הטכניון - מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל ע״ש אנדרו וארנה ויטרבי

המעבדה לבקרה, לרובוטיקה וללמידה חישובית

ספר פרויקט

אלגוריתם DQN לבקרת רמזורים

מבצעים:

נוואף סלמאן Nawaf Salman

נאהל עוידאת Nahel Awidat

מנחה:

דר.איל טייטלר Dr. Ayal Taitler

סמסטר רישום: חורף תשפ"ג

תאריך הגשה: 07/11/2024

תוכן

תקציר

ניהול יעיל של תנועה בצמתים עירוניים הוא מרכיב קריטי בעיר חכמה מודרנית. פרויקט זה מתמקד באופטימיזציה של מדיניות הרמזורים בצומת יחיד לשיפור זרימת התנועה והפחתת עומסים.
באמצעות סביבת הסימולציה SUMO (Simulation of Urban MObility) ושפת התכנות Python, יישמנו אלגוריתם DQN כדי ללמוד ולפתח מדיניות בקרה אופטימלית לרמזורים. באמצעות מודול TRACI (Traffic Control Interface), סוכן ה-DQN מתקשר עם סימולציית ה-SUMO, אוסף נתוני תנועה בזמן אמת ומתאים את תזמוני הרמזורים באופן דינמי.
המטרה המרכזית היא למזער את זמני ההמתנה הממוצעים ואת אורכי התורים של כלי הרכב בצומת.
תוצאותינו מציגות שיפורים ביעילות התנועה, ומדגימות את הפוטנציאל של טכניקות למידת חיזוק בניהול מערכות תנועה. פרויקט זה סולל את הדרך לפתרונות נרחבים בבקרת תנועה חכמה, ותורם לרשתות תחבורה עירוניות יציבות יותר.

Abstract

Efficient traffic management at urban intersections is a critical component of modern smart cities. This project focuses on optimizing traffic light policies at a single junction to enhance traffic flow and reduce congestion.
Utilizing the SUMO (Simulation of Urban MObility) simulation environment and Python, we implemented a Deep Q-Network (DQN) algorithm to learn and develop an optimal traffic signal control policy.
Through the TRACI (Traffic Control Interface) module, the DQN agent interacts with the SUMO simulation, gathering real-time traffic data and adjusting signal timings dynamically. The core objective is to minimize average waiting times and vehicle queue lengths at the intersection.
Our results demonstrate improvements in traffic efficiency, showcasing the potential of reinforcement learning techniques in traffic management systems. This project paves the way for scalable solutions in smart traffic control, contributing to smoother and more sustainable urban transportation networks.

מבוא

ניהול תנועה יעיל בצמתים עירוניים מהווה אתגר מרכזי בערים מודרניות. עם העלייה המתמדת במספר כלי הרכב בכבישים, בעיות כמו עומסי תנועה וזמני המתנה הולכות ומחריפות. בעיות אלו פוגעות באיכות החיים של התושבים, מגבירות את זיהום האוויר ומובילות לבזבוז זמן ואנרגיה. התמודדות עם האתגרים הללו והפחתת עומסי התנועה הפכה למשימה דחופה.

הבעיה המרכזית בפרויקט זה היא ניהול ואופטימיזציה של רמזורים בצומת יחיד כדי לשפר את זרימת התנועה ולהפחית את זמני ההמתנה של כלי הרכב. צמתים אלו מהווים צוואר בקבוק בתשתיות התחבורה, ובקרה יעילה של רמזורים יכולה להביא לשיפור משמעותי בזרימת התנועה העירונית.

פתרונות קיימים לניהול רמזורים כוללים מערכות בקרת תנועה קבועות, אשר מבוססות על מחזורים ותזמונים מוגדרים מראש. מערכות אלו אינן גמישות ואינן מסוגלות להתאים את עצמן בזמן אמת לתנאי התנועה המשתנים, דבר שמוביל ליעילות נמוכה ולבעיות עומס. בנוסף, קיימות מערכות חכמות יותר המשתמשות בחיישנים ובאלגוריתמים פשוטים כדי להתאים את תזמוני הרמזורים, אך גם אלו לעיתים קרובות אינן מצליחות להתמודד בצורה מיטבית עם השינויים הדינמיים בתנועה.

הגדרת בעיה

לפתח מדיניות בקרה אופטימלית לרמזורים בצומת יחיד, תוך התחשבות בתחבורה ציבורית ולתת לה קדימות תחת אלגוריתמים של למידה מחיזוקים (Reinforcement Learning) . המטרה היא להתאים את תזמוני הרמזורים בזמן אמת כדי לשפר את זרימת התנועה ומשך ההמתנה של כלי התחבורה הציבורית ורכבים פרטיים.

לצורך כך, השתמשנו באלגוריתם DQN. ובסימולטור SUMO כדי לבנות מדיניות חכמה לבקרת רמזורים. מטרת הרשת (DQN) היא לקבל מידע עדכני על מצב התנועה באופן דינמי, ולחזות את הפאזה האופטימלית של הצומת מסט פאזות מוגדר מראש. בנוסף, השתמשנו בתגמול משוקלל המעניק משקל גבוה יותר לרכבים ציבוריים, על מנת לעודד מתן עדיפות לתחבורה ציבורית בצומת.

פתרונות אפשריים, והפתרון הנבחר

בפרויקט בחנו כמה פתרונות לבקרת רמזורים: A2C ,אלגורתם Max Pressure, ופתרון מבוסס למידה מחיזוקים DQN.

Max Pressure: אלגוריתם Max Pressure הוא אלגוריתם קלאסי לבקרת רמזורים והוא מתבסס על הפחתת הלחץ בצמתים. הלחץ על כל קשת בכביש מחושב על פי ההפרש בין מספר המכוניות הממתינות בכיוון הנכנס לבין מספר המכוניות הממתינות בכיוון היוצא.
הסוכן בוחר את הפאזה שתפחית את הלחץ בצומת באופן המקסימלי.
זה אלגוריתם פשוט ליישום, דורש עיבוד קטן יחסית. אבל החסרון שלו שהוא לא גמיש לתנאי תנועה משתנים, ופחות יעיל כאשר יש שינויים מהירים בתנועה, וכדי להיות יעיל דורש תנאי תנועה יציבים ורמת עומס נמוכה.

A2C: אלגוריתם A2C (Advantage Actor-Critic) הוא שיטה בלמידה מחיזוקים המורכבת משני מרכיבים עיקריים: "Actor" (שחקן) ו-"Critic" (מבקר). ה-Actor בוחר פעולות על פי מדיניות נוכחית, וה-Critic מעריך את ערך הפעולות שנבחרו. בתהליך האימון, ה-Actor בוחר פעולה, מבצע אותה, מקבל תגמול, וה-Critic מעריך את היתרון של הפעולה כדי לעדכן את המדיניות והערכת הפעולות.
בבעיה שלנו מערכת רמזורים היא מערכת מורכבת עם הרבה מצבים ופעולות אפשריות, בגלל זה האלגוריתם A2C עלול להתקשות להתמודד עם כל המצבים והתרחישים המגוונים באופן יעיל. כמו כן, A2C דורש כוח חישובי רב וזיכרון כדי לעבד את כל המידע ולהתאים את המדיניות, וזה עשוי להיות בעייתי במיוחד בסביבות מורכבות כמו מערכות רמזורים בערים גדולות.

DQN: אלגוריתם DQN (Deep Q-Network) הוא שיטת למידה מחיזוקים שמשתמשת ברשת נוירונים עמוקה כדי ללמוד ולשפר מדיניות לפתרון בעיות קבלת החלטות. DQN היא שיטה פשוטה יחסית ועובדת ברוב המקרים, מה שהופך אותה לבחירה מועדפת בבעיות רבות, במיוחד כאשר יש מרחב מצב גדול ונדרשת הערכה של ערכי Q. במערכת רמזורים, היתרונות הללו יכולים להתבטא ביכולת טובה יותר להתמודד עם מצבים מורכבים ולהביא ללמידה יעילה ופשוטה יותר.
כמו כן, שימוש ב- DQN מאפשר למידה מנתונים אמיתיים שנאספו מהעבר ללא צורך באינטראקציה עם הסביבה. במקרה שלנו, זה יכול להיות שימושי אם נרצה לאמן את המודל על צומת אמיתית ללא שימוש בסימולציה.
לעומת אלגורתמי Actor-Critic שיצטרכו לבצע את הפעולות (הלא אופטימליות עדיין) הנבחרות בתהליך האימון, וזה יכול להיות דבר מסוכן בצמתים ויכול לגרום לפקקי תנועה משמעותיים.

תהליך החלטה מרקובי

תהליך החלטה מרקובי (MDP) הוא שיטה מתמטית למידול בעיות החלטה דינמיות, שבהן התוצאה של כל פעולה אינה ודאית ותלויה במצב הנוכחי ובפעולה שנבחרת. MDPs משמשים לתיאור מצבים בהם יש לקבל החלטות עוקבות לאורך זמן, כאשר המטרה היא למקסם את סך התגמולים לאורך זמן. במודל זה, ההנחה היא שהמעברים בין מצבים (states) הם סטוכסטיים ותלויים במצב הנוכחי ובפעולה (action) הנבחרת, כך שהבחירה בפעולה מסוימת מובילה למעבר למצב חדש ולתגמול מסוים, בהתאם להסתברויות המעבר המוגדרות.
המטרה בשימוש ב-MDP היא למצוא מדיניות אופטימלית שממקסמת את פונקציית הערך V.
פתרון של MDP כולל מציאת מדיניות אופטימלית או פונקציית ערך אופטימלית V^*. אלגוריתמים פופולריים לפתרון של MDP כוללים תכנות דינמי, שיטות Monte-Carlo, וטכניקות למידה מחיזוקים.

