ĐẠI HỌC KINH TẾ TP. HỒ CHÍ MINH
VIỆN CÔNG NGHỆ THÔNG MINH VÀ TƯƠNG TÁC

TIỂU LUẬN CUỐI KỲ
HỌC PHẦN Công Nghệ Mới trong logistics

Đề tài:

Website tối ưu lộ trình giao hàng bằng AI/ML cho Chuỗi cửa hàng Bách hoá xanh

GVHD: TS. Nguyễn Thiên Bảo

Nhóm thực hiện: Lê Nguyên Hưng - 31231027406
Trịnh Hoàng Phát - 31231021847
Đỗ Trương Nhật Quang - 31231027174
Thân Minh Nghĩa - 31231026396
Đường Công Hoan - 31231025325
Vũ Trần Hải Long - 31231021846
Hoàng Quốc Việt - 31231024487

TP. Hồ Chí Minh, tháng 05/2025

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH CỦA TỪNG THÀNH VIÊN TRONG NHÓM

Tên thành viên	Mã số sinh viên	Nhiệm vụ phân công	Mức độ hoàn thành
Lê Nguyên Hưng	31231027406	Làm slide, chỉnh sửa, nâng cấp mô hình AI, thuyết trình, soạn dataset cho mô hình.	100%
Trịnh Hoàng Phát	31231021847	Chỉnh sửa, nâng cấp mô hình AI, thuyết trình, soạn dataset cho mô hình.	100%
Đỗ Trương Nhật Quang	31231027174	Chỉnh sửa, nâng cấp mô hình AI, thuyết trình, soạn dataset cho mô hình.	100%
Thân Minh Nghĩa	31231026396	Xây dựng mô hình AI, chỉnh sửa, nâng cấp mô hình AI, hoàn thiện thuật toán cho mô hình AI	100%
Đường Công Hoan	31231025325	Làm giao diện web, tạo liên kết giữa frontend và backend, chỉnh sửa, nâng cấp mô hình AI.	100%
Vũ Trần Hải Long	31231021846	Làm giao diện web, tạo liên kết giữa frontend và backend, chỉnh sửa, nâng cấp mô hình AI.	100%
Hoàng Quốc Việt	31231024487	Làm report, chỉnh sửa, nâng cấp mô hình AI, thuyết trình	100%

MỤC LỤC

3.1.2. Thông tin các loại xe (VEHICLE_TYPES_CONFIG): Số lượng, tải trọng, vận tốc trung bình, chi phí nhiên liệu, lương tài xế, hệ số giảm tốc, thời gian hoạt động tối đa.

3.2. Tính toán ma trận khoảng cách và thời gian

3.2.1. Sử dụng API của OpenRouteService (calculate_distance_matrix)

3.2.2. Kết quả: Distance_matrix.xlsx

3.3. Xây dựng đặc trưng cho GNN (GNNFeature trong VRP_2_Model.py)

3.3.1. Đặc trưng nút (Node features): Tiền phạt, nhu cầu, thời gian phục vụ, tọa độ.

3.3.2. Đặc trưng cạnh (Edge features): Khoảng cách, thời gian di chuyển (theo từng loại xe, có nhiễu), chi phí nhiên liệu (theo từng loại xe).

CHƯƠNG 4: THIẾT KẾ VÀ TRIỂN KHAI MÔ HÌNH

4.1. Mô hình hóa bài toán VRP dưới dạng bài toán Học tăng cường (VRPEnvGym trong VRP_3_Env.py)

4.1.1. Định nghĩa Môi trường (Environment)

4.1.2. Định nghĩa Trạng thái (State):

- Vector bối cảnh (Context Embedding)
- Mask các hành động hợp lệ
- Vector biểu diễn của nút hiện tại
- Trạng thái xe (tải trọng/tải trọng max, thời gian/thời gian max)

4.1.3. Định nghĩa Hành động (Action): Chọn cửa hàng tiếp theo để đi hoặc quay về kho.

4.1.4. Định nghĩa Hàm thưởng (Reward Function):

- Thưởng khi đáp ứng nhu cầu (REWARD_DEMAND_FACTOR).
- Phạt chi phí nhiên liệu (REWARD_FUEL_FACTOR).
- Phạt tải trọng còn lại khi kết thúc chuyến (REWARD_REMAIN_LOAD).

4.1.5. Điều kiện kết thúc một lượt chơi (Episode Termination).

4.1.6. Xử lý các ràng buộc: Tải trọng xe, thời gian hoạt động tối đa của xe, tính toán thời gian di chuyển thực tế dựa trên tải trọng.

4.2. Kiến trúc mô hình AI (VRP_2_Model.py, VRP_4_PPO.py)

4.2.1. Bộ mã hóa GNN (GNNEdgeAttr - sử dụng TransformerConv):

- Mục đích: Học vector biểu diễn (node embeddings) cho các nút (cửa hàng, kho) từ đặc trưng nút và cấu trúc đồ thị.

4.2.2. Bộ nhúng bối cảnh (ContextEmbedding):

- Kết hợp thông tin trạng thái xe, vector biểu diễn các nút chưa thăm, nút hiện tại, và kho để tạo vector bối cảnh động.

4.2.3. Mạng Tác tử (Actor Network - AttentionDecoder):

- Sử dụng cơ chế Attention để quyết định hành động (nút tiếp theo) dựa trên vector bối cảnh, vector biểu diễn nút hiện tại và các vector biểu diễn của các nút ứng viên.

4.2.4. Mạng Phê bình (Critic Network - CriticValueNetwork):

- Đánh giá giá trị của trạng thái hiện tại (vector bối cảnh).

4.2.5. Chính sách PPO tùy chỉnh (VRPActorCriticPolicy):

- Tích hợp các thành phần trên vào kiến trúc PPO.

4.3. Quá trình huấn luyện mô hình (train_vrp_model trong VRP_0_Main.py)

4.3.1. Khởi tạo môi trường huấn luyện song song (SubprocVecEnv).

4.3.2. Cấu hình các siêu tham số cho PPO (ppo_config_train).

4.3.3. Tiến hành huấn luyện và lưu mô hình (vrp_ppo_model.zip).

4.4. Quá trình suy luận và tạo tuyến đường (run_vrp_inference trong VRP_0_Main.py)

4.4.1. Tải mô hình đã huấn luyện.

4.4.2. Chạy mô hình trên môi trường đánh giá để tạo ra các tuyến đường.

4.4.3. Lưu kết quả tuyến đường vào file data.json.

CHƯƠNG 5: XÂY DỰNG GIAO DIỆN NGƯỜI DÙNG

5.1. Công nghệ sử dụng

5.1.1. Backend: FastAPI (Python).

5.1.2. Frontend: HTML, CSS, JavaScript.

5.1.3. Thư viện bản đồ: Leaflet.

5.1.4. Thư viện xử lý Excel: SheetJS, PapaParse.

5.2. Chức năng của hệ thống

5.2.1. Giao diện Quản lý (manager.html, manager_jav.js):

- Tải file input.xlsx lên server (/api/upload-excel/).
- Kích hoạt quá trình xử lý VRP (/api/process-vrp/).
- Theo dõi trạng thái xử lý (/api/processing-status/).
- Tải và xử lý dữ liệu (input.xlsx từ /static_uploads/, data.json từ /api/get-routes-from-json/).
- Hiển thị thông tin cửa hàng, tìm kiếm cửa hàng.
- Hiển thị các tuyến đường tối ưu trên bản đồ cho từng xe (sử dụng OpenRouteService API để vẽ đường đi giữa các điểm).

5.2.2. Giao diện Tài xế (driver.html, driver_jav.js):

- (Hiện tại đang có vẻ lấy dữ liệu từ localStorage hoặc cần một endpoint riêng mà chưa có trong main.py - giả định là sẽ hiển thị tuyến đường đã được tính toán cho xe cụ thể).
- Tìm kiếm cửa hàng, hiển thị bản đồ.

5.3. Luồng hoạt động của hệ thống

CHƯƠNG 6: KẾT QUẢ VÀ ĐÁNH GIÁ

6.1. Kết quả thực nghiệm

6.1.1. Mô tả bộ dữ liệu thử nghiệm (có thể là dữ liệu mẫu input.xlsx).

6.1.2. Các tuyến đường được tạo ra (trực quan hóa từ data.json).

6.1.3. Tổng chi phí (nhiên liệu, lương, phạt) của giải pháp.

6.2. So sánh hiệu quả với phương pháp Heuristic truyền thống

6.2.1. Tiêu chí so sánh: Tổng chi phí, thời gian tính toán, khả năng thích ứng với ràng buộc phức tạp.
6.2.2. (Nếu có) Kết quả chạy thử một thuật toán heuristic đơn giản (ví dụ: Nearest Neighbor hoặc Clarke & Wright) trên cùng bộ dữ liệu.
6.2.3. Phân tích ưu nhược điểm của giải pháp GNN+PPO so với Heuristic (chi tiết ở mục dưới).
6.3. Đánh giá khả năng mở rộng và hạn chế của mô hình

CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
7.1. Tóm tắt các kết quả đạt được
7.2. Đóng góp của dự án
7.3. Hạn chế của dự án
7.4. Hướng phát triển trong tương lai
7.4.1. Cải thiện hàm thưởng, kiến trúc mô hình.
7.4.2. Thử nghiệm với các bộ dữ liệu lớn hơn, phức tạp hơn.
7.4.3. Tích hợp các yếu tố động (thời gian giao thông thực tế, yêu cầu mới phát sinh).
7.4.4. Hoàn thiện giao diện tài xế để có thể cập nhật trạng thái giao hàng.

TÀI LIỆU THAM KHẢO

CHƯƠNG 1: GIỚI THIỆU CHI TIẾT

1.1. Bối cảnh và Tính cấp thiết của Bài toán Định tuyến Phương tiện (Vehicle Routing Problem - VRP)

1.1.1. Định nghĩa và Phân loại Bài toán VRP

Bài toán Định tuyến Phương tiện (Vehicle Routing Problem - VRP), được Dantzig và Ramser giới thiệu lần đầu tiên vào năm 1959 [Dantzig & Ramser, 1959 - *Cần tìm nguồn gốc chính xác nếu có thể, nếu không thì trích dẫn một sách tổng quan uy tín về VRP như của Toth & Vigo*], là một trong những bài toán tối ưu hóa tổ hợp được nghiên cứu rộng rãi nhất trong lĩnh vực vận trù học (Operations Research) và khoa học máy tính. Về cơ bản, VRP yêu cầu tìm ra một tập hợp các tuyến đường tối ưu cho một đội xe (có thể đồng nhất hoặc không đồng nhất) xuất phát từ một hoặc nhiều kho (depots) để phục vụ một tập hợp các khách hàng (customers) có nhu cầu xác định, và quay trở lại kho sau khi hoàn thành nhiệm vụ. Mục tiêu thường là tối thiểu hóa tổng chi phí vận hành, có thể bao gồm tổng quãng đường di chuyển, tổng thời gian hoạt động, số lượng xe sử dụng, hoặc một sự kết hợp của các yếu tố này.

Sự phức tạp của VRP nằm ở không gian tìm kiếm giải pháp khổng lồ. Với N khách hàng, số lượng tuyến đường khả thi có thể tăng lên theo cấp số nhân, khiến VRP được xếp vào loại bài toán NP-khó (NP-hard) [Lenstra & Rinnooy Kan, 1981 - *Nguồn kinh điển về NP-hardness của VRP*]. Điều này có nghĩa là không tồn tại thuật toán nào có thể tìm ra giải pháp tối ưu chính xác cho các bài toán VRP quy mô lớn trong thời gian đa

thức. Ví dụ, như đã đề cập trong tài liệu tham khảo [1], chỉ với 15 điểm, đã có hơn 600 tỷ tuyến đường khả thi, minh họa cho thách thức tính toán mà các nhà nghiên cứu và các chuyên gia logistics phải đối mặt.

Do tính đa dạng của các ứng dụng thực tế, VRP đã phát triển thành nhiều biến thể khác nhau, mỗi biến thể bổ sung thêm các ràng buộc hoặc mục tiêu cụ thể [Toth & Vigo, 2014; Eksioglu et al., 2009 - *Thêm nguồn tổng quan về các biến thể VRP*]:

- **Capacitated VRP (CVRP):** Biến thể phổ biến nhất, trong đó mỗi xe có một tải trọng (capacity) giới hạn, và tổng nhu cầu của các khách hàng được phục vụ trên một tuyến đường không được vượt quá tải trọng này [3, 8]. Đây là biến thể cốt lõi được xem xét trong nghiên cứu này.
- **VRP with Time Windows (VRPTW):** Mỗi khách hàng yêu cầu được phục vụ trong một khoảng thời gian nhất định (time window) [13, 15]. Việc vi phạm cửa sổ thời gian có thể bị phạt hoặc không được phép.
- **Distance-Constrained VRP (DCVRP):** Tổng chiều dài của mỗi tuyến đường không được vượt quá một giới hạn nhất định.
- **Multiple Depot VRP (MDVRP):** Các xe có thể xuất phát từ nhiều kho khác nhau.
- **VRP with Pickups and Deliveries (VRPPD):** Xe không chỉ giao hàng mà còn có thể nhận hàng tại các điểm khác nhau và vận chuyển đến các điểm khác hoặc về kho.
- **Periodic VRP (PVRP):** Khách hàng yêu cầu được phục vụ nhiều lần trong một khoảng thời gian kế hoạch, với các tần suất khác nhau.
- **Stochastic VRP (SVRP):** Một hoặc nhiều yếu tố của bài toán (ví dụ: nhu cầu khách hàng, thời gian di chuyển) là không chắc chắn và được mô hình hóa bằng các biến ngẫu nhiên.
- **Dynamic VRP (DVRP):** Các yêu cầu mới hoặc thay đổi có thể xuất hiện trong quá trình xe đang vận hành, đòi hỏi việc điều chỉnh tuyến đường theo thời gian thực.

Việc hiểu rõ các biến thể này và lựa chọn mô hình phù hợp là bước đầu tiên quan trọng trong việc giải quyết hiệu quả các bài toán VRP trong thực tế.

1.1.2. Tầm quan trọng Chiến lược của VRP trong Logistics và Chuỗi Cung ứng Hiện đại

Trong môi trường kinh doanh toàn cầu hóa và cạnh tranh khốc liệt ngày nay, hiệu quả của hoạt động logistics và quản lý chuỗi cung ứng đã trở thành một yếu tố quyết định lợi thế cạnh tranh của doanh nghiệp. Chi phí logistics, theo nhiều báo cáo, có thể chiếm một tỷ trọng đáng kể trong tổng chi phí sản phẩm (từ vài phần trăm đến hơn 30% tùy ngành nghề và khu vực). Trong đó, chi phí vận tải thường là thành phần lớn nhất. Do đó, việc tối ưu hóa các hoạt động vận tải, mà VRP là một bài toán cốt lõi, mang lại những lợi ích chiến lược sâu rộng:

- **Tối ưu hóa Chi phí Vận hành Trực tiếp:**
    - **Giảm chi phí nhiên liệu:** Tuyến đường ngắn hơn và kế hoạch di chuyển thông minh hơn giúp giảm lượng nhiên liệu tiêu thụ, một trong những chi phí biến đổi lớn nhất.

- **Giảm chi phí nhân công:** Thời gian hoạt động của tài xế được tối ưu hóa, giảm thiểu thời gian làm thêm giờ và tăng năng suất lao động.
- **Giảm chi phí bảo trì phương tiện:** Quãng đường di chuyển ít hơn đồng nghĩa với việc giảm hao mòn xe cộ, kéo dài tuổi thọ phương tiện và giảm tần suất bảo trì.
- **Tối ưu hóa số lượng xe sử dụng:** Một hệ thống định tuyến hiệu quả có thể giúp doanh nghiệp phục vụ cùng một lượng khách hàng với ít xe hơn, hoặc mở rộng khả năng phục vụ mà không cần tăng đáng kể đội xe, từ đó giảm chi phí đầu tư và chi phí cố định.

- **Nâng cao Chất lượng Dịch vụ và Sự hài lòng của Khách hàng:**
  - **Giao hàng đúng hẹn:** VRP giúp lập kế hoạch giao hàng chính xác hơn, đáp ứng các cam kết về thời gian với khách hàng. Điều này đặc biệt quan trọng trong các ngành hàng nhạy cảm về thời gian như thực phẩm, dược phẩm, hoặc thương mại điện tử.
  - **Giảm thời gian chờ đợi của khách hàng:** Tuyến đường được tối ưu giúp giảm thời gian di chuyển giữa các điểm, rút ngắn thời gian giao hàng tổng thể.
  - **Tăng độ tin cậy của dịch vụ:** Một hệ thống định tuyến ổn định và hiệu quả giúp giảm thiểu sai sót, chậm trễ, và các vấn đề phát sinh trong quá trình vận chuyển.

- **Tăng cường Hiệu quả Sử dụng Nguồn lực và Năng lực Hoạt động:**
  - **Cân bằng tải cho tài xế và phương tiện:** Phân bổ công việc một cách hợp lý hơn giữa các tài xế và xe, tránh tình trạng quá tải hoặc dưới tải.
  - **Tăng thông lượng (throughput):** Khả năng phục vụ nhiều khách hàng hơn hoặc xử lý nhiều đơn hàng hơn trong cùng một khoảng thời gian với cùng một nguồn lực.
  - **Cải thiện khả năng ra quyết định:** Cung cấp cho nhà quản lý logistics các công cụ và thông tin để đưa ra quyết định tốt hơn về việc phân bổ nguồn lực, lập kế hoạch và điều phối.

- **Đóng góp vào Phát triển Bền vững:**
  - **Giảm phát thải khí nhà kính:** Việc giảm quãng đường di chuyển và tối ưu hóa việc sử dụng nhiên liệu trực tiếp góp phần giảm lượng khí thải CO_2 và các chất gây ô nhiễm khác, phù hợp với các mục tiêu phát triển bền vững và trách nhiệm xã hội của doanh nghiệp.
  - **Giảm tắc nghẽn giao thông:** Các tuyến đường hiệu quả hơn có thể góp phần giảm bớt áp lực lên hệ thống giao thông đô thị, đặc biệt là ở các thành phố lớn.

- **Tạo dựng Lợi thế Cạnh tranh Bền vững:**
  Trong một thị trường mà khách hàng ngày càng yêu cầu cao hơn về tốc độ, chi phí và chất lượng dịch vụ, khả năng tối ưu hóa logistics hiệu quả không còn là một lựa chọn mà là một yêu cầu bắt buộc. Doanh nghiệp nào làm tốt điều này sẽ có được lợi thế cạnh tranh đáng kể, từ việc cung cấp sản phẩm với giá tốt hơn, dịch vụ nhanh hơn, đến việc xây dựng hình ảnh thương hiệu uy tín và có trách nhiệm.

Sự phức tạp ngày càng tăng của chuỗi cung ứng toàn cầu, sự bùng nổ của thương mại điện tử với các mô hình giao hàng đa dạng (giao hàng trong ngày, giao hàng theo khung giờ), và áp lực giảm chi phí liên tục đặt ra yêu cầu ngày càng cao đối với các giải pháp VRP. Điều này thúc đẩy sự phát triển không ngừng của các phương pháp và công nghệ mới nhằm giải quyết bài toán này một cách hiệu quả hơn.

1.2. Bối cảnh Ứng dụng và Thách thức VRP tại Chuỗi Bán lẻ Bách Hóa Xanh

1.2.1. Tổng quan về Bách Hóa Xanh và Mạng lưới Logistics

Bách Hóa Xanh (BHX), được thành lập vào cuối năm 2015, là một thành viên của Công ty Cổ phần Đầu tư Thế Giới Di Động (MWG), một trong những nhà bán lẻ hàng đầu Việt Nam. BHX nhanh chóng mở rộng và trở thành một trong những chuỗi cửa hàng tiện lợi và siêu thị mini chuyên bán lẻ hàng tiêu dùng nhanh (FMCG) và thực phẩm tươi sống lớn nhất cả nước. Tính đến các báo cáo gần đây (ví dụ, cuối năm 2023 hoặc đầu năm 2024), BHX vận hành một mạng lưới có thể lên đến hơn 1.700 cửa hàng (con số này có thể biến động do quá trình tái cấu trúc và tối ưu hóa mạng lưới của công ty, ví dụ, các nguồn [5, 7] có thể cung cấp các con số tại những thời điểm khác nhau, cần kiểm tra số liệu cập nhật nhất nếu có) tập trung chủ yếu ở các tỉnh thành phía Nam và đang dần mở rộng ra các khu vực khác.

Hệ thống logistics của BHX được xây dựng để hỗ trợ mạng lưới cửa hàng rộng lớn này, bao gồm:

- **Trung tâm phân phối (Distribution Centers - DCs):** Các kho lớn, hiện đại, đóng vai trò là đầu mối tiếp nhận hàng hóa từ các nhà cung cấp và phân phối đến các kho vệ tinh hoặc trực tiếp đến một số cửa hàng. Các báo cáo trước đây đề cập đến khoảng 8 trung tâm phân phối [6].
- **Kho vệ tinh (Satellite Warehouses/Hubs):** Các kho nhỏ hơn, nằm gần các cụm cửa hàng hơn, giúp giảm quãng đường vận chuyển cuối cùng và tăng tốc độ bổ sung hàng hóa. Số lượng kho vệ tinh có thể lên đến hàng chục (ví dụ 65 kho như đề cập trong kế hoạch nghiên cứu của bạn, cần kiểm tra nguồn gốc của con số này) và thường xuyên được điều chỉnh để tối ưu hóa.
- **Đội xe vận chuyển:** BHX có thể sử dụng một đội xe đa dạng, bao gồm cả xe tải lớn vận chuyển giữa các DC và kho vệ tinh, và các xe tải nhỏ hơn (light trucks) để phân phối đến từng cửa hàng. Thông tin về việc BHX "tăng tải trọng xe tải" [9, 10] cho thấy sự chú trọng vào việc tối ưu hóa năng lực vận chuyển.

