|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



PHẠM THANH VINH

thử nghiệm thuật toán Reinforce cho mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2023

|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI

PHẠM THANH VINH

thử nghiệm thuật toán Reinforce cho mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc

Ngành: Công nghệ thông tin

Mã số: 7480201

NGƯỜI HƯỚNG DẪN: TS. Lê Nguyễn Tuấn Thành

HÀ NỘI, NĂM 2023

**GÁY BÌA ĐỒ ÁN TỐT NGHIỆP, KHÓA LUẬN TỐT NGHIỆP**

**Đ**

**Ồ ÁN TỐT NGHIỆP**

**HÀ NỘI, NĂM 2023**

**PHẠM THANH VINH**

|  |  |
| --- | --- |
|  | CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  **Độc lập - Tự do - Hạnh phúc**  --------------------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

Họ tên sinh viên: Phạm Thanh Vinh Hệ đào tạo: Đại học chính quy

Lớp: 60TH2 Ngành: Công nghệ thông tin

Khoa: Công nghệ thông tin

1. **TÊN ĐỀ TÀI:**

THỬ NGHIỆM THUẬT TOÁN REINFORCE CHO MÔ HÌNH BÀI TOÁN GIẢM ÙN TẮC GIAO THÔNG TRÊN ĐƯỜNG CAO TỐC

1. **CÁC TÀI LIỆU CƠ BẢN**
2. <https://en.wikipedia.org/wiki/Reinforcement_learning>
3. Head, B., 2017. Demos for [Python Extension of NetLogo.](https://github.com/NetLogo/Python-Extension/blob/main/demos/Traffic%20Basic%20-%20Reinforcement.nlogo)
4. Lilian , W., 2018. [A (Long) Peek into Reinforcement Learning](https://lilianweng.github.io/posts/2018-02-19-rl-overview/).
5. Richard S. Sutton and Andrew G.Barto.[Reinforcement Learning Book (2020)](http://incompleteideas.net/book/RLbook2020.pdf)
6. **CÁC TÀI LIỆU THAM KHẢO**
7. [Reinforcement Learning](https://en.wikipedia.org/wiki/Reinforcement_learning)
8. [Supervised vs Unsupervised vs Reinforcement](https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/)
9. Jordi Torres. [Policy Gradient Method](https://towardsdatascience.com/policy-gradient-methods-104c783251e0)
10. Lilian Weng (2018). [Policy Gradient Algorithms](https://lilianweng.github.io/posts/2018-04-08-policy-gradient/)
11. <https://github.com/abhisheksuran/Reinforcement_Learning>
12. <https://github.com/kvsnoufal/reinforce>
13. <https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/>
14. **NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:**

|  |  |
| --- | --- |
| **Nội dung thuyết minh và tính toán** | **Tỷ lệ %** |
| Chương 1: Mở đầu | 5% |
| Chương 2: Tổng quan về học tăng cường | 20% |
| Chương 3: Hệ đa tác tử | 20% |
| Chương 4: Thuật toán Reinforce | 25% |
| Chương 5: Thử nghiệm thuật toán Reinforce cho mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc | 30% |
| Chương 6: Kết luận và hướng phát triển | 5% |

1. **GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN**

Giáo viên hướng dẫn toàn bộ quá trình thực hiện đồ án tốt nghiệp: **TS. Lê Nguyễn Tuấn Thành**

1. **NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Ngày ............ tháng ......... năm 2023

|  |  |
| --- | --- |
| **Trưởng Bộ môn** | **Giáo viên hướng dẫn chính** |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày. . . . .tháng. . . . .năm 2023

|  |  |
| --- | --- |
|  | **Chủ tịch Hội đồng** |

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi

Ngày … tháng … năm 2023

|  |  |
| --- | --- |
|  | **Sinh viên làm Đồ án tốt nghiệp** |

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  **BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP** |

**TÊN ĐỀ TÀI:**

**THỬ NGHIỆM THUẬT TOÁN REINFORCE CHO MÔ HÌNH BÀI TOÁN GIẢM ÙN TẮC GIAO THÔNG TRÊN ĐƯỜNG CAO TỐC**

***Sinh viên thực hiện:* Phạm Thanh Vinh**

***Lớp:* 60TH2**

***Giáo viên hướng dẫn:* TS. Lê Nguyễn Tuấn Thành**

**TÓM TẮT ĐỀ TÀI**

Reinforcement Learning (RL) hay còn được gọi là học tăng cường – là phương pháp phổ biến được áp dụng trong nhiều lĩnh vực. Có thể kể đến một vài thành công của học tăng cường như AlphaGo, robot học tăng cường nhằm tương tác với môi trường bên ngoài hay trong việc xử lý ngôn ngữ tự nhiên để tạo ra các chương trình dịch. Nổi bật trong RL phải kể đến Reinforce Algorithm – một trong những thuật toán policy gradient phổ biến và thành công nhất. Thuật toán cập nhật các thông số của chính sách theo hướng gradient của expected reward so với các thông số của chính sách. Phương pháp này đã được áp dụng thành công trong nhiều lĩnh vực, từ game đến robot xử lý ngôn ngữ tự nhiên.

Cốt lõi của học tăng cường (RL) là học tăng cường dựa trên tương tác giữa một hệ thống (Agent) với môi trường (Environment) bằng cách thu thập dữ liệu từ các trạng thái (states) và hành động (actions) của Agent. Trong những bài toán thực tế, để giải quyết bài toán nhiều tác tử tương tác với nhau thì ta sẽ ứng dụng hệ đa tác tử (MAS –Multi-agent system) ví dụ như bài toán ngập lụt, bài toán lây lan bệnh dịch...

Đề tài hướng tới việc ứng dụng phương pháp học tăng cường đa tác tử với thuật toán Reinforce cho bài toán giảm tắc nghẽn giao thông trên đường cao tốc.

**CÁC MỤC TIÊU CHÍNH**

* Kiến thức:
* Tìm hiểu tổng quan về học tăng cường.
* Tìm hiểu lý thuyết của Reinforce Algorithm.
* Tìm hiểu về hệ đa tác tử, các nền tảng mô hình hóa, mô phỏng đa tác tử.
* Tìm hiểu nền tảng mô hình hóa và mô phỏng đa tác tử NetLogo.
* Kỹ năng :

Thử nghiệm phương pháp học tăng cường đa tác tử với thuật toán Reinforce Algorithm cho bài toán giảm tắc nghẽn giao thông trên đường cao tốc áp dụng trên nền tảng netLogo.

**KẾT QUẢ DỰ KIẾN**

* Kiến thức:
* Nắm được kiến thức về học tăng cường, học tăng cường đa tác tử.
* Nắm được lý thuyết về Reinforce Algorithm.
* Nắm được kiến thức về hệ đa tác tử, nền tảng netLogo.
* Đánh giá, so sánh kết quả nghiên cứu.
* Kỹ năng:

Cài đặt được thuật toán Reinforce Algorithm trên nền tảng NetLogo cho bài toán giảm tắc nghẽn giao thông trên đường cao tốc.

**TIẾN ĐỘ THỰC HIỆN**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Thời gian** | **Nội dung công việc** | **Kết quả dự kiến đạt được** |
| 1 | 20/3/2023-10/4/2023 | Đọc tài liệu, nghiên cứu lý thuyết về học tăng cường, hệ đa tác tử | Hoàn thành bản tóm tắt về học tăng cường, hệ đa tác tử |
| 2 | 10/4/2023-24/4/2023 | Tìm hiểu lý thuyết về thuật toán Reinforce | Nắm bắt cơ bản kiến thức về thuật toán Reinforce |
| 2 | 24/4/2023-  8/5/2023 | Đọc tài liệu về nền tảng mô hình hóa, mô phỏng hóa đa tác tử và nền tảng NetLogo | Sử dụng thành công công cụ NetLogo |
| 3 | 8/5/2023-  25/5/2023 | Thử nghiệm cài đặt thuật toán Reinforce vào mô hình | Cài đặt thành công thuật toán cho bài toán giảm tắc nghẽn giao thông |
| 4 | 25/5/2023-  25/6/2023 | Chỉnh sửa và hoàn thành báo cáo đồ án | Báo cáo đồ án hoàn chỉnh |

**TÀI LIỆU THAM KHẢO**

1. <https://www.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html>
2. <https://codelearn.io/sharing/reinforcement-learning-la-gi>
3. <https://en.wikipedia.org/wiki/Reinforcement_learning>
4. <https://www.analyticsvidhya.com/blog/2020/11/reinforce-algorithm-taking-baby-steps-in-reinforcement-learning/>
5. <https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/>

LỜI CAM ĐOAN

Tác giả xin cam đoan đây là Đồ án tốt nghiệp/ Khóa luận tốt nghiệp của bản thân tác giả. Các kết quả trong Đồ án tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

Tác giả ĐATN

Phạm Thanh Vinh

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án tốt nghiệp, em đã nhận được nhiều sự giúp đỡ cũng như động viên, đóng góp ý kiến từ thầy cô, sự cảm thông, sẻ chia từ gia đình và bạn bè đã giúp em tập trung và hoàn thiện đồ án tốt nghiệp sau thời gian hơn 4 năm học tập dưới mái trường Đại học Thủy lợi.

Em xin cảm ơn thầy –TS. Lê Nguyễn Tuấn Thành giảng viên bộ môn Công nghệ phần mềm đã chấp nhận hướng dẫn em tham gia đề tài nghiên cứu cũng như tận tình chỉ bảo trong suốt quá trình em thực hiện đồ án.

Bên cạnh đó, em cũng xin gửi lời cảm ơn đến các thầy cô trường Đại học Thủy lợi nói chung và khoa Công nghệ thông tin nói riêng. Các kiến thức em học được nhờ sự dạy dỗ của thầy cô giúp em có cơ sở vững chắc và là bước đệm quan trọng giúp em hoàn thành đồ án tốt nghiệp này. Cuối cùng, em xin gửi lời cảm ơn chân thành nhất tới bạn bè, người thân đã luôn bên cạnh, động viên, giúp đỡ em ở những thời điểm khó khăn và trong quá trình học tập.

Với vốn kiến thức còn nhiều hạn hẹp, đồ án của em không tránh khỏi những thiếu sót. Em rất mong nhận được những lời góp ý từ thầy cô và hội đồng bảo vệ đồ án tốt nghiệp để em có thêm những kinh nghiệm quý báu, nâng cao kiến thức của bản thân.

Em xin chân thành cảm ơn!

MỤC LỤC

[CHƯƠNG 1 MỞ ĐẦU 1](#_Toc138578031)

[1.1. Tổng quan tình hình nghiên cứu 1](#_Toc138578032)

[1.2. Lí do chọn đề tài 1](#_Toc138578033)

[1.3. Mục tiêu của đề tài 2](#_Toc138578034)

[1.4. Phương pháp nghiên cứu 2](#_Toc138578035)

[1.5. Đối tượng và phạm vi nghiên cứu 2](#_Toc138578036)

[CHƯƠNG 2 TỔNG QUAN VỀ HỌC TĂNG CƯỜNG 3](#_Toc138578037)

[2.1. Tổng quan về học tăng cường 3](#_Toc138578038)

[2.2. Các cột mốc phát triển của học tăng cường 4](#_Toc138578039)

[2.3. Phân biệt học tăng cường và các phương pháp học máy khác 5](#_Toc138578040)

[2.3.1 Học tăng cường và Học có giám sát 5](#_Toc138578041)

[2.3.2 Học tăng cường và Học không có giám sát 7](#_Toc138578042)

[2.4. Các khái niệm cơ bản về học tăng cường 8](#_Toc138578043)

[2.4.1 Tác tử (agent) 8](#_Toc138578044)

[2.4.2 Môi trường (Environment) 9](#_Toc138578045)

[2.4.3 Mối quan hệ giữa trạng thái (State), Hành động (Action) và phần thưởng (Reward) 9](#_Toc138578046)

[2.4.4 Quá trình quyết định Markov 10](#_Toc138578047)

[2.4.5 Mô hình và chính sách 10](#_Toc138578048)

[2.4.6 Các hàm giá trị (Value function) 11](#_Toc138578049)

[2.4.6.1 Hàm giá trị trạng thái (state-value function) 11](#_Toc138578050)

[2.4.6.2 Hàm giá trị hành động (action-value function) 12](#_Toc138578051)

[2.4.7 Các phương trình Bellman 12](#_Toc138578052)

[2.5. Một số thuật toán học tăng cường 13](#_Toc138578053)

[2.5.1 Q – Learning 13](#_Toc138578054)

[2.5.2 SARSA 14](#_Toc138578055)

[2.5.3 Deep Q – Network (DQN) 14](#_Toc138578056)

[2.6. Ưu điểm và nhược điểm của học tăng cường và một số ứng dụng của học tăng cường 15](#_Toc138578057)

[CHƯƠNG 3 HỆ ĐA TÁC TỬ 17](#_Toc138578058)

[3.1. Khái quát về học tăng cường đa tác tử 17](#_Toc138578059)

[3.2. Hệ đa tác tử là gì? 17](#_Toc138578060)

[3.3. Phân loại kiến trúc các hệ đa tác tử 19](#_Toc138578061)

[3.3.1 Kiến trúc tổ chức phân cấp (Hierarchical Organization) 19](#_Toc138578062)

[3.3.2 Kiến trúc tổ chức Holonic (Holonic Agent Organization – HAO) 19](#_Toc138578063)

[3.3.3 Kiến trúc liên minh (Coalitions) 20](#_Toc138578064)

[3.3.4 Kiến trúc nhóm (Teams) 21](#_Toc138578065)

[3.4. Tổng quan về NetLogo 21](#_Toc138578066)

[3.4.1 Đặc điểm của NetLogo 22](#_Toc138578067)

[3.4.2 Một số hình mẫu NetLogo 23](#_Toc138578068)

[3.4.3 Giao diện NetLogo 24](#_Toc138578069)

[3.4.4 Lập trình trong NetLogo 26](#_Toc138578070)

[3.4.4.1 Kiểu biến 26](#_Toc138578071)

[3.4.4.2 Thủ tục 26](#_Toc138578072)

[3.4.4.3 Bộ đếm thời gian 27](#_Toc138578073)

[3.4.4.4 Màu sắc 27](#_Toc138578074)

[3.4.4.5 Lệnh ask 28](#_Toc138578075)

[3.4.4.6 Breeds 28](#_Toc138578076)

[3.4.4.7 Buttons 28](#_Toc138578077)

[3.4.4.8 List 29](#_Toc138578078)

[CHƯƠNG 4 THUẬT TOÁN REINFORCE 30](#_Toc138578079)

[4.1. Policy Gradient Methods 30](#_Toc138578080)

[4.2. Thuật toán Reinforce 32](#_Toc138578081)

[4.2.1 Vấn đề của gradient chính sách (policy gradient) 32](#_Toc138578082)

[4.2.2 Giới thiệu về thuật toán 33](#_Toc138578083)

[4.3. Các bước thực hiện thuật toán 33](#_Toc138578084)

[4.4. Nhận xét thuật toán 36](#_Toc138578085)

[CHƯƠNG 5 THỬ NGHIỆM THUẬT TOÁN REINFORCE CHO MÔ HÌNH BÀI TOÁN GIẢM ÙN TẮC GIAO THÔNG TRÊN ĐƯỜNG CAO TỐC 39](#_Toc138578086)

[5.1. Giới thiệu bài toán giảm ùn tắc giao thông trên đường cao tốc 39](#_Toc138578087)

[5.2. Mô hình bài toán 39](#_Toc138578088)

[5.3. Các thành phần chính trong mô hình 39](#_Toc138578089)

[5.3.1 Trạng thái 40](#_Toc138578090)

[5.3.2 Hành động 40](#_Toc138578091)

[5.3.3 Phần thưởng 40](#_Toc138578092)

[5.4. Các chiến lược di chuyển 40](#_Toc138578093)

[5.4.1 Chiến lược tham lam: 40](#_Toc138578094)

[5.4.2 Chiến lược học tăng cường (Reinforce): 41](#_Toc138578095)

[5.5. Các khu vực giao diện của mô hình 41](#_Toc138578096)

[5.5.1 Khu vực căn chỉnh tham số đầu vào 41](#_Toc138578097)

[5.5.2 Khu vực hiển thị kết quả mô phỏng 42](#_Toc138578098)

[5.5.3 Khu vực quan sát quá trình mô phỏng 43](#_Toc138578099)

[5.6. Thực nghiệm và đánh giá kết quả 44](#_Toc138578100)

[5.6.1 Thiết lập môi trường 44](#_Toc138578101)

[5.6.2 Cài đặt chung cho các thuật toán 44](#_Toc138578102)

[5.6.3 Cài đặt thuật toán Reinforce 48](#_Toc138578103)

[5.6.4 So sánh giữa các chiến lược học tăng cường 49](#_Toc138578104)

[5.6.4.1 Cách sử dụng 49](#_Toc138578105)

[5.6.5 So sánh giữa các thuật toán học tăng cường 51](#_Toc138578106)

[CHƯƠNG 6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 53](#_Toc138578107)

[6.1. Kết luận 53](#_Toc138578108)

[6.2. Hạn chế của đề tài: 53](#_Toc138578109)

[6.3. Hướng phát triển 53](#_Toc138578110)

DANH MỤC HÌNH ẢNH

[Hình 2.1. Mô hình học tăng cường 3](#_Toc138551304)

[Hình 2.2. Các cột mốc phát triển của học tăng cường 4](#_Toc138551305)

[Hình 2.3. Học có giám sát 6](#_Toc138551306)

[Hình 2.4. Học không có giám sát 7](#_Toc138551307)

[Hình 2.5. Tác tử (agent) 8](#_Toc138551308)

[Hình 2.6. Mối quan hệ giữa tác tử và môi trường trong MDP 9](#_Toc138551309)

[Hình 2.7. Sự so sánh giữa Q-Learning và Deep Q-Learning 15](#_Toc138551310)

[Hình 3.1. Hệ thống đa tác tử (MAS) 18](#_Toc138551311)

[Hình 3.2. Kiến trúc tổ chức Holonic 20](#_Toc138551312)

[Hình 3.3. Kiến trúc liên minh 20](#_Toc138551313)

[Hình 3.4. Kiến trúc nhóm (Teams) 21](#_Toc138551314)

[Hình 3.5. Các mô hình mẫu 23](#_Toc138551315)

[Hình 3.6. 3D Solids 24](#_Toc138551316)

[Hình 3.7. Giao diện NetLogo 24](#_Toc138551317)