מצבים (S): המערכת יכולה להיות במגוון של מצבים או תנאים שונים. המצבים הללו מאפיינים את הקונפיגורציה הנוכחית של המערכת.

פעולות (A): הסוכן הנלמד יכול לבצע פעולות המשפיעות על מעבר של הסביבה ממצב אחד למצב אחר. הפעולות הן ההחלטות או התנועות שסוכן יכול לבצע.

הסתברות מעבר (P): פונקציית הסתברות המעבר מתארת את סיכוי המעבר ממצב נתון למצב אחר בהינתן פעולה מסוימת. היא מייצגת את דינמיקת המערכת.

תגמול (R): הסוכן מקבל רווח מספרי בהתאם למצב הנוכחי ולפעולה שבוצעה. המטרה היא למקסם את הרווח הנצבר לאורך הזמן.

discount factor (γ): פרמטר בין 0 ל-1 שמייצג את מידת החשיבות של תגמולים עתידיים. ערך γ גבוה (קרוב ל-1) אומר שתגובות עתידיות חשובות כמו תגמולים מיידיים, בעוד שערך γ נמוך (קרוב ל-0) מצביע על כך שתגובות מיידיות חשובות יותר. המטרה היא לאזן בין תגמולים בטווח הקצר והארוך.

פונקציית ערך אופטימלית ($V^*(s)$): מוגדרת ע"י התגמול הממוצע שנצבר לאורך הזמן כאשר מתחילים ממצב s ומבצעים פעולות אופטימליות. משוואת בלמן (Bellman Equation) :

$$V^*(s) = \max_{a\in A} \mathbb{E}[R(s,a) + \gamma V^*(s)\,] = \max_{a\in A}[R(s,a) + \gamma \sum_{s'\in S} p(s'|s,a)V^*(s')]$$

פונקצייה Q: דומה לפונקציה V, אבל Q מוגדרת לכל מצב s ופעולה a. ומתארת את התגמול הממוצע שנצבר לאורך הזמן כאשר מתחילים ממצב s ועושים פעולה a וממשיכים אחר כך באופן אופטימלי:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'\in S} p(s'|s,a)V^*(s')$$

מדיניות (π): מדיניות היא אסטרטגיה או סט של כללים המנחים את הסוכן בבחירת פעולות בכל מצב. היא מגדירה את המיפוי ממצבים לפעולות ($\pi: S \to A$) . המטרה היא לחשב מדיניות אופטימלית π^* שממקסמת את הפונקצייה Q:

$$\pi^*(s) = argmax_a Q(s,a)$$

למידה מחיזוקים

למידה מחיזוקים Reinforcement Learning היא פרדיגמת למידת מכונה המתבססת על עקרונות מפסיכולוגיה התנהגותית, ובה סוכן לומד לקבל החלטות על ידי אינטראקציה עם סביבה וקבלת משוב בצורת פרס או עונש. הרעיון המרכזי הוא לאפשר לסוכן ללמוד התנהגויות אופטימליות דרך ניסיון וטעיה, ולא על ידי תכנות מפורש.
ב-RL, תהליך הלמידה מתקיים כאשר סוכן פועל בסביבה, נוקט בפעולות על פי המצב הנוכחי שלה ומקבל משוב בצורת תגמול. מטרתו של הסוכן היא ללמוד מדיניות הממקסמת את סכום התגמולים הצפויים לאורך זמן. הסביבה, במקביל, מגיבה לפעולות של הסוכן ועוברת למצבים חדשים. זהו בעצם יישום של תהליך החלטה מרקובי.
הסוכן חוקר פעולות שונות כדי להבין את ההשלכות שלהן, וכך משפר את אסטרטגיית קבלת ההחלטות שלו.
יכולת הסוכן ללמוד ולהתאים את עצמו למגוון רחב של מצבים ולסביבות דינמיות, הופכת את הלמידה מחיזוקים לכלי עוצמתי בסביבות מורכבות ולא ודאיות, ומדגישה את פוטנציאל השיטה ליישומים בעולם האמיתי.

Exploration נגד Exploitation

אתגר מרכזי בלמידה מחיזוקים הוא חלוקה נכונה בין exploration ל-exploitation .
Exploration: חקירת הסביבה נחוצה כדי שהסוכן יגלה פעולות אופטימליות שאולי טרם גילה.
Exploitation: נחוץ לשם בחירת פעולות ידועות על מנת למקסם רווח מיידי. בחירת הפעולה מסתמכת על המדיניות הנלמדת.
שיטה ידועה לבצע את החלוקה היא $\epsilon - exploration$: בכל צעד נבחר את הפעולה שממקסמת את הפונקצייה Q (פעולה אופטימלית) בהסתברות $1 - \epsilon$, ונבחר פעולה אקראית אחרת בהסתברות ϵ.
נתחיל מערך ϵ גדול ונקטין אותו בכל צעד. זה מבטיח שהמערכת תתחיל להתמקד ב- exploration והופכת עם הזמן להסתמך יותר על הפונקצייה Q הנלמדת ולהתמקד יותר ב- exploitation.

מושגים והנחות

עד כה תיארנו בצורה כללית את האלגוריתמים והשיטות שהשתמשנו בהם.
נתחיל כעת לתאר את האלגוריתמים בצורה ספציפית לבעית הרמזורים.
בהתחלה נגדיר את המושגים וההנחות שנשתמש בהם בהמשך.

צומת: צומת רגילה בכביש שמורכבת מכמה רמזורים.

פאזה: מתארת את המצב של הצומת, איזה רמזורים ירוקים, צהובים או אדומים באותו רגע.
הנחה: נתייחס לנתיבים במקום רמזורים בתיאור הפאזה (נתיבים ירוקים, צהובים או אדומים).

סבב: מעבר על כל הפאזות בסדר מסוים כאשר מבקרים בכל פאזה פעם אחת בדיוק.
הנחה: צריך לעבור על כל הפאזות לפני שחוזרים לפאזה שהיינו בה באותו סבב.

פאזה ירוקה: פאזה שיש בה לפחות נתיב אחד ירוק.

פאזה צהובה: פאזה שצומת צריך לעבור בה כאשר עוברים מפאזה ירוקה לפאזה ירוקה אחרת.
שוהים בה בדיוק 2 שניות.

פאזה אדומה: פאזה שצומת צריך לעבור לה אחרי פאזה צהובה. שוהים בה בדיוק שניה אחת.

תגמול כולל: סכום התגמולים של כל הצעדים בסימולציה.

הנחה: יש לנו סט פאזות מוגדר מראש ועוברים על הפאזות באותו סדר תמיד.

מבוא ל- DQN

ב-DQN משתמשים ברשת נוירונים עמוקה Policy Network להערכת פונקציית Q. הרשת מקבלת כקלט את המצב הנוכחי s ומחזירה את ערכי Q לכל הפעולות האפשריות a במצב זה.

בנוסף לרשת הראשית, יש רשת מטרה Target Network שנעשה לה שינויים קטנים במשקולות (Soft update) . לכל משקל של הרשת הראשית θ, מעדכנים את המשקל המתאים ברשת המטרה בצורה הבאה:

$$\theta' \leftarrow \tau\,\theta + (1 - \tau)\theta'$$

שינויים חלקיים וקטנים במשקלות של רשת המטרה עוזרים לשמור על יציבות במהלך הלמידה. זה מקטין את הסיכון לשינויים דרסטיים במדיניות הפעולה ומפחית את התנודתיות בלמידה.
כמו כן, נשתמש ברשת המטרה כדי לחשב את פונקציית הערך של המצב הבא. בהינתן מצב s נעביר אותו מרשת המטרה (forward pass) וניקח את הערך המקסימלי במוצא:

$$V^*(s') = \max_{a \in A} Q_{target}(s, a)$$

נשים לב שלא משתמשים ב- Policy Network כדי לחשב את הערך של המצב הבא כדי לשמור על יציבות בלמידה.

בנוסף, נשתמש בזיכרון (Replay Memory) כדי לשמור את החוויות מהאינטראקציה עם הסביבה. נשמור בכל צעד רביעיה (מצב נוכחי, פעולה, תגמול, מצב הבא). זיכרון זה מאפשר לאלגוריתם ללמוד מהניסיון בצורה יעילה.

בנוסף, נשתמש בשיטת $\epsilon - exploration$ לאיזון בין exploration ל- exploitation. נתחיל עם $\epsilon = \epsilon_{start}$ ונקטין אותו בכל צעד. לפי הנוסחה הבאה (i הוא מספר הצעד):

$$\epsilon = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end})e^{\frac{-i}{decay}}$$

הגדרות של DQN ו- MDP

נתחיל ממבנה ה- MDP:

מצבים (S): המצב הוא שרשור של הוקטורים הבאים:
- קידוד one-hot של הפאזה הנוכחית. למשל אם יש לצומת 4 פאזות והפאזה הנוכחית היא 0 אז הוקטור יהיה [1,0,0,0]
  ייצוג one-hot מבדיל בצורה ברורה את הפאזות זו מזו. כל פאזה מקבלת וקטור ייחודי, כך שאין קשר מתמטי בין הערכים של הפאזות השונות.
- כמה זמן שהינו בפאזה הנוכחית בסבב הנוכחי. מספר זה מאותחל להיות 0 כאשר מתחילים פאזה חדשה וגדל ב- 1 בכל צעד עד שעוברים לפאזה הבאה.
- וקטור בגודל מספר הנתיבים, כך שבמקום ה- i רשום מספר המכוניות על הנתיב ה- i.
- וקטור בגודל מספר הנתיבים, כך שבמקום ה- i רשום מספר האוטובוסים על הנתיב ה- i.

