

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

**Hệ thống đèn giao thông thông minh ứng dụng
phương pháp học tăng cường**

NGUYỄN TRUNG KIÊN

kien.nt173209@sis.hust.edu.vn

Ngành: Công nghệ thông tin

Chuyên ngành: Công nghệ thông tin

Giảng viên hướng dẫn: PGS. TS. Thân Quang Khoát

Chữ ký GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 08/2022

LỜI CẢM ƠN

Lời đầu tiên, em xin được dành lời cảm ơn đến toàn thể các giảng viên đang công tác tại trường Đại Học Bách Khoa Hà Nội, và đặc biệt là các thầy cô đã tận tình hướng dẫn và chỉ bảo em trong suốt 5 năm học tập tại trường. Em tin chắc rằng những kiến thức mình đã được chỉ dạy sẽ trở thành hành trang vững chắc cho em trong sự nghiệp sau này.

Đặc biệt trong đồ án này, em muốn gửi lời cảm ơn chân thành tới PGS. TS. Thân Quang Khoát, người đã hướng dẫn và theo sát em suốt 4 tháng làm việc và thực hiện đồ án. Em đã học hỏi được rất nhiều kiến thức cũng như kinh nghiệm, cùng với đó là cách làm việc và phân chia công việc một cách hợp lý. Những kinh nghiệm và kiến thức quý báu này sẽ là bước đệm vững chắc trong những dự án tiếp theo mà em tham gia thực hiện.

Em cũng muốn cảm ơn những người bạn cùng lớp CNTT-10, khoá K62 đã đồng hành cùng em trong suốt những năm học tập tại trường Đại Học Bách Khoa Hà Nội. Các bạn chắc chắn đã góp phần làm quãng thời gian sinh viên trở nên vô cùng quý báu.

Cuối cùng, em xin được cảm ơn gia đình, những người đã luôn luôn cố gắng hết sức để hỗ trợ em, tạo điều kiện tốt nhất để em có thể có quãng thời gian học tập hiệu quả, luôn ở bên ủng hộ em những thời điểm tuyệt vời cũng như khó khăn.

Em xin chân thành cảm ơn tất cả mọi người!

TÓM TẮT NỘI DUNG ĐỒ ÁN

Giao thông là thứ mà tất cả chúng ta tham gia và tương tác mỗi ngày. Điều đó dẫn đến sự quan trọng của việc tối ưu hóa hệ thống giao thông. Tuy nhiên, trong thực tế tại Việt Nam nói riêng và trên toàn thế giới nói riêng, ta có thể thấy hệ thống giao thông vẫn tồn tại một số khuyết điểm có thể được cải thiện. Trong đó, trong phạm vi đồ án này, em chọn một vấn đề liên quan đến hệ thống đèn giao thông.

Mỗi chúng ta tham gia giao thông đôi khi đều có thể rơi vào tình huống đứng chờ đèn đỏ một thời gian vô cùng lâu, trong đó đôi khi có những trường hợp mà ta có thể thấy các làn xe khác đang rất thoáng và đèn giao thông nên thay đổi để phù hợp hơn với tình trạng hiện tại. Sự không tối ưu này dẫn đến việc lãng phí thời gian của người tham gia giao thông, đồng thời cũng có những tác động tiêu cực đến cảm xúc cũng như sức khoẻ của họ trong những điều kiện thời tiết cực đoan.

Hiện tại, hệ thống đèn giao thông đang hoạt động theo cơ chế thay đổi tuần tự theo các mức thời gian định trước. Ví dụ ở một cụm đèn sẽ có 30 giây đèn xanh, 30 giây đèn đỏ. Do đó ta có thể thấy rằng nó hoàn toàn không phản ánh được lưu lượng giao thông thực tế tại chính thời điểm đó, từ đó dẫn tới sự không tối ưu.

Giải pháp mà em đề xuất trong phạm vi đồ án tốt nghiệp này là sử dụng một mô hình học tăng cường, cùng với việc giả định có thông tin các cảm biến trên lòng đường để tính toán lưu lượng giao thông, từ đó đưa ra một quyết định thay đổi đèn hợp lý tuỳ thuộc vào tình hình giao thông trong thời gian thực. Hướng tiếp cận này khắc phục được điểm yếu chính của giải pháp hiện tại, đó là mang tính chất thời gian thực (real-time) và các lựa chọn đều được tối ưu cho một hàm mục tiêu cụ thể (ở đây là giảm thiểu tổng thời gian dừng lại của toàn bộ phương tiện tham gia trong môi trường).

Sau quá trình thực hiện, em đã xây dựng một mô trường mô phỏng tương đối hoàn thiện, thể hiện tốt một số tính chất quan trọng trong mô trường thực tế, và huấn luyện được một mô hình học tăng cường có khả năng tối ưu hóa hệ thống đèn giao thông. Mô hình này cũng đạt hiệu năng gấp 2 lần so với hệ thống không tối ưu trước đó.

Với kết quả này, em hi vọng đồ án của mình sẽ phần nào góp phần xây dựng những giải pháp tối ưu hóa trong thực tế. Từ đó mọi người có thể tham gia giao thông một cách nhanh chóng, an toàn, thoải mái hơn.

ABSTRACT

Traffic is something that we encounter everyday; therefore, optimizing the traffic system is crucial. However, we can certainly notice that there are some part of the traffic system that are not fully optimized. In the scope of this thesis, I propose a solution to improve our current traffic light system.

Sometimes when we are in the traffic, we can notice that the traffic light signal decision is not reasonable. For example, when the other lanes are free but we still have to wait for the red light in our lane to get turned off to start moving. This is due to the system not taking into account the real-time traffic flow, the usual behavior is just changing the light signal after a pre-defined amount of time. This can lead to huge waste of time, and also affect our health and mood.

My proposed solution in the scope of this thesis is using reinforcement learning, with the assumption that there are sensors on the street to measure the traffic flow. From there, the reinforcement learning agent can use the information to make decision about the traffic light signal. This solution resolve the current problem of not being real-time and optimize for a certain objective function which is the total stopping time of all vehicles in the environment.

Throughout the course of the thesis, I managed to build a fairly complete environment with current, which represent important properties of real-world environment. Also, I develop and trained a reinforcement learning model that are capable of making optimized decision. The trained model out-perform the current unoptimized behavior by 2 times.

I hope that my thesis and the result I managed to achieve will play a part in future optimizing solution that help to solve the problem in real-world. From that, everyone can join traffic with fast, safe, and please manner.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Các giải pháp hiện tại và hạn chế	1
1.3 Mục tiêu và định hướng giải pháp	2
1.4 Đóng góp của đồ án	2
1.5 Bố cục đồ án	2
CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT	4
2.1 Tổng quan	4
2.2 Ngữ cảnh của bài toán.....	4
2.3 Hệ thống đèn giao thông hiện tại.....	4
2.4 Các kết quả nghiên cứu tương tự	4
2.5 Học tăng cường	5
2.5.1 Mô hình hoá bài toán.....	6
2.5.2 Policy	7
2.5.3 Học tăng cường dựa vào mô hình.....	7
2.5.4 Học tăng cường theo giá trị	8
2.5.5 Policy Gradient.....	12
2.6 Unity3D.....	18
2.6.1 Engine trò chơi của Unity3D	18
2.6.2 Một số cơ chế thường dùng trong Unity.....	23
2.7 Unity ML-Agents	25
2.8 Kết chương.....	27
CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT.....	28
3.1 Tổng quan	28

3.2 Tổng quan giải pháp.....	28
3.3 Xây dựng ngữ cảnh môi trường trong Unity.....	28
3.3.1 Các gói hỗ trợ	28
3.3.2 Nền của môi trường.....	30
3.3.3 Hệ thống phương tiện	30
3.3.4 Hệ thống đèn giao thông.....	33
3.4 Cấu hình Markov Decision Process cho môi trường	34
3.4.1 Tổng quan	34
3.4.2 Thiết kế từng tập của môi trường	34
3.4.3 Thiết kế thông tin trạng thái của môi trường	35
3.4.4 Thiết kế hành động của agent	35
3.4.5 Thiết kế hệ thống điểm thưởng cho từng hành động.....	36
3.5 Tạo cơ chế hành động cơ sở của bài toán.....	36
3.6 Huấn luyện mô hình giải bài toán	37
3.6.1 Tổng quan	37
3.6.2 Quá trình tích luỹ thông tin	37
3.6.3 Huấn luyện mô hình dựa trên dữ liệu đã thu được.....	38
3.7 Kết chương.....	39
CHƯƠNG 4. ĐÁNH GIÁ THỰC NGHIỆM.....	40
4.1 Tổng quan	40
4.2 Mục tiêu đánh giá	40
4.3 Các độ đo đánh giá	40
4.4 Phương pháp thí nghiệm.....	41
4.5 Hiệu năng cơ sở.....	42
4.6 Quá trình huấn luyện mô hình.....	42
4.6.1 Thiết lập siêu tham số cho thuật toán PPO	42

4.6.2 Huấn luyện mô hình với môi trường có hệ thống cảm biến dày đặc .	42
4.6.3 Cắt giảm số lượng cảm biến	44
4.6.4 Hệ thống cảm biến tối giản	46
4.6.5 Đánh giá kết quả các cấu hình hệ thống.....	47
4.7 Đánh giá các hệ thống qua các độ đo đánh giá.....	48
4.7.1 Thời gian chờ trung bình.....	48
4.7.2 Chiều dài hàng chờ trung bình.....	49
4.8 Đánh giá các mục tiêu.....	49
4.9 Kết chương.....	50
CHƯƠNG 5. KẾT LUẬN	51
5.1 Kết luận.....	51
5.1.1 Tổng kết kết quả.....	51
5.1.2 Điểm mạnh.....	51
5.1.3 Điểm hạn chế.....	51
5.2 Hướng phát triển.....	51
5.3 Kết luận.....	52
TÀI LIỆU THAM KHẢO.....	54

DANH MỤC HÌNH VẼ

Hình 2.1	Scene View trong Unity	19
Hình 2.2	Game View trong Unity	19
Hình 2.3	Heirarchy View trong Unity	20
Hình 2.4	Project View trong Unity	21
Hình 2.5	Inspector View trong Unity	22
Hình 2.6	Collider của một cá thể xe taxi	23
Hình 2.7	Máy trạng thái hữu hạn cho hệ thống đèn giao thông	24
Hình 2.8	Sự tương tác của các thành phần trong bộ công cụ ML-Agents	26
Hình 3.1	Gói đồ họa giao thông đơn giản	29
Hình 3.2	Nền của môi trường ngã tư đơn giản	29
Hình 3.3	Hệ thống lộ trình	30
Hình 4.1	Hiệu năng cơ sở	41
Hình 4.2	Hệ thống 252 cảm biến	43
Hình 4.3	Kết quả với hệ thống môi trường nhiều cảm biến	43
Hình 4.4	Hệ thống 104 cảm biến	44
Hình 4.5	Kết quả với hệ thống 104 cảm biến	45
Hình 4.6	Hệ thống 36 cảm biến	45
Hình 4.7	Kết quả của hệ thống cảm biến tối giản	46
Hình 4.8	Giá trị điểm thưởng trung bình cho các cấu hình khác nhau của hệ thống	47
Hình 4.9	Thời gian chờ trung bình	48
Hình 4.10	Chiều dài hàng chờ trung bình	49

DANH MỤC THUẬT NGỮ VÀ TỪ VIỆT TẮT

Thuật ngữ	Ý nghĩa
API	Giao diện lập trình ứng dụng (Application Programming Interface)
DQN	Phương pháp Deep Q Network
MDP	Quá trình quyết định Markov (Markov Decision Process)
ML-Agents	Framework học máy của Unity
PoC	Bằng chứng khái niệm (Proof of Concept)
PPO	Phương pháp Proximal Policy Optimization
SDK	Bộ công cụ phát triển phần mềm (Software Development Kit)

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Hiện nay, trong thời đại cách mạng công nghiệp 4.0, sự phát triển và ứng dụng công nghệ trở thành một trong những yếu tố tiên quyết dẫn đến thành công của nhiều ngành nghề và lĩnh vực. Trong số đó, trí tuệ nhân tạo (AI [1]) được xem là một trong những công nghệ được đánh giá cao, có nhiều ứng dụng và đã mang lại những cải thiện rất đáng kể cho nhiều lĩnh vực. Tuy nhiên, có một số lĩnh vực mà không có nhiều ứng dụng nổi bật của AI, trong số đó có thể kể đến giao thông.

Một hệ thống giao thông tối ưu là nhu cầu quan trọng do đây là thứ mà mỗi cá nhân chúng ta đều tương tác, tham gia và sử dụng mỗi ngày. Tuy nhiên hệ thống giao thông mà hiện chúng ta vẫn đang tham gia và tương tác mỗi ngày vẫn còn nhiều điểm có thể cải tiến. Một phần chưa tối ưu trong hệ thống này có thể kể đến các cụm đèn giao thông. Hiện nay, ta có thể dễ dàng thấy rằng có nhiều trường hợp mà tín hiệu đèn giao thông không phản ánh đúng tình trạng giao thông thực tế. Điều này dẫn đến các quyết định tín hiệu đèn và thời gian đèn được đưa ra một cách không hợp lý, gây ra lãng phí thời gian, ảnh hưởng đến sức khoẻ và tâm lý của người tham gia giao thông.

Chính vì vậy, trong đồ án này em đã chọn ứng dụng AI, mà cụ thể là phương pháp học tăng cường, để tìm ra giải pháp tối ưu hoá những vấn đề của hệ thống đèn giao thông hiện tại. Với đồ án này, em hướng tới việc thực hiện nó như một bằng chứng khái niệm (Proof of concept), để thể hiện sự khả thi của việc ứng dụng học tăng cường trong giải quyết bài toán về lĩnh vực giao thông, hướng tới một hệ thống giao thông thông minh, đem lại giá trị lớn cho xã hội.

1.2 Các giải pháp hiện tại và hạn chế

Hiện nay, giải pháp đang được sử dụng ở ngoài thực tế là một hệ thống đèn giao thông thay đổi tuần tự. Trong đó, các mốc thời gian tương ứng cho từng màu đèn, cũng như thứ tự thay đổi màu của các đèn đều được xác định từ trước. Điều này giải quyết vấn đề tương đối tốt với những trường hợp giao thông xảy ra với đúng như những mức thời gian và những quy luật đã được đề ra từ trước.

Tuy nhiên, những ước lượng và tính toán để ra được các quy luật trên đều được tính một cách trung bình với các mốc thời gian lớn, điều này dẫn tới sai số đối với thực tế. Bởi vì bản chất của phương pháp trên không thể hiện được các thông tin về tình trạng giao thông thực tế ở ngoài môi trường, do đó sẽ không thể đưa ra các tín hiệu chuẩn xác và phù hợp nếu các ước lượng trên không được thoả mãn. Bên

cạnh đó thì việc tính toán trên cũng phải diễn ra thường xuyên để có thể theo kịp với những thay đổi ngoài thực tế, điều này cũng yêu cầu có một nguồn nhân lực và dữ liệu lớn.

1.3 Mục tiêu và định hướng giải pháp

Với các hạn chế được nêu trong phần 1.2, em đã đi đến một hướng giải quyết mới cho bài toán. Với việc giả định là các tuyến đường có thông tin từ cảm biến được chuyển về cho ta biết tại từng vị trí có phương tiện đang chiếm hay không. Từ thông tin đó, mô hình sẽ đưa ra cho ta các quyết định về tín hiệu đèn giao thông phù hợp cho đúng thời điểm đó.

Giải pháp này hướng tới giải quyết các vấn đề của phương pháp truyền thống được nhắc đến trong phần 1.2. Với các thông tin môi trường được cập nhật liên tục, mô hình sẽ đưa ra các quyết định phù hợp nhất với đúng thông tin tương ứng đó, từ đó có thể giải quyết được vấn đề về tín hiệu không tương ứng với thực tế. Bên cạnh đó, ta cũng có thể liên tục tiếp tục việc huấn luyện mô hình với các quyết định thực tế trên, và sử dụng một số hướng triển khai mô hình mới một cách an toàn, từ đó có thể đảm bảo việc luôn luôn có một mô hình hoạt động hiệu quả và phù hợp với môi trường đặt ra, không bị giảm hiệu năng với các thay đổi về môi trường theo thời gian.

1.4 Đóng góp của đồ án

Đồ án này có 3 đóng góp chính như sau:

1. Đề xuất một phương pháp tối ưu hoá hệ thống đèn giao thông, sử dụng các thông tin của môi trường và đưa ra các tín hiệu đèn giao thông phù hợp với trình trạng giao thông thực tế trong thời gian thực.
2. Xây dựng một môi trường mô phỏng với bài toán đặt ra là tối ưu hoá hệ thống đèn giao thông ở một ngã tư đơn giản.
3. Thủ nghiệm một số phương pháp học tăng cường để giải quyết bài toán trên.

1.5 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Ở chương 2, em sẽ giới thiệu về những kiến thức nền tảng cần liên quan đến bài toán đặt ra cũng như những lý thuyết được áp dụng trong giải pháp của bài toán. Thông qua chương này, người đọc có thể hiểu rõ hơn về nền tảng của giải pháp cần trình bày, có khả năng hiểu các khái niệm, kiến thức, thuật toán được sử dụng trong đồ án này. Đồng thời, em cũng trình bày một số phương pháp đã và đang được phát triển để giải quyết bài toán, cùng với đó là những điểm mạnh và hạn chế của các phương pháp.

Chương 3 sẽ bao gồm thông tin chi tiết về giải pháp mà em đưa ra và quá trình phát triển nó. Người đọc có thể hiểu được tư tưởng cốt lõi của giải pháp, đồng thời cũng có khả năng kết hợp với lý thuyết để có thể tái tạo lại được các kết quả mà em đã đạt được.

Chương tiếp theo sẽ bao gồm những kết quả thử nghiệm và đánh giá của em. Trong chương này, em trình bày về một số tham số đánh giá được sử dụng và kết quả đạt được đối với việc thử nghiệm nhiều tham số khác nhau của môi trường và phương pháp học tăng cường.

Trong phần tổng kết em sẽ tổng kết lại thông tin về đồ án, kết quả đạt được và trình bày một số hướng phát triển trong tương lai.

Cuối báo cáo sẽ là danh sách các tài liệu tham khảo mà em sử dụng trong quá trình phát triển đồ án cũng như thực hiện viết báo cáo cho đồ án.

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

2.1 Tổng quan

Đối với bài toán đã được giới thiệu trong Chương 1, để giải quyết được bài toán, trước hết ta cần tìm hiểu một số nền tảng lý thuyết. Từ những lý thuyết này, ta có thể hiểu hơn về bài toán đặt ra cũng như có cái nhìn tổng quan các hướng đi cho việc giải quyết bài toán. Chương này sẽ giới thiệu các kiến thức từ đơn giản tới nâng cao để người đọc có khả năng hiểu được rõ về giải pháp được đề xuất trong đồ án này.

