دانشگاه علم و صنعت ایران

دانشکده مهندسی صنایع

پایان‌نامه کارشناسی

بهینه‌سازی فشار آب در شبکه‌های توزیع با استفاده از یادگیری تقویتی

نگارش

محمدرضا اسکندری

استاد راهنما

دکتر هادی صاحبی

شهریور ۱۴۰۳


بسم الله الرحمن الرحيم

باسمه تعالی

تعهدنامه صحت و اصالت نتایج

اینجانب محمدرضا اسکندری دانشجوی رشته مهندسی صنایع مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه/رساله حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

محمدرضا اسکندری

شهریور ۱۴۰۳

امضا

تقدیم به پدر و مادر عزیز و مهربانم که در سختی‌ها و دشواری‌های زندگی همواره یاوری دلسوز و فداکار و پشتیبانی محکم و مطمئن برایم بوده‌اند.

سپاس‌گزاری

از استاد دلسوز و محترم؛ جناب آقای دکتر صاحبی که با صبر و حوصله و بردباری، از هیچ کمکی در مسیر انجام این پروژه از من دریغ ننمودند و همراهی ایشان علی‌رغم کوتاهی‌های بنده در نوشتن این پایان‌نامه مایه‌ی دلگرمی بود؛ کمال تشکر و قدرانی را دارم.

محمدرضا اسکندری
شهریور ۱۴۰۳

چکیده

آب یک منبع محدود است که تعداد کاربران آن در حال افزایش است. در حقیقت، جمعیت جهان در ۲۰ سال گذشته تقریباً ۱/۵ میلیارد نفر افزایش یافته است. در نتیجه، شهرداری‌ها، شرکت‌های آب و به طور کلی جوامع باید از تکنیک‌های مدیریت آب پایدار استفاده کنند. امروزه با پیشرفت‌های حاصل‌شده در هوش مصنوعی و ظهور یادگیری عمیق و یادگیری تقویتی، قادر به حل مسائل با دقت هم اندازه‌ی دقت انسان و یا حتی فراتر از آن هستیم. از موضوعاتی که زیر سایه‌ی پیشرفت‌های حاصل‌شده رونق گرفته است، می‌توان به کنترل هوشمند فشار شبکه توزیع آب اشاره کرد. منظور از یک سیستم کنترل هوشمند این است که با دخالت محدود انسان یا بدون هیچ دخالتی، توانایی تنظیم فشار گره‌های شبکه آب وجود داشته باشد. کنترل هوشمند شبکه‌ی آب دو هدف را دنبال می‌کند: (الف) کنترل بیدرنگ شبکه در شرایط خاص و وفق‌پذیری سریع آن و (ب) کاهش میانگین فشار شبکه تا هزینه‌ی انرژی و نشتی آب در لوله‌های خراب کاهش یابد. فشار در لوله‌های آب به‌وسیله دریچه‌های کاهش فشار [۱] انجام می‌شود. شبکه باید از کیفیت جریان آب در گره‌های تقاضا اطمینان حاصل کند و این کیفیت با کنترل فشار شبکه به‌وسیله شیرهای کاهش فشار انجام می‌شود و معمولا این عملیات به صورت دستی و غیربهینه انجام می‌شود. هدف بهینه‌سازی به‌وسیله هوش مصنوعی داشتن کیفیت ذکرشده با کمترین فشار ممکن در شبکه است. به‌وسیله یادگیری تقویتی بدون نیاز به دانستن اجزای دقیق شبکه تنها با گرفتن داده‌های جزئی از شبکه مانند وضعیت فشار در گره‌های حساس به عنوان ورودی مسئله، فشاری که شیرهای کاهش فشار باید داشته باشند (به عنوان خروجی مسئله) را تنظیم می‌کنیم و بدین شکل فشار کل شبکه را تنظیم می‌کنیم. در این پروژه، تکنیکی را برای بهینه‌سازی کنترل فشار در سیستم‌های توزیع آب [۲] با استفاده از یادگیری تقویتی عمیق و ایپانت [۳] ارائه می‌کنیم. ایپانت یک نرم‌افزار هیدرولیک پرکاربرد است. ما استدلال خواهیم کرد که این چارچوب به اندازه کافی کلی است تا به طیف گسترده‌ای از مسائل بهینه‌سازی تصمیم متوالی در شبکه آب رسیدگی کند و می‌تواند برای کنترل بیدرنگ در شبکه آب

[1] Pressure Reduce Valve (PVR)

[2] Water Disribution Network

[3] EPANET

هوشمند استفاده شود.

واژه‌های کلیدی:

کنترل لوله‌های آب، تصمیم‌گیری، یادگیری تقویتی، یادگیری عمیق، بی‌درنگ، شبکه‌های آب، بهینه‌سازی

فهرست مطالب

عنوان صفحه

فهرست تصاویر

شکل صفحه

فهرست جداول

جدول صفحه

فهرست نمادها

نماد	مفهوم
t	گام زمانی (گسسته)
r	پاداش
s, s'	حالت
a	عمل
S	مجموعه حالات غیرپایانی
S^+	مجموعه کل حالات
$A(s)$	مجموعه اعمال موجود در حالت s
π	سیاست اتخاذشده توسط عامل
$\pi(s)$	عمل اتخاذشده توسط سیاست عامل در حالت s (سیاست قطعی)
$\pi(a\|s)$	احتمال اتخاذ عمل a در حالت s توسط سیاست عامل (سیاست تصادفی)
R_t	مقدار تابع پاداش دریافت‌شده در گام زمانی t

S_t	حالت قرار گرفته‌شده در گام زمانی t
A_t	عمل اتخاذشده در گام زمانی t
G_t	مقدار پاداش برگردانده‌شده مورد انتظار از گام زمانی t به بعد
$v_\pi(s)$	ارزش حالت s؛ تحت سیاست π
$v_*(s)$	ارزش حالت s؛ تحت سیاست بهینه
$q_\pi(s,a)$	ارزش بودن در حالت s و اتخاذ عمل a؛ تحت سیاست π
$q_*(s,a)$	ارزش بودن در حالت s و اتخاذ عمل a؛ تحت سیاست بهینه
V, V_t	مقدار تخمین زده‌شده از v_π (به ترتیب آرایه تخمین و مقدار تخمین در خانه t)
Q, Q_t	مقدار تخمین زده‌شده از q_π (به ترتیب آرایه تخمین و مقدار تخمین در خانه t)
δ_t	مقدار خطای تی‌دی در گام زمانی t
γ	نرخ تخفیف
ϵ	احتمال اتخاذ یک عمل تصادفی در سیاست اپسلیون‌حریصانه

α, β	اندازه گام
n	تعداد گام‌های راه‌اندازی آغازین
λ	نرخ تنزل (برای اثرات مشمولیت)
w	بردار وزن‌های تابع تقریب
$\hat{v}(s, w)$	ارزش بودن در حالت s ؛ تحت تابع تقریبی با وزن‌های w
$\hat{q}(s, a, w)$	ارزش بودن در حالت s و اتخاذ عمل a؛ تحت تابع تقریبی با وزن‌های w

فصل اول

مقدمه

آب آشامیدنی یکی از منابع حیاتی و ضروری برای سلامتی و رفاه جامعه است. با وجود این، کمبود منابع آب آشامیدنی و آلودگی آنها، به عنوان یک چالش بزرگ، جوامع بشری را تهدید می‌کند. کنترل دقیق و پویای فشار در WDN ها برای حفظ کیفیت آب، جلوگیری از هدررفتن آب و نشت و تضمین تامین آب پایدار از اهمیت بالایی برخوردار است. در این پروژه بیان می‌کنیم که روش‌های پیشین چه مشکلاتی داشتند و با استفاده از روش یادگیری تقویتی که در این پروژه به کار گرفته شده است می‌توان به آن معایب غلبه کرد و فشار را در شبکه کنترل کرد.

۱-۱ تعریف مسئله

در این پروژه قصد داریم یک سامانه کنترل فشار آب برای شبکه‌های آبی طراحی کنیم که به صورت بی‌درنگ [1] شیرهای کاهش آب را به نحوی تنظیم کند که میانگین فشار شبکه آب کمینه شود به طوری که کمینه فشار آب مورد نیاز برای گره‌های تقاضا برآورده شود. در این سامانه فرض می‌شود که نشتی آب وجود ندارد و مصرف آب در ساعت روز تغییرات ناگهانی ندارند. همچنین سامانه باید از تنظیم فشار حداکثری نیز پرهیز کند زیرا در دنیای واقعی فشار زیاد منجر به نشتی‌آب و اتلاف آب می‌شود.

۱-۲ اهداف پروژه

هدف از این پروژه، ارائه یک راه حل نوآورانه برای کنترل فشار در WDNها با استفاده از رویکرد کنترل بدون مدل و یادگیری تقویتی است. این سیستم به دنبال دستیابی به اهداف زیر است:

۱. کاهش هدررفتن آب: با تنظیم دقیق فشار، می‌توان از نشت و هدررفتن آب جلوگیری کرد.
۲. بهبود کیفیت آب: با حفظ فشار مناسب، از ورود آلاینده‌ها به شبکه جلوگیری می‌شود.
۳. افزایش پایداری: سیستم کنترل بدون مدل می تواند با شرایط متغیر شبکه و تقاضای آب سازگار شود و از قطع شدن آب جلوگیری کند.
۴. کاهش وابستگی به مدل: این روش بدون نیاز به مدل دقیق WDN ، عمل می‌کند.

[1] Real-time

۱-۳ پیشینه

به طور کلی روش‌های کنترل فشار آب در شبکه‌های توزیع به دو دسته تقسیم می‌شود:

۱. رویکرد بر پایه مدل

۲. رویکرد بدون مدل

۱-۳-۱ رویکرد بر پایه مدل

در این روش شبکه توزیع را با روابط ریاضی مدلسازی می‌کنیم و سعی می‌کنیم فشار مناسب برای هر لوله را به وسیله آن بدست بیاوریم. این روش در محیط شبیه‌سازی به‌خوبی عمل می‌کند و با کاهش فشار در کل شبکه به طوری که گره‌های تقاضا دارای فشار مناسب باشند همراه است. اما این روش در محیط واقعی خوب عمل نمی‌کند زیرا محیط ثابت نیست. برای مثال ممکن است یک لوله نشتی پیدا کند یا حوادث طبیعی مانند سیل رخ دهد و یا گره جدیدی به شبکه اضافه شود ولی چون مدل با داده‌های قبلی آموزش داده شده در این شرایط به خوبی عمل نمی‌کند.

۱-۳-۲ رویکرد بدون مدل

در این رویکرد به روابط ریاضی در هیدرولیک خیلی توجه نمی‌شود و تنها مشخصات شبکه به عنوان ورودی و برای ساخت محیط در دسترس است و شبکه با این اطلاعات ساخته می‌شود. در این شبکه، چند شیر کنترل فشار وجود دارد که به وسیله آن می‌توان فشار هر لوله در شبکه را کنترل کرد. در این روش عامل ما در واقع مجموعه‌ای از شیرهای کنترل فشار هست و در هر مرحله یک عمل به‌معنی تغییر فشار شیرها را انجام می‌دهد و محیط را در نهایت بررسی می‌کند و اشتباه خود را اصلاح می‌کند. مزیت این رویکرد این است که فراهم آوردن داده‌ی آموزشی ساده است، چرا که تنها کافی است عامل در محیط مورد نظر قرار گیرد و مقادیر حسگرها در کنار دستورات کنترلی ذخیره شوند. از طرفی، این روش می‌تواند در موقعیت‌های جدیدی که پس از فاز یادگیری با آن‌ها روبرو می‌شود به خوبی تصمیم‌گیری کند و خود را بهبود دهد. این روش برخلاف یادگیری تقلیدی است که مبتنی بر داده تولید شده توسط یک فرد ماهر است [۱] .

۱-۳-۲-۱ استفاده از الگوریتم ژنتیک برای مدل‌سازی

روش مبتنی بر الگوریتم‌های ژنتیک تک هدفه و چند هدفه بدین شکل است که با شروع از مدل عددی شبکه، کالیبراسیون را انجام می‌دهد و سپس مکان و کنترل شیرهای کاهش فشار [1] را بهینه می‌کند. محدودیت‌های عملیاتی که باید برآورده شوند. به طور خاص، مدل شبیه‌سازی با یک الگوریتم ژنتیک با کد واقعی و تک هدفه کالیبره می‌شود. استفاده از این روش در یک شبکه واقعی، صرفه‌جویی قابل توجهی در آب را ممکن می‌سازد، همان‌طور که با نظارت سیستم همراه است. دو مزیت دیگر نیز مشهود است: اولاً، تعداد مداخلات برای تعمیر لوله‌ها به دلیل کاهش رژیم فشار، بیش از نصف شده است (بنابراین شرکت آب را قادر می‌سازد تا خدمات بهتری را به طور مؤثرتر و قابل‌اطمینان‌تری ارائه دهد). ثانیاً، آب مازاد به مخزن ذخیره‌سازی یک شبکه پمپاژ شده هدایت می‌شود، بنابراین امکان کاهش قابل توجه هزینه‌های پمپاژ را فراهم می‌کند. ولی این روش بیشتر برای طراحی و ساخت شبکه اولیه مؤثر است و در محیط به صورت بیدرنگ مانند الگوریتم‌های یادگیری تقویتی به خوبی عمل نمی‌کند [۲] .

۱-۳-۲-۲ استفاده از الگوریتم DQN در یادگیری تقویتی

در این روش یک عامل هوشمند وظیفه کنترل شیرهای فشار را دارد و کنشی که در هر مرحله انجام می‌دهد، کم یا زیاد کردن فشار به وسیله شیرهای کنترل است. از مزایای این الگوریتم وفق‌پذیری با محیط است و به مدل وابسته نیست و همچنین با تغییر شبکه یا خرابی آن، این الگوریتم می‌تواند با وفق‌پذیری به کنترل کردن هوشمند خوب ادامه دهد. این روش نیازی به داده‌های اولیه برای آموزش ندارد و تنها با قرار گرفتن در محیط عمل‌های بهینه را یاد می‌گیرد. در هر الگوریتم یادگیری تقویتی یک نحوه پاداش‌دهی به عامل باید وجود داشته باشد و در این روش نحوه پاداش‌دهی به آن چک کردن فشار در گره‌های تقاضا و یا گره‌های حساس است. اگر این فشار در محدوده قابل قبول بود، پاداش داده می‌شود و در غیر این صورت عامل جریمه می‌شود [۳] .

[1] PRV

۱-۴ خلاصه

ما در این بخش به توضیحات کلی و تاریخچه و راه‌های پیشین این مسئله پرداختیم. همانطور که گفته شد راه‌های پیشین برای حل این مسئله مناسب نبودند و برای این منظور ما در این مسئله از الگوریتم یادگیری تقویتی DQN استفاده می‌کنیم. در فصل ۵ نتایج نشان داده شده است که بیانگر این است که این الگوریتم به درستی توانسته به هدف مسئله برسد. در فصل ۲ به مفاهیم پایه این مسئله پرداخته خواهد شد. در فصل ۳ به الگوریتمی که برای پیاده‌سازی انتخاب کرده‌ایم، پرداخته خواهد شد. در فصل ۴ به جزئیات پیاده‌سازی محیط شبیه‌سازی برای اجرای الگوریتم، پرداخته شده است. در فصل ۵ طراحی و پیاده‌سازی عامل هوشمند را شرح می‌دهیم. در فصل ۶ نیز نتایج را مورد بررسی قرار خواهیم داد.

فصل دوم

مفاهیم پایه

در این فصل به بررسی مفاهیم پایه یادگیری تقویتی می‌پردازیم. سپس با یکی از الگوریتم‌های این حوزه به نام DQN [1] در فصل بعدی آشنا می‌شویم.

۲-۱ اجزای تشکیل دهنده شبکه آب

منابع آب: این منابع می‌توانند منابع آب سطحی مانند رودخانه‌ها، دریاچه‌ها و مخازن یا منابع آب زیرزمینی مانند چاه‌ها و چشمه‌ها باشند. منبع آبی که استفاده می‌شود به عواملی مانند مکان، دسترسی و کیفیت آب بستگی دارد.

سیستم توزیع: سیستم توزیع شامل شبکه‌ای از لوله‌ها است که آب تصفیه‌شده را از تصفیه‌خانه به کاربران منتقل می‌کند. همچنین شامل تأسیسات ذخیره‌سازی، شیرآلات، شیر آتش‌نشانی و پمپاژ می‌باشد.

اجزای تشکیل‌دهنده یک شبکه آب به شرح زیر است:

۱. لوله‌های آب

۲. گره‌های تقاضا

۳. شیرهای کنترل فشار

۴. پمپ‌های آب

۵. منابع آب

۶. منابع طبیعی آب

این سامانه شبکه آب، از هر گره، اطلاعات مربوط به فشار را ذخیره می‌کند. سپس از این اطلاعات به عنوان ورودی مسئله استفاده می‌کند تا تصمیمات سطح بالاتر گرفته شود و تنظیمات مربوط به شیرهای فشار انجام شود. در این پروژه تمرکز بر روی تصمیم‌گیری بهتر برای داده گرفته شده از گره‌ها با بهره‌بری از هوش‌مصنوعی به ویژه یادگیری تقویتی است.

[1] Deep Q-network

۲-۲ کلیدواژه‌های یادگیری تقویتی

همین ابتدای کار، نیاز است تا چند کلیدواژه که در ادامه به کار گرفته خواهند شد، تعریف شوند. این کلیدواژه‌ها در ادامه لیست شده‌اند:

- عامل[۱]: موجودی که سعی در انجام یک وظیفه دارد.
- محیط[۲]: هر آنچه که عامل با آن ارتباط برقرار می‌کند.
- حالت[۳]: هر وضعیتی از محیط که عامل می‌تواند در آن قرار داشته باشد.
- عمل[۴]: مجموعه کارهایی که عامل می‌تواند انجام دهد.
- پاداش[۵]: هنگامی که عامل عملی انجام می‌دهد، از محیط یک بازخورد تحت عنوان پاداش دریافت می‌کند.
- سیاست[۶]: مشخص‌کننده این است که عامل در هر حالت محیط چه عملی را انجام می‌دهد .
- تابع ارزش[۷]: مشخص می‌کند که هر حالت محیط و یا هر عمل عامل در حالت جاری، به چه میزان خوب است.
- تابع ارزش حالت[۸]: مشخص می‌کند که هر حالت محیط به چه میزان خوب است.
- تابع ارزش عمل[۹]: مشخص می‌کند که هر عمل عامل در هر حالت محیط به چه میزان خوب است.
- مدل[۱۰]: مدل به نوعی یک تصور انتزاعی از نحوهٔ رفتار محیط می‌باشد. به کمک مدل، می‌توان برآورد کرد که در صورت بودن در یک حالت و اتخاذ یک عمل خاص، چه اتفاقی برای عامل رخ می‌دهد.