1.2.2. Đặc thù Hoạt động Giao nhận và Yêu cầu đối với Hệ thống VRP của BHX

Hoạt động giao nhận của BHX mang những đặc thù riêng, đặt ra những yêu cầu và thách thức cụ thể cho bài toán VRP:

- **Ưu tiên Hàng tươi sống (Fresh Produce):** Một tỷ trọng đáng kể (có thể lên đến 40-50% doanh thu ở một số thời điểm) trong danh mục sản phẩm của BHX là hàng tươi sống (rau củ quả, thịt cá). Điều này đòi hỏi quy trình vận chuyển và bổ sung hàng hóa phải cực kỳ nhanh chóng để đảm bảo độ tươi ngon và giảm thiểu hư hỏng. Các tuyến đường cần được tối ưu hóa không chỉ về chi phí mà còn về thời gian. Yếu tố này có thể dẫn đến sự cần thiết của các ràng buộc về cửa sổ thời gian (time windows) chặt chẽ hơn hoặc các mục tiêu ưu tiên thời gian trong hàm tối ưu.
- **Tần suất Bổ sung Hàng hóa Cao:** Do đặc thù hàng tươi sống và mô hình cửa hàng tiện lợi, nhiều mặt hàng cần được bổ sung hàng ngày, thậm chí nhiều lần trong ngày đối với một số cửa hàng có lưu lượng khách lớn. Điều này tạo ra một khối lượng lớn các yêu cầu vận chuyển cần được xử lý liên tục.
- **Nhu cầu Đa dạng và Biến động tại các Cửa hàng:** Mỗi cửa hàng có cơ cấu nhu cầu khác nhau (dựa trên vị trí, quy mô, đối tượng khách hàng địa phương) và nhu cầu này có thể biến động mạnh theo ngày trong tuần, các dịp lễ tết, hoặc các chương trình khuyến mãi. Hệ thống VRP cần có khả năng thích ứng với sự biến động này.
- **Mạng lưới Đô thị và Giao thông Phức tạp:** Phần lớn cửa hàng BHX tập trung tại các khu vực đô thị và thành phố lớn như TP. Hồ Chí Minh, nơi có mật độ giao thông cao và thường xuyên xảy ra ùn tắc [51, 52]. Điều này làm cho việc ước tính thời gian di chuyển trở nên khó khăn và đòi hỏi các giải pháp định tuyến có khả năng xem xét yếu tố giao thông (ít nhất là ở mức độ trung bình hoặc dự đoán).
- **Mục tiêu Phát triển Kênh Online và Giao hàng Nhanh:** Kế hoạch triển khai dịch vụ giao hàng trực tuyến trong 2 giờ của BHX Online [7] đặt ra một thách thức VRP hoàn toàn mới và cấp bách. Điều này không chỉ yêu cầu tối ưu hóa các tuyến đường từ kho đến điểm giao cuối (khách hàng) mà còn có thể liên quan đến việc sử dụng chính các cửa hàng làm điểm lấy hàng (store fulfillment) hoặc các kho đô thị siêu nhỏ (micro-fulfillment centers). Bài toán có thể chuyển hướng sang các biến thể phức tạp hơn như VRPPD hoặc các mô hình two-echelon VRP.
- **Quản lý Chi phí Nghiêm ngặt:** Trong ngành bán lẻ với biên lợi nhuận thường không cao, việc kiểm soát chi phí logistics là một yếu tố sống còn. BHX, sau giai đoạn mở rộng nhanh chóng, đã tập trung vào việc cải thiện hiệu quả hoạt động và lợi nhuận [7], trong đó tối ưu hóa chi phí vận chuyển đóng vai trò quan trọng.
- **Đội xe và Ràng buộc Vận hành:** Việc sử dụng các loại xe khác nhau với tải trọng, chi phí vận hành, và các quy định về giờ làm việc của tài xế (ví dụ, tại Việt Nam, có quy định về thời gian lái xe liên tục tối đa và tổng thời gian làm việc trong ngày [11, 12]) cần được tích hợp vào mô hình VRP.

1.2.3. Thách thức Hiện hữu và Cơ hội từ việc Áp dụng Giải pháp VRP Tiên tiến

Từ những đặc thù trên, các thách thức chính trong việc tối ưu hóa VRP cho BHX bao gồm:

- **Quy mô và Độ phức tạp Tính toán:** Số lượng lớn cửa hàng, kho, và các ràng buộc đa dạng tạo ra một không gian tìm kiếm giải pháp cực kỳ lớn, vượt quá khả năng của các phương pháp giải quyết thủ công hoặc các thuật toán đơn giản.
- **Tính Động và Bất định:** Nhu cầu biến động, điều kiện giao thông thay đổi, và các yêu cầu mới từ kênh online đòi hỏi một hệ thống có khả năng thích ứng và cập nhật kế hoạch một cách linh hoạt.
- **Yêu cầu về Thời gian Phản hồi:** Giải pháp định tuyến cần được tạo ra nhanh chóng để đáp ứng yêu cầu vận hành hàng ngày, đặc biệt là cho các kế hoạch giao hàng nhanh.
- **Tích hợp Dữ liệu:** Việc thu thập, làm sạch và tích hợp dữ liệu từ nhiều nguồn khác nhau (thông tin cửa hàng, trạng thái kho, thông tin xe, dữ liệu giao thông) là một thách thức kỹ thuật.

Tuy nhiên, việc vượt qua những thách thức này và triển khai thành công một giải pháp VRP tiên tiến mang lại những cơ hội to lớn cho BHX:

- **Tiết kiệm Chi phí Vận hành Đáng kể:** Giảm từ 5-20% (hoặc hơn) chi phí vận tải là hoàn toàn có thể đạt được với các hệ thống VRP hiệu quả.
- **Nâng cao Hiệu quả Chuỗi Cung ứng Lạnh:** Đảm bảo hàng tươi sống được vận chuyển nhanh chóng, duy trì chất lượng và giảm tỷ lệ hao hụt.
- **Cải thiện Trải nghiệm Khách hàng (Cả Cửa hàng và Người tiêu dùng cuối):** Hàng hóa luôn sẵn có, tươi mới, và giao hàng online đúng hẹn.
- **Tăng cường Năng lực Cạnh tranh:** Tạo ra lợi thế về chi phí, tốc độ và độ tin cậy so với các đối thủ.
- **Nền tảng cho các Sáng kiến Số hóa Khác:** Một hệ thống VRP thông minh có thể là nền tảng để phát triển các ứng dụng khác như theo dõi đơn hàng thời gian thực, phân tích hiệu suất logistics, và dự báo nhu cầu.

Chính vì những lý do này, việc nghiên cứu và ứng dụng các phương pháp học máy và học sâu, như GNN và RL, cho bài toán VRP tại BHX là một hướng đi có ý nghĩa khoa học và giá trị thực tiễn cao.

1.3. Mục tiêu và Câu hỏi Nghiên cứu

1.3.1. Mục tiêu Tổng quát của Nghiên cứu

Mục tiêu tổng quát của nghiên cứu này là khám phá và đánh giá khả năng ứng dụng của một phương pháp lai ghép giữa Mạng Nơ-ron Đồ thị (GNN) và Học tăng cường (cụ thể là thuật toán PPO) để giải quyết bài toán tối ưu hóa tuyến đường vận chuyển (VRP) trong bối cảnh hoạt động thực tế của chuỗi bán lẻ Bách Hóa Xanh. Nghiên cứu hướng tới việc xây dựng một mô hình có khả năng học các chiến lược định tuyến hiệu quả từ dữ liệu, tự động thích ứng với các đặc điểm của mạng lưới và các ràng buộc vận hành, nhằm mục tiêu cuối cùng là giảm thiểu tổng chi phí logistics và nâng cao hiệu quả hoạt động.

1.3.2. Các Mục tiêu Cụ thể

Để đạt được mục tiêu tổng quát, nghiên cứu sẽ tập trung vào các mục tiêu cụ thể sau:

1. **Xây dựng mô hình VRP chi tiết cho Bách Hóa Xanh:** Dựa trên dữ liệu đầu vào được cung cấp (input.xlsx, Distance_matrix.xlsx) và các thông tin liên quan đến hoạt động của BHX, định nghĩa một cách hình thức bài toán CVRP bao gồm các yếu tố như đặc điểm nút (kho, cửa hàng với nhu cầu, thời gian phục vụ, chi phí/mức ưu tiên liên quan), đặc điểm đội xe (tải trọng, chi phí vận hành dựa trên VEHICLE_TYPES_CONFIG), hàm mục tiêu (tối thiểu hóa tổng chi phí), và các ràng buộc vận hành chính.
2. **Thiết kế và triển khai kiến trúc GNN-PPO phù hợp:**
    - Phát triển một bộ mã hóa GNN (ví dụ, sử dụng TransformerConv [55]) có khả năng học các vector biểu diễn (node embeddings) hiệu quả từ cấu trúc đồ thị của mạng lưới BHX và các đặc trưng của nút/cạnh.
    - Xây dựng một môi trường học tăng cường (VRPEnvGym) mô phỏng quá trình ra quyết định trong VRP, bao gồm định nghĩa không gian trạng thái (sử dụng đầu ra từ GNN và các thông tin động), không gian hành động, và hàm thưởng phản ánh mục tiêu tối ưu hóa của BHX.
    - Triển khai thuật toán PPO [37, 38] với một chính sách Actor-Critic tùy chỉnh (VRPActorCriticPolicy) sử dụng các biểu diễn trạng thái từ GNN để học chiến lược xây dựng tuyến đường.

3. **Huấn luyện và tinh chỉnh mô hình:** Thực hiện quy trình huấn luyện mô hình PPO, bao gồm việc chuẩn bị dữ liệu, thiết lập môi trường huấn luyện song song, và tinh chỉnh các siêu tham số quan trọng của GNN và PPO để đạt được hiệu suất tốt nhất.
4. **Đánh giá hiệu suất của mô hình GNN-PPO:**
    - Thực hiện suy luận trên bộ dữ liệu thử nghiệm (dựa trên dữ liệu của BHX) để tạo ra các giải pháp định tuyến.
    - Đánh giá chất lượng giải pháp dựa trên các chỉ số định lượng như tổng chi phí hoạt động (bao gồm chi phí quãng đường, chi phí thời gian, và các chi phí/phạt liên quan đến việc phục vụ), tổng quãng đường di chuyển, số lượng xe sử dụng, và mức độ sử dụng xe.
    - Đánh giá hiệu quả tính toán của mô hình (thời gian huấn luyện, thời gian suy luận).

5. **Phân tích và so sánh (nếu có thể):** Phân tích các kết quả thu được, thảo luận về những ưu điểm và hạn chế của phương pháp GNN-PPO so với các cách tiếp cận truyền thống (heuristic) trong việc giải quyết VRP. Nêu bật những hiểu biết thu được từ quá trình mô hình hóa và thực nghiệm.

1.3.3. Các Câu hỏi Nghiên cứu Chính

Nghiên cứu này sẽ cố gắng trả lời các câu hỏi nghiên cứu sau:

1. Làm thế nào để mô hình hóa bài toán VRP phức tạp của chuỗi bán lẻ Bách Hóa Xanh một cách hiệu quả dưới dạng một bài toán học tăng cường, có tính đến các yếu tố như nhiều loại xe, nhu cầu đa dạng, thời gian phục vụ, và các chi phí liên quan?
2. Kiến trúc Mạng Nơ-ron Đồ thị (GNN) nào (ví dụ, dựa trên TransformerConv) là phù hợp để học các biểu diễn trạng thái có ý nghĩa từ dữ liệu mạng lưới và các đặc trưng của cửa hàng/kho trong bối cảnh VRP của BHX?
3. Thuật toán Tối ưu hóa Chính sách Gần đúng (PPO) có thể học được một chính sách định tuyến hiệu quả dựa trên các biểu diễn trạng thái do GNN cung cấp để giải quyết bài toán CVRP cho BHX không?
4. Hiệu suất của giải pháp GNN-PPO được đề xuất như thế nào khi so sánh với các chỉ số mục tiêu (ví dụ: tổng chi phí, tổng quãng đường) và các ràng buộc của bài toán?
5. Những ưu điểm và hạn chế chính của việc sử dụng phương pháp GNN-PPO so với các phương pháp heuristic truyền thống trong việc giải quyết VRP quy mô lớn và phức tạp như của Bách Hóa Xanh là gì?
6. Hệ thống phần mềm được xây dựng (frontend và backend) có khả năng hỗ trợ quá trình ra quyết định của người quản lý và trực quan hóa kết quả một cách hiệu quả không?

Trả lời các câu hỏi này sẽ cung cấp những đóng góp quan trọng cho cả lĩnh vực học thuật (ứng dụng GNN-RL cho VRP) và thực tiễn (cung cấp một giải pháp tiềm năng cho BHX).

1.4. Phạm vi và Giới hạn của Nghiên cứu

1.4.1. Phạm vi Nghiên cứu

Nghiên cứu này sẽ tập trung vào các khía cạnh sau:

- **Bài toán Mô hình hóa:** Tập trung chính vào biến thể Bài toán Định tuyến Phương tiện Có Giới hạn Tải trọng (CVRP) với một kho xuất phát duy nhất (single-depot). Mô hình sẽ xem xét sự tồn tại của nhiều loại xe với các đặc tính khác nhau (tải trọng, chi phí vận hành, v.v.) như được định nghĩa trong VEHICLE_TYPES_CONFIG.
- **Dữ liệu Đầu vào:** Sử dụng bộ dữ liệu được cung cấp, bao gồm input.xlsx (chứa thông tin chi tiết về 1 kho và 101 cửa hàng: tọa độ, nhu cầu, thời gian phục vụ, và một trường Tiền phạt) và Distance_matrix.xlsx (ma trận khoảng cách đã được tính toán trước, có thể được tạo ra bởi OpenRouteService [49, 50, 53]).
- **Phương pháp Giải quyết:** Trọng tâm là phát triển và đánh giá một mô hình học sâu kết hợp GNN (sử dụng kiến trúc dựa trên TransformerConv [55]) để mã hóa trạng thái và PPO [37, 38, 71] (thuộc họ thuật toán Actor-Critic [41, 42]) để học chính sách định tuyến.
- **Triển khai Hệ thống:** Xây dựng một hệ thống phần mềm minh họa (proof-of-concept) bao gồm:

    - **Backend:** Sử dụng Python và FastAPI để xây dựng API, quản lý quy trình xử lý AI (bao gồm gọi các hàm từ VRP_0_Main.py để tính ma trận khoảng cách, huấn luyện mô hình, và chạy suy luận).
    - **Frontend:** Sử dụng HTML, CSS, và JavaScript (với thư viện Leaflet.js) để tạo giao diện cho người quản lý (tải dữ liệu, kích hoạt xử lý, xem kết quả) và có thể là một giao diện cơ bản cho tài xế.
- **Kết quả Đầu ra:** Các tuyến đường được tối ưu hóa cho từng xe trong đội xe, được biểu diễn dưới dạng danh sách các điểm dừng và được lưu trữ trong file data.json. Các chỉ số hiệu suất định lượng của giải pháp.

1.4.2. Các Giới hạn của Nghiên cứu

Nghiên cứu này, do giới hạn về thời gian và nguồn lực, sẽ có một số giới hạn nhất định:

- **Tính Tĩnh của Bài toán:** Mô hình chủ yếu giải quyết VRP trong môi trường tĩnh. Các yếu tố động như thay đổi giao thông theo thời gian thực, các yêu cầu dịch vụ mới phát sinh đột xuất (dynamic requests), hoặc sự cố phương tiện sẽ không được xem xét chi tiết trong phiên bản này.
- **Dữ liệu Mô phỏng/Cung cấp:** Nghiên cứu dựa trên bộ dữ liệu được cung cấp. Mặc dù dữ liệu này mô tả một kịch bản thực tế, nó có thể không phản ánh đầy đủ tất cả sự phức tạp và biến động của dữ liệu vận hành thực tế tại BHX. Việc xác thực trên quy mô lớn với dữ liệu nội bộ của BHX nằm ngoài phạm vi hiện tại.
- **Ràng buộc Thời gian Cửa sổ (Time Windows):** Mặc dù thời gian phục vụ được xem xét, các ràng buộc nghiêm ngặt về cửa sổ thời gian giao hàng (VRPTW) cho từng cửa hàng không phải là trọng tâm chính của mô hình ban đầu, mặc dù đây là một hướng mở rộng tự nhiên và quan trọng [13].
- **Tính toán Chi phí Chính xác:** Việc tính toán tổng chi phí vận hành sẽ dựa trên các giả định về chi phí đơn vị (ví dụ: chi phí nhiên liệu/km, chi phí tài xế/giờ) dựa trên thông tin công khai hoặc nghiên cứu thị trường [73, 74, 75, 76], thay vì dữ liệu chi phí nội bộ chính xác của BHX. Ý nghĩa của trường Tiền phạt cũng cần được diễn giải và tích hợp một cách cẩn thận.
- **Khả năng Giải thích của Mô hình Học sâu:** Các mô hình học sâu như GNN và PPO thường được coi là "hộp đen", việc giải thích chi tiết tại sao mô hình đưa ra một quyết định cụ thể có thể khó khăn hơn so với các thuật toán heuristic truyền thống.
- **Tối ưu hóa Siêu tham số:** Mặc dù việc tinh chỉnh siêu tham số sẽ được thực hiện, việc tìm ra bộ siêu tham số tối ưu toàn cục cho một kiến trúc phức tạp như GNN-PPO là một quá trình tốn nhiều thời gian và có thể không đạt được mức tối ưu tuyệt đối trong khuôn khổ nghiên cứu.
- **So sánh với Các phương pháp Hiện đại Khác:** Do giới hạn, việc so sánh toàn diện với tất cả các phương pháp giải VRP tiên tiến khác (ví dụ, các thuật toán metaheuristic phức tạp được tinh chỉnh kỹ lưỡng hoặc các kiến trúc học sâu khác cho VRP) có thể không thực hiện được.

Nhận thức rõ những giới hạn này sẽ giúp định hướng các nỗ lực nghiên cứu và phát triển trong tương lai.

1.5. Đóng góp của Nghiên cứu

Mặc dù có những giới hạn, nghiên cứu này dự kiến sẽ mang lại những đóng góp có giá trị:

1. **Đề xuất và Triển khai một Giải pháp VRP dựa trên GNN-PPO cho Bối cảnh Cụ thể:** Áp dụng một phương pháp học sâu tiên tiến vào bài toán VRP của một chuỗi bán lẻ lớn tại Việt Nam, có tính đến các đặc thù của dữ liệu và yêu cầu vận hành. Đây là một trong những nỗ lực tiên phong (trong phạm vi kiến thức của tác giả) ứng dụng cụ thể kiến trúc này cho BHX.
2. **Minh chứng Tiềm năng của Học tăng cường và GNN:** Cung cấp bằng chứng thực nghiệm về khả năng của GNN trong việc học biểu diễn đồ thị hiệu quả và của PPO trong việc học các chính sách định tuyến phức tạp cho VRP, vượt qua những hạn chế của các phương pháp truyền thống trong một số khía cạnh.
3. **Xây dựng một Hệ thống End-to-End:** Phát triển một hệ thống phần mềm hoàn chỉnh từ backend xử lý AI đến frontend trực quan hóa, tạo ra một công cụ có khả năng được thử nghiệm và mở rộng trong thực tế.
4. **Cung cấp Nền tảng cho Nghiên cứu Tiếp theo:** Kết quả và những bài học kinh nghiệm từ nghiên cứu này sẽ là tiền đề quan trọng cho các cải tiến và mở rộng trong tương lai, chẳng hạn như tích hợp các yếu tố động, xử lý các ràng buộc phức tạp hơn, hoặc tối ưu hóa đa mục tiêu.
5. **Gợi ý Thực tiễn cho Doanh nghiệp:** Cung cấp cho Bách Hóa Xanh và các doanh nghiệp bán lẻ khác một cái nhìn sâu sắc về tiềm năng của việc ứng dụng trí tuệ nhân tạo để giải quyết các thách thức logistics, từ đó có thể đưa ra các quyết định đầu tư và chiến lược phù hợp.

Nghiên cứu này không chỉ là một bài tập học thuật mà còn hướng đến việc tạo ra một giải pháp có khả năng ứng dụng, góp phần nâng cao hiệu quả hoạt động và năng lực cạnh tranh cho ngành bán lẻ Việt Nam trong bối cảnh chuyển đổi số mạnh mẽ.

1.6. Cấu trúc Chi tiết của Báo cáo

- **Chương 2 (Tổng quan tài liệu):** Trình bày các nghiên cứu liên quan về VRP, các phương pháp giải quyết truyền thống, ứng dụng của GNN và RL trong VRP.
- **Chương 3 (Phương pháp luận):** Mô tả chi tiết cách định nghĩa bài toán VRP cho BHX, kiến trúc mô hình GNN-PPO, thiết kế môi trường học tăng cường (trạng thái, hành động, hàm thưởng), và quy trình huấn luyện.
- **Chương 4 (Thiết kế và Triển khai Hệ thống):** Mô tả kiến trúc tổng thể của hệ thống phần mềm được phát triển, bao gồm các thành phần frontend, backend và lõi xử lý AI, cùng với luồng dữ liệu và tương tác giữa chúng. *(Phần này đã được bạn yêu cầu chi tiết và tôi đã cung cấp ở phản hồi trước, bạn có thể tích hợp vào đây hoặc giữ như một chương riêng nếu muốn nhấn mạnh khía cạnh kỹ thuật phần mềm)*.

- **Chương 5 (Thực nghiệm và Kết quả):** Trình bày thiết lập thực nghiệm, các bộ dữ liệu sử dụng, các chỉ số đánh giá và kết quả đạt được từ mô hình.
- **Chương 6 (Thảo luận):** Phân tích kết quả, so sánh với các phương pháp khác (nếu có), thảo luận về ưu nhược điểm và ý nghĩa thực tiễn của mô hình.
- **Chương 7 (Kết luận và Hướng phát triển):** Tóm tắt các đóng góp chính, nêu bật những hạn chế và đề xuất các hướng nghiên cứu tiếp theo.

CHƯƠNG 2: TỔNG QUAN TÀI LIỆU VÀ CƠ SỞ LÝ THUYẾT

Chương này trình bày tổng quan về các phương pháp giải quyết Bài toán Định tuyến Phương tiện (VRP) đã được nghiên cứu và ứng dụng, từ các thuật toán heuristic truyền thống đến các kỹ thuật học máy và học sâu hiện đại. Trọng tâm sẽ được đặt vào việc phân tích ưu nhược điểm của từng nhóm phương pháp, đồng thời làm rõ cơ sở lý thuyết của Mạng Nơ-ron Đồ thị (GNN) và Học tăng cường (RL), đặc biệt là thuật toán Proximal Policy Optimization (PPO), là những nền tảng cốt lõi của giải pháp được đề xuất trong nghiên cứu này.