2.2 Ngữ cảnh của bài toán

Như đã giới thiệu trong phần trước, bài toán mà ta đang hướng tới là giải quyết một môi trường ngã tư đơn giản bao gồm 4 hướng. Ở đây, đối tượng mà ta cần tối ưu là hệ thống đèn giao thông, trong đó tại mỗi thời điểm đưa ra quyết định, ta có thể lựa chọn tiếp tục màu đèn cũ hoặc thay đổi màu đèn. Khi màu đèn được thay đổi, hệ thống các phương tiện giao thông sẽ tuân thủ theo tín hiệu đèn mà ta đưa ra. Trong phạm vi của bài toán, để đơn giản hoá quá trình này, ta ngầm hiểu hệ thống phương tiện tham gia giao thông sẽ tuân thủ hoàn toàn theo đèn tín hiệu. Bên cạnh đó, để có thể đề xuất một số giải pháp tối ưu, ta cũng ngầm định việc có đủ nguồn lực để thay đổi một số yếu tố môi trường như việc thêm hệ thống cảm biến dưới mặt đường.

2.3 Hệ thống đèn giao thông hiện tại

Hiện tại, hệ thống đèn giao thông đang hoạt động theo phương pháp thay đổi tuần tự dựa trên một khung thời gian đã được cố định từ trước. Khung thời gian này được tính toán thông qua việc lấy các thông tin về mật độ xe trong khu vực trong một thời gian dài và tính toán ra một giá trị trung bình cho các thời gian đó. Từ đó, sử dụng thông tin trên để tìm ra được một bộ số thời gian phù hợp. Phương pháp này có điểm hạn chế lớn là không sử dụng các thông tin trong thời gian thực, do đó không thể tối ưu được cho từng thời điểm cụ thể của môi trường. Bên cạnh đó, các thông tin về giao thông cũng thay đổi theo thời gian, điều này đặt ra yêu cầu về việc tính toán lại các giá trị bộ số trên cho phù hợp với môi trường.

2.4 Các kết quả nghiên cứu tương tự

Học tăng cường đã được ứng dụng trong kiểm soát tín hiệu đèn giao thông trong một số nghiên cứu khoa học trước đây tuy nhiên còn tương đối đơn giản với môi trường mô phỏng thiếu thực tế. Học tăng cường đã bắt đầu được ứng dụng trong kiểm soát tín hiệu đèn giao thông từ những năm của thập niên 90. Nghiên cứu [2]

tổng hợp các phương pháp từ 1997 tới năm 2010 sử dụng học tăng cường trong lĩnh vực này. Trong giai đoạn này, các phương pháp phần lớn bị giới hạn ở Q-learning [3] dạng bảng với giá trị Q được tính toán bằng một hàm tuyến tính. Bên cạnh đó, do những giới hạn về giải thuật thời điểm đó, không gian trạng thái của môi trường thường được thu hẹp lại thành một vài giá trị như số lượng xe đang chờ đèn đỏ, số lượng xe ở mỗi làn đường,... Chính vì thế, thông tin của môi trường không được thể hiện một cách đầy đủ, dẫn tới hiệu năng không được nâng cao.

Với sự phát triển của học tăng cường trong các năm gần đây, các phương pháp mới và hiệu quả liên tục được giới thiệu. Do đó, một số nghiên cứu cũng thử nghiệm ứng dụng học tăng cường vào kiểm soát tín hiệu đèn giao thông. Nghiên cứu [4] và [5] đã sử dụng mạng nơ-ron để xấp sỉ hàm giá trị Q để vượt qua vấn đề về kích thước không gian trạng thái của phương pháp Q-learning truyền thống, đạt được kết quả khả quan hơn so với các phương pháp trước đó. Nghiên cứu [6] đã ứng dụng những phương pháp tiên tiến hơn của học tăng cường trong thời điểm đó như Double DQN [7] hay Prioritised Experience Replay (PER) [8].

Các phương pháp trên đều đạt được kết quả tốt, tuy nhiên vẫn tồn tại một số vấn đề. Các kết quả đạt được đều rất khó hoặc không thể tái tạo lại do sự thiếu hụt thông tin về chi tiết triển khai thuật toán, phương pháp tính toán các độ đo đánh giá và chi tiết phương pháp tạo ra hiệu năng cơ sở. Môi trường được sử dụng trong việc phát triển thuật toán và đánh giá kết quả cũng thiếu sự tùy biến, khó có thể thiết kế và thu thập được thêm các thông tin mới nếu muốn phát triển bài toán. Bên cạnh đó, cùng với sự phát triển liên tục của lĩnh vực học tăng cường, các phương pháp mới được phát triển liên tục và đạt được kết quả cải thiện nhiều so với các phương pháp cũ. Do đó, việc ứng dụng các phương pháp này vào bài kiểm soát tín hiệu đèn giao thông, dưới một môi trường giả lập phù hợp với thực tế là một việc quan trọng, là bước đầu cho quá trình ứng dụng học tăng cường vào vấn đề này ở ngoài thực tế.

2.5 Học tăng cường

Phương pháp học tăng cường (reinforcement learning [9]) là một nhánh con của học máy (machine learning [10]) mà bản chất hoạt động theo cơ chế thử nghiệm nhiều lần và rút ra kinh nghiệm sau mỗi lần thử. Gần đây, học tăng cường đang trở nên khá nổi tiếng nhờ một số những kết quả đạt được như việc AlphaGo [11] thắng áp đảo các tuyển thủ thế giới trong cờ vây, OpenAI [12] chiến thắng các đội mạnh nhất trong trò chơi DOTA2. Trong phần này, chúng ta sẽ tìm hiểu về các kiến thức nền tảng của học tăng cường từ đó đi đến một thuật toán nổi tiếng trong nhóm này là Proximal Policy Optimization [13].

2.5.1 Mô hình hoá bài toán

Trong phần lớn các phương pháp, giải thuật trong trí tuệ nhân tạo và đặc biệt là trong học máy, ta thường hay mô hình hoá bài toán bằng các mô hình toán học. Do đó, đối với học tăng cường, việc mô hình hoá bài toán bằng toán học cũng là một hướng tiếp cận khả dĩ. Cụ thể ở đây, ta sử dụng Quá trình quyết định Markov (Markov Decision Process [14]) để mô hình hoá bài toán.

a, Quá trình quyết định Markov

Markov Decision Process (MDP) cung cấp một nền tảng toán học cho việc mô hình quá việc đưa ra quyết định trong các tình huống mà kết quả là một phần ngẫu nhiên và một phần dưới sự điều khiển của người ra quyết định. Tại mỗi bước thời gian, quá trình này ở trong một trạng thái s , agent đưa ra quyết định có thể chọn bất kỳ hành động a . MDP đáp ứng trong bước thời gian tiếp theo bằng việc đưa ta tới một trạng thái mới s' và cho agent một điểm thưởng tương ứng $R_a(s, s')$.

Xác suất mà agent đi tới trạng thái s' sau khi đưa ra hành động a được tính bởi hàm chuyển tiếp trạng thái $P_a(s, s')$. Do đó, trạng thái kế tiếp s' phụ thuộc vào trạng thái hiện tại s và hành động a .

Một quá trình quyết định Markov là tập các vector 5 chiều $(S, A, P(\cdot, \cdot), R(\cdot, \cdot), \gamma)$, trong đó:

- S là một dãy hữu hạn các trạng thái.
- A là một dãy hữu hạn các hành động.
- $P_a(s, s')$ là xác suất mà hành động a trong trạng thái s tại thời điểm t sẽ dẫn đến trạng thái s' trong thời điểm $t + 1$.
- $R_a(s, s')$ là điểm thưởng mà agent nhận được sau khi chuyển tiếp sang trạng thái s' từ trạng thái s .
- $\gamma \in [0, 1]$ là hệ số chiết khấu, đại diện cho sự khác biệt về giá trị của điểm thưởng nhận ngay lập tức và điểm thưởng nhận trong tương lai.

Bài toán cốt lõi của MDP đó là tìm ra một nguyên tắc (policy) π mà xác định hành động $a = \pi(s)$. Mục đích của bài toán là tìm được policy π tối ưu sao cho hàm tích luỹ điểm thưởng được tối ưu hoá.

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \quad (\text{trong đó ta chọn } a_t = \pi(s_t))$$

2.5.2 Policy

Trong học tăng cường, điều tiên quyết mà ta hướng đến là tìm ra một policy tối ưu. Từ đó, agent có thể biết được nên đưa ra các hành động như thế nào trong từng trạng thái cụ thể của môi trường.

$$a_t = \pi_\theta(s_t)$$

Ở đây, ta thấy π có thể được tham số hoá bằng θ , điều này tương tự với việc các mô hình trong các phương pháp học sâu được tham số hoá.

Như đã đề cập, sau khi mô hình hoá bài toán bằng MDP, mục tiêu của ta là tìm ra policy mà đem về cho ta lượng điểm thưởng tối đa:

$$\max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

Đối với trường hợp ngẫu nhiên, policy sẽ cho chúng ta một phân phối xác suất $p(a_t | s_t) = \pi_\theta(a_t | s_t)$.

Trong phương pháp học tăng cường, ta muốn tìm được một chuỗi các hành động mà dẫn đến cho ta một lượng điểm thưởng kỳ vọng lớn nhất hoặc là giảm tối đa phí (cost). Có nhiều cách để giải được vấn đề này:

- Đánh giá mức độ tốt khi đạt được từng trạng thái nhất định hoặc là thực hiện các hành động nhất định tại từng trạng thái, từ đó chọn hành động tốt nhất (value learning).
- Sử dụng mô hình của môi trường để tìm ra hành động trả về điểm thưởng lớn nhất (ví dụ như việc tính trước các nước đi trong cờ vua khi ta đã biết rõ luật) (model-based learning).
- Tìm ra một policy trực tiếp tối đa hoá lượng điểm thưởng có thể nhận được (policy gradient).

Sau đây ta sẽ tìm hiểu sơ qua về 3 phương pháp trên.

2.5.3 Học tăng cường dựa vào mô hình

Trong bài toán này, ta biết trước được mô hình gồm các tính chất của môi trường, từ đó, ta có thể dự đoán được trạng thái tiếp theo khi ta đưa ra một hành động. Một ví dụ điển hình cho bài toán này là cờ vua. Trong cờ vua, với mỗi trạng thái s là thế cờ hiện tại, khi ta đưa ra hành động a , một nước đi nào đó, ta có thể biết được trạng thái s' tiếp theo.

Điểm cốt lõi của học tăng cường dựa và mô hình là sử dụng các tính chất đã biết của môi trường, cùng với đó là xây dựng các hàm giá trị để tìm ra một chuỗi các trạng thái và thành động (trajectory τ) tối ưu.

2.5.4 Học tăng cường theo giá trị

Trong một số trường hợp, dù biết đầy đủ các tính chất của môi trường, ta vẫn không thể tự mình tìm được các hành động tối ưu. Đặc biệt là các trường hợp mà các khả năng có thể xảy ra ở mỗi trạng thái rất lớn. Thay vì thế, ta có thể chuyển qua đánh giá giá trị của việc đạt được các trạng thái cụ thể, từ đó mà tìm ra các hành động tương ứng nhờ vào mô hình của bài toán. Sự đánh giá này có thể thực hiện thông qua hàm giá trị (value function) $V_\pi(s)$.

a, Hàm giá trị

Hàm giá trị $V_\pi(s_t)$ tính toán lượng điểm thưởng kỳ vọng đã chiết khấu cho trạng thái s_t tại thời điểm t dưới policy hiện tại. Ta cũng có thể hiểu là đơn giản là đây là một hàm có tác dụng đánh giá và dự đoán lượng điểm thưởng mà ta có thể nhận khi đạt một trạng thái và tiếp tục sử dụng policy hiện tại cho các hành động và trạng thái sau đó.

$$V^\pi(s_t) = \sum_{t'=t}^T E_{\pi_\theta} \left[\gamma^{t'-t} r(s_{t'}, \mathbf{a}_{t'}) \mid s_t \right]$$

Ví dụ như trong cờ vua, sẽ có những nước đi mà ở đó ta có lợi thế lớn so với khi ta chuẩn bị thua, thông thường khi theo một policy tối ưu thì hàm giá trị cho nước đi đó sẽ trả về giá trị lớn hơn so với khi ta sắp thua.

Khi ta có được một hàm giá trị hợp lý, có một số cách để ta tìm được một policy tối ưu tương ứng. Tại mỗi bước thời gian, ta sử dụng $V(s)$ để tìm ra trạng thái tối ưu trong bước thời gian tiếp theo mà ta muốn đạt tới, sau đó ta sử dụng các thông tin tính chất của môi trường để từ đó đưa ra được lựa chọn tốt nhất.

Đối với cách bài toán đã biết rõ mô hình thì đây là một điều dễ dàng, tuy nhiên trong trường hợp ta không biết rõ các tính chất của môi trường, việc tìm ra lựa chọn tương ứng để đưa ta tới trạng thái tối ưu tiếp theo mà ta tìm được là rất khó.

b, Tìm kiếm hàm giá trị

Trong một số môi trường, số lượng trạng thái là rất lớn. Điều đó dẫn đến việc lưu trữ giá trị cho từng trạng thái trở thành một việc bất khả thi. Để giải quyết vấn đề này, ta có thể ứng dụng các kết quả đã đạt được trong học giám sát (supervised learning) và huấn luyện một mô hình có khả năng ước lượng được V .

$$\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|V_{\phi}^{\pi}(\mathbf{s}_i) - \mathbf{y}_i\|^2$$

Trong đó:

- \mathbf{y} là giá trị mục tiêu và ta có thể dùng các phương pháp như Monte Carlo [15], Temporal Difference [16] để tính toán.

c, Hàm giá trị hành động

Vấn đề của hàm giá trị thông thường cho từng trạng thái là với việc tìm ra trạng thái chính xác, đôi khi ta vẫn không tìm được các hành động tương ứng dẫn ta tới trạng thái đó. Đặc biệt là ở trong các môi trường mà ta không biết rõ các tính chất, điều là dường như là bất khả thi.

Để giải quyết vấn đề này, ta có thể tính toán giá trị cho từng hành động $Q(s, a)$ thay vì cho mỗi trạng thái của môi trường. Khi này, chúng ta sẽ cần phải theo dõi lượng dữ liệu lớn hơn (có nhiều action tại mỗi state), tuy nhiên, khi ta tìm được một hàm giá trị hành động đủ tốt, ta có thể từ đó mà cho ra ngay lập tức hành động ta cần chọn. Điều này giúp cho ta không còn phụ thuộc vào các tính chất của môi trường một cách hoàn toàn.

Khi ta đã có được một hàm Q đủ tốt, ta tìm hành động tối ưu

$$a^* = \operatorname{argmax}_a Q(s, a)$$

Như vậy, ta có thể nói phương pháp học hàm giá trị hành động là không phụ thuộc vào mô hình.

d, Hàm lợi thế

Trong deep learning, gradient descent [17] thường hoạt động tốt nếu các giá trị của chúng ta được cân bằng hoá xung quanh 0. Ý tưởng này cũng được triển khai trong học tăng cường. Hàm lợi thế là một hàm có tác dụng đánh giá lợi thế của một hành động a trong trạng thái s so với mức điểm thưởng kỳ vọng thông thường ta có thể nhận được từ trạng thái s .

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t)$$

Trong đó:

- $V^{\pi}(s_t)$ là lượng điểm thưởng kỳ vọng mà ta có thể đạt được từ trạng thái s_t và làm theo policy π từ đó trở đi.

- $Q^\pi(s_t, a_t)$ là lượng điểm thưởng kỳ vọng mà ta có thể đạt được nếu thực hiện hành động a_t trong trạng thái s_t và làm theo policy từ đó trở đi.
- $A^\pi(s_t, a_t)$ là lượng lợi thế khi ta thực hiện hành động a_t tại trạng thái s_t và làm theo policy π từ đó trở đi.

e, Phương pháp Q-learning

Epsilon-greedy [3] là một phương pháp giúp ta cân bằng được việc khám phá môi trường và sử dụng policy. Việc khám phá môi trường có mục đích để có thể bao quát được toàn bộ không gian trạng thái và hành động, tránh cho việc agent bị overfit vào một ngữ cảnh cụ thể. Trong khi đó, việc sử dụng policy là mục đích sau cùng của ta khi đã có một policy tối ưu, từ đó các hành động của agent không còn là các hành động ngẫu nhiên mà trên thực tế là có giá trị giúp cho ta đưa ra được lựa chọn sau cùng một cách hợp lý.

Cách hoạt động của phương pháp này khá đơn giản, tại mỗi bước thời gian t , agent sẽ có thể chọn giữa việc ra quyết định ngẫu nhiên (explore), hoặc sử dụng policy để ra quyết định (exploit). Điều này được quyết định thông qua hệ số ϵ . Khi giá trị epsilon sẽ quyết định việc agent chọn explore hay exploit môi trường. Cùng với đó giá trị này cũng sẽ thay đổi theo thời gian, từ đó càng về sau thì tỉ lệ exploit càng cao. Sau cùng agent sẽ đưa ra quyết định hoàn toàn theo policy mà không tiến hành khám phá môi trường nữa.

Thuật toán Q-learning

Với những môi trường mà không gian trạng thái và hành động nhỏ, ta có thể lưu tất cả các giá trị Q cho từng cặp trạng thái và hành động và từ đó tạo ra một bảng Q-table lưu các giá trị này. Khi đó, ta sử dụng thuật toán Q-learning [3] đơn giản với Q-table để tính toán các giá trị Q , thuật toán được trình bày trong mục thuật toán 1.

Algorithm 1 Thuật toán Q-learning

Tham số: kích thước bước $\alpha \in (0, 1]$, giá trị $\epsilon > 0$ nhỏ

Khởi tạo $Q(s, a) \forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)$

for mỗi tập của môi trường **do**

 Khởi tạo s_0

for $t = 0$ tới T **do**

 Với xác suất ϵ , chọn một hành động ngẫu nhiên a_t ,

 hoặc chọn $a_t = \max_a(Q(s_t, a))$

 Thực hiện hành động a_t , theo dõi điểm thưởng r_t , trạng thái mới s_{t+1}

 Cập nhật $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

$t \leftarrow t + 1$

end for

end for

Bằng việc liên tục tương tác với môi trường và lấy được nhiều mẫu, thuật toán giúp ta cập nhập liên tục được các giá trị trên Q-table. Khi Q-table có các giá trị chính xác, ta có thể sử dụng nó để giúp ta đưa ra các hành động (ví dụ ở trạng thái s_t , ta chọn hành động a_t^* trong bảng Q-table mà cho ta giá trị lớn nhất).

f, Phương pháp Deep Q Network

Có những trường hợp môi trường có không gian trạng thái và hành động lớn, ta không thể có đủ không gian để lưu được hết từng giá trị này vào trong một bảng nào đó. Deep Q Network [18] cung cấp giải pháp là sử dụng một mạng neuron network để thực hiện tính toán một cách gần đúng nhất các giá trị Q. Khi này, thuật toán Q-learning cần phải được thay đổi để phù hợp, được thể hiện ở trong mục thuật toán 2.