פעולות (A): יש 2 פעולות אפשריות, לעבור לפאזה הבאה או להישאר בפאזה הנוכחית {לעבור 1, להישאר 0}.

מעברים (P): פונקציית המעבר היא דטרמיניסטית. לכל מצב ופעולה קיים מצב s' שעוברים אליו בהסתברות 1, כלומר אם הוחלט על פעולה היא תתבצע.

תגמול (R): תגמול שלילי על הזמן שכל המכוניות נשארות בסימולציה. Total Travel Time (TTT).
נבחן שתי פונקציות תגמול שונות:
1- TTT: בכל זמן נקבל תגמול של 1- על כל רכב שעוד לא סיים את הסימולציה.
2- weighted TTT: ניתן משקל גבוה לרכבים "חשובים" יותר. למשל אוטובוסים. נניח שמשקל של אוטובוס בפונקצית התגמול הוא 15 ומשקל מכונית הוא 1.
ולכן נקבל תגמול של 1- על כל מכונית שעוד לא סיימה את הסימולציה. ונקבל תגמול של 15- על כל אוטובוס שעוד לא סיים.

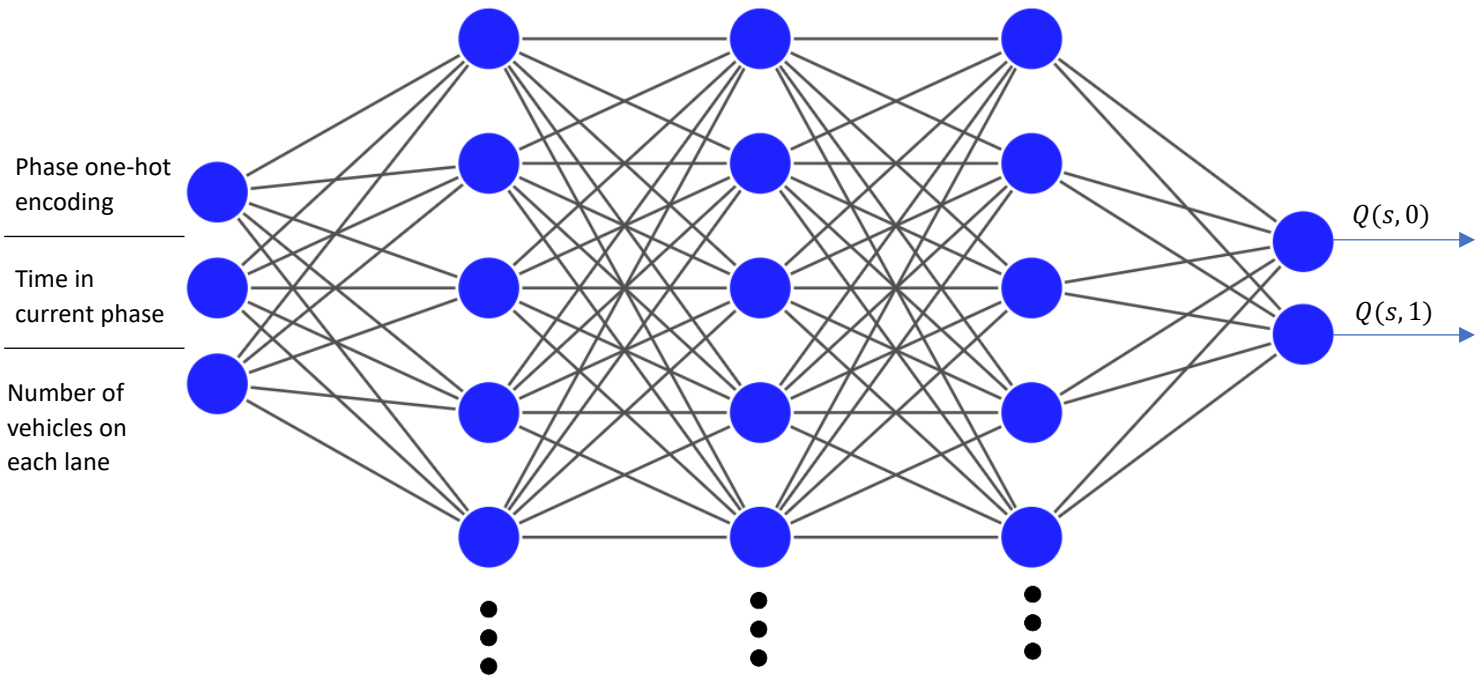
discount factor (γ): נבחר אותו להיות 0.99. חשוב לנו את התגמול הנוכחי יותר מתגמול עתידי. אבל גם התגמול העתידי חשוב בבעיה זאת כי הפעולה שבוצעה עכשיו משפיעה גם על התגמול שנקבל בעתיד. למשל מקרה שבו הרבה רכבים מחכים על נתיב מסוים שנהפוך אותו לירוק רק בעוד 10 מעברי פאזה. אנחנו רוצים שהתגמול שנקבל בעוד 10 מעברים ישפיע על ההחלטה שמקבלים עכשיו ויגרום לנו במקרה זה לעבור לפאזה הבאה.

פונקציית Q: $Q(s,a) = R(s,a) + \gamma \cdot V^*(s')$
פונקציית V: $V(s) = \max_{a \in A}[R(s,a) + \gamma \cdot V^*(s')]$
מדיניות (π): $\pi^*(s) = argmax_a Q(s,a)$

מבנה ה- DQN: לכל רשת ב-DQN יש שלוש שכבות נסתרות עם פונקציות הפעלה ReLU ברוחב 128 נוירונים ושכבה רביעית בלי פינקציית הפעלה ברוחב 2 נוירונים.
לכל מצב בכניסה, הרשת מחשבת את הערך של כל פעולה. מוצא הרשת הוא וקטור בגודל 2:
$[Q(s,0),\ \ Q(s,1)]$

Optimizer & Loss: נשתמש באופטימייזר RMSProp ובפונקצית הפסד Huber Loss.
שמאפשרים למודל להתמודד בצורה יעילה עם תנודתיות ושונות גבוהה בערכי Q, לשפר את יציבות האימון, ולהקטין את ההשפעה של נקודות קיצון. יחד, הם תורמים לשיפור הביצועים וההתכנסות של המודל בבעיות RL.

המטרה היא לאמן את המודל כדי לקבל פונקציית Q_{policy} קרובה כמה שיותר לפונקציה $Q_{expected}$ שמהווה פתרון למשוואת בלמן (ולכן אופטימלית).



תהליך האימון

תהליך זה מתואר גם בדיגראמות בהמשך (איורים 1 ו- 13).

לכל episode מ- 1 עד 200:
לכל צעד (t) מ- 1 עד 3600:
בצע את התהליך המתואר למטה

בכל צעד:

1. נבחר את הפעולה האופטימלית לפי שיטת $\epsilon - exploration$.
בהסתברות ϵ נבחר פעולה אקראית a_t. אחרת (בהסתברות $1 - \epsilon$) נבחר פעולה שממקסמת את הפונקצייה Q:
$$a_t = argmax_a Q_{policy}(s_t, a)$$
נעשה את זה בצורה הבאה:
נעביר את המצב הנוכחי s_t ב- Policy Network, נקבל במוצא וקטור בגודל 2,
$[Q(s,0),\ \ Q(s,1)]$, נבחר את האינדקס של הערך המקסימלי בוקטור זה.

2. נפעיל את a_t ע"י אינטראקציה עם הסביבה, כתוצאה מכך הסביבה תעבור למצב הבא, ונקבל מהסביבה את הרביעיה (מצב נוכחי, פעולה, תגמול, מצב הבא):
$$(s_t\ ,\ a_t\ ,\ r_t\ ,\ s_{t+1})$$

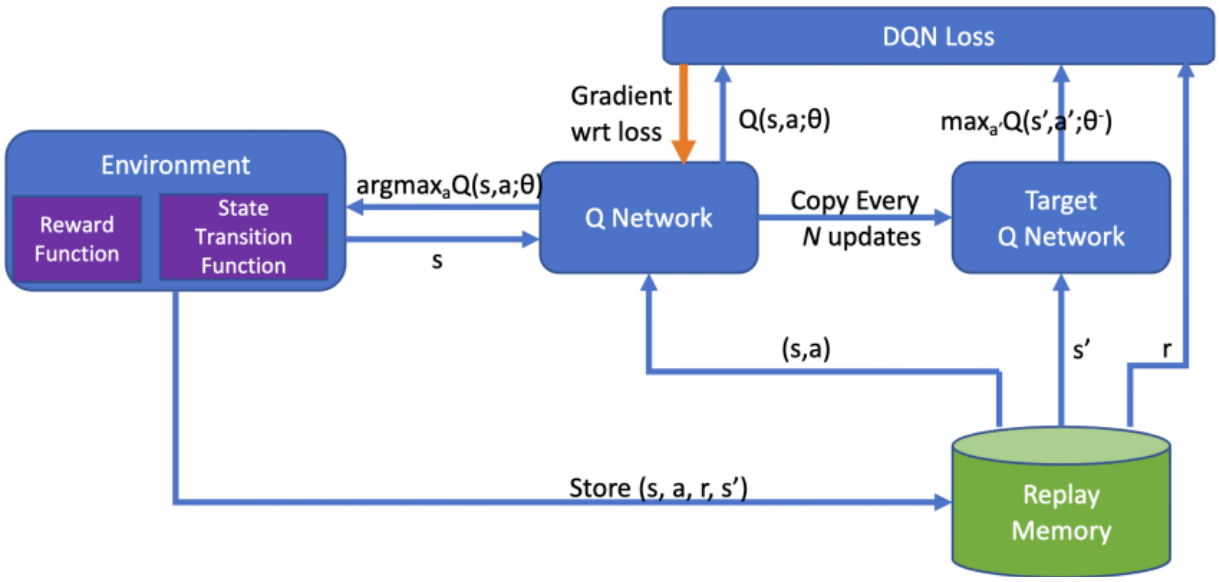
3. נשמור את הרביעיה ב- Replay Memory.