---

[1] Agent
[2] Environment
[3] State
[4] Action
[5] Reward
[6] Policy
[7] Value function
[8] State-value function
[9] Action-value function
[10] Model

۲-۳ یادگیری تقویتی

در این بخش، به تشریح مفاهیم اصلی یادگیری تقویتی پرداخته می‌شود.

۲-۳-۱ مسئله راهزن چند دست

قبل از اینکه به مسئله یادگیری تقویتی کامل[1] پرداخته شود، با یک نسخه ساده‌شده آن آشنا می‌شویم. این نسخه ساده‌شده، همان مسئله راهزن چند دست[2] معروف می‌باشد. در مسئله راهزن چند دست، فرض می‌شود k اهرم[3] موجود است. با کشیدن هر کدام از این اهرم‌ها، مقداری پاداش دریافت می‌شود. نحوهٔ تخصیص پاداش هر اهرم، مطابق با توزیع احتمالاتی[4] پاداشی است که در آن اهرم تعبیه شده است. این توزیع احتمالاتی از دید عامل پنهان است. هدف این مسئله، بیشینه کردن پاداشی است که می‌توان از این اهرم‌ها دریافت کرد.

در ساده‌ترین حالت مسئله راهزن چند دست، توزیع احتمالاتی اهرم‌ها ثابت می‌باشد. این بدین معناست که نحوهٔ تخصیص پاداش اهرم‌ها در طول زمان تغییر نمی‌کند. با در نظر گرفتن این موضوع، یک راه برای حل مسئله راهزن چند دست، این است که هر اهرم به تعداد دفعات زیادی کشیده شود تا بتوان تخمین درستی از میانگین پاداش آن بدست آورد. سپس هنگامی که از تخمین‌ها اطمینان حاصل شد، از آنجا به بعد در هر مرحله اهرمی کشیده شود که برآورد می‌شود بیشترین میانگین پاداش را می‌توان از آن دریافت کرد. به عبارتی دیگر، پس از اینکه توزیع احتمالاتی اهرم‌ها را یافتیم، کافی است نسبت به انتخاب اهرم‌ها حریصانه[5] عمل کرده و اهرمی کشیده شود که بیشترین پاداش آنی[6] را بدهد. نحوهٔ میانگین‌گیری پاداش در معادله پایین آورده شده است:

$$Q_n := \frac{R_1 + R_2 + ... + R_{n-1}}{n-1} \tag{۱-۲}$$

که در آن، Q_n میانگین پاداش به ازای $n-1$ بار کشیدن یک اهرم می‌باشد.

[1] Full reinforcement learning problem
[2] Multi-armed bandit problem
[3] Lever
[4] Probability distribution
[5] Greedy
[6] Immediate reward

تا اینجا، فرض بر این بود که توزیع احتمالاتی پاداش اهرم‌ها، ثابت می‌باشد. اما در مسئله راهزن غیرساکن[۱]، این توزیع‌های احتمالاتی در طول زمان تغییر می‌کنند. با این حساب، نمی‌توان مانند قبل، از روی میانگین‌گیری عادی پاداش‌های دریافت‌شده از یک اهرم، میانگین پاداش آن را محاسبه کرد.

بدیهی است که در مسئله راهزن غیرساکن، پاداش‌هایی که اخیرتر دریافت شده‌اند، قابل‌اتکاتر هستند. دلیل آن این است که این پاداش‌ها، مربوط به توزیع‌های احتمالاتی‌ای بودند که در گام‌های زمانی[۲] نزدیک‌تری رخ داده‌اند. از این رو، می‌توان فرآیند میانگین‌گیری راهزن ساکن[۳] مسئله قبل را به صورت یک میانگین وزن‌دار در آورد که در آن، وزن پاداش‌های اخیر بیشتر است. به کمک قانون به‌روزرسانی[۴] زیر، فرمول جدید محاسبه میانگین پاداش هر اهرم تعریف می‌شود:

$$Q_{n+1} := Q_n + \alpha[R_n - Q_n] \qquad (۲-۲)$$

که در آن، Q_n تخمین قبلی، R_n پاداش دریافت‌شده در گام زمانی فعلی، Q_{n+1} تخمین جدید و α یک ثابت است که اندازه گام را مشخص می‌کند.

در مسئله راهزن چند دست (و در ادامه، یادگیری تقویتی)، یکی از چالش‌های اصلی، ایجاد تعادل بین اکتشاف[۵] و بهره‌جویی[۶] است. این چالش بدین صورت است که در هر مرحله یا می‌توان با در نظر گرفتن تخمین توزیع‌های احتمالاتی بدست آمده تا به الان، نسبت به آن‌ها حریصانه عمل کرد و بهترین اهرم فعلی را انتخاب کرد (بهره‌جویی دانش فعلی) و یا اهرمی به جز اهرم حریصانه فعلی انتخاب کرد (اکتشاف) و با دریافت پاداش جدید، توزیع احتمالاتی آن اهرم را به‌روزرسانی کرد.

بهره‌جویی از این جهت حائز اهمیت است که هدف نهایی ما از حل مسئله می‌باشد. در نهایت قرار است با یافتن میانگین پاداش اهرم‌ها، بهترین اهرم انتخاب شود. از طرفی دیگر، اکتشاف از این جهت مهم است که امکان این را می‌دهد که اهرم‌های مستعدی که در حال حاضر شاید گزینه حریصانه نباشند، با کشف شدن تبدیل به گزینه حریصانه نهایی شوند. بنابراین، نیاز است که پتانسیل این اهرم‌ها را دریافت.

[1] Nonstationary bandit
[2] Time step
[3] Stationary bandit
[4] Update rule
[5] Exploration
[6] Exploitation

با این حساب، در ادامه عامل باید بتواند هم در مواقع مناسب اکتشاف انجام دهد و هم بهره‌جویی.

۲-۳-۲ فرآیندهای تصمیم‌گیری مارکوف

در مسئله راهزن چند دست، مشاهده شد که چطور به کمک میانگین‌گیری از پاداش‌های دریافت‌شده هر اهرم، میانگین پاداش آن تخمین زده می‌شود. این مسئله نسبت به مسئله یادگیری تقویتی کامل، چند فرض در نظر گرفته است که آن را ساده‌تر می‌کند. این فرض‌ها عبارتند از:

۱- توزیع احتمالاتی پاداش‌ها در طول زمان تغییر نمی‌کند (این فرض در اهرم‌های غیرساکن برقرار نمی‌باشد).

۲- پاداش‌ها تنها به صورت آنی در نظر گرفته می‌شوند.

۳- کشیدن یک اهرم، تاثیری روی توزیع احتمالاتی سایر اهرم‌ها ندارد.

۴- کشیدن یک اهرم، تاثیری روی پاداش‌هایی که در آینده می‌توان دریافت کرد، ندارد.

اما در بسیاری از مسائل، تصمیم‌گیری و اتخاذ یک عمل، عواقب خودش را به دنبال دارد. فرض کنید یک ربات در یک اتاق قرار می‌گیرد و مسئولیت تمیز کردن اتاق به آن سپرده شده است. اگر ربات تصمیم بگیرد به سمت چپ حرکت کند، در بخش دیگری از اتاق قرار می‌گیرد. واضح است که در صورتی که ربات تصمیم می‌گرفت به جای حرکت به سمت چپ خود، به سمت راست حرکت کند، در یک بخش متفاوتی از اتاق قرار می‌گرفت. این موضوع نشان می‌دهد که حالتی از محیط که ربات در آن قرار می‌گیرد، تحت تاثیر اعمالی است که تصمیم می‌گیرد انجام دهد.

بنابراین، برای آنکه بتوان تاثیر «عواقب داشتن اتخاذ عمل» را در نظر گرفت، نیاز به یک چهارچوب ریاضیاتی مناسب است که بتوان به کمک آن، تصمیم‌گیری را به صورت ترتیبی مدل کرد. فرآیندهای تصمیم‌گیری مارکوف، مناسب این کار هستند. در فرآیندهای تصمیم‌گیری مارکوف، محیط به صورت مجموعه‌ای از حالات تصور می‌شود. این حالات توسط اعمال قابل‌انجام توسط عامل به هم مرتبط می‌شوند. همچنین، پاداش‌های دریافت‌شده به ازای اتخاذ این اعمال نیز مشخص است.

برای آنکه بتوان مسائل یادگیری تقویتی را به کمک مدل کردن محیط به صورت فرآیند یادگیری مارکوف حل کرد، نیاز است که مدل به گونه‌ای تنظیم شود که دارای خاصیت مارکوف[1] باشد. خاصیت مارکوف، بیان می‌کند که با اطلاع داشتن از حالت فعلی‌ای که عامل در آن قرار دارد، اطلاعات لازم را برای پیش‌بینی

[1] Markov property

حالاتی که در آینده می‌تواند با آن مواجه شود پیدا خواهد کرد. به عبارتی دیگر، نیاز نیست که هیچ اطلاعاتی از حالاتی که قبلاً در آن حضور داشته، نگه دارد. بنابراین برای مدل کردن محیط مسئله تنها به حالت فعلی نیاز داریم و نیازی به نگه‌داری دنباله‌ای از حالت‌ها و اعمالی که در گذشته انجام شده‌اند نداریم. با تعریف مناسب حالت برای مسئله می‌توان این خاصیت را ایجاد نمود.

خاصیت مارکوف، به نوعی تضمین می‌کند که تمام راهبردهایی که عامل بنا دارد بچیند تا به هدفی برسد را، کافی است با در نظر گرفتن حالتی که الان در آن قرار دارد و طرح‌ریزی کردن سلسله اعمال خود از آنجا به بعد، انجام دهد.

۲-۳-۳ هدف مسئله یادگیری تقویتی

در مسئله یادگیری تقویتی، هدف آن است که سیاست عامل به گونه‌ای تنظیم شود که پاداش تجمعی[1] (شامل مجموع پاداش آنی و پاداش تاخیریافته) دریافت‌شده را بیشینه کند. به عبارتی دیگر، با شروع از یک حالت در محیط، عامل بداند چه سلسله‌ای از اعمال را باید انجام دهد تا بتواند به بیشترین میزان ممکن، پاداش تجمعی دریافت کند. به صورت ریاضی:

$$G_t := R_{t+1} + R_{t+2} + ... + R_T \qquad (۲-۳)$$

که در آن، R_T پاداش دریافت‌شده در گام نهایی اپیزود می‌باشد. به G_t، بازگشت مورد انتظار[2] گفته می‌شود. بسته به اینکه عامل چقدر بخواهد بین دریافت پاداش آنی و پاداش تاخیریافته تعادل برقرار کند، فرمول محاسبه بازگشت مورد انتظار به صورت زیر تغییر پیدا می‌کند:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} (0 \leq \gamma \leq 1) \qquad (۲-۴)$$

که در آن، γ همان نرخ تنزل[3] می‌باشد. این مقدار نشان‌دهنده این است که عامل تا چه حدی به پاداش‌های تاخیر یافته (پاداش‌های دریافت‌شده از محیط بعد از گام t) اهمیت می‌دهد. هرچه این مقدار

[1] Cumulative reward
[2] Expected return
[3] Discount rate

به صفر نزدیک باشد، عامل به پاداش‌های دور از گام t وزن کمتری می‌دهد و بالعکس.

۲-۳-۴ مفاهیم متداول یادگیری تقویتی

در هر فرآیند تصمیم‌گیری مارکوف، رابطه‌ای وجود دارد که مشخص می‌کند با چه احتمالی می‌توان از یک حالت به حالات دیگر رفت. به ازای هر حالت موجود در فرآیند تصمیم‌گیری مارکوف، این انتقال‌ها به صورت زیر تعریف می‌شوند:

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s] \tag{۲-۵}$$

این معادله، به نوعی سازوکار[1] فرآیند تصمیم‌گیری مارکوف را توصیف می‌کند. با در اختیار داشتن احتمال انتقال بین حالات، می‌توان ماتریس انتقال حالت[2] مربوط به فرآیند تصمیم‌گیری مارکوف را تشکیل داد. این ماتریس در ادامه آورده شده است:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{1,1} & \mathcal{P}_{1,2} & \cdots & \mathcal{P}_{1,n} \\ \mathcal{P}_{2,1} & \mathcal{P}_{2,2} & \cdots & \mathcal{P}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}_{n,1} & \mathcal{P}_{n,2} & \cdots & \mathcal{P}_{n,n} \end{bmatrix} \tag{۲-۶}$$

برای آنکه بتوان بازگشت مورد انتظار را در فرآیند تصمیم‌گیری مارکوف بیشینه کرد، نیاز است مشخص شود چه حالت‌هایی مطلوب هستند. برای این کار، تابع ارزش حالت به شکل زیر برای هر حالت موجود در فرآیند تصمیم‌گیری مارکوف تعریف می‌شود:

$$v(s) = \mathbb{E}[G_t | S_t = s] \tag{۲-۷}$$

با باز کردن معادله مربوط به تابع ارزش حالت، چنین روابطی نتیجه می‌شوند:

[1] Dynamics

[2] State transition matrix

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s]
\end{aligned}
\tag{۸-۲}
$$

این معادله، که به معادله بلمن[1] مشهور است، نشان می‌دهد که مقادیر تابع ارزش حالتِ حالت‌های مختلف محیط، چگونه به یکدیگر وابسته‌اند.

مشاهده شد که چگونه معادله بلمن، مقادیر تابع ارزش حالت را در حالت‌های مختلف موجود در فرآیند تصمیم‌گیری مارکوف به هم مرتبط می‌کند. اگر فرض شود که فرآیند تصمیم‌گیری مارکوف دارای n حالت می‌باشد، آنگاه به کمک ماتریس انتقال حالت، می‌توان معادله بلمن را به فرم ماتریسی زیر نوشت:

$$
\begin{bmatrix} v(1) \\ v(2) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{1,1} & \mathcal{P}_{1,2} & \cdots & \mathcal{P}_{1,n} \\ \mathcal{P}_{2,1} & \mathcal{P}_{2,2} & \cdots & \mathcal{P}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}_{n,1} & \mathcal{P}_{n,2} & \cdots & \mathcal{P}_{n,n} \end{bmatrix} \begin{bmatrix} v(1) \\ v(2) \\ \vdots \\ v(n) \end{bmatrix}
\tag{۹-۲}
$$

یا به صورت ساده‌شده:

$$
v = R + \gamma \mathcal{P} v \tag{۱۰-۲}
$$

گفته شد که عامل بدنبال یافتن بیشترین بازگشت مورد انتظار است. با در اختیار داشتن معادله ماتریسی بالا، این امر به سادگی حل می‌شود. داریم:

$$
v = (1 - \gamma \mathcal{P})^{-1} R \tag{۱۱-۲}
$$

[1] Bellman equation

با وجود اینکه این معادله به صورت مستقیم ارزش حالات موجود در فرآیند تصمیم‌گیری مارکوف موردنظر را می‌دهد، دو نکته در خصوص این روش حل وجود دارد:

۱- پیچیدگی محاسباتی[1] این روش از مرتبه زمانی $O(n^3)$ می‌باشد. بنابراین، راه‌حل مناسبی برای فرآیندهای تصمیم‌گیری مارکوف با تعداد حالت زیاد نمی‌باشد.

۲- حل این معادله، مستلزم دانستن ماتریس انتقال حالت ($\mathcal{P}$) می‌باشد. در خیلی از مسائل، به حدی از محیط آگاهی وجود ندارد که بتوان به این ماتریس دست پیدا کرد.

با در نظر گرفتن این دو نکته، واضح است که نمی‌توان حل مستقیم معادله بلمن را به مسائل بزرگتر تعمیم داد و به عبارتی، مقیاس‌پذیر[2] نیست. از این رو، نیاز است تا به صورت‌های دیگر معادله بلمن را برای فرآیند تصمیم‌گیری مارکوف موردنظر حل کرد.

برای آنکه بتوان مشخص کرد هر حالت موجود در فرآیند تصمیم‌گیری مارکوف چقدر خوب است، نیاز است ابتدا مشاهده کنیم که سیاست عامل در صورت رسیدن به آن حالت، چه عملی را اتخاذ می‌کند. یعنی:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s] \tag{۱۲-۲}$$

به ازای هر عمل موجود در حالت فعلی، چه مقدار است. واضح است که با دانستن سیاست عامل، می‌توان به کمک معادله بلمن، تابع ارزش حالت تمام حالات موجود در فرآیند تصمیم‌گیری مارکوف مورد نظر را یافت. به عبارتی دیگر:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{۱۳-۲}$$

که در آن، π همان سیاست عامل است. در این تعریف از تابع ارزش حالت، ارزش حالات به فراخور سیاستی که عامل در نظر دارد، تعیین می‌شوند (در ادامه نیز همین نسخه از تابع ارزش حالت مد نظر

[1] Computational complexity
[2] Scalable

است).

حال که سیاست عامل به صورت دقیق تعریف شد، می‌توان مشخص کرد که ارزش اتخاذ یک عمل مشخص در یک حالت معلوم، چقدر است. تابع Q(s, a) برای یک جفت حالت-عمل، ارزش انتظاری [1] از امتیاز کلی که از انجام آن عمل در آن حالت به دست می‌آید، را نشان می‌دهد. رابطه آن در ادامه آورده شده است:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \quad (۱۴-۲)$$

لازم به ذکر است که در صورت اطلاع نداشتن از سازوکار فرآیند تصمیم‌گیری مارکوف، باید مقادیر تابع ارزش عمل را یافت و نمی‌توان به کمک تابع ارزش حالت، معادله بلمن را حل کرد.

با تعریف تابع ارزش حالت و تابع ارزش عمل تحت سیاست π ، می‌توان معادله بلمن متناظر با این سیاست را نیز به شکل زیر تعریف کرد:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \quad (۱۵-۲)$$

و

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \quad (۱۶-۲)$$

از آنجا که انتظار می‌رود بتوان مقادیر تابع ارزش حالت و تابع ارزش عمل تحت سیاست مورد نظر را به نحوی یافت که این معادله‌ها برقرار باشند، به این نسخه به خصوص معادله بلمن، معادله انتظار بلمن[2] گفته می‌شود.

لازم به ذکر است که این نسخه از معادله بلمن نیز می‌تواند به صورت مستقیم حل شود؛ منوط به

[1] Expected value

[2] Bellman expectation equation

اینکه تعداد حالات موجود در فرآیند تصمیم‌گیری مارکوف زیاد نباشند و ماتریس انتقال حالت فرآیند موجود باشد.