Algorithm 2 Deep Q-learning sử dụng kho kinh nghiệm

```

Khởi tạo kho kinh nghiệm  $\mathcal{D}$  với kích thước tối đa  $N$ 
Khởi tạo mạng giá trị hành động  $Q$  với tham số ngẫu nhiên.
for mỗi tập của môi trường do
    Khởi tạo  $s_0$ 
    for  $t = 0$  tới  $T$  do
        Với xác suất  $\epsilon$ , chọn một hành động ngẫu nhiên  $a_t$ ,
        hoặc chọn  $a_t = \max_a(Q^*(s_t, a; \theta))$ 
        Thực hiện hành động  $a_t$ , theo dõi điểm thưởng  $r_t$ , và trạng thái mới  $s_{t+1}$ 
        Lưu trữ bước chuyển  $(s_t, a_t, r_t, s_{t+1})$  vào  $\mathcal{D}$ 
        Gán  $s_{t+1} = s_t$ 
        Lấy mẫu một lượng nhỏ các tập ở trong kho kinh nghiệm  $\mathcal{D}$ 
        Gán  $y_j = \begin{cases} r_j & \text{for terminal } s_{t+1} \\ r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) & \text{for non-terminal } s_{t+1} \end{cases}$ 
        Thực hiện tối ưu hoá bằng gradient descent với  $(y_j - Q(s_t, a_j; \theta))^2$ 
    end for
end for

```

1. Lấy các mẫu hành động từ một trạng thái, có thể sử dụng các phương pháp khác nhau để tăng tính ngẫu nhiên cho các mẫu, từ đó giúp cân bằng được sự đánh đổi giữa khám phá và khai phá.
2. Theo dõi lượng điểm thưởng nhận được từ môi trường và trạng thái môi trường sau khi hành động được thực hiện.
3. Sử dụng hàm Q để tìm hành động a' đem lại cho ta giá trị Q lớn nhất.

$$Q_{k+1}(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} Q_k(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

2.5.5 Policy Gradient

Bên cạnh học tăng cường dựa vào mô hình và giá trị, còn một nhánh các nhóm phương pháp học tăng cường nữa là nhóm học tăng cường dựa vào policy. Các phương pháp này tập trung vào việc trực tiếp tối ưu hoá policy π thay vì tìm policy một cách gián tiếp thông qua việc tối ưu hàm giá trị hay dựa vào các bộ luật cụ thể của môi trường.

Hàm mục tiêu của các phương pháp học tăng cường dựa vào policy gradient [19] sẽ là

$$J(\theta) = \mathbb{E} \left[\sum_{t=0}^H R(s_t, u_t; \pi_\theta) \right] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

Trong đó

- $J(\theta)$ là hàm mục tiêu, là giá trị điểm thưởng kỳ vọng mà ta có thể đạt được với policy π .
- $\sum_{t=0}^H R(s_t, u_t)$ là tổng các điểm thưởng của từng cặp trạng thái và hành động có thể xảy ra.
- $\sum_{\tau} P(\tau; \theta) R(\tau)$ là diễn giải cho về trái, ta lấy tổng của các tích điểm thưởng của một chuỗi hành động và trạng thái τ nhân với khả năng mà chuỗi đó xảy ra.

Mục tiêu của ta là tìm được bộ tham số θ cho policy π_θ , sao cho từ policy này ta tìm được chuỗi τ mà cho ta lượng điểm thưởng lớn nhất.

$$\max_{\theta} J(\theta) = \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

a, Tối ưu hoá

Việc tối ưu hoá bài toán trên cần ta phải sử dụng một số biến đổi toán học đã được chứng minh.

$$f(x) \nabla_{\theta} \log f(x) = f(x) \frac{\nabla_{\theta} f(x)}{f(x)} = \nabla_{\theta} f(x)$$

$$E_{x \sim p(x)}[f(x)] = \int p(x)f(x)dx$$

Ta thay $f(x)$ bằng π ,

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

Bài toán tối ưu khi này trở thành

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \underbrace{\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)}$$

Ta viết lại hàm mục tiêu $J(\theta)$

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau$$

Ta tính gradient để sử dụng cho việc tối ưu

$$\begin{aligned} \nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau \\ &= E_{\tau \sim \pi_\theta(\tau)} [\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \end{aligned}$$

Tuy ta không thể tìm được toàn bộ các giá trị τ để tính được gradient của policy, sau khi biến đổi như trên, $\nabla_\theta J(\theta)$ trở thành một kỳ vọng, do đó ta có thể dùng việc lấy mẫu (sampling) để tính toán xấp xỉ giá trị đó.

Tính $\nabla_\theta \log \pi_\theta(\tau)$

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^T \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

Lấy log 2 về

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^T \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) + \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$\log p(\mathbf{s}_1) v \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ không phụ thuộc vào θ nên ta có thể bỏ, khi đó policy gradient trở thành

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta} (\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Ta dùng policy gradient đã tính toán được này để tối ưu hoá policy.

Policy Gradient sử dụng Monte Carlo

Thuật toán REINFORCE [19] sử dụng Monte Carlo để tính toán các giá trị điểm thưởng một cách chính xác. Quá trình này được thực hiện qua việc chạy các chuỗi tương tác hoàn chỉnh đến khi môi trường đạt trạng thái kết thúc, sau đó mới tiến hành tính các giá trị điểm thưởng trả về thông qua Monte Carlo. Các giá trị này được sử dụng để tính toán policy gradient. Thuật toán được trình bày trong mục thuật toán 3.

Algorithm 3 REINFORCE

Khởi tạo policy $\pi(a|s, \theta)$

Khởi tạo tham số θ

Chọn kích thước bước nhảy $0 < \alpha \leq 1$

Chọn giá trị chiết khấu điểm thưởng $0 < \gamma < 1$

Chọn số lượng tập tối đa N

Chọn số lượng tập cho một lần cập nhật mô hình $K \geq 1$

for tập n tới N **do**

for K bộ các tập của môi trường **do**

 Tạo một chuỗi hoàn chỉnh các bước chuyển τ trong các tập tuân thủ theo policy $\pi(a|s, \theta)$

for mỗi bước t trong tập môi trường **do**

 Tính giá trị điểm thưởng chiết khấu $G_t \leftarrow \sum_{t=1}^T \gamma^t R_t$

end for

 Tính giá trị măt măt policy cho toàn bộ các tập trong bộ các tập

$$\mathcal{L}(\theta) = -\frac{1}{m} \sum_t^T \log (G_t \pi (a_t | s_t, \theta))$$

Cập nhật policy: $\theta \leftarrow \theta + \alpha \nabla \mathcal{L}(\theta)$

Gán $n \leftarrow n + 1$

end for

end for

Ta có thể sử dụng hàm lợi thế A giống ý tưởng khi sử dụng nó trong Q-learning.

Khi này policy gradient trở thành

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) A^{\pi}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Bằng cách này, ta cân bằng được các giá trị điểm thưởng và giúp cho việc học của agent được dễ dàng hơn.

b, Vấn đề của các thuật toán Policy Gradient

Các phương pháp học tăng cường dựa vào policy gradient được gọi là các phương pháp học on-policy. Lý do là để tính policy gradient cho π thì ta cần thu các dữ liệu được thực hiện bằng chính policy π . Điều này dẫn đến việc sử dụng dữ liệu không hiệu quả do toàn bộ lượng lớn dữ liệu sẽ chỉ được dùng cho một bước gradient descent duy nhất.

c, Importance Sampling

Để cải thiện vấn đề sử dụng dữ liệu không hiệu quả, importance sampling [20] cho phép ta tính toán lượng điểm thưởng kỳ vọng bằng một policy và sử dụng nó để học cho một policy khác.

$$J(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{P(\tau | \theta)}{P(\tau | \theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_{\theta} P(\tau | \theta)}{P(\tau | \theta_{\text{old}})} R(\tau) \right]$$

Nếu $\theta = \theta_{\text{old}}$, ta có:

$$\nabla_{\theta} J(\theta)|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_{\theta} P(\tau | \theta)|_{\theta_{\text{old}}}}{P(\tau | \theta_{\text{old}})} R(\tau) \right]$$

$$= \mathbb{E}_{\tau \sim \theta_{\text{old}}} [\nabla_{\theta} \log P(\tau | \theta)|_{\theta_{\text{old}}} R(\tau)]$$

Điều này hoàn toàn đúng với kết quả mà ta có được về policy gradient thông thường ở trên.

Khi này, policy gradient trở thành:

$$\nabla_{\theta} J(\theta) = E_{\tau \sim \bar{\pi}_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \left(\prod_{t'=1}^t \frac{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\bar{\pi}_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

Như vậy, với hàm mục tiêu ban đầu

$$J(\theta) = \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_\theta(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)] = \sum_{t=1}^T E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)] \right]$$

Áp dụng importance sampling ta có:

$$J(\theta') = \sum_{t=1}^T E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[\frac{p_{\theta'}(\mathbf{s}_t)}{p_\theta(\mathbf{s}_t)} E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} r(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

Nếu ta có thể ràng buộc sự thay đổi của policy, ta có thể bỏ qua phần

$$\frac{p_{\theta'}(\mathbf{s}_t)}{p_\theta(\mathbf{s}_t)}$$

Khi đó bài toán trở thành:

$$\begin{aligned} & \underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\ & \text{subject to} \quad \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_\theta(\cdot | s_t)]] \leq \delta \end{aligned}$$

Với việc thêm một ràng buộc rằng policy mới và policy cũ không được cách nhau quá xa, ta có thể sử dụng các thông tin thu thập được một cách hiệu quả hơn.

d, Proximal Policy Optimization

Phương pháp Proximal Policy Optimization [13] (PPO) được phát triển bởi OpenAI và đạt được những kết quả rất ấn tượng, thể hiện được giá trị ứng dụng của học tăng cường trong thực tế. Ý tưởng của phương pháp này là sử dụng importance sampling, nâng cao sự hiệu quả của việc sử dụng dữ liệu.

Trong đó, ta lưu giữ 2 policy $\pi_\theta(a_t | s_t)$ là policy mà ta muốn tối ưu hoá, $\pi_{\theta_k}(a_t | s_t)$ là policy cuối cùng gần nhất mà ta dùng để thu được các chuỗi dữ liệu mà MDP trả về. Sử dụng importance sampling, ta tối ưu hoá policy mới bằng dữ liệu thu được từ policy cũ.

$$\underset{\theta}{\text{maximize}} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right]$$

Tuy nhiên, khi ta cập nhập policy mới trong mỗi bước tối ưu, khoảng cách giữa 2 policy ngày càng xa, sự ước lượng của importance sampling ngày càng lớn. Do đó, sau một số bước tối ưu hoá policy mới, ta lại phải cập nhật lại policy tương tác với môi trường để lấy dữ liệu:

$$\pi_{\theta_{k+1}}(a_t | s_t) \longleftarrow \pi_\theta(a_t | s_t)$$

Để giải quyết vấn đề này, ta thêm vào bước giới hạn sự thay đổi policy. Ta giới hạn điểm lợi thế tính được nếu khoảng cách giữa 2 policy là quá xa. Khi đó, ta sử dụng một hàm mục tiêu mới:

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min \left(r_t(\theta) \hat{A}_t^{\pi_k}, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t^{\pi_k} \right) \right] \right]$$

Chi tiết thuật toán được trình bày trong mục thuật toán 4.

Algorithm 4 Proximal Policy Optimization với hàm mục tiêu bị chặn

Đầu vào: các tham số khởi tạo của policy θ_0 , giới hạn chặn ϵ , số bước cập nhật policy với một bộ các kinh nghiệm thu được

for $k = 0, 1, 2, \dots$ **do**

Thu thập các chuỗi chuyển trạng thái \mathcal{D}_k trên policy $\pi_k = \pi(\theta_k)$

Tính ước lượng giá trị lợi thế $\hat{A}_t^{\pi_k}$ sử dụng các phương pháp tính toán lợi thế

Tính toán cập nhật policy

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

bằng K bước cập nhật các bộ kinh nghiệm thu được với

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min \left(r_t(\theta) \hat{A}_t^{\pi_k}, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t^{\pi_k} \right) \right] \right]$$

end for

Giải thích

Nếu $r_t(\theta)$ nằm ngoài khoảng $(1 - \epsilon)$ và $(1 + \epsilon)$, giá trị này sẽ được chặn lại tại 2 giá trị trên. Ta cũng cần hiểu rằng ở đây, giá trị $r_t(\theta)$ càng lớn càng có nghĩa là hành động hiện đang được chọn có xác suất được chọn cao hơn trên policy hiện tại so với policy cũ. Khi đó có một vài trường hợp mà ta cần chú ý.

1. $A > 0, r_t(\theta) \geq (1 + \epsilon)$: Khi này hành động được chọn được đánh giá là tốt hơn so với kỳ vọng. $r_t(\theta)$ lớn đồng nghĩa với việc hành động này đang sẵn có khả năng được chọn cao hơn nhiều so với policy trước. Đây là điều tốt, tuy nhiên, vì ngay thời điểm này nó đã đang làm khá tốt rồi, ta không muốn model tiếp tục học theo hướng này quá nhiều vì điều đó có khả năng dẫn đến việc bước

quá xa và dễ dẫn đến một bước cập nhật policy quá mức, dẫn đến hỏng quá trình huấn luyện. Do đó ta giới hạn giá trị này lại ở mức $(1 + \epsilon)$.

2. $A > 0, r_t(\theta) \leq (1 + \epsilon)$: Cùng cách đánh giá như trên, khi này ta thấy policy mới của ta tiếp tục cần phải học theo hướng này (tăng khả năng chọn hành động a trong trạng thái s hiện tại). Do đó hàm mục tiêu không bị chặn khi mà với $r_t(\theta) < (1 - \epsilon)$ thì phần đầu của hàm min sẽ được chọn, giúp cho bước cập nhật policy được thực hiện trọn vẹn.
3. $A < 0, r_t(\theta) < (1 - \epsilon)$: Tương tự với ý tưởng của trường hợp 1 nhưng ngược hướng, việc này giúp cho agent có thể tiếp tục cập nhật policy theo hướng đúng nhưng không bị quá mức.
4. $A < 0, r_t(\theta) > 1$: Khi này, hành động này mang lại giá trị lợi thế âm, do đó policy hiện tại nên bước cập nhật policy nên khiến cho xác suất hành động này được chọn tại trạng thái hiện tại giảm đi. Tuy nhiên ở đây, ta thấy nó lại đang tăng lên so với bước cập nhật policy trước đó. Điều này chứng tỏ bước cập nhật trước đã bị sai. Điểm hay ở đây là vì khi này giá trị lợi thế đang âm, do đó về phái của hàm min sẽ được chọn, cho phép agent thực hiện một bước cập nhật policy theo hướng ngược lại với lần cập nhật trước với một giá trị không bị chặn, điều này phần nào hoạt động như một cơ chế sửa sai sau khi thực hiện một bước học sai.

Vậy là chỉ bằng một hàm mục tiêu duy nhất, đơn giản, không có nhiều ràng buộc, phương pháp Proximal Policy Optimization giúp ta giải quyết được rất nhiều điểm yếu của các phương pháp phụ thuộc vào policy gradient truyền thống. PPO hoạt động một cách hiệu quả về mặt dữ liệu, có khả năng hội tụ tốt và tránh được các vấn đề liên quan đến cập nhật policy sai và phá hỏng quá trình huấn luyện.

2.6 Unity3D

2.6.1 Engine trò chơi của Unity3D

Unity [21] cung cấp một engine trò chơi và một IDE bao gồm hệ thống tương tác với môi trường (editor), hệ thống vật dụng (asset), hệ thống xây dựng cảnh, hệ thống mạng và một số yếu tố khác. Và với những yếu tố trên, tất nhiên Unity 3D có thể được sử dụng để làm nhiều việc, không chỉ bị giới hạn ở việc thiết kế trò chơi. Unity3D gồm 5 thành phần chính là project view, scene view, game view, heirarchy view, inspector view.

a, Scene view

Đây là phần chính mà ta sử dụng trong quá trình phát triển ứng dụng. Scene view cung cấp cho ta một cái nhìn tổng quát về ngữ cảnh của môi trường, trong đó



Hình 2.1: Scene View trong Unity



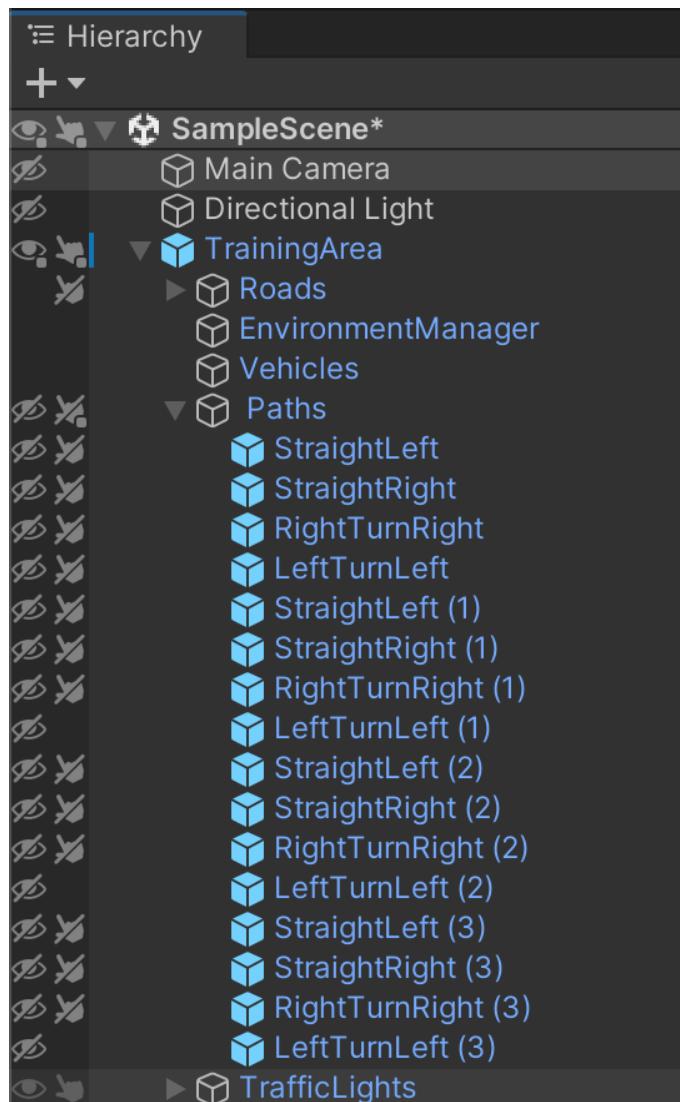
Hình 2.2: Game View trong Unity

ta có thể thấy các vật thể và vị trí của chúng. Scene view được thể hiện trong hình 2.1.

