به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس: بهینه‌سازی و یادگیری توزیع‌شده

(پیاده‌سازی الگوریتم یادگیری تقویت شده)

پروژه شماره ۳

نام و نام خانوادگی : امیرحسین پورداود

شماره دانشجویی : ۸۱۰۱۰۱۱۲۰

بهمن ماه ۱۴۰۲

فهرست

(۱) مقدمه

پیش‌بینی می‌شود که شبکه‌های نسل ششم (6G) طیف وسیعی از برنامه‌های کاربردی مانند حمل‌ونقل هوشمند، واقعیت مجازی/افزوده مانند متاورس و غیره را ارائه دهند. این برنامه‌ها نیاز به ارتباط با قابلیت‌های فوق‌العاده مطمئن و کم تأخیر دارند، بنابراین برای برآورده کردن این الزامات، ارتباطات فوق‌العاده مطمئن و کم تأخیر (URLLC[1]) به عنوان یک سرویس اساسی در شبکه‌های 5G و 6G شناخته شده است.

در این پروژه، مسئله به حداکثر رساندن نرخ کل سیستم (STR[2]) بیان شده است، که در آن شکل دهی و ترکیب بردارها در BS، قدرت انتقال کاربران UL، تضعیف دامنه، و جابجایی فاز STAR-RIS ها به طور مشترک بهینه می شوند. مسئله بیان شده یک مسئله پیوسته غیر محدب است. برای دستیابی به یک راه حل بهینه، می توان از روش های جستجوی جامع استفاده کرد. همچنین، روش‌های مبتنی بر بهینه‌سازی تکراری برای به دست آوردن راه‌حل‌های زیر بهینه، پیچیدگی محاسباتی بالایی دارند و ممکن است از حد بهینه فاصله داشته باشند. برای مقابله با این چالش، اخیراً روش‌های DRL توجه قابل توجهی را به خود جلب کرده‌اند.

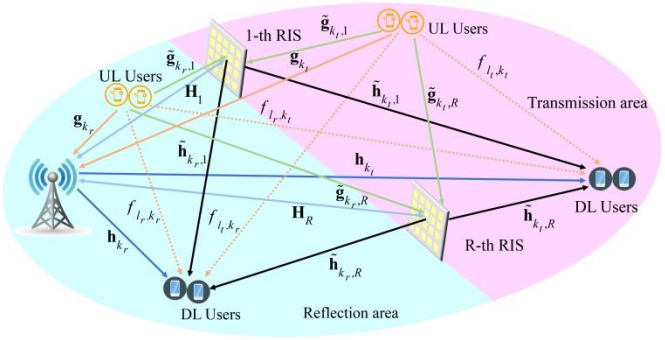
[1] ultra-reliable and low latency communication
[2] System Total Rate

(۲) شرح مسئله و مدل سازی

در این بخش مدل‌سازی مسئله و روابط حاکم بر آن ها را بیان کرده و مسئله نهایی را شرح می‌دهیم و در ادامه به بررسی محیط یادگیری تقویتی حاکم بر این مسئله و نحوه حل آن بیان می‌کنیم.

۱.۲. سیستم مدل و مسئله

همانطور که از شکل زیر مشخص است، مدل حاکم بر این سیستم، شامل کاربران در دو حالت down-link و up-link و یک ایستگاه مرکزی پایه برای ارسال و دریافت سیگنال کاربران می‌باشد. در این سیستم برای جلوگیری از تداخل و بهترشدن ارتباطات و به خصوص برای دستیابی به کمترین تاخیر و اطمینان در شبکه که هدف اصلی این سیستم مدل است، چند STAR-RIS در محیط برای دستیابی به این موضوع قرار داده شده است که در ادامه به نحوه عملکرد و تاثیر آن ها و روابط حاکم بر آن ها میپردازیم.