مشاهده شد که چطور می‌توان با دانستن سیاست عامل، تابع ارزش حالت و تابع ارزش عمل را محاسبه کرد. در نهایت، هدف مسئله این است که عامل طبق سیاستی عمل کند که بتواند در حالت‌های مطلوبی از محیط قرار بگیرد و در آن حالات تصمیمات خوبی بگیرد. واضح است که اگر عامل به بهترین شکل ممکن عمل کند (به عبارتی، بهترین سیاست ممکن را داشته باشد)، آنگاه به تابع ارزش بهینه[1] خواهد رسید. به بیان ریاضی:

$$v_*(s) = \max_\pi v_\pi(s) \tag{۲-۱۷}$$

و

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{۲-۱۸}$$

این معادله‌ها که به ترتیب تابع ارزش حالت بهینه[2] و تابع ارزش عمل بهینه[3] را نمایش می‌دهند، مشخص می‌کنند که عامل چگونه باید به دنبال بهترین عملکرد باشد و به بیان ساده، بهینه[4] رفتار کند. با این حساب، حل مسئله یادگیری تقویتی تقلیل پیدا می‌کند به یافتن تابع ارزش بهینه.

به طور خلاصه، سیاست بهینه[5] سیاستی است که بتواند نسبت به تمام سیاست‌های دیگر برتر عمل کند. این یعنی در هر حالتی که باشد، بهتر از سایر سیاست‌ها بازگشت مورد انتظار دریافت کند و اگر عملی در یک حالت انجام می‌دهد، در ادامه به بهترین شکل ممکن رفتار کند. به عبارتی دیگر، با در نظر

[1] Optimal value function
[2] Optimal state-value function
[3] Optimal action-value function
[4] Optimal
[5] Optimal policy

گرفتن برقراری نامساوی:

$$\pi \geq \pi' \; if \; v_\pi(s) \geq v'_\pi(s), \forall s \tag{۱۹-۲}$$

باید رابطه زیر برقرار باشد:

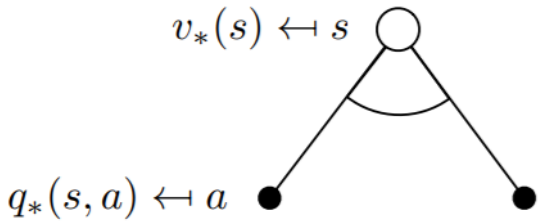
$$\pi_* \geq \pi, \forall \pi \tag{۲۰-۲}$$

که در آن، π_* همان سیاست بهینه است.

اگر قرار باشد بهینه بودن را به زبان معادله‌های بلمن بیان کرد، این طور تفسیر می‌شود که ارتباط بین مقادیر تابع ارزش حالت بهینه و تابع ارزش عمل بهینه باید به نحوی تنظیم شود که با قرار گرفتن در هر حالت، عملی انتخاب شود که تابع ارزش عمل مربوط به آن حالت را بیشینه کند. یعنی:

$$v_*(s) = \max_a q_*(s,a) \tag{۲۱-۲}$$

طرح‌واره‌ای از ارتباط بین این دو تابع ارزش بهینه در شکل ۲-۱ آورده شده است.



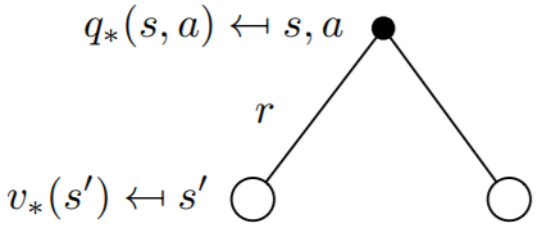
شکل ۲-۱ محاسبه تابع ارزش حالت بهینه از روی تابع ارزش عمل بهینه. در این گراف گره‌های سفید نشان‌دهنده مقادیر تابع ارزش به ازای هر حالت و گره‌های سیاه نشان‌دهنده مقادیر تابع ارزش عمل به ازای حالت گره پدر و هر عمل ممکن در آن حالت است. مقادیر یال‌ها نیز احتمال انتخاب شدن هر عمل را نشان می‌دهند[۴].

همچنین، برای انتخاب اعمال بهینه در هر حالت، نیاز است عامل بداند پس از انتخاب هر عمل در

چه حالاتی قرار می‌گیرد و آن را معیار قرار دهد. به عبارتی:

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_*(s\prime) \qquad (۲۲-۲)$$

که نوعی میانگین‌گیری از مقدار تابع ارزش حالت در حالاتی است که پس از اتخاذ عمل توسط عامل، ممکن است در آن قرار بگیرد (توجه شود که انتخاب اینکه حالت بعدی عامل کدام است، در اختیار عامل نیست و محیط آن را تصمیم می‌گیرد). طرح‌واره این معادله نیز در ۲-۲ آورده شده است. به دو معادله



شکل ۲-۲ محاسبه تابع ارزش عمل بهینه از روی تابع ارزش حالت بهینه. در این گراف گره سیاه نشان دهنده مقدار تابع ارزش عمل به ازای حالت فعلی و عمل انجام شده است. گره‌های سفید نیز نشان‌دهنده مقادیر تابع ارزش به ازای تمامی حالت‌های بعدی ممکن است. مقادیر یال‌ها نیز نشان‌دهنده پاداش مورد انتظار است[۴].

تابع ارزش قبل، معادلات بهینه بلمن[۱] گفته می‌شود.

۲-۳-۵ تفاوت یادگیری تقویتی با سایر الگوریتم‌های یادگیری ماشین

در بخش‌های قبل، مفاهیم پایه‌ای روش‌های حل مسائل یادگیری تقویتی معرفی شدند. یادگیری تقویتی[۲] از جهاتی با سایر الگوریتم‌های یادگیری ماشین تفاوت دارد. در اینجا به چند مورد اشاره می‌شود:

- عامل هیچ ناظری[۳] ندارد. تنها سیگنال پاداش به عامل جهت می‌دهد.
- پاداش معمولاً به صورت آنی دریافت نمی‌شود و در بسیاری از مواقع، تاخیریافته است. بنابراین، این که عامل تشخیص دهد چه حالات و اعمالی مطلوب می‌باشند، چالشی‌تر از الگووارههایی مثل یادگیری عمیق می‌باشد.

[1] Bellman optimality equation
[2] Paradigm
[3] Supervisor

- داده‌ها به صورت سری زمانی[1] وارد می‌شوند. بنابراین، باید به عواقب تصمیمی که عامل در یک گام زمانی می‌گیرد، آگاه باشد.
- عامل همواره باید محیط را اکتشاف کند؛ چراکه ممکن است حالاتی در محیط پیدا کند و اعمالی انجام دهد که پاداش بیشتری به آن بدهند. بنابراین، هیچگاه روند یادگیری عامل متوقف نمی‌شود.

۲-۴ الگوریتم

در این بخش به معرفی الگوریتم استفاده شده در پروژه می‌پردازیم.

۲-۴-۱ عامل یادگیری تقویتی عمیق

در این پروژه عامل شبکه-کیو عمیق [2] پیاده‌سازی کرده‌ایم.

۲-۴-۱-۱ شبکه-کیو عمیق

شبکه-کیو عمیق [۵]، یک تابع تقریب ارزش عمل می‌باشد که با استفاده از یادگیری-کیو[3]، شبکه را آموزش می‌دهد. تابع زیان [4] شبکه عصبی کیو به صورت زیر تعریف می‌شود:

$$L_i(\Theta_i) = \mathbb{E}_{(s,r,a,s')\sim U(D)}[(r + \max_{a'} \mathcal{Q}(s', a'; \Theta_i^-) - \mathcal{Q}(s, a; \Theta_i))^2] \quad (۲۳-۲)$$

که در آن $U(D)$ توزیع داده ذخیره شده در حافظه بازپخش تجربه است، Θ_i وزن‌های شبکه عصبی استفاده شده است و Θ_i^- وزن‌های یک شبکه عصبی کمکی است که پس از تعداد مشخصی به‌روزرسانی، برابر با وزن‌های شبکه عصبی اصلی قرار داده می‌شود. این کار باعث می‌شود که یادگیری عامل پایدارتر شود.

نکات زیر در مورد این الگوریتم قابل توجه هستند:

[1] Time series
[2] Deep Q-network
[3] Q-learning
[4] Loss Function

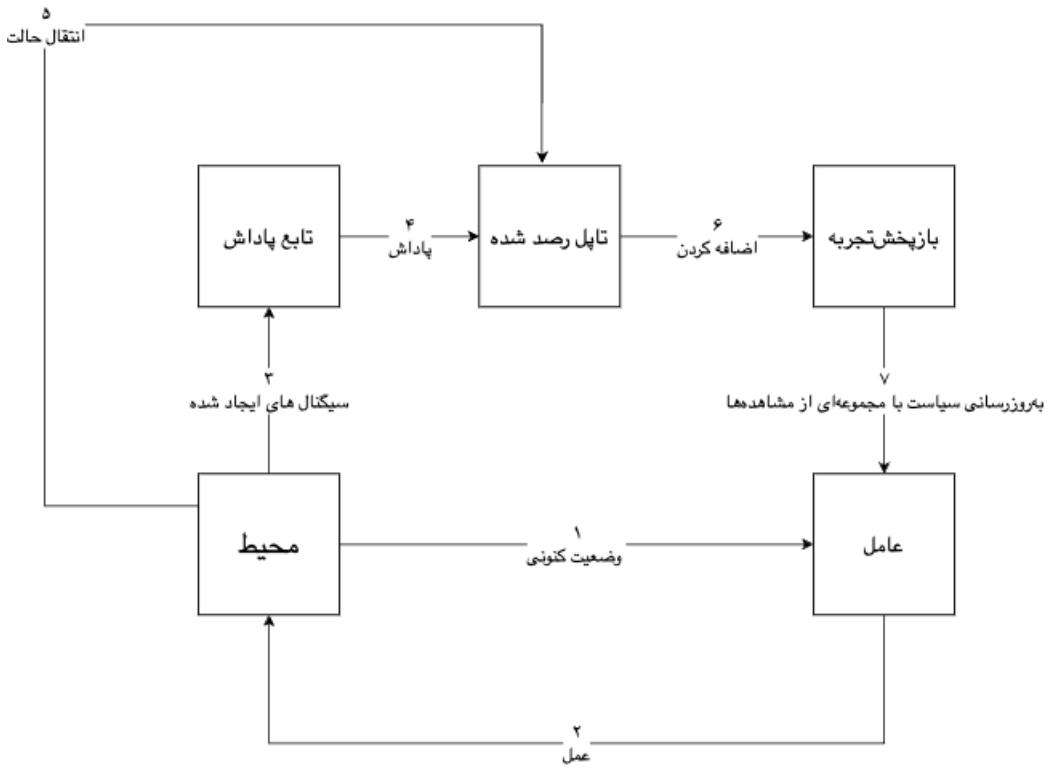
- این الگوریتم خارج از سیاست [1] است به این معنی که از دو سیاست به نام‌های سیاست رفتاری [2] و سیاست هدف [3] در هنگام تعلیم استفاده می‌کند. همچنین مانند دیگر الگوریتم‌های خارج از سیاست از حافظه بازپخش تجربه استفاده می‌کند.
- پیاده‌سازی این الگوریتم نسبت به الگوریتم‌های دیگر در یادگیری تقویتی ساده‌تر است چرا که تنها یک تابع را تخمین می‌زند.
- رفتار الگوریتم در زمان تعلیم پایدار نیست.

در شکل ۲-۳ خلاصه‌ای از نحوه کار کرد این الگوریتم نشان داده شده است. در ابتدا عامل حالت را در یک گام زمانی خاص دریافت می‌کند و با توجه به آن در مورد عمل بعدی تصمیم می‌گیرد. پس از انجام عمل انتخاب شده در محیط، سیگنال‌های تحریک شده مهم از محیط به تابع پاداش داده می‌شوند و تابع پاداش با کمک آن‌ها پاداش را محاسبه می‌کند. پاداش محاسبه شده به همراه حالت جدید، حالت قبلی و عمل انجام‌شده در حافظه بازپخش تجربه ذخیره می‌شوند (این حافظه برای الگوریتم بهینه‌سازی سیاست تقریبی یک بافر موقت به اندازه T است) . سپس برای هر به‌روزرسانی سیاست تعدادی از این مشاهدات به صورت تصادفی انتخاب می‌شوند.

۲-۵ خلاصه

در این فصل با مفاهیم پایه یادگیری تقویتی آشنا شدیم. سپس به الگوریتم یادگیری تقویتی عمیق استفاده‌شده در پروژه پرداختیم. تا به اینجای کار با الگوریم موجود در پروژه آشنا شدیم که نحوه کار کرد آن را در شکل ۲-۳ مشاهده شد. در فصل‌های بعدی به پیاده‌سازی و بررسی نتایج این الگوریتم می‌پردازیم. منبع بیشتر مطالب معرفی‌شده [۴] است .

[1] Off-policy
[2] Behavior Policy
[3] Target Policy



شکل ۲-۳ ساختار کلی عامل یادگیری تقویتی عمیق

فصل سوم

محیط شبیه‌سازی ایپانت

از آنجایی که محک‌زنی[1] الگوریتم مد نظر در محیط ایپانت [2] صورت می‌گیرد، نیاز است با ساختار داخلی آن بیشتر آشنا شویم. در این فصل، به بررسی نحوهٔ عملکرد و تعامل با این محیط پرداخته شده است.

۳-۱ معرفی

ایپانت، یک شبیه‌ساز[3] متن‌باز[4] است که به منظور پژوهش در حوزهٔ شبکه‌های آب‌رسانی استفاده می‌شود[۶]. شبیه‌ساز ایپانت با زبان سی نوشته شده است. ایپانت با این هدف طراحی و توسعه داده شده است که بتوان فرآیندهای هیدرولیکی در شبکه‌های توزیع آب را شبیه‌سازی کند. به کمک ایپانت می‌توان شرایط مختلف محیطی مانند سیل و زلزله را نیز شبیه‌سازی کرد.

شکل ۳-۱ لوگوی ایپانت

۳-۲ مزایای استفاده از محیط شبیه‌ساز

قبل از آنکه به معماری شبیه‌ساز ایپانت پرداخته شود، خوب است در خصوص مزایای استفاده از محیط‌های شبیه‌ساز به جای آزمودن در دنیای واقعی توضیح داده شود. در ادامه، شماری از این مزایا آورده شده است :

- کاهش هزینهٔ سرمایه‌گذاری: اگر بخواهیم الگوریتم‌های خود را در محیط واقعی تست و اصلاح کنیم نیاز به پیاده‌سازی محیط بزرگی برای تست است که این در عمل بهینه نیست. از سوی دیگر تست کردن الگوریتم‌های مختلف روی محیط واقعی نیز هزینه زمانی و ریسک بالایی دارد. بدین منظور استفاده از شبیه‌ساز ایپانت به ما کمک می‌کند تا به این امر سرعت ببخشیم.

[1] Benchmark
[2] EPANET
[3] Simulator
[4] Open-source

- کاهش هزینهٔ جمع‌آوری داده: جمع‌آوری داده برای یک شبکه آب‌رسانی در دنیای واقعی، فرآیندی زمان‌بر و طاقت‌فرسا است. هر شبکه آب تنها می‌تواند به میزان محدودی داده فراهم کند. در محیط شبیه‌ساز، برخلاف دنیای واقعی، می‌توان همزمان ده‌ها شبکه آب را آموزش داد و اعتبارسنجی کرد. از این رو، این ابزار به فرآیند جمع‌آوری داده برای یافتن بهترین الگوریتم سرعت می‌بخشد.
- آموزش سریع‌تر: در محیط واقعی اعمال تغییرات زمان‌بر می‌باشد، همچنین، برای جمع‌آوری داده باید زمان زیادی را صرف کرد اما در محیط شبیه‌ساز می‌توان محیط را سریع‌تر اجرا نمود و روزهای بیشتری را برای یک اجرا در نظر گرفت.
- عمومی‌سازی پژوهش: به علت کم هزینه بودن و آسانی اجرا و در دسترس عموم قرار گرفتن محیط شبیه‌ساز ایپانت در همه جای جهان به صورت موازی به پژوهش روی محیط‌های آبی پرداخت .
- امکان آزمودن موارد گوشه‌ای[1]: در تعدادی از موارد، شبکه آب ممکن است با وضعیتی روبرو شود که به‌ندرت اتفاق می‌افتد. به عنوان مثال، شبکه آبی ممکن است در شرایطی مانند سیل و زلزله قرار بگیرد که بسیار روی محیط آبی تاثیرگذار باشد و آزمودن در این شرایط بسیار به ندرت اتفاق افتد و یا حتی خطرزا باشد. اما در محیط شبیه‌ساز، خطری وجود ندارد و از این رو، می‌توان سامانه‌های مقاوم‌تری [2] را طراحی کرد و توسعه داد.

با در نظر گرفتن این موارد، به نظر می‌رسد تصمیم معقولی است که چالش حل کردن مسئله کنترل فشار آب در شبکه آب را ابتدا با محیط‌های شبیه‌ساز پیش برده و سپس در صورت گذراندن تمام موانع موجود و اطمینان از صحت عملکرد الگوریتم، به پیاده‌سازی سخت‌افزاری و آزمون در دنیای واقعی رو بیاوریم.

۳-۳ معماری ایپانت

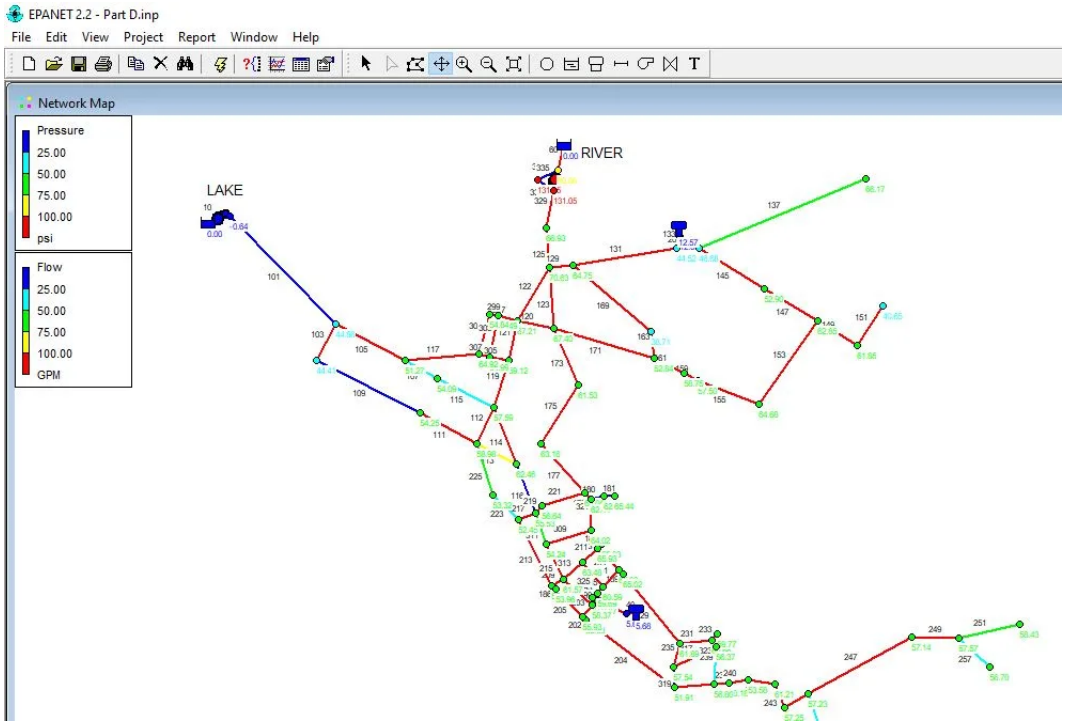
معماری مبتنی بر اجزا [3] : ایپانت از معماری مبتنی بر اجزا پیروی می‌کند، به این معنی که نرم‌افزار از ماژول‌ها یا اجزای مختلفی تشکیل شده است که وظایف خاصی را انجام می‌دهند. این اجزا برای شبیه‌سازی رفتار شبکه توزیع آب با هم کار می‌کنند. اجزای اصلی ایپانت عبارتند از:

- جزء توضیحات شبکه: این قسمت به کاربر اجازه می‌دهد که اجزای شبکه را را تعریف کند. این

[1] Corner case
[2] Robust
[3] Component-based

جزء شامل قابلیت‌هایی برای وارد کردن اطلاعاتی مانند طول لوله، قطر لوله، داده‌های ارتفاع، میزان مصرف گره‌ها و مشخصات مخزن است.