Scene view cho phép người dùng sử dụng các thao tác kéo thả để di chuyển và định hình các vật thể trong môi trường được thực hiện thông qua một số công cụ như bộ công cụ dịch chuyển (translate tool) dùng để dịch chuyển các vật thể, bộ công cụ xoay (rotation tool) dùng để thay đổi độ quay của vật thể theo các trục, bộ công cụ tăng giảm kích thước (dùng để thay đổi kích thước của vật thể theo các trục), và một số công cụ khác. Bên cạnh đó thì scene view cũng cho phép người dùng nhìn được toàn thể các yếu tố của môi trường mà một số trong đó người chơi không thể nhìn được, ví dụ như các thông tin về bộ khung vật lý của vật thể, hay các thông tin phục vụ cho việc sửa các lỗi trong quá trình phát triển trò chơi.

b, Game view

Game view là phần mà người chơi sẽ nhìn thấy khi tương tác với trò chơi được tạo ra. View này cho phép người dùng có thể xem trước được góc nhìn của người chơi, từ đó có thể điều chỉnh các hành động của trò chơi cũng như các vật thể trong

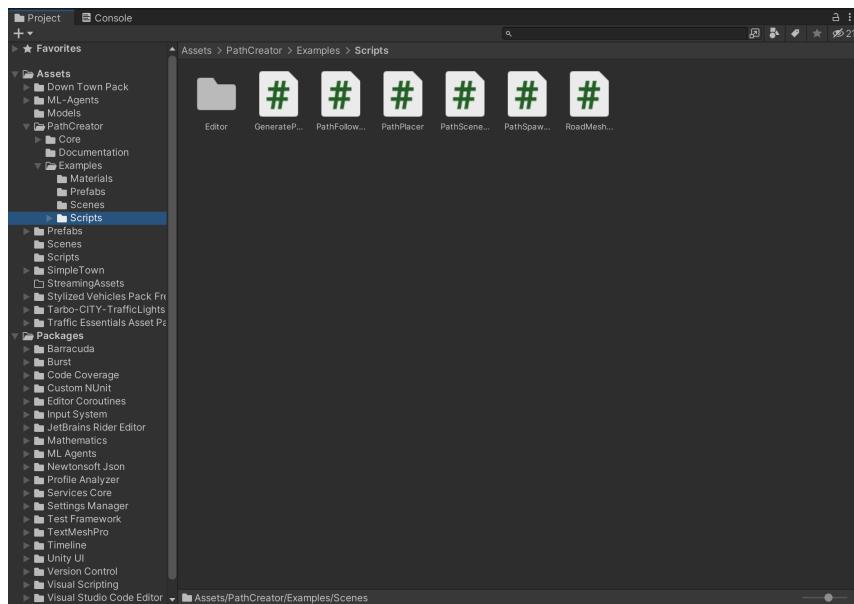


Hình 2.3: Heirarchy View trong Unity

trò chơi một cách hiệu quả. Góc nhìn này được lấy thông qua camera chính của trò chơi được thiết lập vị trí ở trong scene view, qua đó ta có thể giới hạn không gian mà người dùng có thể nhìn thấy trong sản phẩm hoàn thiện, thay vì cho họ nhìn toàn bộ khung cảnh môi trường. Một ví dụ cho game view được thể hiện trong hình 2.2.

c, Hierarchy view

Hierarchy view cho phép ta theo dõi toàn bộ các vật thể đang hiện hữu trong môi trường theo dạng danh sách. Ở đây các vật thể được tạo ra sẽ có dạng cha và con. Các vật thể con sẽ được nằm trong cha và có thể được mở đóng thông qua cha. Trong heirachy view được thể hiện trong hình 2.3, ta có thể thấy tổng quát các vật thể trong môi trường, trong đó các vật thể cha được gắn hình mũi tên ở bên trái, thể hiện nó có thể được mở để ta thấy các vật thể con trong đó.

**Hình 2.4:** Project View trong Unity

d, Project view

Project view 2.4 chứa toàn bộ các thông tin liên quan đến dự án hiện tại của ta, bao gồm các cảnh, đoạn mã nguồn, hay các mô hình đồ họa, và toàn bộ các dữ liệu khác mà ta sử dụng trong dự án. Project view hoạt động gần tương tự như các phần mềm duyệt tập dữ liệu trong máy tính của chúng ta, tuy nhiên một số thông tin không cần thiết đã được lược bỏ. Do đó, view này cũng cho phép người dùng tạo, xoá tập dữ liệu và quản lý toàn bộ các tập dữ liệu này trong dự án một cách có quy củ.

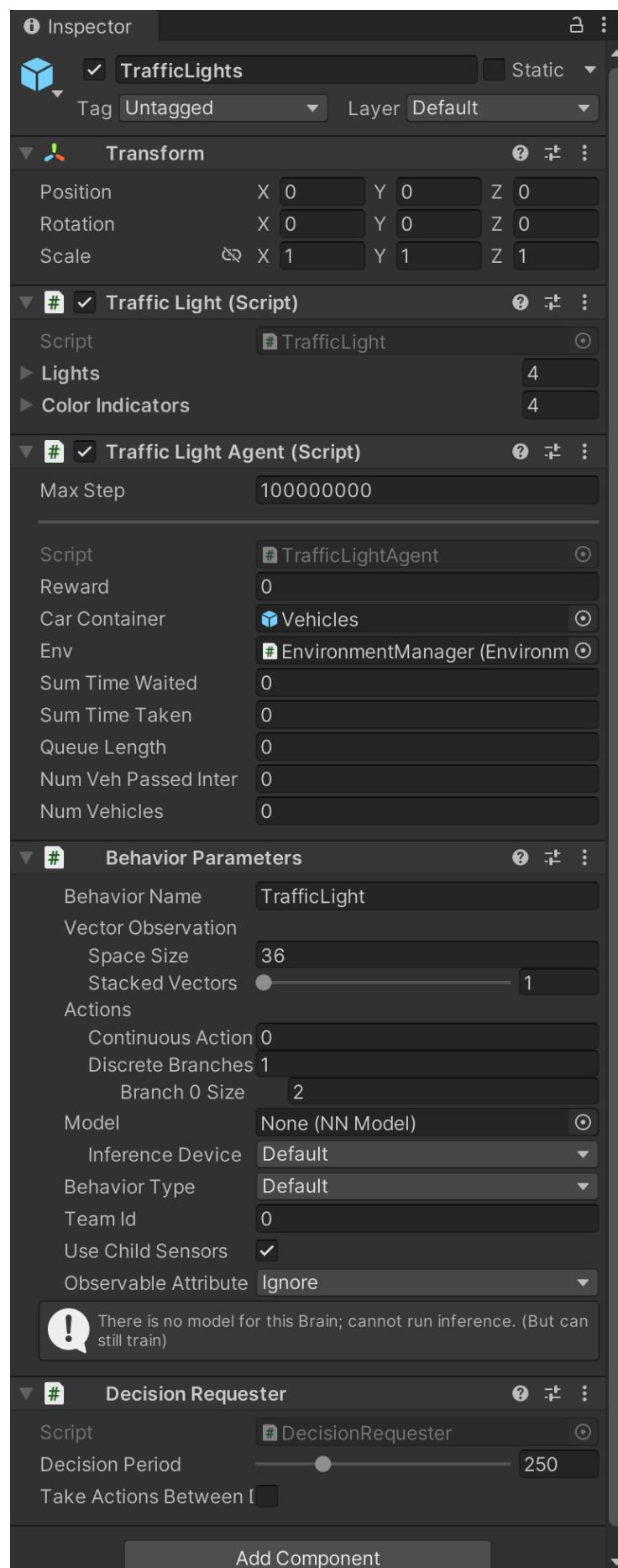
e, Inspector view

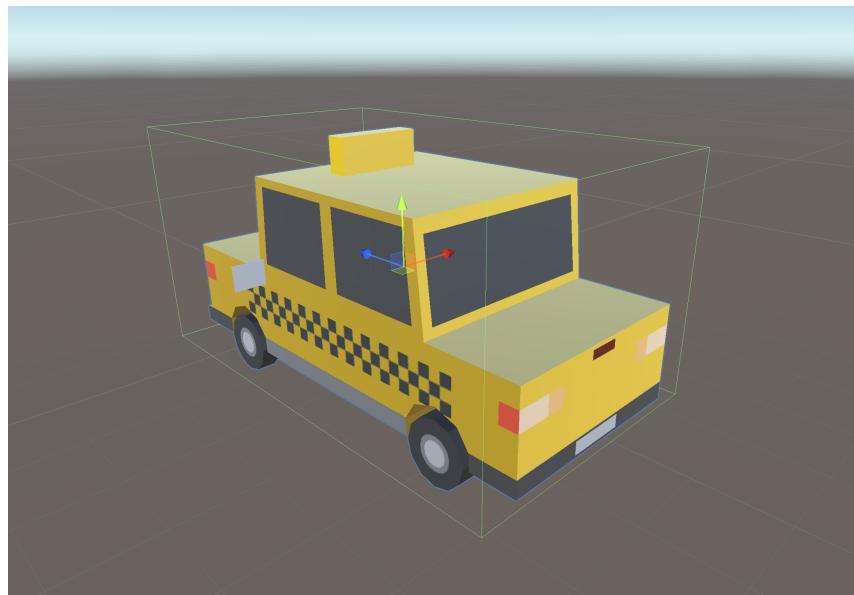
Inspector view 2.5 cho phép ta theo dõi toàn bộ thông tin về một vật thể. Những thông tin này rất đa dạng tùy thuộc vào từng kiểu vật thể mà ta muốn theo dõi, trong đó có những thông tin mà vật thể nào cũng có như vị trí, độ quay, tỉ lệ. Một số thông tin khác có thể có liên quan đến các yếu tố vật lý, cấu tạo màu sắc của vật thể. Bên cạnh đó thì inspector view cũng cho ta thông tin và khả năng thay đổi các thông tin được lấy từ trong các mã nguồn kịch bản gắn với vật thể.

f, Hệ thống mã nguồn xây dựng hành động

Để có thể sử dụng Unity như một công cụ xây dựng môi trường, ta cần sử dụng hệ thống mã nguồn để lập trình từng hành động của các yếu tố trong môi trường. Các thao tác này được thực hiện nhờ vào hệ thống xây dựng các hành động của vật thể thông qua mã nguồn trong Unity.

Trong Unity, mỗi vật thể đều có thể được gắn một thành phần cấu tạo là một hoặc nhiều tập mã nguồn C#. Mã nguồn này sẽ quản lý các hành vi của vật trong

**Hình 2.5:** Inspector View trong Unity



Hình 2.6: Collider của một cá thể xe taxi

môi trường theo yêu cầu của người lập trình. Trong đó, vật thể của ta sẽ kế thừa lớp MonoBehavior, đây là một lớp được thiết kế để thực hiện các công việc như hiển thị vật, thực hiện các sự thay đổi của vật do ta lập trình. Để cho lập trình viên có khả năng này, lớp này cũng có một số hàm mà ta có thể ghi đè để tùy biến hành vi của vật thể.

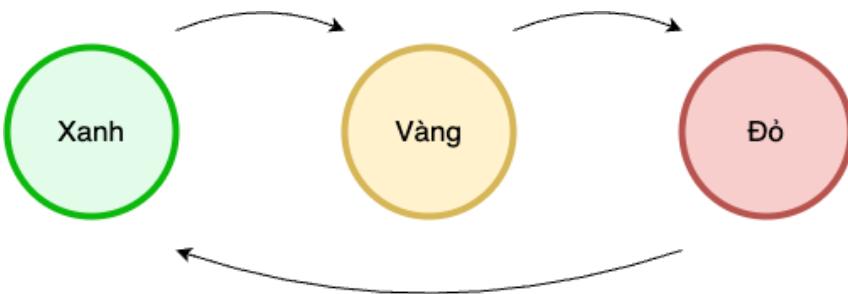
- **Update:** Hàm được gọi liên tục sau đã hoàn thành các thao tác của lần gọi trước. Số lần mà hàm được gọi trong một khoảng thời gian phụ thuộc vào hiệu năng của máy và hiệu năng của phần chương trình được viết trong hàm.
- **FixedUpdate:** Hàm được gọi liên tục với một tần suất cố định. Giá trị mặc định cho tần suất này là 50 lần một giây và có thể được thay đổi bởi người dùng. Do đó hàm này có số lần gọi cố định trong một giây, điều này giúp ta có khả năng tính toán một cách chính xác hơn.
- **Start:** Hàm được gọi khi object được khởi tạo, cho ta khả năng thay đổi các thông tin trước khi hàm Update và FixedUpdate được gọi.

Bên cạnh đó thì Unity cung cấp một số API hữu dụng khác để thao tác với các object phục vụ các mục đích khác nhau như tạo vật thể và xoá vật thể khỏi môi trường.

2.6.2 Một số cơ chế thường dùng trong Unity

a, Nhận diện va chạm

Nhận diện va chạm là nhận biết khi có hai vật thể lấn chiếm lên nhau. Điều này xảy ra khi hai vật có vị trí gần nhau hơn là mức kích thước của chúng cho phép. Trong Unity, cơ chế này thường được lập trình theo hướng sử dụng hệ thống khung



Hình 2.7: Máy trạng thái hữu hạn cho hệ thống đèn giao thông

va chạm (collider) của Unity.

Khung va chạm (hình 2.6) có thể hiểu là một bộ khung do người dùng tự định nghĩa cho mỗi vật thể, đây là yếu tố có thể thêm vào hoặc không tùy thuộc vào mục đích của người lập trình viên đối với từng vật thể. Để sử dụng cơ chế này, Unity cung cấp API `OnCollisionEnter` trong mã nguồn của mỗi vật thể. Hàm này cho phép ta thao tác với khung va chạm của vật thể mà đang va chạm với khung của bản thân. Từ đó, ta có thể lấy được hầu hết các thông tin của vật như vị trí, độ quay, tỉ lệ và các thuộc tính công khai khác của vật thể va chạm.

b, Máy trạng thái hữu hạn

Máy trạng thái hữu hạn (Finite State Machine [22]) (FSM) là một cách thiết kế mô hình thể hiện cách mà một vật thể trong môi trường sẽ hoạt động. Trong đó, vật thể này sẽ hoạt động dựa vào một số hữu hạn các trạng thái khác nhau, trong đó, tại mỗi khung thời gian, vật thể này sẽ đổi chiều giá trị trạng thái hiện tại và với mỗi trạng thái sẽ đưa ra hành động cụ thể. Bên cạnh đó, việc xử lý thông tin để thay đổi trạng thái hợp lý hoàn toàn được tách biệt với quá trình hành động.

FSM có thể được áp dụng vào việc thiết kế hệ thống lớn như quản lý môi trường, hay cũng có thể sử dụng trong từng object nhỏ trong môi trường như là mỗi xe. Một ví dụ về FSM được thể hiện trong hình 2.7 với các trạng thái là xanh, vàng và đỏ. Trong đó, mỗi trạng thái sẽ có cơ chế chuyển trạng thái khác nhau, ví dụ từ đèn xanh khi hết thời gian sẽ chuyển sang vàng, khi hết thời gian vàng thì chuyển qua đỏ. Bên cạnh đó, một số trạng thái nhất định sẽ chỉ chuyển được tới một vài trạng thái nhất định khác, ví dụ như đỏ không thể chuyển thành vàng. Một điểm quan trọng khác là cơ chế tác động của đèn lên môi trường không phụ thuộc vào phần cơ chế thay đổi trạng thái. Tại mỗi khung thời gian của môi trường, hệ thống chỉ kiểm tra trạng thái hiện tại và hoạt động theo logic của trạng thái đó.

c, Hệ thống đếm giờ

Hệ thống đếm giờ là phương pháp được sử dụng khá nhiều trong Unity. Phương pháp này được sử dụng khi ta cần thực hiện một việc nào đó sau một khoảng thời

gian cho trước. Các ví dụ có thể kể đến như các đồng hồ đếm giờ, đồng hồ đếm ngược, hay hệ thống điểm thưởng theo giờ,... Chính vì việc có nhiều trường hợp sử dụng như vậy, Unity cung cấp cho ta một số giải pháp liên quan đến tính toán thời gian, một giải pháp được sử dụng nhiều nhất đó là API `Time.deltaTime`. Đây là một phương thức tĩnh của lớp `Time`, cho phép ta trả ra khoảng cách giữa khung thời gian hiện tại và ngay trước đó, từ đó ta có thể sử dụng giá trị này phù hợp với mục đích của mình.

Với ví dụ về việc sử dụng hệ thống đếm giờ trong việc sinh xe trong một khoảng thời gian. Ta sử dụng một biến để tích trữ thời gian, khởi đầu là 0, sau đó tại mỗi khung thời gian ta tiến hành cộng nó với giá trị `Time.deltaTime`, và kiểm tra xem nó đã vượt qua khoảng thời gian mà ta mong muốn chưa, nếu đã vượt qua thì ta tiến hành thực hiện hành động cần thiết và chuyển lại giá trị tích trữ về 0.

d, Hệ thống Prefab

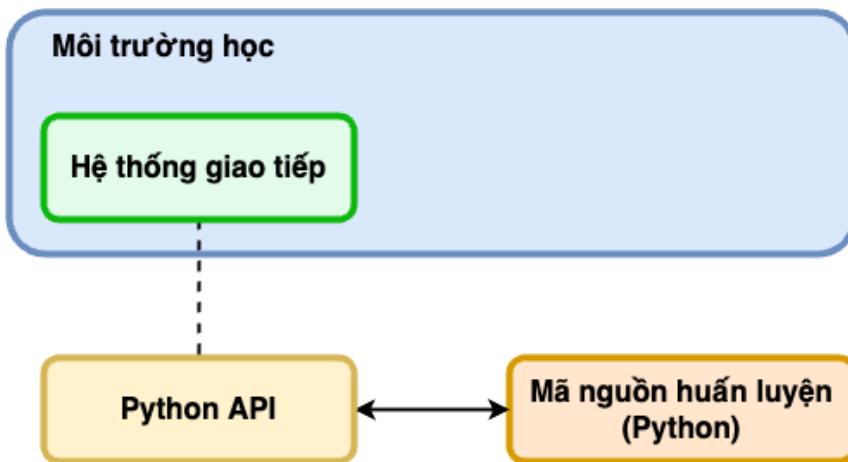
Prefab là các vật thể được tạo trước, chính vì vậy hệ thống prefab ở đây là một hệ thống cho phép ta tạo và quản lý các vật thể một cách bài bản. Có thể hiểu prefab như một bản mẫu cho một object được cấu tạo một cách hoàn chỉnh gồm bất kỳ yếu tố nào mà lập trình viên mong muốn. Các prefab này sau đó được chuyển thành các vật thể trong môi trường thông qua việc người dùng thêm chúng vào trong scene view hoặc thông qua việc tạo chúng ở trong mã nguồn.

Đối với việc tạo vật thể trong scene view, hệ thống này cho ta quyền quản lý các vật thể này thông qua prefab của chúng. Ta có thể thay đổi một số thuộc tính trong mỗi vật thể trong môi trường một cách khác nhau, khiến chúng khác đi so với prefab. Tuy nhiên khi cần, ta có thể thay đổi một số thông tin trong prefab vào áp dụng chúng vào toàn bộ các vật thể đang có trong môi trường. Việc này giống với cơ chế kế thừa trong lập trình hướng đối tượng.

Để tạo các vật thể trong môi trường từ những prefab này thông qua lập trình mã nguồn, Unity cung cấp API `Instantiate`. API này nhận vào một prefab và một số thông tin như vị trí tạo, độ quay, và vật thể cha, sau đó tạo vật thể này trong môi trường và trả về chính nó. Từ đó, ta có thể thay đổi một số thuộc tính công khai của vật thể phù hợp với mục đích của mình. Cơ chế tạo vật thể cho phép ta tự động hóa quá trình tạo vật thể trong môi trường, rất hữu dụng trong một số trường hợp cụ thể.

2.7 Unity ML-Agents

Bộ công cụ Unity ML-Agents [23] cung cấp một SDK được phát triển bởi Unity và được sử dụng trong các môi trường Unity để tạo ra một môi trường học máy. Bộ công cụ cho phép ta tạo ra một môi trường mô phỏng trong Unity có các yếu tố để



Hình 2.8: Sự tương tác của các thành phần trong bộ công cụ ML-Agents

hỗ trợ cho việc học máy, bên cạnh đó cũng cung cấp cho ta một hệ thống giao tiếp với môi trường này thông qua các Python API. Các thành phần chính trong Unity ML-Agents là cảm biến, các chủ thể hành động (agent), và một hệ thống học viện (academy).