ما یک شبکه FD[1] با کمک چند STAR-RIS را برای ارائه خدمات URLLC به کاربران K-DL[2] و L-UL[3] در نظر می‌گیریم. همانطور که در شکل بالا نشان داده شده است، شبکه در نظر گرفته شده از یک BS با N آنتن تشکیل شده است که می تواند به طور همزمان دریافت و ارسال کند. همچنین، به تعداد R، STAR-RIS وجود دارد که به BS در ارائه خدمات URLLC کمک می کند.

هر کاربر DL و UL به یک آنتن مجهز است و q'th STAR-RIS دارای M_q عنصر برای انتقال و انعکاس همزمان سیگنال‌ها است. فرض بر این است که پوشش BS به دو ناحیه مجزا تقسیم می شود:

۱- ناحیه انتقال (TA) و ۲- ناحیه بازتاب (RA).

---

[1] Full Duplex
[2] Down link
[3] Up link

عناصر STAR-RIS سیگنال های دریافتی را با تقسیم کردن آنها به دو جزء دستکاری می کنند. اولین جزء، سیگنال منعکس شده، به طور هوشمندانه به سمت ناحیه بازتاب هدایت می شود. در همین حال، جزء دوم، سیگنال ارسالی، به منطقه انتقال ارسال می شود. این فرآیند پارتیشن بندی با تنظیم جریان های الکتریکی و مغناطیسی عناصر STAR-RIS، با استفاده از ضرایب انتقال و بازتاب به دست می آید. این پیکربندی، تنظیم مستقل سیگنال های ارسالی و منعکس شده را تسهیل می کند و در نتیجه عملکرد و پوشش کلی سیستم را به طور قابل توجهی افزایش می دهد.

ضرایب انتقال و انعکاس هر STAR-RIS بصورت جداگانه به شکل زیر نوشته میشود:

$$\mathbf{\Theta}_q^{\bar{b}} = \text{diag}\{\eta_{q,1}^b e^{j\vartheta_{q,1}^b}, \eta_{q,2}^b e^{j\vartheta_{q,2}^b}, \cdots, \eta_{q,M_q}^b e^{j\vartheta_{q,M_q}^b}\}$$

تمامی کاربران UL و DL بصورت یکنواخت در هر دو ناحیه توزیع شده اند. فرض کنید سیگنال ارسالی از BS به کاربران DL بصورت زیر تعریف میشود که در آن w، بردار شکل‌دهی پرتو و s، سمبل های ارسالی هر کاربر می‌باشد:

$$\mathbf{x}^{\text{DL}} = \sum_{\forall b \in \mathcal{B}} \sum_{k_b=1}^{K_b} \mathbf{w}_{k_b} s_{k_b}$$

همچنین فرض میکنیم سیگنال ارسالی کاربران UL نیز بصورت زیر می‌باشد که در آن ρ توان و q سمبل های ارسالی کاربر می‌باشد:

$$x_{l_b}^{\text{UL}} = \sqrt{\rho_{l_b}} q_{l_b}$$

با توجه به سیگنال های ارسالی و دریافتی تعریف شده، سیگنال دریافتی در هر کاربر DL با توجه به ضرایب کانال های موجود که در شکل مدل سیستم نیز مشخص است، بصورت زیر نوشته میشود، که در آن عبارت های دوم به بعد نشان دهنده نویز و تداخل از سمت سایر کابران DL و UL می‌باشد:

$$\begin{aligned} y_{k_b}^{\text{DL}} = &(\mathbf{h}_{k_b}^H + \sum_{q=1}^{R} \tilde{\mathbf{h}}_{k_b,q}^H \mathbf{\Theta}_q \mathbf{H}_q) \mathbf{w}_{k_b} s_{k_b} + (\mathbf{h}_{k_b}^H + \sum_{q=1}^{R} \tilde{\mathbf{h}}_{k_b,q}^H \mathbf{\Theta}_q \mathbf{H}_q) \\ &\times \sum_{i=1, i\neq k_b}^{K} \mathbf{w}_i s_i + \sum_{l_b=1}^{L_b} (f_{l_b,k_b} + \sum_{q=1}^{R} \tilde{\mathbf{h}}_{k_b,q}^H \mathbf{\Theta}_q \tilde{\mathbf{g}}_{l_b,q}) x_{l_b}^{\text{UL}} \\ &+ \sum_{o\in\mathcal{B}\setminus\{b\}, l_o=L_o+1}^{L} (f_{l_o,k_b} + \sum_{q=1}^{R} \tilde{\mathbf{h}}_{k_b,q}^H \mathbf{\Theta}_q \tilde{\mathbf{g}}_{l_o,q}) x_{l_o}^{\text{UL}} + n_{k_b}^{\text{DL}}, \end{aligned}$$

همچنین مجموع سیگنال های دریافتی کاربران UL در ایستگاه پایه را میتوان بصورت زیر نوشت:

$$\mathbf{y}^{\mathrm{UL}}=\sum_{\forall b\in\mathcal{B}}\sum_{l_b=1}^{L_b}(\mathbf{g}_{l_b}+\sum_{q=1}^{R}\mathbf{H}_q^H\mathbf{\Theta}_q^b\tilde{\mathbf{g}}_{l_b,q})x_{l_b}^{\mathrm{UL}}+\mathbf{H}^{\mathrm{SI}}\mathbf{x}_l^{\mathrm{DL}}+\mathbf{n}^{\mathrm{UL}},$$

حال میزان SNR دریافتی در هر کاربر DL و ایستگاه پایه که نشان دهنده توان سیگنال دریافتی در هر کاربر نسبت به توان سیگنال های دریافتی تداخلی و نویز می‌باشد را محاسبه می‌کنیم و در ادامه با استفاده از آن، نرخ ارسال و دریافت را محاسبه می‌نماییم:

$$\gamma_{k_b}^{\mathrm{DL}} = \frac{|\bar{\mathbf{h}}_{k_b}^H\mathbf{w}_{k_b}|_2^2}{I_{k_b}^{\mathrm{DL}}} \qquad \gamma_{l_b}^{\mathrm{UL}} = \frac{\rho_{l_b}|\mathbf{u}_{l_b}^H\bar{\mathbf{g}}_{l_b}|_2^2}{I_{l_b}^{\mathrm{UL}}}$$

نرخ دریافت هر کاربر DL بصورت زیر محاسبه میشود که در آن احتمال خطای بازیابی سمبل ها برای قابلیت اطمینان و همچنین کمترین تاخیر(ζ) نیز برای دستیابی به هدف URLLC قرار داده شده است:

$$R_{k_b}^{\mathrm{DL}} = W[\log_2(1+\gamma_{k_b}^{\mathrm{DL}}) - \sqrt{\frac{V_{k_b}^{\mathrm{DL}}}{L}}Q^{-1}(\zeta)\log_2 e],$$

$$R_{l_b}^{\mathrm{UL}} = W[\log_2(1+\gamma_{l_b}^{\mathrm{UL}}) - \sqrt{\frac{V_{l_b}^{\mathrm{UL}}}{L}}Q^{-1}(\zeta)\log_2 e],$$

علاوه بر این، تأخیر در شبکه دسترسی رادیویی را می توان با نسبت اندازه بسته و نرخ داده تعیین کرد که به صورت زیر نشان داده شده است.

$$T_u^{\mathrm{UL/DL}} = \frac{D_u}{R_u^{\mathrm{UL/DL}}},$$

در نتیجه، به حداقل رساندن مشکل تأخیر را می توان به عنوان حداکثر کردن مشکل نرخ کل به صورت زیر فرموله کرد:

$$\begin{aligned}
\mathcal{P}_1 : & \max_{\mathbf{w}_{k_b},\mathbf{u}_{l_b},\rho_{l_b},\mathbf{\Theta}_q} \sum\nolimits_{\forall b\in\mathcal{B}}\Big[\alpha\sum\nolimits_{k_b=1}^{K_b}R_{k_b}^{\mathrm{DL}}+(1-\alpha)\sum\nolimits_{l_b=1}^{L_b}R_{l_b}^{\mathrm{UL}}\Big] \\
\text{s.t. } & \mathrm{C1}: T_{k_b}^{\mathrm{DL}} \le \widehat{T}_{k_b}^{\mathrm{DL}}, \forall b\in\mathcal{B}, \forall k_b\in\{1,\cdots,K_b\}, \\
& \mathrm{C2}: T_{l_b}^{\mathrm{UL}} \le \widehat{T}_{l_b}^{\mathrm{UL}}, \forall b\in\mathcal{B}, \forall l_b\in\{1,\cdots,L_b\}, \\
& \mathrm{C3}: \sum\nolimits_{k=1}^{K}\|\mathbf{w}_k\|_2^2 \le P_{\mathrm{BS}}^{\max} \\
& \mathrm{C4}: \rho_l \le P_l^{\max}, \forall l\in\{1,2,...,L\} \\
& \mathrm{C5}: 0\le\vartheta_{q,i}\le 2\pi, \forall q\in\{1,...,R\}, i\in\{1,...,M_q\}, \qquad (6)
\end{aligned}$$

بنابراین، محدودیت C1 و C2، حداکثر نیاز تاخیر کاربر DL و UL را تضمین می کنند. علاوه بر این، C3 و C4 محدودیت‌هایی را بر حداکثر توان انتقال در BS و کاربر UL دوم اعمال می‌کنند و در نهایت، C5 محدودیتی را بر روی مقدار تغییر فاز اعمال می کند که باید در محدوده ۰ تا ۳۶۰ درجه باشد. مسئله بهینه سازی P1 غیر محدب است و به طور کلی حل بهینه آن چالش برانگیز است. از این رو، ما یک الگوریتم یادگیری تقویتی را برای حل موثر این مشکل اتخاذ می کنیم.

۲.۲. محیط یادگیری تقویتی

در این بخش یک الگوریتم RL برای حل این مسئله ارائه میدهیم. مسئله P1 را می توان به عنوان یک فرآیند تصمیم گیری مارکوف فرموله کرد که با (S; A; T; R; λ) نشان داده شده است. در اینجا، S مجموعه ای از حالات[1] را نشان می دهد، A نشان دهنده مجموعه اقدامات[2] موجود برای عامل در هر حالت، T نشان دهنده انتقال احتمال از حالت فعلی s به حالت بعدی s_0 است، R پاداشی[3] است که برای ارزیابی عملکرد حالت کنونی استفاده می شود. λ یک ضریب نزول در محدوده (۱، ۰) است که وزن پاداش های فوری و آینده را متعادل می کند.

برای پرداختن به حل P1، سیستم ارتباطی اشاره شده را به عنوان محیطی در نظر می گیریم که در آن متغیرهای تصمیم گیری مسئله P1 توسط عامل تعیین می شوند. در اینجا ما یک محیط، مطابق مسئله بهینه سازی نوشته شده بالا ایجاد میکنیم که شامل یک تابع هدف برای بیشینه شدن و محدودیت هایی برای توان و متغیر ها ایجاد میکند.

- **اکشن ها:**

در این مسئله، اکشن های ما، همان متغیر های مسئله بهینه سازی می‌باشند که سعی در بهینه سازی و یافتن بهترین متغیر (اکشن) برای بیشینه کردن تابع هدف می‌باشیم.

$$a_t = \left[\{\mathbf{w}_k\}_{k=1}^{K}, \{\mathbf{u}_l\}_{l=1}^{L}, \{\rho_l\}_{l=1}^{L}, \{\nu_{q,i}\}_{q=1,i=1}^{R,M_q}\right]$$

بعد فضای اکشن ها، به ترتیب متغیر های بالا، بصورت زیر میباشد:

[(Re{w_k}: **N_t * K**, Im{w_k}: **N_t * K**), (Re{u_L}: **N_t * L**, Im{u_L}: **N_t * L**),

ρ: L, v: **2 * R * M**]

[1] State
[2] Action
[3] Reward

- **حالت ها:**

در این مسئله، حالت ها را به گونه ای باید قرار دهیم که نشان دهنده محیط باشد. در مسئله های مخابراتی معمولا کانال ها را بعنوان حالتی از محیط درنظر میگیرند که از سک توزیع خاص نشات گرفته میشود، اما در اینجا با توجه به رندوم بودن این حالت ها نیاز داریم که این حالت ها از اکشن نیز تاثیر بپذیرند و هر حالت بر روی اکشن بعدی نیز تاثیر گذار باشد که مسئله MDP وجود داشته باشد، بنابراین علاوه بر حالت های ذکر شده، ما مقدار تابع هدف که متاثر از اکشن و کانال ها می‌باشد را نیز بعنوان یک حالت در نظر میگیریم.

$$\mathcal{S}=\left[\left\{\{\mathbf{h}_{k_b},\tilde{\mathbf{h}}_{k_b,q},\mathbf{H}_q,f_{l_b,k_b},\tilde{\mathbf{g}}_{l_b,q}\},\forall k_b,\forall l_b,\forall q\right\},\mathrm{STR}\right]$$

بعد فضای حالت ها، به ترتیب متغیر های بالا، بصورت زیر میباشد:

[h: $\mathbf{N_t * K}$, g: $\mathbf{N_t * L}$, h_s: $\mathbf{M * R * K}$, g_s: $\mathbf{M * R * L}$, H: $\mathbf{M * N_t * R}$,

f: $\mathbf{L * K}$, STR: $\mathbf{1}$]

- **پاداش:**

در این مسئله ما مقادیر پاداش را با توجه به ارضا شدن محدودیت ها، ضریبی از تابع هدف قرار میدهیم. به دلیل اینکه خروجی actor ما، بین صفر و یک قرار دارد بنابراین می‌توانیم محدودیت سوم و چهارم را با ضرب در ماکزیمم توان و ماکزیمم فاز ارضا کنیم و عملا نیازی به تعریف این محدودیت ها نداریم. از طرفی میزان پاداش را به گونه ای قرار میدهیم که با ارضا شدن ۳ محدودیت اول، عامل به مقدار تابع هدف، پاداش دریافت کند و در غیر اینصورت با توجه به اهمیت قید ها یک جریمه دریافت کند، جریمه دریافتی در صورت ارضا نشدن قید C3 بصورت TR- و در صورت ارضا نشدن قید های C1 و C2 بصورت 2*TR- خواهد بود.

در محیط تعریف شده، کانال ها، تحت توزیع رایسین، رایلی و دید مستقیم در هر step حالت دهی میشوند و دینامیک مسئله بالا را در محیط مورد نظر تحت شرایط موجود، برای یادگیری و بهینه سازی پیاده میکنیم.

(۳) الگوریتم های یادگیری تقویت شده

در ادامه با توجه به تعریف و پیاده سازی محیط، الگوریتم های مختلف RL اعم از temporal difference، Actor-Critic و Multi-Agent RL را توضیح داده و بهترین الگوریتم را برای حل این مسئله انتخاب می‌نماییم.