- جزء حل‌کننده هیدرولیک: حل‌کننده هیدرولیک مسئول محاسبه دبی و فشار در سراسر شبکه است. از الگوریتم‌ها و معادلات ریاضی برای حل معادلات هیدرولیکی حاکم بر رفتار لوله‌ها، پمپ‌ها، شیرها و سایر عناصر شبکه استفاده می‌کند.
- جزء تجزیه و تحلیل کیفیت آب: ایپانت همچنین شامل یک جزء تجزیه و تحلیل کیفیت آب است که کاربران را قادر می‌سازد تا حرکت و اختلاط اجزای آب (مانند کلر یا آلاینده‌ها) را در سیستم توزیع شبیه‌سازی کنند. این جزء به ارزیابی ویژگی‌های کیفیت آب، شناسایی خطرات احتمالی آلودگی و ارزیابی اثربخشی فرآیندهای تصفیه آب کمک می‌کند.
- جزء نتایج و تجسم: ایپانت ابزارهایی برای تجسم و تفسیر نتایج شبیه‌سازی فراهم می‌کند. این مولفه به کاربران اجازه می‌دهد تا گزارش‌ها، جداول، نمودارها و نقشه‌هایی تولید کنند که معیارهای مختلف هیدرولیک و کیفیت آب را نمایش دهد، تجزیه و تحلیل داده‌ها را تسهیل کند و به تصمیم‌گیری آگاهانه در مورد بهینه‌سازی و نگهداری شبکه کمک کند [۶] . نمونه‌ای از این تجسم در شکل ۳-۲ نشان داده شده است.



شکل ۳-۲ نمونه‌ای از شبکه توزیع آب در ایپانت

۳-۴ قابلیت‌های موجود در ایپانت

واسط برنامه‌نویسی کاربردی[1] ایپانت، قابلیت‌هایی را فراهم کرده است که بتوان به کمک آن فرآیند طراحی، توسعه و آزمودن شبکه‌های توزیع آب را تسهیل کرد و سرعت بخشید. در این بخش، به بررسی اجمالی این قابلیت‌ها می‌پردازیم. همچنین ابزاری به عنوان بسته توسعه نرم‌افزار [2] در اختیار ما گذاشته است که برخی از این قابلیت‌ها را برای ما بهبود و توسعه داده است که در این بخش به توضیح این قابلیت‌ها میپردازیم:

۳-۴-۱ قابلیت اعمال شرایط محیطی مختلف

در محیط ایپانت و با استفاده از بسته توسعه نرم‌افزار WNTR می‌توان شبکه آب را در شرایط بحرانی و محیطی مختلف سنجید که به شرح زیر است [۷] :

- زلزله: زلزله می‌تواند برخی از ناگهانی‌ترین و تاثیرگذارترین بلایایی باشد که یک سیستم آبی تجربه می‌کند. یک زلزله می‌تواند آسیب‌های پایداری را به سیستم وارد کند که ترمیم کامل آن ممکن است هفته‌ها یا ماه‌ها طول بکشد. زلزله می‌تواند به لوله‌ها، مخازن، پمپ‌ها و سایر زیرساخت‌ها آسیب برساند. علاوه بر این، زلزله می‌تواند باعث قطع برق و آتش‌سوزی شود.
- آتش: کتابخانه WNTR را می‌توان برای شبیه‌سازی آسیب وارد شده به اجزای سیستم در اثر آتش‌سوزی و یا برای شبیه‌سازی مصرف آب به دلیل اطفاء حریق استفاده کرد. برای مقابله با آتش‌سوزی، آب بیشتری از سیستم گرفته می‌شود. برای مثال، آتش سوزی‌های کوچک مسکونی ممکن است به ۱۵۰۰ گالن در دقیقه برای ۲ ساعت نیاز داشته باشد و یا فضاهای تجاری بزرگ ممکن است به ۸۰۰۰ گالن در دقیقه برای ۴ ساعت نیاز داشته باشند. این تقاضای اضافی می‌تواند تأثیر زیادی بر فشار آب در سیستم داشته باشد.
- قطعی برق: قطع برق می‌تواند کوچک و کوتاه باشد، یا می‌تواند چندین روز طول بکشد و کل مناطق را تحت تأثیر قرار دهد. در سیستم‌های توزیع آب، قطع برق می‌تواند باعث خاموش شدن ایستگاه‌های پمپ و کاهش فشار آب شود. این می‌تواند منجر به کمبود فشار در برخی از مناطق سیستم شود.

[1] Application Programming Interface (API)

[2] SDK

- تغییرات محیطی: تغییرات محیطی یک مشکل طولانی مدت برای سیستم‌های توزیع آب است. تغییرات در محیط می‌تواند منجر به کاهش دسترسی به آب، آسیب ناشی از حوادث آب و هوایی یا حتی آسیب ناشی از فرونشست شود. این امر به ویژه در شهرهایی که بر روی خاک‌های ناپایدار ساخته شده‌اند که در حال فرونشست زمین هستند، رایج است.
- نشتی یا شکستن لوله: لوله‌ها مستعد نشتی هستند. نشت می‌تواند به دلیل فرسودگی زیرساخت، فرآیند یخزدگی، ذوب، افزایش تقاضا یا تغییرات فشار ایجاد شود. این نوع آسیب به ویژه در شهرهای قدیمی که سیستم‌های توزیع از مواد قدیمی مانند چدن و حتی چوب ساخته شده اند، رایج است.

۳-۴-۲ انواع معیارها

نرم‌افزار ایپانت معیارهای مختلفی را برای کاربر فراهم کرده است که به‌وسیله آن می‌توان ویژگی‌های مختلف شبکه آب را اندازه‌گیری کرد. از جمله از این معیارها، می‌توان به معیارهای کیفیت آب ، فشار آب، ارتفاع آب، میزان جریان آب و جهت آب اشاره کرد. این معیارها را می‌توانید در جدول ۳-۲ مشاهده کنید.

۳-۵ امکان ادغام ایپانت با زبان‌های برنامه‌نویسی

همانطور که در بخش‌های قبلی نیز گفته شد، هسته [1] ایپانت با زبان برنامه‌نویسی C نوشته شده است، و استفاده از این ابزار را سخت می‌کند، منتها در برخی از زبان‌ها مانند زبان پایتون [2] کتابخانه‌هایی وجود دارد که از این هسته استفاده می‌کنند و برنامه‌نوشتن و استفاده از ایپانت را برای ما آسان‌تر می‌کنند. از جمله از این کتابخانه‌ها می‌توان به WNTR و epanet-python و Epyt اشاره کرد که ما در بخش‌های بعدی توضیح خواهیم داد که استفاده بیشتر ما از WNTR [۷] خواهد بود.

[1] Core
[2] Python

جدول ۳-۱ معیارهای موجود در نرم‌افزار ایپانت به زبان فارسی

واحد اندازه‌گیری	معیار
میلی‌گرم بر لیتر یا میکروگرم بر لیتر	غلظت
(مشاهده واحدهای جریان)	تقاضا
اینچ، میلیمتر	قطر (لوله‌ها)
فوت، متر	قطر (مخازن)
درصد	کارایی
فوت، متر	ارتفاع
واحد جریان/ (پوند بر اینچ مربع)$^{1/2}$، واحد جریان/ (متر مربع)$^{1/2}$	ضریب گزارش‌دهی
کیلووات-ساعت	انرژی
CFS (فوت مکعب بر ثانیه)، GPM، MGD، IMGD، AFD، LPS، LPM، MLD، CMH، CMD	جریان
بدون واحد	ضریب اصطکاک
فوت، متر	سرشاری هیدرولیک
فوت، متر	طول
بدون واحد	ضریب افت فشار کوچک
اسب بخار، کیلووات	قدرت
پوند بر اینچ مربع، متر	فشار
۱-مرتبه ۱/روز	ضریب واکنش (جمع‌بندی)
۰-مرتبه کمیت/لیتر/روز، ۱-مرتبه فوت/روز، ۰-مرتبه کمیت/لیتر/روز، ۱-مرتبه متر/روز	ضریب واکنش (دیوار)
$۱۰^{-۳}$ فوت، میلی‌متر	ضریب خشنی
کمیت/دقیقه	تزریق کمیت جرم منبع
فوت بر ثانیه، متر بر ثانیه	سرعت
فوت مکعب، متر مکعب	حجم
ساعت	سن آب

جدول ۳-۲ معیارهای موجود در نرم‌افزار ایپانت به زبان انگلیسی

معیار	واحد اندازه‌گیری
Concentration	mg/L or µg/L
Demand	(see Flow units)
Diameter (Pipes)	inch, millimeter
Diameter (Tanks)	foot, meter
Efficiency	percent
Elevation	foot, meter
Emitter Coefficient	flow unit/ $(psi)^{1/2}$, flow unit/ $(meter)^{1/2}$
Energy	kilowatt-hour
Flow	CFS (cu foot/sec), GPM, MGD, IMGD, AFD, LPS, LPM, MLD, CMH, CMD
Friction Factor	unitless
Hydraulic Head	foot, meter
Length	foot, meter
Minor Loss Coefficient	unitless
Power	horsepower, kilowatt
Pressure	pounds per square inch, meter
Reaction Coefficient (Bulk)	1st-order 1/day
Reaction Coefficient (Wall)	0-order mass/L/day, 1st-order ft/day, 0-order mass/L/day, 1st-order meter/day
Roughness Coefficient	Darcy-Weisbach $۱۰^{-۳}$ foot, Otherwise unitless, Darcy-Weisbach millimeter, Otherwise unitless
Source Mass Injection	mass/minute
Velocity	foot/second, meter/second
Volume	cubic foot, cubic meter
Water Age	hour

۳-۶ خلاصه

در این فصل به معرفی شبیه‌ساز ایپانت پرداختیم و با امکانات آن آشنا شدیم. حال در فصل بعد به تنظیماتی که برای استفاده از آن انجام داده‌ایم، می‌پردازیم.

فصل چهارم

طراحی و پیاده‌سازی

در این فصل به توضیحات مربوط به پیاده‌سازی‌ها و راه حل پیشنهادی می‌پردازیم. ابتدا به فضای حالت مسئله و تابع پاداش می‌پردازیم، سپس به تعریف ابرپارامترهای الگوریتم و مقادیری که به آن‌ها اختصاص داده‌ایم، می‌پردازیم. در انتها نیز نتایج بدست آمده را تحلیل می‌کنیم.

۴-۱ فضای حالت

یکی از ارکان اساسی در الگوریتم‌های یادگیری تقویتی، تعریف فضای حالت می‌باشد. تعیین یک فضای حالت مناسب از اهمیت خاصی برخوردار است؛ زیرا این فضا به عامل امکان تصمیم‌گیری و اجرای یک سیاست بهینه را می‌دهد. در این بخش، به بررسی ابعاد مختلف فضای حالت می‌پردازیم و نحوه‌ی انتخاب و تعریف آن را مورد بررسی قرار می‌دهیم. برای ساختن فضای حالت، اطلاعات فراوانی در محیط قابل استخراج و استفاده وجود دارند. این اطلاعات می‌توانند شامل معیارهایی نظیر فشار، جریان آب، کیفیت آب و سایر ویژگی‌های محیط باشند. این ابعاد مختلف به عامل امکان می‌دهند تا وضعیت کنونی محیط را به درستی درک کرده و بر اساس آن تصمیمات مناسبی را اتخاذ کند. از سوی دیگر، برخی از این ابعاد با ساختار شبکه الگوریتم مرتبط هستند. این متغیرها ممکن است شامل ساختار گرافی شبکه، جنس لوله‌ها، قطر و طول لوله‌ها و ویژگی‌های دیگر باشند که تأثیر بسزایی در عملکرد سیستم دارند. به‌طور کلی، اضافه کردن همه این ابعاد به عنوان ورودی مسئله یادگیری تقویتی، مسئله را پیچیده‌تر و چالش‌برانگیزتر می‌کند. در این پروژه، هدف ما از طراحی الگوریتم، این است که بدون نیاز به جزئیات دقیق از ساختار شبکه، یک الگوریتم کارآمد و انعطاف‌پذیر را ارائه دهیم. برای ساده‌سازی مسئله و بهبود زمان حل مسئله، سه نکته زیر را در نظرگرفته‌ایم:

- طراحی الگوریتم با انعطاف‌پذیری بالا: الگوریتمی ارائه داده‌ایم که بتواند با تغییرات در ابعاد حالت، همچنان به‌صورت کارآمد عمل کند و نیاز به تغییرات اساسی در ساختار الگوریتم نداشته باشد.
- تنها از معیار فشار به عنوان ورودی مسئله استفاده می‌کنیم.
- حتی برای افزایش سرعت زمان اجرا، می‌توانیم فشار همه نقاط را به عنوان ورودی در نظر نگیریم و تنها برخی از نقاط حساس شبکه آب را در نظر بگیریم. هرچند با در نظر گرفتن گره‌های بیشتری در شبکه به عنوان ورودی مسئله، تخمین دقیق‌تر و اجرای بهتری را خواهیم داشت.

۴-۱-۱ انتخاب ویژگی

همانطور که گفته شد ویژگی‌های مختلفی را برای ورودی مسئله می‌توان در نظر گرفت ولی از آنجایی که اکثر ویژگی‌های گفته شده به فشار آب مرتبط [1] هستند، اضافه کردن این ویژگی‌ها به عنوان ورودی مسئله تنها باعث ایجاد پیچیدگی بیشتر در محسابات می‌شود. پس فشار را به عنوان مهم‌ترین ویژگی مسئله می‌توان در نظر گرفت [۱] . برای مسائلی که تمام نشدن میزان حجم آب تانک‌های آب نیز جزو اهداف باشد، می‌توان از سطح آب تانک و زمان نیز که به فشار آب وابسته نیستند، علاوه بر فشار، آب به عنوان ویژگی ورودی مسئله استفاده کرد [۳]. با توجه به تعریف مسئله و اهدافی که در فصل مقدمه برای این پروژه بیان شد، در اینجا تنها فشار به عنوان ورودی مسئله در نظر گرفته می‌شود.

۴-۲ فضای عمل

یکی از مهم‌ترین کارهایی که برای یک مدل یادگیری عمیق باید انجام داد این است که فضای عمل [2] آن را مشخص نمود. برای این مسئله، عمل‌های [3] ما تنظیم مقادیر شیرهای کاهنده فشار آب در شبکه آب است. از آنجایی که هر شیر کاهش فشار آب [4] می‌تواند مقادیر پیوسته زیادی را داشته باشد، با افزایش تعداد شیرهای آب تعداد عمل‌ها نیز به‌صورت نمایی افزایش خواهد یافت. برای مثال اگر در شبکه ۳ شیر وجود داشته باشد و هر شیر ۱۰ مقدار متفاوت بتواند بپذیرد، عامل در هر گام باید بین $۱۰^۳$ عمل مختلف، عمل بهینه را انتخاب کرد. از طرفی انتخاب کردن مقادیر ممکن فشار برای هر شیر آب یکی از چالش‌های انتخاب عمل است. در این پروژه ما برای تبدیل مقادیر پیوسته فشار شیرهای آب به مقادیر گسسته، آن را به بازه‌هایی به طول ۱۰ تقسیم‌بندی کرده‌ایم. عمل‌های عامل مجموعه‌ای به طول تعداد شیرهای شبکه خواهد بود که اعداد این مجموعه اعداد زیر خواهند بود.

$$\{۱۰, ۲۰, ۳۰, \dots , ۲۰۰\} \quad (۱-۴)$$

[1] Correlated
[2] Action Space
[3] Actions
[4] PRV (Pressure Reduce Valve)

بنابراین تعداد عمل‌های ممکن از رابطه زیر به‌دست می‌آید.

$$count(actions) = n^c \quad (۴-۲)$$

در اینجا n تعداد اعدادی است که می‌توان برای یک شیر آب تعیین کرد و همچنین c تعداد شیرهای آب در شبکه است.

۴-۳ تابع پاداش

خروجی تابع پاداش، رفتار عامل را شکل می‌دهد. با تعریف مناسب این تابع، می‌توان عامل را به سمت سیاست لازم برای حل مسئله راهنمایی کرد. علاوه بر تعریف مناسب، بزرگی و کوچکی مقادیر پاداش‌ها نیز از اهمیت بالایی برخوردارند. مقادیر بزرگ برای پاداش‌ها به یادگیری سرعت می‌دهند اما بزرگ بودن بیش از حد آنها باعث می‌شود که الگوریتم با یک سیاست محلی بهینه گیر افتد. از طرفی مقادیر بسیار کوچک نیز باعث می‌شوند که عامل نتواند از سیگنال‌های مفیدی که از محیط دریافت می‌کند به خوبی استفاده کند. ما در این بخش به بررسی دو تابع پاداش مختلفی که برای این مسئله در نظر گرفتیم می‌پردازیم.

- پاداش شماره یک: در این پاداش هدف کنترل میانگین فشار کل شبکه است.
- پاداش شماره دو: در این پاداش هدف کنترل میانگین فشار آب است با این تفاوت که فشار هیچ یک از گره‌های تقاضا نیز زیر فشار کیمنه [۱] نباشد

در ادامه به تعریف هر یک از این پاداش‌ها می‌پردازیم.