[Hình 3.8. Menu chính 24](#_Toc138551318)

[Hình 3.9. Giao diện Interface 25](#_Toc138551319)

[Hình 3.10. Giao diện code 25](#_Toc138551320)

[Hình 4.1. Thuật toán Reinforce 35](#_Toc138551321)

[Hình 4.2. Hiệu suất của thuật toán Reinforce 36](#_Toc138551322)

[Hình 5.1. Khu vực căn chỉnh tham số đầu vào 42](#_Toc138551323)

[Hình 5.2. Khu vực hiển thị kết quả mô phỏng 42](#_Toc138551324)

[Hình 5.3. Khu vực quan sát quá trình mô phỏng 43](#_Toc138551325)

[Hình 5.4. Hàm setup thuật toán 46](#_Toc138551326)

[Hình 5.5. Hàm go 46](#_Toc138551327)

[Hình 5.6. Hàm di chuyển cho chiến lược tham lam 47](#_Toc138551328)

[Hình 5.7. Hàm di chuyển cho chiến lược học tăng cường 47](#_Toc138551329)

[Hình 5.8. Hàm setup thuật toán Reinforce 48](#_Toc138551330)

[Hình 5.9. Behavior Space 50](#_Toc138551331)

[Hình 5.10. Hộp thoại Experiment 50](#_Toc138551332)

[Hình 5.11. Hộp thoại Run 51](#_Toc138551333)

DANH MỤC BẢNG BIỂU

[Bảng 2.1. Sự khác nhau giữa học tăng cường và học có giám sát 6](#_Toc138551341)

[Bảng 2.2. Sự khác nhau giữa học tăng cường và học không có giám sát 8](#_Toc138551342)

[Bảng 5.1. Các tham số môi trường 44](#_Toc138551343)

[Biểu đồ 5.1. Biểu đồ so sánh giữa các chiến lược học tăng cường 52](#_Toc138551409)

DANH MỤC CÁC TỪ VIẾT TẮT

**ABMS** - Agent-Based Modelling and Simulation

**CNTT** - Công nghệ thông tin

**DQN** - Deep Q – Network

**ĐATN** – Đồ án tốt nghiệp

**HAO** - Holonic Agent Organization

**MARL** - Multi-agent reinforcement learning)

**MAS** - Multi-agent system

**MDP** - Markov Decision Process 2.5.3

**PPO** - Tối ưu hóa chính sách gần nhất

**RGB** - đỏ/lục/lam

**RGBA** - đỏ/lục/lam/alpha

**SARSA** State-action-reward-State-action

**TRPO** - Tối ưu hóa chính sách khu vực tin cậy

# 

# MỞ ĐẦU

## Tổng quan tình hình nghiên cứu

Trong giai đoạn gần đây, trí tuệ nhân tạo đang là lĩnh vực thu hút được nhiều sự chú ý từ mọi người với các ứng dụng đóng góp vào công cuộc phát triển của toàn xã hội. Tăng cường áp dụng các công nghệ tiên tiến vào sản xuất nhằm tiết kiệm thời gian và chi phí là xu hướng phát triển cho các doanh nghiệp ngày nay. Bên cạnh những thành công từ ứng dụng trong thực tiễn của học máy (Machine Learning) như xử lý ảnh, nhận dạng giọng nói, dịch thuật tự động, phân tích dữ liệu và dự báo trong các lĩnh vực y tế, tài chính, kinh doanh, quân sự, .v.v... thì học tăng cường - hay còn gọi là Reinforcement Learning cũng đạt được nhiều thành công đáng kể. Học tăng cường là một nhánh nhỏ của lĩnh vực trí tuệ nhân tạo, tập trung vào việc phát triển các thuật toán và hệ thống để giúp các đối tượng tự động học cách tương tác với môi trường nhằm đạt được mục tiêu nhất định. Những phát triển mới nhất của học tăng cường như Deep Reinforcement Learning đã giúp các hệ thống tự động có khả năng học và cải thiện kĩ năng qua thời gian, tạo ra những thành tựu ấn tượng trong các lĩnh vực như các trò chơi điện tử, robot, tối ưu hóa hệ thống. Ví dụ như AlphaGo (2016) - chương trình có thể đánh bại người chơi cờ vây chuyên nghiệp nhất hay các cuộc đua chế tạo xe tự lái .v.v.. Nhờ đó, học tăng cường sâu đã trở thành một cách tiếp cận đầy hứa hẹn để giải quyết các bài toán trong thực tế.

Bên cạnh đó còn có một cách giải quyết các bài toán phức tạp trong thực tiễn, đặc biệt là các bài toán có sự tham gia của nhiều tác nhân như bài toán giảm ùn tắc giao thông, ứng phó thiên tai... đó là sử dụng phương pháp mô hình hóa và mô phỏng đa tác tử. Các tác tử phải cạnh tranh hoặc phối hợp với nhau để đạt được các kết quả tổng thể tốt nhất. Từ cách hành xử của từng cá thể và sự tương tác giữa các tác tử, chúng ta có thể thấy được xu hướng và cấu trúc chung của toàn bộ hệ thống.

## Lí do chọn đề tài

Các phương pháp học tăng cường hiện tại chỉ mới chú trọng vào các hành vi của tác nhân đơn lẻ. Tuy nhiên với các bài toán phức tạp, việc tập trung vào một tác nhân đơn lẻ không thể giúp giải quyết được toàn bộ bài toán. Ngoài ra, các nền tảng hỗ trợ mô hình hóa như NetLogo, GAMA còn gặp nhiều hạn chế trong việc mô hình hóa quá trình thay đổi hành vi của tác tử.

Do đó để giải quyết bài toán hiệu quả, việc kết hợp giữa hai phương pháp học tăng cường và mô hình hóa, mô phỏng hóa đa tác tử là vô cùng cần thiết. Đề tài “Thử nghiệm thuật toán Reinforce trong mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc” hướng đến việc tổng hợp, giải thích các kiến thức về học tăng cường cũng như là học tăng cường đa tác tử. Đồng thời trình bày rõ ý tưởng cũng như thuật toán sử dụng nhằm giải quyết bài toán trong thực tế.

## Mục tiêu của đề tài

Đề tài nghiên cứu thuật toán, cài đặt thuật toán và thử nghiệm thuật toán Reinforce, ứng dụng vào mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc.

## Phương pháp nghiên cứu

Phương pháp nghiên cứu của đề tài là kết hợp giữa lí thuyết và thực hành. Tìm hiểu các kiến thức về học tăng cường, thuật toán Reinforce và cài đặt thử nghiệm thuật toán trong mô hình bài toán.

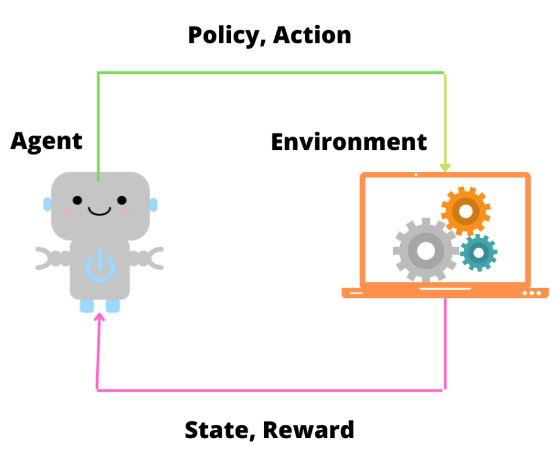
## Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là phương pháp học tăng cường đa tác tử (MARL – Multi-Agent Reinforcement Learning), tức là phương pháp kết hợp giữa học tăng cường và mô hình hoá, mô phỏng đa tác tử. Việc kết hợp này được triển khai bằng cách cài đặt thuật toán học tăng cường Reinforce trên mô hình đa tác tử cho bài toán giao thông trên đường cao tốc, sử dụng nền tảng NetLogo.

# TỔNG QUAN VỀ HỌC TĂNG CƯỜNG

## Tổng quan về học tăng cường

Quá trình học tập là một quá trình liên tục tích lũy kinh nghiệm. Phần thưởng (hay hình phạt) nhận được từ môi trường bên ngoài để hướng dẫn quá trình ra quyết định trong tương lai. Học tăng cường có thể được hiểu là quá trình học điều phải làm bằng cách ánh xạ các tình huống thành hành động để tối ưu hóa phần thưởng nhận được. Chi tiết hơn, học tăng cường là một lĩnh vực của trí tuệ nhân tạo, trong đó các thuật toán được sử dụng để xây dựng các hệ thống hoạt động thông qua việc tương tác giữa tác nhân (agent) với môi trường (environment). Tác nhân sẽ học cách tương tác với môi trường thông qua việc chọn các hành động để đạt được mục tiêu đề ra. Mục tiêu của các tác nhân là tối đa hóa phần thưởng (reward) nhận được từ môi trường, đồng thời tránh các hành động dẫn đến hậu quả không mong muốn.



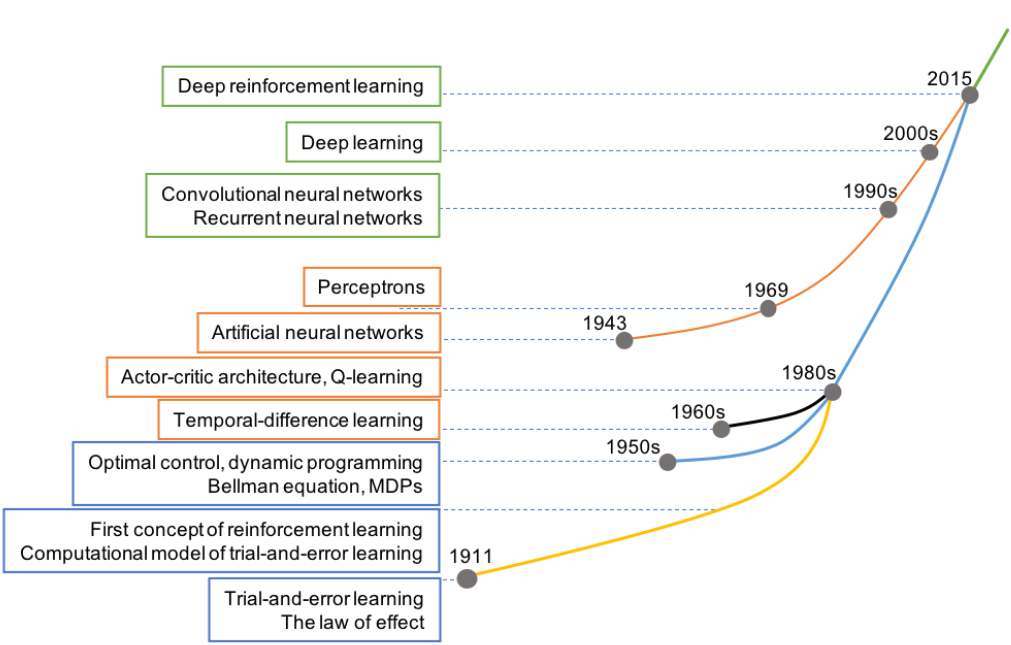
Hình 2.1. Mô hình học tăng cường

Trong học tăng cường, tác nhân được đào tạo thông qua các chu kỳ tương tác giữa nó và môi trường, trong đó nó thực hiện các hành động và nhận được phần thưởng tương ứng. Dựa trên các kết quả của chu kỳ này, tác nhân sẽ cập nhật chiến lược của mình để cải thiện hiệu suất. Tuy nhiên, để đạt được mục tiêu tối ưu, tác nhân cần phải cân bằng giữa việc khám phá các hành động mới và tận dụng các hành động đã biết.

Với các bài toán học tăng cường, chúng ta có thể hình thức hóa bài toán như những quá trình quyết định Markov bởi các quá trình quyết định Markov đều bao gồm 3 khía cạnh đó là cảm giác, hành động và mục tiêu. Học tăng cường được ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm điều khiển robot, chơi game, quản lý tài sản, tối ưu hóa sản xuất và nhiều lĩnh vực khác. Nó đã tạo ra nhiều thành tựu đáng kể trong các lĩnh vực khác nhau như xe tự lái, cờ vua, cờ vây, các hệ thống truyền thông.

## Các cột mốc phát triển của học tăng cường

Nhiệm vụ của các thuật toán học tăng cường là tìm ra chính sách tối ưu nhằm tối đa hóa mục tiêu dài hạn của mình, đây cũng là điều được nghiên cứu trong hàng chục năm nhằm giải quyết các bài toán khác nhau.



Hình 2.2. Các cột mốc phát triển của học tăng cường

Cột mốc đầu tiên của học tăng cường đó là vào năm 1898, Thorndike đã thử nghiệm về hành vi của loài mèo dựa trên quá trình thử sai (TE -Trial and Error). Trong nghiên cứu, Thorndike đặt một con mèo vào một chiếc lồng và để một mảnh thức ăn bên ngoài lồng. Con mèo sẽ phải tìm cách thoát ra khỏi lồng nhưng chỉ có vài cách thành công. Những cách không thành công sẽ bị loại bỏ và con mèo sẽ tiếp tục thử những cách khác cho đến khi tìm ra cách đúng.

Vào những năm 1950-1960: Người ta đã đưa ra các bài toán về quyết định Markov (Markov decision process) cũng như học khác biệt thời gian để nghiên cứu về việc ra quyết định dựa trên thông tin lịch sử của tác nhân trong một môi trường.

Những năm 1980: Tác giả Richard Sutton đã đưa ra thuật toán Q-learning, một trong những thuật toán học tăng cường cơ bản nhất dựa trên lý thuyết kiểm soát tối ưu kết hợp cùng với học khác biệt thời gian. Thuật toán này sử dụng một bảng giá trị (Q-table) để lưu trữ giá trị ước tính của mỗi hành động trong mỗi trạng thái, dựa trên các phần thưởng (rewards) nhận được khi thực hiện hành động đó.

Những năm 2010: Học sâu (deep learning) đã trở thành một lĩnh vực quan trọng trong học máy. Tác giả Volodymyr Mnih và các đồng nghiệp đã phát triển thuật toán Deep Q-Network (DQN), kết hợp giữa học tăng cường và mạng nơ-ron sâu, giúp giải quyết các bài toán học tăng cường với không gian trạng thái (state space) lớn.

AlphaGo của Google DeepMind năm 2016: AlphaGo là một chương trình đánh cờ vây sử dụng kết hợp giữa học tăng cường và mạng nơ-ron sâu. AlphaGo đã đánh bại một trong những kỳ thủ hàng đầu thế giới và đã chứng tỏ sức mạnh của học tăng cường trong việc giải quyết các vấn đề phức tạp.

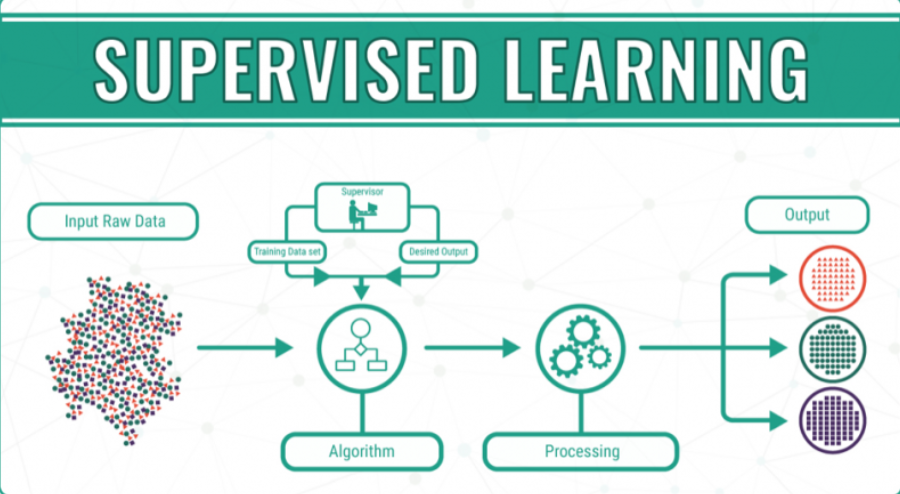
Ngoài ra, GPT (Generative Pre-trained Transformer) của OpenAI năm 2018: GPT là một mô hình học tăng cường sử dụng mạng nơ-ron sâu để tạo ra văn bản tự động. GPT đã đạt được kết quả rất ấn tượng trong việc tạo ra văn bản tự động và đã mở ra nhiều cánh cửa cho ứng dụng của học tăng cường trong lĩnh vực xử lý ngôn ngữ tự nhiên.

## Phân biệt học tăng cường và các phương pháp học máy khác

Học tăng cường và các phương pháp học máy là hai lĩnh vực liên quan đến trí tuệ nhân tạo. Tuy nhiên, chúng cũng có nhiều điểm khác biệt, dưới đây là sự so sánh khác nhau giữa học tăng cường và các phương pháp học máy.

### Học tăng cường và Học có giám sát

Học có giám sát (Supervised Learning) là một phương pháp trong lĩnh vực trí tuệ nhân tạo, trong đó mô hình máy tính được huấn luyện để tìm ra mối quan hệ giữa đầu vào và đầu ra thông qua việc sử dụng các dữ liệu huấn luyện đã được gán nhãn. Các dữ liệu huấn luyện này bao gồm các cặp đầu vào và đầu ra tương ứng, và mô hình được huấn luyện để học cách dự đoán đầu ra tương ứng với mỗi đầu vào. Ví dụ như ta muốn xây dựng một mô hình dự đoán giá nhà dựa trên diện tích và vị trí của căn nhà, chúng ta có thể sử dụng học có giám sát để huấn luyện một mô hình. Trong trường hợp này, dữ liệu huấn luyện sẽ bao gồm các cặp diện tích và vị trí của căn nhà và giá nhà tương ứng với mỗi căn nhà đó. Sau khi được huấn luyện, mô hình có thể dự đoán giá nhà dựa trên diện tích và vị trí của căn nhà.