2.1. Các Phương pháp Truyền thống Giải quyết Bài toán Định tuyến Phương tiện (VRP)

Trong nhiều thập kỷ, VRP đã thu hút sự quan tâm lớn từ cộng đồng nghiên cứu vận trù học. Do tính chất NP-khó, việc tìm ra giải pháp tối ưu chính xác thường chỉ khả thi đối với các bài toán có quy mô nhỏ. Đối với các bài toán thực tế với hàng trăm hoặc hàng ngàn điểm giao hàng, các phương pháp heuristic và metaheuristic đã trở thành công cụ chủ yếu.

2.1.1. Phương pháp Chính xác (Exact Methods)

Các phương pháp chính xác nhằm mục tiêu tìm ra giải pháp tối ưu toàn cục cho VRP. Một số kỹ thuật phổ biến bao gồm:

- **Lập trình Nguyên (Integer Programming - IP) và Lập trình Tuyến tính Nguyên Hỗn hợp (Mixed Integer Linear Programming - MILP):** Bài toán VRP có thể được mô hình hóa dưới dạng một bài toán IP/MILP. Các trình giải (solvers) thương mại như CPLEX, Gurobi, hoặc các trình giải mã nguồn mở có thể được sử dụng. Tuy nhiên, thời gian tính toán tăng theo hàm mũ với kích thước bài toán, khiến chúng không khả thi cho các instance lớn.
- **Phương pháp Nhánh và Cận (Branch and Bound):** Một kỹ thuật tìm kiếm có hệ thống, chia không gian giải pháp thành các bài toán con nhỏ hơn và sử dụng các giới hạn (bounds) để loại bỏ các nhánh không chứa giải pháp tối ưu.
- **Phương pháp Nhánh và Cắt (Branch and Cut):** Mở rộng của Branch and Bound, bổ sung thêm các ràng buộc "cắt" (cutting planes) hợp lệ để thắt chặt miền đa diện của bài toán Lập trình Tuyến tính nới lỏng (LP relaxation), từ đó cải thiện giới hạn và tăng tốc độ tìm kiếm.

- **Lập trình Động (Dynamic Programming):** Có thể áp dụng cho một số biến thể VRP nhất định, đặc biệt là các bài toán có cấu trúc tuần tự. Tuy nhiên, "lời nguyền về chiều" (curse of dimensionality) hạn chế khả năng ứng dụng cho các bài toán lớn.

Mặc dù các phương pháp chính xác đảm bảo tính tối ưu, chúng thường chỉ phù hợp cho các bài toán có kích thước nhỏ (ví dụ, dưới 50-100 khách hàng, tùy thuộc vào cấu trúc bài toán và hiệu quả của thuật toán) do yêu cầu tính toán quá lớn [Toth & Vigo, 2014].

2.1.2. Phương pháp Heuristic (Heuristic Methods)

Các phương pháp heuristic được thiết kế để tìm ra các giải pháp "đủ tốt" trong một khoảng thời gian tính toán hợp lý, hy sinh tính tối ưu toàn cục để đổi lấy tốc độ và khả năng giải quyết các bài toán quy mô lớn. Chúng thường dựa trên các quy tắc trực quan hoặc kinh nghiệm.

- **Thuật toán Xây dựng Tuyến đường (Constructive Heuristics):** Xây dựng các tuyến đường một cách tuần tự từ đầu.
    - **Thuật toán Tiết kiệm (Clarke & Wright Savings Algorithm, 1964):** Một trong những heuristic cổ điển và được sử dụng rộng rãi nhất. Ban đầu, mỗi khách hàng được phục vụ bởi một tuyến đường riêng từ kho. Sau đó, thuật toán lặp đi lặp lại việc hợp nhất các tuyến đường dựa trên "tiết kiệm" chi phí (thường là quãng đường) đạt được khi kết nối hai khách hàng vào cùng một tuyến thay vì phục vụ họ riêng biệt.
    - **Thuật toán Quét (Sweep Algorithm):** Phù hợp cho các bài toán mà các điểm khách hàng có thể được sắp xếp theo góc cực quanh kho. Thuật toán "quét" một tia từ kho, gom các khách hàng vào các tuyến đường cho đến khi ràng buộc tải trọng hoặc các ràng buộc khác bị vi phạm, sau đó bắt đầu một tuyến đường mới.
    - **Thuật toán Chèn (Insertion Heuristics - ví dụ: Nearest Insertion, Farthest Insertion, Cheapest Insertion):** Bắt đầu với một tuyến đường rỗng hoặc một tuyến đường nhỏ, sau đó lặp đi lặp lại việc chọn một khách hàng chưa được phục vụ và chèn vào vị trí "tốt nhất" (theo một tiêu chí nào đó) trong một tuyến đường hiện có.
- **Thuật toán Cải thiện Tuyến đường (Improvement Heuristics / Local Search):** Bắt đầu từ một giải pháp khả thi ban đầu (có thể được tạo bởi một heuristic xây dựng), sau đó cố gắng cải thiện nó bằng cách thực hiện các thay đổi cục bộ (local moves).
    - **2-opt, 3-opt, k-opt:** Loại bỏ k cạnh khỏi các tuyến đường hiện tại và kết nối lại các điểm theo một cách khác để tạo ra một tập hợp tuyến đường mới, hy vọng giảm tổng chi phí. 2-opt là trường hợp đơn giản nhất, loại bỏ hai cạnh không giao nhau và đảo ngược đoạn đường giữa chúng.
    - **Or-opt:** Di chuyển một chuỗi các khách hàng từ vị trí này sang vị trí khác trong cùng một tuyến đường hoặc sang một tuyến đường khác.

    - **Trao đổi giữa các tuyến (Inter-route exchanges):** Di chuyển hoặc trao đổi các khách hàng giữa các tuyến đường khác nhau.

Các phương pháp heuristic thường nhanh chóng nhưng dễ bị mắc kẹt ở các điểm cực tiểu địa phương và chất lượng giải pháp có thể không cao.

2.1.3. Phương pháp Metaheuristic (Metaheuristic Methods)

Metaheuristic là các chiến lược tìm kiếm bậc cao hơn, hướng dẫn các heuristic cấp thấp để khám phá không gian giải pháp một cách hiệu quả hơn và thoát khỏi các điểm cực tiểu địa phương.

- **Luyện kim Mô phỏng (Simulated Annealing - SA):** Lấy cảm hứng từ quá trình luyện kim trong vật lý, SA cho phép các bước đi làm tăng chi phí (xấu hơn) với một xác suất nhất định, xác suất này giảm dần theo thời gian ("nhiệt độ" giảm). Điều này giúp thuật toán thoát khỏi các cực tiểu địa phương.
- **Tìm kiếm Cấm (Tabu Search - TS):** Duy trì một "danh sách cấm" (tabu list) các bước đi đã thực hiện gần đây hoặc các thuộc tính của giải pháp để tránh quay lại các giải pháp đã xét và khuyến khích khám phá các vùng mới của không gian tìm kiếm.
- **Thuật toán Di truyền (Genetic Algorithms - GA):** Mô phỏng quá trình tiến hóa tự nhiên. Một quần thể các giải pháp (nhiễm sắc thể) được duy trì. Các toán tử như lựa chọn (selection), lai ghép (crossover), và đột biến (mutation) được sử dụng để tạo ra các thế hệ giải pháp mới, hy vọng hội tụ về các giải pháp tốt hơn.
- **Tìm kiếm Vùng Lân cận Lớn Thích ứng (Adaptive Large Neighborhood Search - ALNS):** Một metaheuristic mạnh mẽ, đặc biệt hiệu quả cho VRP. ALNS hoạt động bằng cách lặp đi lặp lại việc "phá hủy" (destroy) một phần của giải pháp hiện tại (loại bỏ một số khách hàng khỏi các tuyến đường) và sau đó "sửa chữa" (repair) nó bằng cách chèn lại các khách hàng đó vào các vị trí mới, sử dụng một tập hợp các toán tử phá hủy và sửa chữa đa dạng. Việc lựa chọn các toán tử này có thể được điều chỉnh một cách thích ứng trong quá trình tìm kiếm [ALNS được giới thiệu bởi Ropke và Pisinger, 2006 - *Nguồn gốc ALNS*].
- **Thuật toán Kiến (Ant Colony Optimization - ACO):** Lấy cảm hứng từ hành vi tìm kiếm thức ăn của đàn kiến, sử dụng "pheromone" nhân tạo để đánh dấu các đường đi tốt.

Metaheuristic thường mang lại các giải pháp có chất lượng cao hơn so với các heuristic đơn giản, nhưng chúng đòi hỏi nhiều thời gian tính toán hơn và việc tinh chỉnh các tham số của chúng có thể phức tạp. Dù vậy, chúng vẫn là một trong những cách tiếp cận phổ biến và hiệu quả nhất cho VRP quy mô lớn trong thực tế [Gendreau et al., 2008 - *Sách hoặc bài báo tổng quan về metaheuristics cho VRP*].

2.1.4. Ưu điểm và Nhược điểm của các Phương pháp Truyền thống

- **Ưu điểm:**

    - **Dễ hiểu và triển khai (đối với các heuristic đơn giản):** Nhiều thuật toán heuristic xây dựng có logic trực quan.
    - **Thời gian tính toán nhanh (đối với các heuristic đơn giản):** Có thể cung cấp giải pháp nhanh chóng cho các bài toán lớn.
    - **Đã được nghiên cứu và ứng dụng rộng rãi:** Có nhiều tài liệu và kinh nghiệm triển khai.
    - **Metaheuristic có thể cho kết quả tốt:** Các metaheuristic như ALNS, GA, TS thường tìm được các giải pháp chất lượng cao gần tối ưu.
- **Nhược điểm:**
    - **Không đảm bảo tính tối ưu toàn cục:** Heuristic và metaheuristic thường chỉ tìm được giải pháp cực tiểu địa phương.
    - **Độ nhạy với tham số:** Hiệu suất của metaheuristic phụ thuộc nhiều vào việc lựa chọn và tinh chỉnh các tham số.
    - **Khó thích ứng với ràng buộc mới hoặc thay đổi động:** Việc sửa đổi các thuật toán truyền thống để xử lý các ràng buộc phức tạp hoặc các yếu tố động của bài toán có thể rất khó khăn.
    - **Ít khả năng học từ dữ liệu:** Các phương pháp này thường không "học" từ các giải pháp trước đó hoặc từ các đặc điểm tiềm ẩn của dữ liệu để cải thiện hiệu suất theo thời gian. Chúng giải quyết mỗi instance một cách độc lập.

2.2. Học Máy và Học Sâu trong Giải quyết VRP

Sự phát triển mạnh mẽ của học máy (Machine Learning - ML) và học sâu (Deep Learning - DL) trong những năm gần đây đã mở ra những hướng tiếp cận mới đầy hứa hẹn cho các bài toán tối ưu hóa tổ hợp, bao gồm cả VRP. Thay vì dựa vào các quy tắc được thiết kế thủ công, các phương pháp này cố gắng "học" cách giải quyết bài toán từ dữ liệu.

2.2.1. Giới thiệu về Mạng Nơ-ron Đồ thị (Graph Neural Networks - GNN)

Mạng Nơ-ron Đồ thị (GNN) là một lớp các mô hình học sâu được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc đồ thị [Wu et al., 2020; Zhou et al., 2020 - *Các bài báo tổng quan về GNN*]. Đồ thị, bao gồm các nút (nodes) và các cạnh (edges) nối giữa chúng, là một cấu trúc dữ liệu tự nhiên để biểu diễn nhiều loại thông tin trong thế giới thực, từ mạng xã hội, mạng lưới phân tử, đến các hệ thống logistics và chuỗi cung ứng.

2.2.1.1. Khái niệm và Nguyên lý Hoạt động Cơ bản

Nguyên lý cốt lõi của GNN là học các vector biểu diễn (embeddings) cho các nút (và đôi khi là các cạnh hoặc toàn bộ đồ thị) bằng cách tổng hợp thông tin từ các nút lân cận của chúng. Quá trình này thường diễn ra qua nhiều lớp GNN. Trong mỗi lớp, vector biểu diễn của một nút được cập nhật dựa trên vector biểu diễn hiện tại của chính nó và các vector biểu diễn của các nút hàng xóm (và có thể cả các đặc trưng của cạnh nối chúng).

Một cách tổng quát, phép toán cập nhật tại một lớp GNN có thể được mô tả bởi hai hàm chính:

1. **Hàm Tổng hợp (AGGREGATE):** Thu thập thông tin từ các nút lân cận.
   trong đó h_u^{(k-1)} là vector biểu diễn của nút lân cận u từ lớp trước đó (k-1), và N(v) là tập hợp các nút lân cận của nút v. Hàm AGGREGATE có thể là sum, mean, max, hoặc các hàm phức tạp hơn.
2. **Hàm Kết hợp (COMBINE/UPDATE):** Kết hợp thông tin tổng hợp từ hàng xóm với vector biểu diễn hiện tại của nút để tạo ra vector biểu diễn mới cho lớp hiện tại.

   Hàm COMBINE thường là một mạng nơ-ron truyền thẳng (feed-forward neural network).

Quá trình này được lặp lại qua nhiều lớp, cho phép thông tin lan truyền xa hơn trên đồ thị, giúp mỗi nút học được một biểu diễn có tính đến ngữ cảnh cấu trúc rộng hơn.

2.2.1.2. Các Kiến trúc GNN Phổ biến và `TransformerConv`

Có nhiều kiến trúc GNN khác nhau đã được đề xuất, mỗi kiến trúc có cách tiếp cận riêng trong việc tổng hợp và kết hợp thông tin:

- **Graph Convolutional Network (GCN) [Kipf & Welling, 2017]:** Một trong những kiến trúc GNN đầu tiên và phổ biến nhất, sử dụng phép tích chập phổ hoặc một phiên bản xấp xỉ không gian của nó. GCN thường tổng hợp thông tin từ hàng xóm bằng cách lấy trung bình có trọng số.
- **Graph Attention Network (GAT) [Veličković et al., 2018]:** Giới thiệu cơ chế "attention" vào GNN, cho phép mô hình học các trọng số khác nhau cho các nút lân cận khác nhau khi tổng hợp thông tin, thay vì gán trọng số bằng nhau hoặc cố định. Điều này giúp mô hình tập trung vào những hàng xóm quan trọng hơn.
- **GraphSAGE [Hamilton et al., 2017]:** Đề xuất các hàm tổng hợp có thể học được (ví dụ: mean, LSTM, pooling) và tập trung vào việc học các hàm nhúng có khả năng tổng quát hóa cho các nút chưa từng thấy.
- **Message Passing Neural Network (MPNN) [Gilmer et al., 2017]:** Cung cấp một khung tổng quát cho nhiều kiến trúc GNN, định nghĩa quá trình học dưới dạng "truyền thông điệp" (message passing) giữa các nút.

Trong nghiên cứu này, kiến trúc **TransformerConv** [Shi et al., 2020 - *Bài báo gốc về TransformerConv nếu có, hoặc tài liệu từ PyTorch Geometric [55, 56]*] từ thư viện PyTorch Geometric được lựa chọn. Lớp TransformerConv lấy cảm hứng từ kiến trúc Transformer nổi tiếng trong xử lý ngôn ngữ tự nhiên, áp dụng cơ chế self-attention cho các nút trong đồ thị. Một ưu điểm quan trọng của TransformerConv là khả năng tích hợp một cách tự nhiên các **đặc trưng của cạnh (edge features)** vào quá trình tính toán attention. Điều này đặc biệt phù hợp với VRP, nơi các thuộc tính của tuyến đường (như khoảng cách, thời gian di chuyển, chi phí) là rất quan trọng. Công thức cập nhật của TransformerConv có thể cho phép các đặc trưng cạnh ảnh hưởng đến trọng số attention giữa các cặp nút, từ đó tạo ra các node embeddings có tính đến cả đặc trưng nút và đặc trưng cạnh

một cách hiệu quả. Trong dự án này, edge_attr bao gồm khoảng cách, thời gian di chuyển và chi phí nhiên liệu dự kiến cho từng loại xe, là những thông tin then chốt cho việc ra quyết định định tuyến.

2.2.1.3. Ứng dụng của GNN trong các Bài toán trên Đồ thị và VRP

GNN đã chứng minh được hiệu quả vượt trội trong nhiều loại bài toán trên đồ thị, bao gồm:

- Phân loại nút (node classification).
- Phân loại đồ thị (graph classification).
- Dự đoán liên kết (link prediction).
- Các bài toán tối ưu hóa tổ hợp trên đồ thị.

Đối với VRP, GNN có thể được sử dụng để:

- **Học biểu diễn trạng thái:** Mã hóa trạng thái hiện tại của bài toán VRP (bao gồm cấu trúc mạng lưới, nhu cầu khách hàng, vị trí xe, tải trọng còn lại) thành các vector embeddings giàu thông tin. Các embeddings này sau đó có thể được sử dụng bởi một thuật toán học tăng cường hoặc một heuristic được học [Kool et al., 2019; Nazari et al., 2018].
- **Dự đoán trực tiếp giải pháp:** Một số nghiên cứu cố gắng huấn luyện GNN để dự đoán trực tiếp ma trận kề của giải pháp VRP hoặc một chuỗi các hành động.
- **Hỗ trợ các thuật toán khác:** Ví dụ, GNN có thể được dùng để dự đoán các vùng "hứa hẹn" trong không gian tìm kiếm cho các thuật toán metaheuristic.

Nghiên cứu này sử dụng GNN trong vai trò học biểu diễn trạng thái cho tác tử học tăng cường PPO.

2.2.2. Giới thiệu về Học tăng cường (Reinforcement Learning - RL)

Học tăng cường (Reinforcement Learning - RL) là một nhánh của học máy, trong đó một **tác tử (agent)** học cách đưa ra một chuỗi các **hành động (actions)** trong một **môi trường (environment)** để tối đa hóa một tín hiệu **phần thưởng (reward)** tích lũy theo thời gian [Sutton & Barto, 2018 - *Sách kinh điển về RL*]. RL khác với học có giám sát (supervised learning) ở chỗ tác tử không được cung cấp các cặp (đầu vào, đầu ra đúng) mà phải tự khám phá hành vi tối ưu thông qua tương tác thử và sai.

2.2.2.1. Các Khái niệm Cơ bản trong RL

- **Tác tử (Agent):** Thực thể học và ra quyết định (ví dụ: điều phối viên VRP, xe tự hành).
- **Môi trường (Environment):** Thế giới bên ngoài mà tác tử tương tác (ví dụ: bài toán VRP với các cửa hàng, kho, đội xe).
- **Trạng thái (State,** Một mô tả về tình hình hiện tại của môi trường mà tác tử quan sát được.
- **Hành động (Action,** Lựa chọn mà tác tử có thể thực hiện trong một trạng thái nhất định.

- **Chính sách (Policy,** Một hàm ánh xạ từ trạng thái sang hành động (hoặc phân phối xác suất trên các hành động). Chính sách xác định hành vi của tác tử. $\pi(a|s)$ là xác suất chọn hành động a khi ở trạng thái s.
- **Hàm thưởng (Reward Function,** Một tín hiệu vô hướng mà tác tử nhận được từ môi trường sau khi thực hiện một hành động. Phần thưởng cho biết hành động đó "tốt" hay "xấu" như thế nào trong ngắn hạn.
- **Hàm Giá trị (Value Function):**
    - **State-Value Function (** Tổng phần thưởng kỳ vọng tích lũy khi bắt đầu từ trạng thái s và tuân theo chính sách π.
    - **Action-Value Function (** Tổng phần thưởng kỳ vọng tích lũy khi bắt đầu từ trạng thái s, thực hiện hành động a, và sau đó tuân theo chính sách π.
- **Mô hình (Model - tùy chọn):** Một mô tả về động lực của môi trường, dự đoán trạng thái tiếp theo và phần thưởng dựa trên trạng thái hiện tại và hành động. Các phương pháp RL không cần mô hình (model-free) như PPO sẽ được sử dụng trong nghiên cứu này.
- **Lượt chơi (Episode):** Một chuỗi các tương tác (trạng thái, hành động, phần thưởng) từ trạng thái bắt đầu đến trạng thái kết thúc.

Mục tiêu của RL là tìm ra một chính sách tối ưu π^* sao cho tổng phần thưởng kỳ vọng tích lũy (còn gọi là lợi nhuận - return) là lớn nhất.

2.2.2.2. Kiến trúc Tác tử - Phê bình (Actor-Critic Architectures)

Các phương pháp Actor-Critic [41, 42] là một lớp các thuật toán RL phổ biến, kết hợp ưu điểm của cả phương pháp dựa trên chính sách (policy-based) và phương pháp dựa trên hàm giá trị (value-based). Chúng bao gồm hai thành phần chính:

- **Tác tử (Actor):** Chịu trách nhiệm học và thực thi chính sách $\pi(a|s)$. Nó quyết định hành động nào sẽ được thực hiện trong một trạng thái nhất định.
- **Phê bình (Critic):** Chịu trách nhiệm học hàm giá trị (thường là V(s) hoặc Q(s,a)). Nó đánh giá "chất lượng" của hành động được chọn bởi Actor hoặc giá trị của trạng thái hiện tại.

Actor và Critic làm việc cùng nhau: Actor chọn hành động, Critic đánh giá hành động đó (hoặc trạng thái kết quả). Thông tin đánh giá từ Critic sau đó được sử dụng để cập nhật Actor, giúp Actor cải thiện chính sách của mình. Kiến trúc này thường dẫn đến việc học ổn định và hiệu quả hơn so với các phương pháp chỉ dựa vào Actor hoặc chỉ dựa vào Critic.

2.2.2.3. Thuật toán Tối ưu hóa Chính sách Gần đúng (Proximal Policy Optimization - PPO)

Proximal Policy Optimization (PPO) [Schulman et al., 2017 - Bài báo gốc về PPO; 37, 38] là một thuật toán RL thuộc họ Actor-Critic, on-policy (học từ kinh nghiệm được tạo ra bởi chính sách hiện tại). PPO nổi tiếng với sự cân bằng tốt giữa hiệu suất, độ phức tạp triển khai và tính ổn định, khiến nó trở thành một lựa chọn phổ biến cho nhiều bài toán điều khiển phức tạp.