Unity ML-Agents biến một môi trường thông thường trong Unity trở thành một môi trường học máy. Trong đó, các agent hoạt động thông qua các bước thời gian. Trong mỗi bước thời gian, từng agent sẽ thu thập các thông tin về môi trường và sử dụng chúng và policy tương ứng để tìm ra hành động tối ưu cho trạng thái hiện tại, đồng thời nhận về điểm thưởng cho từng hành động. Việc biến đổi các vật thể thông thường thành các agent được thực hiện thông qua việc kế thừa lớp Agent được cung cấp bởi bộ công cụ. Lớp Agent cung cấp cho ta các chức năng mà lớp MonoBehavior có, đồng thời cung cấp thêm các logic liên quan đến môi trường học máy và một số phương thức giúp ta tuỳ chỉnh phương thức tương tác của agent với môi trường. Một số phương thức quan trọng mà ta cần thực hiện ghi đè để hoàn thiện mã nguồn Agent gồm có:

- `OnEpisodeBegin`: Cho phép lập trình viên tùy biến các giá trị hay các hoạt động mỗi khi tập của môi trường được bắt đầu.
- `ResetEnv`: Cho phép ta kết thúc tập hiện tại, có thể sử dụng kèm một số điều kiện nhất định để đạt được các cơ chế hoạt động mong muốn.
- `CollectObservations`: Cho phép ta thiết lập các thông tin mà agent nhận được trong mỗi lần yêu cầu hành động, ở đây ta có thể hiểu là tương ứng với trạng thái s .
- `OnActionReceived`: Trả về cho ta hành động của agent và cho phép ta thực hiện thay đổi môi trường tương ứng với hành động đó.

Bên cạnh đó, ta cũng được quyền thay đổi một số thông tin liên quan đến hành vi của agent thông qua một số thuộc tính công khai của agent trong Inspector View 2.5. Trong đó có một số thông tin quan trọng như:

- Kích thước số chiều của các quan sát về môi trường.
- Kích thước số chiều của hành động.
- Số bước tối đa - số bước thời gian tối đa trong một tập môi trường, khi bước thời gian đạt giới hạn này, môi trường sẽ tự động thực hiện ResetEnv mà không cần chờ điều kiện mà lập trình viên thiết kế.

Hệ thống học viện (academy) sẽ là thành phần thực hiện quản lý các agent. Các chức năng của học viện bao gồm theo dõi các bước thời gian cho từng agent và quản lý các thông tin liên quan đến agent. Trong quá trình huấn luyện, học viện sẽ đóng vai trò giao tiếp với các mã nguồn huấn luyện trong Python và chạy một loạt các tập của môi trường khác nhau. Trong mỗi tập này sẽ bao gồm nhiều bước thời gian, tại đó học viện tổng hợp các quan sát từ agent và gửi qua mã nguồn huấn luyện, nhận lại các hành động tương ứng và gửi chúng lại cho từng agent.

Các mã nguồn huấn luyện trong Python không phải là một phần của môi trường. Các mã nguồn này chứa các thuật toán học máy khác nhau để có thể huấn luyện agent trong môi trường học. Python API là một giao thức giữa ứng dụng huấn luyện và hệ thống giao tiếp. Hệ thống giao tiếp nằm ở trong môi trường và có tác dụng kết nối môi trường này tới Python API.

Các thành phần trong bộ công cụ ML-Agents được thể hiện trong hình 2.8.

2.8 Kết chương

Trong chương này, em đã trình bày về các lý thuyết và kỹ thuật cần có để có thể thực hiện giải pháp được đề xuất. Các phương pháp tối ưu dựa vào mô hình có điểm mạnh là sự đơn giản nhưng lại phụ thuộc hoàn toàn vào môi trường bài toán. Phương pháp tiên tiến hơn là Deep Q Network đã khắc phục điểm yếu trên nhưng vẫn tồn tại một số vấn đề về tính tối ưu do không trực tiếp tối ưu hoá policy. Phương pháp Proximal Policy Optimization thể hiện khả năng tối ưu tốt và tốc độ hội tụ nhanh, phù hợp với phần lớn các bài toán hiện nay. Bên cạnh đó, ta cũng tìm hiểu về các kỹ thuật cơ bản trong Unity và gói công cụ Unity ML-Agent, từ đó giúp ta có khả năng phát triển một môi trường hoàn thiện và kết nối tới các thuật toán huấn luyện. Những kiến thức này sẽ được sử dụng trong chương tiếp theo để phát triển và giải quyết bài toán đề ra.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1 Tổng quan

Với những kiến thức nền tảng được trình bày từ Chương 2, em tiến hành áp dụng chúng vào việc phát triển môi trường và sử dụng phương pháp Proximal Policy Optimization để giải bài toán trên môi trường đã được xây dựng. Trong chương này, em sẽ trình bày về các gói hỗ trợ, thư viện được sử dụng, quá trình phát triển môi trường và ứng dụng PPO để tìm được một policy tối ưu cho môi trường đó.

3.2 Tổng quan giải pháp

Trong đồ án này, em đưa ra giải pháp sử dụng Unity và bộ công cụ ML-Agents để xây dựng một môi trường học cho bài toán hệ thống đèn giao thông thông minh. Em sử dụng phương pháp học tăng cường và cụ thể là giải thuật Proximal Policy Gradient, tương tác với môi trường đã tạo thông qua các Python API được cung cấp, từ đó xây dựng được một policy hợp lý và tối ưu, giúp cho agent đưa ra được các quyết định về tín hiệu đèn giao thông một cách hợp lý.

3.3 Xây dựng ngữ cảnh môi trường trong Unity

3.3.1 Các gói hỗ trợ

a, Mô hình đồ họa đơn giản

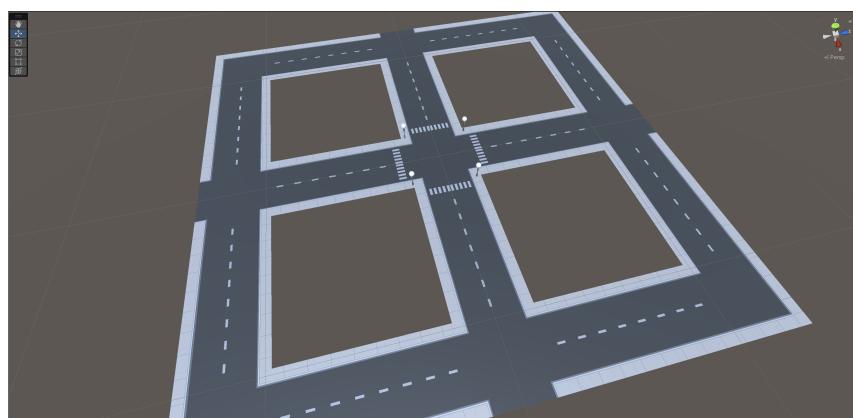
Để có thể phát triển được một môi trường chân thực, giống với thực tế, em sử dụng gói đồ họa giao thông đơn giản được mua ở trên hệ thống cửa hàng của Unity. Gói này cung cấp cho ta các mô hình đồ họa chân thực liên quan đến giao thông như đường, vạch phân cách, xe cộ,... Với các mô hình đồ họa có sẵn này, ta có thể bỏ qua bước thiết kế đồ họa để giảm thiểu được thời gian phát triển môi trường, phù hợp với phạm vi của một đồ án. Một số các mô hình đồ họa được gói này hỗ trợ được thể hiện ở hình 3.1.

b, Hệ thống tạo đường đi

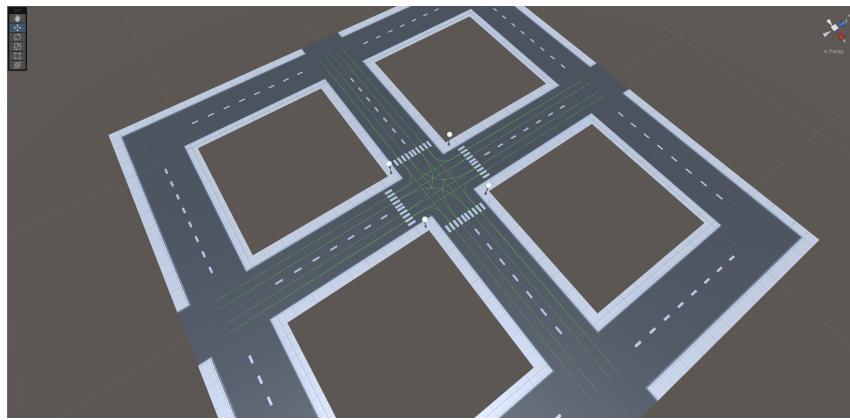
Để có thể định hướng di chuyển của các phương tiện trong môi trường một cách chính xác và giống với thực tế, em sử dụng gói hỗ trợ tạo đường đi là Path Creator. Tác dụng của gói hỗ trợ này là tạo cho ta các đường cong và cung cấp cho ta một số API để lấy được điểm nằm trên đường cong trên gần nhất với một điểm mà ta cần tìm. Từ đó ta có thể sử dụng thông tin này vào việc di chuyển của các phương tiện, giúp các phương tiện của ta có thể di chuyển trong môi trường một cách hợp lý, góp phần giúp cho môi trường trở nên thêm chân thực và thể hiện được đúng các tính chất của việc di chuyển ngoài đời thực.



Hình 3.1: Gói đồ họa giao thông đơn giản



Hình 3.2: Nền của môi trường ngã tư đơn giản



Hình 3.3: Hệ thống lộ trình

3.3.2 Nền của môi trường

Với bài toán đặt ra là hệ thống đèn giao thông tại một ngã tư đơn giản, em đã xây dựng một ngã tư có với 4 hướng đường hai chiều, trong đó mỗi chiều có 2 làn đường xe chạy. Cụ thể phần nền của môi trường được trình bày như hình 3.2. Như có thể thấy trong hình, môi trường sẽ bao gồm 4 hướng đi giao nhau tại một ngã tư. Ở đây, mỗi hướng đi sẽ có một đèn giao thông tương ứng quyết định việc có cho xe di chuyển hay không. Hệ thống bao gồm 4 đèn giao thông này sẽ đóng vai trò quản lý việc lưu thông của các phương tiện giao thông trong ngã tư này, đồng thời cũng chính là đối tượng mà ta đang hướng tới để tối ưu hoá các quyết định.

Trước hết, em sử dụng các mô hình đường để tạo ra các làn và sau đó là thêm vào các hình giải phân cách để phân chia làn đường. Tiếp đó là thêm vào phần vỉa hè để hoàn thiện mô hình một làn đường đơn giản. Từ đó ta có thể lặp lại các bước trên và xây dựng được mô hình nền đường hoàn chỉnh.

Để các xe có thể di chuyển trên môi trường, ta tạo các đoạn đường lộ trình mà các xe sẽ chọn để đi theo, các đoạn đường này sẽ được lưu lại và trong thao tác tạo ra xe và thêm vào môi trường, các xe đồng thời cũng sẽ được gắn cho một lộ trình cụ thể để đi theo, từ đó đảm bảo được tính đúng đắn của môi trường. Các đoạn đường được thêm vào môi trường được thể hiện bởi các đường màu xanh trong hình 3.3.

3.3.3 Hệ thống phương tiện

Để tạo được một hệ thống phương tiện hợp lý phù hợp với nhu cầu của bài toán, ta cần phải hiểu là đây không thể là các object cố định luôn hiện hữu trong môi trường được, ta cần liên tục sinh ra vật thể và xoá chúng đi sau khi đã đi ra khỏi môi trường. Bên cạnh đó thì mỗi cá thể phương tiện sẽ là yếu tố chịu trách nhiệm cho việc đảm bảo các yêu cầu của môi trường giao thông, các cơ chế tránh va chạm hay dừng đèn đỏ.

a, Hệ thống quản lý phương tiện

Để đảm bảo môi trường có tính bao quát, thể hiện được toàn bộ các tình huống của có thể xảy ra của môi trường thực tế, em sử dụng một số thông số để kiểm soát môi trường.

- Giãn cách tạo xe: Khoảng cách giữa 2 lần tạo xe và thêm vào môi trường, việc này giúp ta có thể kiểm soát được mật độ xe nói chung trên toàn bộ môi trường.
- Thứ tự ưu tiên của các làn: Độ ưu tiên dành cho từng làn đường khác nhau, làn nào có độ ưu tiên cao thì khả năng xe được sinh ra trên làn đó càng cao. Điều này giúp ta có thể tạo được những tình huống khác nhau của giao thông hiện tại.
- Độ chênh lệch của các mức ưu tiên: Bên cạnh thứ tự ưu tiên, ta cũng dùng các tham số để kiểm soát độ chênh lệch của các mức ưu tiên với nhau, kết hợp với sự thay đổi thứ tự ưu tiên giúp ta tạo được một môi trường đa dạng, mô phỏng được các trường hợp có thể xảy ra ngoài thực tế.

Bằng việc tạo ra và kiểm soát, thay đổi các thông số trên liên tục, ta có thể tạo ra một bộ mô phỏng hoàn thiện, ứng với môi trường đề ra. Tương ứng với mỗi tập của môi trường học, hiện tại đang là 5 phút, các thông số trên sẽ được thay đổi sao cho vừa đủ để có thể hoàn thành được một bộ trạng thái. Cụ thể, với mỗi tập của môi trường:

- Giãn cách tạo xe sẽ thay đổi từ 0.5s tới 1.5s. Điều này khiến cho lượng xe trên toàn bộ môi trường được thay đổi. Ở tập tiếp theo của môi trường, sự thay đổi này sẽ ngược lại, tức là từ 1.5s tới 0.5s. Điều này là để đảm bảo hoạt động tốt với các thông số thứ tự ưu tiên lẫn mật độ của các mức ưu tiên.
- Thứ tự ưu tiên giữ nguyên để tạo môi trường với ưu tiên đó, và thay đổi khi ta chuyển qua tập mới của môi trường.
- Độ chênh lệch của các mức ưu tiên sẽ liên tục chuyển từ (0.25, 0.25, 0.25, 0.25) thành (0.55, 0.2, 0.15, 0.10) trong quá trình 5 phút. Tức là trong 5 phút này ta liên tục có các độ chênh lệch khác nhau của môi trường thoả mãn với một thứ tự ưu tiên cố định.

Với phương pháp trên, sau 24 tập của môi trường, ta có được đầy đủ các thứ tự ưu tiên khác nhau của môi trường, và với mỗi thứ tự ưu tiên khác nhau, ta lại có các mật độ khác nhau cho các mức ưu tiên. Kết hợp với giãn cách tạo xe, sau 48 tập của môi trường, ta có thể có được toàn bộ các trạng thái về mặt lưu lượng ở từng làn có thể xuất hiện trong môi trường.

Sau khi đảm bảo được tính bao quát trên, tiếp đến là bước thực hiện sinh xe. Với việc các mô hình xe khác nhau được thiết kế hoàn thiện, sau mỗi khoảng giãn cách được định nghĩa ở trên, ta tiến hành tạo ra xe, đồng thời gắn cho mỗi xe một lộ trình. Việc lựa chọn lộ trình là một quá trình ngẫu nhiên có kiểm soát, được định đoạt thông qua các thông số về thứ tự ưu tiên và mật độ của các mức ưu tiên. Việc lựa chọn lộ trình sẽ giúp cho ta có thể sử dụng được các thông số trên để kiểm soát môi trường.

Để đảm bảo không xảy ra hiện tượng các xe khi sinh ra bị chèn vào nhau, sau khi ta tìm được một lộ trình, ta sẽ tạo ra một điểm ngẫu nhiên gần với đầu của lộ trình, điểm này sẽ được coi là điểm xuất phát. Tuy nhiên, ta cần kiểm tra xem khu vực xung quanh điểm xuất phát này có phương tiện nào đang lấn chiếm không, nếu có ta sẽ cần chọn lại một điểm xuất phát mới. Sau khi đã tìm được một điểm xuất phát đảm bảo có thể sinh xe an toàn, công việc sinh xe mới được diễn ra thông qua cơ chế `Instantiate` của Unity.

Bên cạnh việc liên tục tạo ra các xe, để đảm bảo môi trường được diễn ra trơn tru, ta cũng cần tiến hành xử lý các vấn đề liên quan đến tương tác của các xe với nhau. Ở đây, để đảm bảo được hiệu năng của môi trường, tránh việc mô phỏng quá nặng, sau khi xe đi hết khỏi phần đường trước đèn đỏ, các cơ chế tránh va chạm của xe sẽ được giảm thiểu. Điều này có thể được thực hiện là do đối với bài toán hiện tại, ta chỉ quan tâm về phần di chuyển trước đèn đỏ của xe.Thêm vào đó, ta sẽ tiến hành xoá xe khỏi môi trường khi xe đi tới điểm kết thúc của lộ trình.

b, Hệ thống hoạt động của các phương tiện

Các xe trong môi trường hoạt động theo cơ chế đi theo một lộ trình đã có. Các lộ trình này đã được gán cho mỗi xe trong quá trình tạo ra xe trong môi trường, là yếu tố chính quyết định sự di chuyển của xe. Bên cạnh đó, ta có thêm thông số về vận tốc, giúp ta kiểm soát được tốc độ xe di chuyển theo lộ trình định sẵn.

Để cho việc xe di chuyển trong môi trường một cách hợp lý và đồng thời có được các cơ chế tránh va chạm, các xe sẽ hoạt động theo cơ chế máy trạng thái hữu hạn (FSM). Các trạng thái của xe gồm có:

- DRIVE: di chuyển thông thường với vận tốc tối đa đã định nghĩa.
- RED_LIGHT_STOP: vào trạng thái dừng do gặp đèn đỏ.
- VEHICLE_COLLISION_STOP: vào trạng thái dừng do gặp phương tiện.
- ACCELERATE: vào trạng thái tăng tốc để chuyển từ trạng thái dừng sang di chuyển thông thường.

Quá trình thay đổi trạng thái này được thực hiện thông qua một cảm biến được

gắn ở đầu xe, cảm biến này cho ta biết trong đoạn thẳng trước mặt có vật thể nào không và đồng thời biết được phân loại của object đó. Việc này được thực hiện thông qua hệ thống RaycastHit của Unity, cho phép ta kiểm tra thông tin trên một đoạn thẳng trước mặt. Trong trường hợp có vật cản được nhận diện trong khoảng đoạn thẳng trước mặt, ta có thể biết được khoảng cách của vật cản, đồng thời thay đổi trạng thái của xe nếu các điều kiện tránh va chạm thỏa mãn:

- Nếu khoảng cách đủ nhỏ và phân loại của vật cản phía trước là đèn giao thông thì ta chuyển trạng thái thành RED_LIGHT_STOP.
- Nếu khoảng cách đủ nhỏ và phân loại của vật cản phía trước là phương tiện thì ta chuyển trạng thái thành VEHICLE_COLLISION_STOP.
- Nếu phía trước không có vật cản và trạng thái hiện tại đang là một trạng thái dừng, lúc này có nghĩa là xe đang dừng và cần bắt đầu di chuyển, ta chuyển trạng thái thành ACCELERATE.
- Nếu phía trước không có vật cản và vận tốc hiện tại đang đạt vận tốc tối đa, ta chuyển trạng thái hoặc giữ nguyên trạng thái là DRIVE.

Với các cơ chế trên, ta có thể đảm bảo xe lưu thông theo đúng những luật lệ đề ra, phù hợp với thực tế. Đồng thời ta cũng có thể đảm bảo được quá trình dừng đèn đỏ của xe khi kết hợp với việc bật tắt đèn để tạo ra các vật cản ảo.