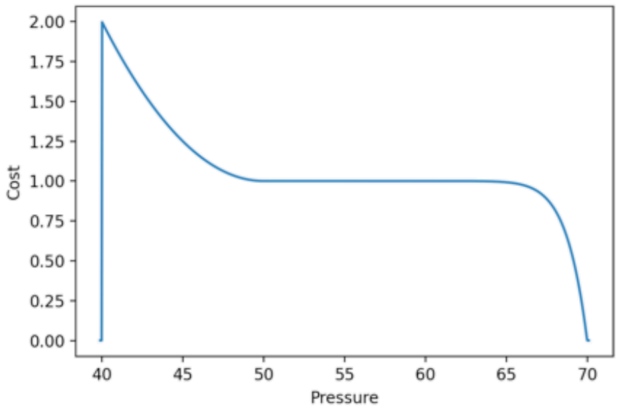
[1] Minimum

۴-۳-۱ پاداش شماره یک

از اهداف مهم عامل، نگه داشتن فشار در یک بازه مشخص قبل از پایان اپیزودها [۱] است. که معمولا این مقدار فشار عددی بین ۴۰ تا ۷۰ پوند بر اینچ مربع [۲] است که مقدار قابل تحملی برای گره‌های تقاضا [۳] است. برای رسیدن به این هدف تابع پاداش باید تابعی از این بازه عددی باشد. برای همین تابع هدف ما به‌صورت زیر تعریف می‌شود.

$$r = \frac{\sum_{i=۱}^{N} i \text{ فشار گره } \times cost(i)}{N} \qquad (۴-۳)$$

در این معادله منظور از r ، پاداش مربوط به فشار در شبکه است. همچنین N در این معادله نشان‌دهنده تعداد گره‌های مشخص‌شده برای استخراج فشار است. تابع $cost(i)$ نیز در شکل ۴-۱ نمایش داده شده است. دلیل استفاده از این تابع این است که عامل سعی کند نه‌تنها فشار میانگین را در بازه ۴۰ تا ۷۰ نگه دارد بلکه سعی کند این فشار میانگین را به عدد ۴۰ نزدیک کند تا انرژی کمتری مصرف بشود.



شکل ۴-۱ هزینه در نظر گرفته شده برای بهینه کردن میانگین

[1] Episodes
[2] PSI
[3] Demand Nodes

۴-۳-۲ پاداش شماره دو

این پاداش برگرفته‌شده از پاداش شماره یک است و همان هدف را دنبال می‌کند با این تفاوت که در پاداش شماره یک، یک مشکل اساسی وجود دارد و آن مشکل این است که با وجود اینکه میانگین فشار شبکه حفظ می‌شود ولی ممکن است برخی از گره‌های تفاضا حداقل فشار مورد نیاز را نداشته باشند یا فشار در برخی از نقاط صفر و یا حتی منفی شود. برای جلوگیری از این اتفاقات در پاداش شماره دو ملاحظاتی انجام شده است که از این اتفاقات جلوگیری می‌کند. فرمول پاداش بدین شکل خواهد بود:

$$r = \frac{\sum_{i=۱}^{N} p(i)}{N} \tag{۴-۴}$$

$$p(i) = \begin{cases} ۰, & ۰ < i \leq ۴۰ \ or \ i > ۷۰ \\ i \times cost(i), & ۴۰ < i \leq ۷۰ \\ -۱۰۰۰, & i \leq ۰ \end{cases} \tag{۴-۵}$$

در اینجا نیز از فانکشن cost(i) که در شکل ۴-۱ آمده است، استفاده شده است. دلیل اینکه از پاداش بسیار منفی مانند ۱۰۰۰- استفاده شده است، این است که عامل سعی کند به هیچ‌وجه اجازه به‌وجود آمدن فشار منفی در یک گره را ندهد حتی اگر این امر باعث افزایش میانگین فشار شبکه شود.

۴-۴ پیاده‌سازی عامل

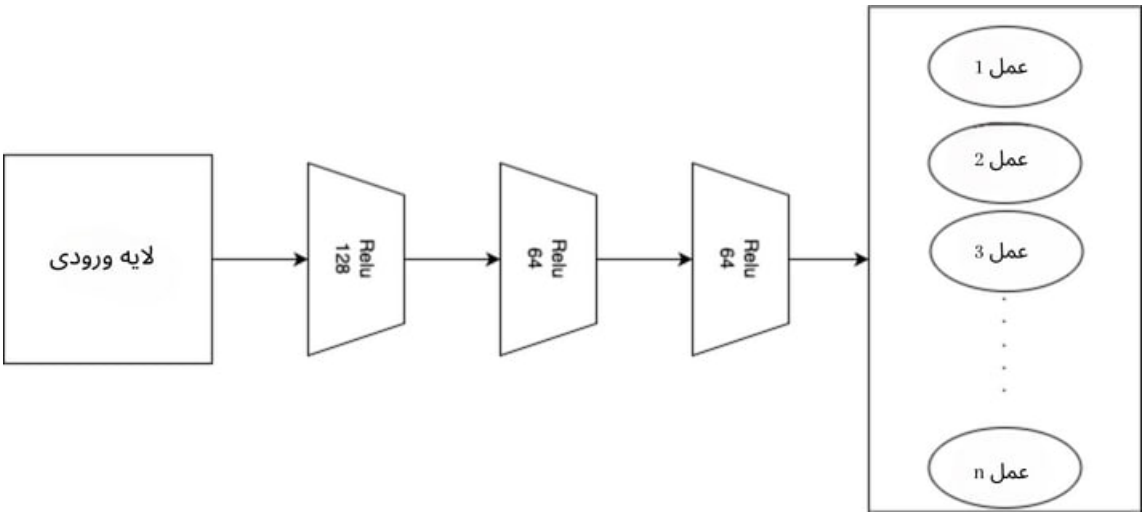
در این قسمت سعی داریم تا ساختار عامل را شرح دهیم. یکی از قسمت‌های مهم مدل یادگیری عمیق پیاده‌سازی عامل [۱] است. عامل وظیفه انتخاب عمل مناسب با توجه به وضعیت [۲] مشاهده‌شده در محیط را دارد. به دلیل اینکه ورودی مسئله ما برداری از اعداد است که هر عدد نیز حالات بسیار مختلفی را

[1] Agent
[2] State

شامل می‌شود و همچنین عمل‌ها نیز به تعداد زیادی وجود دارند، برای انتخاب مناسب‌ترین عمل باید از مدل‌های آماری استفاده کرد که بتواند وضعیت دریافت‌شده از محیط را به یک عمل مناسب متصل کند و آن را اجرا کند. بدین منظور ما از شبکه‌های عصبی استفاده می‌کنیم.

۴-۴-۱ مدل عصبی

در واقع بخش اصلی سازنده عامل ما و قوه تصمیم‌گیری آن مدل آماری عصبی خواهد بود. برای مسئله کنترل فشار ما از یک شبکه عصبی معمولی استفاده کرده‌ایم. بدین شکل که لایه اول مجموعه‌ای از اعداد فشار هستند و سه لایه مخفی [۱] که از نوع کامل متصل به هم [۲] هستند استفاده کرده‌ایم. در آخر نیز لایه خروجی وجود دارد که برداری به طول تعداد عمل‌های ممکن است. در واقع هر عدد در لایه آخر میزان پاداش توسط انجام آن عمل در آینده را تخمین می‌زند و وظیفه عامل انتخاب عملی است که این عدد را بیشینه کند. ساختار مدل در شکل ۴-۲ نشان داده شده است.



شکل ۴-۲ معماری شبکه عصبی استفاده‌شده برای پیاده‌سازی عامل در الگوریتم

[1] Hidden Layer
[2] Fully Connected

۴-۴-۱-۱ تنظیمات شبکه عصبی

در جدول ۴-۱ مشخصات شبکه عصبی استفاده شده در پروژه بیان شده است. به طور خلاصه، در این پروژه، از یک شبکه عصبی کاملا متصل [۱] استفاده شده است. برای پیاده‌سازی این مدل از کتابخانه پایتونی Keras استفاده شده است.

جدول ۴-۱ مشخصات شبکه عصبی

ویژگی	مقدار
تعداد لایه‌های پنهانی	۳
اندازه لایه پنهانی یک	۱۲۸
اندازه لایه پنهانی دو	۶۴
اندازه لایه پنهانی سه	۶۴
توابع فعال‌سازی لایه‌های پنهانی	Relu
تابع فعال‌سازی لایه آخر	Linear
تابع خطا	Huber
بهینه‌ساز	Adam
میزان یادگیری	۰/۰۲۵

۴-۵ تنظیمات محیط شبیه‌سازی

در این بخش به تنظیمات مربوط به محیط شبیه‌سازی می‌پردازیم. تمامی تنظیمات ساختاری شبکه‌های مختلف در پوشه /networks ذخیره شده‌اند. برای نمونه مشخصات شبکه net1 [۸] ، در جدول ۴-۲ نشان داده شده است. به ازای هر شبکه آب ساختار شبکه و تنظیمات مرتبط با شبیه‌سازی آن وجود دارد. پیاده‌سازی‌های مربوط به شبیه‌ساز در فایل WNTR_environment.py آمده است. در این فایل کلاس WaterNetworkEnv وجود دارد که وظیفه آن پیکربندی محیط ایپانت و اتصال به آن است. این کلاس مانند محیط‌های معروفی که در کتابخانه GYM وجود دارد، پیاده‌سازی شده است. برخی از متغیرها و توابع مهم آن در ادامه لیست شده است:

- تابع __init__: در این تابع، به ایپانت متصل شده و عامل ساخته می‌شود.
- تابع reset: در این تابع محیط ایپانت از ابتدا و با مشخصات اولیه‌ای که از روی فایل پیکربندی [۲] خوانده می‌شود، ساخته می‌شود.

[1] Fully Connected

[2] config file

- تابع step: در این تابع، پس از دریافت عمل اتخاذشده توسط عامل، عمل را در محیط شبیه‌ساز اعمال کرده و متناظر با نحوهٔ پاداش‌دهی محیط، پاداش را برمی‌گرداند. همچنین، مشخص می‌کند که آیا اپیزود خاتمه پیدا کرده است یا خیر.
- EPISODE_LENGTH: نشان‌دهنده طول اپیزود است. در آزمایش‌های انجام‌شده طول هر اپیزود ۷۲ گام زمانی است که به عبارتی دیگر برابر با ۳ روز است.
- Network_NAME: نام فایل مشخصات شبکه آب مدنظر برای اجرا شدن روی محیط ایپانت است.
- DEFAULT_ACTION_ZONE: همانطور که گفته شد، برای گسسته کردن فضای عمل‌ها اعدادی را به عنوان عمل‌های مجاز انتخاب کرده و به محیط می‌دهیم. بدیهی است که هر چه تعداد این اعداد و فاصله آنها کمتر باشد دقت تصمیم‌گیری بیشتر خواهد بود.
- REWARD_FUNCTION_TYPE: مشخص می‌کنیم که از کدام یک از توابع پاداش استفاده شود و دو مقدار REWARD_FUNCTION_2 و REWARD_FUNCTION_1 را می‌پذیرد.

۴-۶ تنظیمات شبکه-کیو عمیق

ابرپارامترهای این الگوریتم در جدول ۴-۳ آمده است.

این پارامترها عبارت‌اند از:

- learning_rate: این پارامتر نرخ یادگیری است. این پارامتر اندازه گرادیان‌ها را در هنگام به‌روزرسانی شبکه کنترل می‌کند.
- gamma: این ابرپارامتر نرخ تخفیف است [۹] .
- batch_size: تعداد مثال‌هایی که برای آموزش مدل عصبی انتخاب می‌شود و آن را آموزش می‌دهد.
- epsilon: احتمال انتخاب تصادفی اعمال به جای استفاده از شبکه عصبی را نشان می‌دهد. این احتمال در طول تعلیم با نرخ eps_dec کاهش می‌یابد تا به مقدار eps_end برسد.
- max_mem_size: اندازه حافظه بازپخش تجربه.
- n_time_steps: تعداد گام‌های زمانی فاز آموزش است. این عدد برای آموزش شبکه‌ها، بین ۲۵۰ تا ۵۰۰ گام است.

پیاده‌سازی‌های مربوط به این الگوریتم در پوشه DQN و در فایل train.py آمده است.

جدول ۴-۲ پارامترهای ساختاری شبکه net1 به

تقاطع‌ها

شناسه	ارتفاع	تقاضا	الگو
۱۰	۷۱۰	۰	۲
۱۱	۷۱۰	۱۵۰	۲
۱۲	۷۰۰	۱۵۰	۲
۱۳	۶۹۵	۱۰۰	۲
۲۱	۷۰۰	۱۵۰	۱
۲۲	۶۹۵	۲۰۰	۱
۲۳	۶۹۰	۱۵۰	۱
۳۱	۷۰۰	۱۰۰	۲
۳۲	۷۱۰	۱۰۰	۲

مخازن

شناسه	ارتفاع	الگو
۹	۱۹۰۰	

لوله‌ها

شناسه	گره ۱	گره ۲	طول	قطر	خشنودگی	تلفات کوچک	وضعیت
۱۱	۱۱	۱۲	۵۲۸۰	۱۴	۱۰۰	۰	باز
۱۲	۱۲	۱۳	۵۲۸۰	۱۰	۱۰۰	۰	باز
۲۱	۲۱	۲۲	۵۲۸۰	۱۰	۱۰۰	۰	باز
۲۲	۲۲	۲۳	۵۲۸۰	۱۲	۱۰۰	۰	باز
۳۱	۳۱	۳۲	۵۲۸۰	۶	۱۰۰	۰	باز
۱۲۱	۲۱	۳۱	۵۲۸۰	۸	۱۰۰	۰	باز
۱۲۲	۲۲	۳۲	۵۲۸۰	۶	۱۰۰	۰	باز
۲	۱۳	۲۳	۱۰۰۰	۱۲	۱۰۰	۰	باز
۳	۱۲	۲۲	۱۰۰۰	۱۲	۱۰۰	۰	باز
۴	۱۱	۲۱	۱۰۰۰	۱۲	۱۰۰	۰	باز
۵	۹	۱۰	۱۰۰۰	۱۲	۱۰۰	۰	باز

شیرها

شناسه	گره ۱	گره ۲	قطر	نوع	تنظیم	اتلاف جزئی
۶	۱۰	۱۱	۱۲	PRV	۲۰۰	۰

جدول ۴-۳ ابرپارامترهای الگوریتم شبکه-کیو عمیق

ابرپارامتر	مقدار
learning_rate	۰/۰۰۰۱
gamma	۰/۹۹
batch_size	۳۲
epsilon	۱
eps_dec	۰/۰۰۰۹۹
eps_end	۰/۰۱
n_time_steps	۵۰۰۰۰

فصل پنجم

نتایج و چالش‌ها

۵-۱ نحوه بررسی عملکرد عامل یادگیری تقویتی عمیق

تا به اینجا تمامی پارامترهای مهم الگوریتم و محیط شبیه‌سازی را معرفی کرده‌ایم. حال در این بخش به بررسی نتایج حاصل در زمان یادگیری می‌پردازیم. سه معیار زیر برای ارزیابی عملکرد عامل استفاده می‌شوند:

- مجموع پاداش‌ها در انتهای هر اپیزود
- میانگین فشار کل شبکه که بعد از یادگیری عامل محاسبه می‌شود
- بررسی فشار در برخی از نقاط که می‌دانیم باید چه مقداری داشته باشند ،

در بیشتر مسائلی که از یادگیری تقویتی برای حل آنها استفاده می‌شود، بررسی نمودار پاداش کافی است. اما در مسئله ما از آنجایی که محیط پویا است، نیاز به معیارهای سنجش دیگری نیز است. از آنجایی که اجرای الگوریتم‌های انتخاب‌شده برای پروژه بر روی شبیه‌ساز ایپانت وقت زیادی می‌برد، الگوریتم بین ۲۵۰ تا ۵۰۰ گام زمانی اجرا شده است. برای رسیدن به سیاست بهتر توسط عامل بهتر است که الگوریتم مدت زمان بیشتری اجرا شود.

نتایج مربوط به آزمایش در هنگام تعلیم، جمع‌آوری شده است. در هنگام تعلیم، در انتهای هر اپیزود [۱] شماره‌ی گام زمانی به همراه مقدار متغیر خواسته‌شده برای این نمودارها ذخیره شده است. برای تشکیل نمودارهایی مانند شکل ۵-۴ داده‌ها در قالب آرایه‌هایی از زوج‌های مقدار پاداش و گام زمانی دریافت شدند و سپس با کمک روش میانگین متحرک [۲] ، هموارسازی [۳] شدند. هموارسازی به این دلیل انجام شد که نوسانات زیاد، بررسی روند پیشرفت الگوریتم را از روی نمودارها سخت کرده بود. سپس تعدادی شبکه آب با همان ساختار اما با الگوی مصرف‌های متفاوت ساخته شد که دارای یک شیر آب اصلی متصل به منبع اصلی آب هستند. سپس مقدار بهینه فشار برای شیر اصلی هر یک به‌صورت دستی و با سعی و خطا به‌دست آمد، به‌طوری که هدف ما یعنی بهینه‌کردن فشار را برای ما برآورده سازد. سپس الگوریتم روی این شبکه‌ها اجرا شد و الگوریتم نیز به همان عدد بهینه برای کنترل فشار شبکه رسید که این کار به نوعی نحوه ارزیابی درست رفتار کردن الگوریتم است.

[1] Episode
[2] Moving Average
[3] Smooth

۵-۱-۱ خروجی از عملکرد

برای استفاده از عامل بوجود آمده، تنها کافیست عامل را پس از آموزش با فانکشن Save در یک مسیر ذخیره نمود و در گام بعدی با لود کردن آن، عامل را اجرا نمود. برای بررسی عملکرد عامل در طی آموزش نیز پاداش‌هایی که عامل در هر گام جمع کرده است در یک آرایه ذخیره می‌شود. سپس با استفاده از این آرایه نمودار پاداش جمع‌شده ساخته می‌شود که نشان‌دهنده یادگیری یا عدم یادگیری عامل است. اگر این نمودار به‌صورت شهودی صعودی باشد به معنای یادگیری عامل است. نمونه‌هایی از این نمودارها در بخش ۵-۲ آمده است.

۵-۲ نتایج و تست‌های بیشتر

در بخش قبلی نحوه خروجی گرفتن و بررسی عملکرد در این مسئله بیان شد. در این قسمت سعی داریم تا مشاهده‌های بیشتری که از الگوریتم داشتیم را نشان دهیم. دراین بخش ما چهار شبکه با اندازه‌های مختلف را مورد بررسی قرار می‌دهیم. چهار شبکه آب به نام‌های زیر هستند:

- SimpleNet1 [۸]
- Hanoi_CMH [۸]
- GES [۸]
- SimpleNet2 [۸]

این شبکه‌ها از وب‌گاه‌های مختلف پیدا شده‌اند و در برخی از موارد به آن‌ها شیرهای آب اضافه شده است تا بتوان از آن در مسئله استفاده کرد. سعی شده است با انتخاب این شبکه‌ها موارد مختلفی از جمله ناپایداری الگوریتم DQN در برخی از موارد، بررسی شود. برای شروع، نتایج شبکه SimpleNet1 را بررسی می‌کنیم.