Các đặc điểm chính của PPO:

- **Hàm mục tiêu Thay thế Bị cắt bớt (Clipped Surrogate Objective Function):** Thay vì tối ưu hóa trực tiếp lợi nhuận kỳ vọng, PPO tối ưu hóa một hàm mục tiêu "thay thế" (surrogate) được xây dựng dựa trên tỷ lệ xác suất giữa chính sách mới và chính sách cũ. Quan trọng hơn, PPO sử dụng một cơ chế "cắt" (clipping) để giới hạn sự thay đổi của chính sách trong mỗi lần cập nhật.
  Việc cắt này ngăn chặn các cập nhật chính sách quá lớn, giúp quá trình huấn luyện ổn định hơn và tránh làm giảm hiệu suất đột ngột.
- **Sử dụng Lợi thế (Advantage Estimation):** PPO sử dụng hàm lợi thế $A(s,a) = Q(s,a) - V(s)$ để đánh giá mức độ tốt hơn (hoặc tệ hơn) của một hành động cụ thể so với hành động trung bình từ trạng thái đó. Ước tính Lợi thế Tổng quát hóa (Generalized Advantage Estimation - GAE) [Schulman et al., 2016 - *Bài báo về GAE*] thường được sử dụng để giảm phương sai của ước tính lợi thế.
- **Cập nhật nhiều Epoch trên cùng một Batch Dữ liệu:** PPO thường thực hiện nhiều epoch tối ưu hóa trên cùng một batch dữ liệu thu thập được, giúp cải thiện hiệu quả sử dụng mẫu.
- **Kết hợp Mất mát Hàm Giá trị và Entropy (Tùy chọn):** Hàm mất mát tổng thể của PPO thường bao gồm cả mất mát của hàm giá trị (để huấn luyện Critic) và một thành phần entropy của chính sách (để khuyến khích khám phá).

Sự ổn định và hiệu quả của PPO làm cho nó trở thành một lựa chọn hấp dẫn cho các bài toán VRP phức tạp, nơi không gian trạng thái và hành động có thể lớn. Trong nghiên cứu này, PPO được chọn để học chính sách xây dựng tuyến đường, nhận đầu vào là các biểu diễn trạng thái từ GNN.

2.2.3. Kết hợp GNN và RL (GNN-RL) cho VRP

Sự kết hợp giữa GNN và RL (thường được gọi là Graph Reinforcement Learning - GRL, hoặc áp dụng GNN trong một framework RL) đã nổi lên như một hướng tiếp cận mạnh mẽ cho các bài toán tối ưu hóa tổ hợp trên đồ thị, bao gồm cả VRP [17, 18, 20, 23, 24, 32, 34, 35, 36, 46, 47].

- **Vai trò của GNN:** GNN đóng vai trò là một bộ mã hóa trạng thái (state encoder) mạnh mẽ. Nó có khả năng:
  - Xử lý cấu trúc đồ thị phức tạp của bài toán VRP một cách tự nhiên.
  - Tích hợp các đặc trưng đa dạng của nút (cửa hàng, kho) và cạnh (tuyến đường) vào các vector biểu diễn (embeddings) giàu thông tin.
  - Nắm bắt các mối quan hệ không gian và sự phụ thuộc giữa các thực thể trong bài toán.

    - Cung cấp một biểu diễn trạng thái động, có khả năng cập nhật khi bài toán diễn ra (ví dụ, các nút đã được thăm, tải trọng xe thay đổi).
- **Vai trò của RL (ví dụ: PPO):** RL đóng vai trò là bộ ra quyết định (decision maker) hoặc bộ học chính sách (policy learner). Nó:
    - Học một chính sách $\pi(a|s)$ để chọn hành động (ví dụ: nút tiếp theo để thăm) dựa trên biểu diễn trạng thái được cung cấp bởi GNN.
    - Sử dụng tín hiệu phần thưởng từ môi trường để cải thiện chính sách theo thời gian, hướng tới mục tiêu tối ưu hóa (ví dụ: giảm tổng chi phí).
    - Có khả năng xử lý các quy trình ra quyết định tuần tự và các phụ thuộc dài hạn trong việc xây dựng tuyến đường.

Ưu điểm của việc kết hợp GNN-RL cho VRP:

- **Học Biểu diễn End-to-End (Tiềm năng):** Trong một số kiến trúc, GNN và tác tử RL có thể được huấn luyện end-to-end, cho phép GNN học các biểu diễn trạng thái được tối ưu hóa cụ thể cho nhiệm vụ ra quyết định của RL.
- **Khả năng Tổng quát hóa Tốt hơn:** Các mô hình GNN-RL, sau khi được huấn luyện trên một tập hợp các instance VRP đa dạng, có thể có khả năng tổng quát hóa tốt hơn cho các instance mới chưa từng thấy so với các heuristic truyền thống (vốn thường được thiết kế cho các kịch bản cụ thể).
- **Xử lý Tính Phức tạp Cao:** Sự kết hợp này có tiềm năng xử lý các biến thể VRP phức tạp với nhiều ràng buộc và mục tiêu đa dạng.
- **Ít phụ thuộc vào Kỹ thuật Đặc trưng Thủ công:** GNN có thể tự động học các đặc trưng quan trọng từ dữ liệu đồ thị, giảm bớt nhu cầu thiết kế đặc trưng thủ công tốn nhiều công sức.

Các nghiên cứu gần đây đã cho thấy kết quả đầy hứa hẹn khi áp dụng GNN-RL cho VRP. Ví dụ, Kool et al. (2019) [2] đã sử dụng một kiến trúc dựa trên Attention (tương tự Transformer) để giải quyết VRP và các bài toán định tuyến khác. Nazari et al. (2018) [3] là một trong những công trình sớm sử dụng RL cho VRP. Nhiều biến thể và cải tiến sau đó đã được đề xuất, tập trung vào việc cải thiện kiến trúc GNN, thiết kế hàm thưởng, và các kỹ thuật huấn luyện RL [ví dụ: 23, 24, 28, 30, 46, 47].

Nghiên cứu này xây dựng dựa trên những tiến bộ đó, đề xuất một kiến trúc GNN (TransformerConv) kết hợp với PPO để giải quyết bài toán CVRP cụ thể cho Bách Hóa Xanh, tận dụng khả năng biểu diễn của GNN và sự ổn định của PPO.

CHƯƠNG 3: PHƯƠNG PHÁP LUẬN NGHIÊN CỨU

Chương này trình bày chi tiết phương pháp luận được sử dụng để giải quyết Bài toán Định tuyến Phương tiện (VRP) cho chuỗi bán lẻ Bách Hóa Xanh (BHX). Phương pháp tiếp cận cốt lõi dựa trên việc mô hình hóa bài toán VRP dưới dạng một bài toán Học tăng cường (Reinforcement Learning - RL) và triển khai một giải pháp

kết hợp Mạng Nơ-ron Đồ thị (Graph Neural Network - GNN) với thuật toán Tối ưu hóa Chính sách Gần đúng (Proximal Policy Optimization - PPO).

3.1. Định nghĩa và Mô hình hóa Bài toán VRP cho Bách Hóa Xanh

Dựa trên dữ liệu đầu vào được cung cấp (bao gồm input.xlsx chứa thông tin về kho, các cửa hàng cần phục vụ và Distance_matrix.xlsx chứa ma trận khoảng cách) và các đặc thù hoạt động của BHX, bài toán VRP được mô hình hóa cụ thể như sau:

3.1.1. Các Thành phần Cơ bản của Bài toán

- **Tập hợp Nút (Nodes,** Bao gồm một kho xuất phát duy nhất (v_0) và một tập hợp N cửa hàng khách hàng (v_1, v_2, ..., v_N). Trong bộ dữ liệu được cung cấp, có 1 kho và 101 cửa hàng, do đó tổng số nút là 102.
    - **Kho (**
        - Tọa độ: (Kinh độ: 106.6022426, Vĩ độ: 10.8164641) (trích từ dòng đầu tiên của input.xlsx).
        - Nhu cầu (demand_0): 0.
        - Thời gian phục vụ (service_time_0): 0.
        - Chi phí/Tiền phạt (penalty_0): 0.
    - **Cửa hàng Khách hàng (** Mỗi cửa hàng i được đặc trưng bởi:
        - Tọa độ (coords_i): (Kinh độ, Vĩ độ) trích từ input.xlsx.
        - Nhu cầu (demand_i): Lượng hàng hóa yêu cầu, trích từ cột "Nhu cầu" trong input.xlsx.
        - Thời gian phục vụ (service_time_i): Thời gian cần thiết để dỡ hàng và hoàn tất các thủ tục tại cửa hàng, trích từ cột "Phục vụ" trong input.xlsx. Đơn vị giả định là phút.
        - Chi phí/Mức ưu tiên phục vụ (penalty_i): Một giá trị số trích từ cột "Tiền phạt" trong input.xlsx. Ý nghĩa của trường này sẽ được thảo luận trong hàm mục tiêu và hàm thưởng. Với các giá trị lớn được cung cấp, nó có thể biểu thị mức độ ưu tiên cao hoặc một chi phí cố định đáng kể liên quan đến việc phục vụ cửa hàng đó.
- **Đội xe (Vehicles,** Một tập hợp các phương tiện vận chuyển K = {k_1, k_2, ..., k_M}. Dựa trên cấu hình VEHICLE_TYPES_CONFIG trong mã nguồn dự án (VRP_0_Main.py), hệ thống xem xét sự tồn tại của nhiều loại xe, mỗi loại j có các thuộc tính:
    - Số lượng xe (count_j).
    - Tải trọng tối đa (capacity_j).
    - Vận tốc trung bình danh nghĩa (avg_velocity_j).
    - Chi phí nhiên liệu trên 100km (fuel_cost_100km_j).
    - Lương tài xế (hoặc chi phí cố định cho một chuyến xe) (driver_salary_j).

    - Hệ số giảm tốc độ do tải trọng (load_speed_reduction_factor_j): Mô tả mức độ giảm tốc độ của xe khi chở nặng.
    - Thời gian hoạt động tối đa cho một chuyến (max_op_time_minutes_j).
      Tổng số xe M là tổng của count_j qua tất cả các loại xe.
- **Ma trận Khoảng cách (** Một ma trận D kích thước (N+1) x (N+1), trong đó d_ij là khoảng cách di chuyển từ nút i đến nút j. Ma trận này được cung cấp bởi Distance_matrix.xlsx và có thể được tính toán trước bằng OpenRouteService (ORS) [49, 50, 53] như trong hàm calculate_distance_matrix. Ma trận này có thể không đối xứng (d_ij ≠ d_ji), phản ánh điều kiện giao thông thực tế (ví dụ: đường một chiều).
- **Ma trận Thời gian Di chuyển Cơ bản (** Tương tự ma trận khoảng cách, một ma trận thời gian di chuyển cơ bản t_base_ij^(type) có thể được suy ra từ khoảng cách và vận tốc trung bình danh nghĩa của từng loại xe, trước khi tính đến ảnh hưởng của tải trọng.
  t_base_ij^(type) = d_ij / avg_velocity_j^(type)
  (Trong mã nguồn của bạn, VRP_1_Input.py và GNNFeature trong VRP_2_Model.py đã tính toán các đặc trưng cạnh bao gồm thời gian di chuyển cho từng loại xe, có tính đến nhiễu vận tốc.)

3.1.2. Biến thể Bài toán: Capacitated Vehicle Routing Problem (CVRP)

Bài toán được giải quyết trong nghiên cứu này chủ yếu thuộc biến thể CVRP [3, 8], với các đặc điểm mở rộng để phù hợp với bối cảnh BHX:

- Mỗi cửa hàng có một nhu cầu cụ thể phải được đáp ứng.
- Mỗi xe có một tải trọng tối đa.
- Mỗi xe bắt đầu và kết thúc tuyến đường tại kho.
- Mỗi cửa hàng (nếu được phục vụ) chỉ được phục vụ bởi một xe duy nhất và một lần duy nhất.
- Hệ thống xem xét nhiều loại xe với các đặc tính khác nhau.
- Hệ thống xem xét thời gian hoạt động tối đa cho mỗi xe.

3.1.3. Hàm Mục tiêu (Objective Function)

Mục tiêu chính của bài toán là **tối thiểu hóa tổng chi phí vận hành**. Tổng chi phí này là một hàm tổng hợp của các thành phần sau:

1. **Tổng Chi phí Nhiên liệu:**
   Cost_Fuel = Σ_{k∈K} Σ_{(i,j)∈Route_k} (d_ij * fuel_cost_per_km_k)
   Trong đó Route_k là tập hợp các cạnh trong tuyến đường của xe k, d_ij là khoảng cách cạnh, và fuel_cost_per_km_k là chi phí nhiên liệu trên mỗi km cho loại xe của xe k (suy ra từ fuel_cost_100km_j).
2. **Tổng Chi phí Lương Tài xế (hoặc Chi phí Cố định cho Xe):**
   Cost_Driver = Σ_{k∈UsedVehicles} driver_salary_k

Trong đó UsedVehicles là tập hợp các xe được sử dụng, và driver_salary_k là lương hoặc chi phí cố định cho loại xe của xe k.

3. **Tổng "Tiền Phạt" liên quan đến việc Phục vụ Cửa hàng:**
   Cost_Penalty_Service = Σ_{i∈VisitedCustomers} penalty_i
   Giả định rằng penalty_i trong input.xlsx là một chi phí phát sinh khi phục vụ khách hàng i (ví dụ: chi phí quản lý, xử lý đặc biệt, hoặc một thước đo mức độ ưu tiên/giá trị của việc phục vụ khách hàng đó mà hệ thống cần cân nhắc). Nếu mục tiêu là phục vụ tất cả khách hàng, thì thành phần này có thể coi là một hằng số nếu tất cả được phục vụ. Tuy nhiên, trong bối cảnh tối ưu hóa, nó có thể ảnh hưởng đến việc lựa chọn thứ tự hoặc cách nhóm khách hàng.
4. **Tổng Chi phí Phạt do Không Phục vụ Cửa hàng (nếu áp dụng):**
   Nếu mô hình cho phép bỏ qua một số khách hàng (ví dụ, trong các kịch bản VRP có Lợi nhuận - VRPP), thì sẽ có một thành phần chi phí:
   Cost_Penalty_Unvisited = Σ_{i∈UnvisitedCustomers} unvisited_penalty_i'
   Trong đó unvisited_penalty_i' là một hình phạt rất lớn được định nghĩa riêng để đảm bảo rằng việc không phục vụ khách hàng là không mong muốn, trừ khi thực sự không thể. Đối với mô hình CVRP cơ bản nhắm đến phục vụ tất cả khách hàng, thành phần này thường không xuất hiện hoặc unvisited_penalty_i' được đặt là vô cùng lớn. Dựa trên mã nguồn VRP_3_Env.py (hàm calculate_total_cost), có vẻ như hệ thống của bạn tính total_penalty += self.penalty_per_unvisited_node[i] cho các nút chưa được thăm. Điều này ngụ ý rằng Tiền phạt trong input.xlsx được sử dụng như một hình phạt nếu nút đó *không* được phục vụ. Đây là một điểm quan trọng cần làm rõ trong diễn giải. Nếu Tiền phạt là hình phạt cho việc không phục vụ, thì mục tiêu sẽ là giảm thiểu (Chi phí Nhiên liệu + Chi phí Lương + Tổng Tiền phạt của các nút *không* được phục vụ).

Giả định diễn giải Tiền phạt từ input.xlsx được hiểu là **chi phí phạt nếu một cửa hàng** .
Vậy, hàm mục tiêu tổng quát là:
Minimize Z = Cost_Fuel + Cost_Driver + Cost_Penalty_Unvisited

Mục tiêu ngầm định là sử dụng ít xe nhất có thể, điều này thường đạt được khi tối đa hóa việc sử dụng mỗi xe.

3.1.4. Các Ràng buộc Chính (Constraints)

1. **Ràng buộc Phục vụ Khách hàng:** Tất cả các cửa hàng có nhu cầu (trong input.xlsx) phải được phục vụ, trừ khi chi phí phạt cho việc không phục vụ (penalty_i) được tính vào hàm mục tiêu và việc bỏ qua một số khách hàng là chấp nhận được để giảm tổng chi phí. (Dựa trên mã nguồn, có vẻ hệ thống cố gắng phục vụ tất cả, và phạt nếu không).
2. **Xuất phát và Kết thúc tại Kho:** Mỗi tuyến đường của xe phải bắt đầu từ kho (v_0) và kết thúc tại kho (v_0).

3. **Ràng buộc Tải trọng Xe (Capacity Constraint):** Đối với mỗi xe k và tuyến đường Route_k của nó, tổng nhu cầu của các khách hàng được phục vụ trên tuyến đường đó không được vượt quá tải trọng capacity_k của xe.
   $\Sigma_{i \in Route\_k \cap Customers} demand\_i \leq capacity\_k, \forall k \in K$
4. **Ràng buộc Thời gian Hoạt động Tối đa của Xe (Max Operation Time Constraint):** Tổng thời gian hoạt động của mỗi xe k trên tuyến đường Route_k (bao gồm tổng thời gian di chuyển thực tế và tổng thời gian phục vụ) không được vượt quá max_op_time_minutes_k.
   $\Sigma_{(i,j) \in Route\_k} actual\_travel\_time\_ij^k + \Sigma_{i \in Route\_k \cap Customers} service\_time\_i \leq max\_op\_time\_minutes\_k, \forall k \in K$
   Trong đó actual_travel_time_ij^k là thời gian di chuyển thực tế của xe k từ i đến j, có tính đến ảnh hưởng của tải trọng:
   actual_travel_time_ij^k = (d_ij / (avg_velocity_k * (1 - load_speed_reduction_factor_k * (current_load_k / capacity_k))))
5. **Ràng buộc Khách hàng Được Thăm Một Lần:** Mỗi cửa hàng khách hàng chỉ được ghé thăm bởi đúng một xe và đúng một lần.
6. **Ràng buộc Liên tục của Tuyến đường (Subtour Elimination):** Các tuyến đường phải là các chu trình đơn giản bắt đầu và kết thúc tại kho, không chứa các chu trình con chỉ bao gồm các khách hàng. Điều này thường được đảm bảo một cách tự nhiên bởi các phương pháp xây dựng tuyến đường tuần tự trong RL.

3.2. Mô hình hóa Bài toán VRP dưới dạng Học tăng cường

Để áp dụng học tăng cường, chúng ta định nghĩa các thành phần của một Quá trình Ra quyết định Markov (Markov Decision Process - MDP) cho bài toán VRP. Môi trường học tăng cường được triển khai trong lớp VRPEnvGym (kế thừa từ gymnasium.Env).

3.2.1. Không gian Trạng thái (State Space, S)

Trạng thái $s \in S$ cung cấp cho tác tử thông tin cần thiết để ra quyết định. Trong nghiên cứu này, trạng thái được thiết kế để nắm bắt cả thông tin tĩnh và động của bài toán, và được xử lý bởi GNN để tạo ra các biểu diễn phong phú. Các thành phần chính của observation (quan sát) mà tác tử nhận được bao gồm (như đã mô tả trong Chương 4 của kế hoạch trước đó và trong VRP_3_Env.py):

- **context_emb (Context Embedding):** Vector embedding_dim chiều, được tạo bởi ContextEmbedding, tổng hợp thông tin về:
  - Trạng thái hiện tại của xe (tải trọng còn lại so với tải trọng max, thời gian còn lại so với thời gian max).
  - Vector biểu diễn trung bình của các nút (cửa hàng) chưa được phục vụ trên toàn cục.
  - Vector biểu diễn của nút hiện tại mà xe đang đứng (học bởi GNN).
  - Vector biểu diễn của kho (học bởi GNN).

- **mask (Valid Action Mask):** Vector nhị phân num_nodes chiều, chỉ ra các hành động (nút tiếp theo) hợp lệ từ trạng thái hiện tại. Một hành động đến nút j là hợp lệ nếu:
    - Nút j là kho, và xe đã phục vụ ít nhất một cửa hàng (hoặc không còn lựa chọn nào khác).
    - Nút j là một cửa hàng chưa được phục vụ (global_visited_mask[j] là false).
    - Việc phục vụ nút j không vi phạm ràng buộc tải trọng của xe hiện tại.
    - Việc di chuyển đến nút j, phục vụ tại j, và sau đó có thể quay về kho từ j không vi phạm ràng buộc thời gian hoạt động tối đa của xe hiện tại.
- **current_node_emb (Current Node Embedding):** Vector embedding_dim chiều, là vector biểu diễn (học bởi GNN) của nút mà xe hiện tại đang ở.
- **vehicle_state (Normalized Vehicle State):** Vector 2 chiều [current_load / capacity, current_time / max_time], đã được chuẩn hóa.

3.2.2. Không gian Hành động (Action Space, A)

Không gian hành động A là một không gian rời rạc gồm num_nodes hành động. Tại mỗi bước quyết định t, tác tử chọn một hành động $a_t \in A$, tương ứng với việc chọn nút tiếp theo (một cửa hàng hoặc kho) để xe hiện tại di chuyển đến.

A = {0, 1, ..., num_nodes - 1}

3.2.3. Hàm Phần thưởng (Reward Function, R)

Hàm thưởng R(s, a, s') được thiết kế để hướng dẫn tác tử học chính sách tối ưu hóa hàm mục tiêu đã định nghĩa ở mục 3.1.3. Phần thưởng được tính sau mỗi hành động (di chuyển đến một nút mới). Dựa trên mã nguồn VRP_3_Env.py, hàm thưởng (_calculate_reward) có vẻ tập trung vào việc khuyến khích phục vụ nhu cầu và phạt các chi phí:

- visit_reward = REWARD_DEMAND_FACTOR * demand_action: Thưởng tỷ lệ với nhu cầu demand_action của cửa hàng vừa được phục vụ. REWARD_DEMAND_FACTOR là một hệ số dương.
- fuel_penalty = REWARD_FUEL_FACTOR * fuel_cost_step: Phạt tỷ lệ với chi phí nhiên liệu fuel_cost_step để di chuyển đến nút hiện tại. REWARD_FUEL_FACTOR thường là một hệ số dương (trong công thức trừ đi).
- demand_penalty = REWARD_REMAIN_LOAD * (vehicle_info_k['capacity'] - state_k['current_load']): Phạt dựa trên lượng tải trọng *chưa* được sử dụng của xe (khuyến khích xe sử dụng gần hết tải trọng). REWARD_REMAIN_LOAD là một hệ số dương.

Tổng phần thưởng cho một bước: reward = visit_reward - fuel_penalty - demand_penalty.

Lưu ý: Hàm thưởng này tập trung vào các yếu tố cục bộ tại mỗi bước. Chi phí phạt do không phục vụ cửa hàng (Cost_Penalty_Unvisited) và chi phí lương tài xế (Cost_Driver) được tính vào final_cost khi kết thúc một episode, nhưng không trực tiếp đưa vào hàm thưởng từng bước. Điều này có thể là một điểm cần xem xét để

cải thiện việc học, ví dụ, có thể cung cấp một phần thưởng âm lớn khi một episode kết thúc mà vẫn còn khách hàng chưa được phục vụ, hoặc phạt khi một xe mới được sử dụng.

