho bài toán, bước quan trọng tiếp theo là thiết kế hàm phần thưởng – thành phần cốt lõi quyết định hành vi học của từng tác tử. Trong môi trường đa tác tử, hàm phần thưởng không chỉ phải phản ánh mục tiêu tối ưu toàn cục mà còn phải cân bằng giữa lợi ích cá nhân và sự phối hợp giữa các tác tử. Vì vậy, việc xây dựng một hàm phần thưởng phù hợp đóng vai trò quyết định đối với khả năng hội tụ, tính ổn định và hiệu suất tổng thể của hệ thống MADRL. Hàm phần thưởng trong mô hình được xây dựng nhằm phản ánh đầy đủ lợi ích và chi phí phát sinh trong quá trình các xe tự hành lựa chọn nhiệm vụ và xử lý tác vụ. Mỗi nhiệm vụ khi được hoàn thành sẽ mang lại một giá trị lợi nhuận cơ bản, đồng thời nếu hoàn thành trước thời hạn thì phần thưởng được tăng thêm theo mức độ sớm so với thời hạn. Ngược lại, khi xe quyết định dỡ tải tác vụ lên MEC server, một phần lợi nhuận tương ứng sẽ bị trừ để phản ánh chi phí truyền và xử lý tại tầng biên, trong khi xử lý tại xe không bị phạt cũng không được thưởng thêm. Chi phí di chuyển trên đường được tính vào phần thưởng dưới dạng một đại lượng trừ, dựa trên quãng đường di chuyển và điều kiện giao thông mô phỏng. Bên cạnh đó, các nhiệm vụ có quan hệ phụ thuộc cũng được tính đến: khi một nhiệm vụ được hoàn thành, phần thưởng sẽ được cộng thêm một phần lợi ích từ các nhiệm vụ phụ thuộc đã hoặc sẽ hoàn thành, nhằm khuyến khích xe lựa chọn những chuỗi nhiệm vụ mang lại hiệu quả tổng thể cao hơn. Nhờ cơ chế này, hàm phần thưởng không chỉ hướng tới tối ưu hóa lợi ích riêng lẻ của từng nhiệm vụ mà còn cân bằng giữa thời gian, chi phí dỡ tải, chi phí di chuyển và giá trị lan truyền từ cấu trúc phụ thuộc của tập nhiệm vụ.

*Phần thưởng từ giá trị nhiệm vụ và hoàn thành sớm*

Mỗi nhiệm vụ mang một giá trị lợi ích cơ bản, được cộng trực tiếp vào phần thưởng khi nhiệm vụ được hoàn thành. Bên cạnh đó, nếu xe hoàn thành nhiệm vụ trước thời hạn cho phép, hệ thống thưởng thêm một khoản tương ứng với mức độ sớm so với deadline. Khoản thưởng này được điều chỉnh bởi hệ số $r_1$ và phản ánh mức ưu tiên của việc hoàn thành kịp hoặc sớm hơn thời gian yêu cầu. Nhờ đó, agent không chỉ hướng đến việc hoàn thành nhiệm vụ mà còn tối ưu hoá thời gian xử lý để gia tăng lợi ích tổng thể. Tôi định nghĩa $TCR_i$ là lợi ích khi hoàn thành nhiệm vụ $m_i$ và hoàn thành sớm (Task Completion Reward)

$$TCR_i = \max\left(0, \frac{T_i - t_i}{T_i}\right) \cdot \text{profit}_i \tag{2.20}$$

*Chi phí dỡ tải tác vụ lên MEC server*

Khi xe lựa chọn dỡ tải một tác vụ nhỏ lên MEC server thay vì xử lý cục bộ, một khoản chi phí được trừ khỏi phần thưởng. Chi phí này, được điều chỉnh bởi hệ số $r_2$, thể hiện cái giá phải trả cho việc sử dụng tài nguyên mạng và tài nguyên tính toán của MEC server. Cơ chế phạt này khuyến khích agent chỉ offload khi thật sự cần thiết, ví dụ như khi tác vụ vượt quá khả năng xử lý của xe hoặc khi xử lý tại local dẫn đến rủi ro trễ deadline. Nhờ vậy, hệ thống duy trì sự cân bằng giữa năng lực cục bộ và hạ tầng MEC. Công thức $OC_i$ (Offloading Cost) dưới đây xác định giá trị chi phí dỡ tải tác vụ lên MEC server

$$OC_i = \sum\nolimits_{j \in \text{Task}(i)} \mathbf{1}_{\{\text{offload}_{ij}\}} \tag{2.21}$$

Trong đó $OC$ là cấu hình chi phí cố định cho một hành động dỡ tải tác vụ lên MEC Server.

*Chi phí di chuyển để hoàn thành nhiệm vụ*

Khi xe di chuyển đến vị trí nhiệm vụ đã lựa chọn, chi phí di chuyển được trừ khỏi phần thưởng nhằm mô phỏng tiêu hao tài nguyên và thời gian khi di chuyển. Chi phí này tỷ lệ với quãng đường di chuyển nhân với vận tốc trung bình của từng đoạn đường, và được điều chỉnh qua hệ số $r_3$. Ta gọi chi phí này là $TC_i$ (Travel Cost) Cách tính này giúp mô hình phản ánh chính xác hơn ảnh hưởng của điều kiện giao thông, giúp agent ưu tiên các nhiệm vụ gần hoặc có đường đi thuận lợi thay vì chỉ chọn nhiệm vụ có lợi ích cao nhưng tốn quá nhiều thời gian và chi phí di chuyển.

$$TC_i = TC \cdot \sum_{\ell \in \text{route}_i} \left(\text{distance}_\ell \cdot \text{average_speed}_\ell\right) \tag{2.22}$$

Trong đó $TC$ là cấu hình chi phí cố định khi phương tiện di chuyển trong khoảng thời gian 01 giây.

*Lợi ích từ việc hoàn thành các nhiệm vụ phụ thuộc*

Một số nhiệm vụ chỉ có thể thực hiện được sau khi các nhiệm vụ khác đã hoàn thành, tạo nên cấu trúc phụ thuộc. Khi một nhiệm vụ được hoàn thành và nó đồng thời mở khoá hoặc kích hoạt lợi ích cho các nhiệm vụ phụ thuộc, hệ thống thưởng thêm một khoản do hệ số $r_4$ điều chỉnh. Khoản thưởng này khuyến khích agent xem xét chiến lược dài hạn, ưu tiên các nhiệm vụ mang tính then chốt thay vì chỉ tối ưu lợi ích cục bộ trước mắt. Điều này cải thiện khả năng phối hợp tổng thể giữa các xe và nâng cao hiệu quả hoàn thành các chuỗi nhiệm vụ có quan hệ chặt chẽ. Mỗi khi một nhiệm vụ $m_k$ hoàn thành, tôi sẽ cộng bổ sung phần thưởng cho các xe hoàn thành nhiệm vụ $m_i$ là nhiệm vụ mà $m_k$ phụ thuộc vào. Do đó, tổng lợi ích đạt được do hoàn thành các nhiệm vụ phụ thuộc được gọi là $DR_i$ (Dependency Reward) được tính bằng công thức

$$DR_i = \sum_{k \in \text{Dep}(i)} \mathbf{1}_{\{\text{complete}_k\}} \cdot \text{profit}_k \tag{2.23}$$

*Hàm phần thưởng tổng quát*

Từ 4 công thức trên, ta có được công thức tính hàm phần thưởng $R_i$ cho một nhiệm vụ $m_i$ như sau

$$R_i = \mathbf{1}_{\{\text{complete}_i\}}[\text{profit}_i + r_1 \cdot TCR_i - r_2 \cdot OC_i - r_3 \cdot TC_i + r_4 \cdot DR_i] \quad (2.24)$$

Các hệ số $r_1$, $r_2$, $r_3$, $r_4$ là các hệ số điều chỉnh hàm trong phần thưởng. Trong nghiên cứu này, tôi để các giá trị này bằng nhau và bằng 1. Tôi không đặt ưu tiên cho bất kỳ yếu tố nào. Nhưng các giá trị này cũng có thể thay đổi trong tương lai nhưng cần nhiều thời gian thử nghiệm hơn. Cuối cùng, ta có thể tính toán được tổng phần thưởng của cả hệ thống bằng công thức

$$R_{total} = \sum_i R_i \quad (2.25)$$

Tổng phần thưởng $R_{total}$ được tính bằng tổng các phần thưởng $R_i$ của từng nhiệm vụ $i$ mà agent hoặc hệ thống đã hoàn thành. Việc này giúp đánh giá hiệu quả tổng thể của chiến lược phân phối và xử lý tác vụ, đồng thời làm cơ sở để huấn luyện agent tối ưu hóa lựa chọn nhiệm vụ và phương án xử lý.

## 2.3 Xây dựng mô hình học tăng cường sâu đa tác tử cho bài toán tối ưu hóa phân phối tác vụ trong xe tự hành

### *2.3.1 Kiến trúc mạng nơ-ron trong mô hình*

Để lựa chọn được mô hình học tăng cường phù hợp cho bài toán lập lịch nhiệm vụ của xe tự hành, trước hết cần xem xét hai hướng tiếp cận quan trọng trong học tăng cường hiện đại: nhóm phương pháp dựa trên giá trị và nhóm phương pháp dựa trên chính sách *[11]*. Trong hướng tiếp cận dựa trên giá trị, DQN là thuật toán tiêu biểu nhờ khả năng học nhanh và hiệu quả trong những môi trường có phần thưởng được phản hồi ngay sau mỗi hành động. Ngược lại, hướng tiếp cận dựa trên chính sách và các biến thể kết hợp như actor–critic lại có thế mạnh trong việc xử lý các chiến lược dài hạn và khả năng

thăm dò sâu, với các thuật toán đại diện như PPO, vốn nổi bật về độ ổn định khi cập nhật chính sách, và A2C, một biến thể đơn giản nhưng giàu khả năng mô hình hóa.