4. נדגום אקראית batch בגודל קבוע מה- Replay Memory. לכל רביעיה $(s\ ,\ a\ ,\ r\ ,\ s')$ ב- batch:
- נחשב את פונקציית הערך $V^*(s')$ של המצב הבא ע"י שימוש ב- Target Network:
$$V^*(s') = \max_{a \in A} Q_{target}(s,a)$$
- נחשב את הערך הצפוי ל- Q לפי נוסחת ה- Q-Learning:
$$Q_{expected}(s,a) = r + \gamma \cdot V^*(s')$$
- נחשב את הערך של Q הנלמד $Q_{policy}(s,a)$ ע"י העברת s ברשת הראשית (forward pass) ולקיחת הערך בויקטור המוצא באינדקס a.

5. נחשב את ה- loss בין Q_{policy} לבין $Q_{expected}$ של ה- batch. נחשב את הנגזרת של ה- loss ע"י backpropagation ונשתמש באופטימייזר RMSProp כדי לעדכן את משקולות הרשת הראשית (Policy Network).

6. נבצע עדכון Soft update למשקולות של Target Network:
$$\theta' \leftarrow \tau\,\theta + (1 - \tau)\theta'$$

דיגראמה שמתארת את תהליך האימון של DQN



איור 1

Hyperparameters Tuning

תהליך ה- Hyperparameters Tuning הוא שלב חשוב באימון רשתות נוירונים. מטרת התהליך היא למצוא את הערכים הטובים ביותר עבור הפרמטרים המשפיעים על האימון כמו גודל ה- Batch, קצב הלמידה, GAMMA ...
במקרה שלנו, בחרנו להפעיל מספר ניסויים על גבי GPU של Nvidia שרץ על השרתים של הטכניון. כאשר כל פעם הרצנו 4 ניסויים במקביל (זה מספר הניסויים המקסימלי שאפשר להריץ במקביל בגלל מגבלת הזיכרון של ה- GPU) בכל ניסוי ניסינו ערכים שונים של היפרפרמטרים.
התהליך כלל שינוי ערכים אלו, ריצה של האימונים במשך זמן רב, והשוואה בין תוצאות הניסויים כדי להבין אילו ערכים נותנים ביצועים אופטימליים. תהליך זה הוא קריטי כיוון שהיפרפרמטרים משפיעים בצורה ישירה על איכות המודל, מהירות הלמידה שלו, והתגמול הכולל.
אחרי סיום תהליך זה קיבלנו את הערכים האלו:

BATCH_SIZE = 64
GAMMA = 0.99
EPS_START = 0.9
EPS_END = 0.01
EPS_DECAY = 20000
TAU = 0.005
LR = 0.001

LR הוא קצב הלמידה של שתי הרשתות.
TAU (τ) הוא הקצב של ה- soft update ברשת ה- Target Network.
EPS_START / END / DECAY הם הפרמטרים של $\epsilon - exploration$.

כלים

SUMO-"Simulation of Urban Mobility", היא תוכנת סימולציה פתוחה לתחבורת עירונית המיועדת ליצירת מודלים ולדימוי תנועה בסביבות עירוניות. היא תומכת במיקרו-סימולציה של רכבים, תרחישים מרובים-מודאליים (כולל רכבי רגל ותחבורה ציבורית), ובהדמיה של מערכות בקרת תנועה. התוכנה גמישה מאוד ומאפשרת עבודה בסדרי גודל שונים של מערכת, מה שהופך אותה לכלי יקר ערך לחקר זרימת תנועה, לבדיקת שינויי תשתיות, ולהערכת אסטרטגיות ניהול תנועה בסביבות עירוניות שונות.

TRACI-"Traffic Control Interface", היא ספרית פייתון המסופקת ע"י SUMO. היא מאפשרת לתוכניות וכלים חיצוניים לתקשר בזמן אמת עם הסימולציה, ומספקת דרך לשלוט ולעקוב אחר מספר רב של אספקטים בסימולציה. דרך TRACI משתמשים יכולים להריץ ניסויים, לבדוק אלגוריתמים ולנתח תרחישי תנועה בסביבה וירטואלית לפני יישום השינויים בעולם האמיתי.

PyTorch- ספריית למידת מכונה open source שפותחה על ידי מעבדת AI Research של פייסבוק. היא מאפשרת בנייה של מודלים של למידת מכונה, כולל רשתות נוירונים, בצורה דינמית וגמישה. כמו כן ספרייה זאת מאפשרת לבצע גזירה ועדכון משקולות הרשת (backpropagation & optimization) בצורה קלה ויעילה.
נשתמש ב-PyTorch בעיקר כדי להשתמש במודלים ופונקציות פעולה בנויות כמו NN ו- ReLU, וכדי לבצע חישובים כבדים על GPU.

TensorFlow - היא ספריית open source שפותחה על ידי google ומשמשת ללמידת מכונה, למידה עמוקה ולניתוח נתונים. הספרייה מאפשרת בניית מודלים מתקדמים ואימון רשתות נוירונים בצורה קלה ואפקטיבית. חלק מרכזי ב-TensorFlow הוא TensorBoard, הוא כלי ויזואליזציה שמסופק יחד עם TensorFlow שמאפשר להציג גרפים של המודלים, מעקב אחר מטריקות של האימון, צפייה בתמונות, היסטוגרמות, ועוד. ניתן לעקוב אחרי מטריקות כמו הפסד (loss), דיוק (accuracy), גרדיאנטים ועוד, במהלך האימון.
זה עוזר לדיבוג, להבין איך המודל משתפר (או למה לא משתפר) עם הזמן.

סימולציות ותוצאות

את הסימולציה של הצומת העירוני (איור 2) שמהווה סביבה לסוכן הלומד, יצרנו באמצעות SUMO .
הסימולציה בנויה מצומת כאשר כל כניסה לאזור מהווה נקודת התחלה וסיום נסיעה. בצומת יש רמזורים לכל כיווני התנועה, מכיוון הגעה לשאר הכיוונים.
משך הסימולציה הינו 3600 שניות, המדמות שעה בעולם האמיתי.
לצומת יש רשימת פאזות מוגדרת מראש, בסימולציה הזאת יצרנו צומת עם 4 פאזות ירוקות (איורים 3 ו- 4). לכל פאזה יש אינדקס ומחרוזת שמתארת את הצבעים של כל הנתיבים (למשל לפאזה הראשונה יש אינדקס 0 ומחרוזת GrrrGrrr, בפאזה זאת כל הנתיבים באדום חוץ מהנתיבים הראשון והחמישי).

ליצירת התנועה בעיר השתמשנו ברכבים פרטיים ובאוטובוסים, כל מסלולי ועומסי הזרימה מוגדרים בקובץ מסוג rou.xml (דוגמה באיור 5). כל שורה בקובץ זה מתארת זרימת רכבים מצומת לצומת.
למשל, יש שורה מתאימה לזרימת רכבים פרטיים מצומת J17 לצומת J21 שמתחילה בזמן 0 ומסתיימת בזמן 3600.
ועבור זרימת אוטובוסים נוסף Type באותה שורה שבעזרתו נוכל להבדיל בין רכב פרטי לאוטובוס.
כל רכב מופיע בנתיב מסוים על פי התפלגות מעריכית, כפי שמוגדר בפרמטר "period="exp(XX
המשמעות של ההתפלגות המעריכית היא שרכבים ייכנסו למערכת במרווחי זמן אקראיים.
התפלגות מעריכית מאפשרת לנו לדמות זרימה אקראית של תנועה, כמו שקורה בעולם האמיתי, בו רכבים מגיעים למערכת לא בהכרח באופן אחיד אלא בתבנית אקראית מסוימת.

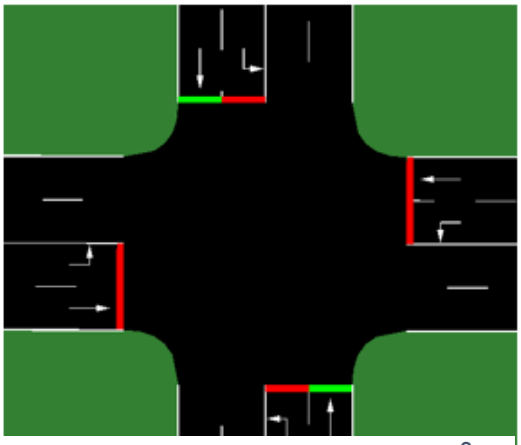
נשתמש בספריית TRACI ב- python. נטען בה את קובץ הסימולציה מסוג sumocfg וזה טוען אוטומטית את הקובץ שמתאר את עומס הזרימה (מסוג rou.xml).
בעזרת TRACI נוכל לבצע כל פעולה על הצומת ולקבל מידע על המצב הנוכחי שלו. בכל צעד נשתמש ב- API של TRACI כדי לבצע את הפעולה הנבחרה (a) וכדי לקבל את הרביעיה (מצב נוכחי, פעולה, תגמול, מצב הבא) שתשמש אותנו בתהליך האימון.

נריץ 3 סוגים של סימולציות:
בסימולציה הראשונה נשתמש ברכבים פרטיים בלבד עם עומס נמוך. ונאמן מודל שבו פונקציית התגמול היא: מינוס של TTT.
בסימולציה השניה והשלישית נשתמש גם ברכבים פרטיים וגם באוטובוסים עם עומס גבוה.
בסימולציה השניה נאמן מודל שבו פונקציית התגמול היא: מינוס של TTT
בסימולציה השלישית נאמן מודל שבו פונקציית התגמול היא: מינוס של weighted TTT.
(כמתואר בפרק : **הגדרות של DQN ו- MDP**).