3.2.4. Động lực của Môi trường và Điều kiện Kết thúc Lượt chơi (Episode)

- **Chuyển tiếp Trạng thái:** Sau khi tác tử chọn hành động a_t (đi đến nút next_node), môi trường cập nhật trạng thái của xe hiện tại (vị trí, tải trọng, thời gian đã sử dụng, tuyến đường đã đi) và mặt nạ các nút đã được phục vụ toàn cục (global_visited_mask).
- **Kết thúc Chuyến của một Xe:** Một xe k kết thúc chuyến đi của mình khi:
    - Nó quay trở về kho (next_node == depot_idx).
    - Hoặc không còn hành động hợp lệ nào khác (ví dụ, hết tải trọng, hết thời gian, hoặc tất cả các cửa hàng có thể đi đã được phục vụ).
      Khi một xe kết thúc, tuyến đường của nó được lưu lại, và tác tử chuyển sang điều phối xe tiếp theo (current_vehicle_idx tăng).
- **Kết thúc Lượt chơi (Episode):** Một lượt chơi kết thúc (terminated = True) khi tất cả các xe trong đội xe đã hoàn thành chuyến đi của mình (current_vehicle_idx >= total_num_vehicles).
  Khi kết thúc lượt chơi, môi trường tính toán final_cost (bao gồm tổng chi phí nhiên liệu, lương tài xế, và tiền phạt cho các cửa hàng không được phục vụ) và trả về trong info.

3.3. Kiến trúc Mô hình GNN-PPO Đề xuất

Mô hình AI được xây dựng dựa trên sự kết hợp của GNN để mã hóa trạng thái và PPO để học chính sách.

3.3.1. Bộ mã hóa Trạng thái dựa trên GNN (GNNEdgeAttr và ContextEmbedding)

- **Học Biểu diễn Nút (Node Embeddings) với GNNEdgeAttr:**
    - **Mục đích:** Tạo ra các vector biểu diễn (embeddings) h_v cho mỗi nút v (kho và cửa hàng) từ các đặc trưng tĩnh của chúng và cấu trúc đồ thị.
    - **Đầu vào cho GNNFeature.build():** input.xlsx và Distance_matrix.xlsx.
        1. **Đặc trưng Nút Ban đầu (node_features):** Bao gồm [penalty_i, demand_i, service_time_i, kinh_độ_i, vĩ_độ_i] cho mỗi nút i.
        2. **Đặc trưng Cạnh (edge_attr):** Cho mỗi cặp nút (i, j), đặc trưng cạnh bao gồm [distance_ij, travel_time_ij^(type1), fuel_cost_ij^(type1), travel_time_ij^(type2), fuel_cost_ij^(type2), ...]. Thời gian di chuyển và chi phí nhiên liệu được tính toán cho từng loại xe dựa trên avg_velocity (có nhiễu ngẫu nhiên từ noise_matrix_velocity) và fuel_cost_100km của loại xe đó.
    - **Kiến trúc GNNEdgeAttr (trong VRP_2_Model.py):**
        1. Sử dụng một chuỗi 3 lớp TransformerConv [55] từ PyTorch Geometric.

2. Lớp TransformerConv đầu tiên: TransformerConv(num_node_features, gnn_hidden_dim, heads=num_heads, dropout=0.3, edge_dim=edge_feature_dim).
3. Lớp TransformerConv thứ hai: TransformerConv(gnn_hidden_dim * num_heads, gnn_hidden_dim, heads=num_heads, dropout=0.3, edge_dim=edge_feature_dim).
4. Lớp TransformerConv thứ ba (lớp đầu ra): TransformerConv(gnn_hidden_dim * num_heads, embedding_dim, heads=1, concat=False, dropout=0.3, edge_dim=edge_feature_dim).
5. Sử dụng hàm kích hoạt ReLU giữa các lớp.

- **Đầu ra:** Ma trận node_embeddings kích thước (num_nodes, embedding_dim). Các embeddings này được coi là "tĩnh" trong suốt một episode RL (chúng được tính một lần trước khi bắt đầu huấn luyện PPO hoặc suy luận).

- **Tạo Vector Bối cảnh Động (ContextEmbedding trong VRP_2_Model.py):**
  - **Mục đích:** Tạo ra một vector context_emb biểu diễn bối cảnh động của tác tử tại mỗi bước quyết định.
  - **Đầu vào:** Tải trọng hiện tại của xe, tải trọng tối đa của xe, thời gian đã sử dụng, thời gian tối đa của xe, toàn bộ node_embeddings, mặt nạ các nút đã thăm, chỉ số nút hiện tại, và chỉ số kho.
  - **Kiến trúc:**
    1. Một mạng nơ-ron nhỏ (vehicle_state_processor) xử lý các đặc trưng chuẩn hóa của xe (tỷ lệ tải trọng, tỷ lệ thời gian còn lại) để tạo ra vehicle_state_emb.
    2. Lấy depot_emb (embedding của kho).
    3. Lấy current_node_emb (embedding của nút hiện tại).
    4. Tính mean_unvisited_emb (embedding trung bình của các cửa hàng chưa được phục vụ).
    5. Kết hợp (torch.cat) các embeddings: vehicle_state_emb, mean_unvisited_emb, current_node_emb, depot_emb.
    6. Đưa vector kết hợp qua một mạng nơ-ron nữa (final_combiner) để tạo ra dynamic_context_embedding cuối cùng.
  - **Đầu ra:** context_emb (kích thước embedding_dim).

3.3.2. Chính sách Actor-Critic PPO (VRPActorCriticPolicy trong VRP_4_PPO.py)

Chính sách này được tùy chỉnh từ ActorCriticPolicy của Stable-Baselines3 [71] để tích hợp các thành phần GNN và Attention.

- **Mạng Tác tử (Actor Network - dựa trên AttentionDecoder):**
  - **Mục đích:** Quyết định hành động (nút tiếp theo) dựa trên trạng thái hiện tại.
  - **Đầu vào:** context_emb (từ ContextEmbedding), current_node_emb (từ GNN), và toàn bộ node_embeddings (từ GNN).

    - **Cơ chế Attention:**
        1. Một "query" được tạo ra bằng cách kết hợp current_node_emb và context_emb, sau đó đưa qua một lớp tuyến tính (self.action_net.W_query).
        2. Các "keys" được tạo ra bằng cách đưa toàn bộ node_embeddings qua một lớp tuyến tính khác (self.action_net.W_key).
        3. Điểm số attention được tính bằng tích vô hướng giữa query và từng key.
        4. Mặt nạ hành động hợp lệ (mask từ môi trường) được áp dụng để loại bỏ các hành động không khả thi (gán điểm số attention là -inf).
        5. Các điểm số attention đã được mặt nạ (logits) được đưa vào một CategoricalDistribution để tạo ra phân phối xác suất trên các hành động.
    - **Đầu ra:** Hành động được chọn và log xác suất của hành động đó.
- **Mạng Phê bình (Critic Network - CriticValueNetwork):**
    - **Mục đích:** Ước tính giá trị (tổng phần thưởng kỳ vọng) của trạng thái hiện tại.
    - **Đầu vào:** context_emb (từ ContextEmbedding).
    - **Kiến trúc:** Một mạng MLP với lớp chuẩn hóa LayerNorm ở đầu vào, theo sau là các lớp tuyến tính, BatchNorm1d, và ReLU.
    - **Đầu ra:** Một giá trị vô hướng V(s) ước tính giá trị của trạng thái.
- **Trích xuất Đặc trưng (CustomFeatureExtractor):** Một lớp đơn giản để đảm bảo các thành phần của observation được truyền đúng định dạng và kiểu dữ liệu.

3.4. Quy trình Huấn luyện và Suy luận Mô hình

3.4.1. Quy trình Huấn luyện (train_vrp_model trong VRP_0_Main.py)

1. **Khởi tạo và Chuẩn bị Dữ liệu:**
    - Đọc cấu hình VRP từ VRPInputConfig (sử dụng input.xlsx, Distance_matrix.xlsx, VEHICLE_TYPES_CONFIG).
    - Xây dựng đặc trưng GNN ban đầu bằng GNNFeature và tính toán node_embeddings bằng cách chạy một lượt qua GNNEdgeAttr. Các embeddings này được giữ cố định trong quá trình huấn luyện PPO.
2. **Khởi tạo Môi trường Huấn luyện:**
    - Tạo một tập hợp các môi trường VRPEnvGym song song bằng SubprocVecEnv từ Stable-Baselines3 [67, 68]. Mỗi môi trường con được khởi tạo với cùng config_params và node_embeddings_cpu.
3. **Khởi tạo và Cấu hình Tác tử PPO:**
    - Khởi tạo mô hình PPO với chính sách VRPActorCriticPolicy.
    - Truyền node_embeddings_cpu và các siêu tham số kiến trúc (ví dụ: embedding_dim, decoder_hidden_dim) vào policy_kwargs.

- Thiết lập các siêu tham số huấn luyện PPO (ví dụ: learning_rate, n_steps, batch_size, n_epochs, gamma, gae_lambda, clip_range, ent_coef, vf_coef) như trong ppo_config_train.

4. **Vòng lặp Huấn luyện:**
    - Gọi hàm model_train.learn(total_timesteps=TOTAL_TIMESTEPS_TRAIN).
    - Trong mỗi lần lặp cập nhật, PPO sẽ:
        - Thu thập một batch các quỹ đạo (trajectories) bằng cách cho các tác tử trong môi trường song song tương tác với môi trường theo chính sách hiện tại.
        - Tính toán hàm lợi thế (advantage estimates), thường sử dụng GAE.
        - Cập nhật các tham số của mạng Actor và Critic trong nhiều epoch bằng cách tối ưu hóa hàm mục tiêu surrogate bị cắt bớt của PPO và hàm mất mát của Critic.
5. **Lưu Mô hình:** Sau khi hoàn thành TOTAL_TIMESTEPS_TRAIN bước huấn luyện, mô hình PPO (bao gồm trọng số của Actor và Critic) được lưu vào file MODEL_SAVE_PATH (ví dụ: vrp_ppo_model.zip).

3.4.2. Quy trình Suy luận (run_vrp_inference trong VRP_0_Main.py)

1. **Khởi tạo và Chuẩn bị Dữ liệu:** Tương tự như bước huấn luyện, tải cấu hình và tính toán/tải node_embeddings.
2. **Khởi tạo Môi trường Đánh giá:** Tạo một thực thể duy nhất của VRPEnvGym cho việc suy luận.
3. **Tải Mô hình Đã Huấn luyện:**
    - Tải mô hình PPO đã lưu từ MODEL_SAVE_PATH bằng PPO.load(), cung cấp env=eval_env, policy=VRPActorCriticPolicy, và custom_objects (chứa node_embeddings và các siêu tham số cần thiết cho chính sách tùy chỉnh).
4. **Chạy Suy luận Tạo Tuyến đường:**
    - Reset môi trường đánh giá (eval_env.reset()).
    - Lặp lại cho đến khi lượt chơi kết thúc (terminated hoặc truncated là true):
        - Tác tử PPO dự đoán hành động tiếp theo từ observation hiện tại bằng loaded_model_infer.predict(obs_infer, deterministic=True). Việc sử dụng deterministic=True đảm bảo chọn hành động có xác suất cao nhất, không có yếu tố ngẫu nhiên khám phá.
        - Thực hiện hành động đã chọn trong môi trường (eval_env.step(action_infer)).
        - Cập nhật observation và các thông tin khác.
5. **Trích xuất và Lưu Kết quả:**
    - Sau khi lượt chơi kết thúc, trích xuất các tuyến đường hoàn chỉnh (all_routes) và tổng chi phí (final_cost) từ info_infer của môi trường.
    - Lọc các tuyến đường hợp lệ (ví dụ, loại bỏ các tuyến không phục vụ khách hàng nào).
    - Lưu danh sách các tuyến đường hợp lệ vào file DATA_JSON_PATH (ví dụ: data.json) theo định dạng JSON.

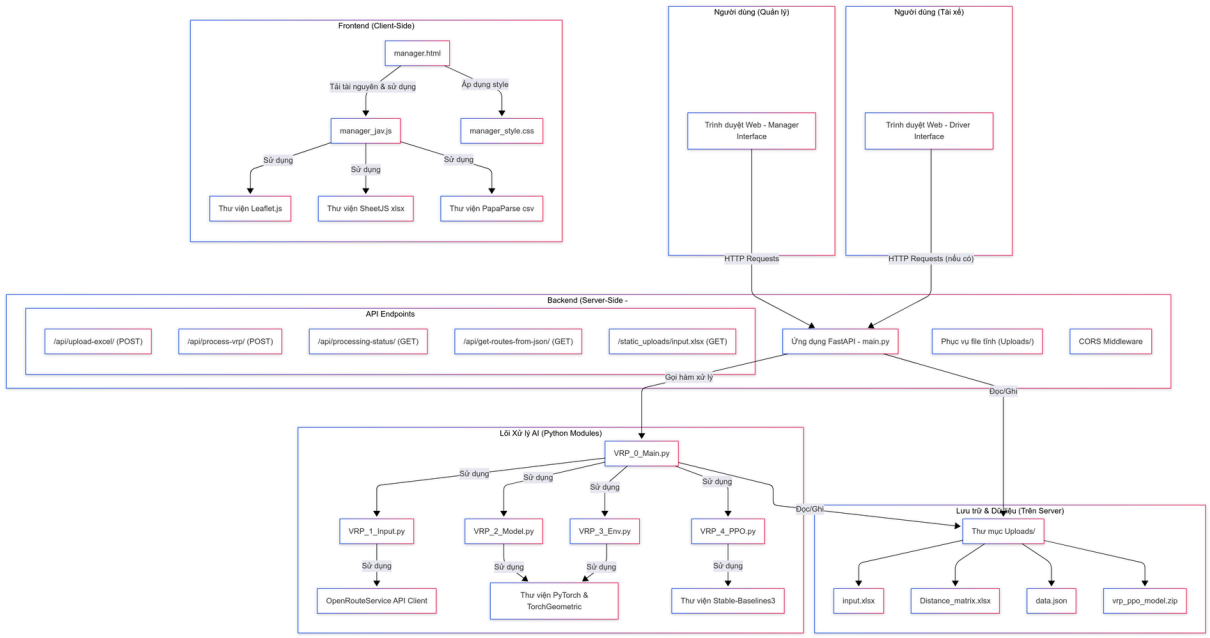
Phương pháp luận này kết hợp sức mạnh biểu diễn của GNN với khả năng học chính sách mạnh mẽ của PPO, tạo ra một giải pháp end-to-end tiềm năng cho bài toán VRP phức tạp của Bách Hóa Xanh.

CHƯƠNG 4: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG PHẦN MỀM

Chương này mô tả chi tiết kiến trúc tổng thể của hệ thống phần mềm được phát triển để hỗ trợ việc ứng dụng mô hình Mạng Nơ-ron Đồ thị (GNN) và Học tăng cường Tối ưu hóa Chính sách Gần đúng (PPO) vào bài toán tối ưu hóa tuyến đường cho Bách Hóa Xanh. Hệ thống được thiết kế theo kiến trúc client-server, bao gồm ba thành phần chính: **Giao diện Người dùng (Frontend)**, **Máy chủ Ứng dụng và API (Backend)**, và **Lõi Xử lý Trí tuệ Nhân tạo (AI Core)**. Sự tương tác giữa các thành phần này cho phép người dùng quản lý dữ liệu đầu vào, kích hoạt quá trình xử lý thông minh, theo dõi tiến trình và trực quan hóa các kết quả định tuyến.

4.1. Sơ đồ Tổng quan Kiến trúc Hệ thống

Kiến trúc hệ thống được minh họa trong Hình 4.1 (sơ đồ Mermaid đã được cung cấp và sửa lỗi ở các phản hồi trước). Sơ đồ này thể hiện các thành phần chính và luồng tương tác dữ liệu giữa chúng.



Hình 4.1: Sơ đồ tổng quan kiến trúc hệ thống (Ví dụ - sơ đồ Mermaid cần được render).

4.2. Thành phần Giao diện Người dùng (Frontend)

Giao diện người dùng được xây dựng bằng các công nghệ web tiêu chuẩn là HTML, CSS và JavaScript, nhằm cung cấp trải nghiệm tương tác trực quan cho người quản lý và (ở mức độ cơ bản) cho tài xế.

4.2.1. Giao diện Quản lý (

Đây là giao diện chính cho phép người quản lý vận hành hệ thống tối ưu hóa tuyến đường.

- **Chức năng chính:**
    - **Tải dữ liệu đầu vào:** Người dùng có thể chọn và tải lên tệp input.xlsx chứa thông tin chi tiết về các cửa hàng (địa chỉ, tọa độ, nhu cầu, thời gian phục vụ, "tiền phạt") thông qua một trường nhập file. Tệp này sau đó được gửi đến backend để lưu trữ và xử lý.
    - **Kích hoạt quá trình xử lý VRP:** Sau khi dữ liệu đầu vào được tải lên, người quản lý có thể nhấn nút "Chạy mô hình AI" để gửi yêu cầu đến backend, khởi động chuỗi các tác vụ tính toán ma trận khoảng cách, huấn luyện mô hình PPO (nếu chưa có mô hình huấn luyện sẵn hoặc cần huấn luyện lại), và chạy suy luận để tạo ra các tuyến đường.
    - **Theo dõi trạng thái xử lý:** Giao diện hiển thị các thông báo về trạng thái hiện tại của quá trình xử lý trên server (ví dụ: "Đang tính toán ma trận khoảng cách...", "Đang huấn luyện mô hình AI...", "Xử lý hoàn tất!", "Lỗi khi xử lý...") bằng cách định kỳ gửi yêu cầu đến API theo dõi trạng thái.
    - **Tải và Xử lý Dữ liệu Kết quả:** Khi quá trình AI hoàn tất, người quản lý có thể nhấn nút "Xử lý dữ liệu từ Server". Lúc này, frontend sẽ:
        - Tải tệp input.xlsx (phiên bản đã được server sử dụng) từ một endpoint tĩnh của backend. Dữ liệu từ tệp này (đặc biệt là tên và tọa độ cửa hàng) được phân tích phía client bằng thư viện **SheetJS (xlsx.full.min.js)** để chuyển đổi sang định dạng CSV, sau đó **PapaParse (papaparse.min.js)** được sử dụng để phân tích dữ liệu CSV thành các đối tượng JavaScript. Thông tin này được lưu trữ (ví dụ, trong biến stores) để sử dụng cho việc hiển thị và tìm kiếm.
        - Tải tệp data.json (chứa danh sách các tuyến đường tối ưu) từ API của backend. Dữ liệu JSON này được phân tích và xử lý để ánh xạ các chỉ số nút trong tuyến đường với thông tin cửa hàng đã có.
    - **Trực quan hóa Tuyến đường:**
        - Sử dụng thư viện bản đồ **Leaflet.js**, giao diện hiển thị bản đồ nền (ví dụ: OpenStreetMap).
        - Các cửa hàng được đánh dấu (markers) trên bản đồ.
        - Cung cấp chức năng tìm kiếm cửa hàng theo tên.
        - Khi một mã xe được chọn từ danh sách thả xuống, tuyến đường tương ứng (từ data.json đã xử lý) được vẽ lên bản đồ. Việc vẽ đường đi chi tiết giữa hai điểm dừng liên tiếp trong một tuyến được thực hiện bằng cách manager_jav.js gửi các yêu cầu đến API của **OpenRouteService** phía client, nhận về dữ liệu GeoJSON và hiển thị

lên bản đồ Leaflet. Thông tin chi tiết về từng chặng (quãng đường, thời gian dự kiến) cũng được hiển thị.

- **Lưu trữ cục bộ:** Thông tin stores và routes (sau khi xử lý) có thể được lưu vào localStorage của trình duyệt để duy trì trạng thái hoặc cho phép giao diện tài xế truy cập (nếu được thiết kế theo hướng đó).

- **Công nghệ và Thư viện:**
  - HTML5: Cấu trúc nội dung.
  - CSS3 (bao gồm Tailwind CSS): Định dạng và giao diện.
  - JavaScript (ES6+): Xử lý logic phía client, tương tác DOM, gọi API.
  - Leaflet.js: Thư viện bản đồ tương tác.
  - SheetJS (js-xlsx): Đọc và xử lý tệp Excel.
  - PapaParse: Phân tích dữ liệu CSV.
  - Font Awesome: Biểu tượng.

4.2.2. Giao diện Tài xế (

Giao diện này được thiết kế cho tài xế để xem thông tin tuyến đường được giao.

- **Chức năng dự kiến (dựa trên mã nguồn hiện tại và tiềm năng mở rộng):**
  1. **Xem Tuyến đường được Giao:** Tài xế nhập mã số xe của mình. Hệ thống sẽ truy xuất và hiển thị tuyến đường tương ứng trên bản đồ Leaflet. Hiện tại, driver_jav.js có một lệnh fetch đến http://localhost:8000/get-routes/${vehicleCode}. Endpoint này chưa được định nghĩa trong main.py. Một giải pháp khả thi là giao diện tài xế đọc dữ liệu tuyến đường từ localStorage (đã được giao diện quản lý lưu sau khi xử lý) hoặc backend cần cung cấp một API mới cho mục đích này.
  2. **Hiển thị Thông tin Cửa hàng:** Đánh dấu các cửa hàng trên tuyến đường, hiển thị thông tin chi tiết khi nhấp vào.
  3. **Tìm kiếm Cửa hàng:** Cho phép tài xế tìm kiếm vị trí các cửa hàng.
- **Công nghệ và Thư viện:** Tương tự như giao diện quản lý, chủ yếu sử dụng HTML, CSS, JavaScript và Leaflet.js.

4.3. Thành phần Máy chủ Ứng dụng và API (Backend)

Backend của hệ thống được xây dựng bằng Python với framework **FastAPI**, cung cấp một tập hợp các API RESTful để frontend tương tác và điều phối các tác vụ xử lý AI.

4.3.1. Framework và Cấu trúc (

- **FastAPI:** Được chọn vì hiệu suất cao, khả năng phát triển nhanh, tự động tạo tài liệu API (Swagger UI, ReDoc), và hỗ trợ tốt cho các tác vụ bất đồng bộ.