**Bảng 2.3 So sánh 3 thuật toán DQN, PPO, A2C**

| Thuật toán | Ưu điểm | Nhược điểm/Hạn chế |
|---|---|---|
| DQN | Tốc độ học nhanh, ổn định nhờ cơ chế lưu trữ kinh nghiệm<br>Phù hợp với môi trường có phần thưởng tức thời | Dễ gặp vấn đề đánh giá quá cao giá trị Q<br>Không thật sự hiệu quả trong các môi trường đòi hỏi thăm dò sâu |
| PPO | Mạnh về cập nhật chính sách ổn định và cân bằng tốt giữa hiệu quả mẫu và độ phức tạp<br>Thích hợp cho các chiến lược dài hạn | Nhạy với tốc độ học<br>Chậm hơn so với DQN |
| A2C | Có khả năng biểu diễn chiến lược tốt nhờ cơ chế actor–critic đơn giản, giảm bớt sự phức tạp của A3C | Rất nhạy với tham số<br>thiếu cơ chế ổn định như PPO<br>Đòi hỏi lượng dữ liệu lớn để đạt hiệu suất cao |

Trong nghiên cứu này, tôi lựa chọn thuật toán Double DQN (DDQN) cho bài toán lập lịch nhiệm vụ của xe tự hành do tính phù hợp tự nhiên với đặc thù môi trường và yêu cầu của hệ thống. Thứ nhất, bài toán phân phối tác vụ trong VEC có cấu trúc phần thưởng rõ ràng và tức thời, khi mỗi hành động dỡ tải dẫn đến chi phí truyền và xử lý được phản hồi ngay lập tức. DDQN, vốn là phương pháp dựa trên Q-learning, hoạt động đặc biệt hiệu quả trong các môi trường có mối tương quan trực tiếp giữa hành động và phần thưởng, giúp mô hình học chính sách tối ưu nhanh và ổn định. Thứ hai, DDQN thừa hưởng ưu điểm tốc độ học nhanh của DQN nhưng đồng thời khắc phục lỗi đánh giá quá cao Q-value nhờ sử dụng hai mạng tách biệt cho việc chọn và đánh giá hành động. Điều

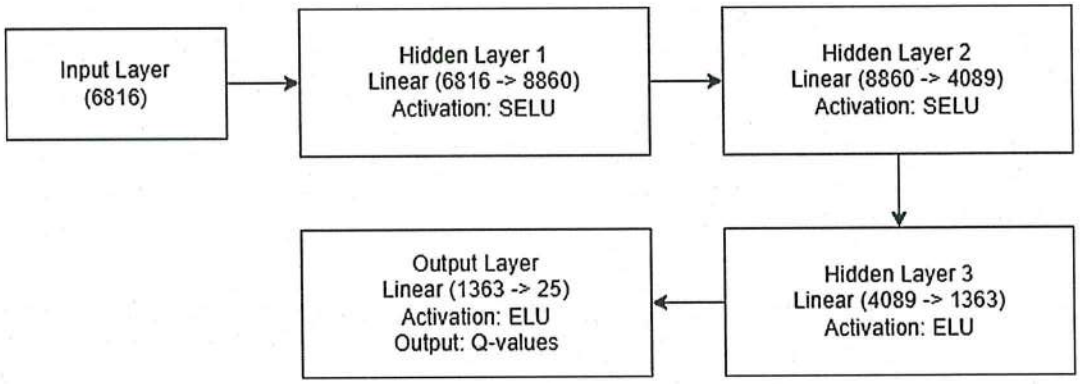
này làm giảm nhiễu trong quá trình học, tăng tính ổn định và giúp mô hình thích ứng tốt hơn với sự biến động của tải mạng và độ trễ trong hệ thống xe tự hành. Nhờ khả năng học hiệu quả, ổn định và phù hợp với đặc tính phần thưởng tức thời, DDQN trở thành lựa chọn hợp lý và hiệu quả cho bài toán lập lịch nhiệm vụ trong môi trường VEC.

Trong mô hình học tăng cường sâu đa tác tử, mỗi tác tử được trang bị một mạng Q-Network nhằm xấp xỉ hàm giá trị hành động $Q(s, a)$, từ đó hỗ trợ tác tử lựa chọn hành động tối ưu trong từng trạng thái của môi trường. Kiến trúc mạng Q được xây dựng để xử lý các trạng thái phức tạp, phi tuyến và có thể thay đổi liên tục theo thời gian, đảm bảo khả năng dự đoán chính xác giá trị Q cho toàn bộ không gian hành động. Mỗi phương tiện được trang bị một mạng Q riêng, từ đó có thể học cách ra quyết định tối ưu dựa trên trạng thái cục bộ của chính nó, bao gồm vị trí hiện tại, danh sách nhiệm vụ, lợi nhuận và chi phí khi hoàn thành các nhiệm vụ. Nhờ không phụ thuộc vào tham số của các tác tử khác, mỗi tác tử có thể thích nghi linh hoạt với thay đổi của môi trường mà không bị ảnh hưởng bởi sai lệch hoặc nhiễu trong quá trình học của toàn hệ thống.

Trong nghiên cứu này, tôi sử dụng một mạng Nơ-ron được thiết kế theo cấu trúc nhiều lớp nhằm học và xấp xỉ hàm giá trị hành động $Q(s, a)$ một cách hiệu quả trong không gian trạng thái có số chiều lớn. Kiến trúc mạng bao gồm một tầng đầu vào, ba tầng ẩn và một tầng đầu ra, sử dụng các hàm kích hoạt phi tuyến SELU và ELU để tăng tính ổn định và khả năng biểu diễn của mô hình.

Cụ thể, tầng đầu vào nhận vector trạng thái có kích thước bằng số chiều của trạng thái. Ba tầng ẩn được xây dựng với kích thước giảm dần, lần lượt bằng 130%, 60% và 20% số chiều của trạng thái đầu vào. Hai tầng ẩn đầu sử dụng hàm kích hoạt SELU, giúp mạng tự điều chuẩn (self-normalizing) và duy trì ổn định trong quá trình lan truyền ngược. Tầng ẩn thứ ba sử dụng hàm kích hoạt ELU, hỗ trợ mô hình học tốt hơn ở các giá trị âm và giảm hiện tượng chết nơ-ron. Tầng đầu ra là một lớp tuyến tính ánh xạ từ tầng ẩn cuối sang số chiều hành động. Hàm kích hoạt ELU tiếp tục được sử dụng tại tầng

này nhằm tạo ra các giá trị Q mềm hơn và giảm hiện tượng gradient bão hòa. Ngoài ra, mạng còn định nghĩa thêm một lớp Softmax để chuẩn hóa phân phối xác suất hành động. Kiến trúc này cho phép tác tử Double DQN học biểu diễn đặc trưng tốt hơn, đồng thời giảm sai lệch ước lượng Q-value, hỗ trợ tác tử đưa ra quyết định tối ưu.




**Hình 2.3. Sơ đồ kiến trúc mạng Double DQN**

Trong mô hình nhiều tác tử, mỗi xe tự hành được trang bị một mạng nơ-ron riêng biệt để ước lượng giá trị hành động, và do đó cần duy trì một bộ nhớ kinh nghiệm độc lập. Bộ nhớ này lưu trữ các bộ dữ liệu dạng (state, action, reward, next_state, done) tương ứng với từng bước tương tác của tác tử với môi trường. Cấu trúc replay buffer riêng cho từng tác tử giúp mô hình học phản ánh chính xác đặc trưng động lực học mà mỗi xe trải qua, tránh ảnh hưởng chéo giữa các tác tử có trạng thái hoặc chiến lược khác nhau. Đồng thời, việc lấy mẫu ngẫu nhiên từ bộ nhớ cục bộ giúp phá vỡ tương quan tạm thời trong dữ liệu, ổn định quá trình tối ưu và tăng khả năng hội tụ của mỗi tác tử trong môi trường phân tán.

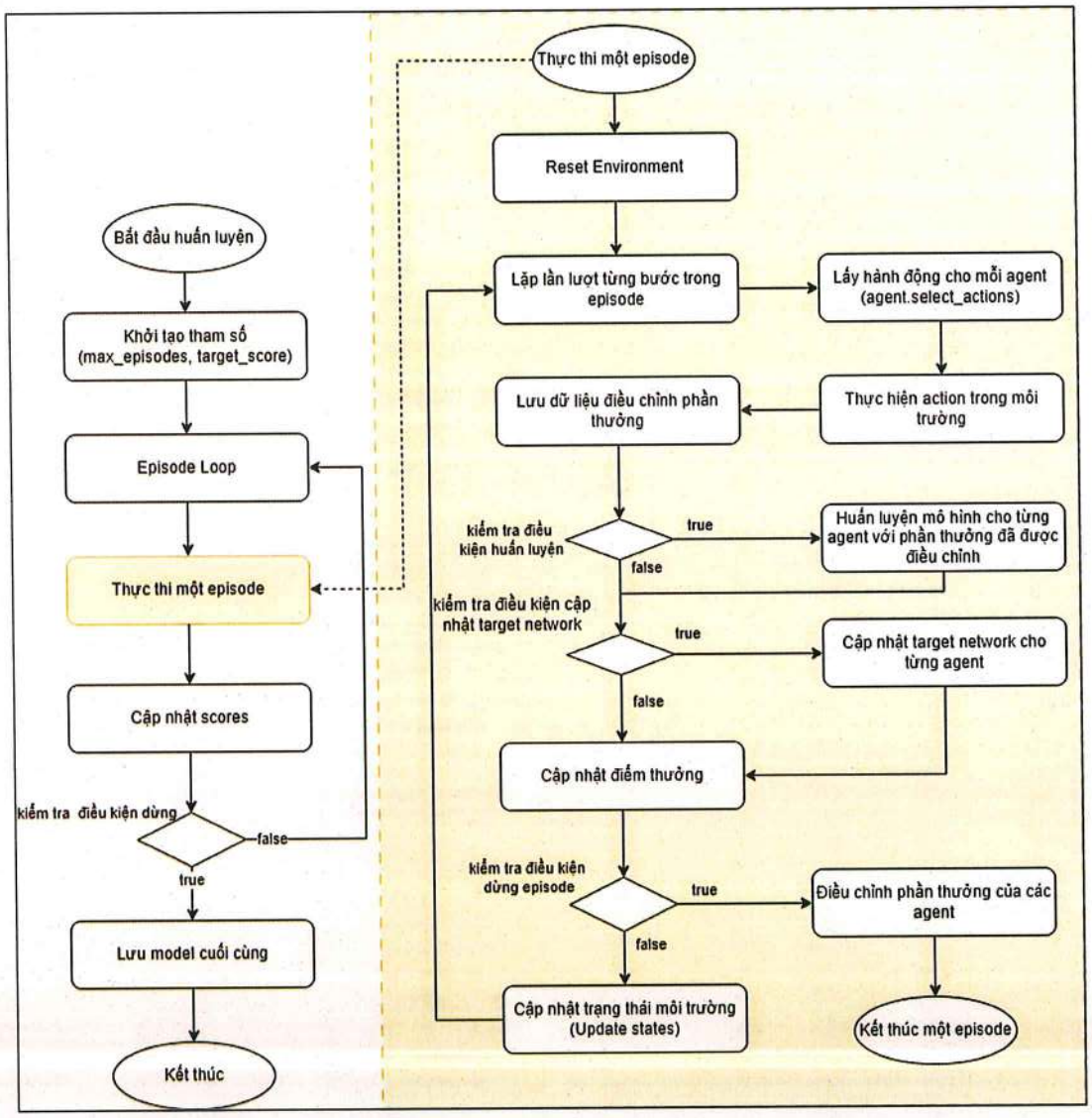
Bên cạnh bộ nhớ cục bộ, hệ thống duy trì một bộ nhớ kinh nghiệm chung ở mức toàn cục để tích hợp thông tin thu thập từ tất cả các tác tử. Replay buffer toàn cục ghi nhận các kinh nghiệm (state, action, reward, next_state, done) của mọi xe trong môi trường, cho phép thuật toán tận dụng sự đa dạng về trạng thái và hành động của tập thể. Bộ nhớ chung này giúp cải thiện khả năng tổng quát hóa, giảm nguy cơ tác tử học lệch