Khi xe dừng lại do đèn đỏ hoặc do một xe khác đang chịu tác động của đèn đỏ, ta sẽ lưu lại các giá trị về thời gian dừng, giá trị này sẽ có tác dụng đánh giá độ hiệu quả của tín hiệu đèn giao thông, từ đó đưa ra các mức điểm thưởng phù hợp.

3.3.4 Hệ thống đèn giao thông

Hệ thống đèn giao thông cho mỗi trường một ngã tư đơn giản bao gồm 4 đèn giao thông tương ứng với 4 hướng đi của ngã tư, được kiểm soát bởi một vật thể cha đóng vai trò quản lý hoạt động của 4 đèn này.

Cách hoạt động của hệ thống này là tại các vị trí đèn đỏ, phần khung va chạm của vật thể đèn giao thông đó sẽ được chuyển trạng thái thành bật. Việc này sẽ góp phần tạo thành một vật cản vô hình mà các xe có thể thấy được thông qua cảm biến đặt phía trước xe. Để đạt được điều này, ta thêm mã nguồn C# vào vật thể quản lý hệ thống đèn, cho nó quyền thao tác với 4 đèn giao thông tương ứng. Trong đó, mỗi khi chuyển trạng thái đèn, ta cần thay đổi các thông số của cả 4 đèn trong hệ thống trên. Đồng thời, ta cần có thay đổi để có thể dễ dàng quan sát tín hiệu đèn hơn, do khi đặt hệ thống máy quay lên cao thì mô hình đèn trở nên khá nhỏ và khó quan sát. Để khắc phục vấn đề này, ở mỗi đèn ta thêm một khối cầu đơn giản có tác dụng thể hiện màu đèn. Sự thay đổi màu của khối cầu cùng đồng thời được

thực hiện cùng với việc thay đổi trạng thái đèn, đảm bảo cung cấp được các thông tin chính xác về môi trường.

Các logic trên được chuyển hoá thành một API, từ đó bất kỳ vật thể nào có quyền truy cập API này đều có khả năng thay đổi trạng thái đèn. Điều này giúp cho agent của ta có khả năng thực thi hành động, tạo ra các tác động trên môi trường, từ đó có thể tạo thành một môi trường huấn luyện hoàn chỉnh.

3.4 Cấu hình Markov Decision Process cho môi trường

Cấu hình Markov Decision Process cho môi trường hiện tại bao gồm một số bước cơ bản như định nghĩa một tập (episode) của môi trường, định nghĩa trạng thái (state) của môi trường, định nghĩa hành động (action) của agent, tạo quy luật điểm thưởng. Với nhu cầu của từng bài toán khác nhau, các định nghĩa và quy định trên có thể được thay đổi để phù hợp với bài toán đó. Do đó, việc định nghĩa một cách chi tiết các yếu tố của MDP trong bài toán hiện tại là việc quan trọng hàng đầu.

3.4.1 Tổng quan

Đối với bài toán hiện tại, môi trường được đặt ra là một mô phỏng ngã tư đơn giản với 4 hướng đi, mỗi hướng đi bao gồm 2 làn đường. Giảm cách đưa ra quyết định của agent là 5 giây, giúp cho việc quy định ý nghĩa của hành động được rõ ràng hơn. Ví dụ, ta có thể thấy, nếu ta đưa ra quyết định mỗi khung thời gian với khoảng 50 khung thời gian một giây thì từng quyết định một sẽ không đáng kể và phải lặp rất nhiều lần để tạo thành một chuỗi thời gian đèn xanh hoặc đỏ, điều này ảnh hưởng rất nhiều tới việc học của agent. Trước mỗi khi đưa ra quyết định, agent sẽ nhận được thông tin trạng thái của môi trường. Ở trong bài toán hiện tại, trạng thái của môi trường được thể hiện thông qua thông tin về các cảm biến đặt trên đường, chúng sẽ trả về 1 nếu có xe đang đứng ở trên và 0 trong trường hợp ngược lại. Cùng với đó, thời gian dừng tích luỹ của các phương tiện trong khoảng thời gian 5 giây trước đó được tính vào làm điểm thưởng cho agent cho hành động trước đó.

Để thực hiện các cấu hình này, ta cần thêm vào vật thể quản lý hệ thống đèn giao thông một tệp mã nguồn C# là TrafficLightAgent. Mã nguồn này sẽ kế thừa lớp Agent được cung cấp bởi gói ML-Agents, giúp ta thực hiện việc quản lý môi trường và giao tiếp với agent từ mã nguồn trong python.

3.4.2 Thiết kế từng tập của môi trường

Mỗi tập của môi trường sẽ bao gồm 64 bước. Điều này giúp cho mỗi tập sẽ bao quát được một mức thứ tự ưu tiên của các phần đường, giúp cho việc quản lý

môi trường dễ dàng hơn. Mỗi tập sẽ bắt đầu bằng một môi trường trống không có phương tiện, và từ đây là các phương tiện sẽ được sinh ra theo quy luật đã được trình bày ở trên. Ở đây, do đặc thù của bài toán, ta không có các bước thời gian kết thúc giúp ta kết thúc tập, mà chỉ thực hiện làm mới môi trường mỗi khi đã đạt 64 bước thời gian. Khi được làm mới, môi trường sẽ tiến hành xoá toàn bộ các phương tiện đang hoạt động trong đó, đồng thời đưa các giá trị tích luỹ trở về giá trị mặc định.

Ta tiến hành thay đổi một số thông số của TrafficLightAgent để phù hợp với các thông tin trên. Vì bản thân gói ML-Agents không hỗ trợ gián cách đưa quyết định của môi trường mức 5 giây, ta cần vào phần mã nguồn gốc của gói để thay đổi. Sau đó ta chuyển giá trị DecisionPeriod (tương đương với số khung thời gian của môi trường giữa 2 lần yêu cầu quyết định từ agent) thành 250. Ở đây, môi trường hoạt động với 50FPS, do đó 250 tương ứng với 5 giây thực tế.

Trong mã nguồn của agent, ta tính số lần đưa ra quyết định của agent, sau khi số này đạt 64, ta thực hiện việc làm mới môi trường. Ở đây, ta lấy toàn bộ các vật thể nằm trong vật chứa và tiến hành xoá từng cá thể. Sau đó, ta chuyển các giá trị khác về giá trị khởi tạo của chúng. Sau quá trình này, ta sẽ có một môi trường ở trạng thái bắt đầu, với các thông số về thứ tự ưu tiên, mật độ ưu tiên, mật độ xe khác nhau.

3.4.3 Thiết kế thông tin trạng thái của môi trường

Để lấy được thông tin về mật độ xe của từng làn đường của môi trường, ta sử dụng các cảm biến được đặt dưới đường. Từng cảm biến này sẽ cho ta thông tin về sự tồn tại của các phương tiện tại từng điểm trên môi trường. Ở môi trường cơ bản, em sử dụng 252 cảm biến đặt ở khắp các làn đường đi tới ngã tư (ở đây ta không cần quan tâm tới thông tin về các làn đường sau khi ra khỏi ngã tư). Các thông tin này được lấy thông qua việc kiểm tra sự hiện diện của vật thể ở trong một vùng, được cung cấp thông qua phương thức OverlapBox được Unity cung cấp. Các thông tin này sẽ được chuyển thành một vector rồi gửi tới cho agent để xử lý.

3.4.4 Thiết kế hành động của agent

Đối với môi trường đơn giản, hành động của agent chỉ là các quyết định gián đoạn (discrete) với lựa chọn một hành động tại một thời điểm, ở đây agent sẽ đưa ra giá trị 0 hoặc 1 tương ứng với việc cắp đèn 0 hoặc 1 chuyển thành màu đỏ. Khi đó, sử dụng API ta có từ script gốc của hệ thống đèn giao thông, ta có thể thực hiện hành động này lên môi trường.

3.4.5 Thiết kế hệ thống điểm thưởng cho từng hành động

Điểm thưởng cho agent trong từng hành động sẽ được tích luỹ trong vòng 5 giây sau khi thực hiện hành động đó. Ở đây, điểm thưởng được sử dụng chính là âm của tổng thời gian mà từng phương tiện phải dừng lại từ hệ quả của đèn đỏ. Bên cạnh đó, để đảm bảo tránh trường hợp một phương tiện bất kỳ phải chờ quá lâu, điểm thưởng này cũng được nhân lên cùng với số thời gian mà xe đã chờ. Điều này giúp cho ta cân bằng được việc tối ưu tổng thời gian chờ cho toàn bộ xe cũng như không để từng xe phải chờ quá lâu.

Để tích luỹ được điểm thưởng cho toàn bộ xe trong môi trường, việc cập nhật điểm thưởng phải diễn ra trên từng xe chứ không phải trong bản thân agent. Do đó, trong mỗi khung thời gian, từng xe sẽ có cơ chế tích luỹ thời gian chờ và thêm nó vào trong điểm thưởng của môi trường. Như vậy, con số này trong agent sẽ được tích luỹ liên tục trong quá trình 5 giây, tăng được sự chính xác cho việc đánh giá.

3.5 Tạo cơ chế hành động cơ sở của bài toán

Để đánh giá hiệu năng của các phương pháp được sử dụng, trước hết ta cần có một bản mô phỏng hệ thống đèn giao thông chưa tối ưu và đánh giá nó để làm mức hiệu năng gốc, từ đó ta mới đánh giá hiệu năng của các phương pháp khác so với mức gốc đó. Như đã đề cập, nguyên lý hoạt động của hệ thống đèn giao thông chưa tối ưu hiện tại là sử dụng các dữ liệu về lưu lượng giao thông trung bình trong thời gian dài để tìm ra được một bộ số chia thời gian phù hợp với các tình huống trung bình của môi trường. Việc tính toán này có thể chia thành nhiều quãng thời gian trong ngày, ví dụ như buổi sáng hoặc buổi chiều có thể có các mốc thời gian khác nhau. Từ cơ sở này, em xây dựng hệ thống mô phỏng lại nguyên lý hoạt động trên.

Do môi trường có sự thay đổi liên tục về thứ tự ưu tiên của các làn cũng như mật độ xe trên từng mức độ ưu tiên, để việc thiết lập một hiệu năng cơ sở được hợp lý nhất, em đã xây dựng các luật chia thời gian cụ thể cho từng giai đoạn của môi trường. Cụ thể, để tìm được giá trị tỉ lệ xe của 2 cặp hướng đi vuông góc nhau ở ngã tư, ta lấy giá trị trung bình mật độ của 2 cặp hướng đi chia cho nhau. Ví dụ với thứ tự ưu tiên là (0, 1, 2, 3) tương ứng với 4 hướng đi mà ở đây (0, 2) và (1, 3) vuông góc với nhau, ta lấy giá trị mật độ ưu tiên trung bình của 2 cặp này để tìm ra tỉ lệ hợp lý là $\frac{0.55+0.25+0.15+0.25}{0.20+0.25+0.10+0.25} = 1.5$. Như vậy, trong trường hợp này, tỉ lệ mật độ xe trung bình của hai phần đường tương ứng tín hiệu giao thông ngược nhau là 1.5. Ta có thể sử dụng tỉ lệ này như một sự tham khảo cho việc tạo ra một bộ số thời gian đèn một cách hợp lý. Tất nhiên, với chỉ tỉ lệ trên thì ta không đủ để xây dựng một bộ hệ số tối ưu, do đó, em đã thử nhiều bộ số thời gian khác nhau để tìm ra được bộ số tối ưu nhất cho các đoạn đường. Với bài toán này, em chọn quy luật

phân chia thời gian như sau:

- Đối với 2 phía đường không được ưu tiên về mặt tạo xe: $2.5 + \text{ratio} \times 5$, tương ứng với 7.5 giây đối với ví dụ trên do ratio ở đây là 1.
- Đối với 2 phía đường được ưu tiên về mặt tạo xe: $2.5 + \text{ratio} \times 5$, tương ứng với 10s đối với ví dụ trên do ratio ở đây là 1.5.

3.6 Huấn luyện mô hình giải bài toán

Sử dụng phương pháp Proximal Policy Gradient, em tiến hành huấn luyện agent để tối ưu hoá policy của nó để giải được bài toán đề ra.

Trong đó, policy của bài toán được thể hiện thông qua một mạng neuron cơ bản với đầu vào là 252 đơn vị tương ứng với 252 giá trị thông tin nhận được từ các cảm biến (ở đây chính là trạng thái s của môi trường). Từ 252 node này, mạng sẽ bao gồm 5 lớp ẩn với 128 đơn vị ẩn ở mỗi lớp. Đầu ra của mạng là 2 đơn vị tương ứng với giá trị chọn từng hành động (ở đây chính là chọn tín hiệu cho từng cặp đèn đối xứng).

3.6.1 Tổng quan

Trong quá trình huấn luyện, agent và môi trường sẽ được chạy liên tục nhiều lần để tích luỹ được lượng dữ liệu dưới policy hiện tại, lượng dữ liệu này sẽ được lưu lại trong bộ nhớ. Sau đó khi lượng dữ liệu này đủ lớn, ta tiến hành sử dụng các thông tin trên để cập nhật policy hiện tại. Ở đây, như đã trình bày trong phần lý thuyết, thuật toán PPO cho phép ta sử dụng dữ liệu này cho việc cập nhật policy này nhiều lần, miễn là số lần thực hiện đảm bảo policy không thay đổi quá xa so với policy gốc. Với quá trình như vậy, sau một khoảng thời gian đủ lớn và với những cấu hình phù hợp của môi trường cũng như những tham số của việc huấn luyện, agent sẽ học được một policy hợp lý và giải quyết được môi trường một cách tối ưu.

3.6.2 Quá trình tích luỹ thông tin

Ta tạo một bộ nhớ để agent có thể lưu trữ được các thông tin trên. Trong đó, tại mỗi bước thời gian mà agent trải qua, ta lưu lại các giá trị:

- s_t - trạng thái của môi trường.
 - a_t - hành động được lựa chọn.
 - R_t - điểm thưởng nhận được do hệ quả của hành động a .
 - s_{t+1} - trạng thái tiếp theo của môi trường
 - $\pi(a_t|s_t)$ - xác suất chọn hành động a_t trong trạng thái s_t với policy hiện tại.
- Thông tin này sẽ được sử dụng trong việc tính ra tỉ lệ giữa hai policy trong quá trình tính toán hàm mục tiêu của PPO.

- Giá trị thể hiện bước thời gian hiện tại có phải bước kết thúc hay không.

Các thông tin này sẽ được lưu lại dưới dạng một bộ dữ liệu, khi số lượng đủ lớn, ta tiến hành lấy mẫu các thông tin từ đây theo từng bước thời gian để sử dụng cho việc cập nhập mô hình.

3.6.3 Huấn luyện mô hình dựa trên dữ liệu đã thu được

Với các dữ liệu đã thu được trong quá trình tương tác với môi trường, agent sẽ sử dụng các dữ liệu này để tiến hành cập nhật policy của mình. Các thông tin ta có là chuỗi các bước thời gian mà ở mỗi bước ta có trạng thái môi trường, hành động được lựa chọn, điểm thưởng cho hành động, trạng thái tiếp theo, xác suất chọn hành động với policy hiện tại.

Trước hết, sau khi có một chuỗi các bước thời gian, ta có thể sử dụng phương pháp Monte Carlo chuẩn hoá giá trị điểm thưởng tại mỗi bước thời gian. Như đã trình bày trong phần lý thuyết, việc này sẽ giúp giá trị tại mỗi bước thời gian có tính liên kết hơn và tính toán một cách chính xác hơn cho MDP. Thông tin này sẽ được sử dụng thay cho các thông tin điểm thưởng thông thường của môi trường.

Sau đó, ta tiến hành cập nhật policy dựa vào các thông tin đã có. Ở đây, vì ta sử dụng phương pháp PPO, ta có thể thực hiện nhiều lần cập nhật với bộ dữ liệu. Cụ thể, tại mỗi lần thực hiện cập nhật (tương ứng với mỗi epoch), ta tiến hành tính toán:

1. Tính toán các giá trị xác suất chọn hành động a_t tại trạng thái s_t với policy mới nhất. Từ đó tính được tỉ lệ giữa policy cũ và mới.
2. Sử dụng hàm giá trị để tính toán giá trị của các trạng thái trong tập dữ liệu.
3. Tính toán giá trị lợi thế tại từng bước thời gian bằng cách lấy giá trị điểm thưởng đã được chuẩn hoá (qua Monte Carlo) trừ đi giá trị của trạng thái được tính từ hàm giá trị.
4. Tính toán hàm măt mát của PPO và tiến hành tối ưu hoá policy.

Ở đây, cụ thể trong hàm măt mát của PPO sẽ bao gồm 2 phần, măt mát cho hàm mục tiêu của PPO và măt mát cho hàm giá trị.

- Hàm mục tiêu của PPO là giá trị min của hai phần là tỉ lệ giữa 2 policy nhân với giá trị lợi thế và phiên bản bị chặn của tích trên.
- Măt mát của hàm giá trị được tính toán dựa trên mục tiêu là giá trị điểm thưởng đã được chuẩn hoá (nhận được từ Monte Carlo).

Sau khi có 2 hàm măt mát trên, ta sử dụng phương pháp tối ưu Adam để tiến hành tối ưu hoá policy cũng như hàm giá trị.

3.7 Kết chương

Trong chương này, em đã trình bày về quá trình phát triển môi trường và huấn luyện mô hình giải quyết bài toán. Trong đó, em đã trình bày về các bước cơ sở như phát triển nền môi trường, thêm các logic về hệ thống đèn giao thông và hệ thống quản lý phương tiện. Sau đó, môi trường được phát triển thành một quá trình quyết định Markov đóng vai trò là môi trường học cho agent. Từ đó, ta tiến hành huấn luyện agent và tìm được policy tối ưu cho bài toán.

CHƯƠNG 4. ĐÁNH GIÁ THỰC NGHIỆM

4.1 Tổng quan

Với môi trường và thuật toán đã được phát triển trong chương 3, em tiến hành thử nghiệm huấn luyện mô hình với một số cấu hình khác nhau của môi trường và giải thuật. Các thử nghiệm này được thực hiện theo hướng phát triển mô hình và tối ưu hoá số lượng cảm biến và kích thước của mô hình, từ đó giảm thiểu chi phí cho giải pháp. Trong chương này, em sẽ trình bày về các cấu hình được sử dụng cho việc thử nghiệm và các kết quả đạt được. Bên cạnh đó, em cũng đưa ra một số đánh giá về các kết quả và giá trị của chúng trong việc giải quyết bài toán.

4.2 Mục tiêu đánh giá

Đối với bài toán đặt ra và giải pháp đã trình bày, mục tiêu đánh giá của em trong đồ án này như sau:

- Thử nghiệm các cấu hình khác nhau của môi trường.
- Thử nghiệm các độ lớn khác nhau của mô hình.
- Đánh giá vai trò của phương pháp học tăng cường trong việc giải bài toán đề ra.
- Đánh giá khả năng ứng dụng của giải pháp trong thực tế.