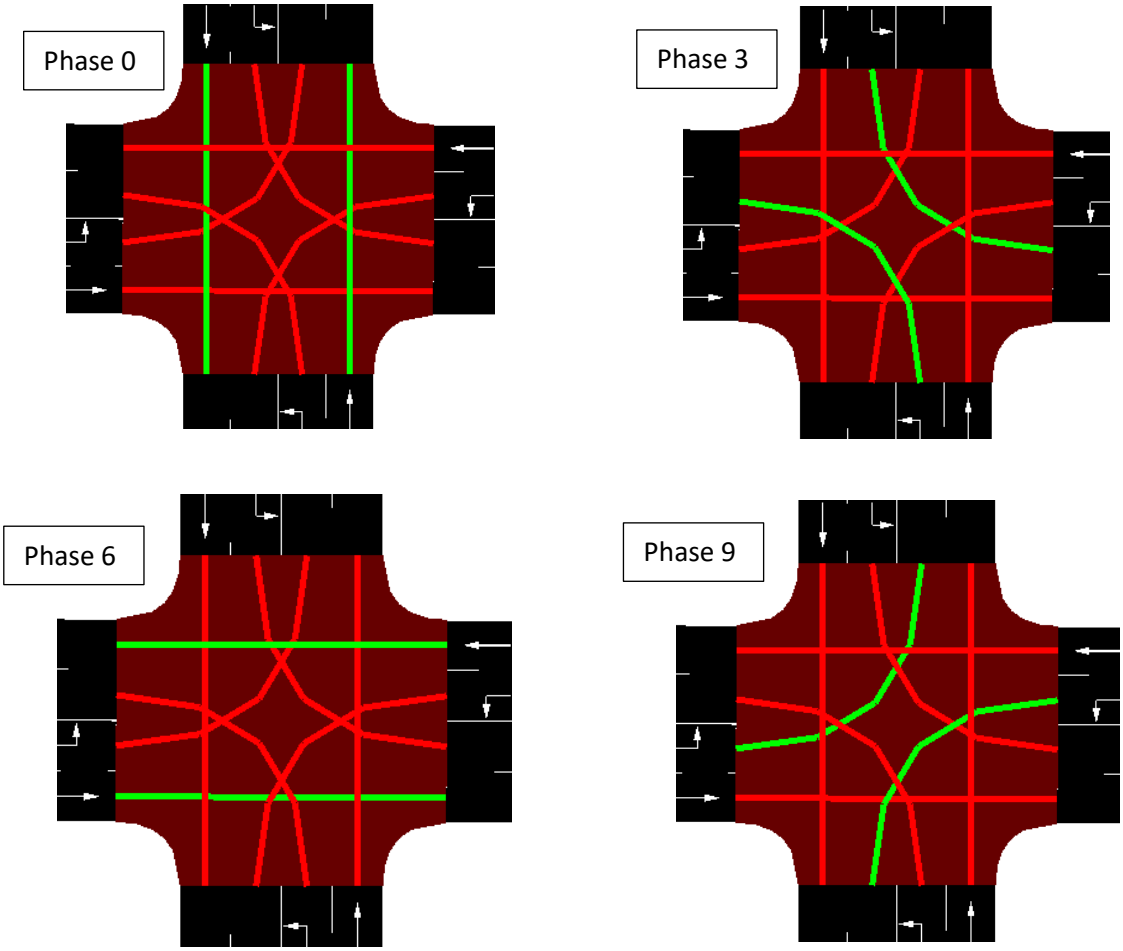
לכל סוג סימולציות כזה:
נריץ אותו 10 פעמים, כל פעם עם seed שונה, וכך נאמן 10 מודלים.
בסוף נחשב ונצייר את התוחלת וסטיית התקן של התגמולים של כל המודלים (דוגמה באיור 6).
ונשווה את התוצאות של DQN עם Max Pressure.

	dur	state	next
0	15.00	GrrrGrrr	1
1	2.00	yrrryrrr	2
2	1.00	rrrrrrrr	3
3	15.00	rGrrrGrr	4
4	2.00	ryrrryrr	5
5	1.00	rrrrrrrr	6
6	15.00	rrGrrrGr	7
7	2.00	rryrrryr	8
8	1.00	rrrrrrrr	9
9	15.00	rrrGrrrG	10
10	2.00	rrryrrry	11
11	1.00	rrrrrrrr	0
Σ	72.00	Links: 8	

איור 3



איור 2



איור 4

סימולציה ראשונה

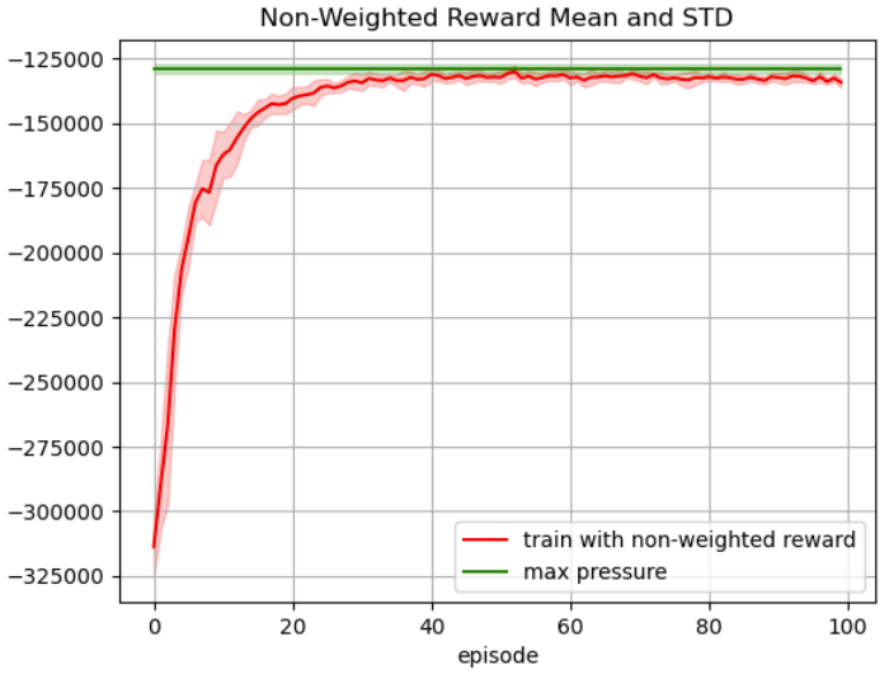
בסימולציה זו בחרנו זרימה של רכבים פרטיים על כל נתיב בהתפלגות מעריכית עם פרמטר 0.07.
(איור 5).
זה עומס יחסית נמוך.
השתמשנו בפונקצית תגמול: מינוס של TTT.

```
<!-- VTypes -->
<vType id="BUS_TYPE" vClass="passenger"/>
<vType id="CAR_TYPE" vClass="passenger"/>
<!-- Vehicles, persons and containers (sorted by depart) -->
<flow id="f_0" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J17" toJunction="J20" end="3600.00" period="exp(0.07)"/>
<flow id="f_1" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J20" toJunction="J17" end="3600.00" period="exp(0.07)"/>
<flow id="f_2" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J17" toJunction="J21" end="3600.00" period="exp(0.07)"/>
<flow id="f_3" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J20" toJunction="J22" end="3600.00" period="exp(0.07)"/>
<flow id="f_4" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J21" toJunction="J22" end="3600.00" period="exp(0.07)"/>
<flow id="f_5" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J22" toJunction="J21" end="3600.00" period="exp(0.07)"/>
<flow id="f_6" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J22" toJunction="J17" end="3600.00" period="exp(0.07)"/>
<flow id="f_7" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J21" toJunction="J20" end="3600.00" period="exp(0.07)"/>
```

איור 5 – rou.xml

תוצאות:
כמו שניתן לראות באיור 6 בצבע אדום, התגמול הממוצע (על 10 המודלים) מתחיל ב- (-330000) ועולה ככל שהאימון מתקדם עד שמתייצב על מקסימום של (-130000).

כידוע, Max Pressure הוא אלגוריתם אופטימלי ביחס לפונקציית תגמול TTT בלחץ נמוך ולכן נותן תוצאות אופטימליות עבור לחץ נמוך.
ניתן לראות שהרשת שלנו מתכנסת לאופטימום זה, דבר המוכיח את הצלחת המודל בהתאמה לתנאי לחץ נמוך.



איור 6

סימולציה שניה ושלישית

בשתי הסימולציות הגדרנו זרימות של שני סוגי רכבים - רכבים פרטיים ואוטובוסים - על נתיבים שונים בהתפלגות מעריכית (איור 7).

- רכבים פרטיים מוגדרים עם תקופת הופעה בתדירות של exp(0.07) עד exp(0.10) דבר שמדמה זרימה בתדירות בינונית.

- אוטובוסים מוגדרים עם תקופת הופעה בתדירות נמוכה יותר, בין exp(0.01) ל- exp(0.03), מה שמדמה זרימה נמוכה, בהתאמה לתדירות הנמוכה בה אוטובוסים מופיעים באמת.

ההבדל הוא שבסימולציה השניה השתמשנו בפונקציית תגמול של מינוס TTT באימון. ובסימולציה השלישית השתמשנו בפונקציית תגמול של מינוס Weighted TTT באימון.