theo các mẫu trải nghiệm hạn chế, đồng thời phản ánh tốt hơn các đặc tính mang tính hệ thống như tải mạng, mức sử dụng tài nguyên hoặc xung đột truy cập. Khi cập nhật mô hình từ bộ nhớ toàn cục, thuật toán học có thể nắm bắt được xu hướng chung của môi trường, từ đó gia tăng hiệu quả của chiến lược lập lịch nhiệm vụ trên toàn hệ thống xe tự hành.

### *2.3.2 Quy trình huấn luyện mô hình*

Quy trình huấn luyện tác tử Double DQN được thực hiện theo vòng lặp tương tác liên tục giữa tác tử và môi trường. Mỗi tác tử quan sát trạng thái, lựa chọn hành động, nhận phần thưởng rồi cập nhật mạng Q thông qua cơ chế Double Q-Learning nhằm giảm sai lệch ước lượng giá trị Q. Toàn bộ quá trình huấn luyện có thể mô tả theo sơ đồ trong Hình 2.4. Thuật toán huấn luyện được vận hành theo cơ chế lặp lại qua từng episode. Trong mỗi episode, trainer sẽ kích hoạt hàm thực hiện một episode, tại đó toàn bộ quá trình tương tác giữa các tác tử và môi trường được thực hiện: tác tử quan sát trạng thái, lựa chọn hành động, nhận phần thưởng và cập nhật dữ liệu phục vụ học tập. Sau khi một episode kết thúc, hệ thống ghi nhận điểm thưởng của episode đó vào lịch sử và tính toán giá trị trung bình dựa trên một cửa sổ trượt gồm các episode gần nhất. Cách tiếp cận này giúp đánh giá hiệu năng của mô hình một cách ổn định và giảm thiểu tác động của các dao động ngẫu nhiên trong phần thưởng.

Song song với quá trình đánh giá, mô hình cũng được lưu định kỳ sau một số lượng episode xác định nhằm đảm bảo khả năng theo dõi tiến độ và phục hồi khi cần thiết. Các số liệu thống kê và biểu đồ theo dõi kết quả huấn luyện được cập nhật đồng thời, giúp giám sát quá trình hội tụ của chính sách. Thuật toán sẽ dừng lại khi điểm thưởng trung bình vượt qua ngưỡng mục tiêu do người dùng thiết lập hoặc khi số lượng episode đạt giới hạn tối đa. Khi kết thúc, mô hình được lưu lại và môi trường được đóng một cách an toàn.




**Hình 2.4. Sơ đồ luồng huấn luyện mô hình cho các tác tử**

Trong mỗi episode, hệ thống khởi tạo lại môi trường để các agent có thể học được các kinh nghiệm mới. Tại mỗi bước, tất cả các tác tử quan sát trạng thái hiện tại và lựa chọn hành động tương ứng thông qua chính sách đã học. Cơ chế này đồng thời kiểm soát việc trùng lặp hành động nhằm giảm thiểu các tình huống xung đột hoặc trạng thái lặp lại giữa các tác tử. Sau khi thu thập tập hành động hợp lệ, môi trường được tiến hành mô phỏng để trả về trạng thái kế tiếp, phần thưởng tức thời, thông tin kết thúc, cùng các dữ

liệu hỗ trợ cần thiết. Toàn bộ thông tin của từng bước (bao gồm trạng thái trước, hành động, phần thưởng thô, trạng thái mới và thông tin sửa đổi) được lưu trữ phục vụ cho giai đoạn điều chỉnh phần thưởng ở cuối episode.

Trong quá trình thực thi, hệ thống theo dõi số bước toàn cục để xác định thời điểm kích hoạt cơ chế huấn luyện mô hình. Khi đạt đến tần suất cập nhật định trước, mỗi tác tử chủ động thực hiện quá trình cập nhật tham số. Song song đó, mạng mục tiêu cũng được đồng bộ hoá định kỳ nhằm ổn định quá trình tối ưu. Sau mỗi bước, phần thưởng thô của từng tác tử được cập nhật vào lịch sử theo dõi. Episode kết thúc khi các tác tử hoàn thành nhiệm vụ hoặc khi đạt đến thời gian tối đa cho phép. Kết thúc toàn bộ episode, hệ thống tính toán số lượng nhiệm vụ đã hoàn thành và tổng lợi nhuận thu được. Toàn bộ dữ liệu ghi nhận xuyên suốt episode được đưa vào cơ chế điều chỉnh phần thưởng, giúp cải thiện khả năng học dựa trên mục tiêu tối ưu hoá lợi ích tổng thể. Cuối cùng, giá trị điểm thưởng của từng tác tử trong episode được trả về để phục vụ cho việc đánh giá hiệu năng huấn luyện. Thông qua quá trình học lặp lại và thử–sai trong nhiều episode, hệ thống sẽ dần khám phá và tối ưu hóa chính sách hành động.

Với không gian trạng thái đầu vào có kích thước $K$ chiều, mạng nơ-ron trong thuật toán DDQN được xây dựng gồm ba tầng ẩn có kích thước lần lượt là 1.3K (130%), 0.6K (60%) và 0.2K (20%), cùng với tầng đầu ra gồm A hành động. Chi phí tính toán chủ yếu đến từ các phép nhân và cộng trong giữa các tầng cùa mạng nơ-ron. Cụ thể, một lượt lan truyền xuôi yêu cầu khoảng $(1 \times 1.3 + 1.3 \times 0.6 + 0.6 \times 0.2)K^2 + AK \approx 2.2K^2$ phép toán. Trong mỗi bước huấn luyện, thuật toán DDQN thực hiện hai lượt lan truyền xuôi (qua mạng online và mạng target) và một lượt lan truyền ngược để cập nhật tham số mạng online. Do đó, độ phức tạp tính toán tổng thể của thuật toán xấp xỉ $O(K^2BMN)$ trong đó $B$ là kích thước minibatch và $N$ là tổng số bước huấn luyện và M là tổng số xe tự hành. Tuy nhiên, chi phí tính toán chủ yếu không đến từ số lượng xe mà từ kích thước không gian trạng thái ban đầu, thể hiện qua tham số $K$, do các phép tính trong mạng nơ-

ron sâu phụ thuộc trực tiếp vào số chiều của trạng thái đầu vào. Do đó, mặc dù việc mở rộng hệ thống với nhiều xe tự hành làm gia tăng chi phí tính toán, yếu tố chi phối chính đến độ phức tạp vẫn là không gian trạng thái $K$.

## 2.4 Kết luận chương 2

Trong chương này, tôi đã đề xuất kiến trúc hệ thống theo mô hình ba lớp Cloud–Edge–End nhằm tối ưu hóa việc xử lý tác vụ với các mức yêu cầu độ trễ khác nhau. Cụ thể, lớp Cloud đảm nhiệm các tác vụ không thời gian thực, lớp Edge xử lý các tác vụ gần thời gian thực trên các máy chủ MEC, trong khi lớp End thực hiện các tác vụ thời gian thực trực tiếp trên phương tiện. Tiếp đến, tôi đã đưa ra mô hình mô phỏng chuẩn hóa số lượng phương tiện và nhiệm vụ để tạo đầu vào cố định, giúp việc huấn luyện thuật toán phân công nhiệm vụ và dỡ tải tác vụ trở nên ổn định và nhất quán. Bài toán tối ưu hướng tới việc gán nhiệm vụ cho các phương tiện và lựa chọn chiến lược xử lý để tối đa hóa số nhiệm vụ hoàn thành đúng hạn. Trong phần cuối chương, tôi trình bày về việc xây dựng mô hình học tăng cường sâu đa tác tử. Tôi lựa chọn Double DQN vì nó phù hợp với đặc thù phần thưởng tức thời của bài toán lập lịch tác vụ trong VEC, đồng thời khắc phục nhược điểm đánh giá quá cao Q-value của DQN và mang lại khả năng học ổn định hơn so với các thuật toán khác như PPO và A2C.

Nhìn chung, các nội dung trong chương 2 đã đặt nền móng quan trọng cho việc triển khai và đánh giá thực nghiệm ở chương tiếp theo. Mô hình đề xuất không chỉ đáp ứng yêu cầu của bài toán phân phối tác vụ trong hệ thống xe tự hành mà còn mở ra hướng tiếp cận hiệu quả cho các bài toán tối ưu hóa trong môi trường phân tán, nhiều tác tử và có tính thời gian thực.

# Chương 3. Kiểm nghiệm và đánh giá mô hình đề xuất trong tối ưu hóa phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên

Chương này tôi sẽ trình bày toàn bộ quá trình thực nghiệm nhằm đánh giá hiệu quả của phương pháp phân phối tác vụ dựa trên học tăng cường sâu đa tác tử. Trước tiên, chương này mô tả cách xây dựng dữ liệu mô phỏng và thiết lập các tham số huấn luyện cho mô hình, bao gồm cấu hình môi trường, đặc tính tác vụ và các tham số tối ưu hóa. Tiếp theo, kết quả huấn luyện mô hình sẽ được phân tích chi tiết, cùng với việc nạp mô hình vào từng tác tử và tiến hành chạy thử nghiệm so sánh với các phương pháp lập lịch truyền thống để làm nổi bật ưu thế của MADRL. Cuối cùng, tôi đề cập đến khả năng triển khai thực tế của hệ thống, bao gồm các yêu cầu, thách thức và định hướng ứng dụng.