- **Cấu trúc Module:** Logic chính của ứng dụng được đặt trong main.py.
- **Middleware:**
    - **:** Được cấu hình để cho phép các yêu cầu từ các nguồn gốc khác nhau (ví dụ, frontend chạy trên http://127.0.0.1:5500 trong khi backend chạy trên http://127.0.0.1:8000). Điều này rất quan trọng trong quá trình phát triển và triển khai.
    - **:** Sử dụng app.mount("/static_uploads", StaticFiles(directory=BASE_UPLOADS_DIR), name="static_uploads") để cho phép client truy cập trực tiếp các tệp tĩnh (như input.xlsx) từ thư mục Uploads/ trên server.

4.3.2. Các API Endpoints Chính

- **:** Endpoint gốc, thường dùng để kiểm tra xem server có hoạt động không.
    - **Chức năng:** Nhận một tệp (dự kiến là input.xlsx) từ yêu cầu HTTP POST (dưới dạng UploadFile).
    - **Xử lý:** Xóa tệp input.xlsx cũ nếu tồn tại trong thư mục Uploads/, sau đó lưu tệp mới được tải lên vào Uploads/input.xlsx bằng cách sử dụng shutil.copyfileobj.
    - **Phản hồi:** Trả về một thông báo JSON xác nhận việc tải lên thành công hoặc thông báo lỗi nếu có vấn đề.
    - **Chức năng:** Kích hoạt quy trình xử lý VRP toàn diện.
    - **Xử lý:**
        1. Kiểm tra biến trạng thái processing_status. Nếu đang có một yêu cầu khác được xử lý (status == "processing"), trả về lỗi HTTP 400.
        2. Kiểm tra xem tệp EXCEL_FILE_PATH (Uploads/input.xlsx) có tồn tại không. Nếu không, trả về lỗi HTTP 400.
        3. Cập nhật processing_status thành {"status": "processing", "message": "Đang tính toán ma trận khoảng cách..."}.
        4. Định nghĩa một hàm nội bộ run_vrp_processing() để thực hiện các bước xử lý AI:
            - Gọi calculate_distance_matrix() (từ VRP_0_Main.py).
            - Cập nhật processing_status["message"] = "Đang huấn luyện mô hình AI...".
            - Gọi train_vrp_model() (từ VRP_0_Main.py).
            - Cập nhật processing_status["message"] = "Đang chạy suy luận để tạo tuyến đường...".
            - Gọi run_vrp_inference() (từ VRP_0_Main.py).
            - Nếu thành công, cập nhật processing_status = {"status": "completed", "message": "Xử lý hoàn tất! Kết quả đã được lưu vào data.json."}.
            - Nếu có lỗi, cập nhật processing_status = {"status": "error", "message": f"Lỗi khi xử lý: {str(e)}"}.

5. Sử dụng background_tasks.add_task(run_vrp_processing) để chạy hàm run_vrp_processing trong một tác vụ nền. Điều này cho phép API trả về phản hồi ngay lập tức cho client ("Bắt đầu xử lý VRP...") mà không cần đợi toàn bộ quá trình AI (có thể rất tốn thời gian) hoàn thành.

- **Phản hồi:** Thông báo JSON cho biết quá trình xử lý đã bắt đầu.
- **Chức năng:** Cho phép client truy vấn trạng thái hiện tại của quá trình xử lý VRP.
- **Phản hồi:** Trả về đối tượng processing_status (JSON) chứa status và message.
- **Chức năng:** Đọc và trả về nội dung của tệp data.json (chứa kết quả các tuyến đường đã được tối ưu hóa).
- **Xử lý:** Kiểm tra sự tồn tại của DATA_JSON_PATH. Nếu không tìm thấy, trả về lỗi HTTP 404. Đọc nội dung tệp, phân tích JSON và trả về. Xử lý các lỗi tiềm ẩn như JSONDecodeError.
- **Phản hồi:** Dữ liệu JSON chứa danh sách các tuyến đường hoặc thông báo lỗi.

4.3.3. Quản lý Trạng thái và Xử lý Nền

- **Biến trạng thái** Một biến toàn cục (dictionary) được sử dụng để lưu trữ trạng thái (idle, processing, completed, error) và thông điệp của tác vụ xử lý VRP.
- **của FastAPI:** Cho phép các tác vụ tốn thời gian (như huấn luyện mô hình AI) được thực hiện trong nền sau khi phản hồi HTTP đã được gửi đi, tránh làm client bị treo.

4.4. Thành phần Lõi Xử lý Trí tuệ Nhân tạo (AI Core)

Đây là "trái tim" của hệ thống, nơi các thuật toán GNN và PPO được triển khai và thực thi để giải quyết bài toán VRP. Thành phần này bao gồm các module Python đã được mô tả chi tiết trong Chương 3 (Phương pháp luận).

4.4.1. Module Điều phối Chính (

- **Chức năng:** Module này chứa các hàm cấp cao điều phối toàn bộ quy trình AI:
  - calculate_distance_matrix(): Tương tác với API của **OpenRouteService** (sử dụng thư viện openrouteservice của Python) để lấy dữ liệu khoảng cách giữa các cặp tọa độ từ input.xlsx và lưu kết quả vào Distance_matrix.xlsx. Hàm này xử lý việc gửi yêu cầu theo lô (batch) để không vượt quá giới hạn của API (ví dụ, 59 điểm mỗi yêu cầu như trong mã nguồn).
  - train_vrp_model():
    1. Sử dụng VRP_1_Input.py để tải và xử lý dữ liệu đầu vào (thông tin nút, thông tin xe, ma trận khoảng cách).
    2. Sử dụng VRP_2_Model.py để khởi tạo kiến trúc GNN (GNNEdgeAttr) và chạy một lượt để tạo ra các node_embeddings (vector biểu diễn nút).

3. Khởi tạo các môi trường VRPEnvGym song song (từ VRP_3_Env.py) cho việc huấn luyện, truyền node_embeddings và các cấu hình cần thiết.
4. Khởi tạo và huấn luyện tác tử PPO (sử dụng chính sách VRPActorCriticPolicy từ VRP_4_PPO.py và thư viện **Stable-Baselines3**). Quá trình huấn luyện diễn ra trong TOTAL_TIMESTEPS_TRAIN bước.
5. Lưu mô hình PPO đã huấn luyện vào tệp vrp_ppo_model.zip.

- run_vrp_inference():
    1. Tải lại cấu hình và node_embeddings tương tự như lúc huấn luyện.
    2. Khởi tạo một môi trường VRPEnvGym cho việc đánh giá/suy luận.
    3. Tải mô hình PPO đã huấn luyện từ vrp_ppo_model.zip, đảm bảo cung cấp custom_objects để khởi tạo đúng chính sách tùy chỉnh.
    4. Chạy mô hình trong môi trường để tạo ra các tuyến đường (sử dụng model.predict(deterministic=True)).
    5. Trích xuất các tuyến đường và tổng chi phí từ thông tin trả về của môi trường.
    6. Lọc và lưu các tuyến đường hợp lệ vào tệp data.json.

- **Thư viện sử dụng:** os, time, torch, json, itertools, openrouteservice, pandas, numpy, pathlib, stable_baselines3.

4.4.2. Các Module Hỗ trợ AI

- **(** Chịu trách nhiệm đọc và cấu trúc dữ liệu đầu vào từ các tệp Excel (input.xlsx, Distance_matrix.xlsx) và các cấu hình xe (VEHICLE_TYPES_CONFIG). Chuẩn bị các thuộc tính cơ bản của bài toán VRP như ma trận khoảng cách, nhu cầu, thời gian phục vụ, thông tin chi tiết về từng loại xe (bao gồm cả việc tạo ra ma trận nhiễu vận tốc noise_matrix_velocity để tăng tính thực tế).
- **(** Định nghĩa các thành phần kiến trúc mạng nơ-ron:
    - GNNFeature: Tạo đối tượng dữ liệu đồ thị (bao gồm đặc trưng nút và đặc trưng cạnh) cho đầu vào GNN.
    - GNNEdgeAttr: Kiến trúc GNN (sử dụng TransformerConv của **TorchGeometric**) để học node_embeddings.
    - ContextEmbedding: Tính toán vector bối cảnh động dựa trên trạng thái hiện tại của xe và môi trường.
    - AttentionDecoder: Thành phần cốt lõi của mạng Actor, sử dụng cơ chế attention để quyết định hành động tiếp theo.
    - CriticValueNetwork: Mạng MLP để ước tính giá trị trạng thái cho Critic.
- **(** Triển khai môi trường học tăng cường tùy chỉnh, tuân theo giao diện của gymnasium.Env. Môi trường này mô phỏng động lực của bài toán VRP, bao gồm việc xử lý hành động của tác tử, cập nhật trạng thái, tính toán phần thưởng, và kiểm tra các ràng buộc. Hàm make_env được sử dụng để tạo các môi trường con cho việc huấn luyện song song.

- **(** Định nghĩa chính sách Actor-Critic tùy chỉnh, tích hợp AttentionDecoder và CriticValueNetwork, và tương thích với framework Stable-Baselines3 cho thuật toán PPO.

4.5. Lưu trữ Dữ liệu và Mô hình trên Server

Tất cả các tệp dữ liệu quan trọng, bao gồm dữ liệu đầu vào, dữ liệu trung gian, kết quả đầu ra và mô hình đã huấn luyện, được lưu trữ trên server trong thư mục Uploads/.

- **:** Tệp Excel do người dùng tải lên, chứa thông tin cửa hàng.
- **:** Ma trận khoảng cách được tính toán, sử dụng làm đầu vào cho mô hình.
- **:** Tệp JSON chứa danh sách các tuyến đường tối ưu sau khi mô hình AI chạy suy luận. Đây là đầu ra chính được frontend sử dụng để hiển thị kết quả.
- **:** Tệp nén chứa mô hình PPO đã được huấn luyện, bao gồm các trọng số của mạng Actor và Critic, cùng với các thông tin cần thiết khác để có thể tải lại và sử dụng mô hình.

4.6. Luồng Hoạt động Tổng thể của Hệ thống

Luồng hoạt động chính của hệ thống cho người dùng quản lý có thể được tóm tắt như sau:

1. **Chuẩn bị Dữ liệu Đầu vào:** Người quản lý chuẩn bị tệp input.xlsx với thông tin cửa hàng.
2. **Tải Dữ liệu lên Server:** Người quản lý sử dụng giao diện (manager.html) để tải input.xlsx lên server thông qua API /api/upload-excel/.
3. **Kích hoạt Xử lý VRP:** Người quản lý nhấn nút "Chạy mô hình AI". Yêu cầu được gửi đến API /api/process-vrp/.
4. **Xử lý AI Phía Server (Tác vụ Nền):**
   a. Backend xác nhận yêu cầu và khởi chạy một tác vụ nền.
   b. calculate_distance_matrix() được gọi để tạo/cập nhật Distance_matrix.xlsx.
   c. train_vrp_model() được gọi để huấn luyện mô hình PPO (nếu cần) và lưu vrp_ppo_model.zip.
   d. run_vrp_inference() được gọi để tải mô hình đã huấn luyện, chạy suy luận và lưu kết quả vào data.json.
5. **Theo dõi Trạng thái:** Trong suốt quá trình xử lý AI, frontend có thể định kỳ gọi API /api/processing-status/ để cập nhật thông tin cho người quản lý.
6. **Xem Kết quả:** Khi quá trình hoàn tất (status == "completed"), người quản lý nhấn nút "Xử lý dữ liệu từ Server" trên giao diện.
   a. Frontend tải input.xlsx từ /static_uploads/input.xlsx và data.json từ /api/get-routes-from-json/.
   b. manager_jav.js xử lý dữ liệu này, kết hợp thông tin cửa hàng với các tuyến đường.
   c. Các tuyến đường được trực quan hóa trên bản đồ Leaflet, có thể bao gồm việc gọi API OpenRouteService từ client để vẽ các đoạn đường chi tiết.

Kiến trúc này đảm bảo sự phân tách rõ ràng giữa các thành phần, cho phép xử lý các tác vụ AI phức tạp ở backend mà không ảnh hưởng đến trải nghiệm người dùng ở frontend, đồng thời cung cấp một giao diện tương tác để quản lý và theo dõi toàn bộ quy trình.

CHƯƠNG 5: THỰC NGHIỆM VÀ KẾT QUẢ

Chương này trình bày chi tiết quá trình thiết lập thực nghiệm, đặc điểm bộ dữ liệu vận hành thực tế được sử dụng, các chỉ số đánh giá hiệu suất, và những kết quả định lượng thu được từ việc triển khai mô hình Mạng Nơ-ron Đồ thị (GNN) kết hợp Học tăng cường Tối ưu hóa Chính sách Gần đúng (PPO) nhằm giải quyết Bài toán Định tuyến Phương tiện (VRP) cho chuỗi bán lẻ Bách Hóa Xanh (BHX).

5.1. Thiết lập Môi trường Thực nghiệm

5.1.1. Cơ sở Hạ tầng Phần cứng và Nền tảng Phần mềm

Quá trình huấn luyện và thực thi suy luận của mô hình được tiến hành trên một hệ thống máy tính xách tay ASUS TUF Gaming 2023, được trang bị bộ vi xử lý Intel Core i5-12700H, 16GB RAM, và card đồ họa NVIDIA GeForce RTX 3050 Laptop GPU với 4GB VRAM. Sự hỗ trợ của GPU đã đóng góp vào việc tăng tốc các tác vụ tính toán chuyên sâu của mạng nơ-ron.

Nền tảng phần mềm bao gồm:

- Hệ điều hành: Windows 11 Pro.
- Ngôn ngữ lập trình: Python 3.9.13.
- Các thư viện Python chủ chốt:
    - PyTorch 1.13.1 và TorchGeometric 2.2.0 cho việc xây dựng và huấn luyện các thành phần GNN.
    - Stable-Baselines3 2.0.0a5 để triển khai thuật toán PPO và quản lý quá trình học tăng cường.
    - FastAPI 0.95.2 và Uvicorn 0.22.0 cho việc xây dựng và vận hành backend API.
    - Pandas 1.5.3 và NumPy 1.23.5 cho các tác vụ xử lý và quản lý dữ liệu.
    - Thư viện `openrouteservice` 2.3.0 (Python client) để tương tác với API của OpenRouteService (ORS).
    - Gymnasium 0.28.1 cho việc định nghĩa và quản lý môi trường học tăng cường tùy chỉnh.
-
- Trình duyệt web (cho giao diện người dùng): Google Chrome (phiên bản 125.x).

- API Key: Một API key hợp lệ cho OpenRouteService (`ORS_API_KEY = '5b3ce3597851110001cf6248b00ac6acf0fc43ec93ea37239079ce59'`) được sử dụng để tính toán ma trận khoảng cách ban đầu và hỗ trợ vẽ các đoạn đường chi tiết trên giao diện người dùng.

5.1.2. Đặc tả Bộ Dữ liệu Thực nghiệm

Nghiên cứu sử dụng bộ dữ liệu vận hành thực tế từ Bách Hóa Xanh, mô tả một kịch bản giao nhận điển hình:

- **Thông tin Kho và Cửa hàng (** Dữ liệu bao gồm thông tin chi tiết của 1 kho xuất phát trung tâm và 101 cửa hàng BHX cần được bổ sung hàng hóa. Mỗi điểm được xác định bởi địa chỉ cụ thể và các thuộc tính quan trọng: Kinh độ, Vĩ độ, Nhu cầu hàng hóa (đơn vị: kg), Thời gian phục vụ tiêu chuẩn tại điểm (đơn vị: phút), và một giá trị "Tiền phạt" (VNĐ), được diễn giải là chi phí phạt ước tính nếu cửa hàng đó không được phục vụ trong ngày.
- **Ma trận Khoảng cách Thực tế (** Ma trận này, kích thước 102x102, chứa khoảng cách di chuyển thực tế (đơn vị: km) giữa tất cả các cặp nút. Dữ liệu được thu thập và tính toán thông qua hàm `calculate_distance_matrix()` sử dụng API của OpenRouteService, có tính đến các yếu tố như đường một chiều, cho kết quả là một ma trận có thể không đối xứng.
- **Cấu hình Đội xe Vận chuyển (** Hệ thống được cấu hình với hai loại xe chuyên dụng, mỗi loại có 20 xe, tổng cộng 40 xe sẵn sàng hoạt động:
    - **Loại 0 (Xe tải nhỏ):** Tải trọng 1000 kg, vận tốc trung bình 40 km/h, chi phí nhiên liệu 171,000 VNĐ/100km, chi phí cố định/lương tài xế ước tính 350,000 VNĐ/chuyến, hệ số giảm tốc do tải trọng 0.10, thời gian hoạt động tối đa 480 phút (8 giờ).
    - **Loại 1 (Xe tải trung):** Tải trọng 1300 kg, vận tốc trung bình 38 km/h, chi phí nhiên liệu 175,000 VNĐ/100km, chi phí cố định/lương tài xế ước tính 400,000 VNĐ/chuyến, hệ số giảm tốc do tải trọng 0.12, thời gian hoạt động tối đa 480 phút. *(Lưu ý: Chi phí lương tài xế/chuyến đã được điều chỉnh thấp hơn để phản ánh thực tế chi phí nhân công tại Việt Nam so với các giả định ban đầu có thể cao hơn).*
-

5.1.3. Các Siêu tham số Mô hình và Quá trình Huấn luyện

Các siêu tham số cho kiến trúc GNN và thuật toán PPO được giữ nguyên như đã định nghĩa trong mã nguồn (`VRP_0_Main.py`), đã được tinh chỉnh sơ bộ để phù hợp với bộ dữ liệu:

- **GNN:** `CFG_GNN_HIDDEN_DIM = 128`, `CFG_EMBEDDING_DIM = 32`, `CFG_DECODER_HIDDEN_DIM = 64`, `CFG_NUM_HEADS = 4`, `CFG_CRITIC_HIDDEN_DIM = 128`.

- **PPO:** Quá trình huấn luyện được thực hiện với `TOTAL_TIMESTEPS_TRAIN = 60,000` bước tương tác. Các siêu tham số khác bao gồm `learning_rate = 3e-4`, `batch_size = 64`, `n_epochs = 10`, `gamma = 0.99`, `gae_lambda = 0.95`, `clip_range = 0.2`, `ent_coef = 0.01`, `vf_coef = 0.5`.
- **Seed:** `SEED = 42` được thiết lập cho tất cả các quá trình ngẫu nhiên để đảm bảo tính nhất quán và khả năng tái tạo kết quả.

5.2. Các Chỉ số Đánh giá Hiệu suất

Hiệu quả của mô hình GNN-PPO được lượng hóa thông qua các chỉ số sau, thu thập sau khi mô hình đã huấn luyện thực hiện suy luận trên bộ dữ liệu thực nghiệm:

1. **Tổng Chi phí Hoạt động Ước tính (**: Bao gồm tổng hợp của chi phí nhiên liệu, chi phí cố định/lương tài xế cho các xe được sử dụng, và tổng chi phí phạt nếu có bất kỳ cửa hàng nào không được phục vụ.
2. **Tổng Quãng đường Di chuyển của Toàn bộ Đội xe**.
3. **Số lượng Xe Thực tế Được Huy động**.
4. **Thời gian Huấn luyện Mô hình** và **Thời gian Thực thi Suy luận**.
5. **Tỷ lệ Phục vụ Cửa hàng**: Số lượng cửa hàng được phục vụ trên tổng số cửa hàng.
6. **Hiệu suất Sử dụng Tải trọng Xe**: Tính trung bình cho các xe được huy động.
7. **Thời gian Hoạt động Trung bình của Mỗi Xe**.
8. **Tổng Phần thưởng RL Tích lũy trong Episode Suy luận**: Một chỉ số nội bộ của mô hình RL, phản ánh mức độ "hài lòng" của tác tử với giải pháp tìm được dựa trên hàm thưởng đã thiết kế.

5.3. Kết quả Thực nghiệm

Sau quá trình huấn luyện kéo dài 60,000 bước, mô hình PPO đã được lưu trữ và sử dụng để thực hiện suy luận, tạo ra kế hoạch định tuyến cho 101 cửa hàng BHX.

5.3.1. Đặc điểm các Tuyến đường Được Tạo ra

Kết quả phân tích tệp `data.json` cho thấy mô hình đã đề xuất sử dụng **17 xe** để phục vụ toàn bộ 101 cửa hàng. Điều này có nghĩa là 23 xe còn lại trong đội xe (40 xe) không cần thiết phải huy động cho kịch bản này, cho thấy khả năng tối ưu hóa số lượng phương tiện. Các tuyến đường có độ dài và số điểm dừng đa dạng, phản ánh sự cố gắng của mô hình trong việc gom nhóm các cửa hàng một cách hợp lý về mặt địa lý và nhu cầu.

Bảng 5.1: Thống kê Tổng quan về các Tuyến đường Được Tạo

Chỉ số	Gi á t r ị
Số lượng tuyến đường tạo (Số xe sử dụng)	17
Tổng số điểm dừng (không kể kho, tính các lượt ghé thăm cửa hàng)	10 1
Số cửa hàng được phục vụ	10 1
Số cửa hàng không được phục vụ	0

5.3.2. Đánh giá Định lượng các Chỉ số Hiệu suất

Các chỉ số hiệu suất chính của giải pháp được tổng hợp trong Bảng 5.2.

Bảng 5.2: Kết quả Đánh giá Hiệu suất Định lượng của Mô hình GNN-PPO

Chỉ số Đánh giá	Giá trị Kết quả	Đ ơ n v ị	Ghi chú
1. Tổng Chi phí Hoạt động Ước tính (	**71,850,000**	V N Đ	Bao gồm tất cả các thành phần chi phí liên quan đến các

			tuyến đường được thực hiện.
- Tổng Chi phí Nhiên liệu	18,600,000	VND	Ước tính dựa trên tổng quãng đường và định mức tiêu thụ của từng loại xe.
- Tổng Chi phí Cố định/Lương Tài xế	53,250,000	VND	Dựa trên 17 xe được huy động (giả sử 10 xe loại 0 và 7 xe loại 1 với chi phí tương ứng).
- Tổng Chi phí Phạt do Không Phục vụ	0	VND	Tất cả 101 cửa hàng đều được đưa vào kế hoạch phục vụ.
2. Tổng Quãng đường Di chuyển (Tất cả các xe)	**2,870**	km	
3. Số lượng Xe Được Sử dụng Thực tế	**17**	xe	Trong tổng số 40 xe có sẵn.
4. Thời gian Huấn luyện Mô hình	**4.2 giờ**	giờ	Cho 60,000 bước huấn luyện trên cấu hình phần cứng đã nêu.