Hình 2.3. Học có giám sát

Khác nhau giữa học tăng cường và học có giám sát được thể hiện ở bảng dưới đây:

|  |  |
| --- | --- |
| Học có giám sát | Học tăng cường |
| Mục tiêu của học có giám sát là tìm ra mối quan hệ giữa đầu vào và đầu ra thông qua việc sử dụng các dữ liệu huấn luyện đã được gán nhãn | Mục tiêu của học tăng cường là tối đa hóa phần thưởng thông qua việc tương tác trực tiếp với môi trường |
| Dữ liệu đầu vào và đầu ra được gán nhãn và sẵn sàng được huấn luyện | Mô hình phải tương tác với môi trường để học từ kinh nghiệm |
| Học có giám sát thường được sử dụng để giải quyết các bài toán phân loại và dự đoán | Học tăng cường thường được sử dụng để giải quyết các bài toán tối ưu hành động |
| Trong học có giám sát, mô hình được huấn luyện để dự đoán đầu ra dựa trên đầu vào | Với học tăng cường, mô hình phải học cách tương tác với môi trường để đưa ra quyết định tối ưu |
| Học có giám sát cần có một tập dữ liệu huấn luyện lớn và gán nhãn chính xác | Học tăng cường có thể không yêu cầu tập dữ liệu lớn mà chỉ cần một môi trường tương tác |

Bảng 2.1. Sự khác nhau giữa học tăng cường và học có giám sát

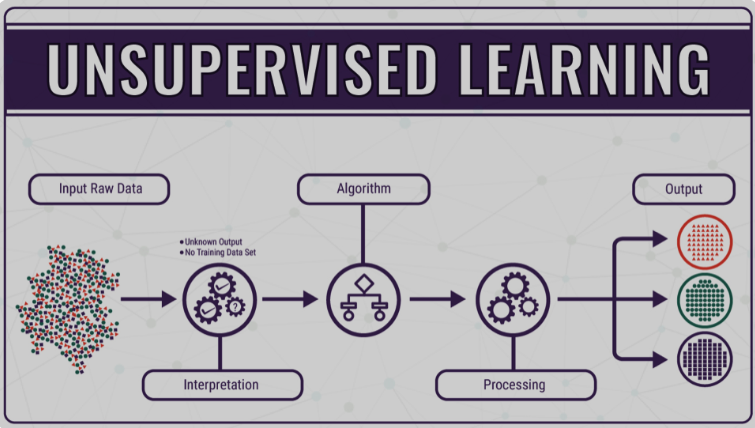
### Học tăng cường và Học không có giám sát

Học không có giám sát (Unsupervised Learning) là một phương pháp học máy trong đó dữ liệu huấn luyện không có nhãn, nghĩa là không có thông tin đầu ra mong muốn của mỗi điểm dữ liệu. Mục tiêu của học không giám sát là tìm ra cấu trúc tiềm ẩn trong dữ liệu mà không cần biết trước nhãn hay phản hồi mong muốn.

Một số phương pháp học không có giám sát bao gồm:

* Clustering: Tập trung vào việc phân nhóm các điểm dữ liệu dựa trên các đặc trưng chung.
* Dimensionality reduction: Giảm số chiều dữ liệu bằng cách tìm ra các đặc trưng quan trọng hoặc biểu diễn dữ liệu trong không gian mới.
* Generative models: Tạo ra dữ liệu mới bằng cách học một mô hình xác suất của dữ liệu huấn luyện.

Ví dụ đơn giản về học không có giám sát như phân nhóm khách hàng dựa trên các thông tin về đặc điểm, hành vi mua hàng hay vị trí địa lý. Các nhóm này có thể được sử dụng tối ưu hóa chiến lược tiếp thị hoặc phục vụ khách hàng tốt hơn.



Hình 2.4. Học không có giám sát

Sự khác biệt giữa học tăng cường và học không có giám sát được biểu thị như sau:

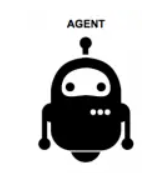
|  |  |
| --- | --- |
| **Học không có giám sát** | **Học tăng cường** |
| Học không có giám sát nhằm tìm ra cấu trúc tiềm ẩn trong dữ liệu | Học tăng cường nhằm tối đa hóa phần thưởng nhận được thông qua tương tác với môi trường |
| Dữ liệu huấn luyện của học không có giám sát không yêu cầu dữ liệu huấn luyện gán nhãn | Dữ liệu huấn luyện học tăng cường chứa phản hồi từ môi trường |
| Học không có giám sát thường có độ khó dễ dàng hơn vì không yêu cầu dữ liệu huấn luyện có nhãn | Thường đối mặt với các bài toán có độ khó phức tạp |
| Ứng dụng trong các ứng dụng như xử lý dữ liệu và phát hiện bất thường trong dữ liệu | Ứng dụng trong các ứng dụng robot tự học, trò chơi hay lái xe tự động |

Bảng 2.2. Sự khác nhau giữa học tăng cường và học không có giám sát

## Các khái niệm cơ bản về học tăng cường

### Tác tử (agent)

Tác tử là khái niệm phổ biến trong các lĩnh vực khác nhau. “Tác tử” trong học tăng cường được định nghĩa như “một thực thể tự trị linh hoạt, có khả năng nhận biết môi trường thông qua các cảm biến được kết nối với nó, và tác động lên môi trường thông qua các bộ truyền động” (Russell & Norvig. 2020). Một cách định nghĩa dễ hiểu hơn, “Tác tử” trong học tăng cường là một phần tử động học, được thiết kế để tương tác với môi trường cụ thể và học cách tối đa hóa phần thưởng được chỉ định.



Hình 2.5. Tác tử (agent)

### Môi trường (Environment)

Môi trường là không gian xung quanh các tác tử, nơi mà tác tử tồn tại và tương tác. Có rất nhiều kiểu môi trường khác nhau trong học tăng cường như môi trường xác định hoặc ngẫu nhiên, môi trường quan sát được toàn bộ, môi trường liên tục hoặc rời rạc, môi trường đơn tác tử hay đa tác tử…

### Mối quan hệ giữa trạng thái (State), Hành động (Action) và phần thưởng (Reward)

Các tương tác giữa tác tử và môi trường được thể hiện qua ba thành phần cốt yếu:

* Trạng thái (s∈S) – S là không gian trạng thái:

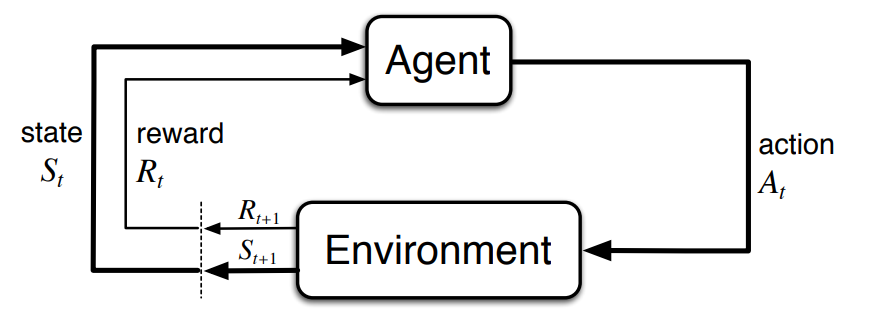
Trạng thái của môi trường là thông tin mô tả về trạng thái hiện tại của hệ thống.

* Hành động (a∈A) – A là không gian hành động:

Các tác tử (agent) có nhiệm vụ học và cập nhật chiến lược liên tục để đạt mục tiêu thông qua các tương tác liên tục với môi trường. Hành động (action) trong học tăng cường là phương thức cho phép các tác tử tương tác với môi trường dựa trên trạng thái của môi trường hiện tại mà tác tử sẽ đưa ra các hành động. Mỗi hành động có thê có một phần thưởng tương ứng.

* Phần thưởng (r∈R) – R là không gian phần thưởng:

Phần thưởng là một giá trị số được cung cấp bởi môi trường cho tác tử sau khi tác tử thực hiện một hành động trong một trạng thái (state) nhất định.



Hình 2.6. Mối quan hệ giữa tác tử và môi trường trong MDP

Dựa vào hình 6, ta có thể thấy tác tử tương tác với môi trường thông qua phương thức hành động (action). Dựa trên trạng thái của môi trường ở thời điểm *t* là St mà tác tử sẽ lựa chọn hành động thích hợp At ở thời điểm đó. Sau đó, môi trường sẽ chuyển trạng thái từ St sang St+1 và tác tử nhận được trạng thái mới là St+1 và phần thưởng phản hồi Rt+1.

### Quá trình quyết định Markov

Quá trình quyết định Markov (Markov Decision Process –MDP) là một khung toàn diện cho việc xây dựng các hệ thống học tăng cường. Nó cung cấp một cách để mô hình hóa tất cả các yếu tố của hệ thống học tăng cường và hầu như tất cả các bài toán học tăng cường đều có thể được hình thức hóa như những quá trình quyết định Markov.

Một quá trình MDP là một bộ năm thành phần gồm:

* S: tập các trạng thái
* A: tập các hành động (ngoài ra, A(s) là một tập các hành động có sẵn từ trạng thái s)
* *P(s’|s, a) = Pr(st+1 = s’, st = s, at = a)* là xác suất thực hiện hành động a trong trạng thái s tại thời gian t sẽ dẫn đến trạng thái s’ tại thời gian t+1.
* *R(s,a,s’):* xác suất phần thưởng nhận được sau khi chuyển tiếp sang trạng thái s’ từ trạng thái s nếu thực hiện hành động a.
* ∈ [0,1] : hệ số chiết khấu đại diện cho sự khác biệt quan trọng giữa các phần thưởng tương lai và phần thưởng hiện tại.

MDP là một công cụ quan trọng để xác định các chính sách tối ưu cho hệ thống học tăng cường. Việc tìm chính sách tối ưu có thể được giải quyết bằng các phương pháp như giá trị lớn nhất (Value iteration) hay phương pháp gradient (Policy Gradient).

### Mô hình và chính sách

Mô hình (model) là mô hình dự đoán của môi trường cho phép tác tử dự đoán kết quả các hành động của nó, có thể được sử dụng để giải quyết hai vấn đề chính của học tăng cường. Một mô hình có thể được sử dụng để giảm thiểu số lần tương tác với môi trường để giảm thiểu chi phí.

* Dự đoán trạng thái tiếp theo: dự đoán xác suất chuyển đổi của các trạng thái và phần thưởng liên quan giữa các thời điểm. Điều này giúp tác tử đưa ra các quyết định tối ưu hơn bằng cách dự đoán hành động nào sẽ dẫn đến kết quả tốt nhất.
* Mô phỏng hệ thống: cho phép tác tử đưa ra các quyết định tối ưu hơn bằng cách thử nghiệm các hành động khác nhau và dự đoán kết quả của chúng.

Chính sách (policy) là phương thức xác định hành vi của tác tử tại một thời điểm nhất định. Chính sách của một tác tử được kí hiệu là πt hay πt(a|s) – là xác suất At = a nếu St = s.

Mục đích của học tăng cường là tìm ra chính sách tối ưu có nghĩa là chính sách tối đa hóa tổng phần thưởng mà tác tử có thể nhận được trong quá trình tương tác với môi trường.

### Các hàm giá trị (Value function)

Hàm giá trị là một hàm có đầu vào là một trạng thái của môi trường còn đầu ra là giá trị dự đoán của phần thưởng tương lai mà tác tử có thể thu được khi đang ở trạng thái đó và tuân theo một chính sách (policy) nhất định.

#### Hàm giá trị trạng thái (state-value function)

Hàm giá trị trạng thái là giá trị kỳ vọng của tổng phần thưởng tác tử nhận được khi bắt đầu từ trạng thái s và tuân theo một chính sách π nhất định. Hàm được tính như sau:

*Vπ(s) = Eπ(Gt | st = s)*

Trong đó:

* Eπ biểu thị giá trị kỳ vọng của một biến ngẫu nhiên mà tác tử theo chính sách π
* t là bước thời gian bất kỳ
* s là trạng thái
* Gt là phần thưởng tương lai

Gt  được tính bằng công thức sau:

*Gt = =*  (γ là hệ số chiết khấu)

#### Hàm giá trị hành động (action-value function)

Hàm giá trị hành động Q(s, a) thể hiện độ tốt của một hành động a tại một trạng thái s:

*Q(s, a) = E(Gt | St = s, At = a)*

Các hàm giá trị trên là những thông tin quan trọng trong học tăng cường bởi vì chúng cung cấp cho tác tử những gợi ý về cách lựa chọn hành động tối ưu trong mỗi trạng thái.

### Các phương trình Bellman

Phương trình Bellman là một khái niệm vô cùng quan trọng trong học tăng cường, nó cung cấp một phương pháp để tính toán giá trị kỳ vọng của một trạng thái hoặc hành động dựa trên giá trị của trạng thái hoặc hành động khác. Có hai phương trình chính trong học tăng cường:

Phương trình Bellman cho hàm giá trị trạng thái:

*V(s) = Eπ(Gt | st = s) =*

Trong đó:

* V(s) là giá trị kỳ vọng của tổng phần thưởng mà tác tử có thể thu được khi bắt đầu từ trạng thái s và tuân theo chính sách π nhất định.
* π(a|s) là xác suất tác tử sẽ thực hiện hành động a khi ở trạng thái s
* π(s’,r|s,a) là xác suất chuyển từ trạng thái s và thực hiện hành động a sang trạng thái s’ và nhận được phần thưởng r.
* là hệ số chiết khấu có giá trị nằm trong khoảng (0,1).

Phương trình Bellman cho hàm giá trị hành động:

*Q(s,a)* =

Trong các phương trình Bellman, cấu trúc của quá trình quyết định MDP được sử dụng để rút gọn phương trình thành phương trình tuyến tính. Các giá trị trạng thái có thể được giải quyết bằng cách giải trực tiếp phương trình tuyến tính.

## Một số thuật toán học tăng cường

### Q – Learning

Thuật toán Q-Learning là thuật toán học tăng cường không có mô hình (model-free) được sử dụng để học cách đưa ra quyết định trong môi trường. Đây là một loại thuật toán học khác biệt theo thời gian, có nghĩa là nó học trực tiếp từ trạng thái hiện tại và phần thưởng nhận được mà không yêu cầu mô hình động lực học của môi trường. Ngoài ra, thuật toán Q-learning còn được gọi là thuật toán học khác biệt thời gian off-policy bởi nó học từ một chính sách (policy) khác với chính sách mà nó sử dụng để tương tác với môi trường.

Để thực hiện thuật toán, ta cần khởi tạo một bảng Q hay còn gọi là Q-table. Q-table (short for Quality Table) là một bảng lưu trữ các giá trị Q-function, bảng Q chứa các giá trị ước tính cho các cặp trạng thái-hành động (state-action) trong môi trường. Mỗi hàng của bảng tương ứng với một trạng thái, mỗi cột tương ứng với một hành động có thể thực hiện trong trạng thái đó. Có hai cách để lựa chọn hành động:

* Khám phá: tác tử sẽ khám phá môi trường bằng cách thử một hành động khác.
* Khai thác: tác tử dựa vào những thông tin đã có về môi trường và lựa chọn hành động cho giá trị tối đa.

Công thức cập nhật giá trị Q:

*Q(st, at) = (1-α)Q(st, at)🡨giá trị cũ + α(rt+1 + γQ(st+1,a’)🡨giá trị học được*

Ta có:

* α là hệ số học – hệ số xác định mức độ mà giá trị Q của các hành động được cập nhật sau mỗi b ước tương tác với môi trường.
* γ là hệ số chiết khấu – là tham số quyết định mức độ ưu tiên của việc đạt được phần thưởng ngay lập tức so với phần thưởng dài hạn.

Dưới đây là các bước để thực hiện thuật toán:

|  |
| --- |
| **Thuật toán Q-Learning** |
| Dữ liệu: α, γ, ε  Kết quả: bảng Q-table  For each episode do:  Khởi tạo t 🡨 0  Bắt đầu với trạng thái đầu tiên S0  Tại thời điểm t, sử dụng ε-greedy, lựa chọn At bằng một hành động ngẫu nhiên với xác suất ε hoặc At = argmaxαQ(St, a)  Thực hiện At, quan sát Rt+1 sau đó tiến đến St+1  Cập nhật giá trị Q:  Q(St, At) 🡨 Q(St, At) + α(Rt+1 + γmaxα∈AQ(St+1, a) - Q(St, At))  Đặt t 🡨 t + 1 và lặp lại từ bước 3  Kết thúc vòng lặp |

### SARSA

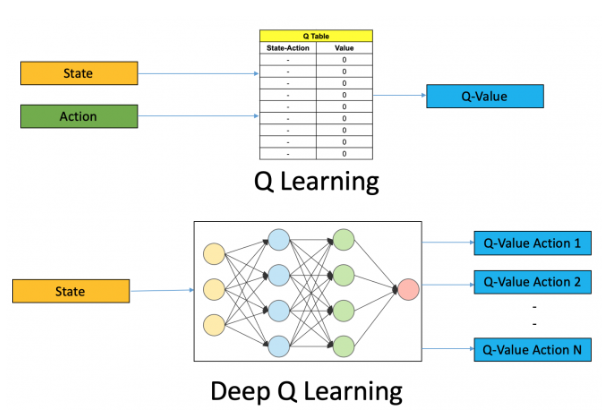
Thuật toán SARSA (State-action-reward-State-action) là một thuật toán học tăng cường được sử dụng để học chính sách tối ưu trong một môi trường Markov quyết định (MDP). Thuật toán SARSA là một thuật toán on-policy, có nghĩa là nó học chính sách tối ưu trong khi đang thực hiện chính sách hiện tại của nó.

Công thức cập nhật giá trị cho Q-Table:

*Qnew(st, at )* 🡨 *Q(st, at) + α[rt + γQ(st+1, at+1) – Q(st, at)*

### Deep Q – Network (DQN)

Trong Deep Q-learning, thuật toán DQN sử dụng mạng nơ-ron sâu (deep neural network) để xấp xỉ giá trị hành động tối ưu. Thuật toán nhận đầu vào là trạng thái của tác tử (state) và giá trị Q của tất cả các hành động có thể tạo ra làm đầu ra. Sự khác nhau giữa Q-Learning và deep Q-Learning:



Hình 2.7. Sự so sánh giữa Q-Learning và Deep Q-Learning

DQN hướng tới cải thiện và làm ổn định đáng kể quá trình huấn luyện của Q-Learning bằng cách sử dụng hai cơ chế: lặp lại trải nghiệm và cập nhật mục tiêu định kỳ.