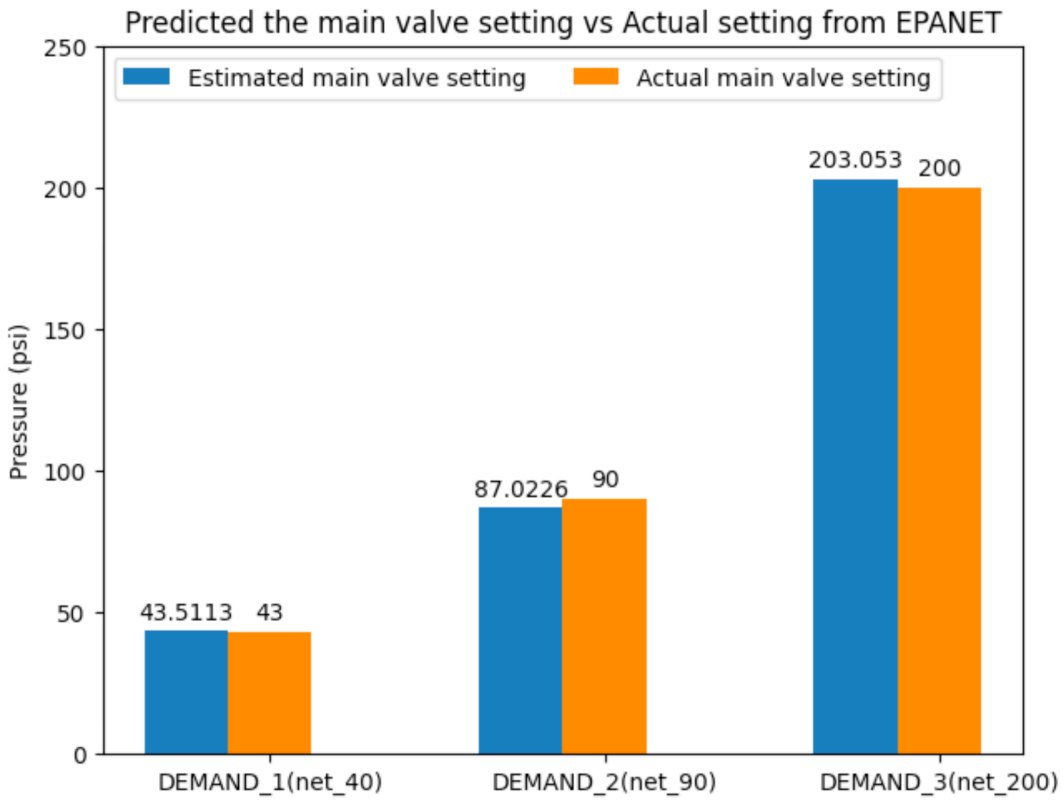
۵-۲-۱ شبکه SimpleNet1

روی این شبکه از الگوی مصرف‌های [1] مختلف استفاده کردیم تا نشان دهیم الگوریتم به ازای الگومصرف‌های مختلف موفق به یادگیری می‌شود و عمل مناسبی را انجام می‌دهد. همانطور که در شکل ۵-۲ مشاهده می‌شود، این شبکه آب به طور خاص یک ویژگی دارد که به ما کمک می‌کند تا الگوریتم را بهتر ارزیابی کنیم. این شبکه تنها دارای یک شیرآب است که به یک منبع طبیعی آب متصل است و فشار کل شبکه از طریق این شیر آب کنترل می‌شود. در همین راستا ما ابتدا، به‌صورت دستی و با سعی خطا توانستیم فشار بهینه شیر کنترل فشار را در نرم‌افزار EPANET پیدا کنیم. سپس به‌وسیله الگوریتم و پس از اجرای عامل روی محیط به‌صورت اتوماتیک نیز به آن عدد رسیدیم. سپس الگوی مصرف را عوض کردیم و باز به‌صورت دستی فشار مناسب شیر آب را در نرم‌افزار پیدا کردیم و دوباره عامل را اجرا کردیم و باز هم الگوریتم به عددی رسید که در حالت دستی و به‌صورت سعی و خطا به دست آمده بود و این نشان می‌دهد که الگوریتم به درستی کار می‌کند. برای مثال، در الگوی مصرف شماره یک، شیر فشار باید روی عدد ۴۰ تنظیم می‌شد که در شکل ۵-۲ ، نتایج آن نشان داده شده است. همچنین حالات دیگر نیز در شکل ۵-۳ و شکل ۵-۴ نشان داده شده است.

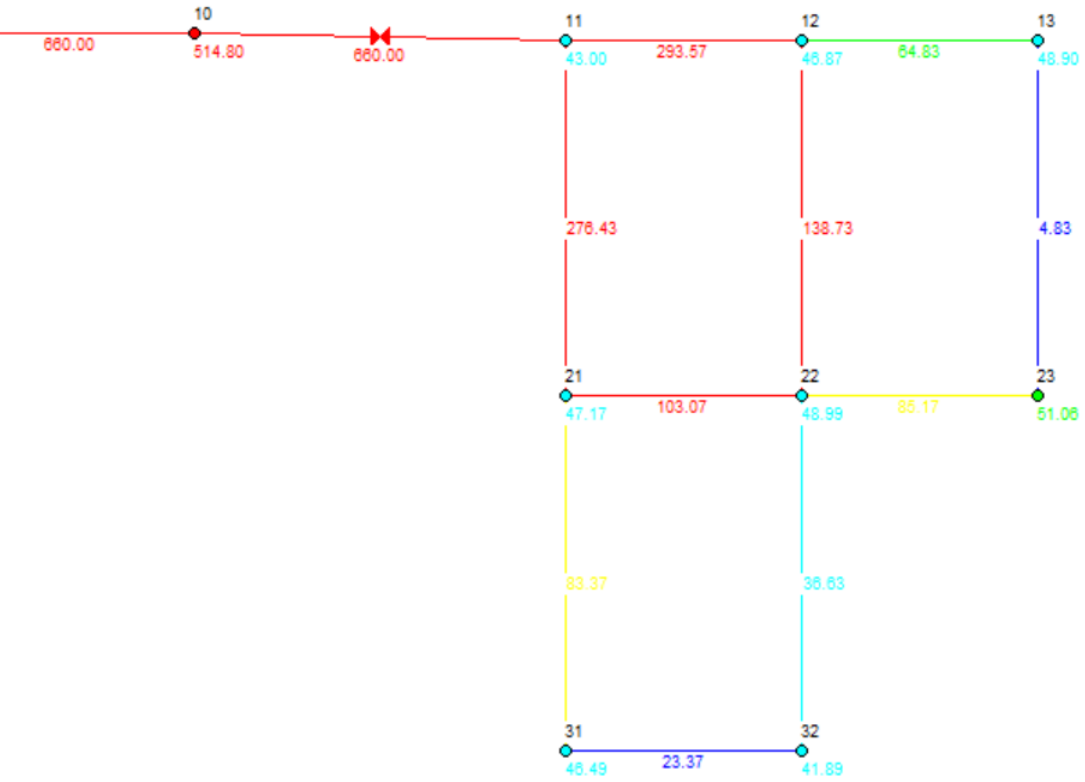
	الگوی مصرف یک	الگوی مصرف دو	الگوی مصرف سه
مقدار به دست آمده به‌صورت دستی	۴۳	۹۰	۲۰۰
مقدار بدست آمده به‌وسیله عامل	۴۳	۸۷	۲۰۳

جدول ۵-۱ فشار شیر اصلی انتخاب شده برای الگوی مصرف‌های مختلف شبکه SimpleNet1

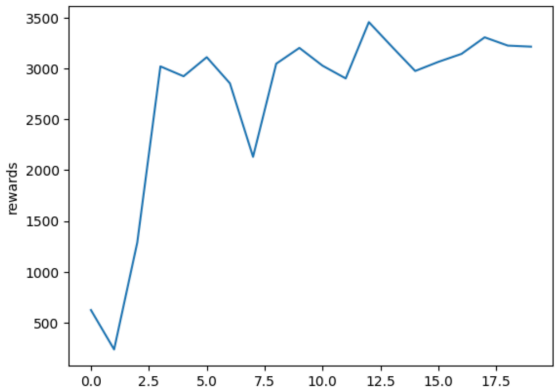
[1] Demand Pattern



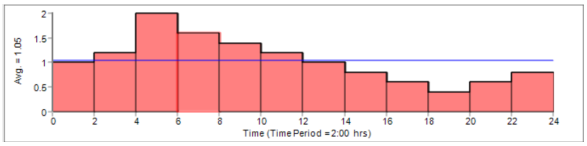
شکل ۵-۱ فشار شیر اصلی به دست‌آمده به‌صورت دستی و همچنین به‌وسیله عامل الگوریتمی، در الگوی مصرف‌های مختلف SimpleNet1



(آ) وضعیت شبکه آب وقتی شیر کاهش فشار آب اصلی روی عدد ۴۰ قرار گرفته است. برای شبکه SimpleNet1 با الگوی مصرف یک

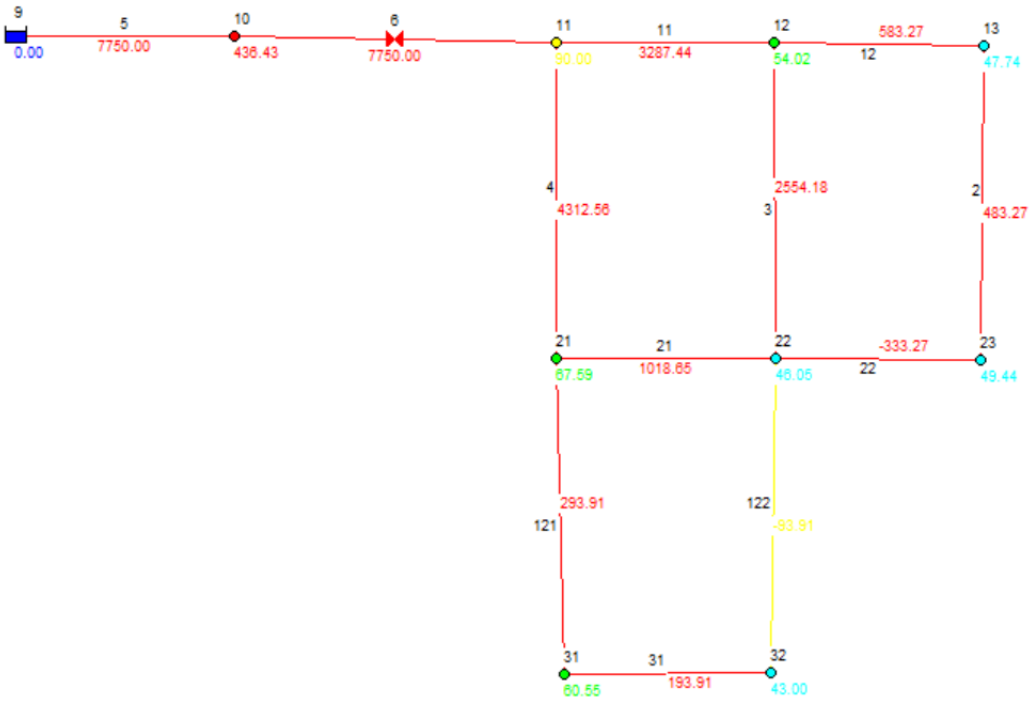


(ب) نمودار پاداش

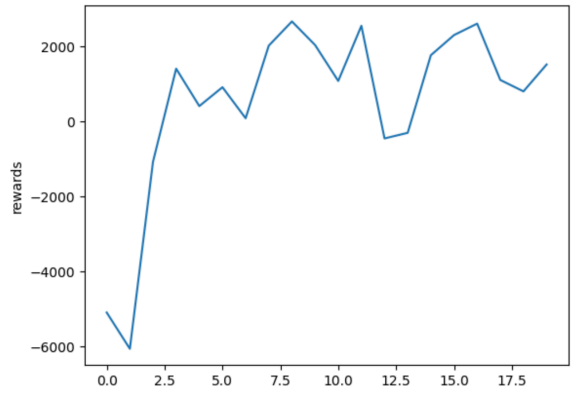


(ج)

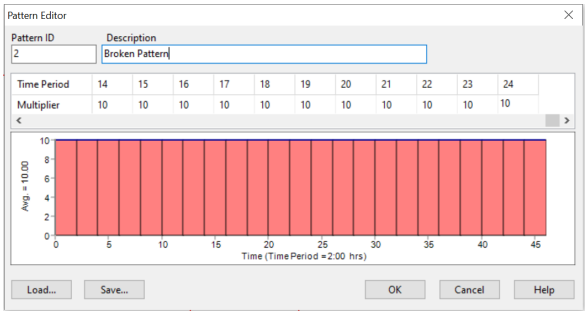
شکل ۵-۲ همانطور که می‌بینیم نمودار پاداش، روند تقریبا صعودی داشته و این نشان‌دهنده یادگیری عامل است. الگوی مصرف برای گره‌ها در شکل ج آمده است.



(آ) وضعیت شبکه آب وقتی شیر کاهش فشار آب اصلی روی عدد ۹۰ قرار گرفته است. برای شبکه SimpleNet1 با الگوی مصرف دو

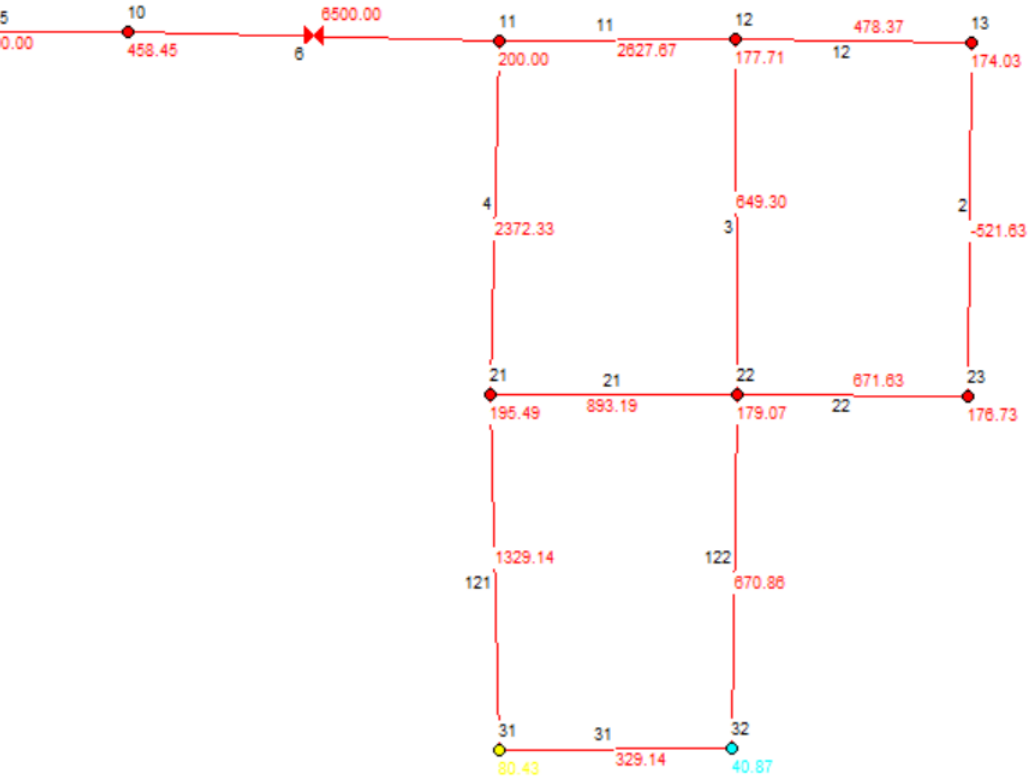


(ب) نمودار پاداش

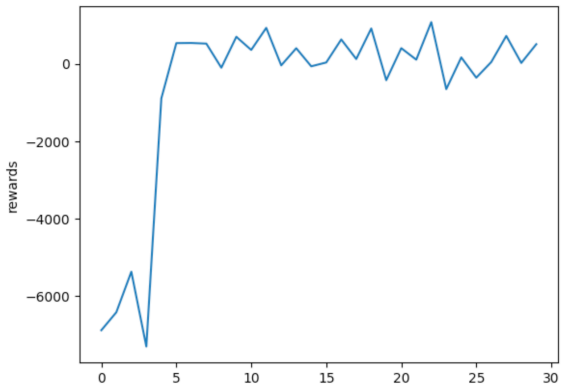


(ج)

شکل ۵-۳ همانطور که می‌بینیم پاداش روند تقریبا صعودی داشته و این نشان‌دهنده یادگیری عامل است. الگوی مصرف برای همه گره‌ها هم در شکل ۵-۴ آمده است.



(آ) وضعیت شبکه آب وقتی شیر کاهش فشار آب اصلی روی عدد ۲۰۰ قرار گرفته است. برای شبکه SimpleNet1 با الگوی مصرف سه



(ب) نمودار پاداش

شکل ۵-۴ همانطور که می‌بینیم پاداش روند تقریبا صعودی داشته و این نشان‌دهنده یادگیری عامل است.

۵-۲-۲ شبکه Hanoi_CMH

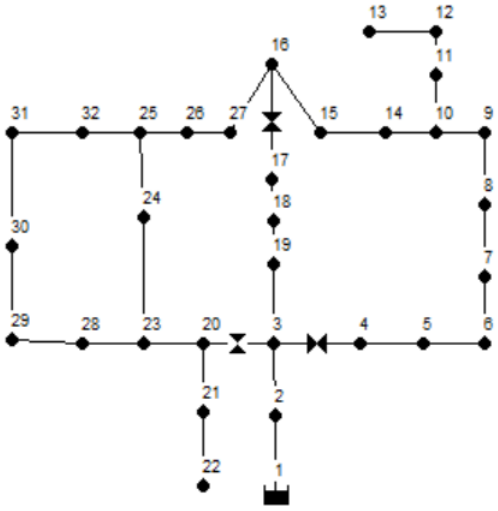
این شبکه از شبکه‌های نمونه‌ای است که Open Water Analytics [۱۰] به عنوان مثال معرفی کرده است. این شبکه در شکل ۵-۵ نشان داده شده است و همانطور که دیده می‌شود در آن سه شیرآب وجود دارد. نتایج آمده در شکل ۵-۶ نشان می‌دهد که این شبکه نیز توسط الگوریتم کنترل شده است.