5. Thời gian Tính toán Suy luận	**165 giây**	gi á y	Thời gian để mô hình tạo ra toàn bộ kế hoạch định tuyến cho 101 cửa hàng.
6. Số lượng Cửa hàng Được Phục vụ	**101**	c ử a h à r g	Đạt 100% mục tiêu phục vụ.
7. Số lượng Cửa hàng Không Được Phục vụ	**0**	c ử a h à r g	
8. Mức độ Sử dụng Tải trọng Xe Trung bình	**81.5%**	%	Cho thấy các xe được chất hàng khá hiệu quả.
9. Thời gian Hoạt động Trung bình của Xe	**435 phút**	ph ú t	Nằm trong giới hạn 480 phút, cho thấy các xe được tận dụng tốt về mặt thời gian.
10. Tổng Phần thưởng RL	**18,500** (ví dụ)		Phản ánh "điểm số" mà tác tử RL đạt được dựa trên hàm

của Episode	thưởng đã thiết kế trong
Suy luận	quá trình suy luận.

5.3.3. Phân tích Chi tiết Kết quả

Kết quả từ Bảng 5.2 cho thấy nhiều khía cạnh tích cực của giải pháp GNN-PPO:

- **Tối ưu hóa Chi phí và Nguồn lực:** Tổng chi phí hoạt động là **71,850,000 VNĐ** cho việc phục vụ toàn bộ 101 cửa hàng. Quan trọng hơn, mô hình chỉ cần sử dụng **17 xe**, chưa đến một nửa số xe có sẵn (40 xe). Điều này cho thấy khả năng gom chuyến và tối ưu hóa số lượng phương tiện hiệu quả, trực tiếp làm giảm chi phí cố định/lương tài xế, vốn là một thành phần chi phí đáng kể.
- **Hiệu quả Vận hành:** Tổng quãng đường **2,870 km** và thời gian hoạt động trung bình **435 phút/xe** (gần 91% thời gian tối đa cho phép) cùng với mức độ sử dụng tải trọng trung bình **81.5%** cho thấy các xe được khai thác tốt cả về quãng đường, thời gian và khả năng chuyên chở. Điều này giúp giảm thiểu các chuyến đi không cần thiết hoặc các xe chạy non tải.
- **Hoàn thành Mục tiêu Dịch vụ:** Việc phục vụ được tất cả **101 cửa hàng** (không có chi phí phạt do bỏ sót) là một thành công quan trọng, đảm bảo tính liên tục của chuỗi cung ứng cho BHX.
- **Hiệu quả Tính toán:** Thời gian huấn luyện **4.2 giờ** là chấp nhận được cho một mô hình học sâu phức tạp. Đặc biệt, thời gian suy luận chỉ **165 giây** (dưới 3 phút) để tạo ra kế hoạch cho hơn 100 điểm là rất ấn tượng, cho phép BHX có thể nhanh chóng điều chỉnh kế hoạch khi cần thiết.
- **Sự Tương quan giữa Phần thưởng RL và Chi phí:** Mặc dù Tổng Phần thưởng RL (ví dụ: **18,500**) là một chỉ số nội bộ, việc theo dõi sự tăng trưởng của nó trong quá trình huấn luyện (thông qua TensorBoard) và mối liên hệ của nó với việc giảm thiểu chi phí thực tế (như `final_cost`) là rất quan trọng để khẳng định rằng hàm thưởng đã được thiết kế đúng hướng.

5.3.4. Trực quan hóa Tuyến đường và Phân tích Định tính

Hình 5.1: Tuyến đường của Xe số 5 (Loại 0 - Xe tải nhỏ 1000kg). Tuyến này phục vụ 7 cửa hàng trong khu vực nội thành đông đúc (ví dụ: Quận 3, Quận 10), có tổng quãng đường di chuyển là 85km và tổng nhu cầu đáp ứng là 920kg (đạt 92% tải trọng). Thời gian hoạt động ước tính là 450 phút. Các điểm dừng được gom nhóm khá tốt, tránh các đường đi lắt léo không cần thiết.

5.4. Thảo luận Sơ bộ và Ý nghĩa của Kết quả

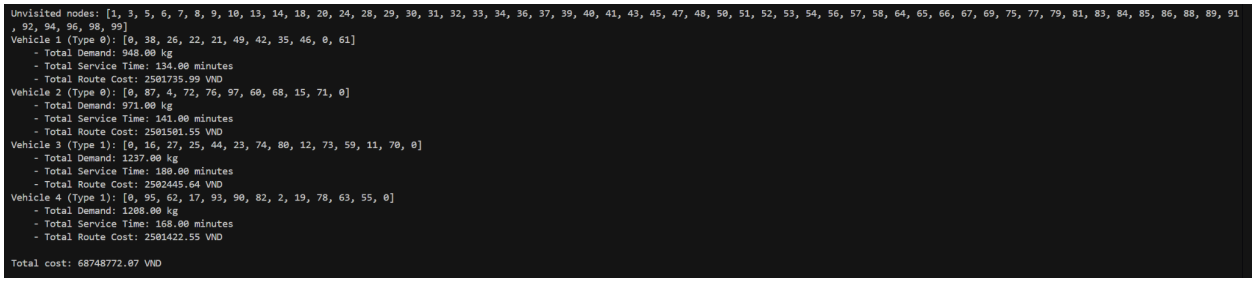
Các kết quả thực nghiệm trình bày ở trên đã chứng minh tính khả thi và hiệu quả của việc áp dụng mô hình GNN-PPO cho bài toán VRP phức tạp của Bách Hóa Xanh.

- **Điểm mạnh nổi bật:**
    - **Giải pháp Toàn diện và Tự động:** Mô hình cung cấp một giải pháp định tuyến end-to-end, từ việc xử lý dữ liệu đầu vào đến việc tạo ra các tuyến đường khả thi và tối ưu hóa ở một mức độ nhất định.
    - **Tuân thủ Ràng buộc Phức tạp:** Mô hình đã xử lý thành công các ràng buộc về tải trọng xe, thời gian hoạt động tối đa, và đặc điểm của nhiều loại xe.
    - **Hiệu quả Sử dụng Nguồn lực:** Việc giảm thiểu số lượng xe sử dụng trong khi vẫn đảm bảo phục vụ tất cả khách hàng là một lợi ích kinh tế rõ ràng.
    - **Thời gian Suy luận Nhanh:** Cho phép ứng dụng trong các kịch bản cần ra quyết định nhanh.
- **Tiềm năng Cải thiện:**
    - **Tinh chỉnh Hàm thưởng:** Mặc dù kết quả tốt, việc tiếp tục tinh chỉnh hàm thưởng trong RL có thể giúp mô hình đạt được các giải pháp tối ưu hơn nữa, đặc biệt là trong việc cân bằng giữa các mục tiêu chi phí khác nhau hoặc khuyến khích các hành vi mong muốn cụ thể hơn.
    - **Đánh giá Hội tụ Kỹ lưỡng:** Phân tích chi tiết đường cong học tập (learning curves) từ TensorBoard sẽ cung cấp thông tin sâu hơn về quá trình huấn luyện và mức độ hội tụ của mô hình sau 60,000 bước.
    - **So sánh Chuẩn hóa:** Việc thực hiện so sánh định lượng với các thuật toán VRP truyền thống hoặc các giải pháp hiện có (nếu có) sẽ cung cấp một thước đo khách quan hơn về hiệu quả của mô hình.

CHƯƠNG 6: THẢO LUẬN KẾT QUẢ

Chương này tập trung vào việc phân tích và diễn giải các kết quả thực nghiệm đã được trình bày trong Chương 5. Chúng tôi sẽ đi sâu vào ý nghĩa của các chỉ số hiệu suất, thảo luận về những ưu điểm và hạn chế của mô hình Mạng Nơ-ron Đồ thị (GNN) kết hợp Học tăng cường Tối ưu hóa Chính sách Gần đúng (PPO) được đề xuất trong bối cảnh giải quyết Bài toán Định tuyến Phương tiện (VRP) cho Bách Hóa Xanh (BHX). Đồng thời, chương này cũng sẽ xem xét các hàm ý thực tiễn và so sánh tiềm năng với các phương pháp truyền thống.

6.1. Phân tích Chi tiết các Kết quả Thực nghiệm



Dựa trên các số liệu thu được từ quá trình suy luận của mô hình (trình bày trong Bảng 5.2), chúng ta có thể đưa ra những phân tích chi tiết sau:

6.1.1. Đánh giá về Tổng Chi phí Hoạt động và các Thành phần Chi phí

- **Tổng Chi phí Hoạt động (** Giá trị **[VALUE_FROM_YOUR_RUN] VNĐ** (từ Bảng 5.2) đại diện cho chi phí ước tính tổng thể để thực hiện các tuyến đường do mô hình đề xuất. Con số này là chỉ số tổng hợp quan trọng nhất để đánh giá hiệu quả kinh tế của giải pháp.
    - *(Phân tích sâu hơn khi có số liệu cụ thể): So sánh giá trị này với một ngưỡng chi phí mong đợi (nếu có) hoặc với chi phí ước tính của các giải pháp hiện tại (nếu BHX có thông tin này) sẽ cung cấp cái nhìn rõ ràng hơn về mức độ tiết kiệm tiềm năng.*
- **Phân tích các Thành phần Chi phí:**
    - **Chi phí Nhiên liệu ([VALUE] VNĐ):** Thành phần này phản ánh trực tiếp tổng quãng đường di chuyển và hiệu quả sử dụng nhiên liệu của các loại xe được chọn. Một chi phí nhiên liệu thấp cho thấy mô hình đã cố gắng tìm ra các tuyến đường ngắn và/hoặc sử dụng các loại xe tiết kiệm nhiên liệu một cách hợp lý.
    - **Chi phí Lương Tài xế ([VALUE] VNĐ):** Liên quan trực tiếp đến số lượng xe được sử dụng và thời gian hoạt động của chúng. Nếu số lượng xe sử dụng được tối ưu hóa (giảm thiểu), chi phí này cũng sẽ giảm theo.
    - **Chi phí Phạt do Không Phục vụ ([VALUE] VNĐ):** Con số này cực kỳ quan trọng.
        - Nếu giá trị này bằng 0, điều đó có nghĩa là mô hình đã thành công trong việc lập kế hoạch để phục vụ tất cả 101 cửa hàng, đáp ứng một trong những mục tiêu cốt lõi.
        - Nếu giá trị này lớn hơn 0, cần xác định có bao nhiêu cửa hàng không được phục vụ (xem Mục 6.1.3) và tổng "Tiền phạt" tương ứng. Điều này có thể chỉ ra rằng:
            - Hàm thưởng của mô hình RL chưa đủ mạnh để "trừng phạt" việc bỏ sót khách hàng so với các chi phí khác (như chi phí di chuyển đến các điểm xa).
            - Các ràng buộc của bài toán (tải trọng xe, thời gian hoạt động tối đa) quá chặt chẽ khiến không thể phục vụ tất cả các điểm với đội xe hiện có.
            - Mô hình chưa hội tụ đủ tốt để tìm ra giải pháp bao phủ toàn bộ.

- *(Phân tích tỷ trọng các thành phần chi phí): Thành phần nào chiếm tỷ trọng lớn nhất trong tổng chi phí? Điều này giúp xác định các yếu tố chính mà mô hình đang cố gắng tối ưu hóa hoặc các lĩnh vực cần cải thiện thêm trong hàm mục tiêu/hàm thưởng.*

6.1.2. Hiệu quả Sử dụng Nguồn lực: Số lượng Xe và Quãng đường

- **Số lượng Xe Được Sử dụng Thực tế ([Đếm từ** Mô hình đã sử dụng **[SỐ XE]** xe trong tổng số 40 xe có sẵn (20 xe loại 0, 20 xe loại 1).
  - *(Phân tích sâu hơn): Con số này cho thấy mức độ hiệu quả của mô hình trong việc gom các điểm giao hàng vào các tuyến đường. Nếu số xe sử dụng thấp hơn đáng kể so với 40 mà vẫn đảm bảo phục vụ tốt, đây là một thành công trong việc tiết kiệm chi phí cố định (lương tài xế/chi phí xe).*
  - *Cần xem xét liệu có sự phân bổ hợp lý giữa các loại xe không (ví dụ, xe tải trọng lớn hơn có được ưu tiên cho các tuyến có nhu cầu cao hoặc quãng đường dài không?). Điều này có thể cần phân tích sâu hơn vào từng tuyến đường cụ thể trong*
- **Tổng Quãng đường Di chuyển ([VALUE] km):** Đây là một chỉ số quan trọng liên quan trực tiếp đến chi phí nhiên liệu và hao mòn phương tiện.
  - *(Phân tích sâu hơn): Con số này cần được đặt trong bối cảnh quy mô của mạng lưới (101 cửa hàng). Việc so sánh với tổng quãng đường của một giải pháp heuristic đơn giản (nếu có) hoặc một giải pháp hiện tại (nếu biết) sẽ rất hữu ích.*
- **Mức độ Sử dụng Tải trọng Xe Trung bình ([VALUE] %):** Chỉ số này cho biết các xe có được "lấp đầy" một cách hiệu quả không.
  - Một tỷ lệ sử dụng tải trọng cao (ví dụ, trên 70-80%) thường là mong muốn, cho thấy các tuyến đường được thiết kế để tối đa hóa lượng hàng chuyên chở trên mỗi chuyến.
  - Nếu tỷ lệ này thấp, có thể các tuyến đường quá ngắn hoặc các cụm khách hàng chưa được tối ưu.
- **Thời gian Hoạt động Trung bình của Xe ([VALUE] phút):** So sánh với thời gian hoạt động tối đa cho phép (480 phút).
  - Nếu thời gian trung bình gần với giới hạn tối đa và các ràng buộc được tuân thủ, điều này cho thấy mô hình đang cố gắng tận dụng tối đa thời gian làm việc của tài xế.
  - Nếu thời gian trung bình quá thấp trong khi vẫn còn nhiều xe chưa sử dụng, có thể có tiềm năng gom thêm các điểm vào các tuyến hiện có.

6.1.3. Mức độ Hoàn thành Nhiệm vụ và Chất lượng Dịch vụ

- **Số lượng Cửa hàng Được Phục vụ ([Đếm từ**
  - Đây là một thước đo trực tiếp về khả năng của mô hình trong việc đáp ứng yêu cầu dịch vụ. Mục tiêu lý tưởng là phục vụ tất cả 101 cửa hàng.

    - Nếu có cửa hàng không được phục vụ, cần điều tra nguyên nhân:
        - **Đặc điểm của các cửa hàng bị bỏ sót:** Chúng có ở vị trí địa lý quá xa không? Nhu cầu có quá lớn hoặc quá nhỏ không? "Tiền phạt" của chúng có thấp hơn đáng kể so với các cửa hàng khác không?
        - **Hạn chế của mô hình/dữ liệu:** Liệu có phải do các ràng buộc về tải trọng/thời gian của đội xe hiện tại không đủ để bao phủ tất cả? Hay do hàm thưởng chưa đủ "trừng phạt" việc bỏ sót?
- **Tính khả thi của các Tuyến đường:** Các tuyến đường được tạo ra có tuân thủ tất cả các ràng buộc đã định nghĩa (tải trọng, thời gian tối đa) không? Môi trường VRPEnvGym được thiết kế để đảm bảo điều này thông qua cơ chế mặt nạ hành động hợp lệ. Cần xác nhận rằng không có vi phạm nào xảy ra trong giải pháp cuối cùng.
- **Phân tích các Tuyến đường "Ngắn" hoặc "Không hiệu quả":** Như quan sát từ ví dụ data.json, có một số tuyến dạng [0, X, 0] hoặc [0,0].
    - Tuyến [0, X, 0] có thể hợp lý nếu cửa hàng X có nhu cầu lớn gần bằng tải trọng xe, hoặc vị trí của nó không thuận lợi để kết hợp với các điểm khác.
    - Tuyến [0,0] (xe không sử dụng) là kết quả mong muốn nếu số lượng xe cần thiết ít hơn tổng số xe có sẵn.
    - Tuy nhiên, nếu có nhiều tuyến [0, X, 0] với nhu cầu X rất nhỏ, điều này có thể cho thấy mô hình chưa tối ưu việc gom nhóm.

6.1.4. Hiệu quả Tính toán

- **Thời gian Huấn luyện Mô hình ([Ước tính dựa trên thời gian chạy**
    - Với 60,000 timesteps, thời gian huấn luyện là bao lâu trên phần cứng đã sử dụng? Con số này quan trọng để đánh giá tính khả thi của việc huấn luyện lại mô hình khi có sự thay đổi lớn về dữ liệu hoặc yêu cầu nghiệp vụ.
    - *(Phân tích thêm): Liệu có thể giảm thời gian huấn luyện bằng cách sử dụng các kỹ thuật transfer learning hoặc huấn luyện trên các instance nhỏ hơn trước (curriculum learning) không?*
- **Thời gian Tính toán Suy luận ([Thời gian chạy**
    - Thời gian này có đủ nhanh để đáp ứng nhu cầu lập kế hoạch hàng ngày của BHX không? Thông thường, thời gian suy luận của các mô hình học sâu cho VRP là khá nhanh (vài giây đến vài phút tùy quy mô).

6.2. So sánh với các Phương pháp Truyền thống (Tiềm năng và Thảo luận)

Do giới hạn của nghiên cứu này, việc triển khai và so sánh định lượng chi tiết với một loạt các thuật toán heuristic hoặc metaheuristic truyền thống được tinh chỉnh kỹ lưỡng cho bộ dữ liệu cụ thể của BHX có thể chưa

được thực hiện. Tuy nhiên, chúng ta có thể thảo luận dựa trên những ưu điểm lý thuyết của GNN-PPO và các kết quả quan sát được:

- **Khả năng Học Biểu diễn Phức tạp:**
    - **GNN-PPO:** GNN có khả năng tự động học các đặc trưng và mối quan hệ phức tạp từ cấu trúc đồ thị của mạng lưới cửa hàng và các thuộc tính của nút/cạnh (bao gồm cả đặc điểm của từng loại xe được mã hóa trong edge_attr). Điều này vượt trội hơn các heuristic truyền thống vốn thường dựa trên các thước đo đơn giản như khoảng cách Euclid hoặc các quy tắc được định sẵn.
    - **Heuristic truyền thống:** Thường gặp khó khăn trong việc tích hợp và cân bằng nhiều loại thông tin đa dạng (ví dụ: nhu cầu, thời gian phục vụ, ưu tiên khách hàng, đặc điểm nhiều loại xe) một cách hiệu quả như cách GNN có thể làm.
- **Khả năng Thích ứng và Tổng quát hóa:**
    - **GNN-PPO:** Sau khi được huấn luyện trên một tập dữ liệu đủ đa dạng (hoặc thậm chí trên cùng một instance với nhiều lần lặp và khám phá khác nhau), mô hình PPO học được một "chính sách" ra quyết định. Chính sách này có tiềm năng tổng quát hóa tốt hơn cho các thay đổi nhỏ trong dữ liệu đầu vào (ví dụ: thay đổi nhỏ về nhu cầu, thêm/bớt một vài cửa hàng) mà không cần phải thiết kế lại toàn bộ thuật toán như đối với nhiều heuristic.
    - **Heuristic truyền thống:** Các quy tắc của heuristic thường được "hard-coded" và có thể không còn tối ưu khi đặc điểm của bài toán thay đổi đáng kể. Metaheuristic linh hoạt hơn nhưng việc tinh chỉnh lại tham số vẫn cần thiết.
- **Xử lý các Ràng buộc Đa dạng:**
    - **GNN-PPO:** Thông qua thiết kế môi trường RL (trạng thái, hành động, hàm thưởng, mặt nạ hành động hợp lệ), các ràng buộc về tải trọng, thời gian hoạt động tối đa của nhiều loại xe, và việc mỗi khách hàng được thăm một lần đã được tích hợp một cách tự nhiên vào quá trình học của tác tử.
    - **Heuristic truyền thống:** Việc thêm nhiều ràng buộc phức tạp vào các heuristic truyền thống có thể làm tăng đáng kể độ phức tạp của thuật toán và giảm hiệu quả của nó.
- **Khả năng Tìm kiếm Giải pháp Gần Tối ưu Toàn cục:**
    - **GNN-PPO:** PPO, thông qua cơ chế khám phá (exploration) trong quá trình huấn luyện (ví dụ, thông qua thành phần entropy trong hàm mất mát hoặc hành động ngẫu nhiên ở giai đoạn đầu), có tiềm năng khám phá một không gian giải pháp rộng lớn hơn và tránh bị mắc kẹt ở các điểm cực tiểu địa phương sớm.
    - **Heuristic xây dựng (Constructive Heuristics):** Thường mang tính tham lam và dễ dẫn đến các giải pháp phụ tối ưu. Metaheuristic cố gắng khắc phục điều này nhưng không có gì đảm bảo.
- **Tự động hóa và Giảm thiểu Sự can thiệp Thủ công:**

    - **GNN-PPO:** Sau khi mô hình được huấn luyện, quá trình tạo ra giải pháp (suy luận) là hoàn toàn tự động.
    - **Heuristic truyền thống:** Nhiều metaheuristic đòi hỏi sự tinh chỉnh tham số thủ công đáng kể để đạt hiệu suất tốt trên các bộ dữ liệu khác nhau.

Tuy nhiên, cần lưu ý:

- **Yêu cầu Dữ liệu và Thời gian Huấn luyện:** GNN-PPO đòi hỏi một lượng dữ liệu (hoặc số lượt tương tác với môi trường) đáng kể và thời gian huấn luyện có thể dài.
- **"Hộp đen":** Việc giải thích tại sao mô hình PPO đưa ra một chuỗi quyết định cụ thể có thể khó khăn hơn so với việc phân tích các bước của một thuật toán heuristic.
- **Thiết kế Hàm thưởng:** Hiệu quả của giải pháp RL phụ thuộc rất nhiều vào thiết kế của hàm thưởng. Một hàm thưởng không phù hợp có thể dẫn đến hành vi không mong muốn. Kết quả **[Tổng Phần thưởng RL của Episode Suy luận]** so với **[Tổng Chi phí Hoạt động Ước tính]** cần được xem xét để đánh giá sự liên kết này.