כפי שניתן לראות באיור 7, בזרימות f1 ו- f3 תדירות הופעת אוטובוסים היא exp(0.03). ובזרימות f9 ו- f11 תדירות הופעת אוטובוסים היא exp(0.01). ולכן מצפים ש:
- בסימולציה השלישית המודל ייתן עדיפות גבוהה להיות בפאזה 0 (שבה נותנים ירוק ל- f1 ו- f3) ועדיפות נמוכה לפאזה 6 (שבה נותנים ירוק ל- f9 ו- f11). כי יש שימוש בתגמול משוקלל.
- בסימולציה השניה המודל לא ייתן עדיפות לפאזה אחת על השנייה כי אין שימוש בתגמול משוקלל.

```
<!-- VTypes -->
<vType id="BUS_TYPE" vClass="passenger"/>
<vType id="CAR_TYPE" vClass="passenger"/>
<!-- Vehicles, persons and containers (sorted by depart) -->
<flow id="f_0" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J17" toJunction="J20" end="3600.00" period="exp(0.07)"/>
<flow id="f_1" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J17" toJunction="J20" end="3600.00" period="exp(0.03)"/>

<flow id="f_2" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J20" toJunction="J17" end="3600.00" period="exp(0.07)"/>
<flow id="f_3" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J20" toJunction="J17" end="3600.00" period="exp(0.03)"/>

<flow id="f_4" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J17" toJunction="J21" end="3600.00" period="exp(0.08)"/>
<flow id="f_5" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J17" toJunction="J21" end="3600.00" period="exp(0.02)"/>

<flow id="f_6" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J20" toJunction="J22" end="3600.00" period="exp(0.08)"/>
<flow id="f_7" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J20" toJunction="J22" end="3600.00" period="exp(0.02)"/>

<flow id="f_8" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J21" toJunction="J22" end="3600.00" period="exp(0.09)"/>
<flow id="f_9" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J21" toJunction="J22" end="3600.00" period="exp(0.01)"/>

<flow id="f_10" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J22" toJunction="J21" end="3600.00" period="exp(0.09)"/>
<flow id="f_11" type="BUS_TYPE" begin="0.00" color="green"  fromJunction="J22" toJunction="J21" end="3600.00" period="exp(0.01)"/>

<flow id="f_12" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J22" toJunction="J17" end="3600.00" period="exp(0.10)"/>

<flow id="f_14" type="CAR_TYPE" begin="0.00" color="yellow" fromJunction="J21" toJunction="J20" end="3600.00" period="exp(0.10)"/>
```

איור 7

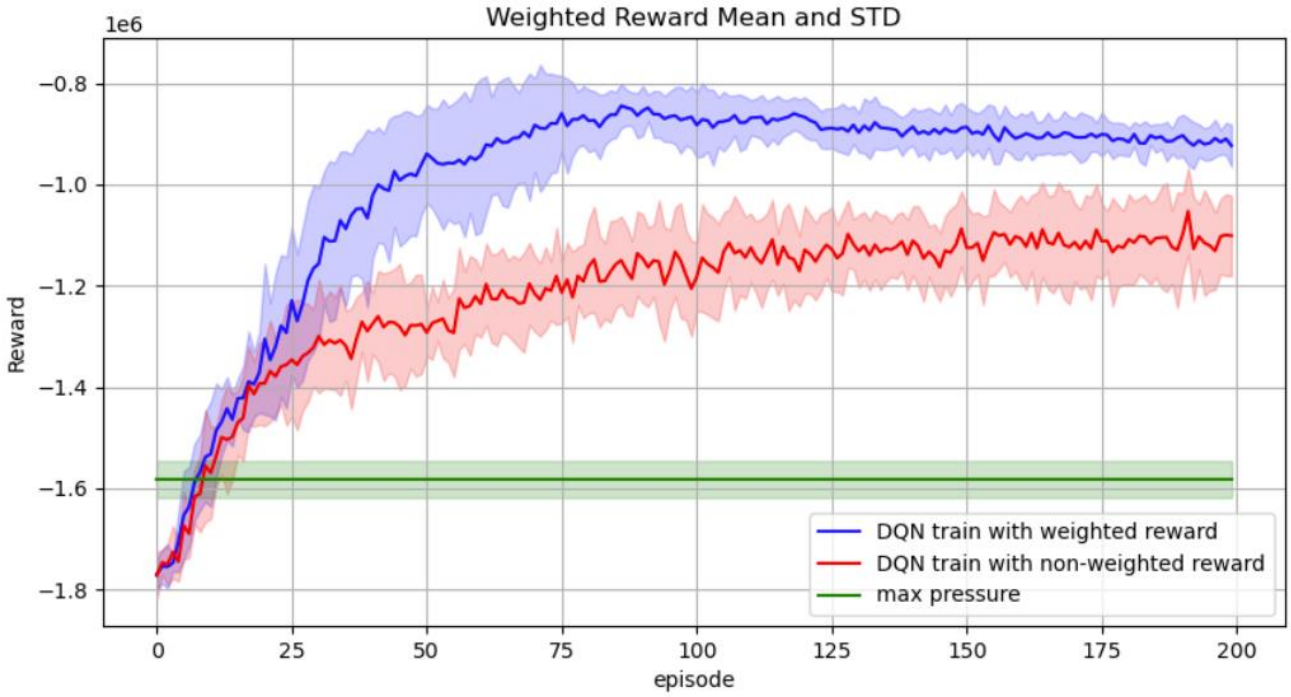
תוצאות:

נציג שני גרפים, הגרף הראשון (איור 8) מציג את התגמול המשוקלל של שתי הסימולציות ושל אלגוריתם max pressure.
הגרף השני (איור 9) מציג את התגמול הלא משוקלל של שתי הסימולציות ושל אלגוריתם max pressure.

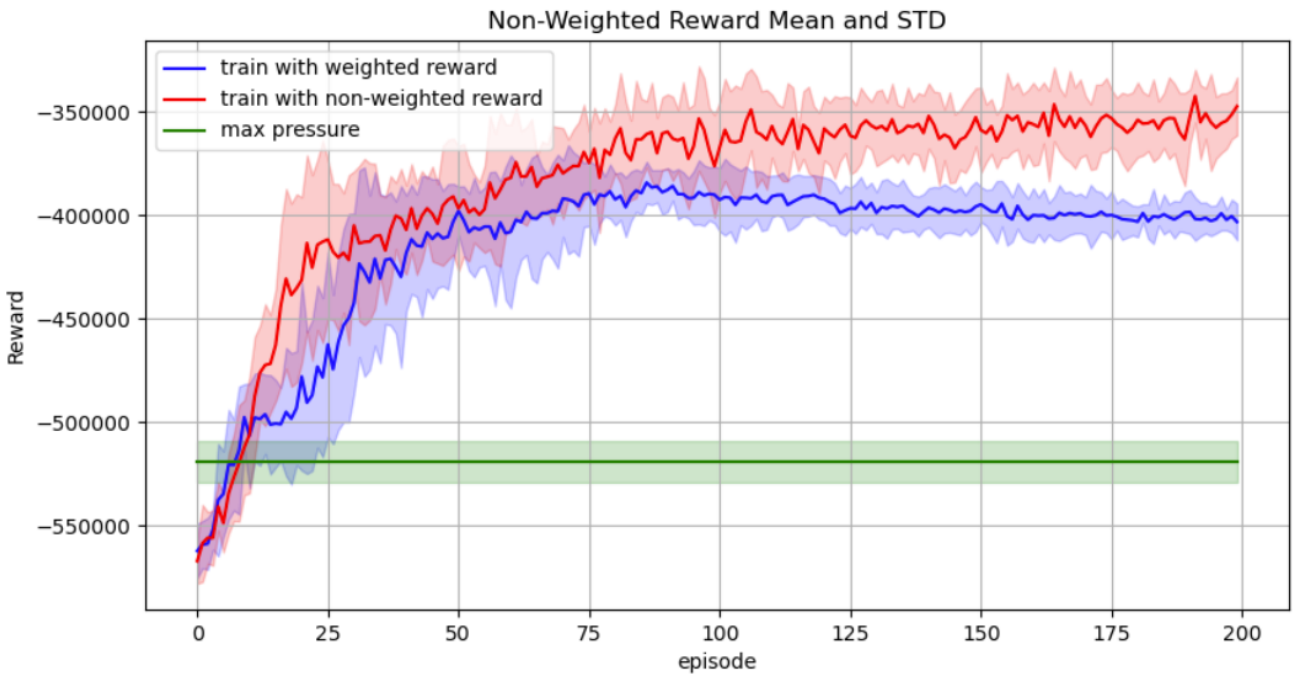
בגרף הראשון, ניתן לראות שהקווים הכחול והאדום מייצגים את האימונים עם תגמול משוקלל ורגיל בהתאמה. המודל שאומן עם תגמול משוקלל נותן עדיפות גבוהה יותר לאוטובוסים, כלומר ערך שלילי גדול יותר (15-) לאוטובוסים לעומת מכוניות (1-). התוצאה של גישה זו היא שבמהלך האימונים, המודל לומד לתת עדיפות למעבר מהיר יותר של אוטובוסים בצמתים, וכתוצאה מזה יהיה פחות אוטובוסים בסימולציה ולכן התגמול המשוקלל יהיה גדול יותר.

בגרף השני, ניתן לראות כי האימון על פי תגמול רגיל (לא משוקלל) מראה ביצועים טובים יותר בהשוואה לאימון עם תגמול משוקלל. מכיוון שבתגמול הלא משוקלל אין העדפה לאוטובוסים או מכוניות, המודל מתמקד בניסיון להקטין את זמן הנסיעה הכולל (Total Travel Time - TTT) עבור כלל כלי הרכב בצורה שווה. לכן, ביצועים אלו עשויים להיות טובים יותר כאשר אין חשיבות לסוג הרכב, כי המודל לא נדרש לתת עדיפות לרכבים מסוימים, והוא מתמקד בייעול כולל של התנועה. לעומת מודל שאומן עם תגמול משוקלל שנותן עדיפות לאוטוביוסים ולכן הוא לא אופטימלי ביחס לפונקציה תגמול רגילה (TTT).