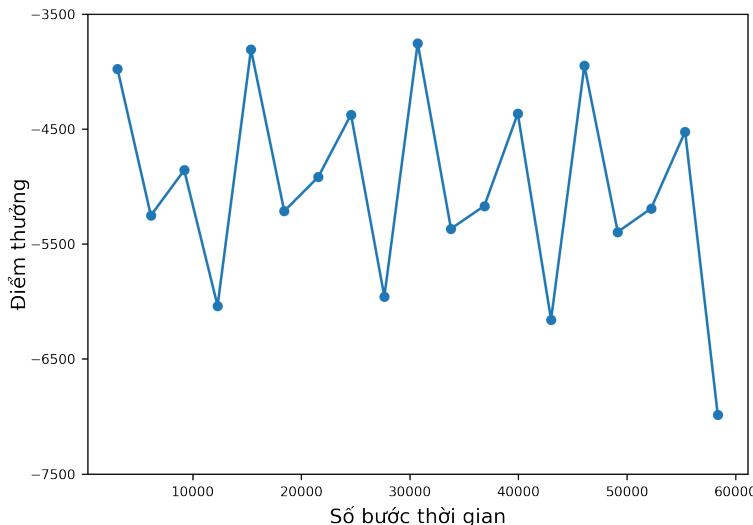
4.3 Các độ đo đánh giá

Để đánh giá hiệu năng của các phương pháp trên với nhau cũng như với hiệu năng cơ sở, em sử dụng một số độ đo đánh giá như sau:

- Thời gian chờ trung bình: Giá trị trung bình mỗi xe chờ đèn đỏ.
- Chiều dài hàng đợi: Chiều dài của một dãy xe chờ tín hiệu đèn chuyển qua màu xanh.

Thời gian chờ trung bình cho phép ta nhìn vào hiệu năng của các phương pháp đối với một mục tiêu quan trọng của bài toán là giảm thiểu thời gian mà mọi người tham gia giao thông phải chờ đợi tín hiệu đèn. Độ đo đánh giá giúp ta có thể ước lượng được các giá trị mà giải pháp mang lại.

Chiều dài hàng chờ cũng là một độ đo đánh giá quan trọng. Khi chiều dài dãy các xe đang chờ đèn đỏ càng lớn có nghĩa là số lượng các xe cùng dừng tại một điểm đèn đỏ càng lớn. Điều này gây ra sự trì hoãn khi chuỗi xe nhận được tín hiệu đèn xanh để đi, do các xe đều có khoảng thời gian chờ xe phía trước cách xa trước khi nhấn ga. Bên cạnh đó, khi chuỗi xe dài không được giải phóng cùng lúc sẽ gây

**Hình 4.1:** Hiệu năng cơ sở

ra tình trạng tăng tốc rồi giảm tốc liên tục, kết hợp với sự trì hoãn đã giải thích ở trên, khiến cho đoạn đường càng thêm tắc nghẽn, cùng với đó là thảm ra nhiều loại khí thải có hại cho sức khoẻ và môi trường.

4.4 Phương pháp thí nghiệm

Để đánh giá hiệu năng của từng phương pháp, em cấu hình môi trường phù hợp với việc đánh giá. Một số thông số sau được cố định:

- Tổng số bước thời gian cho mỗi lần sinh giá trị đánh giá là 3072 bước. Điều này đảm bảo bao quát được hết toàn bộ các trường hợp của môi trường. Ở đây giá trị 3072 được tính thông qua $24 \times 64 \times 2$ (Số lượng các thứ tự ưu tiên khác nhau) \times (Số lượng bước thời gian trong một giai đoạn của thứ tự ưu tiên) \times (Hai chiều tăng giảm của giãn cách tạo xe).
- Với mỗi phương pháp, thực hiện tính toán các giá trị độ đo đánh giá đối với bội số đủ lớn của 3072 bước thời gian để tính toán giá trị trung bình một cách chính xác.
- Môi trường sử dụng hạt giống cố định cho các hàm số ngẫu nhiên. Điều này giúp ta đảm bảo được môi trường giống nhau giữa các lần đánh giá, tăng được sự công bằng cho các phương pháp.

Bên cạnh đó, môi trường được xây dựng thành một ứng dụng, nâng cao hiệu năng và độ chính xác cho đánh giá.

4.5 Hiệu năng cơ sở

Với môi trường được thiết kế cho việc đưa ra các quyết định giống với cơ chế hiện tại ở ngoài thực tế, sau quá trình đánh giá, hệ thống đạt được hiệu năng không tốt và không ổn định được thể hiện trong hình 4.1. Như ta có thể thấy, tổng điểm thưởng trong mỗi tập của môi trường có sự khác biệt khá nhiều. Cho thấy điểm hạn chế không thể thích ứng với các trạng thái khác nhau của môi trường, trong đó ta sẽ có những giai đoạn mà hệ thống này giải quyết tốt, nhưng cũng có những giai đoạn mà hiệu năng của hệ thống là rất tệ. Giá trị điểm thưởng trung bình đạt được -5013.

4.6 Quá trình huấn luyện mô hình

4.6.1 Thiết lập siêu tham số cho thuật toán PPO

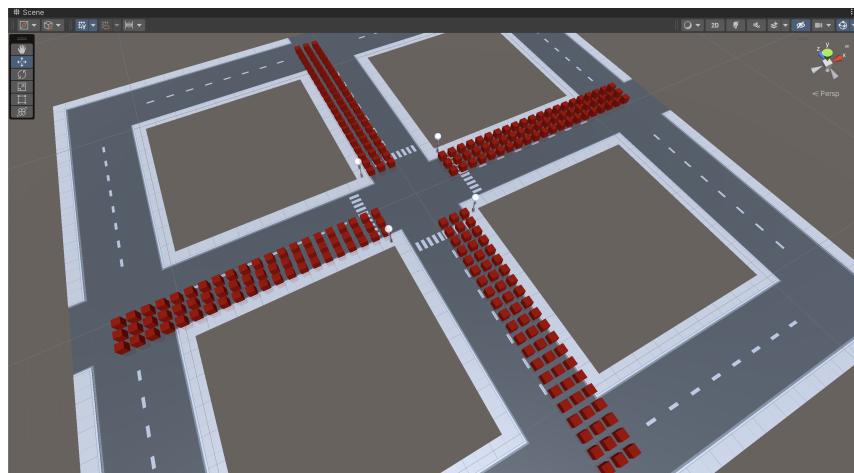
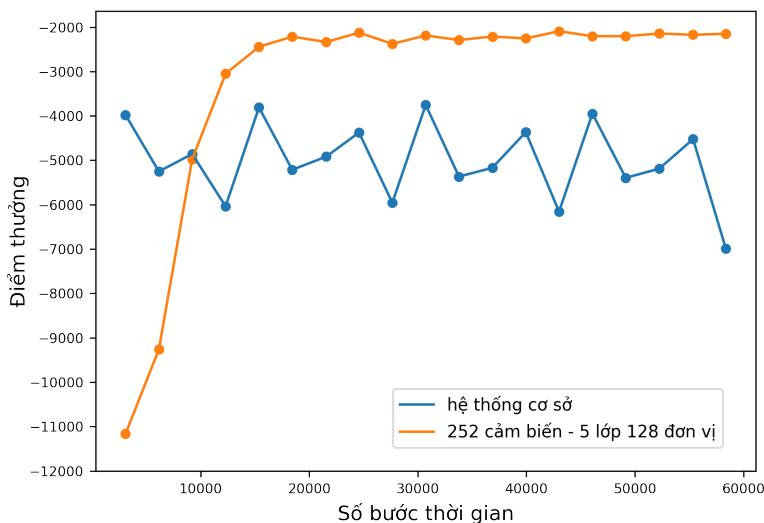
Để huấn luyện mô hình, trước tiên em thử một hệ thống siêu tham số cơ bản cho agent và phát triển từ đó đối với các kết quả đạt được khác nhau. Các hệ số cần chú ý gồm có:

- buffer_size = 512: Số lượng bước thời gian cần thu thập trước khi thực hiện bước cập nhật policy.
- batch_size = 64: Số lượng bước thời gian được sử dụng cho một lần tính toán gradient trong bước tối ưu hoá policy.
- learning_rate= 3e-5: Tốc độ học, kiểm soát tốc độ thay đổi của mô hình trong mỗi lần cập nhật.
- num_layers = 5: Số lớp ẩn trong mạng neuron sử dụng.
- hidden_units = 128: Số đơn vị ẩn trong một lớp ẩn trong mạng neuron sử dụng.
- epsilon = 0.2: Giá trị ϵ trong hàm mục tiêu của phương pháp PPO. Giá trị 0.2 được tác giả khuyến nghị cho hầu hết bài toán.
- gamma = 0.95: Hệ số chiết khấu, thể hiện sự thuyên giảm giá trị điểm thưởng nhận được trong tương lai, tính cho thời điểm hiện tại.

Với hệ thống siêu tham số như trên, ta có thể tiến hành huấn luyện mô hình và sử dụng kết quả huấn luyện để tiếp tục quá trình tối ưu hệ thống tham số.

4.6.2 Huấn luyện mô hình với môi trường có hệ thống cảm biến dày đặc

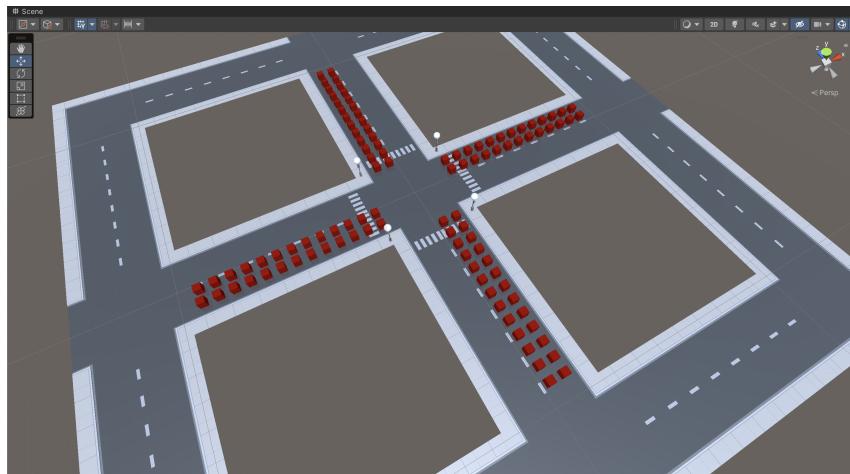
Để khởi đầu quá trình phát triển, trước hết em sử dụng một hệ thống môi trường có hệ thống cảm biến rất dày, được đặt ở hầu hết các vị trí trong môi trường. Trong đó, mỗi hướng đường bao gồm 63 cảm biến trong đoạn đường dài 100m và rộng 15m được thể hiện trong hình 4.2. Điều này đảm bảo agent có đủ thông tin nhận

**Hình 4.2:** Hệ thống 252 cảm biến**Hình 4.3:** Kết quả với hệ thống môi trường nhiều cảm biến

thức về môi trường do hệ thống cảm biến rất dày và không bị mất mát trong các quãng giãn cách.

Với cấu hình môi trường này, em sử dụng một kiến trúc mạng neuron đơn giản với 5 lớp ẩn, trong đó mỗi lớp có 128 đơn vị ẩn và với các giá trị siêu tham số cơ bản đã được chọn, quá trình huấn luyện diễn ra khá ổn định và hiệu quả, được thể hiện qua hình 4.3. Trong đó, ta có thể thấy mô hình học một cách tương đối ổn định và hội tụ sau khoảng 25000 bước thời gian.

Với kết quả khá tốt, đạt giá trị điểm thưởng trung bình -2162. Giảm 57% so với hiệu năng cơ sở. Bên cạnh đó, sau khi hội tụ, lượng điểm thưởng nhận được của mô hình này cũng rất ổn định, thay đổi trong khoảng 216. Sự vượt trội này được thể hiện rõ trong hình 4.3 so sánh kết quả của hai hệ thống. Những kết quả này thể



Hình 4.4: Hệ thống 104 cảm biến

hiện hiệu năng áp đảo của hệ thống thông minh so với hệ thống truyền thống về cả số điểm thường trung bình lẫn sự ổn định. Với hệ thống này, ta có thể đảm bảo được việc tối ưu các quyết định với mục đích hướng tới là sự thoái mái của người tham gia giao thông.

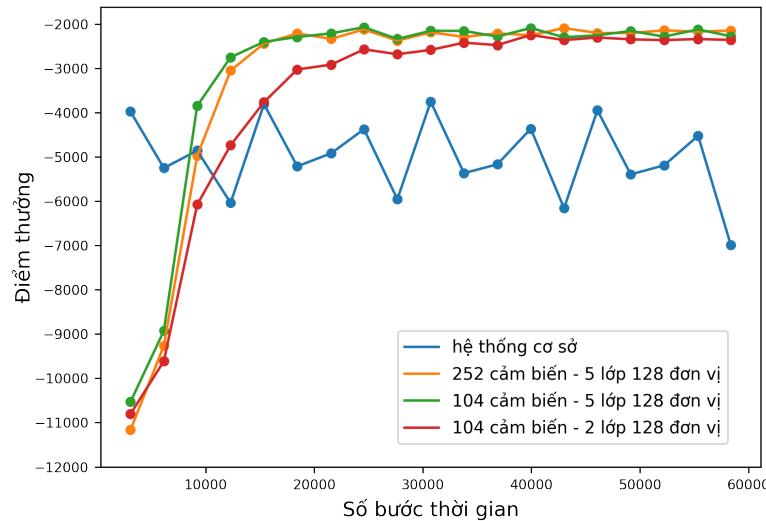
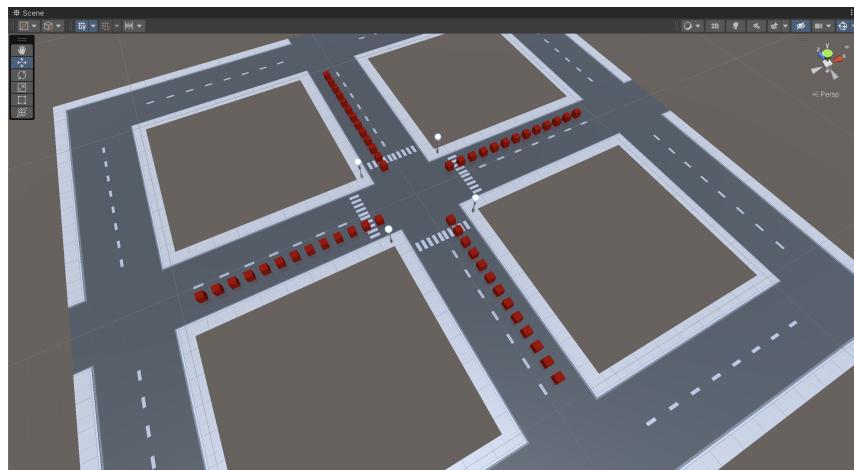
Với nhận định rằng các thông tin quan sát của agent về môi trường là tương đối đầy đủ do có số lượng cảm biến khá dày. Ta thấy hệ thống có thể được cải tiến về mặt mô hình và thử nghiệm giảm số lượng cảm biến để tối ưu hoá hệ thống, giảm thiểu chi phí.

4.6.3 Cắt giảm số lượng cảm biến

Để giảm thiểu một lượng lớn chi phí cho việc mua và lắp đặt các cảm biến, em thử nghiệm giảm lượng lớn các cảm biến trong môi trường. Cụ thể, thay lắp đặt 3 dải cảm biến song song như trong hệ thống ban đầu, ta có thể sử dụng hai dải cảm biến do trên môi trường hiện tại chỉ có 2 làn đường xe chạy. Đồng thời, ta cũng giảm bớt số lượng cảm biến thông qua việc đặt chúng cách xa nhau hơn, làm tăng độ thưa của chúng. Hệ thống cảm biến mới được thể hiện ở hình 4.4, số lượng cảm biến được giảm từ 252 xuống còn 104.

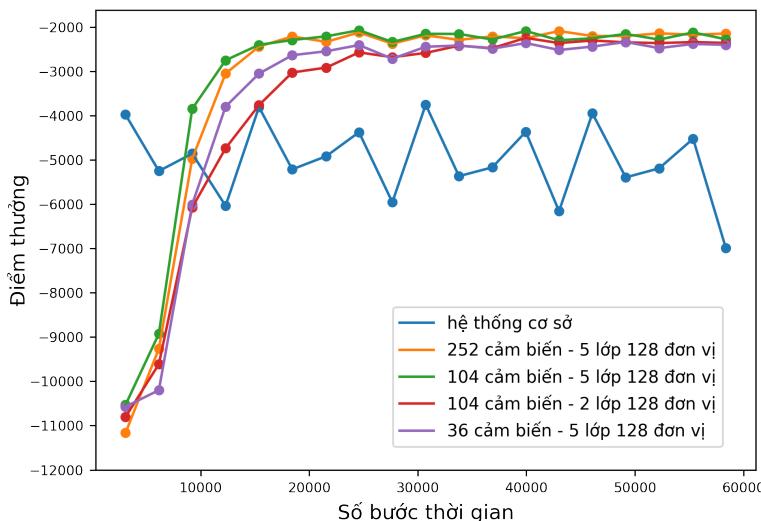
Với số lượng cảm biến giảm đáng kể, chi phí cho việc mua cũng như lắp đặt hệ thống giảm đáng kể. Trong khi đó, theo kết quả huấn luyện mô hình được thể hiện trong hình, ta vẫn đạt được mức điểm thưởng tương đương như trong hệ thống cảm biến dày đặc, được thể hiện trong hình 4.5 Chỉ với lượng cảm biến tương đối thưa về môi trường, agent vẫn có thể đưa ra được các quyết định hợp lý và đạt kết quả ổn định.

Khi giảm kích thước mô hình một cách tương đối nhiều với số lượng lớp ẩn giảm từ 5 xuống 2. Hiệu năng trên đã có giảm nhẹ và có sự thay đổi về quá trình hội tụ

**Hình 4.5:** Kết quả với hệ thống 104 cảm biến**Hình 4.6:** Hệ thống 36 cảm biến

của mô hình. Hình 4.5 thể hiện rõ sự so sánh giữa 3 hệ thống thông minh với các cấu hình khác nhau và với kết quả của hệ thống cơ sở. Khi giảm kích thước mô hình, hiệu năng của hệ thống khi hội tụ có sự giảm nhẹ so với hai hệ thống trước. Điều này là do kích thước mô hình quá nhỏ dẫn tới không thể học được đủ các đặc tính của môi trường. Từ đó, trong một số trường hợp không hiểu rõ, hệ thống có thể đưa ra một số hành động không tối ưu nhất.

Tuy nhiên, với những kết quả trên, ta vẫn có thể đánh giá rằng 104 cảm biến là một số lượng đủ để agent có thể nắm bắt được các thông tin cần thiết về lưu lượng giao thông trong thời gian thực trong môi trường. Do đó, ta có thể thử nghiệm tiếp tục giảm số lượng cảm biến.