## 3.1 Dữ liệu thực nghiệm

Để hiện thực hóa mô hình phân phối tác vụ trong hệ thống xe tự hành trên nền tảng điện toán biên, toàn bộ kiến trúc môi trường, tác tử và cơ chế huấn luyện được triển khai dưới dạng mã nguồn. Việc mô hình hóa bằng code cho phép mô phỏng chính xác các thành phần của hệ thống. Các lớp và hàm được thiết kế theo cấu trúc mô-đun, đảm bảo khả năng tái sử dụng, mở rộng và kiểm thử. Nhờ việc mô hình hóa bằng mã nguồn, toàn bộ hệ thống có thể được kiểm nghiệm lặp lại trong nhiều kịch bản khác nhau, qua đó đánh giá một cách định lượng hiệu năng của phương pháp đề xuất.

### *3.1.1 Nhiệm vụ và tác vụ*

Trong mô hình thực nghiệm, hệ thống được khởi tạo với 25 nhiệm vụ độc lập, mỗi nhiệm vụ đại diện cho một yêu cầu vận chuyển hoặc xử lý mà xe tự hành cần thực hiện. Các nhiệm vụ này được phân phối cho 5 xe tự hành, và bài toán đặt ra là xác định chiến lược lập lịch tối ưu để đảm bảo các nhiệm vụ được hoàn thành đúng thứ tự, đúng

thời gian và đạt lợi nhuận cao nhất. Mỗi nhiệm vụ được mô tả dưới dạng một cấu trúc dữ liệu bao gồm các thành phần chính:

- Điểm bắt đầu (start_point): tọa độ ví trí bắt đầu nhiệm vụ.
- Điểm kết thúc (end_point): tọa độ vị trí đích đến để hoàn thành nhiệm vụ.
- Danh sách nhiệm vụ tiền nhiệm (dependencies): tập các nhiệm vụ bắt buộc phải hoàn thành trước khi nhiệm vụ hiện tại có thể được thực thi.
- Lợi nhuận (profit): Lợi ích nhận được sau khi nhiệm vụ hoàn tất

Trong ví dụ dưới đây, nhiệm vụ có ID = 0 yêu cầu xe tự hành di chuyển từ một điểm bắt đầu đến điểm kết thúc xác định, mang lại lợi nhuận 92. Tuy nhiên, nhiệm vụ này chỉ có thể được thực hiện khi hai nhiệm vụ tiền nhiệm (ID 13 và 20) đã hoàn thành. Điều này tạo ra một cấu trúc phụ thuộc dạng đồ thị có hướng giữa các nhiệm vụ, khiến bài toán lập lịch trở nên phức tạp và phù hợp để áp dụng mô hình học tăng cường sâu đa tác tử. Toàn bộ 25 nhiệm vụ được tổ chức theo cách tương tự, tạo thành một tập dữ liệu đại diện cho một môi trường thực tế với các ràng buộc không gian, tuyến đường và thứ tự phụ thuộc. Dữ liệu này được sử dụng làm cơ sở cho việc mô phỏng, huấn luyện và đánh giá mô hình tối ưu hóa phân phối nhiệm vụ trong hệ thống xe tự hành.


```
{
  "id": 0,
  "start_point": [1206464.2212596382, 2336286.92208349],
  "end_point": [1207531.136085073, 2333003.7365319855],
  "dependencies": [13, 20],
  "profit": 92
}
```


**Hình 3.1. Mô phỏng một nhiệm vụ**

Trong thực tế, trong quá trình di chuyển để hoàn thành các nhiệm vụ chính, mỗi xe tự hành phải xử lý nhiều tác vụ nhỏ phát sinh liên tục như cập nhật điều kiện giao thông theo thời gian thực, thu thập và gửi dữ liệu cảm biến, quét môi trường xung quanh,

hay thực hiện các tác vụ liên lạc cần thiết. Để đơn giản hóa quá trình mô phỏng nhưng vẫn đảm bảo tính thực tế, hệ thống của tôi thực hiện việc tiền xử lý và sinh trước tập tác vụ nhỏ tương ứng với từng đoạn đường mà xe có thể đi qua. Mỗi đoạn đường sẽ được mô tả bằng một tập thông tin cố định, bao gồm độ dài, tốc độ di chuyển dự kiến và tổng số lượng tác vụ trên đường. Các tác vụ nhỏ được gán trước cho từng đoạn đường. Trong ví dụ dưới đây, mỗi đoạn đường bao gồm danh sách các tác vụ, mỗi tác vụ được mô hình hóa bằng một bộ ba giá trị bao gồm kích thước dữ liệu đầu vào cần gửi lên MEC Server (nếu cần), độ phức tạp tính toán và ID của tác vụ. Khi xe di chuyển qua một quãng được, nó sẽ thực hiện tuần tự các tác vụ này như thể chúng được sinh ra tự nhiên trong môi trường.


```
{
  "segment_id": 0,
  "status": 0,
  "distance": 10.692181996575897,
  "speed": 100.0,
  "max_tasks": 2,
  "tasks": {
    "0": [135, 2, 0],
    "1": [361, 1, 1]
  }
}
```


**Hình 3.2. Mô hình hóa tập hợp tác vụ trên đường**

Cách tiếp cận này cho phép mô phỏng hành vi của xe tự hành một cách trực quan và kiểm soát được mức độ phức tạp, đồng thời giúp hệ thống lập lịch có thể dự đoán, phân bổ và tối ưu hóa quá trình xử lý tác vụ trên toàn bộ hành trình của từng xe.

### *3.1.2 Xe tự hành và tác tử*

Để mô phỏng hành vi của xe tự hành trong hệ thống giao thông thông minh, tôi xây dựng một lớp đối tượng đại diện cho mỗi phương tiện. Các phương tiện này được xem như một tác tử thông minh có khả năng ra quyết định trong môi trường động.

Phương tiện có thể lựa chọn nhiệm vụ cần thực hiện, xử lý các tác vụ nhận được trên đường tại chính thiết bị (local), hoặc dỡ tải các tác vụ lên các máy chủ MEC nhằm giảm độ trễ và tối ưu thời gian xử lý. Khả năng lựa chọn giữa xử lý trên xe và dỡ tải cho phép phương tiện cân bằng giữa tài nguyên tính toán hạn chế và yêu cầu về thời gian hoàn thành nhiệm vụ. Trong đề án này, tôi đặt giá trị tần số CPU của các xe là 5 (Hz).

Mỗi phương tiện duy trì ba nhóm nhiệm vụ riêng biệt: danh sách nhiệm vụ đã nhận thực hiện, danh sách nhiệm vụ sẵn sàng (những nhiệm vụ không có nhiệm vụ tiền nhiệm hoặc tất cả các nhiệm vụ tiền nhiệm đã được hoàn thành), và danh sách nhiệm vụ đang được xử lý. Cơ chế tổ chức này giúp phương tiện chủ động theo dõi tiến độ, xác định có thể thực hiện nhiệm vụ hay không tại một thời điểm cố định. Phương tiện cũng được trang bị khả năng tính toán thời gian hoàn thành nhiệm vụ, bao gồm thời gian di chuyển, thời gian xử lý tại thiết bị và thời gian dỡ tải lên MEC Server. Dựa trên các ước lượng này, phương tiện đánh giá lợi nhuận còn lại sau khi trừ đi chi phí thực hiện nhiệm vụ. Việc cập nhật lợi nhuận giúp tác tử đánh giá chất lượng chiến lược đã lựa chọn và đóng vai trò quan trọng trong cơ chế học tăng cường sâu.

Ngoài ra, mỗi khi hoàn thành một nhiệm vụ, phương tiện sẽ thông báo trạng thái này tới các phương tiện khác trong môi trường. Cơ chế thông báo này mô phỏng quá trình đồng bộ thông tin trong hệ thống thực, giúp các tác tử khác cập nhật lại danh sách nhiệm vụ của mình, đặc biệt là những nhiệm vụ có phụ thuộc vào nhiệm vụ vừa được hoàn thành. Điều này đảm bảo tính nhất quán trong toàn bộ hệ thống và giúp quá trình lập lịch được triển khai theo đúng điều kiện ràng buộc của bài toán.

### *3.1.3 MEC Servers*

MEC Server được mô phỏng bằng một lớp có vai trò cung cấp tài nguyên xử lý bổ sung cho các phương tiện tự hành khi thực hiện dỡ tải tác vụ. Mỗi đối tượng MEC được gắn với một vị trí cố định trên bản đồ, biểu diễn bằng một điểm tọa độ hai chiều. Vị trí này cho phép mô phỏng quá trình xe tự hành kết nối đến các máy chủ MEC lân

cận dựa trên vị trí địa lý thực tế. Ngoài vị trí, mỗi MEC có một tham số quan trọng là tần số CPU, biểu thị năng lực xử lý tính toán của máy chủ. Tần số CPU (đơn vị Hz) quyết định tốc độ thực hiện các tác vụ được dỡ tải từ phương tiện. Trong đề án này, tôi đặt giá trị này ngẫu nhiên trong khoảng từ 100 – 300 Hz (Cao gấp từ 20 – 60 lần so với tần số CPU trên các xe tự hành). Thông qua mô hình này, hệ thống có thể mô phỏng quá trình dỡ tải từ xe tự hành sang các MEC server, từ đó đánh giá hiệu quả của chiến lược phân phối tác vụ trong môi trường có nhiều nguồn tài nguyên tính toán khác nhau.