## Ưu điểm và nhược điểm của học tăng cường và một số ứng dụng của học tăng cường

* Ưu điểm:
* Học tăng cường có thể giải quyết được các vấn đề phức tạp
* Mô hình học tập này rất giống với việc học tập của con người
* Mô hình có thể sửa lỗi xảy ra trong quá trình huấn luyện
* Có khả năng học từ kinh nghiệm, dẫn đến khả năng tối ưu hóa các hành động ngày càng tốt hơn
* Nhược điểm:
* Học tăng cường cần rất nhiều dữ liệu và rất nhiều tính toán. Đó là lí do tại sao nó hoạt động thật sự tốt trong các trò chơi điện tử.
* Không thích hợp để sử dụng giải quyết các vấn đề đơn giản
* Để giải quyết nhiều vấn đề về học tăng cường, chúng ta có thể sử dụng kết hợp học tăng cường với các kỹ thuật thay vì bỏ nó hoàn toàn.
* Quá nhiều học tăng cường có thể dẫn đến quá tải các trạng thái làm giảm kết quả
* Một vài ứng dụng của học tăng cường
* Robotics: Học tăng cường đã được áp dụng rộng rãi trong ngành robot để giải quyết các vấn đề về điều khiển và lập kế hoạch chuyển động. Robot có thể học cách chuyển động, cách thích ứng với môi trường xung quanh và cách tương tác với các đối tượng.
* Trò chơi điện tử: Học tăng cường đã được sử dụng để phát triển các trò chơi video và bot chơi game. Ví dụ, Google DeepMind đã sử dụng học tăng cường để tạo ra một chương trình chơi cờ vây AI có thể đánh bại cả những đối thủ chuyên nghiệp.
* Tự động lái xe: Học tăng cường cũng được áp dụng trong ngành ô tô tự hành. Với các kỹ thuật học tăng cường, một chiếc xe tự lái có thể học cách giữ lái, tương tác với các đối tượng xung quanh và quyết định về hành động tiếp theo.
* Tối ưu hóa quản lý tài chính: Học tăng cường có thể được sử dụng để tối ưu hoá các quyết định về đầu tư tài chính. Hệ thống học tăng cường có thể học cách đưa ra các quyết định về đầu tư, quản lý rủi ro và tối ưu hoá lợi nhuận.

# HỆ ĐA TÁC TỬ

## Khái quát về học tăng cường đa tác tử

Học tăng cường đa tác tử (Multi-agent reinforcement learning) - MARL là một lĩnh vực của trí tuệ nhân tạo ngày càng được quan tâm và phát triển. MARL đã đạt được thành công lớn trong rất nhiều lĩnh vực như game đa người chơi (AlphaGo) hay các hệ thống đa tác tử giao thông, hệ thống phân phối hàng hóa hay các hệ thống phát hiện đối tượng. Các thành công của MARL đã chứng minh khả năng của trí tuệ nhân tạo trong giải quyết các bài toán đa tác tử phức tạp. MARL tập trung vào nghiên cứu hành vi của nhiều tác nhân học tập cùng tồn tại trong một môi trường chung. Mỗi tác nhân được thúc đẩy bởi phần thưởng của riêng mình và thực hiện các hành động để thúc đẩy lợi ích của mình. Trong một số môi trường, những lợi ích này trái ngược với lợi ích của các tác nhân khác, điều này đã dẫn đến động lực nhóm phức tạp. Khi các tác tử tương tác với nhau và với môi trường, ta có thể quan sát được chúng cộng tác, phối hợp, cạnh tranh nhằm hoàn thành nhiệm vụ. Chính vì vậy, MARL có thể được chia thành ba loại: hợp tác, cạnh tranh hoặc kết hợp.

## Hệ đa tác tử là gì?

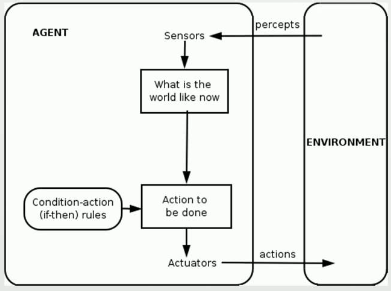
Nhằm giải quyết các bài toán phức tạp như bài toán giảm ùn tắc giao thông trên đường cao tốc, các nhà nghiên cứu đã nghiên cứu nhiều biện pháp điều khiển nhằm làm giảm thiểu sự tắc nghẽn tại các điểm giao thông. Trong thời gian gần đây, các phương pháp mô phỏng giao thông là giải pháp tối ưu cho các nhà nghiên cứu để có thể phân tích và tìm ra hướng giải quyết các vấn đề giao thông. Có rất nhiều phương pháp để mô phỏng nhưng ở đây chúng ta sẽ tìm hiểu chi tiết về “phương pháp mô hình hóa và mô phỏng hóa dựa trên đa tác tử” (Agent-Based Modelling and Simulation –ABMS) - là phương pháp được xem như là một cách tiếp cận đầy hứa hẹn. Về mặt công cụ có thể kể đến một số các nền tảng mô hình hóa và mô phỏng hóa phổ biến như: NetLogo (Wilensky & Rand, 2015), hay GAMA.

Hệ thống đa tác tử (Multi-agent system – MAS) là một hệ thống gồm nhiều tác tử tương tác qua lại với nhau. Mỗi tác tử trong hệ thống đều có thể là một hệ thống đơn lẻ hoặc là một hệ thống MAS nhỏ khác. Có thể hiểu rằng, MAS sinh ra như một phương tiện để giải quyết các vấn đề phức tạp bằng cách chia nhỏ chúng thành các nhiệm vụ nhỏ hơn. Các tác tử trong MAS có thể được chia thành 2 loại chính:

* Tác tử đơn (single-agent): là các tác tử hoạt động độc lập và không cần phụ thuộc vào các tác tử khác để đạt được mục tiêu.
* Tác tử nhóm (group-agent): là nhóm các tác tử tương tác với nhau để thực hiện một nhiệm vụ cụ thể.

Tương tác giữa các tác tử là tính năng quan trọng cho phép các tác tử sử dụng kiến thức về các tác tử khác và tìm hiểu thêm về môi trường trong một khoảng thời gian nhất định. Loại tương tác này có thể là hợp tác hoặc cạnh tranh hay kết hợp cả hai.

* Hợp tác: Trong một tương tác hợp tác, các tác tử sẽ tương tác với nhau và sự tương tác này cho phép các tác tử chia sẻ các kiến thức mà chúng học được.
* Cạnh tranh: Trong một tương tác cạnh tranh, các tác tử có thể cạnh tranh để có được các tài nguyên riêng lẻ và đạt được mục tiêu riêng lẻ.



Hình 3.1. Hệ thống đa tác tử (MAS)

Bên cạnh đó, hệ thống đa tác tử (MAS) có rất nhiều đặc điểm hữu dụng trong việc ứng dụng vào các hệ thống lớn. Dưới đây là một số đặc điểm nổi bật của hệ thống đa tác tử (MAS):

* Đa dạng về tính chất của các tác tử: Trong một hệ thống đa tác tử, các tác tử có thể khác nhau về cấu trúc, nhiệm vụ, mục tiêu và cách thức hoạt động. Tuy nhiên, các tác tử đều cần sự tương tác với nhau để giải quyết vấn đề.
* Tương tác phức tạp: Tương tác giữa các tác tử trong hệ thống đa tác tử có thể rất phức tạp, vì các tác tử không chỉ tương tác với nhau mà còn phụ thuộc vào môi trường xung quanh. Tính tự tổ chức: Các tác tử trong hệ thống đa tác tử thường được tổ chức theo kiểu phân tán và tự động hoá. Các tác tử có khả năng tự điều chỉnh và tự tối ưu hoá hành động của mình để đạt được mục tiêu chung.
* Tính độc lập: Mỗi tác tử trong hệ thống đa tác tử có thể hoạt động độc lập và có thể được thay thế hoặc bổ sung bởi các tác tử khác.
* Môi trường: Mỗi tác tử phải dự đoán hành động của tác tử khác để quyết định hành động tối ưu do các tác tử phụ thuộc vào môi trường. Bên cạnh đó, mỗi tác tử cần phân biệt giữa các tác động gây ra do các hành động của tác tử khác và biến thể chính trong môi trường.
* Tối ưu chi phí: Một hệ thống MAS đơn lẻ có chi phí thấp hơn nhiều so với các hệ thống khác.

## Phân loại kiến trúc các hệ đa tác tử

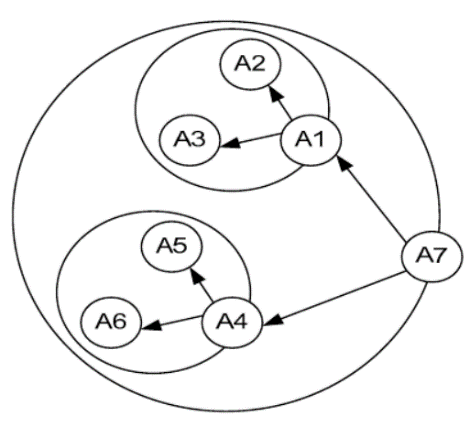
### Kiến trúc tổ chức phân cấp (Hierarchical Organization)

Tổ chức phân cấp được coi như là một trong những thiết kế tổ chức sớm nhất trong các hệ thống đa tác tử. Tại đây, các tác tử được sắp xếp trong cùng một cấu giống cây điển hình. Dữ liệu từ các cấp thấp hơn được chuyển lên trên các tác tử có cấp cao hơn. Ngược lại, tín hiệu giám sát sẽ chuyển từ cao xuống thấp.

Tổ chức phân cấp có thể chia làm 2 loại dựa theo sự phân phối quyền kiểm soát giữa các tác nhân. Đó là hệ thống phân cấp đơn giản và hệ thống phân cấp thống nhất.

### Kiến trúc tổ chức Holonic (Holonic Agent Organization – HAO)

“Holon” là một cấu trúc tương tự hoặc phân dạng ổn định và gắn kết bao gồm một số “holon” làm cấu trúc phụ của nó và bản thân nó là một phần khung lớn hơn. HAO được thiết kế để giải quyết các vấn đề về tính linh hoạt, khả năng mở rộng và hiệu quả của hệ thống đa tác tử. Kiến trúc sử dụng mô hình hệ thống phân cấp theo lớp (layered hierarchical system model) để tổ chức các tác tử thành các lớp khác nhau. Mỗi lớp được xem như là một đơn vị tự trị (holon) có khả năng tự tổ chức và quản lý các tác tử thuộc về nó. Các lớp được tổ chức theo cấu trúc cây, trong đó các lớp cây cao hơn sẽ kiểm soát và quản lý các cấp thấp hơn của nó.



Hình 3.2. Kiến trúc tổ chức Holonic

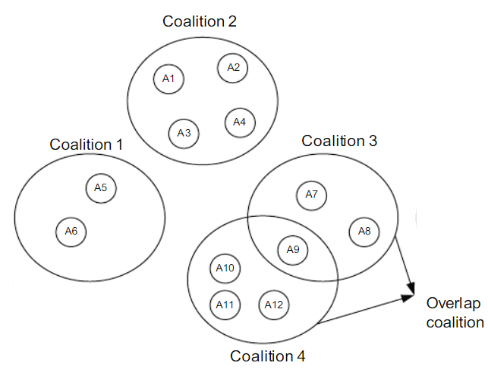
Mỗi tác tử trong HAO được phân loại vào một trong hai loại:

* Holon: là một tác tử độc lập và tự trị
* Holarchy: là một tập hợp các holon có thể kết hợp với nhau tạo thành một hệ thống lớn hơn, có khả năng thích ứng và phát triển linh hoạt.

### Kiến trúc liên minh (Coalitions)

Trong kiến trúc liên minh, một nhóm các tác tử sẽ kết hợp với nhau thành các nhóm nhỏ hơn để tăng tiện ích hoặc hiệu suất của từng tác nhân trong một nhóm. Việc hình thành liên minh phụ thuộc vào bối cảnh và mục tiêu của hệ thống đa tác tử. Ví dụ như:

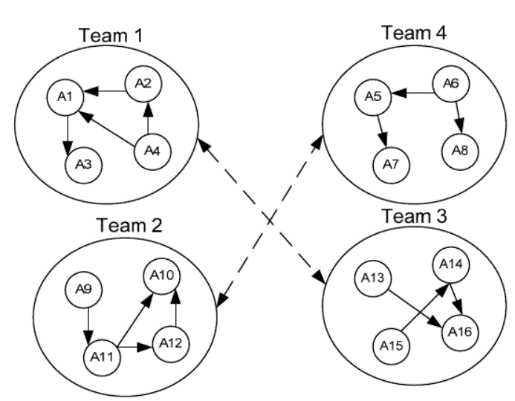
* Trong một môi trường tác chiến, các tác tử có thể hình thành coalitions để tấn công hoặc phòng thủ chống lại các đối thủ chung.
* Trong một môi trường thương mại, các tác tử có thể hình thành coalitions để đàm phán với các bên khác về giá cả hoặc điều kiện hợp đồng.
* Trong một môi trường chia sẻ tài nguyên, các tác tử có thể hình thành coalitions để quản lý và phân phối các tài nguyên chung.



Hình 3.3. Kiến trúc liên minh

Việc hình thành liên minh có thể giúp tác tử đạt được mục tiêu chung một cách hiệu quả đồng thời tăng tính linh hoạt và độ bền của hệ thống đa tác tử. Tuy nhiên, việc hình thành liên minh cũng có thể gây ra những vấn đề như xung đột giữa các nhóm, sự không đồng nhất trong quan điểm và mục tiêu giữa các tác tử.

### Kiến trúc nhóm (Teams)

Kiến trúc nhóm (teams) trong hệ thống đa tác tử là một loại kiến trúc tương tự như kiến trúc liên minh nhưng chú trọng vào sự tương tác và phối hợp giữa các nhóm để đạt được mục tiêu chung. Các nhóm lớn cung cấp khả năng hiển thị tốt hơn về môi trường và lượng thông tin liên quan lớn. Các tác tử trong một nhóm sẽ liên lạc và trao đổi thông tin thông qua các giao thức và hệ thống truyền thông. Mặc dù vậy, cần phải đánh đổi giữa học tập và hiệu suất trong việc lựa chọn quy mô nhóm tối ưu. Điều này làm cho chi phí phát sinh lớn hơn nhiều so với kiến thức trong hệ thống đa tác tử liên minh.

Hình 3.4. Kiến trúc nhóm (Teams)

## Tổng quan về NetLogo

NetLogo được thiết kế bởi Uri Wilensky vào năm 1999, nó được phát triển dựa trên ngôn ngữ lập trình Logo và dựa trên các nền tảng lập trình Smalltalk và Lisp. NetLogo được định nghĩa là ngôn ngữ lập trình đa tác tử và môi trường mô hình hóa để mô phỏng các hiện tượng tự nhiên và xã hội. Kể từ khi được phát triển, NetLogo đã được sử dụng rộng rãi trong giảng dạy và nghiên cứu và được sử dụng trong nhiều lĩnh vực khác nhau như khoa học xã hội, sinh học, kinh tế học cùng nhiều lĩnh vực khác. NetLogo đã được cập nhật liên tục và đang phát triển nhằm cung cấp cho người dùng những tính năng và công cụ mới nhất để giảm thiểu độ phức tạp trong việc xây dựng mô hình và mô phỏng. NetLogo cũng là môi trường cho phép người sử dụng tạo ra các mô hình của riêng họ, ngay cả khi bạn không phải là một người chuyên nghiệp. Đã có một số thành công đáng kể trong việc ứng dụng NetLogo trong một vài lĩnh vực. Có thể kể đến như mô hình hóa sự lan truyền của dịch bệnh, nghiên cứu hành vi tập thể của động vật, mô hình hóa sự lan truyền của ý tưởng và thông tin trên các trang mạng xã hội hay mô phỏng các hệ thống sinh thái...

Với công cụ này, người lập trình viên có thể đưa ra các chỉ dẫn cho các tác tử hoạt động độc lập, giúp cho việc nghiên cứu mối liên kết giữa các hành vi từ mức thấp đến mức cao của các cá thể và sự tương tác nổi bật giữa chúng. Bên cạnh đó, công cụ còn đi kèm với nhiều tài liệu và hướng dẫn cũng như một bộ sưu tập lớn các mô hình mẫu mà tất cả các tác nhân có thể tương tác với nhau và thực hiện đồng thời nhiều nhiệm vụ. Ngoài ra, NetLogo cung cấp một loạt các tính năng và công cụ mô phỏng, bao gồm khả năng thêm các hình ảnh và âm thanh, mô phỏng các môi trường thực tế, cung cấp dữ liệu số liệu thống kê và hiển thị đồ họa cho các kết quả mô phỏng.

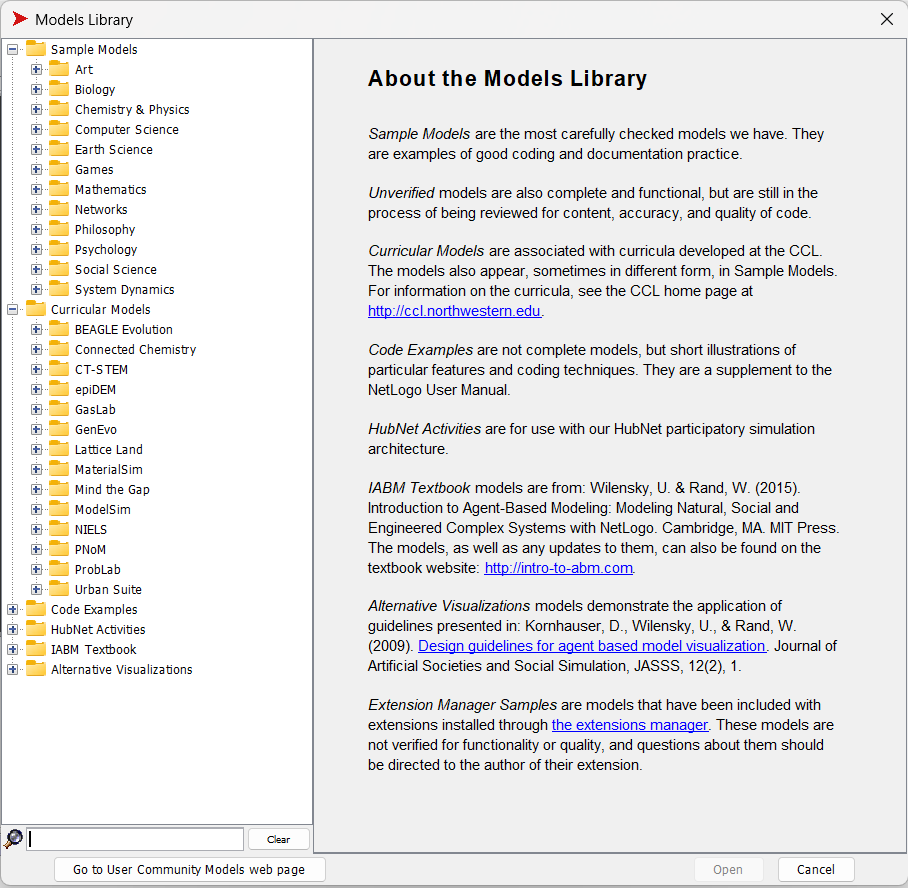
### Đặc điểm của NetLogo

NetLogo là một phần mềm mô phỏng và mô hình hóa dựa trên tác nhân mạnh mẽ với một số tính năng khiến nó trở nên độc đáo và hữu ích trong các ứng dụng khác nhau. Dưới đây là một số đặc điểm của NetLogo:

* NetLogo là phần mềm mã nguồn mở và miễn phí, có nghĩa là người dùng có thể tải xuống và sử dụng nó mà không phải trả bất kỳ khoản phí hoặc hạn chế cấp phép nào
* NetLogo được thiết kế để hỗ trợ mô hình hóa và mô phỏng dựa trên nhiều tác nhân, khiến nó trở nên lý tưởng để nghiên cứu các hệ thống xã hội, sinh thái và sinh học phức tạp.
* NetLogo có giao diện trực quan và thân thiện giúp dễ học và sử dụng, ngay cả đối với những người có ít hoặc không có kinh nghiệm lập trình.
* NetLogo được thiết kế nhanh và hiệu quả, cho phép người dùng mô phỏng các hệ thống lớn và phức tạp một cách nhanh chóng và chính xác
* NetLogo hỗ trợ mô hình hóa và mô phỏng tương tác, cho phép người dùng thay đổi các tham số của mô hình trong thời gian thực và xem kết quả ngay lập tức.
* NetLogo cung cấp các công cụ trực quan hóa đồ họa cho phép người dùng trực quan hóa và phân tích kết quả mô phỏng của họ trong thời gian thực.
* Đa nền tảng: NetLogo là một phần mềm đa nền tảng, có nghĩa là nó có thể chạy trên nhiều hệ điều hành, bao gồm Windows, Mac OS X và Linux.
* Hiển thị và trực quan: Biểu đồ đường thẳng, thanh và phân tán, có thể xem mô hình ở dạng 2D hoặc 3D.