בשני הגרפים אפשר לראות גם את תוצאות ההשוואה של DQN לאלגוריתם "max pressure" המיוצג בקו ירוק.
במקרה של עומס גבוה, ניתן לראות כי המודלים של DQN (בין אם עם תגמול משוקלל או לא) מצליחים להניב ביצועים טובים יותר מאשר max pressure. הסיבה לכך היא שהמודלים של DQN מסוגלים ללמוד דפוסי תנועה לאורך זמן ולהתאים את עצמם למצב התנועה, בעוד ש-max pressure מסתמך על כללים קבועים ואינו מצליח להתמודד היטב עם דינמיקה משתנה ולחצים גבוהים. כך, כאשר יש תנועת כלי רכב מרובה, DQN מצליח לווסת את התנועה בצורה יעילה יותר ולהפחית את זמן הנסיעה הכולל, בעוד ש-max pressure עלול להיתקע ב- "מעגלי עומס" ולא להיות אפקטיבי.


Weighted Reward Mean and STD
1e6
−0.8
−1.0
−1.2
−1.4
−1.6
−1.8
Reward
0
25
50
75
100
125
150
175
200
episode
DQN train with weighted reward
DQN train with non-weighted reward
max pressure

איור 8


Non-Weighted Reward Mean and STD
train with weighted reward
train with non-weighted reward
max pressure
−350000
−400000
−450000
−500000
−550000
Reward
0
25
50
75
100
125
150
175
200
episode

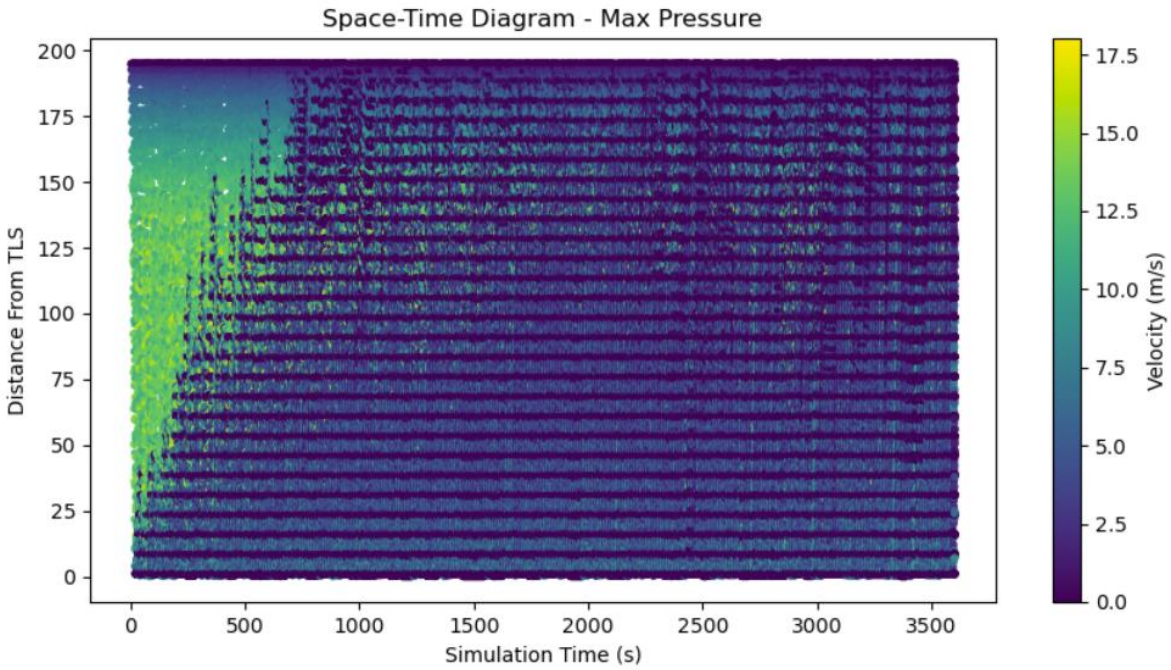
איור 9

Space-time plots

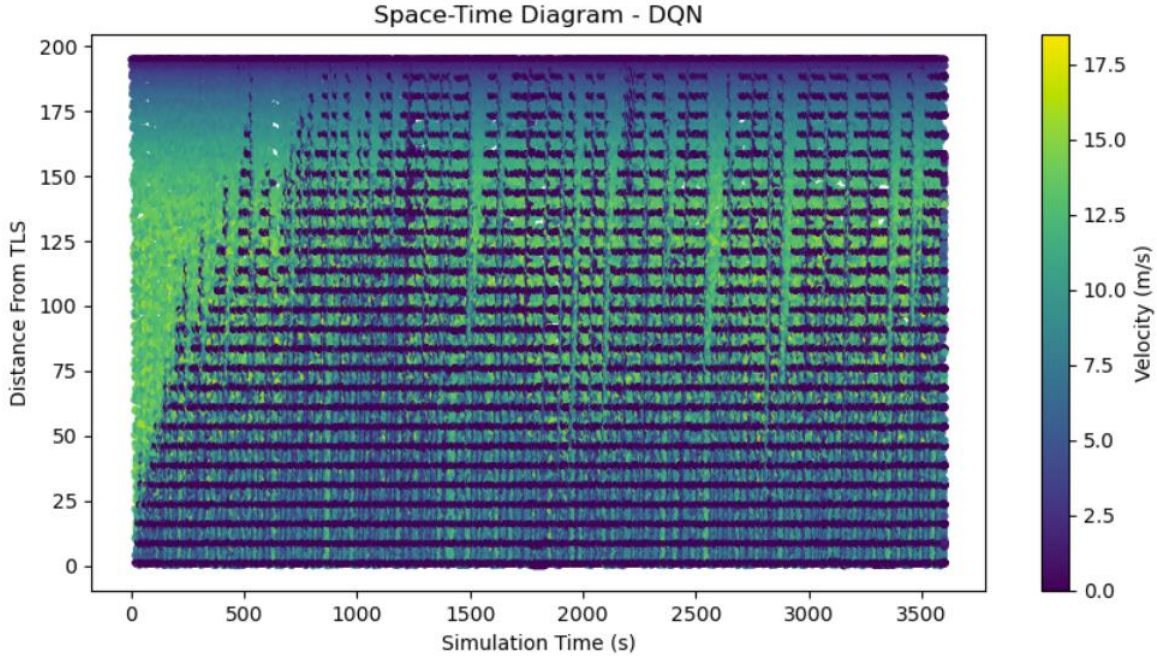
נציג שני גרפים שמתארים את המרחק והמהירות של כל הרכבים לאורך הזמן, גרף של Max Pressure וגרף של DQN.
כל נקודה על הגרף מייצגת רכב, וצבע הנקודה מייצג את המהירות שלו.
צבע כחול: רכב בעל מהירות נמוכה או עוצר.
צבע צהוב או ירוק: רכב בעל מהירות גבוהה.

בתרשים של Max Pressure (איור 10), ניתן לראות דפוס מחזורי קבוע של מהירויות, בעיקר באזורים הקרובים לרמזור. הדבר מצביע על כך שהרכבים נוטים להאט ולעצור בצורה תכופה, מה שמוביל לגלי עומס החוזרים על עצמם. תנודות התנועה כאן בולטות מאוד, ומתפשטות למרחק רב מהצומת לאורך הכביש, כאשר ניתן לראות שכל שהרכבים מתקרבים לרמזור, מהירותם יורדת באופן חד ונוצרות "שכבות" של עצירה ותנועה מחדש.
התוצאה של דפוס זה היא זרימה לא עקבית של התנועה, כאשר הרכבים נעים בגלי עצירה-נסיעה תכופים, מה שגורם לעומס משמעותי ומעכב את התנועה הכוללת.

לעומת זאת, בתרשים של DQN (איור 11), ניתן לראות תבנית שונה שבה תנודות התנועה פחות בולטות, במיוחד במרחקים גדולים יותר מהרמזור. השיטה של DQN מתאפיינת בכך שהיא לומדת לזהות דפוסים שונים בתנועה ומסוגלת להתאים את התזמון של הרמזורים בצורה שמצמצמת את גלי העומס. כתוצאה מכך, הרכבים חווים פחות עצירות ותנודות, ונראה שזרימת התנועה חלקה יותר, עם פחות שכבות של עצירה-נסיעה.