### *3.1.4 Mô phỏng mạng truyền thông*

Hệ thống bản đồ được xây dựng từ dữ liệu thực tế lấy từ OpenStreetMap (OSM). Sau khi tải dữ liệu, các tuyến đường được trích xuất và lọc lại, loại bỏ những đường quá nhỏ hoặc ít được sử dụng – những tuyến đường này không phù hợp cho xe tự hành vận hành. Việc này giúp hệ thống tập trung vào các tuyến đường khả thi, đảm bảo tính thực tế và khả năng điều hướng an toàn của các phương tiện tự hành. Từ bản đồ đã xử lý, hệ thống xác định các đoạn đường thẳng và các nút giao quan trọng. Các nút giao này là các điểm nối giữa các đoạn đường phục vụ cho việc lập lịch và định tuyến cho xe tự hành. Nhờ việc biểu diễn bản đồ dưới dạng đồ thị, các thuật toán điều phối nhiệm vụ và phân phối tác vụ có thể hoạt động hiệu quả và trực quan hơn.

Tiếp theo, tôi bổ sung 5 trạng thái giao thông trong hệ thống lần lượt là Tự do, Ổn định, Chậm, Tắc nghẽn và Tê liệt. Mức độ đông đúc của từng tuyến đường được sinh ngẫu nhiên dựa trên ma trận xác suất trong **Error! Reference source not found.**. Trong đó mỗi hàng thể hiện phân bố xác suất cho các trạng thái giao thông khác nhau từ tự do đến tắc nghiêm trọng. Giá trị cụ thể của trạng thái giao thông trên mỗi đoạn đường được xác định dựa trên cấu hình độ bận rộn trung bình của toàn bản đồ. Ví dụ nếu trạng thái giao thông trên toàn bản đồ là 2, tức là giờ cao điểm, thì ma trận xác suất quyết định một đoạn đường có 40% khả năng là đường di chuyển Tự do, 30% là đường di chuyển Ổn định, 10% đối với mỗi đường di chuyển Chậm, đường Tắc nghẽn và đường bị Tê liệt.

Cơ chế này giúp mô phỏng sát hơn các điều kiện giao thông thực tế và cung cấp thông tin cần thiết cho quá trình ra quyết định của các xe.

**Bảng 3.1. Phân bố xác xuất mức độ giao thông trên các đoạn đường**

| | Tự do | Ổn định | Chậm | Tắc nghẽn | Tê liệt |
|---|---|---|---|---|---|
| Không tắc | 100% | 0% | 0% | 0% | 0% |
| Lưu thông bình thường | 70% | 20% | 10% | 0% | 0% |
| Giờ cao điểm | 40% | 30% | 10% | 10% | 10% |
| Tắc nghẽn | 10% | 10% | 20% | 30% | 30% |
| Tắc nghiêm trọng | 0% | 10% | 10% | 30% | 50% |

Trong mô hình hệ thống, các MEC server được đặt cố định tại các giao lộ của bản đồ để hỗ trợ xử lý tác vụ gần với phương tiện. Tổng cộng, hệ thống triển khai 20 MEC server, mỗi server được gán một vị trí ngẫu nhiên trên các điểm giao lộ đảm bảo phân bố đều và hợp lý. Việc lựa chọn các giao lộ làm vị trí đặt MEC giúp tối ưu hóa khả năng giảm độ trễ truyền dữ liệu, vì các giao lộ là các nút quan trọng mà phương tiện thường đi qua. Vị trí cụ thể của từng MEC server được sinh ngẫu nhiên dựa trên danh sách các giao lộ, nhưng vẫn tuân thủ các ràng buộc như không đặt quá gần nhau hoặc ngoài khu vực bản đồ, nhằm đảm bảo mỗi server có vùng phủ sóng hiệu quả và hỗ trợ tốt cho các phương tiện khi di chuyển.

Bảng dưới đây mô tả các thông số cấu hình trong môi trường phục vụ cho bài toán phân phối tác vụ cho xe tự hành. Các tham số của môi trường mạng được kế thừa và tham khảo theo mô hình được trình bày trong tài liệu *[10]*.

**Bảng 3.2. Cấu hình tham số mô phỏng môi trường**

| Cấu hình | Tên cấu hình | Giá trị | Ý nghĩa |
|---|---|---|---|
| map_config | real_center_point | 20.98, 105.78 | Tọa độ của trung tâm bản đồ |
| | radius | 2500 | Bán kính bản đồ |
| | traffic_level | 1 | Trạng thái giao thông chung toàn bản đồ |
| | from_file | 1 | 0 – Tạo mới từ OSM<br>1 – Load từ bản đồ đã tạo trước đó |
| network_config | maximum_MECs | 20 | Số lượng MEC Servers trên bản đồ |
| | cpu_freq_range | [100, 300] | Khoảng tần số CPU của MEC Server (Hz) |
| | vehicle_cpu_freq | 5 | Tần số CPU của một xe (Hz) |
| | best_rate_radius | 100 | Bán kính vùng tối ưu của MEC (m) |
| | transmit_power | 199.526e-3 | Công suất phát của thiết bị (W) |
| | noise_density | 3.98e-21 | Mật độ nhiễu phổ |
| | num_channels | 10 | Tổng số kênh mà băng thông được chia |
| | total_bandwidth | 20e6 | Tổng băng thông khả dụng (Hz) |
| | path_loss_exp | 3 | Hệ số suy hao đường truyền: 2 - không gian tự do; 3–4 - đô thị, nhiều vật cản |

## 3.2 Cài đặt thực nghiệm

Mỗi phương tiện tự hành được mô hình hóa như một tác tử trong môi trường học tăng cường đa tác tử. Kiến trúc mạng nơ-ron sử dụng trong mô hình được trình bày chi tiết ở Chương 2. Quá trình huấn luyện được cấu hình với các tham số chính: tốc độ học (learning rate) xác định mức độ điều chỉnh trọng số mạng, kích thước batch (batch size) dùng trong mỗi lần cập nhật, kích thước bộ nhớ replay (replay buffer size) để lưu trữ các trải nghiệm, hệ số chiết khấu $\gamma$ (discount factor) để tính giá trị tương lai, tham số thăm dò ($\varepsilon$) cùng hệ số giảm $\varepsilon$ theo thời gian để cân bằng giữa khám phá và khai thác, cũng như số bước để cập nhật mạng mục tiêu (target network). Ngoài ra, mỗi episode được

giới hạn số bước và tổng số episode huấn luyện cũng được xác định để đảm bảo quá trình huấn luyện hoàn chỉnh.

Mỗi episode mô phỏng tương ứng với một khoảng thời gian xác định trong môi trường giao thông, trong đó các phương tiện thực hiện các nhiệm vụ di chuyển và xử lý tác vụ. Mỗi phương tiện có giới hạn số lượng nhiệm vụ tối đa trong một episode, đảm bảo cân bằng tải và phản ánh khả năng xử lý thực tế. Quy tắc dừng huấn luyện được thiết lập dựa trên điểm thưởng trung bình, tức là khi các phương tiện đạt hiệu năng tối ưu theo chỉ số này, quá trình huấn luyện sẽ được kết thúc sớm để tiết kiệm tài nguyên tính toán.

**Bảng 3.3. Tham số huấn luyện mô hình**

| Hệ số | Ý nghĩa | Giá trị |
|---|---|---|
| learning rate | Tốc độ học | 1e-5 |
| batch size | Số lượng mẫu mini-batch khi huấn luyện | 512 |
| replay buffer size | Kích thước tối đa của replay buffer | 10,000,000 |
| discount factor | Hệ số chiết khấu γ, xác định mức độ ảnh hưởng của phần thưởng tương lai | 0.95 |
| epsilon | Xác suất khám phá ban đầu epsilon-greedy | 1.0 |
| epsilon_decay | Tốc độ giảm epsilon sau mỗi bước huấn luyện | 0.99 |

Hệ thống được mô phỏng trên server chạy hệ điều hành CentOS 7.9 với CPU Intel Xeon Gold 6226R 64 lõi (2 socket × 16 lõi, hỗ trợ Hyper-Threading) và bộ nhớ đệm L1/L2/L3 lần lượt 1 MiB/32 MiB/44 MiB. Hệ thống trang bị 2 GPU NVIDIA Quadro RTX 6000 với 24 GB VRAM mỗi card, driver 550.144.03 và CUDA 12.4, sẵn sàng tăng tốc các tác vụ tính toán. Cấu hình này đủ mạnh để thực thi các mô hình học tăng cường sâu đa tác tử cho xe tự hành. Phần mềm sử dụng Python với phiên bản 3.9.23, kết hợp với các thư viện học máy như PyTorch để xây dựng và huấn luyện mạng nơ-ron. Các thư viện phụ trợ khác cũng được sử dụng để quản lý dữ liệu, mô phỏng giao thông và trực

quan hóa kết quả.

## 3.3 Đánh giá thực nghiệm

### *3.3.1 Các chỉ số đánh giá*

Trong các thí nghiệm mô phỏng, Tổng lợi nhuận (Total Profit) được sử dụng làm chỉ số chính để đánh giá hiệu quả của phương pháp phân phối tác vụ. Chỉ số này tính tổng lợi nhuận tích lũy từ các nhiệm vụ mà các xe tự hành hoàn thành trong quá trình mô phỏng. Giá trị Tổng lợi nhuận không chỉ phản ánh khả năng hoàn thành nhiệm vụ mà còn thể hiện hiệu quả trong việc tối ưu hóa phân phối tác vụ, giúp so sánh trực tiếp giữa các phương pháp lập lịch khác nhau và đánh giá mức độ cải thiện mà mô hình mang lại. Sau khi huấn luyện mô hình học sâu đa tác tử với tối đa 100,000 episode, các tác tử được triển khai để thực hiện mô phỏng 100 lần, từ đó thu thập các chỉ số đánh giá bao gồm tổng số nhiệm vụ hoàn thành, tổng lợi nhuận thu được từ các phương tiện. Kết quả này được so sánh với các phương pháp lập lịch truyền thống như Random Scheduling và Greedy Scheduling. Việc so sánh cho phép làm nổi bật hiệu quả vượt trội của mô hình học sâu đa tác tử trong việc phân phối tác vụ, thể hiện ở mức tổng lợi nhuận cao hơn, độ trễ thấp hơn và khả năng khai thác tài nguyên MEC tối ưu hơn so với các phương pháp truyền thống. Trong nghiên cứu này, với kết quả từ R lần mô phỏng, tại lần thứ i ta thu được tổng lợi nhuận của các tác tử và tổng số nhiệm vụ hoàn thành lần lượt là $P_i$ và $C_i$. Công thức để đánh giá bao gồm