۵-۲-۳ شبکه GES

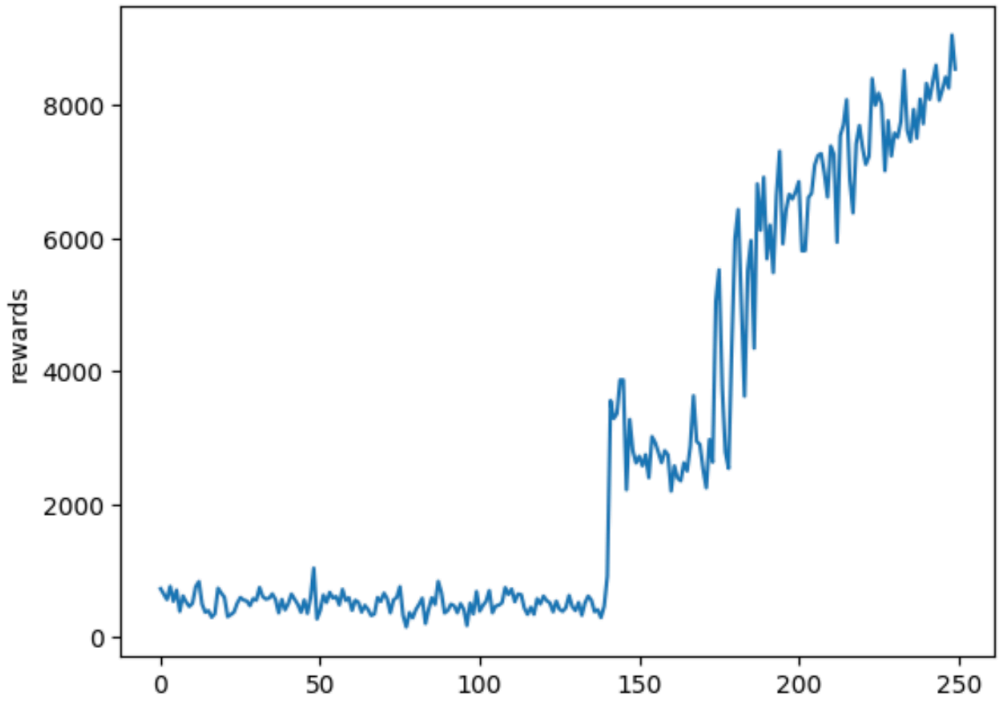
این شبکه نیز یک شبکه آب است که به آن شیرهای آب اضافه شده است تا بتوان از آن در الگوریتم استفاده کرد. در توپولوژی این شبکه در شکل ۵-۸ نشان داده شده است. نکته قابل توجه این شبکه این است که با وجود تقریبا صعودی بودن نمودار، این نمودار همچنان نویزی است. دلیل این اتفاق ماهیت الگوریتم DQN است که پایداری در طی آموزش ندارد. دلیل این اتفاق در بخش ۵-۳-۱-۱ توضیح داده شده است.

۵-۲-۴ شبکه SimpleNet2

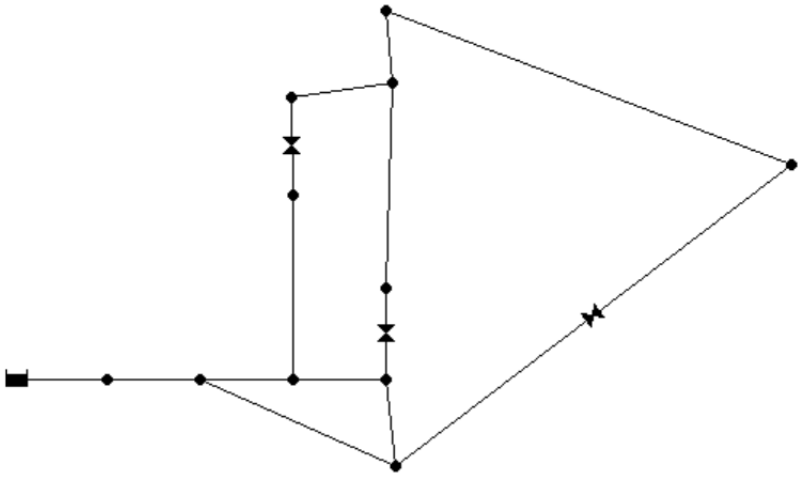
این شبکه نیز مانند شبکه آب قبلی یکی از مثال‌های OpenWaterAnalytics می‌باشد. در این شبکه نیز برای بررسی و اجرای الگوریتم سه شیر آب به آن اضافه شده است. این شبکه در شکل ۵-۹ نشان داده شده است. نتایج بدست آمده در شکل ۵-۱۰ نشان داده شده است. همانطور که در این نمودار مشاهده می‌شود پاداش بالایی از همان ابتدای یادگیری وجود دارد. همچنین نمودار تقریبا خطی است و عامل یادگیری بخصوصی ندارد. دلیل این اتفاق این است که شیرهای آب تاثیر زیادی روی این شبکه آب ندارند. این نمودار به خوبی نشان می‌دهد که شیرهای کنترل فشار در این شبکه به درستی استفاده نشده است و در جای درستی در توپولوژی شبکه استفاده نشده‌اند.



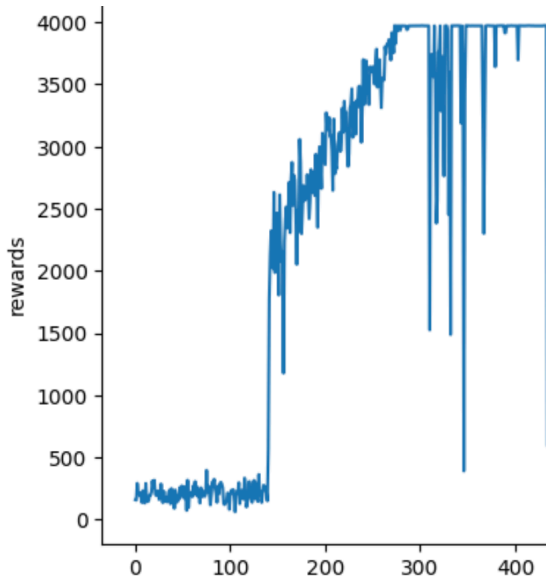
شکل ۵-۵ شبکه Hanoi_CMH



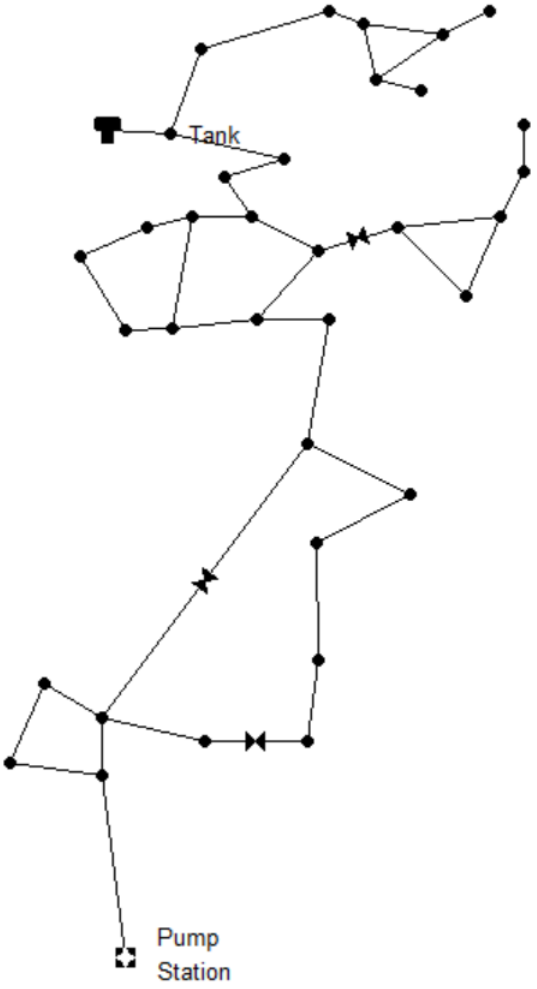
شکل ۵-۶ پاداش به دست آمده توسط الگوریتم در شبکه Hanoi_CMH



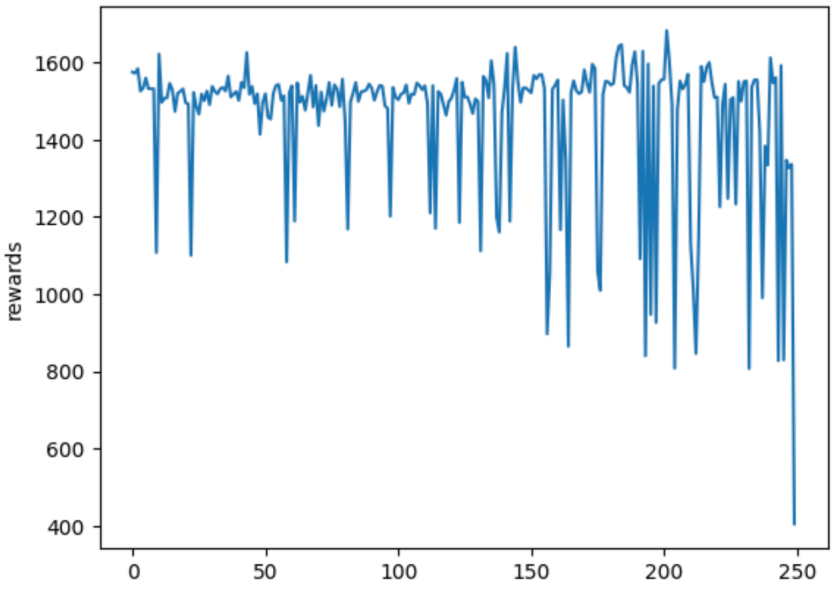
شکل ۵-۷ شمایی از شبکه GES



شکل ۵-۸ پاداش به دست آمده توسط الگوریتم در شبکه GES



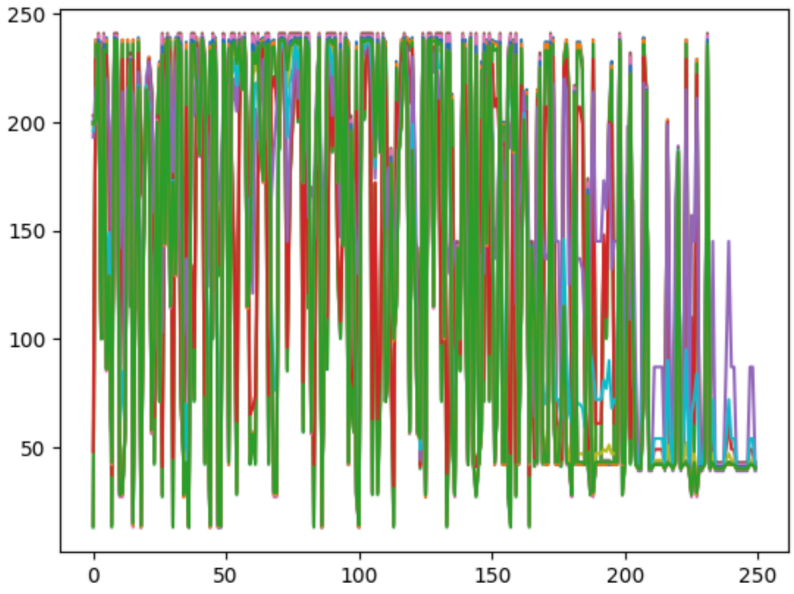
شکل ۵-۹ شبکه SimpleNet2



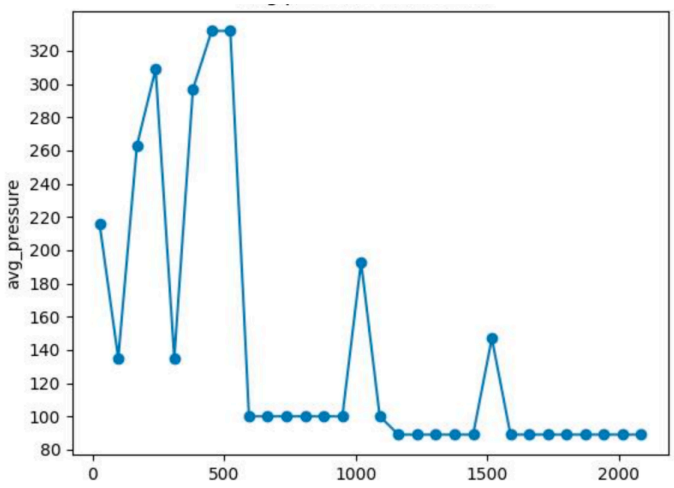
شکل ۵-۱۰ پاداش به دست آمده توسط الگوریتم در شبکه SimpleNet2

۵-۲-۵ بررسی نمودارهای پاداش و میانگین فشار شبکه

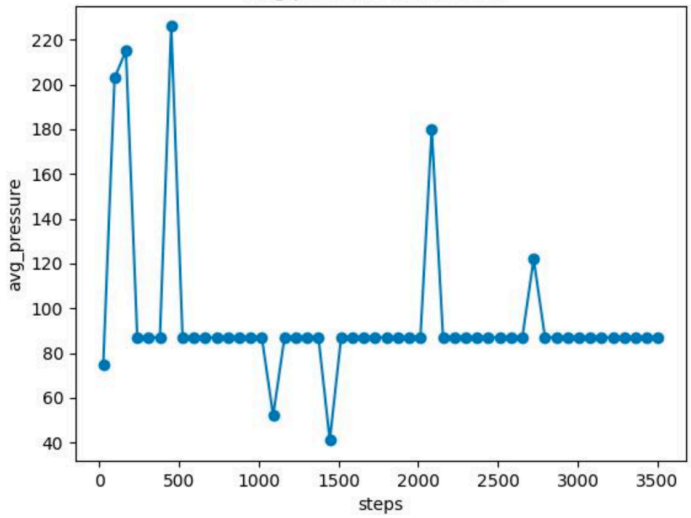
همانطور که در نمودارهای پاداش مشاهده کردیم، الگوریتم کیو عمیق پیشرفتی صعودی داشته و توانسته پس از گام‌هایی به کنترل بهینه با توجه به تابع پاداش مشخص شده، برسد. این عامل توانسته با بهره‌گیری از حافظه باز پخش تجربه و استفاده بهینه از داده‌های استخراج شده به سمت هدف مسئله حرکت کند. برای مسئله‌های بزرگتر با اجرای بیشتر این الگوریتم می‌توان به دقت بیشتری رسید. از طرفی دیگر همانطور که در شکل‌های زیر مشاهد می‌شود میانگین فشار شبکه در طول یادگیری، به یک عدد بهینه همگرا شده است. در شکل ۵-۱۲ برای شبکه SimpleNet1 سه الگوی مصرف قرار دادیم و فشار هر یک، همانطور که در شکل آمده است به یک عدد همگرا شدند که فشار میانگین بهینه شبکه است. در شکل ۵-۱۳ نیز که فشار میانگین برای شبکه Hanoi رسم شده است نیز این اتفاق به درستی افتاده است. همچنین در شکل ۵-۱۴ که برای شبکه GES است، از آنجایی که به یک عدد مشخص همگرا شده و نویز به مرور کم شده، به خوبی نشان‌دهنده موفقیت الگوریتم پس از آموزش است. برای نمونه نیز ما فشار برخی از گره‌های شبکه Hanoi را در شکل ۵-۱۱ رسم کرده‌ایم و همانطور که مشاهده می‌شود فشار گره‌ها در فرآیند یادگیری در بازه مناسب ۴۰ تا ۶۰ PSI همگرا شده است.



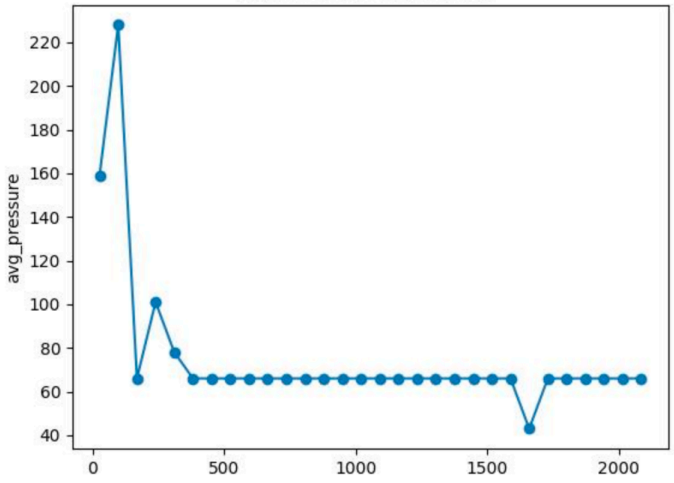
شکل ۵-۱۱ فشار گره‌های شبکه Hanoi طی یادگیری، محور افقی گام‌های یادگیری و محور عمودی فشار بر حسب PSI می‌باشد.



(آ) شبکه در وضعیت الگوی مصرف یک

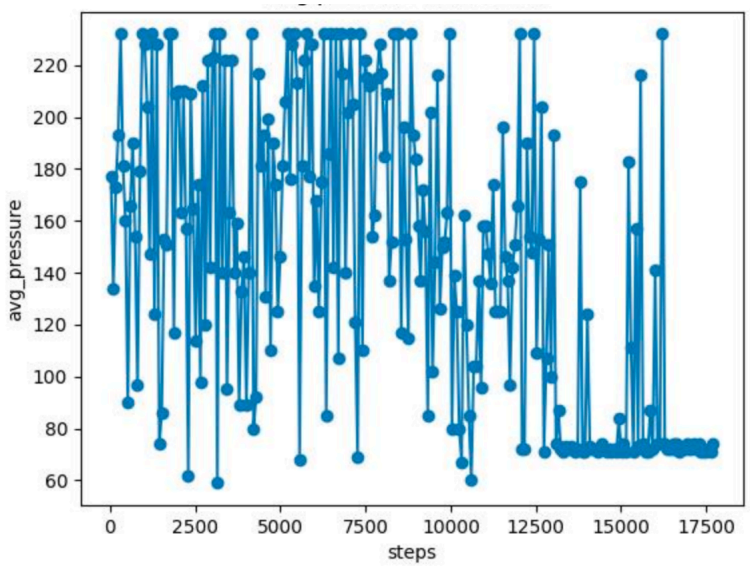


(ب) شبکه در وضعیت الگوی مصرف دو



(ج) شبکه در وضعیت الگوی مصرف سه

شکل ۵-۱۲ میانگین فشار شبکه SimpleNet1 در یک ساعت مشخص در فرآیند یادگیری، در سه الگوی مصرف مختلف



شکل ۵-۱۳ میانگین فشار شبک Hanoi در یک ساعت مشخص در فرآیند یادگیری

۵-۳ چالش‌ها و محدودیت‌های انجام پروژه

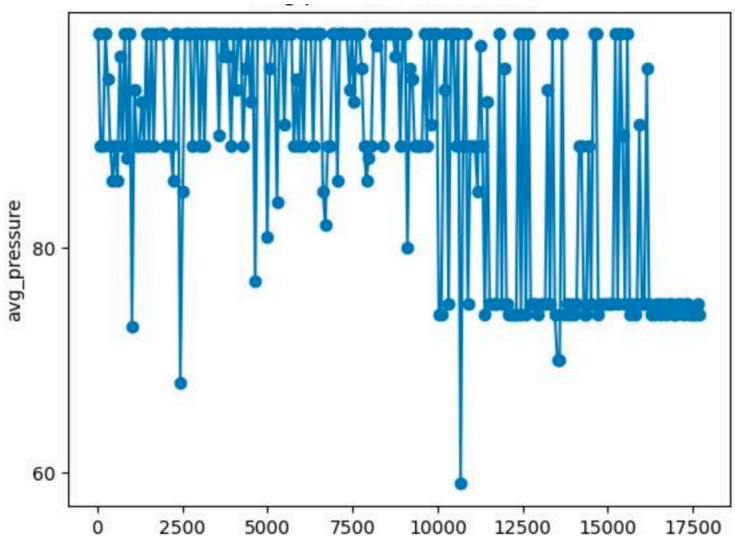
در فرآیند انجام پروژه و پیاده‌سازی الگوریتم‌های مدنظر، چالش‌های متعددی خودنمایی کردند. در این بخش، به بررسی تعدادی از این چالش‌ها پرداخته شده است. این چالش‌ها به طور کلی به سه دسته چالش‌های عمومی مسائل یادگیری تقویتی عمیق، چالش‌های نرم‌افزاری و چالش‌های سخت‌افزاری پروژه تقسیم می‌شوند.