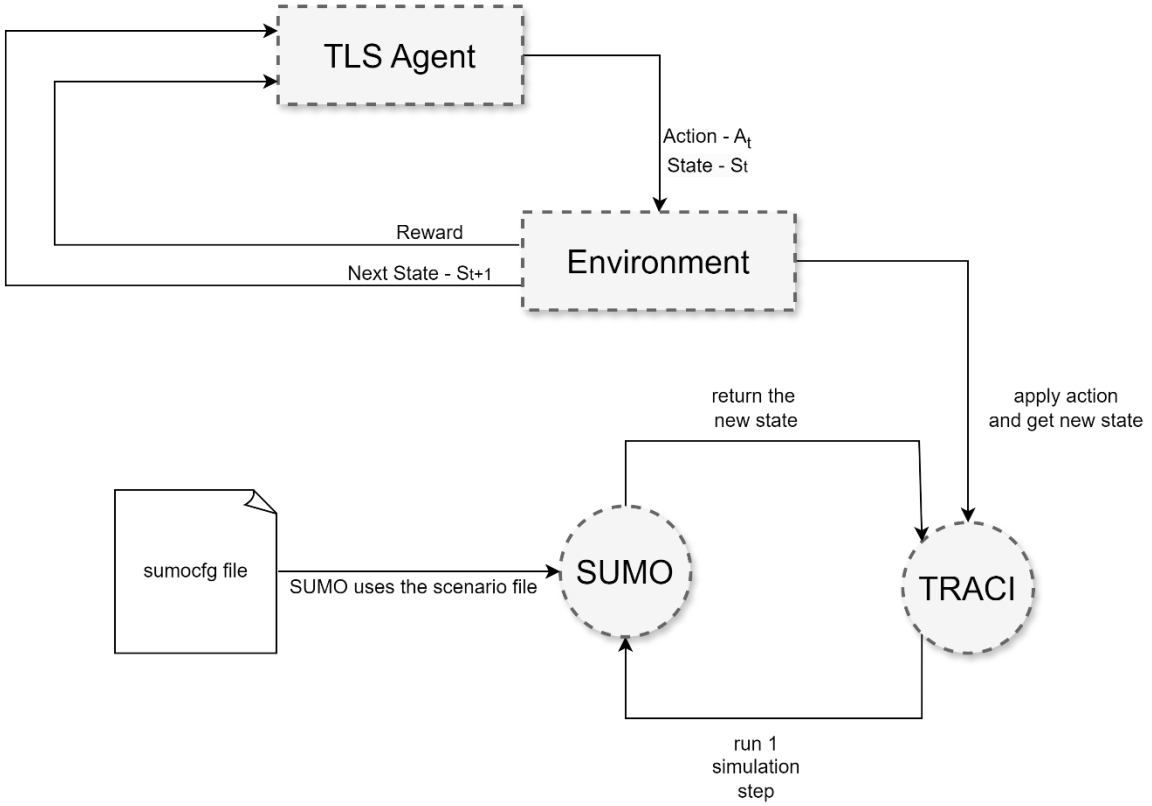


איור 10 (TLS = Traffic Light System)



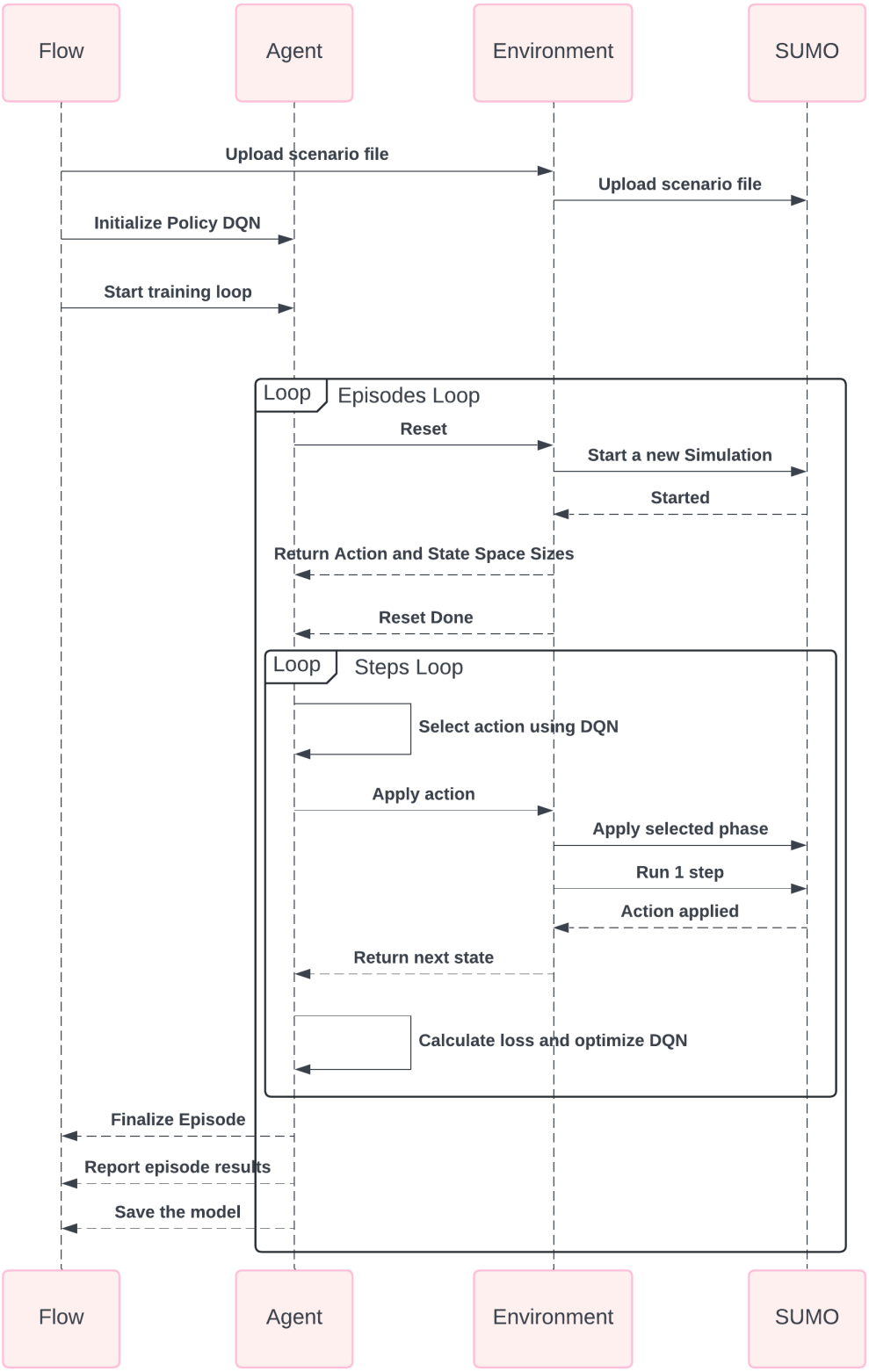
איור 11 (TLS = Traffic Light System)

Simulation High Level Block Diagram



איור 12

Simulation Sequence Diagram



איור 13

סיכום

בפרויקט זה התמקדנו בשימוש באלגוריתם DQN על מנת לאמן סוכן חכם לניהול שליטה בתעבורה בצמתים מרומזרים. הפרויקט כלל שלבים של בניית מודל מבוסס למידה עמוקה, ניסויים נרחבים לכיול היפרפרמטרים ואופטימיזציה של תהליך הלמידה. השתמשנו בטכניקות של Replay Buffer, Target Network ושימוש בגרדיאנטים מבוססי RMSprop כדי לשפר את יציבות האימון ואת תהליך ההתכנסות של הרשת.
במהלך הפרויקט ביצענו ניסויים רבים על גבי כרטיסי GPU על מנת לייעל את זמן האימון ולמקסם את השימוש במשאבי המחשוב, תוך חקר יעיל של מרחב הפרמטרים. הפרויקט הציג שיפורים ניכרים בביצועים של מערכת ניהול התעבורה בהשוואה לגישות קונבנציונליות ולמערכות שליטה סטנדרטיות.

עבודות עתידיות

כדי לשפר את תוצאות הפרויקט, ישנם מספר כיוונים אפשריים לעבודות עתידיות. ראשית, ניתן ליישם שיפורים במבנה הרשת הנוירונית, כגון שילוב של שכבות קונבולוציה או שימוש ברשתות מסוג LSTM לניהול רצף התעבורה בצורה יעילה יותר.
שיפור אפשרי ל-DQN הוא וקטור מצב (s) רחב יותר שמכיל עוד מידע בנוסף למידע הקיים על הפאזה הנוכחית ומספר המכוניות בכל נתיב.
בנוסף, ניתן לחקור שיטות RL אחרות כגון PPO (Proximal Policy Optimization) או A3C (Asynchronous Advantage Actor-Critic) כדי לבחון אם ניתן להשיג תוצאות טובות יותר מ- DQN.
כמו כן, אפשר להתמקד במצבים עם מספר סוכנים (Multi-Agent Reinforcement Learning) לניהול תעבורה בכמה צמתים בו-זמנית, מה שידמה בצורה מדויקת יותר את המציאות.

References

[1]. E. Kreidieh, C. Wu, and A. M. Bayen, “Flow: Deep Reinforcement Learning for Control in SUMO,” 2018.

[2]. C. Wu, E. Kreidieh, K. Parvate, A. Kejriwal, and A. M. Bayen, “Benchmarking Model-Free Reinforcement Learning for Traffic Control,” 2017.

[3]. R. Vinitsky, E. Kreidieh, A. M. Bayen, “Auto-MAP: A DQN Framework for Exploring Distributed Execution Plans for DNN Workloads,” 2020.

[4]. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., “Playing Atari with Deep Reinforcement Learning,” 2013.

[5]. C. Wu, E. Kreidieh, K. Parvate, and A. M. Bayen, “Framework for Control and Deep Reinforcement Learning in Traffic,” IEEE Transactions on Intelligent Transportation Systems, 2018.

[6]. Y. Zhao, X. Feng, X. Chen, L. Liu, and B. Xu, “Deep Reinforcement Learning for Traffic Light Control in Intelligent Transportation Systems,” 2023.

[7]. S. H. Lee, and K. H. Lee, “A SUMO Based Simulation Framework for Intelligent Traffic Management System,” Journal of Traffic and Logistics Engineering, vol. 8, no. 1, pp. 42-48, 2020.

[8]. M. B. Sharma, A. J. Singh, and T. P. Singh, “A Review of Adaptive Signal Control Systems Based on Changes in Traffic Environments,” Journal of Science and Engineering Research, vol. 11, no. 2, pp. 129-134, 2024.