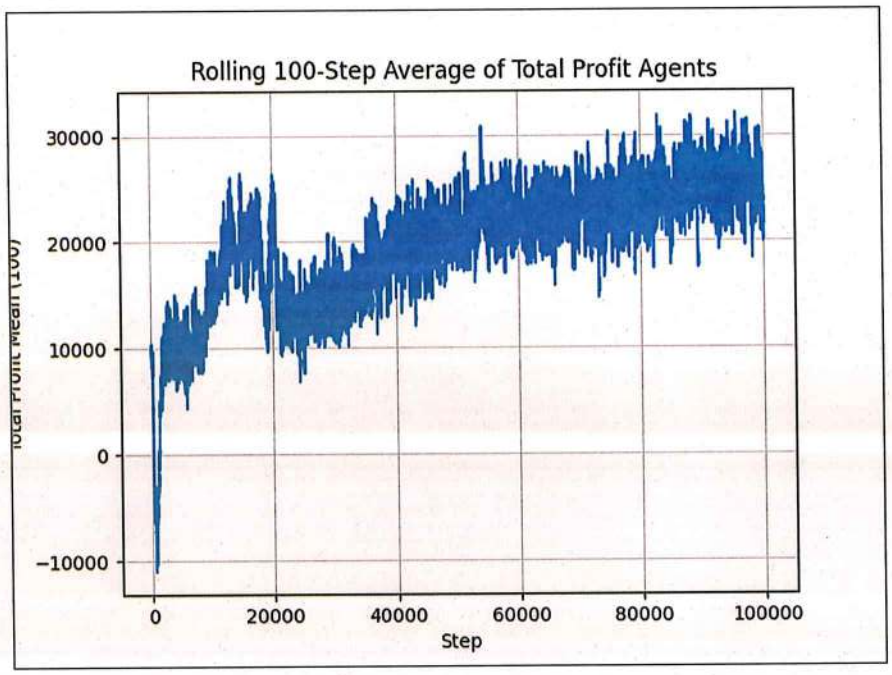
$$\bar{P} = \frac{1}{R}\sum_{1}^{R} P_i \, , \bar{C} = \frac{1}{R}\sum_{1}^{R} C_i \tag{3.1}$$

Trong đó $\bar{P}$ và $\bar{C}$ lần lượt là các giá trị trung bình tổng lợi nhuận của các tác tử và trung bình tổng số nhiệm vụ hoàn thành. Các giá trị thu được từ thuật toán đề xuất sẽ được so sánh với hai phương pháp cơ sở là Random Scheduling và Greedy Scheduling nhằm đánh giá hiệu quả tương đối. Việc so sánh này giúp làm rõ mức độ cải thiện của

thuật toán trong các kịch bản khác nhau và chứng minh tính ưu việt so với các chiến lược lập lịch đơn giản.

### *3.3.2 Kết quả thực nghiệm*

Hình ảnh dưới đây trình bày kết quả huấn luyện của mô hình MA-DDQN, cho thấy sự hội tụ và ổn định của các agent cùng với xu hướng tăng của phần thưởng tối đa trong suốt 100,000 episode. Đường màu xanh thể hiện giá trị lợi nhuận tổng thể của 5 tác tử đạt được trung bình 100 lần thử nghiệm gần nhất tại mỗi bước huấn luyện. Ta có thể quan sát rằng sau giai đoạn đầu với sự dao động lớn, đường tổng lợi nhuận trung bình (Total Profit) liên tục tăng và đạt trạng thái ổn định minh chứng cho khả năng phối hợp giữa các agent trong việc tối ưu hóa tổng lợi nhuận của hệ thống.




**Hình 3.3. Kết quả huấn luyện mô hình MA-DDQN**

Trong các thực nghiệm, tôi so sánh mô hình MA-DDQN với hai thuật toán cơ bản.

- Thuật toán chọn ngẫu nhiên (Random Selection) thực hiện phân phối ngẫu nhiên

các nhiệm vụ cho các xe mà không dựa trên bất kỳ tiêu chí tối ưu nào

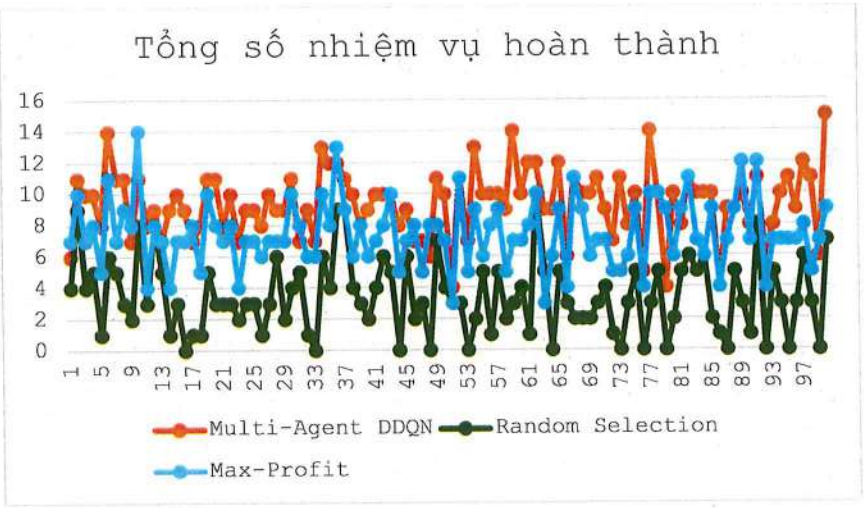
- Thuật toán Max-Profit thực hiện theo chiến lược tham lam: trước tiên sắp xếp các nhiệm vụ theo thứ tự dựa trên các phụ thuộc giữa nhiệm vụ, sau đó phân chia lần lượt cho các xe. Thuật toán này đảm bảo rằng các nhiệm vụ được thực hiện theo thứ tự hợp lệ, nhưng không cân nhắc đến hiệu quả phối hợp tổng thể giữa các xe.

Để đánh giá hiệu quả của các thuật toán, tôi thực hiện mô phỏng trên 100 môi trường ngẫu nhiên khác nhau. Trong phụ lục 01 bảng kết quả chi tiết về tổng số nhiệm vụ hoàn thành và tổng lợi nhuận của các tác tử đối với mỗi thuật toán trong cùng một môi trường mô phỏng. Dưới đây là kết quả trung bình 100 lần thực hiện mô phỏng với mỗi thuật toán

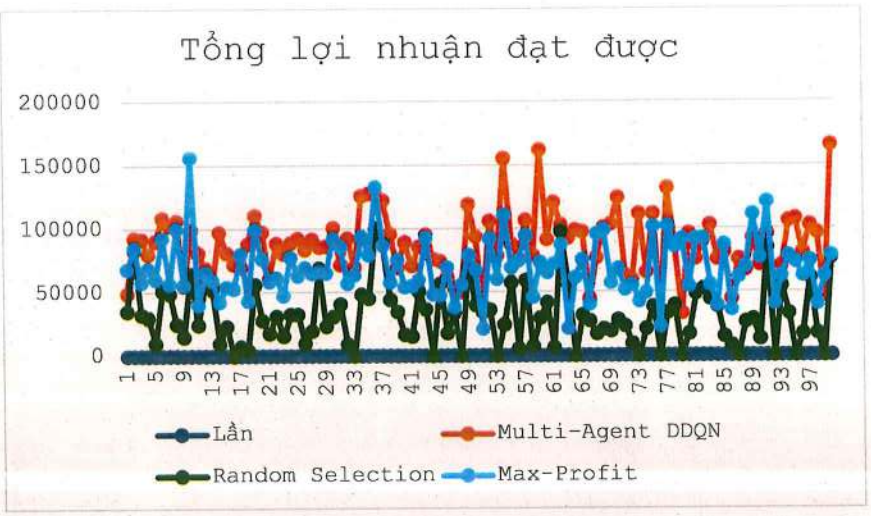
**Bảng 3.4. Kết quả trung bình 100 lần mô phỏng các thuật toán đầu tiên**

| | Multi-Agent DDQN | | Random Selection | | Max-Profit | |
|---|---|---|---|---|---|---|
| | Hoàn thành | Lợi nhuận | Hoàn thành | Lợi nhuận | Hoàn thành | Lợi nhuận |
| Trung bình | **9.4** | **87968.2** | 3.4 | 31026.86 | 7.45 | 69563.1 |

Từ kết quả thực nghiệm ta có thể thấy sau 100 lần chạy, ta có thể thấy đa số các lần chạy thì MA-DDQN cho kết quả vượt trội so với các phương pháp truyền thống (87/100 lần). Kết quả này có thể được lý giải bởi các phương pháp truyền thống thường bị giới hạn bởi giả định môi trường tĩnh và thường tối ưu một mục tiêu đơn, dẫn đến hiệu suất tổng thể kém hơn. Trong khi đó MA-DDQN có khả năng cập nhật chính sách liên tục dựa trên phản hồi từ môi trường động và có thể trực tiếp học từ hàm phần thưởng phức tạp. Điều này giúp tác tử phản ứng kịp thời trước những biến động mà thuật toán cố định không xử lý tốt. MA-DDQN đạt hiệu suất tổng thể cao nhất, với số nhiệm vụ hoàn thành và tổng lợi nhuận thường xuyên vượt trội so với Max-Profit và Random Selection.




**Hình 3.4. Kết quả tổng nhiệm vụ hoàn thành**



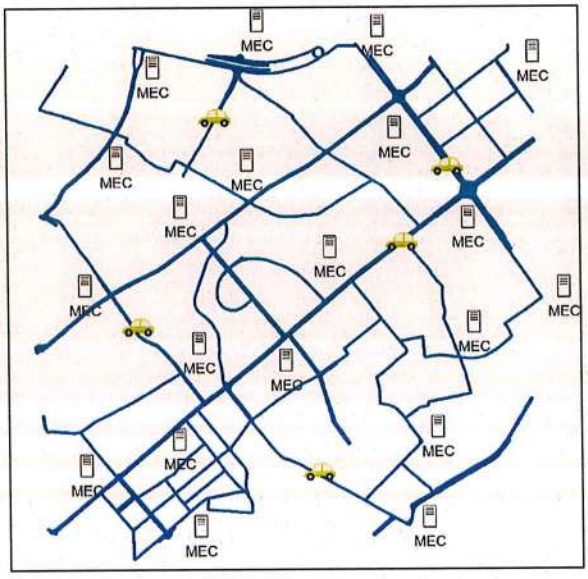

**Hình 3.5. Kết quả tổng lợi nhuận đạt được**

Kết quả trung bình cũng cho thấy số lượng nhiệm vụ hoàn thành trung bình cao hơn 26.17% so với Max-Profit; tổng lợi nhuận trung bình cũng cao hơn 26.45%. MA-DDQN học được chính sách phân phối tác vụ dựa trên tổng lợi nhuận toàn hệ thống, thể hiện khả năng phối hợp hiệu quả giữa các agent. Trong khi đó, Max-Profit chỉ tối ưu cục bộ và không cân nhắc tổng thể, dẫn đến đôi khi lợi nhuận thấp hơn so với Random Selection. Nhìn chung, MA-DDQN vượt trội về cả số nhiệm vụ hoàn thành và tổng lợi nhuận trung bình, chứng tỏ mô hình học được chiến lược phân phối tác vụ hiệu quả trong

môi trường đa tác vụ phức tạp. Random Selection thể hiện tính ngẫu nhiên, không ổn định đa số đều kém hiệu quả. Max-Profit ổn định về số nhiệm vụ nhưng không tối ưu hóa lợi nhuận tổng thể.