۵-۳-۱ چالش‌های عمومی مسائل یادگیری تقویتی عمیق

در این بخش، به بررسی چالش‌های پیش‌آمده در عمل و هنگام طراحی و توسعه عامل یادگیری تقویتی مورد نظر می‌پردازیم.

۵-۳-۱-۱ عدم دسترسی به مدل محیط (یادگیری مستقل از مدل)

همانطور که در ؟؟ اشاره شد، یکی از چالش‌های اصلی اکثر مسائل مهم حوزهٔ یادگیری تقویتی عمیق، این است که به سازوکار فرآیند تصمیم‌گیری مارکوف و به عبارتی دیگر، مدل محیط دسترسی نداریم. از این رو، نیاز است به روش‌های یادگیری تقویتی مستقل از مدل را آورد. این چالش خصوصاً زمانی خود را بیشتر نشان می‌دهد که بدانیم در محیط‌هایی که مدلی از آن نداریم، باید همواره اکتشاف انجام دهیم و بدین ترتیب، روند یادگیری عامل هرگز متوقف نمی‌شود. در مسئله شبکه آب نیز این مشکل وجود دارد؛



شکل ۵-۱۴ میانگین فشار شبکه GES در یک ساعت مشخص در فرآیند یادگیری

ما اطلاعات دقیقی از محیط نداریم و در صورت تغییر اندک محیط نیز الگوریتم ما باید بتواند خود را به مرور با محیط تطبیق بدهد.

۵-۳-۱-۲ ناپایداری شبکه عصبی نسبت به قانون به‌روزرسانی

به کار بردن یک تابع تقریب (در اینجا، شبکه عصبی) می‌تواند کمک به یافتن ارزش حالاتی کند که تابحال ندیده‌ایم. اما برازش منحنی‌ای که صورت می‌گیرد، در عمل این چالش را ایجاد می‌کند که هنگام به‌روزرسانی ارزش یک حالت، ناخواسته بقیه حالات مجاور آن نیز به میزانی به‌روزرسانی می‌شوند. با در نظر گرفتن این موضوع، دو مورد ممکن است در فرآیند به‌روزرسانی‌های شبکه عصبی بوجود آید:

۱- روند به‌روزرسانی به گونه‌ای پیش برود که دسته مشخصی از حالات مجاور، همواره خود و بقیه حالات دسته را به‌روزرسانی کنند. این به‌روزرسانی‌ها منجر به واگرایی شبکه عصبی و ناپایداری آن خواهد شد.

۲- از آنجایی که به‌روزرسانی‌های یک حالت وابسته به به‌روزرسانی‌های حالات مجاور آن نیز می‌باشد، امکان دارد هرگز به تابع ارزش بهینه دست پیدا نکنیم و همواره میزانی خطا داشته باشیم.

در بخش بعد درباره روش‌هایی که برای مقابله با ناپایداری شبکه عصبی به کار گرفته شدند، بحث خواهد شد.

۵-۳-۱-۳ روش‌های مقابله با ناپایداری شبکه عصبی

برای آنکه بتوانیم چالش ناپایداری شبکه عصبی نسبت به قانون به‌روزرسانی را رفع کنیم، از دو روش بازپخش تجربه[1] و اهداف کیو ثابت[2] استفاده می‌کنیم.

در روش بازپخش تجربه، ما یک حافظه بازپخش [3] به‌صورت زیر تعریف می‌کنیم:

$$M = \{(s_t, a_t, r_{t+1}, s_{t+1})_1, (s_t, a_t, r_{t+1}, s_{t+1})_2, ...(s_t, a_t, r_{t+1}, s_{t+1})_T\} \quad (۱-۵)$$

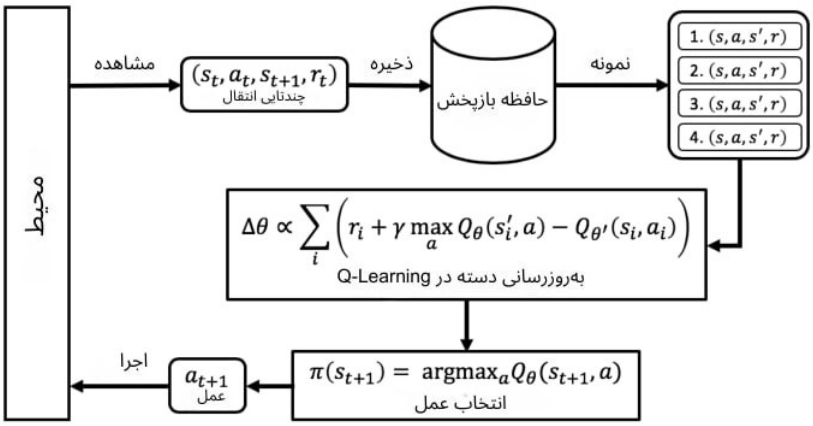
و سپس روند به‌روزرسانی را بدین صورت انجام می‌دهیم که در هر گام، به طور تصادفی یک تجربه از این حافظه انتخاب کرده و قانون به‌روزرسانی شبکه را با توجه به آن تجربه انجام می‌دهیم. این کار دو مزیت دارد:

۱- برخلاف قبل که با دیدن هر تجربه، به‌روزرسانی را انجام داده و تجربه را دور می‌ریزیم، در اینجا از هر تجربه بیش از یک مرتبه استفاده می‌شود. در نتیجه، تاثیر بیشتری در یافتن تابع ارزش مد نظر دارد. (مشابه این مزیت را در روش‌های دسته‌ای نیز داشتیم.)

۲- از آنجایی که عامل در هر اپیزود از اجرا، توالی‌ای از تجربه‌ها را بدست می‌آورد، از حالاتی از محیط عبور می‌کند که به طور زنجیره‌ای به هم وصل هستند. با اعمال قانون به‌روزرسانی از تجربه‌های انتخاب‌شده، ما وابستگی این زنجیره را از میان برده و در نتیجه، می‌توانیم به‌روزرسانی‌های پایدارتری انجام دهیم.

ما از این روش در الگوریتم خود استفاده کرده‌ایم و ناپایداری را تا حدی کاهش داده‌ایم. هرچند استفاده از این حافظه همچنان ضعف‌هایی دارد. از آنجایی که داده‌ها از این حافظه به‌صورت تصادفی انتخاب می‌شوند، شانس هر کدام برای انتخاب شدن یکسان است. اما می‌دانیم که برخی از داده‌های جمع‌آوری‌شده برای عامل از تازگی و اهمیت بیشتری برخوردار هستند. لذا نیاز به راه حلی است که احتمال انتخاب بیشتری به این نوع داده‌ها بدهد. در روش اهداف کیو ثابت، ما دو شبکه عصبی تعریف می‌کنیم: روی یک شبکه عصبی، مقادیر تابع ارزش را یافته و روی شبکه عصبی دیگر، به‌روزرسانی

[1] Experience replay
[2] Fixed Q-targets
[3] Replay memory



شکل ۵-۱۵ بازپخش تجربه

پارامترهای شبکه عصبی را براساس مقادیر تابع ارزش یافته‌شده در شبکه عصبی اول انجام می‌دهیم. این کار باعث می‌شود تا فرآیند یافتن مقدار خطا و به‌روزرسانی آن از هم جدا شده و به‌روزرسانی‌ها به‌صورت پایداری انجام شوند. به شبکه عصبی اول، شبکه هدف[1] و به شبکه عصبی دوم، شبکه اصلی[2] گفته می‌شود. پس از طی شدن تعدادی اپیزود مشخص، پارامترهای شبکه هدف را برابر با پارامترهای شبکه اصلی قرار می‌دهیم. به عبارتی دیگر، پارامترهای شبکه هدف را همواره با تاخیر به‌روزرسانی می‌کنیم.

۵-۳-۱-۴ فضای حالت

از مهمترین بخش‌های طراحی یک سامانه بر پایه الگوریتم‌های یادگیری تقویتی، طراحی فضای حالت مناسب است. فضای حالت طراحی‌شده باید تا حد امکان ابعاد کوچکی داشته باشد و همچنین بتواند تمامی اطلاعات لازم برای حل مسئله را در خود ذخیره کند. پس از بررسی‌هایی که صورت گرفت به این نتیجه رسیدیم که تنها با استفاده از مجموعه‌ای از فشارها در یک شبکه آبی می‌توان وضعیت محیط را تخمین زد. پس با استفاده از آن نیز می‌توان به خوبی عمل مناسب را پیدا نمود.

[1] Target network
[2] Main network

۵-۳-۲ چالش‌های نرم‌افزاری پروژه

۵-۳-۲-۱ محدودیت الگوریتم

علی‌رغم پیشرفت‌های چشمگیری که الگوریتم‌های یادگیری تقویتی عمیق در کاهش زمان لازم برای همگرایی به تابع ارزش بهینه و سیاست بهینه داشته‌اند، هنوز هم فرآیند آموزش نیازمند زمان بسیار زیادی برای به ثمرنشستن است. گاهی نیاز است عامل چند صد هزار اپیزود تجربه کسب کند تا بتواند به تدریج در مسیر همگرایی قرار گیرد. در خیلی از مسائل، از مدل‌هایی که قبلاً آموزش دیده‌اند و وظایفی را توانستند یاد بگیرند، استفاده می‌شود. گاهی نیز باید ابرپارامترها را آنقدر تغییر داد تا بتوان نتیجه معقولی گرفت. همه این موارد نشان می‌دهد که آموزش دادن عامل‌های یادگیری تقویتی عمیق، چقدر مستلزم صبر و حوصله و زمان کافی برای نتیجه گرفتن است. در این پروژه نیز تعدادی از این راهکارها برای زودتر به نتیجه رسیدن استفاده شده است. با این وجود، دور از ذهن نیست که همواره زمان بیشتر برای اجرا، می‌تواند منجر به بهتر شدن خروجی شود.

۵-۳-۲-۲ محدودیت ایپانت

یکی از مشکلاتی که باعث کند شدن حل مسئله و الگوریتم می‌شود این است که شبیه‌ساز ایپانت با تمام محبوبیت و پیاده‌سازی‌ای که دارد تنها از CPU برای پردازش‌ها استفاده می‌کند و این باعث می‌شود در محیط‌های بسیار بزرگ زمان اجرا بسیار طولانی شود.

۵-۳-۲-۳ عدم وجود فایل شبکه آب مناسب

همانطور که گفته شد برای ساخت محیط نیاز به فایل‌های شبکه آب است. این شبکه‌ها می‌توانند برگرفته از دنیای واقعی باشند یا به‌صورت دستی و تستی ساخته شده باشند. مشکل اساسی‌ای که وجود دارد این است که اکثر شبکه‌های آبی که در فضای اینترنت وجود دارد به نحوی است که از شیرهای فشار به ندرت در آن‌ها استفاده شده است. برای همین پیدا کردن شبکه‌ای که واقعی‌تر باشد و از شیرهای آب در آن استفاده شده باشد، چالش اصلی این پروژه است. در برخی از موارد نیاز به اضافه کردن شیرهای آب به شبکه‌های موجود بود زیرا ماهیت عمل‌های الگوریتم ما در این پروژه شیرهای آب هستند.

۵-۳-۳ محدودیت‌های سخت‌افزاری در این پروژه

در زمینه تحقیقات هوش مصنوعی امروزه، اغلب آزمایشات توسط شرکت‌های بزرگی همچون گوگل با بهره‌گیری از صدها واحد پردازشی گرافیکی[1] و واحدهای پردازش تنسوری[2] اجرا می‌شوند. این شرایط موجب می‌شود تا مسیر پیشرفت و رقابت در عرصه هوش مصنوعی بیشتر بر روی افراد و شرکت‌هایی با توان مالی و امکانات مناسب برای سرمایه‌گذاری در این زمینه متمرکز شود. همچنین، لازم به ذکر است که نرم‌افزار ایپانت، با محدودیت‌های سخت‌افزاری مواجه است. زیرا از پردازنده گرافیکی به درستی بهره نمی‌برد. این امر به نوعی یک محدودیت سخت‌افزاری محسوب می‌شود که در پروژه باید به آن توجه شود.

۵-۴ خلاصه

در این فصل به جزئیات مهم در مورد ورودی الگوریتم، تابع پاداش و مقادیر ابرپارامترها پرداختیم. ورودی الگوریتم شامل مجموعه‌ای از اعداد است. تابع پاداش نیز از روی فشار برخی از گره‌های شبکه بدست آمد. در انتها نیز به بررسی نتایج بدست آمده از عملکرد عامل پرداختیم. در نهایت نیز به چالش‌های پروژه پرداختیم و با چالش‌های پیاده‌سازی، نرم‌افزاری و سخت‌افزاری آن آشنا شدیم.

[1] Graphics Processing Unit (GPU)

[2] Tensor Processing Unit (TPU)

فصل ششم

جمع‌بندی و پیشنهادها

۶-۱ جمع‌بندی

در این پروژه، یک کنترل‌کننده شبکه آب طراحی شد که به وسیله آن بتوان بدون دانستن اطلاعاتی از ساختار محیط، میانگین فشار لوله‌های آب را به طوری کم کرد که هیچ یک از گره‌های تقاضا کمبود فشار نداشته باشند و همچنین کم‌کردن فشار کلی در شبکه باعث کنترل نشتی‌ها و حفظ آب می‌شود.

فرایند کار بدین شکل است که عامل، فشارهای برخی از نقاط شبکه آب را دریافت می‌کند و با استفاده از مدل آماری که به وسیله آن آموزش داده شده است، اعمالی را انجام می‌دهد. این اعمال مجموعه‌ای از فشارها هستند که برای شیرهای کاهش فشار تعیین شده است تا فشار را در شبکه کم کنند. رفتار عامل با کمک تابع پاداش انتخاب می‌شود.

پس از آن، پیاده‌سازی الگوریتم شبکه-کیو عمیق انجام شد. در ابتدا آزمایش‌هایی برای آشنایی با خصوصیات این الگوریتم انجام شد و سپس پیاده‌سازی‌های آن‌ها برای استفاده در محیط شبیه‌ساز ایپانت تغییر یافت. از جزئیات مهم پیاده‌سازی‌ها می‌توان به استفاده از حافظه بازپخش تجربه برای الگوریتم شبکه-کیو عمیق اشاره کرد. همچنین برای استفاده بهتر و ماژولارتر، محیط الگوریتم در پایتون پیاده‌سازی شد که به نرم‌افزار ایپانت متصل می‌شود.

پس از انجام پیاده‌سازی‌های مربوط به الگوریتم، تنظیمات شبیه‌ساز انجام شد. در انتها نمودارهای پاداش، میانگین فشار شبکه و فشار در هر گره مورد بررسی قرار گرفت.

پیاده‌سازی‌های این پروژه در وبگاه گیتاب [۸] قرار گرفته است. تمامی پیاده‌سازی‌ها با زبان برنامه‌نویسی پایتون انجام شده‌است. چارچوب استفاده‌شده برای پیاده‌سازی، تنسورفلو [1] می‌باشد.

۶-۲ پیشنهادها و کارهای آینده

از مهم‌ترین اقداماتی که برای بهبود این پروژه می‌توان انجام داد، حل مسئله شکل‌دهی پاداش[2] در محیط شبیه‌ساز و هنگام آموزش عامل یادگیری تقویتی عمیق می‌باشد. شکل‌دهی پاداش عبارت است از تعیین هر گونه شیوه پاداش‌دهی که منجر به یادگیری سریع‌تر یک وظیفه توسط عامل شود. برای مثال عدد منفی‌ای که در فرمول پاداش است را می‌شود با اعداد دیگری جایگزین و آزمایش کرد. همچنین می‌توان

[1] Tensorflow

[2] Reward shaping

شرط‌های بیشتری را به تابع پاداش اضافه کرد، برای مثال می‌توان به ازای تمام شدن منابع آب پاداش منفی در نظر گرفت. به طور کلی تعریف مشخصی از پاداش در این مسئله وجود ندارد و با آزمون و خطا می‌توان به تابع پاداش بهتری رسید. تابع پاداش نقش بسزایی در بهبود یادگیری در مسئله دارد.

بهبود دیگری که می‌تواند در این پروژه انجام شود تغییر در تعریف حالت است. تعاریف متفاوتی می‌تواند برای فضای حالت در نظر گرفته شود، برای مثال می‌توان ویژگی‌های دیگری مانند زمان را به عنوان ورودی به مسئله داد. یکی دیگر از بهبودهایی که می‌توان به پروژه داد، استفاده از مدل‌های عصبی پیچیده‌تر است که ممکن است به سرعت و دقت یادگیری کمک کنند.

در این پروژه و برای سهولت کار، فضای عمل به صورت گسسته و دارای تنها اعمال محدود تعریف شده است. پرواضح است که با گسترش دادن فضای عمل و افزایش تعداد اعمالی که می‌توان انجام داد، به عاملی دست پیدا خواهیم کرد که منعطف‌تر عمل می‌کند. بنابراین، گسترش فضای عمل مسئله نیز می‌تواند منجر به بهبود این پروژه شود.

کتاب‌نامه

[1] Mosetlhe, Thapelo Cornelius. Model-free pressure control in water distribution networks. Ph.D. thesis, Université Paris-Saclay; Tshwane University of Technology, 2021.

[2] Nicolini, Matteo. Optimal pressure management in water networks: increased efficiency and reduced energy costs. in 2011 Defense Science Research Conference and Expo (DSR), pp. 1–4. IEEE, 2011.

[3] Belfadil, Anas, Modesto, David, et al. Drl-epanet: Deep reinforcement learning for optimal control at scale in water distribution systems. in Deep Reinforcement Learning Workshop NeurIPS 2022, 2022.

[4] Sutton, Richard S and Barto, Andrew G. Reinforcement learning: An introduction. MIT press, 2018.

[5] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.

[6] Agency, United States Environmental Protection. Epanet, 2014.

[7] Klise, Katherine. Usepa/wntr: An epanet compatible python package to simulate water networks, 2022.

[8] Lookzadeh, Hossein. hoseinlook/wdn: Implementation of the dqn algorithm in the epanet environment using python, 2023.

[9] Schaul, Tom, Quan, John, Antonoglou, Ioannis, and Silver, David. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.

[10] Openwateranalytics: The water distribution system hydraulic and water quality analysis toolkit, 2018.