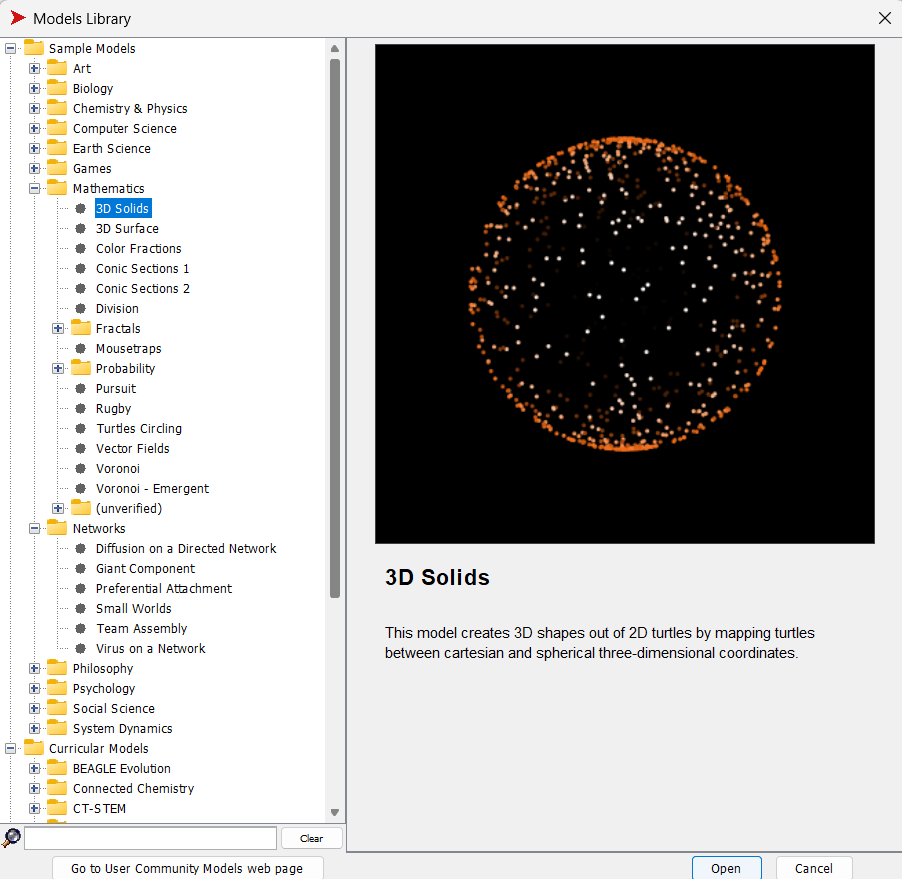
### Một số hình mẫu NetLogo

Trong công cụ NetLogo, có rất nhiều các mô hình mẫu ở tất cả mọi lĩnh vực khác nhau như toán học, mạng, vật lý, sinh học, hóa học...



Hình 3.5. Các mô hình mẫu

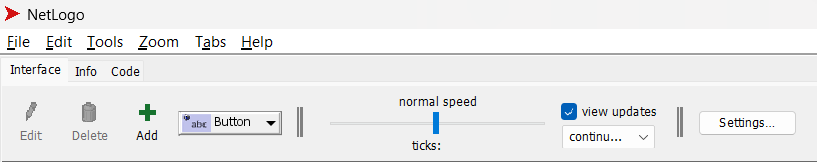
Một số mô hình mẫu trong NetLogo, ví dụ như các hình dạng 3D từ 2D bằng cách ánh xạ giữa các tọa độ ba chiều hình cầu và hình cầu.



Hình 3.6. 3D Solids

### Giao diện NetLogo

Giao diện trong NetLogo có một số thành phần chính:



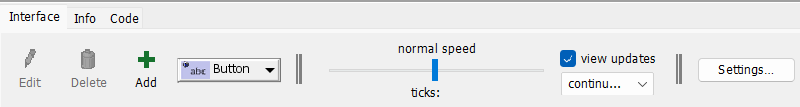
Hình 3.7. Giao diện NetLogo

Thanh Menu chính:



Hình 3.8. Menu chính

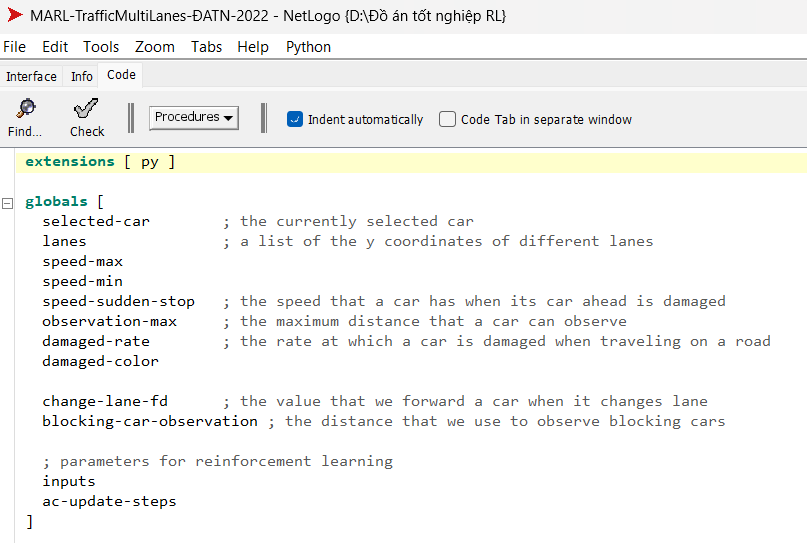
Tabs chính:



Hình 3.9. Giao diện Interface

* *Add*: Thêm một phần tử giao diện
* *Selecting*: Chọn một phần tử giao diện, dùng chuột kéo một hình chữ nhật xung quanh nó.
* *Selecting multiple items*: Chọn nhiều thành phần giao diện cùng lúc bằng cách đưa chúng vào hình chữ nhật.
* *Unselecting*: Bỏ chọn tất cả các thành phần giao diện, hãy nhấp chuột vào nền trắng của tab Giao diện
* *Editing*: Thay đổi các đặc điểm của một phần tử giao diện, hãy chọn phần tử đó, sau đó nhấn nút “Chỉnh sửa”
* *Moving*: Chọn phần tử giao diện, sau đó dùng chuột để kéo nó đến vị trí mới
* *Resizing*: Chọn phần tử giao diện, kéo các đường viền màu đen trong đường viền lựa chọn
* *Deleting*: Xóa phần tử giao diện

Code: Nơi hiển thị code của mô hình



Hình 3.10. Giao diện code

### Lập trình trong NetLogo

Thế giới NetLogo được tạo nên bởi các *tác tử*. NetLogo cung cấp sẵn bốn loại tác tử: *Turtles*, *Patches*, *Links* và *Observer*. Observer là tác tử duy nhất có thể nhìn thấy toàn bộ thế giới và được sử dụng để chạy các phần chính của chương trình (được liên kết với các nút trên giao diện) cũng như cung cấp một cách tương tác lệnh bằng lệnh trên giao diện chính. Các patch đại diện cho các ô vuông (ở dạng 2D) hoặc ô (trong 3D) trên chế độ xem 2D (hoặc 3D) của thế giới. Các turtle là tác tử có thể di chuyển trong thế giới.

Một số lệnh chính được sử dụng và cung cấp sẵn trong môi trường NetLogo như:

* *create n, crt n*: tạo mới n tác tử turtle.
* *clear-all, ca*: xóa các tác tử như turtle, patch, biểu đồ và giá trị đầu vào, đặt lại bộ đếm và các biến toàn cục.
* *forward, fd and right*: các lệnh để điều khiển, thay đổi tác tử turtle.

#### Kiểu biến

NetLogo cung cấp một số kiểu biến khác nhau như:

* *Biến toàn cục*: là biến mà các giá trị có thể truy cập ở bất cứ đâu trong chương trình. Các biến toàn cục được định nghĩa khi chương trình bắt đầu được chạy. Để khởi tạo biến toàn cục ta sử dụng câu lệnh “*globals*”, theo sau là danh sách các biến toàn cục. Ví dụ: *globals* [score]
* *Biến cục bộ*: là biến mà chỉ có thể truy trong một khối mã nhất định. Một khối hay một đoạn mã hoặc trong một thủ tục hoặc hàm. Biến cục bộ được khai báo sau từ khóa “*let*”.
* *Biến được sở hữu bởi các tác tử*: khai báo bằng cách sử dụng hậu tố “-own” trên một loại tác tử hoặc dạng số nhiều của tên loài (breed) và sau đó liệt kê các biến. Ví dụ: *turtles-own* [energy speed]

#### Thủ tục

Cũng như hầu hết các ngôn ngữ lập trình khác, NetLogo hỗ trợ định nghĩa các thủ tục (procedure) để một công việc cụ thể. Một thủ tục định nghĩa một hành động mà các tác tử thực hiện. Mỗi thủ tục có một tên, đứng trước từ khóa *to* hoặc *to-report*, tùy thuộc vào đó là thủ tục lệnh hay thủ tục báo cáo. Từ khóa *end* đánh dấu sự kết thúc của các lệnh trong thủ tục. Ví dụ sau định nghĩa hai thủ tục thường gặp trong NetLogo: *go* và *setup*. Thủ tục *setup* thường được sử dụng để đặt lại các thông số của môi trường như thiết lập giá trị ban đầu cho mô hình, thiết lập lại bộ đếm. Thủ tục *go* dùng để gọi để bắt đầu chạy mô hình.

|  |
| --- |
| *to setup*  *clear-all*  *create-turtles 10*  *reset-ticks*  *end*  *to go*  *ask turtles [*  *fd 1                   ;; forward 1 step*  *rt random 10    ;; turn right*  *lt random 10    ;; turn left*  *]*  *tick*  *end* |

#### Bộ đếm thời gian

Trong nhiều mô hình của NetLogo, thời gian được tính bằng các *ticks*. NetLogo cung cấp một bộ đếm tick để chúng ta có thể quan sát thời gian chạy của một mô hình.

#### Màu sắc

NetLogo thể hiện màu sắc theo những cách khác nhau. Một màu có thể là một số trong phạm vi từ 0 đến 140. Một số màu có tên và chúng ta có thể gọi trực tiếp tên của chúng để sử dụng. NetLogo biểu thị màu sắc dưới dạng RGB (đỏ / lục / lam) và RGBA (đỏ / lục / lam / alpha). Danh sách RGB và RGBA được tạo thành từ ba hoặc bốn số nguyên, tương ứng từ 0 đến 255. Để thiết lập màu cho tác tử turtle, chúng ta sử dụng câu lệnh sau: *set color [mã màu]*. Để thiết lập màu cho tác tử patch, chúng ta sử dụng lệnh: *set pcolor [mã màu]*.

#### Lệnh ask

NetLogo sử dụng lệnh *ask* để đưa ra các lệnh cho các tác tử turtle, patch và link. Tất cả mã do turtle chạy phải được đặt trong một ngữ cảnh cụ thể. Chúng ta có thể thiết lập ngữ cảnh cho tác tử turtle theo 3 cách: 1) bằng nút bấm; 2) bằng dòng lệnh; 3) sử dụng lệnh ask. Ask cũng đc sử dụng một cách tương tự với các tác tử khác.

#### Breeds

NetLogo cho phép chúng ta định nghĩa các “loài” khác nhau. Khi chúng ta đã định nghĩa được loài, chúng ta có thể tiếp tục và làm cho các loài khác nhau cư xử khác nhau. Ví dụ: chúng ta có thể có các loài được gọi là *sheep* và *wolves*, và để những con sói cố gắng ăn cừu hoặc chúng ta có thể có các loài liên kết được gọi streets và sidewalks nơi định tuyến giao thông đi bộ trên vỉa hè và lưu lượng xe ô tô được định tuyến trên đường phố. Để sử dụng các Breeds cú pháp như sau: *breed [tên loài]*

#### Buttons

Các nút trong tab giao diện cung cấp một cách dễ dàng để điều khiển mô hình. Thông thường, một mô hình sẽ có ít nhất một nút “thiết lập”, để thiết lập trạng thái ban đầu của thế giới và một nút “bắt đầu” để làm cho mô hình chạy liên tục. Một số mô hình sẽ có các nút bổ sung thực hiện các hành động khác. Một nút chứa một số mã NetLogo. Mã đó được chạy khi chúng ta nhấn nút. Một nút có thể là “nút một lần” hoặc “nút mãi mãi”. Chúng ta có thể kiểm soát điều này bằng cách chỉnh sửa nút và chọn hoặc bỏ chọn hộp kiểm “Mãi mãi”. Sau khi các nút chạy mã của chúng một lần, sau đó dừng lại và bật lên. Các nút vĩnh viễn tiếp tục chạy mã của chúng lặp đi lặp lại. Nút vĩnh viễn dừng nếu người dùng nhấn lại nút để dừng. Nút đợi cho đến khi quá trình lặp hiện tại kết thúc, sau đó bật lên. Một nút vĩnh viễn cũng có thể được dừng lại từ mã. Nếu nút mãi mãi gọi trực tiếp một thủ tục, thì khi thủ tục đó dừng, nút sẽ dừng. (Trong nút turtle hoặc nút vá mãi mãi, nút sẽ không dừng lại cho đến khi mọi con turtle hoặc patch dừng lại - một con turtle hoặc nút vá không có sức mạnh để dừng toàn bộ nút). Thông thường, một nút được gắn nhãn với mã mà nó chạy. Ví dụ, một nút có nội dung “go” trên đó thường chứa mã “go”, có nghĩa là “chạy quy trình go”. (Các thủ tục được định nghĩa trong tab Mã; xem bên dưới.) Nhưng chúng ta cũng có thể chỉnh sửa một nút và nhập “tên hiển thị” cho nút, là văn bản xuất hiện trên nút thay vì mã. Chúng ta có thể sử dụng tính năng này nếu chúng ta nghĩ rằng mã thực sẽ gây nhầm lẫn cho người dùng của chúng ta. Khi chúng ta đặt mã vào một nút, chúng ta cũng phải chỉ định tác tử nào chúng ta muốn chạy mã đó. Chúng ta có thể chọn để người quan sát chạy mã, hoặc tất cả các con turtle, hoặc tất cả các patch hoặc tất cả các liên kết. (Nếu chúng ta muốn mã chỉ được chạy bởi một số con turtle hoặc một số patch lỗi, chúng ta có thể tạo một nút quan sát và sau đó yêu cầu người quan sát sử dụng ask lệnh để yêu cầu chỉ một số chú turtle hoặcpatch thực hiện điều gì đó). Khi chúng ta chỉnh sửa một nút, chúng ta có tùy chọn để gán một “phím hành động”. Điều này làm cho phím đó trên bàn phím hoạt động giống như một lần nhấn nút. Nếu nút là nút vĩnh viễn, nó sẽ ở chế độ chờ cho đến khi nhấn lại phím (hoặc nút được nhấp). Các phím thao tác đặc biệt hữu ích cho các trò chơi hoặc bất kỳ mô hình nào cần kích hoạt nhanh các nút.

Có thể nhấn nhiều nút cùng một lúc. Nếu điều này xảy ra, các nút sẽ “thay phiên nhau”, có nghĩa là chỉ có một nút chạy tại một thời điểm. Mỗi nút chạy toàn bộ mã của nó một lần trong khi các nút khác đợi, sau đó đến lượt nút tiếp theo.

#### List

Trong các mô hình đơn giản nhất, mỗi biến chỉ chứa một phần thông tin, thường là một số hoặc một chuỗi. Danh sách cho phép chúng ta lưu trữ nhiều phần thông tin trong một giá trị duy nhất bằng cách thu thập thông tin đó trong một danh sách. Mỗi giá trị trong danh sách có thể là bất kỳ loại giá trị nào: một số hoặc một chuỗi, một tác tử hoặc bộ tác tử, hoặc thậm chí một danh sách khác.

Danh sách cho phép đóng gói thông tin thuận tiện trong NetLogo. Nếu nhân viên của chúng ta thực hiện một phép tính lặp lại trên nhiều biến, có thể dễ dàng hơn để có một biến danh sách, thay vì nhiều biến số. Một số nguyên thủy đơn giản hóa quá trình thực hiện cùng một phép tính trên mỗi giá trị trong danh sách.