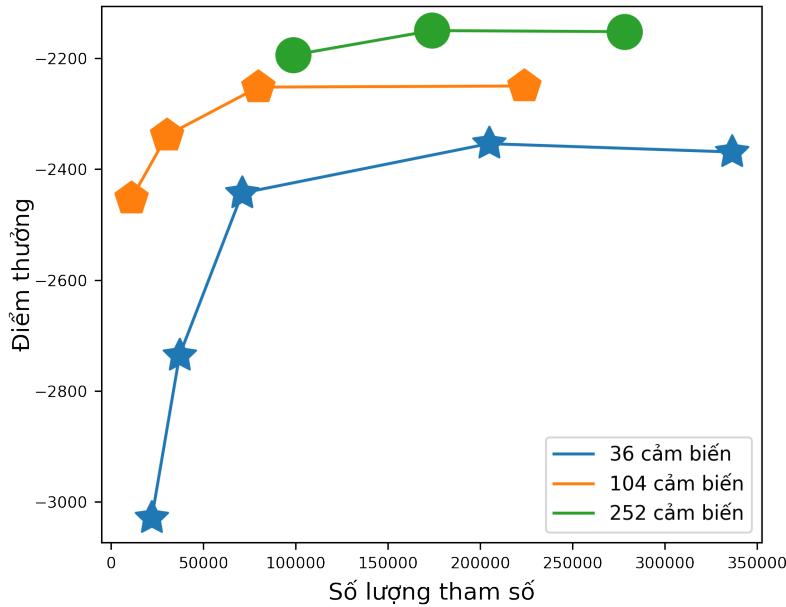
Hình 4.7: Kết quả của hệ thống cảm biến tối giản

4.6.4 Hệ thống cảm biến tối giản

Để cắt giảm số lượng cảm biến trong môi trường. Ta tiến hành cắt giảm 2 dải cảm biến xuống một, từ đó ta chọn điểm trung tâm 2 làn đường để đặt cảm biến, từ đó có thể nhận được thông tin về cả 2 làn đường. Hệ thống được thể hiện trong hình 4.6.

Với số lượng cảm biến giảm đáng kể, kết quả ta đạt được thể hiện sự sụt giảm tương đối so với hệ thống 104 cảm biến. Điểm thường trung bình đạt mức -2431 với độ chênh lệch giữa các tập môi trường khá thấp. Nhìn vào hình 4.7, ta có thể thấy được một sự so sánh về kết quả của các hệ thống với cấu hình khác nhau. Ở đây, hệ thống với lượng cảm biến tối giản vẫn đem lại một kết quả khả quan khi chỉ thấp hơn hai hệ thống tốn kém nhất và ngang bằng với hiệu năng của hệ thống 104 cảm biến với mô hình nhỏ.

Bên cạnh đó, sự sụt giảm hiệu năng này có thể hiểu được do, do ta chỉ sử dụng 9 cảm biến ở mỗi hướng đường, trong đó cảm biến được đặt ở giữa 2 làn đường và không có sự phân biệt giữa việc có 1 xe hay 2 xe đang trong vùng cảm biến. Khi này, những quan sát về môi trường sẽ quá thưa và có một chút sai lệch so với thực tế. Tuy nhiên, ta có thể thấy kết quả của mô hình vẫn khá ổn định qua các tập môi trường với các tình trạng môi trường khác nhau, thể hiện được sự thích nghi của mô hình với mọi tính huống trong môi trường. Những kết quả đạt được vẫn là tương đối tốt so với hệ thống tốt nhất, giảm 12%. Và so với mức hiệu năng cơ sở thì ta vẫn đạt mức rất tốt, tăng 51%.



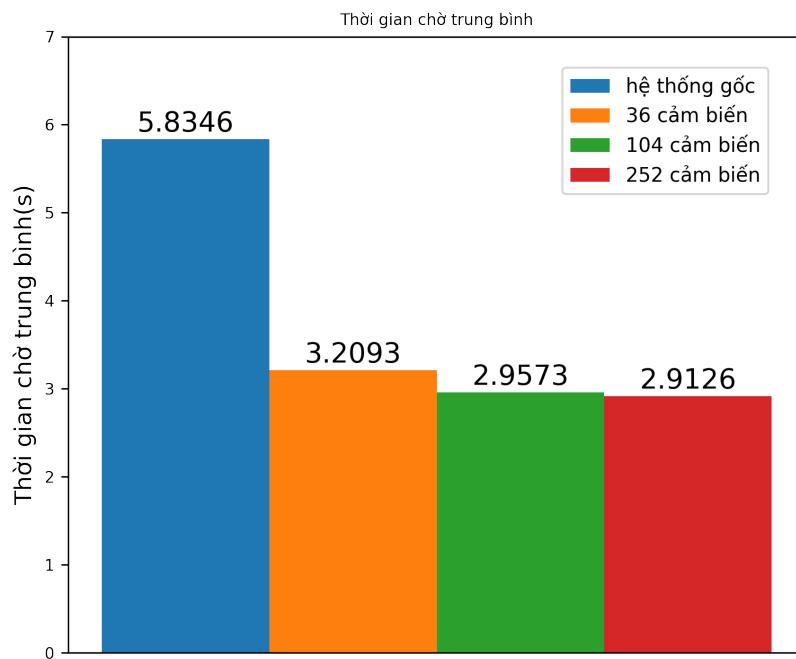
Hình 4.8: Giá trị điểm thưởng trung bình cho các cấu hình khác nhau của hệ thống

4.6.5 Đánh giá kết quả các cấu hình hệ thống

Sau quá trình huấn luyện mô hình, em tiến hành đánh giá kết quả của các cấu hình hệ thống khác nhau trên môi trường kiểm tra. Hình 4.8 thể hiện cái nhìn tổng quan về kết quả của các cấu hình khác nhau của hệ thống cảm biến cũng như kích thước mô hình sử dụng. Ta có thể thấy những kết quả của hệ thống có sự khác biệt giữa các cấu hình.

Hệ thống 252 cảm biến cung cấp mức điểm thưởng trung bình cao nhất trong 3 cấu hình cảm biến. Trong đó, mức tối đa đạt được tại mô hình với 152283 tham số. Thực nghiệm cho thấy không có thay đổi đáng kể khi tăng kích thước mô hình. Tuy nhiên, hệ thống này có những điểm hạn chế như yêu cầu quá nhiều cảm biến. Đồng thời, do đầu vào của hệ thống có nhiều cảm biến hơn nên số tham số của mô hình cũng tăng đáng kể.

Hệ thống 104 cảm biến có sự giảm về điểm thưởng so với hệ thống cảm biến dày đặc trên. Điều này là do sự giảm thiểu lớn về độ dày đặc của những thông tin môi trường nhận được từ các cảm biến cùng với đó là sự giảm thiểu kích thước mô hình. Tuy nhiên, kết quả nhận được vẫn rất khả quan và cách biệt không quá nhiều so với mức hiệu năng tối đa của hệ thống cảm biến dày đặc. Bên cạnh đó, do số lượng cảm biến giảm đáng kể, với cùng một số lượng đơn vị ẩn trong mỗi lớp và số lớp ẩn trong mô hình, lượng tham số của mô hình cũng giảm đi đáng kể. Ở một mức lượng tham số khá tương đồng, hệ thống này đạt hiệu năng tương đương với

**Hình 4.9:** Thời gian chờ trung bình

hệ thống cảm biến dày đặc với cùng kích thước mô hình.

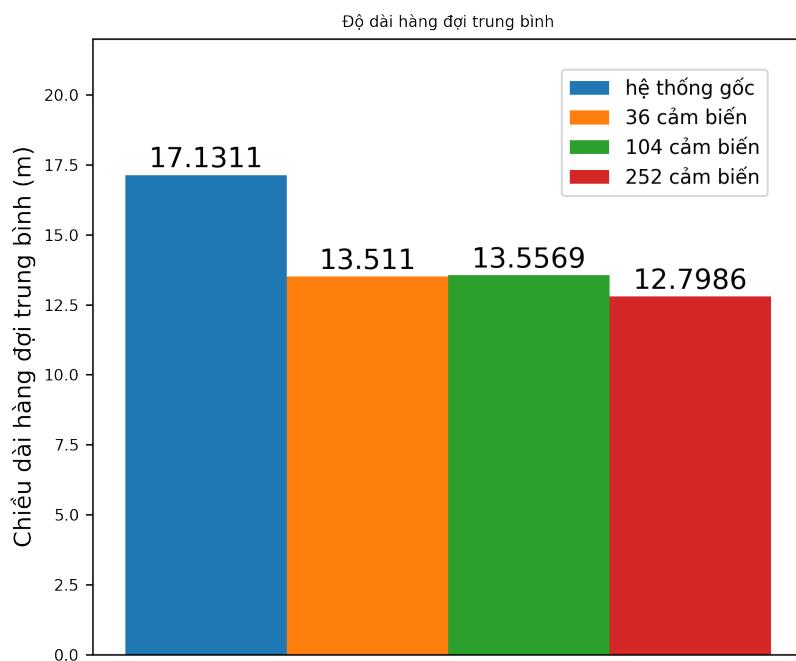
Hệ thống 36 cảm biến có lượng điểm thưởng thấp hơn so với hai hệ thống kể trên. Mặc dù có số lượng cảm biến khá ít và thông tin đầu vào từ môi trường rất thưa, bằng việc tăng kích thước mô hình, ta vẫn có thể xây dựng được hệ thống này với kết quả tương đối tốt. Nếu so sánh với hệ thống cơ sở thì ta vẫn có một sự khác biệt lớn. Hơn nữa, việc giảm lượng cảm biến từ 252 xuống 36 là sự giảm thiểu rất lớn về chi phí cần thiết cho giải pháp, từ đó giúp cho tính ứng dụng của giải pháp được tăng cường, giúp giải quyết được bài toán thực tế.

Qua những đánh giá này, ta có thể thấy được một cái nhìn tổng quan về quá trình huấn luyện mô hình và những kết quả đạt được với những cấu hình khác nhau của hệ thống. Từ đó, tuỳ thuộc vào nhu cầu thực tế mà ta có thể sử dụng các cấu hình khác nhau cho các bài toán cụ thể.

4.7 Đánh giá các hệ thống qua các độ đo đánh giá

4.7.1 Thời gian chờ trung bình

Thời gian chờ trung bình là trọng tâm tối ưu hoá trong bài toán đặt ra. Đánh giá này thể hiện giá trị trung bình mà mỗi xe xuất hiện trong môi trường phải chờ. Mục tiêu của bài toán là giảm giá trị này tới mức tối thiểu, đồng nghĩa với việc người tham gia giao thông có thể lưu thông một cách nhanh và thoải mái hơn, đồng thời tiết kiệm được lượng thời gian tham gia giao thông hàng ngày. Sau quá trình đánh giá hệ số này trên nhiều cấu hình hệ thống và kích thước của mô hình, một số kết

**Hình 4.10:** Chiều dài hàng chờ trung bình

quả tiêu biểu được thể hiện trong hình 4.9.

Ta có thể thấy, hàm mục tiêu của mô hình ta đặt ra có phần khác so với hệ số đánh giá này, trong đó hàm mục tiêu của ta có đưa thêm một trọng số liên quan đến thời gian chờ của xe (xe chờ càng lâu thì trọng số điểm thưởng âm càng lớn). Mặc dù vậy, hệ thống được xây dựng từ mô hình trên vẫn cho ta một kết quả vượt trội so với hệ thống gốc. Hệ thống tối giản với 36 cảm biến đạt mức thời gian chờ trung bình giảm 45% so với hệ thống gốc. Và ở các hệ thống với nhiều cảm biến hơn con số này còn lớn hơn nữa.

4.7.2 Chiều dài hàng chờ trung bình

Kết quả đánh giá các hệ thống với độ đo đánh giá chiều dài hàng chờ trung bình được thể hiện trong hình 4.10. Ở đây ta cũng thấy được sự vượt trội của hệ thống tối ưu so với hệ thống gốc. Các cấu hình hệ thống tối ưu khác nhau không có quá nhiều sự chênh lệch về hiệu năng, đều ở mức giảm 21% chiều dài hàng chờ trung bình so với hệ thống gốc.

4.8 Đánh giá các mục tiêu

- Với các cấu hình khác nhau của môi trường, em đã thực hiện đánh giá và rút ra kết quả các mức hiệu năng khác nhau với từng cấu hình. Đồng thời, mỗi cấu hình cũng sẽ phù hợp với từng điều kiện thực tế của bài toán như mức hiệu năng yêu cầu, mức chi phí.
- Với các độ lớn khác nhau của mô hình, kết quả thử nghiệm cho thấy hiệu năng

tăng lên khi ta tăng kích thước của mô hình. Trong đó, mức kích thước với 5 lớp ẩn gồm 128 đơn vị ẩn là một mốc giới hạn mà từ đó khi tăng kích thước mô hình hiệu năng không có nhiều thay đổi.

- Trong việc giải bài toán đề ra, các phương pháp học tăng cường giúp tối ưu hoá các quyết định tín hiệu đèn giao thông một cách hiệu quả dựa trên chuỗi các kinh nghiệm tích luỹ trong quá khứ. Do đó, phương pháp này giúp cho các quyết định này không bị phụ thuộc vào các thông tin nhãn từ con người - điều khó thực hiện trong bài toán mà ta đặt ra. Chính vì vậy, giải pháp giúp ta vượt qua được vật cản về việc tạo thông tin nhãn cho hành động, và đồng thời cung cấp một mô hình tối ưu dựa trên điều kiện thực tế, giúp ta nâng cao được giá trị ứng dụng của nó.
- Các kết quả đạt được cho thấy, với các mức chi phí yêu cầu khác nhau, kết quả đạt được có sự thay đổi nhưng đều vượt trội so với hệ thống không tối ưu hiện tại. Do đó, với lượng dữ liệu thật ngoài thực tế, sau quá trình huấn luyện và thử nghiệm kết quả, hệ thống có thể được đem vào thí điểm tại một số cụm đèn giao thông, và từ đó đem vào sử dụng thực tế nếu đạt kết quả tốt. Từ việc có thể tiếp tục nhận các thông tin và đưa ra các hành động khi được đem vào thực tế sử dụng, hệ thống có thể liên tục học và cập nhật policy của mình. Kết hợp với một phương pháp triển khai hệ thống một cách an toàn, việc sử dụng và liên tục tối ưu hoá hệ thống là khả quan.

4.9 Kết chương

Như vậy, trong suốt quá trình thực hiện đồ án, em đã xây dựng được một môi trường học tăng cường với bài toán tối ưu hoá các tín hiệu đèn giao thông tại một ngã tư đơn giản, xây dựng các hệ thống cảm biến khác nhau và thử nghiệm phương pháp học tăng cường vào tối ưu hoá hệ thống đèn giao thông. Em đã đưa ra kết quả khả thi và đạt kỳ vọng đặt ra khi hệ thống tối ưu có hiệu năng vượt trội so với hệ thống gốc. Kết quả này khẳng định tính khả thi của giải pháp và khả năng phát triển và ứng dụng vào thực tiễn trong tương lai.

CHƯƠNG 5. KẾT LUẬN

5.1 Kết luận

5.1.1 Tổng kết kết quả

Trong nội dung đồ án, em đã đạt được một số kết quả chính như sau:

- Xây dựng môi trường hệ thống giao thông ở một ngã tư đơn giản, hoạt động một cách hiệu quả và có khả năng tùy biến.
- Thủ nghiệm phương pháp học tăng cường với các cấu hình khác nhau của môi trường và mô hình. Đạt kết quả tốt với mức thời gian chờ trung bình của các phương tiện giảm 45% và độ dài hàng chờ giảm 21%.

5.1.2 Điểm mạnh

Giải pháp đưa ra có hiệu năng tăng nhiều so với hiệu năng cơ sở, cùng với đó là khả năng ứng dụng cao trong thực tế. Do đó, hệ thống có thể được phát triển để sử dụng ngoài đời thực. Bên cạnh đó, môi trường được xây dựng trong đồ án có thể được phát triển để tăng độ phức tạp của bài toán. Mô hình bài toán cũng là một điểm mạnh khi có thể áp dụng vào một số bài toán khác.

5.1.3 Điểm hạn chế

Điểm hạn chế của đồ án là môi trường đang xét tới còn tương đối đơn giản, do đó việc học của agent chưa thực sự quá khó khăn để cần tới các bước tinh chỉnh phức tạp. Các đánh giá còn tương đối chủ quan, cần có sự tham gia của các chuyên gia trong lĩnh vực giao thông để có thể tạo nên một hệ thống tham số đánh giá hợp lý và có tính chất ứng dụng cao hơn nữa.

5.2 Hướng phát triển

- Xây dựng môi trường phức tạp với quy mô lớn hơn, ví dụ như quy mô hệ thống trong một thành phố, hay một quận, huyện. Với môi trường này, ta có thể tận dụng được tính tối ưu của phương pháp học tăng cường. Khi đó, nhờ khả năng tối ưu hoá cho hàm mục tiêu chung trên toàn bộ môi trường, agent có thể học được những tính chất liên quan đến phân luông giao thông, đưa ra các quyết định tín hiệu đèn để tối ưu trên cả môi trường thay vì một cụm đèn duy nhất.
- Thay đổi mô-đun cảm biến của môi trường bằng hình ảnh. Theo đó, ta có thể để agent học từ đầu vào là ảnh, và từ đó ta có thể huấn luyện một mô hình đưa ra các quyết định đèn tín hiệu, hoặc thêm một mô-đun đánh giá lưu lượng xe ở các hướng dựa vào ảnh. Từ đây, ta giảm được chi phí cho việc phải sử dụng một số lượng cảm biến khá lớn trên đường.

5.3 Kết luận

Như vậy, trong nội dung báo cáo đồ án tốt nghiệp, em đã trình bày về các phương pháp học tăng cường, quá trình phát triển môi trường cho bài toán hệ thống giao thông ở một ngã tư, và từ đó ứng dụng các phương pháp trên vào tìm giải pháp tối ưu cho bài toán. Những kết quả đạt được từ đồ án khá tốt và sẽ là tiền đề để em có thể tiếp tục phát triển bài toán và tiến đến ứng dụng vào giải quyết vấn đề cho xã hội. Do thời gian thực hiện đồ án có hạn nên kết quả của đồ án còn có những hạn chế. Em mong có thể có được sự góp ý và hỗ trợ từ thầy cô, bạn bè để giúp em phát triển đồ án thành một dự án lớn hơn, mang lại những giá trị cho xã hội.

TÀI LIỆU THAM KHẢO

- [1] S. Dick, “Artificial intelligence,” 2019.
- [2] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Design of reinforcement learning parameters for seamless application of adaptive traffic signal control,” *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 227–245, 2014.
- [3] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.
- [4] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, “Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network,” *arXiv preprint arXiv:1705.02755*, 2017.
- [5] C.-H. Wan and M.-C. Hwang, “Value-based deep reinforcement learning for adaptive isolated intersection signal control,” *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1005–1010, 2018.
- [6] X. Liang, X. Du, G. Wang, and Z. Han, “Deep reinforcement learning for traffic light control in vehicular networks,” *arXiv preprint arXiv:1803.11115*, 2018.
- [7] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.
- [8] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.
- [9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [10] T. M. Mitchell, *Machine learning*, 9. McGraw-hill New York, 1997, vol. 1.
- [11] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [12] C. Berner, G. Brockman, B. Chan, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.
- [13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [14] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [15] N. Metropolis and S. Ulam, “The monte carlo method,” *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

- [16] G. Tesauro *et al.*, “Temporal difference learning and td-gammon,” *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.
- [17] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.
- [18] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [19] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in neural information processing systems*, vol. 12, 1999.
- [20] S. T. Tokdar and R. E. Kass, “Importance sampling: A review,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 54–60, 2010.
- [21] J. K. Haas, “A history of the unity game engine,” 2014.
- [22] X. Lin, “Multi-behaviors finite state machine,” in *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, 2009, pp. 201–203. DOI: 10.1109/YCICT.2009.5382390.
- [23] A. Juliani, V.-P. Berges, E. Teng, *et al.*, “Unity: A general platform for intelligent agents,” *arXiv preprint arXiv:1809.02627*, 2018.

PHỤ LỤC