## 3.4 Xây dựng thử nghiệm hệ thống phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên

Trong thử nghiệm, tôi xây dựng một mô hình hệ thống phân phối nhiệm vụ cho 5 xe tự hành và 25 nhiệm vụ khác nhau trên nền tảng điện toán biên (MEC). Hệ thống bản đồ được xây dựng từ dữ liệu thực tế lấy từ OSM với trung tâm là Học viện Công nghệ Bưu Chính Viễn thông, bán kính 2.5km. Các đường nhỏ, không phù hợp với mô hình được loại bỏ. Kết quả bản đồ được thể hiện trong **Error! Reference source not found.**. Từ dữ liệu bản đồ thực tế, hệ thống tạo ra 568 đoạn đường, mỗi đoạn đường dài từ 10m đến 1.3km, trung bình các đoạn đường dài 203m. Các đoạn đường này có từ 2 đến 717 tác vụ, trung bình 63 tác vụ trên một đoạn đường. Thông tin chi tiết các nhiệm vụ, tác vụ được mô tả trong mục 3.1.1.




**Hình 3.6. Bản đồ mô phỏng khu vực Học viện Công nghệ Bưu Chính Viễn Thông**

Trong thực tế, số lượng xe và nhiệm vụ có thể lớn hơn nhiều. Để kiểm tra khả năng mở rộng của hệ thống, các xe có thể được chia thành nhiều nhóm 5 xe, mỗi nhóm vận hành mô hình 5 agent đã được huấn luyện, phối hợp cùng nhau để thực hiện nhiệm vụ. Ví dụ, trong một kịch bản với 100 xe và 1000 nhiệm vụ, có thể chia 100 xe thành 20 nhóm, mỗi nhóm đảm nhận 50 nhiệm vụ. Với cách tiếp cận này, hệ thống có thể mở rộng quy mô gần như không giới hạn mà không cần huấn luyện lại mô hình, đồng thời vẫn đảm bảo hiệu quả phân phối nhiệm vụ.

## 3.5 Kết luận chương 3

Trong Chương 3, tôi đã xây dựng và triển khai các thử nghiệm nhằm kiểm nghiệm và đánh giá mô hình đề xuất trong việc tối ưu hóa phân phối tác vụ cho xe tự hành trên nền tảng điện toán biên (MEC). Trước tiên, tôi giới thiệu dữ liệu thực nghiệm, bao gồm nhiệm vụ và tác vụ, các đặc điểm của xe tự hành và tác tử, cấu hình của MEC Servers, cũng như mô phỏng mạng truyền thông để tái hiện môi trường vận hành thực tế. Tiếp theo, quá trình cài đặt thực nghiệm được triển khai, trong đó các thuật toán phân phối nhiệm vụ – bao gồm MA-DDQN, Max-Profit và Random Selection – được áp dụng để phân phối các nhiệm vụ cho xe tự hành. Tôi đã sử dụng các chỉ số đánh giá như số nhiệm vụ hoàn thành, tổng lợi nhuận và hiệu quả phối hợp giữa các agent để đo lường hiệu quả của các thuật toán. Kết quả thực nghiệm cho thấy, MA-DDQN vượt trội hơn hẳn so với các chiến lược cơ bản, không chỉ hoàn thành nhiều nhiệm vụ hơn mà còn tối ưu hóa tổng lợi nhuận, đồng thời thể hiện khả năng phối hợp giữa các agent trong môi trường đa tác vụ. Cuối cùng là cách mở rộng mô hình cho nhiều xe và nhiều nhiệm vụ hơn bằng cách chia nhóm xe và phân phối nhiệm vụ tương ứng, cho phép áp dụng trong môi trường quy mô lớn mà không cần huấn luyện lại mô hình.

# KẾT LUẬN CHUNG

Trong đề án, tôi đã tiến hành nghiên cứu cơ sở lý thuyết về điện toán biên (MEC) và các nguyên tắc phân phối tác vụ cho xe tự hành. Quá trình nghiên cứu bao gồm việc phân tích các yếu tố liên quan như số lượng xe, số lượng nhiệm vụ, phụ thuộc giữa các nhiệm vụ, cũng như các chỉ số đánh giá hiệu quả phân phối. Những nghiên cứu này đã giúp làm rõ bài toán phân phối tác vụ trong môi trường đa tác vụ phức tạp, đồng thời làm cơ sở cho việc phát triển các giải pháp tối ưu hóa. Trên cơ sở đó, đề án đã đề xuất và triển khai thuật toán MA-DDQN để phân phối nhiệm vụ giữa các xe tự hành trên nền tảng MEC. Thuật toán này cho phép các agent phối hợp chặt chẽ với nhau, tối ưu hóa cả số lượng nhiệm vụ hoàn thành và tổng lợi nhuận hệ thống. Kết quả thực nghiệm cho thấy MA-DDQN vượt trội hơn hẳn so với các phương pháp cơ bản như Max-Profit và Random Selection, chứng minh hiệu quả trong việc phân phối tác vụ thông minh và khả năng mở rộng cho môi trường nhiều xe, nhiều nhiệm vụ. Những kết quả đạt được từ nghiên cứu và triển khai thuật toán không chỉ cung cấp một giải pháp thực tiễn cho quản lý tác vụ của xe tự hành mà còn tạo cơ sở cho các nghiên cứu mở rộng và ứng dụng trong các hệ thống vận hành thực tế với quy mô lớn hơn.

Trong tương lai, một hướng phát triển quan trọng là mô phỏng hệ thống sát thực tế hơn. Điều này bao gồm việc xây dựng môi trường thử nghiệm phản ánh chính xác hơn số lượng xe, nhiệm vụ, điều kiện giao thông và mạng truyền thông thực tế, từ đó nâng cao độ tin cậy của kết quả phân tích và đánh giá thuật toán phân phối tác vụ. Bên cạnh đó, đề án có thể được mở rộng sang bài toán offload tác vụ giữa các xe tự hành. Thay vì chỉ phân phối tác vụ giữa xe và các MEC Servers, các xe có thể trao đổi và chia sẻ nhiệm vụ với nhau, tận dụng khả năng xử lý phân tán nội bộ để tối ưu hóa hiệu suất hệ thống, giảm độ trễ và tăng tổng lợi nhuận. Hướng nghiên cứu này hứa hẹn nâng cao tính linh hoạt và hiệu quả của hệ thống phân phối tác vụ trong các môi trường xe tự hành thực tế.

# TÀI LIỆU THAM KHẢO


[1] Bing Lin, Kai Lin, Changhang Lin, Yu Lu, Ziqing Huang, Xinwei Chen (2021), "Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing" *Journal of Cloud Computing*

[2] Ashab Uddin, Ning Zhang, Ahmed Hamdi Sakr (2025) "Intelligent Offloading in Vehicular Edge Computing: A Comprehensive Review of Deep Reinforcement Learning Approaches and Architectures", *International Journal of Intelligent Systems*

[3] Penglin Dai, Xiao Wu, Huanlai Xing, Zhaofei Yu (2020), "A Probabilistic Approach for Cooperative Computation Offloading in MEC-Assisted Vehicular Networks", *IEEE Transactions on Intelligent Transportation Systems*

[4] Mohammad Fardad, Gabriel-Miro Muntean, Irina Tal (2024), "Decentralized Vehicular Edge Computing Framework for Energy-Efficient Task Coordination", *IEEE Vehicular Technology Conference – Spring 2024*

[5] Mohamed Abdel-Basset, Reda Mohamed, Amira Salam, Karam M. Sallam Ibrahim M. Hezam, Ibrahim Radwan (2024), "Intelligent Joint Optimization of Deployment and Task Scheduling for Mobile Users in Multi-UAV-Assisted MEC System", *International Journal of Intelligent Systems*

[6] Yiwei Zhang, Xin Cui, Qinghui Zhao (2025), "A Multi-Objective Joint Task Offloading Scheme for Vehicular Edge Computing", *Computers, Materials & Continua*

[7] Yaozhong Song (2018), "An Approach to QoS-based Task Distribution in Edge Computing Networks for IoT Applications", *Ph.D. dissertation,*

*Arizona State University, Tempe, AZ, USA*

[8] Hongru Li, Jiawei Shao, Hengtao He, Shenghui Song, Jun Zhang, Khaled B. Letaief (2025), "Tackling Distribution Shifts in Task-Oriented Communication with Information Bottleneck", *IEEE Journal on Selected Areas in Communications*

[9] Ting duan, Weiping Wang, Tao Wang, Xiaofan Chen, Xiaobo Li (2020), "Dynamic tasks scheduling model of uavcluster based on flexible network architecture", *IEEE Access, vol. 8, pp. 115448-115460*

[10] Nguyen Ngoc Hung, Nguyen Van Thieu, Luu Quang Trung, Vo Phi Son, Nguyen Van Dinh (2025), "A Metaheuristic Approach for Mission Assignment and Task Offloading in Open RAN-Enabled Intelligent Transport Systems", *IEEE Global Communications Conference*

[11] Neil de la Fuente, Daniel A. Vidal Guerra (2024), "A comparative study of deep reinforcement learning models: DQN vs PPO vs A2C", *arXiv preprint arXiv:2407.14151*