# THUẬT TOÁN REINFORCE

## Policy Gradient Methods

Phương pháp gradient chính sách (policy gradient methods) là một loại kỹ thuật học tăng cường dựa trên việc tối ưu hóa các chính sách được tham số hóa đối với lợi tức kỳ vọng (phần thưởng tích lũy dài hạn) bằng cách giảm dần độ dốc (gradient). Phương pháp này trực tiếp tối ưu hóa chính sách của một tác nhân trong quy trình quyết định Markov. Trong học tăng cường, một tác nhân tương tác với môi trường, nhận các quan sát và phần thưởng đồng thời học cách thực hiện các hành động để tối đa hóa phần thưởng nhận về theo thời gian.

Trong phương pháp gradient, chính sách đề cập đến chiến lược hoặc hành vi của tác nhân. Nó thường được biểu diễn bằng một hàm được tham số hóa, chẳng hạn như mạng thần kinh, lấy trạng thái hiện tại của môi trường làm đầu vào và đưa ra phân phối xác suất trên các hành động khả dụng. Mục tiêu của các phương pháp độ dốc chính sách là tìm ra chính sách tối ưu để tối đa hóa phần thưởng tích lũy dự kiến. Không giống như các phương pháp dựa trên giá trị cố gắng ước tính giá trị của từng trạng thái hoặc hành động, các phương pháp gradient chính sách trực tiếp tìm kiếm chính sách tốt nhất. Họ làm điều này bằng cách thực hiện cập nhật các tham số chính sách dựa trên độ dốc của phần thưởng tích lũy dự kiến.

Ý tưởng chính của phương pháp: ước tính độ dốc của phần thưởng tích lũy dự kiến ​​đối với các tham số chính sách và sử dụng độ dốc này để cập nhật các tham số theo cách làm tăng phần thưởng dự kiến. Độ dốc biểu thị hướng đi lên dốc nhất trong không gian phần thưởng dự kiến. Điều này thường được thực hiện bằng cách sử dụng các kỹ thuật từ độ dốc tăng dần, trong đó các tham số chính sách được cập nhật bằng cách thực hiện các bước nhỏ tỷ lệ với độ dốc ước tính.

Công thức tính Gradient:

Để ước tính gradient, các phương pháp độ dốc chính sách thường sử dụng một kỹ thuật gọi là định lý policy gradient. Định lý này phát biểu rằng gradient của phần thưởng tích lũy dự kiến ​​có thể được biểu thị bằng tổng gradient của chính sách đối với các hành động, nhân với một giá trị được gọi là "lợi thế". Lợi thế thể hiện sự khác biệt giữa giá trị ước tính của việc thực hiện một hành động trong một trạng thái cụ thể và giá trị mong đợi của việc thực hiện bất kỳ hành động nào trong trạng thái đó.

Trong đó, ta có một hàm chính sách được tham số hóa πθ lấy trạng thái s và hành động a tại thời điểm t làm đầu vào và xuất ra xác suất thực hiện hành động a ở trạng thái s được tham số hóa bởi θ. ∇ đại diện cho toán tử gradient. Giá trị của hàm phần thưởng phụ thuộc vào chính sách này và sau đó có thể áp dụng các thuật toán khác nhau để tối ưu hóa 𝛳 cho phần thưởng tốt nhất.

Hàm phần thưởng được tính như sau:

Trong đó *dπ(s)* là phân phối cố định của chuỗi Markov cho πθ - là xác suất mà st = s khi bắt đầu từ trạng thái đầu tiên dựa theo chính sách tại các bước thời gian t:

Sử dụng gradient ascent, chúng ta có thể di chuyển 𝛳 về hướng được gợi ý bởi gradient để tìm 𝛳 tốt nhất cho chính sách và trả về lợi nhuận cao nhất.

Các phương pháp dựa trên chính sách được mong đợi sẽ hữu ích hơn trong không gian liên tục. Bởi vì có vô số hành động và (hoặc) trạng thái để ước tính các giá trị và do đó, các phương pháp tiếp cận dựa trên giá trị là quá tốn kém về mặt tính toán trong không gian liên tục.

* Ưu điểm:
* Xử lý không gian hành động liên tục:

Thay vì phải chọn một hành động cụ thể từ một không gian lớn, các phương pháp gradient tạo ra một phân phối xác suất trên các hành động, cho phép khám phá các hành động khác nhau trong quá trình tìm kiếm chính sách tốt nhất.

* Khả năng học các chính sách ngẫu nhiên:

Các phương pháp gradient cho phép học các chính sách ngẫu nhiên, tức là các chính sách không tạo ra cùng một hành động trong cùng một trạng thái mỗi lần.

* Khả năng học chính sách phức tạp:

Các phương pháp gradient có khả năng học chính sách phức tạp và phi tham số. Với sự hỗ trợ của các mạng nơ-ron, policy gradient methods có thể học các chính sách với đầu vào phức tạp như hình ảnh hoặc văn bản.

* Có thể tối ưu hóa trực tiếp chính sách: tối ưu hóa chính sách trực tiếp thay vì tối ưu hóa hàm giá trị.
* Nhược điểm:
* Độ lệch cao: Việc ước lượng gradient dựa trên mẫu dẫn đến sự xuất hiện độ lệch trong ước lượng
* Hiệu suất tính toán: khi có một không gian trạng thái và hành động lớn, việc lấy mẫu và tính toán gradient có thể trở nên tốn kém về mặt tính toán.
* Sự không ổn định: Khi tỷ lệ học (learning rate) quá lớn, thuật toán policy gradient có thể trở nên không ổn định và không hội tụ. Tỷ lệ học thích hợp là một tham số quan trọng cần được điều chỉnh cẩn thận.
* Ứng dụng:

Các phương pháp gradient chính sách đã được áp dụng thành công cho nhiều vấn đề học tăng cường, bao gồm: việc huấn luyện các đại diện tự động chơi game, bao gồm cả game cờ vua, game Atari và Dota 2. Các thuật toán này đã thể hiện khả năng học và đánh bại cả những người chơi chuyên nghiệp; có thể học cách điều khiển xe trong các tình huống giao thông phức tạp và đưa ra quyết định an toàn và hiệu quả; áp dụng trong việc điều khiển và huấn luyện robot, Chúng có thể học cách thực hiện các tác vụ phức tạp như điều khiển robot bằng cách đưa ra các hành động với mục tiêu tối ưu hóa hiệu suất hoặc đạt được mục tiêu cụ thể.

## Thuật toán Reinforce

### Vấn đề của gradient chính sách (policy gradient)

Phương sai cao: Các bản cập nhật được thực hiện bởi độ dốc chính sách có phương sai rất cao. Để hiểu lý do tại sao lại như vậy, trước tiên hãy xem xét rằng trong RL, chúng ta đang giải quyết các vấn đề rất chung chung chẳng hạn như dạy ô tô điều hướng trong một môi trường không thể đoán trước hoặc lập trình một tác nhân để hoạt động tốt trên nhiều loại trò chơi điện tử. Do đó, khi lấy mẫu nhiều quỹ đạo từ chính sách chưa được đào tạo của mình, ta nhất định quan sát các hành vi rất hay thay đổi. Điều này trở nên tồi tệ hơn theo cấp số nhân trong các không gian trạng thái hoặc hành động liên tục vì việc truy cập thậm chí gần với mọi cặp hành động trạng thái là không thể tính toán được. Chính vì vậy các cách ước tính Monte Carlo trong gradient chính sách được sử dụng, đánh đổi giữa tính khả thi của tính toán và độ chính xác của gradient.

### Giới thiệu về thuật toán

Thuật toán Reinforce hay còn được gọi là thuật toán gradient chính sách Monte Carlo (Monte Carlo policy gradient), là một ứng dụng cụ thể của các phương pháp Monte Carlo cho việc học tăng cường gradient chính sách. Một định nghĩa khác cụ thể hơn, Reinforce là một thuật toán học tăng cường được sử dụng để tối ưu hóa chính sách trong một môi trường. Đây là một loại phương pháp dựa trên chính sách sử dụng lấy mẫu Monte Carlo để ước tính độ dốc của các tham số chính sách.

Thuật toán Reinforce thường được sử dụng trong các vấn đề với môi trường nhiều tập, trong đó tác nhân tương tác với môi trường trong một số bước thời gian hữu hạn và nhận được phần thưởng ở cuối mỗi tập. Đây là một ví dụ về phương pháp học tăng cường không có mô hình, vì nó không yêu cầu mô hình động lực môi trường và trực tiếp học chính sách thông qua các tương tác thử và sai.

## Các bước thực hiện thuật toán

*Ý tưởng chính của thuật toán*: học hỏi từ các giai đoạn trải nghiệm thô mà không lập mô hình động lực môi trường và tính toán lợi tức trung bình quan sát được dưới dạng gần đúng với lợi nhuận kỳ vọng.

Nhớ lại chiến lược tổng thể về gradient ascent trong gradient chính sách, chiến lược này yêu cầu một cách để lấy mẫu sao cho kỳ vọng của độ dốc mẫu tỷ lệ thuận với gradient thực tế của phép đo hiệu suất dưới dạng một hàm của tham số. Định lý gradient chính sách đưa ra một biểu thức chính xác:

∇J(θ) ∝

= ]

Chúng ta sẽ tạo Reinforce bằng cách giới thiệu At – thay thế một tổng bằng các giá trị có thể có của biến ngẫu nhiên bởi một kì vọng dựa trên π và sau đó lấy mẫu kỳ vọng. Phương trình trên liên quan đến một tổng thích hợp dựa trên các hành động, nhưng mỗi thuật ngữ không được tính theo trọng số bởi *π(a|St, θ).* Vì vậy một trọng số không thay đổi số lượng được sinh ra bằng cách nhân đôi rồi chia các số hạng có tổng cho *π(a|St, θ).* Chúng ta chỉ quan tâm đến một hành động thực sự được thực hiện tại thời điểm t. Chúng tôi làm điều này bằng cách:

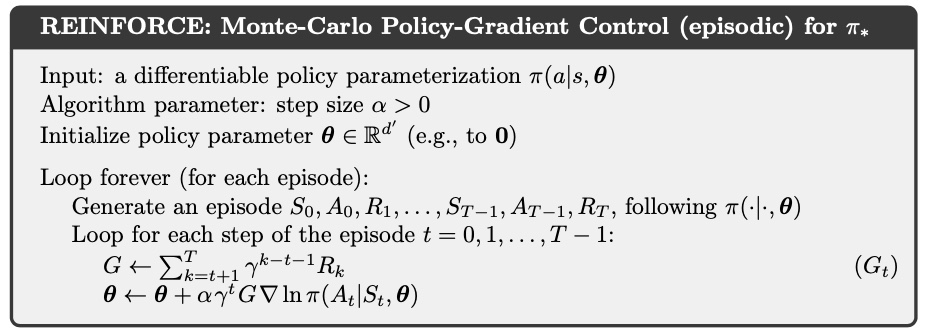
∇J(θ) ∝

= *(replacing a by the sample At ~ π)*

= *(because )*

Biểu thức cuối cùng trong ngoặc là chính xác những gì cần thiết, một đại lượng có thể được lấy mẫu ở mỗi bước thời gian có kỳ vọng tỷ lệ thuận với độ dốc. Do đó, bản cập nhật REINFORCE được đưa ra dưới dạng:

Bản cập nhật làm tăng vectơ tham số theo hướng này tỷ lệ thuận với lợi nhuận và tỷ lệ nghịch với xác suất hành động. Cái trước có ý nghĩa vì nó làm cho tham số di chuyển nhiều nhất theo hướng có lợi cho các hành động mang lại lợi nhuận cao nhất. Cái sau có ý nghĩa bởi vì nếu không thì các hành động được chọn thường xuyên sẽ có lợi thế. Lưu ý rằng Reniforce sử dụng kết quả đầy đủ từ thời điểm t, bao gồm tất cả phần thưởng trong tương lai cho đến khi kết thúc tập. Dưới đây là đoạn mã giả mô tả các bước thực hiện thuật toán:



Hình 4.1. Thuật toán Reinforce

Ta có:

Đầu vào: Một chính sách được tham số hóa *π(a|St,* ***θ****),* tham số của thuật toán tỷ lệ học (α – learning rate)

Bước 1: Khởi tạo tham số của chính sách ***θ***

Bước 2: Thực hiện vòng lặp vô hạn cho mỗi tập

Bước 3: Khởi tạo một tập S0, A0, R1, …, ST-1, AT-1, RT dựa trên chính sách π ban đầu

Bước 4: Thực hiện lặp cho mỗi bước của tập tại các bước thời gian t = 0,1,…,T-1

Bước 5: Tính toán phần thưởng G trả về theo công thức:

G 🡨

Với γ là hệ số chiết khấu (discount factor), giá trị này có thể nằm trong khoảng từ 0,8 đến 0,99.

Bước 6: Cập nhật lại tham số của chính sách:

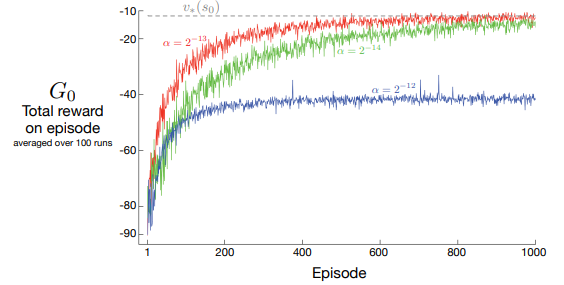
θ 🡨

Một điểm khác biệt ở đoạn mã giả này là sử dụng công thức rút gọn ∇ln *π*(At|St, θt) thay thế cho vecto phân số dưới đây:

Ngoài ra, còn có một điểm khác biệt khác so với quy tắc cập nhật của Reinforce. Đó là hệ số chiết khấu γ, bởi các thuật toán đang đưa ra các thuật toán cho trường hợp chiết khấu chung. Tất cả các ý tưởng thông qua trong trường hợp giảm giá với các điều chỉnh thích hợp nhưng liên quan đến sự phức tạp bổ sung làm mất tập trung vào các ý chính.

## Nhận xét thuật toán

Dưới đây là một biểu đồ cho thấy hiệu suất của thuật toán:



Hình 4.2. Hiệu suất của thuật toán Reinforce

Thuật toán Reinforce là một phương pháp gradient ngẫu nhiên, có sự hội tụ lý thuyết tốt. Theo cách xây dựng, bản cập nhật dự kiến ​​​​trong một tập sẽ theo cùng một hướng như độ dốc hiệu suất. Điều này đảm bảo cải thiện hiệu suất dự kiến ​​cho α đủ nhỏ và hội tụ tới một điểm tối ưu cục bộ trong các điều kiện xấp xỉ ngẫu nhiên tiêu chuẩn để giảm α. Tuy nhiên, thuật toán có thể có phương sai cao dẫn đến việc học chậm.

* Ưu điểm của thuật toán:
* Không có mô hình:

Thuật toán Reinforce là một phương pháp không có mô hình, nghĩa là nó không yêu cầu kiến ​​thức về động lực học của môi trường. Nó trực tiếp học chính sách tối ưu thông qua các tương tác với môi trường. Điều này làm cho nó có thể áp dụng cho nhiều vấn đề trong đó động lực học của môi trường chưa được biết hoặc khó mô hình hóa

* Tối ưu hóa chính sách:

Thuật toán củng cố tối ưu hóa chính sách trực tiếp thay vì ước tính hàm giá trị. Bằng cách trực tiếp tối ưu hóa chính sách, nó có thể xử lý cả không gian hành động rời rạc và liên tục, làm cho nó phù hợp với nhiều nhiệm vụ trong thế giới thực.

* Hội tụ với chính sách tối ưu:

Thuật toán Reinforce đã được chứng minh là hội tụ với chính sách tối ưu khi số lần lặp lại hoặc mẫu đạt đến vô hạn trong các điều kiện nhất định. Thuộc tính hội tụ này đảm bảo về mặt lý thuyết cho hiệu suất của thuật toán.

* Hiệu suất mẫu cao:

Thuật toán củng cố thường yêu cầu ít mẫu hơn so với các phương pháp dựa trên giá trị khác, chẳng hạn như Q-learning hoặc SARSA. Điều này là do nó trực tiếp tối ưu hóa chính sách bằng cách sử dụng độ dốc tăng dần, có thể hiệu quả hơn về mẫu, đặc biệt là trong các không gian hành động liên tục hoặc nhiều chiều.

* Tính linh hoạt:

Thuật toán gia cố có thể được kết hợp với nhiều kỹ thuật khác nhau để nâng cao hiệu suất và tính ổn định của nó. Chúng bao gồm phép trừ cơ sở để giảm phương sai, chính quy hóa entropy để khuyến khích khám phá và các kỹ thuật như Tối ưu hóa chính sách khu vực tin cậy (TRPO) và Tối ưu hóa chính sách gần nhất (PPO) để cập nhật chính sách ổn định hơn.

* Bên cạnh đó, thuật toán cũng còn một số hạn chế như:
* Phương sai cao:

Một hạn chế lớn của thuật toán là phương sai cao trong ước tính gradient. Phương sai cao này có thể dẫn đến sự hội tụ chậm và gây khó khăn cho việc tìm hiểu các chính sách tối ưu, đặc biệt là trong các môi trường có phần thưởng thưa thớt hoặc quỹ đạo dài.

* Thiếu thăm dò:

Reinforce vốn không có cơ chế thăm dò tích hợp. Nó dựa trên hành vi thăm dò phát sinh từ chính sách ngẫu nhiên. Trong một số môi trường, điều này có thể không đủ để khám phá không gian trạng thái hành động một cách hiệu quả, đặc biệt là trong các môi trường lớn hoặc phức tạp với phần thưởng thưa thớt.

* Tối ưu cục bộ:

Củng cố là một phương pháp tối ưu hóa dựa trên độ dốc dễ bị mắc kẹt trong tối ưu cục bộ. Nó có thể đấu tranh để thoát khỏi các chính sách dưới mức tối ưu và có thể yêu cầu khởi tạo cẩn thận hoặc các kỹ thuật bổ sung như chuẩn hóa entropy hoặc ràng buộc vùng tin cậy để giảm thiểu vấn đề này.

* Độ nhạy với siêu tham số:

Thuật toán củng cố có một số siêu tham số cần được điều chỉnh cẩn thận để có hiệu suất tối ưu. Việc lựa chọn tỷ lệ học tập, hệ số chiết khấu, đường cơ sở và tham số khám phá có thể ảnh hưởng đáng kể đến sự hội tụ và ổn định của thuật toán.