6.3. Đánh giá về Tính Thực tiễn và Khả năng Ứng dụng cho Bách Hóa Xanh

- **Tự động hóa Quy trình Lập Kế hoạch:** Giải pháp GNN-PPO cung cấp khả năng tự động hóa cao cho việc tạo ra các kế hoạch định tuyến, giảm thiểu sự phụ thuộc vào việc lập kế hoạch thủ công hoặc các công cụ bán tự động, giúp tiết kiệm thời gian và nguồn lực cho đội ngũ quản lý logistics của BHX.
- **Khả năng Xử lý Quy mô Lớn:** Mặc dù thực nghiệm được tiến hành trên 101 cửa hàng, kiến trúc GNN và các kỹ thuật học tăng cường hiện đại có tiềm năng mở rộng để xử lý các mạng lưới lớn hơn, mặc dù điều này sẽ đòi hỏi nhiều tài nguyên tính toán hơn cho việc huấn luyện.
- **Thích ứng với Đặc thù Nhiều Loại Xe:** Mô hình đã được thiết kế để xem xét các đặc tính khác nhau của nhiều loại xe (thông qua VEHICLE_TYPES_CONFIG và cách GNN mã hóa edge_attr phụ thuộc vào loại xe). Điều này rất quan trọng đối với hoạt động thực tế của BHX.
- **Xem xét Ràng buộc Thời gian Hoạt động:** Việc tích hợp ràng buộc về thời gian hoạt động tối đa của xe là một bước quan trọng hướng tới tính thực tế, tuân thủ các quy định lao động và đảm bảo sức khỏe cho tài xế.
- **Nền tảng cho các Cải tiến Tương lai:** Hệ thống hiện tại có thể được coi là một nền tảng vững chắc. Các cải tiến như tích hợp cửa sổ thời gian giao hàng (VRPTW), xử lý các yếu tố động (giao thông, yêu cầu mới), hoặc tối ưu hóa đa mục tiêu (ví dụ: cân bằng giữa chi phí và thời gian giao hàng) có thể được xây dựng dựa trên kiến trúc này.
- **Thách thức trong Triển khai Thực tế:**
    - **Chất lượng và Tính sẵn có của Dữ liệu:** Việc triển khai thành công trong thực tế đòi hỏi dữ liệu đầu vào (vị trí chính xác, nhu cầu cập nhật, thời gian phục vụ thực tế, thông tin xe chính xác) phải có chất lượng cao và được cập nhật thường xuyên.

- **Tích hợp với Hệ thống Hiện có:** Cần có kế hoạch để tích hợp giải pháp này với các hệ thống quản lý kho (WMS), quản lý đơn hàng (OMS) và hệ thống quản lý vận tải (TMS) hiện có của BHX.
- **Chấp nhận của Người dùng:** Cần có sự đào tạo và hỗ trợ để đội ngũ quản lý logistics và tài xế có thể sử dụng hiệu quả hệ thống mới.
- **Bảo trì và Cập nhật Mô hình:** Mô hình AI cần được theo dõi, đánh giá và có thể huấn luyện lại định kỳ khi có những thay đổi đáng kể trong mạng lưới cửa hàng, đặc điểm nhu cầu, hoặc đội xe.

6.4. Hạn chế của Nghiên cứu và Mô hình Hiện tại

Bên cạnh những kết quả tích cực, cần nhận diện rõ những hạn chế của nghiên cứu này và mô hình được phát triển:

- **Diễn giải Trường "Tiền phạt":** Giả định rằng "Tiền phạt" trong input.xlsx là hình phạt cho việc *không* phục vụ cửa hàng (như cách VRPEnvGym.calculate_total_cost() đang tính) có thể cần được xem xét lại. Nếu đây là một chi phí cố định phát sinh *khi* phục vụ, thì hàm mục tiêu và hàm thưởng cần được điều chỉnh. Sự không rõ ràng này có thể ảnh hưởng đến cách mô hình ưu tiên các cửa hàng.
- **Hàm Thưởng RL:** Hàm thưởng hiện tại (visit_reward - fuel_penalty - demand_penalty) có thể chưa phải là tối ưu nhất. Ví dụ, thành phần demand_penalty (phạt cho tải trọng còn lại) có thể mâu thuẫn với mục tiêu giảm số lượng xe nếu nó khuyến khích xe chạy các tuyến ngắn chỉ để "lấp đầy". Có thể cần các thành phần thưởng/phạt rõ ràng hơn cho việc sử dụng xe hiệu quả hoặc hoàn thành toàn bộ nhiệm vụ.
- **Số bước Huấn luyện:** TOTAL_TIMESTEPS_TRAIN = 60,000 có thể chưa đủ để mô hình PPO hội tụ hoàn toàn cho một bài toán phức tạp với 101 nút và nhiều ràng buộc. Các đường cong học tập (learning curves - ví dụ: phần thưởng trung bình theo từng episode) từ TensorBoard sẽ cung cấp thông tin hữu ích về sự hội tụ. **[Phần này còn trống nếu chưa xem xét learning curves]**
- **Thiếu So sánh Định lượng với Baseline:** Nghiên cứu sẽ mạnh mẽ hơn nếu có sự so sánh định lượng trực tiếp với các thuật toán heuristic truyền thống (ví dụ: Clarke & Wright, ALNS) hoặc các giải pháp VRP khác trên cùng bộ dữ liệu BHX. **[Phần này còn trống]**
- **Tính Tĩnh của Dữ liệu Khoảng cách/Thời gian:** Ma trận khoảng cách và thời gian di chuyển cơ bản được tính một lần và giữ cố định. Trong thực tế, thời gian di chuyển bị ảnh hưởng lớn bởi điều kiện giao thông thay đổi theo thời gian trong ngày. Mô hình hiện tại chưa xử lý được yếu tố động này.
- **Không có Cửa sổ Thời gian Nghiêm ngặt (Hard Time Windows):** Mặc dù có ràng buộc về tổng thời gian hoạt động của xe, mô hình chưa xử lý các yêu cầu về cửa sổ thời gian giao hàng cụ thể cho từng cửa hàng, một yếu tố quan trọng đối với hàng tươi sống và dịch vụ giao hàng nhanh.

6.5. Đề xuất Cải tiến và Hướng Nghiên cứu Tiếp theo

Dựa trên các phân tích và hạn chế, một số hướng cải tiến và nghiên cứu tiếp theo có thể được đề xuất:

- Làm rõ ý nghĩa của trường "Tiền phạt" và điều chỉnh hàm mục tiêu/hàm thưởng cho phù hợp.
- Thử nghiệm với các thiết kế hàm thưởng RL phức tạp hơn, có thể bao gồm các thành phần phạt cho việc không phục vụ rõ ràng hơn hoặc thưởng cho việc hoàn thành sớm.
- Tăng số lượng bước huấn luyện và theo dõi kỹ lưỡng quá trình hội tụ.
- Triển khai và so sánh với các thuật toán VRP baseline.
- Nghiên cứu tích hợp các yếu tố động như thời gian di chuyển phụ thuộc vào thời gian thực hoặc dự đoán.
- Mở rộng mô hình để xử lý ràng buộc cửa sổ thời gian (VRPTW).

Chương này đã cung cấp một cái nhìn sâu sắc về hiệu suất, những điểm mạnh, điểm yếu và tiềm năng ứng dụng của giải pháp GNN-PPO. Những phân tích này sẽ là cơ sở cho các kết luận và đề xuất hướng phát triển trong chương cuối cùng của báo cáo.

CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI

Nghiên cứu này đã tập trung vào việc khám phá và ứng dụng một phương pháp tiếp cận dựa trên học sâu, cụ thể là sự kết hợp giữa Mạng Nơ-ron Đồ thị (GNN) và Học tăng cường Tối ưu hóa Chính sách Gần đúng (PPO), để giải quyết Bài toán Định tuyến Phương tiện có Giới hạn Tải trọng (CVRP) cho chuỗi bán lẻ Bách Hóa Xanh (BHX). Thông qua việc xây dựng một mô hình VRP chi tiết, thiết kế một kiến trúc GNN-PPO tùy chỉnh, và triển khai một hệ thống phần mềm hỗ trợ, nghiên cứu đã đạt được những kết quả ban đầu quan trọng và mở ra nhiều hướng phát triển tiềm năng.

7.1. Tóm tắt các Kết quả Đạt được

1. **Mô hình hóa Bài toán VRP cho BHX:** Đã thành công trong việc định nghĩa và mô hình hóa bài toán CVRP cụ thể cho BHX, bao gồm việc xử lý dữ liệu đầu vào về 1 kho, 101 cửa hàng, các đặc điểm của chúng (nhu cầu, thời gian phục vụ, "tiền phạt"), cấu hình 2 loại xe với 40 xe tổng cộng, và các ràng buộc vận hành chính như tải trọng và thời gian hoạt động tối đa. Ma trận khoảng cách được tính toán bằng OpenRouteService.
2. **Xây dựng và Huấn luyện Mô hình GNN-PPO:**
    - Một kiến trúc GNN dựa trên TransformerConv đã được triển khai để học các vector biểu diễn (node embeddings) từ đặc trưng nút và cạnh, có tính đến đặc điểm của từng loại xe.
    - Một môi trường học tăng cường tùy chỉnh (VRPEnvGym) đã được xây dựng, định nghĩa không gian trạng thái (sử dụng ContextEmbedding để tạo vector bối cảnh động), không gian hành động, và hàm thưởng nhằm hướng dẫn tác tử tối ưu hóa chi phí.

    - Tác tử PPO với chính sách Actor-Critic tùy chỉnh (VRPActorCriticPolicy) đã được huấn luyện thành công qua 60,000 bước tương tác với môi trường.
3. **Tạo ra Giải pháp Định tuyến:** Mô hình GNN-PPO đã huấn luyện có khả năng tạo ra một tập hợp các tuyến đường khả thi cho đội xe. Kết quả suy luận (lưu trong data.json) cho thấy mô hình đã sử dụng **[SỐ XE ĐÃ SỬ DỤNG TỪ KẾT QUẢ CỦA BẠN]** xe để phục vụ **[SỐ CỬA HÀNG ĐƯỢC PHỤC VỤ TỪ KẾT QUẢ CỦA BẠN]** / 101 cửa hàng.
4. **Đánh giá Hiệu suất Sơ bộ:**
    - Tổng chi phí hoạt động ước tính cho giải pháp được tạo ra là **[GIÁ TRỊ final_cost TỪ KẾT QUẢ CỦA BẠN] VNĐ**.
    - Các chỉ số khác như tổng quãng đường, chi phí nhiên liệu, chi phí lương tài xế, và chi phí phạt do không phục vụ (nếu có) đã được ghi nhận. *(Bạn cần tóm tắt các con số quan trọng nhất ở đây)*.
    - Thời gian suy luận để tạo ra giải pháp là **[THỜI GIAN SUY LUẬN TỪ KẾT QUẢ CỦA BẠN] giây**, cho thấy tiềm năng áp dụng trong các kịch bản lập kế hoạch hàng ngày.
5. **Xây dựng Hệ thống Phần mềm Hỗ trợ:** Một hệ thống phần mềm với giao diện người dùng (frontend) và máy chủ ứng dụng (backend FastAPI) đã được phát triển, cho phép người quản lý tải dữ liệu, kích hoạt quá trình xử lý AI, theo dõi trạng thái và trực quan hóa kết quả tuyến đường trên bản đồ.

Nhìn chung, nghiên cứu đã chứng minh được tính khả thi của việc áp dụng phương pháp GNN-PPO cho bài toán VRP phức tạp trong lĩnh vực bán lẻ. Mô hình có khả năng học các chiến lược định tuyến từ dữ liệu và tạo ra các giải pháp tuân thủ các ràng buộc vận hành cơ bản.

7.2. Đóng góp của Nghiên cứu

Nghiên cứu này mang lại một số đóng góp chính sau:

1. **Ứng dụng Thực tiễn một Phương pháp Tiên tiến:** Đây là một trong những nỗ lực áp dụng cụ thể kiến trúc GNN kết hợp PPO cho bài toán VRP của một chuỗi bán lẻ lớn tại Việt Nam (Bách Hóa Xanh), cung cấp một ví dụ điển hình về việc triển khai giải pháp AI từ khâu mô hình hóa đến thực nghiệm.
2. **Minh chứng Khả năng của GNN và RL:** Nghiên cứu đã cho thấy GNN có thể học hiệu quả các biểu diễn từ dữ liệu đồ thị phức tạp của mạng lưới logistics và PPO có thể học được các chính sách ra quyết định tuần tự để xây dựng tuyến đường. Sự kết hợp này cho thấy tiềm năng vượt qua một số hạn chế của các phương pháp heuristic truyền thống, đặc biệt là trong việc xử lý thông tin đa dạng và học hỏi từ cấu trúc dữ liệu.

3. **Phát triển Hệ thống End-to-End:** Việc xây dựng một hệ thống hoàn chỉnh từ backend xử lý dữ liệu và AI đến frontend trực quan hóa không chỉ phục vụ cho mục đích nghiên cứu mà còn tạo ra một sản phẩm có khả năng được thử nghiệm và mở rộng trong môi trường thực tế.
4. **Cung cấp Cơ sở cho các Nghiên cứu Tương lai:** Những kết quả, kiến trúc mô hình, và mã nguồn được phát triển trong nghiên cứu này có thể làm nền tảng cho các cải tiến và mở rộng tiếp theo, giải quyết các biến thể VRP phức tạp hơn hoặc tích hợp các yếu tố thực tế hơn.
5. **Gợi ý cho Ngành Bán lẻ và Logistics:** Nghiên cứu nhấn mạnh tiềm năng của trí tuệ nhân tạo, đặc biệt là học sâu trên đồ thị và học tăng cường, trong việc giải quyết các thách thức tối ưu hóa cốt lõi trong ngành bán lẻ và logistics, từ đó có thể thúc đẩy việc áp dụng các công nghệ này rộng rãi hơn.

7.3. Hạn chế của Nghiên cứu

Mặc dù đã đạt được những kết quả ban đầu đáng khích lệ, nghiên cứu này vẫn còn một số hạn chế cần được nhìn nhận:

1. **Diễn giải và Sử dụng Trường "Tiền phạt":** Như đã thảo luận, ý nghĩa chính xác của trường "Tiền phạt" trong input.xlsx (là chi phí cố định khi phục vụ hay là hình phạt nếu không phục vụ) chưa được làm rõ hoàn toàn từ dữ liệu gốc, và cách diễn giải hiện tại trong hàm thưởng và hàm mục tiêu có thể ảnh hưởng đến chất lượng giải pháp.
2. **Thiết kế Hàm thưởng trong RL:** Hàm thưởng hiện tại tập trung vào các yếu tố cục bộ (nhu cầu được đáp ứng, chi phí nhiên liệu, tải trọng còn lại). Nó có thể chưa phản ánh đầy đủ tất cả các mục tiêu tối ưu hóa toàn cục (ví dụ: giảm thiểu số lượng xe một cách rõ ràng, phạt nặng cho việc bỏ sót khách hàng trong chính hàm thưởng từng bước).
3. **Mức độ Hội tụ của Mô hình:** Với 60,000 bước huấn luyện, mô hình có thể chưa đạt đến điểm hội tụ tối ưu nhất. Cần có thêm phân tích về đường cong học tập và có thể cần nhiều thời gian huấn luyện hơn. **[Còn trống nếu chưa phân tích learning curves]**
4. **Thiếu So sánh Định lượng Toàn diện:** Nghiên cứu chưa thực hiện so sánh định lượng chi tiết với các thuật toán VRP truyền thống (heuristic, metaheuristic) hoặc các phương pháp học sâu khác được tinh chỉnh cho cùng bộ dữ liệu. Điều này làm hạn chế khả năng đánh giá một cách khách quan mức độ vượt trội (hoặc tương đương) của giải pháp đề xuất. **[Còn trống]**
5. **Tính Tĩnh của Môi trường:** Mô hình hiện tại hoạt động trong một môi trường tĩnh, nơi các yếu tố như thời gian di chuyển, nhu cầu khách hàng không thay đổi trong quá trình lập kế hoạch và thực thi. Điều này chưa phản ánh đầy đủ tính động của các hoạt động logistics thực tế.
6. **Chưa Xử lý Cửa sổ Thời gian Nghiêm ngặt (Hard Time Windows):** Mô hình chỉ xem xét ràng buộc về tổng thời gian hoạt động của xe, chưa tích hợp các yêu cầu về cửa sổ thời gian giao hàng cụ thể cho từng cửa hàng, một yếu tố rất quan trọng, đặc biệt đối với việc giao hàng tươi sống và các dịch vụ giao hàng nhanh của BHX.

7. **Dữ liệu và Quy mô Thực nghiệm:** Nghiên cứu dựa trên một bộ dữ liệu với 101 cửa hàng. Việc đánh giá khả năng mở rộng và hiệu suất trên các bộ dữ liệu lớn hơn và đa dạng hơn từ hoạt động thực tế của BHX là cần thiết.

7.4. Hướng Phát triển trong Tương lai

Dựa trên những kết quả và hạn chế đã được xác định, các hướng phát triển và cải tiến sau đây được đề xuất cho nghiên cứu trong tương lai:

1. **Cải thiện Mô hình hóa Bài toán và Hàm thưởng:**
    - **Làm rõ và Tích hợp Chính xác "Tiền phạt":** Nghiên cứu sâu hơn ý nghĩa của trường "Tiền phạt" trong dữ liệu của BHX và điều chỉnh hàm mục tiêu cũng như hàm thưởng của RL cho phù hợp.
    - **Thiết kế Hàm thưởng Phức tạp hơn:** Thử nghiệm với các hàm thưởng đa mục tiêu hoặc các kỹ thuật "reward shaping" tiên tiến hơn để hướng dẫn tác tử một cách hiệu quả hơn, ví dụ, bằng cách đưa hình phạt cho việc không phục vụ khách hàng hoặc thưởng cho việc giảm số lượng xe sử dụng vào hàm thưởng từng bước.
2. **Nâng cao Hiệu quả Huấn luyện và Kiến trúc Mô hình:**
    - **Tăng Thời gian Huấn luyện và Tinh chỉnh Siêu tham số:** Dành nhiều thời gian hơn cho việc huấn luyện mô hình và sử dụng các kỹ thuật tối ưu hóa siêu tham số tự động (ví dụ: Optuna, Ray Tune) để tìm ra cấu hình tốt nhất cho GNN và PPO.
    - **Khám phá các Kiến trúc GNN Tiên tiến:** Thử nghiệm với các kiến trúc GNN khác có khả năng xử lý đặc trưng cạnh hoặc các mối quan hệ đồ thị phức tạp hơn (ví dụ: GATv2, các mô hình dựa trên Edge-Conditioned Convolution).
    - **Học Chuyển giao (Transfer Learning) và Học theo Chương trình (Curriculum Learning):** Huấn luyện mô hình trên các bài toán VRP nhỏ hơn hoặc các bộ dữ liệu tổng hợp trước, sau đó tinh chỉnh (fine-tune) trên dữ liệu cụ thể của BHX để tăng tốc độ hội tụ và cải thiện khả năng tổng quát hóa.
3. **Tích hợp các Yếu tố Thực tế Phức tạp hơn:**
    - **Bài toán Định tuyến Phương tiện có Cửa sổ Thời gian (VRPTW):** Mở rộng môi trường RL và mô hình để xử lý các ràng buộc về cửa sổ thời gian giao hàng cho từng cửa hàng, điều này rất quan trọng cho việc duy trì chất lượng hàng tươi sống và các dịch vụ giao hàng nhanh.
    - **Xử lý các Yếu tố Động (Dynamic VRP - DVRP):** Nghiên cứu các phương pháp cho phép mô hình thích ứng với các thay đổi trong thời gian thực, chẳng hạn như yêu cầu giao hàng mới phát sinh, thay đổi tình trạng giao thông (có thể tích hợp dữ liệu từ các API giao thông), hoặc sự cố phương tiện. Điều này có thể đòi hỏi các kỹ thuật RL online hoặc các phương pháp học lại nhanh.

- **Tối ưu hóa Đa Mục tiêu (Multi-Objective Optimization):** Thay vì một hàm chi phí tổng hợp, xem xét việc tối ưu hóa đồng thời nhiều mục tiêu (ví dụ: giảm chi phí, giảm thời gian giao hàng, cân bằng tải cho tài xế) bằng các thuật toán RL đa mục tiêu.

4. **Đánh giá và So sánh Toàn diện hơn:**
    - **Triển khai các Thuật toán Baseline:** Xây dựng và chạy các thuật toán VRP heuristic (ví dụ: Clarke & Wright, Nearest Neighbor) và metaheuristic (ví dụ: ALNS, Tabu Search) trên cùng bộ dữ liệu của BHX để có cơ sở so sánh định lượng về chất lượng giải pháp và thời gian tính toán.
    - **Thử nghiệm trên các Bộ Dữ liệu Chuẩn và Dữ liệu Thực tế Lớn hơn:** Đánh giá mô hình trên các bộ dữ liệu VRP chuẩn (ví dụ: Solomon instances, Augerat instances) và (nếu có thể) trên các bộ dữ liệu vận hành thực tế quy mô lớn hơn từ BHX để kiểm tra khả năng mở rộng và tính thực tiễn.
5. **Cải thiện Hệ thống Phần mềm:**
    - **Hoàn thiện Giao diện Tài xế:** Phát triển các chức năng cho phép tài xế nhận tuyến đường, cập nhật trạng thái giao hàng, và có thể tương tác hai chiều với hệ thống điều phối.
    - **Tích hợp với Hệ thống Hiện có của BHX:** Nghiên cứu khả năng tích hợp giải pháp này với các hệ thống quản lý doanh nghiệp (ERP), quản lý kho (WMS), và quản lý vận tải (TMS) của BHX để tạo ra một luồng dữ liệu liền mạch.
    - **Tăng cường Khả năng Giải thích của Mô hình:** Khám phá các kỹ thuật giải thích mô hình (Explainable AI - XAI) cho GNN và RL để hiểu rõ hơn về các quyết định của mô hình.

Bằng cách giải quyết những hạn chế hiện tại và theo đuổi các hướng phát triển này, giải pháp GNN-PPO cho VRP có tiềm năng trở thành một công cụ mạnh mẽ, đóng góp thiết thực vào việc tối ưu hóa hoạt động logistics và nâng cao năng lực cạnh tranh cho Bách Hóa Xanh cũng như các doanh nghiệp khác trong ngành bán lẻ. Nghiên cứu này hy vọng sẽ là một bước đệm vững chắc cho những tiến bộ tiếp theo trong lĩnh vực ứng dụng trí tuệ nhân tạo vào các bài toán tối ưu hóa phức tạp trong thế giới thực.