# PHỤ LỤC

*Phụ lục 1. Kết quả 100 lần thực hiện mô phỏng đánh giá*

| Lần | Multi-Agent DDQN | | Random Selection | | Max-Profit | |
|---|---|---|---|---|---|---|
| | Hoàn thành | Lợi nhuận | Hoàn thành | Lợi nhuận | Hoàn thành | Lợi nhuận |
| 1 | 6 | 49524.82 | 4 | 35445.09 | **7** | **68281.27** |
| 2 | **11** | **93422.81** | 9 | 83659.76 | 10 | 84997.63 |
| 3 | **10** | **92452.96** | 4 | 32865.49 | 7 | 58203.61 |
| 4 | **10** | **80676.14** | 5 | 30167.23 | 8 | 68892.47 |
| 5 | **8** | **91017.53** | 1 | 10119.54 | 5 | 60577.76 |
| 6 | **14** | **108981.4** | 6 | 51567.68 | 11 | 92872.66 |
| 7 | **11** | **102857** | 5 | 49229.73 | 7 | 56969.94 |
| 8 | **11** | **106893.9** | 3 | 25253.22 | 9 | 100171.8 |
| 9 | 7 | 55512.11 | 2 | 15163.59 | **8** | **55930.49** |
| 10 | 11 | 99741.71 | 8 | 64769.16 | **14** | **156402** |
| 11 | **8** | **81176.59** | 3 | 24719.96 | 4 | 40420.99 |
| 12 | **9** | **69349.62** | 8 | 58228.99 | 8 | 65740.13 |
| 13 | **7** | 57891.52 | 5 | 52392.69 | **7** | **59664.15** |
| 14 | **9** | **97915.44** | 1 | 10121.31 | 4 | 42686.28 |
| 15 | **10** | **81604.21** | 3 | 23285.21 | 7 | 54489.05 |
| 16 | **9** | **72580.5** | 0 | 0 | 7 | 52922.57 |
| 17 | 7 | 70111.2 | 1 | 7447.42 | **8** | **80499.94** |
| 18 | **8** | **88722.71** | 1 | 4426.533 | 5 | 43577.25 |
| 19 | **11** | **111032.8** | 5 | 56356.42 | 10 | 99682.32 |
| 20 | **11** | **97935.17** | 3 | 28274.1 | 8 | 76435.45 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 21 | **8** | **61288.08** | 3 | 17898.1 | 7 | 58677.23 |
| 22 | **10** | **88903.51** | 3 | 31455.29 | 8 | 61624.07 |
| 23 | **7** | **80968.39** | 2 | 15971.8 | 4 | 47650.3 |
| 24 | **9** | **89263.58** | 3 | 32774.68 | 7 | 77702.39 |
| 25 | **9** | **92973.3** | 3 | 32875.01 | 7 | 63100.41 |
| 26 | **8** | **84144.49** | 1 | 9969.136 | 6 | 69499.47 |
| 27 | **10** | **92245.2** | 3 | 19952.51 | 7 | 64940.4 |
| 28 | **9** | **86038.72** | 6 | 69038.94 | 7 | 66577.96 |
| 29 | **9** | **86562.07** | 2 | 23481.49 | 7 | 65078.34 |
| 30 | **11** | **101318.4** | 4 | 31012.52 | 10 | 92430.84 |
| 31 | 7 | 72017.78 | 5 | 41008.7 | **8** | **86104.84** |
| 32 | **9** | **92787.17** | 1 | 9344.085 | 6 | 57141.52 |
| 33 | **7** | **70432.63** | 0 | 0 | 6 | 63651.51 |
| 34 | **13** | **126139.6** | 6 | 48571.98 | 10 | 94208.5 |
| 35 | **12** | **127830.3** | 4 | 45298.47 | 8 | 79417.66 |
| 36 | 12 | 127331.1 | 9 | 100540.8 | **13** | **132990.9** |
| 37 | **11** | **122490.6** | 9 | 88818.87 | 9 | 87023.66 |
| 38 | **10** | **96289.85** | 4 | 44341.71 | 6 | 57756.65 |
| 39 | **8** | 69677.73 | 3 | 34913.39 | **8** | **75797** |
| 40 | **9** | **89701.15** | 2 | 16549.25 | 6 | 54253.7 |
| 41 | **10** | **71194.65** | 4 | 15330.43 | 7 | 54927.14 |
| 42 | **10** | **85974.04** | 6 | 50829.68 | 8 | 59675.67 |
| 43 | **10** | **95496.09** | 5 | 36396.52 | **10** | 93179.77 |
| 44 | **8** | **76482.03** | 0 | 0 | 5 | 48306.24 |
| 45 | **9** | **74685.98** | 6 | 56730.52 | 7 | 47508.54 |
| 46 | 7 | 61911.52 | 2 | 17770.38 | **8** | **69644.04** |

| | | | | | | |
|---|---|---|---|---|---|---|
| 47 | **7** | **53798.59** | 3 | 33918.73 | 5 | 38136.77 |
| 48 | 6 | 50124.15 | 0 | 0 | **8** | **56000.19** |
| 49 | **11** | **119263.2** | 7 | 69529.97 | 8 | 79296.52 |
| 50 | **10** | **94116.3** | 4 | 40035.26 | 7 | 67683.36 |
| 51 | **4** | **39065.31** | 3 | 34280.49 | 3 | 20983.79 |
| 52 | **11** | **106204.4** | 3 | 36145.26 | **11** | 92292.24 |
| 53 | **7** | **78981.19** | 0 | 0 | 5 | 60171.22 |
| 54 | **13** | **155561.7** | 2 | 24592.42 | 9 | 110718.7 |
| 55 | **10** | **88173.57** | 5 | 57695.55 | 6 | 69448.94 |
| 56 | **10** | **76101.05** | 1 | 5632.151 | 8 | 74817.89 |
| 57 | **10** | **106979.7** | 5 | 59201.93 | 9 | 94898.14 |
| 58 | **9** | **79219.11** | 2 | 7631.066 | 5 | 46119.39 |
| 59 | **14** | **162244.6** | 3 | 29524.4 | 7 | 73414.6 |
| 60 | **10** | **91675.64** | 4 | 41841 | 7 | 65477.18 |
| 61 | **12** | **120950.9** | 1 | 6340.607 | 8 | 72210.5 |
| 62 | **12** | **103496.5** | 10 | 97746.29 | 10 | 87603.8 |
| 63 | **9** | **92762.92** | 5 | 55777.87 | 3 | 21291.01 |
| 64 | **9** | **98942.38** | 0 | 0 | 6 | 61432.91 |
| 65 | **12** | **97601.48** | 5 | 31775 | 9 | 75480.52 |
| 66 | **6** | **44866.96** | 3 | 27484.21 | 4 | 40352.34 |
| 67 | **11** | 77197.21 | 2 | 17020.8 | **11** | **95827.03** |
| 68 | **10** | **101024.6** | 2 | 20940.93 | 9 | 98930.97 |
| 69 | **10** | **103758.7** | 2 | 19358.61 | 6 | 57243.51 |
| 70 | **11** | **124450.2** | 3 | 28142.5 | 7 | 68671.48 |
| 71 | **9** | **63587.53** | 4 | 23223.08 | 7 | 51911.76 |
| 72 | **7** | **60166.31** | 1 | 9262.122 | 5 | 55419.5 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 73 | **11** | **111676.1** | 0 | 0 | 5 | 42756.42 |
| 74 | **8** | **65723.01** | 3 | 21164.86 | 6 | 49790.57 |
| 75 | **10** | **111729.2** | 5 | 38531.87 | 9 | 101588.3 |
| 76 | **5** | **31503.81** | 0 | 0 | 4 | 23144.27 |
| 77 | **14** | **132178.5** | 3 | 33846.49 | 10 | 102043.5 |
| 78 | **10** | **87098.42** | 5 | 39774.9 | **10** | 85897.52 |
| 79 | 4 | 33298.34 | 0 | 0 | **9** | **92034.89** |
| 80 | **10** | **96380.88** | 2 | 17026.2 | 6 | 54573 |
| 81 | 8 | 76116.65 | 5 | 48792.28 | **9** | **92038.87** |
| 82 | **11** | 90440.62 | 6 | 52500.28 | **11** | **92483.93** |
| 83 | **10** | **103000.7** | 5 | 44702.1 | 7 | 55925.61 |
| 84 | **10** | **76024.56** | 6 | 38639.68 | 6 | 39490.1 |
| 85 | **10** | 81650.14 | 2 | 14886.79 | 9 | **87380.35** |
| 86 | **4** | **44584.14** | 1 | 7939.292 | **4** | 35986.77 |
| 87 | **9** | **75959.96** | 0 | 0 | 7 | 62109.05 |
| 88 | **9** | **67404.75** | 5 | 26451.92 | 9 | 70118.37 |
| 89 | 10 | 92544.38 | 3 | 28299.47 | **12** | **111192.2** |
| 90 | **8** | 71216.51 | 1 | 12965.73 | 7 | **77382.54** |
| 91 | 11 | 95866.82 | 8 | 85129.53 | **12** | **121082.6** |
| 92 | **6** | **51601.49** | 0 | 0 | 4 | 40884.32 |
| 93 | **8** | **70796.54** | 5 | 54659.47 | 7 | 62547.38 |
| 94 | **10** | **106173.1** | 3 | 33067.39 | 7 | 78753.95 |
| 95 | **11** | **107957.3** | 0 | 0 | 7 | 73547.53 |
| 96 | **9** | **81390.08** | 3 | 17003.98 | 7 | 62731.8 |
| 97 | **12** | **102677.7** | 6 | 61351.57 | 8 | 74723.76 |
| 98 | **11** | **96455.51** | 3 | 17439.1 | 5 | 40495.15 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 99 | 6 | 45721.55 | 0 | 0 | **7** | **62802.4** |
| 100 | **15** | **165787.1** | 7 | 77045.39 | 9 | 78684.35 |
| AVG | **9.4** | **87968.2** | 3.4 | 31026.86 | 7.45 | 69563.1 |

turnitin Trang 2/92 - Tổng quan về tính toàn vẹn ID bài nộp trn:oid:::1:3473021738

## 18% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

### Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn
- Kết quả trùng khớp nhỏ (ít hơn 8 từ)

### Các tiêu chí loại trừ

- 5 nguồn đã loại trừ

### Nguồn hàng đầu

16% Nguồn Internet
5% Ấn bản
5% Bài tập được nộp (bài của học sinh)

**Học viên** **Người hướng dẫn khoa học**

Đỗ Thị Liên