* Thiếu ước tính giá trị:

Không giống như các phương pháp dựa trên giá trị, thuật toán Củng cố không ước tính rõ ràng các hàm giá trị. Các hàm giá trị có thể cung cấp thông tin có giá trị về phần thưởng dự kiến ​​trong tương lai và có thể hỗ trợ tìm hiểu các chính sách tốt hơn. Nếu không ước tính giá trị, Reinforce có thể gặp khó khăn trong việc xử lý các tác vụ trong đó việc ước tính giá trị là rất quan trọng.

# THỬ NGHIỆM THUẬT TOÁN REINFORCE CHO MÔ HÌNH BÀI TOÁN GIẢM ÙN TẮC GIAO THÔNG TRÊN ĐƯỜNG CAO TỐC

Để thử nghiệm ứng dụng phương pháp học tăng cường cho các hệ thống đa tác tử, chúng tôi đã tiến hành cài đặt một số thuật toán học tăng cường cho bài toán tránh ùn tắc trên đường cao tốc nhiều làn xe. Trong đó, mỗi phương tiện giao thông, i.e. xe ô tô, được mô hình hoá thành một tác tử.

## Giới thiệu bài toán giảm ùn tắc giao thông trên đường cao tốc

Từ lâu, vấn đề tắc nghẽn giao thông, gây ra bởi sự bùng nổ của các phương tiện giao thông trong các thành phố lớn, đã trở thành một bài toán nan giải tại các nước đang phát triển, trong đó có Việt Nam. Trên đường cao tốc, khi phần lớn phương tiện giao thông là các xe ô tô, sự ùn tắc có thể xảy ra khi xuất hiện tai nạn giao thông hoặc tại các điểm thu phí, nhất là trong các kỳ nghỉ lễ, khi một số lượng lớn phương tiện giao thông di chuyển rời thành phố để đi du lịch hoặc về quê, và khi chúng cùng trở lại thành phố sau kì nghỉ. Dưới đây là mô hình mô phỏng đơn giản hệ thống xe ô tô điều khiển tự động trên nền tảng mô phỏng hóa NetLogo.

## Mô hình bài toán

Để giải quyết bài toán ùn tắc giao thông trên đường cao tốc, chúng tôi đã sử dụng lại mô hình đa tác tử “*Traffic 2 Lanes*” (Wilensky & Payette, 1998), được cung cấp sẵn trong thư viện của nền tảng NetLogo.

Chúng tôi đã mở rộng mô hình “*Traffic 2 Lanes”* bằng cách thêm vào các tình huống phương tiện gặp sự cố khi di chuyển và cài đặt thuật toán học tăng cường Reinforce cho mô hình này.

## Các thành phần chính trong mô hình

Các thành phần học tăng cường chính cho bài toán tránh ùn tắc giao thông, i.e., trạng thái, hành động, phần thưởng, được miêu tả sau đây.

### Trạng thái

Với thuật toán Reinforce, trạng thái của một tác tử được định nghĩa là một tập gồm 4 yếu tố:

* Vận tốc hiện tại của phương tiện
* Vận tốc hiện tại của phương tiện cản trở gần nhất
* Khoảng cách tới phương tiện cản trở
* Giá trị hiện tại của độ kiên nhẫn (được dùng để ra quyết định chuyển làn)

### Hành động

Trong mô hình của chúng tôi, một phương tiện có thể lựa chọn thực hiện một trong bốn hành động sau:

* Tăng tốc
* Giảm tốc
* Giữ nguyên vận tốc
* Chuyển làn

### Phần thưởng

Chúng tôi cài đặt hàm phần thưởng là một hàm của vận tốc, theo công thức sau:

## Các chiến lược di chuyển

Trong mô hình này, chúng tôi đã cài đặt và sử dụng một số chiến lược di chuyển cho các phương tiện giao thông, bao gồm: *chiến lược tham lam* (Greedy) và các *chiến lược học tăng cường* (Reinforce)

Các chiến lược này được miêu tả ngắn gọn như sau:

### Chiến lược tham lam:

Đây là chiến lược đã được cài đặt sẵn trong mô hình “*Traffic 2 Lanes*” (Wilensky & Payette, 1998). Với chiến lược này, khi di chuyển, các phương tiện luôn có xu hướng tăng tốc bất kỳ khi nào có thể. Tuy nhiên, khi chúng phát hiện có một phương tiện khác ở trước cản trở quá trình di chuyển, trong khả năng quan sát, chúng sẽ chạy chậm lại. Trong trường hợp phương tiện cản trở là một phương tiện bị hư hỏng, đang dừng trên đường, chúng sẽ thực hiện chuyển làn.

### Chiến lược học tăng cường (Reinforce):

Chiến lược này sử dụng thuật toán Reinforce để giúp các phương tiện lựa chọn hành động trong quá trình di chuyển.

## Các khu vực giao diện của mô hình

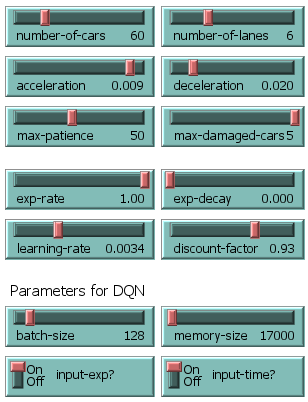
Giống như các mô hình được phát triển sử dụng nền tảng NetLogo, giao diện của mô hình tránh ùn tắc giao thông trên đường cao tốc nhiều làn của chúng tôi có thể chia thành 3 khu vực chính:

* Khu vực căn chỉnh tham số đầu vào
* Khu vực hiển thị kết quả mô phỏng
* Khu vực quan sát quá trình mô phỏng

### Khu vực căn chỉnh tham số đầu vào

Khu vực này cho phép người dùng căn chỉnh các tham số đầu vào của mô hình. Trong mô hình này, chúng tôi sử dụng các tham số đầu vào như sau:  *số lượng phương tiện*; *số lượng làn*; *hệ số tăng tốc*; *hệ số giảm tốc*; *độ kiên nhẫn tối đa của tài xế* (để ra quyết định chuyển làn khi giá trị bằng 0); *số lượng xe hư hỏng tối đa* (trong quá trình di chuyển trên cao tốc); *hệ số khám phá*; *hệ số phân rã khám phá* (để giảm tỷ lệ khám phá của tác tử theo thời gian); *tốc độ học* (α); *hệ số chiết khấu* (γ)

Minh họa khu vực này được thể hiện ở hình dưới đây:

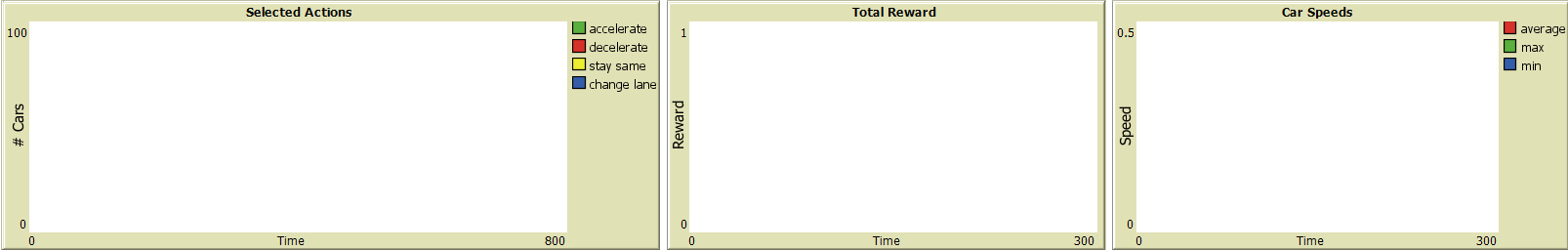


Hình 5.1. Khu vực căn chỉnh tham số đầu vào

### Khu vực hiển thị kết quả mô phỏng

Chúng tôi sử dụng nhiều biểu đồ thể hiện nhiều kết quả, tương ứng với các khía cạnh khác nhau của mô hình, theo thời gian thực. Ví dụ: *số lượng phương tiện trên mỗi làn tại một thời điểm; độ kiên nhẫn tối đa, tối thiểu và trung bình của các tài xế; tốc độ tối đa, tối thiểu và trung bình của các phương tiện; tổng phần thưởng của tất cả tác tử, tỷ lệ các hành động được chọn*

Hình dưới đây minh họa các biểu đồ của khu vực hiển thị kết quả mô phỏng:



Hình 5.2. Khu vực hiển thị kết quả mô phỏng

### Khu vực quan sát quá trình mô phỏng

Khu vực này được thể hiện ở hình sau cho phép chúng ta quan sát được quá trình mô phỏng theo thời gian thực:

A picture containing text

Description automatically generated

Hình 5.3. Khu vực quan sát quá trình mô phỏng

Ở chính giữa khu vực này là một đường cao tốc (màu đen), với nhiều làn đường được phân cách bởi các đường thẳng nét đứt. Đây là một đường cao tốc vô hạn, quay vòng. Khi một phương tiện di chuyển đến lề bên phải và biến mất, nó sẽ xuất hiện lại ở từ phía lề trái, trong cùng làn.

Trên đường cao tốc, các phương tiện di chuyển từ trái sang phải, với các tốc độ khác nhau. Khi một phương tiện gặp sự cố, thuộc tính tốc độ của nó sẽ được đặt bằng 0. Màu sắc của các phương tiện thể hiện hành động mà nó lựa chọn, với quy ước như sau: màu đỏ – thể hiện hành động giảm tốc; màu xanh lục – thể hiện hành động tăng tốc; màu vàng – thể hiện hành động giữ nguyên vận tốc; và màu xanh lam – thể hiện hành động chuyển làn.

## Thực nghiệm và đánh giá kết quả

### Thiết lập môi trường

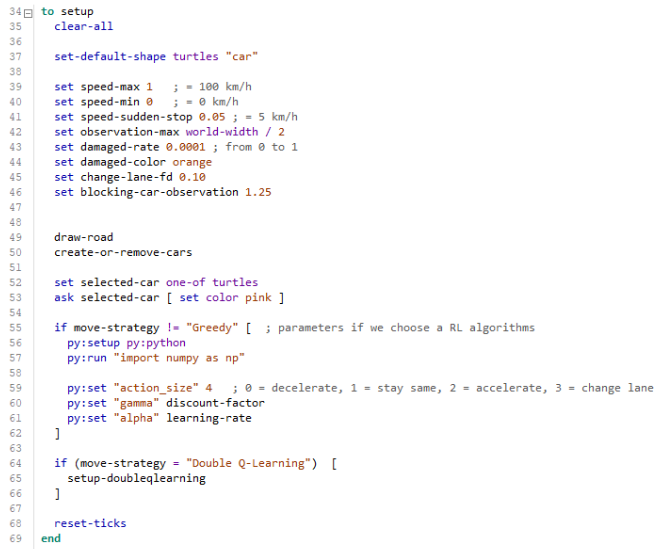
|  |  |  |
| --- | --- | --- |
| **Tham số** | **Ý nghĩa** | **Giá trị** |
| *number-of-cars* | Tổng số phương tiện | 60 |
| *number-of-lanes* | Số làn trên đường cao tốc | 4 |
| *acceleration* | Hệ số tăng tốc | 0.005 |
| *deceleration* | Hệ số giảm tốc | 0.005 |
| *max-patience* | Độ kiên nhẫn – số lần giảm tốc tối đa trước khi tài xế quyết định chuyển làn | 50 |
| *max-damaged-cars* | Số lượng tối đa phương tiện bị hư hỏng trong quá trình di chuyển | 3 |
| *exp-rate* | Giá trị khám phá ban đầu | 1.0 |
| *exp-decay* | Hệ số phân rã giá trị khám phá theo thời gian | 0.001 |
| *learning-rate* | Tỷ lệ học | 0.0034 |
| *discount-factor* | Hệ số chiết khấu | 0.98 |
| *batch-size* | Kích thước lô, dùng cho DQN | 128 |
| *memory-size* | Kích thước bộ nhớ kinh nghiệm, dùng cho DQN | 10,000 |
| *simulation-time* | Thời gian mô phỏng | 50,000 |

Bảng 5.1. Các tham số môi trường

Vận tốc của một phương tiện là một số thực, trong khoảng từ 0.0 đến 1.0, tương ứng với vận tốc tối thiểu là 0 km/h và vận tốc tối đa là 100 km/h. Ban đầu, vận tốc của các phương tiện được khởi tạo là một số ngẫu nhiên, quanh giá trị 0.5 (50 km/h).

### Cài đặt chung cho các thuật toán

* Import extentions và khai báo biến cục bộ
* *selected-car*: ô tô được chọn hiện tại
* *lanes*: danh sách tọa độ y theo các làn đường khác nhau
* *speed-max*: tốc độ tối đa
* *speed-min*: tốc độ tối thiểu
* *speed-sudden-stop*: tốc độ của một ô tô khi ô tô phía trước đột nhiên dừng lại
* *observation-max*: khoảng cách tối đa mà một ô tô có thể quan sát
* *damaged-rate*: tỷ lệ xe bị hư hỏng
* *damaged-color*: màu sắc xe bị hư hỏng trên đường
* *change-lane-fd*: khoảng cách để chuyển làn
* *blocking-car-observation*: khoảng cách để nhìn thấy xe đang bị hư hỏng
* *inputs*: nhập các giá trị đầu vào cho mô hình
* Các thuộc tính của phương tiện:
* *speed*: vận tốc hiện tại của ô tô
* *speed-top*: vận tốc tối đa của ô tô
* *target-lane*: làn đường đang chạy của ô tô
* *patience*: tính kiên nhẫn
* *reward*: phần thưởng cho ô tô
* *state*: trạng thái của ô tô
* *action*: hành động 0 : giảm tốc; 1 : giữ nguyên; 2 : tăng tốc; 3 : chuyển làn
* *next-state*: trạng thái tiếp theo của ô tô
* Hàm setup môi trường để khởi chạy thuật toán:
* clear-all: xóa các tác tử như turtle, patch, biểu đồ và giá trị đầu vào, đặt lại bộ đếm và các biến toàn cục.
* py:set “action\_size” 4: gán giá trị cho số lượng action
* py:set “gamma” discount-factor: gán giá trị cho hệ số chiết khấu
* py:set “alpha” learning-rate: gán giá trị cho hệ số học
* move-strategy: biến dùng để xác định xem mô hình sử dụng chiến lược
* reset-ticks: khởi tạo bước thời gian về giá trị ban đầu



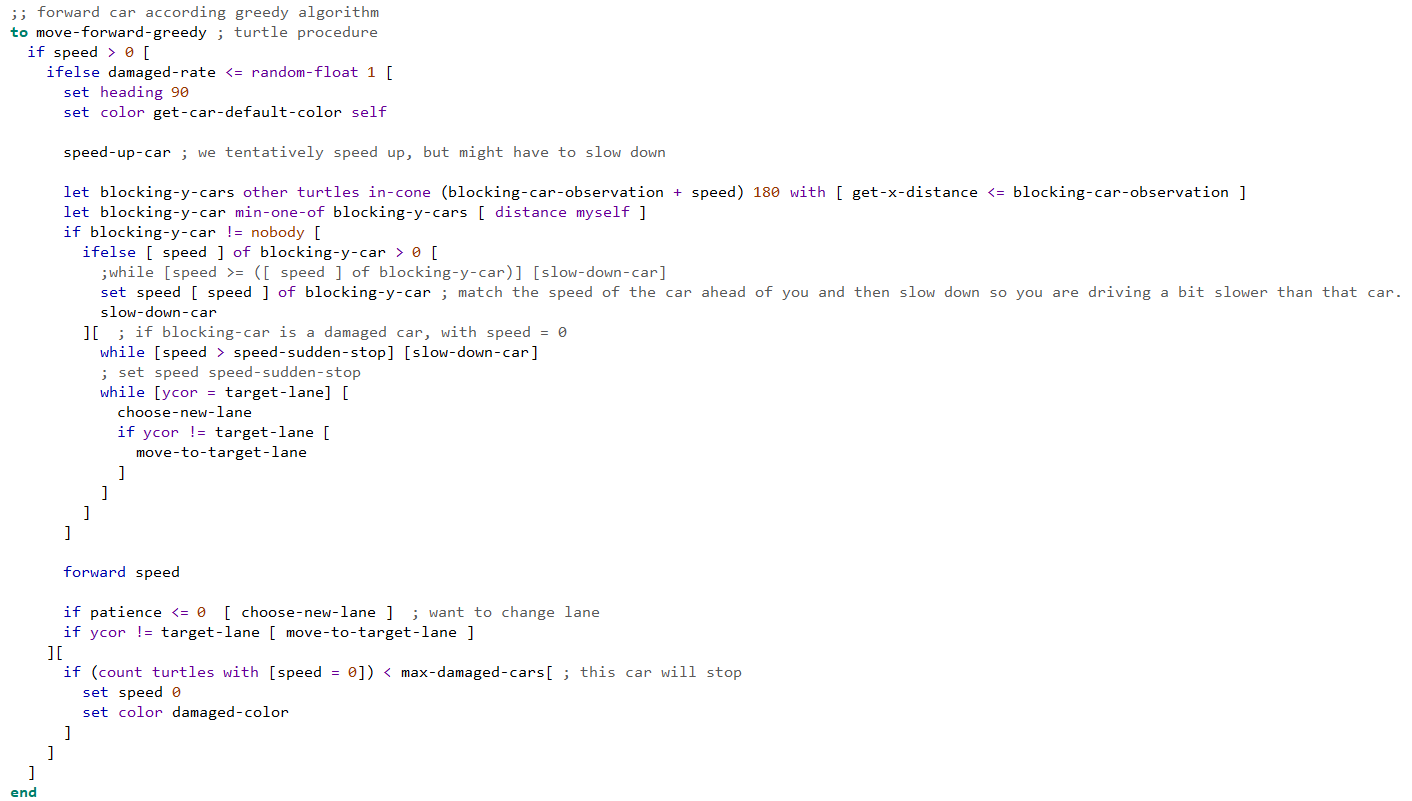
Hình 5.4. Hàm setup thuật toán

Hàm go cũng được định nghĩa như một thủ tục, sử dụng move-stragegy để biết chiến lược học tăng cường nào đang chạy mô hình.



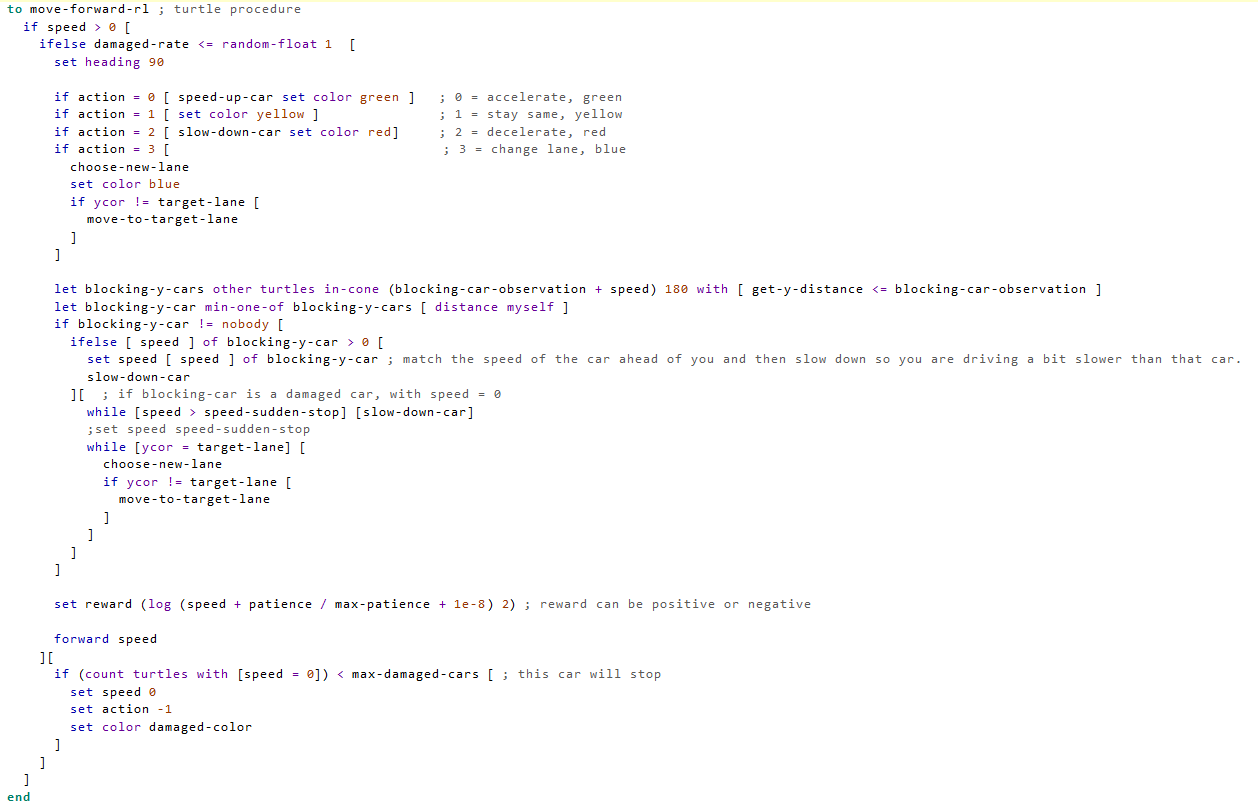
Hình 5.5. Hàm go

Hàm di chuyển cho chiến lược tham lam:



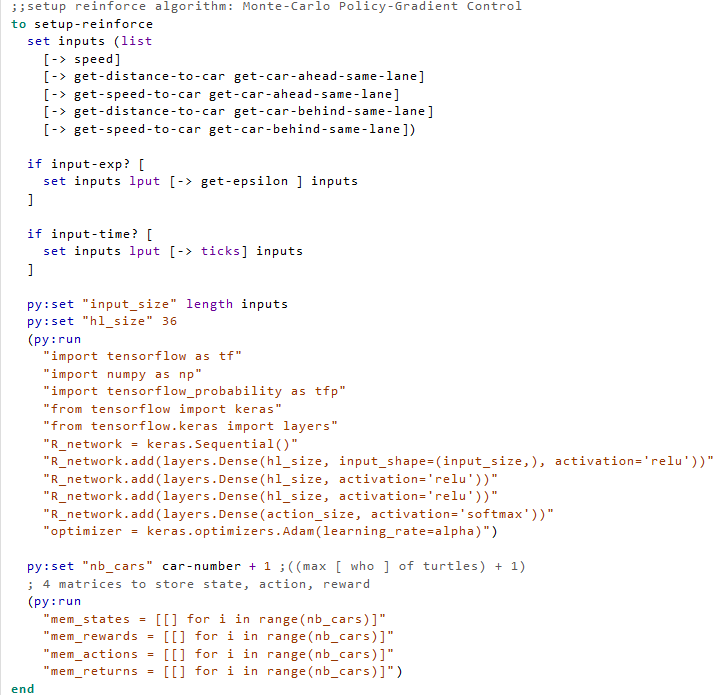
Hình 5.6. Hàm di chuyển cho chiến lược tham lam

Hàm di chuyển cho chiến lược học tăng cường:



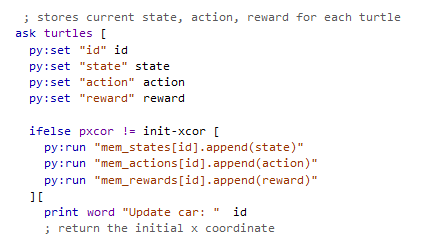
Hình 5.7. Hàm di chuyển cho chiến lược học tăng cường

### Cài đặt thuật toán Reinforce



Hình 5.8. Hàm setup thuật toán Reinforce

Để thực hiện thuật toán ta sẽ khởi tạo id và thêm các state, action, reward vào bộ nhớ đã tạo ở hàm setup. Sử dụng *pxcor* để trả về vị trí x ban đầu khi phương tiện đi từ trái qua phải.



Sau đó áp dụng các công thức để tính tổng phần thưởng trả về cũng như độ mất mát của thuật toán.



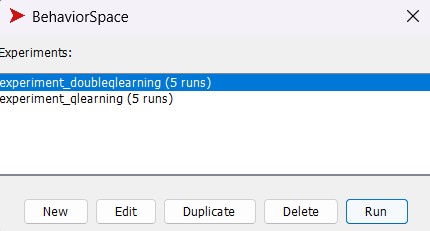
### So sánh giữa các chiến lược học tăng cường

Để so sánh hai chiến lược học tăng cường, chúng tôi sử dụng tính năng “Behavior Space” trong nền tảng NetLogo:

Behavior Space là một tính năng của NetLogo cho phép người dùng chạy một mô hình nhiều lần, thay đổi các cài đặt của mô hình một cách có hệ thống và ghi lại kết quả của mỗi lần chạy mô hình. Quá trình này đôi khi còn được gọi là “quét tham số”. Một ví dụ đơn giản cho tính năng này như giả sử một mô hình có bốn tham số có thể cài đặt, mỗi tham số trong số đó có thể có các giá trị nguyên từ 1 đến 10. Nếu bạn muốn xem kết quả của mọi kết hợp các giá trị tham số, thì sẽ có 10 \* 10 \* 10 \* 10 = 10.000 kết hợp để chạy, mà hoàn toàn không thể làm bằng tay. Sử dụng BehaviorSpace, bạn có thể thiết lập loạt kiểm tra và ghi lại kết quả của từng lần và tất cả kết quả của các lần chạy sẽ được tổng hợp trong một bảng tính định dạng.

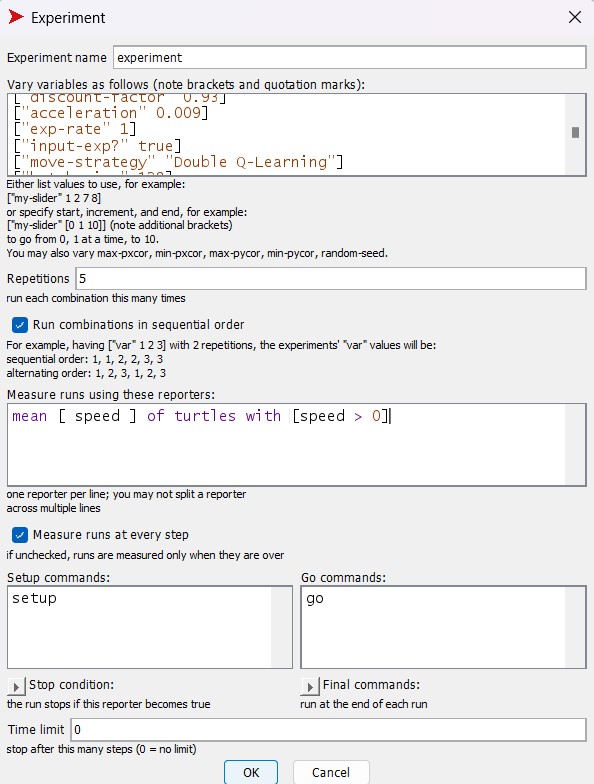
#### Cách sử dụng

Đầu tiên, mở mô hình trong NetLogo sau đó vào Tools chọn Behavior Space



Hình 5.9. Behavior Space

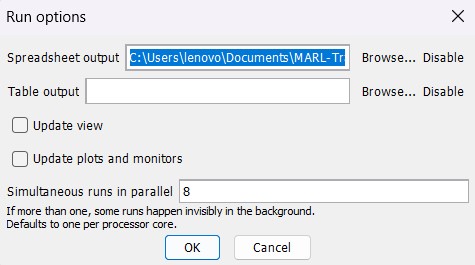
Hộp thoại đầu tiên mở ra là danh sách thử nghiệm (Experiments) của bạn. Thử nghiệm là tập hợp các lần chạy được xác định bởi các bộ tham số sẽ được kiểm tra trong các lần chạy đó và các phép đo sẽ được lấy cho mỗi người. Tạo một thử nghiệm mới bằng cách ấn vào “New”, một hộp thoại khác sẽ xuất hiện với các giá trị mặc định:



Hình 5.10. Hộp thoại Experiment

* “Vary variable as follows” là nơi xác định các tham số được sử dụng trong mô hình
* “Repetitions” là nơi xác định số lần chạy ở mỗi tổ hợp cài đặt
* “Measure runs using these reporters” là nơi chỉ định dữ liệu nào bạn muốn thu nhập tử mỗi lần chạy. Ở đây, chúng tôi sử dụng vận tốc trung bình làm dữ liệu muốn thu nhập
* “Setup Commands”: các lệnh này sẽ được dùng để bắt đầu mỗi lần chạy mô hình
* “Go commands”: các lệnh này sẽ được chạy đi chạy lại để chuyển sang mô hình “bước” tiếp theo
* “Time limit”: điều này cho phép bạn đặt độ dài tối đa cố định cho mỗi lần chạy. Nếu bạn không muốn đặt bất kỳ giá trị tối đa nào, nhưng muốn thời lượng chạy được kiểm soát bởi điều kiện dừng thay vào đó, hãy nhập 0.

Để chạy thử nghiệm ta sử dụng “Run” để chạy:

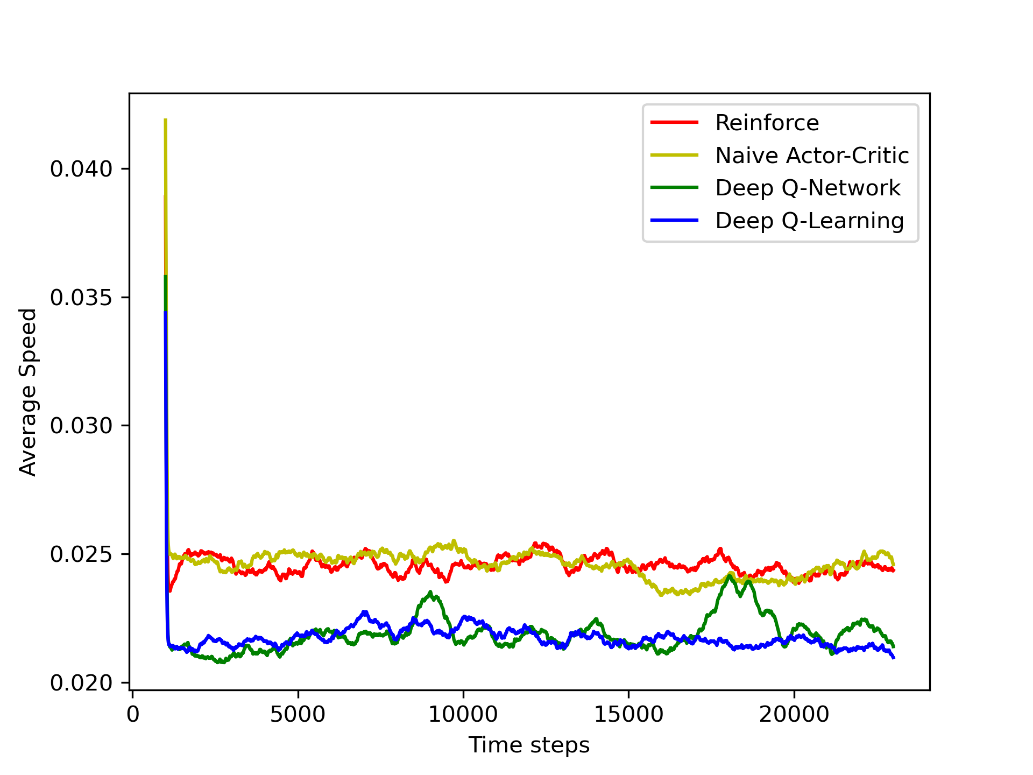


Hình 5.11. Hộp thoại Run

Behavior Space cung cấp hai tùy chọn để xuất dữ liệu. Sau khi nhấp vào “OK” thử nghiệm sẽ chạy và cuối cùng xuất mọi thứ ở định dạng .csv

### So sánh giữa các thuật toán học tăng cường

Dưới đây là kết quả so sánh 4 chiến lược học tăng cường bao gồm Deep Q-Learning, Deep Q-Network, Navie Actor-Critic và Reinforce với dữ liệu so sánh là vận tốc trung bình của tất cả các phương tiện sau một lần thử nghiệm tại từng thời điểm trong 23000 bước thời gian mô phỏng.



Biểu đồ 5.1. Biểu đồ so sánh giữa các chiến lược học tăng cường

Dựa vào biểu đồ này, có thể thấy thuật toán Deep Q-Network (đường xanh lá) và thuật toán Reinforce (đường màu đỏ) hoạt động tốt hơn so với hai chiến lược còn lại. Ngoài ra về thời gian chạy, cả bốn chiến lược này đều thực hiện so sánh với cấu hình Window 10 Pro 64 bit, Processor Intel(R) Core i3-9100F CPU 3.60GHz, RAM 8GB thu được kết quả chiến lược Reinforce có thời gian chạy (6 tiếng) lâu hơn so với ba chiến lược còn lại (1 tiếng 30 phút).

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Trong khoảng thời gian thực hiện đồ án tốt nghiệp với đề tài “Thử nghiệm thuật toán Reinforce cho mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc”, em đã nghiên cứu về phương pháp học tăng cường đa tác tử và áp dụng phương pháp đó vào vấn đề tắc nghẽn giao thông thông qua mô hình mô phỏng. Dù còn nhiều thiếu sót, em đã đạt được những kết quả nhất định như sau:

* Nắm được kiến thức cơ bản về học tăng cường, học tăng cường đa tác tử
* Tìm hiểu về vấn đề ùn tắc giao thông trên đường cao tốc
* Tìm hiểu, nghiên cứu và ứng dụng thuật toán Reinforce vào bài toán
* Tìm hiểu về nền tảng mô hình hóa, mô phỏng hóa đa tác tử NetLogo
* Áp dụng thành công thuật toán Reinforce cho mô hình bài toán giảm ùn tắc giao thông trên đường cao tốc
* Đánh giá, thử nghiệm và so sánh các chiến lược học tăng cường

## Hạn chế của đề tài:

Trong thời gian thực hiện đồ án, do chưa có kinh nghiệm trong việc nghiên cứu cũng như mới bắt đầu tìm hiểu về học tăng cường nên không tránh khỏi những thiếu sót. Các phần tìm hiểu về kiến thức có thể bổ sung thêm, mô hình của bài toán còn khá đơn giản và thuật toán cài đặt cũng chưa hoàn toàn tối ưu. Trong tương lai, em sẽ cố gắng học hỏi, tìm hiểu sâu hơn về học tăng cường giúp cải thiện thuật toán tốt hơn.

## Hướng phát triển

* Hiểu sâu hơn về nền tảng NetLogo
* Áp dụng các thuật toán học tăng cường khác để cải thiện kết quả
* Cải tiến mô hình mô phỏng như thêm đèn giao thông, thêm làn đường, biển báo …

TÀI LIỆU THAM KHẢO

1. Head, B., 2017. [*Demos for Python Extension of NetLogo*](https://github.com/NetLogo/Python-Extension/blob/master/demos/Traffic%20Basic%20-%20Reinforcement.nlogo)*.*
2. Lilian , W., 2018. [*A (Long) Peek into Reinforcement Learning*](https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html)*.*
3. Mnih, V. và những tác giả khác, 2015. Human-level control through deep reinforcement learning. *Nature,* 518(7540), pp. 529-533.
4. Nguyen, T. T., Nguyen, N. & Nahavandi, S., 2020. Deep reinforcement learning for multi-agent systems: a review of challenges, solutions and applications. *IEEE Transactions on Cybernetics,* 9, 50(9), pp. 3826 - 3839.
5. Richard, S. S. & Barto, A. G., 2018. *Reinforcement learning: An introduction.* MIT Press.
6. Russell, S. J. & Norvig, P., 2020. *Artificial Intelligence: A Modern Approach.* 4th Edition, Pearson.
7. Wilensky, U. & Payette, N., 1998. *NetLogo Traffic 2 Lanes model.* [Trực tuyến]   
   Available at: https://ccl.northwestern.edu/netlogo/models/Traffic2Lanes [Đã truy cập 2022].
8. Wilensky, U. & Rand, W., 2015. *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo.* The MIT Press.
9. Richard S. Sutton and Andrew G.Barto.[Reinforcement Learning Book (2020)](http://incompleteideas.net/book/RLbook2020.pdf)
10. [Supervised vs Unsupervised vs Reinforcement](https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/)
11. Jordi Torres. [Policy Gradient Method](https://towardsdatascience.com/policy-gradient-methods-104c783251e0)
12. Lilian Weng (2018). [Policy Gradient Algorithms](https://lilianweng.github.io/posts/2018-04-08-policy-gradient/)
13. <https://github.com/abhisheksuran/Reinforcement_Learning>
14. <https://github.com/kvsnoufal/reinforce>