۱.۳. الگوریتم های TEMPORAL DIFFERENCE

یادگیری تفاوت زمانی (TD) به کلاسی از روش‌های یادگیری تقویتی بدون مدل اشاره دارد که با بوت‌استرپ کردن از تخمین فعلی تابع ارزش یاد می‌گیرند. این روش‌ها مانند روش‌های مونت کارلو، از محیط نمونه‌برداری می‌کنند، و به‌روزرسانی‌ها را مانند روش‌های برنامه‌نویسی پویا، بر اساس تخمین های فعلی انجام می‌دهند. در حالی که روش‌های مونت کارلو تنها زمانی تخمین‌های خود را تنظیم می‌کنند که نتیجه نهایی مشخص شود، روش‌های TD پیش‌بینی‌ها را تنظیم می‌کنند تا پیش‌بینی‌های بعدی و دقیق‌تر در مورد آینده را قبل از مشخص شدن نتیجه نهایی مطابقت دهند. این بوت‌استرپینگ به روش‌های TD اجازه می‌دهد تا تخمین‌های خود را به‌صورت آنلاین، برخلاف روش‌های مونت کارلو که نیاز به اتمام اپیزود قبل از به‌روزرسانی دارند، پس از هر مرحله زمانی، به‌روزرسانی کنند.

قانون به‌روزرسانی در روش‌های TD معمولاً شامل تنظیم تخمین تابع مقدار بر اساس تفاوت بین مقدار وضعیت فعلی (یا جفت حالت-عمل) و مقدار وضعیت بعدی (یا جفت حالت-عمل) مشاهده شده در محیط است. این تفاوت با پارامتر نرخ یادگیری مقیاس بندی می شود.

SARSA (State-Action-Reward-State-Action) و Q-learning دو الگوریتم TD محبوب هستند که در یادگیری تقویتی استفاده می شوند. هدف هر دو الگوریتم یادگیری تابع مقدار عمل بهینه $Q(s,a)$ است، که نشان دهنده پاداش تجمعی مورد انتظار است که با انجام عمل a در حالت s و پیروی از یک سیاست خاص پس از آن به دست می آید. با این حال، آنها در نحوه به روز رسانی تخمین های خود و نحوه انتخاب اقدامات متفاوت هستند. در زیر بصورت خلاصه به دو مورد از الگوریتم های TD یعنی SARSA و Q-learning میپردازیم.

SARSA (STATE-ACTION-REWARD-STATE-ACTION)

SARSA یک الگوریتم کنترل TD روی سیاست (on-policy) است، به این معنی که ارزش سیاست مورد پیگیری را می‌آموزد و تخمین‌های ارزش عمل خود را بر اساس انتقال‌ها و پاداش‌های مشاهده‌شده به‌روزرسانی می‌کند.

یک عامل SARSA با محیط تعامل می کند و سیاست را بر اساس اقدامات انجام شده به روز می کند، از این رو این الگوریتم یادگیری روی سیاست شناخته می شود. مقدار Q برای یک عمل-حالت با یک خطا به روز می شود که با نرخ یادگیری α تنظیم می شود. مقادیر Q نشان دهنده پاداش احتمالی دریافت شده در مرحله زمانی بعدی برای انجام اقدام a در حالت s، به علاوه پاداش discount شده دریافت شده از مشاهده اقدام بعدی است.

الگوریتم آن بصورت زیر میباشد:

۱- *مقداردهی اولیه:*

تابع action-value Q(s,a) را به طور دلخواه برای همه جفت های حالت-عمل مقدار دهی میکنیم.

۲- *انتخاب عمل (سیاست):*

یک عمل a را با استفاده از استراتژی اکتشاف-بهره برداری، مانند ε-greedy، بر اساس وضعیت فعلی s انتخاب می‌کنیم.

۳- *تعامل با محیط:*

عمل a را در حالت s انجام میدهیم، حالت بعدی s و پاداش R را دریافت میکنیم.

۴- *به روز رسانی تابع Action-Value:*

تابع action-value را با استفاده از انتقال بین حالت ها و پاداش مشاهده شده به روز میکنیم:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma Q(s',a') - Q(s,a)$$

a' عمل بعدی، تحت سیاست فعلی میباشد.

۵- *به روز رسانی حالت ها:*

s را روی s' قرار میدهیم و مراحل ۲-۴ را تکرار میکنیم تا شرط پایان برآورده شود.

Q-LEARNING

Q-learning یک الگوریتم کنترل TD خارج از سیاست (off-policy) است، به این معنی که ارزش سیاست بهینه را در حالی که از یک سیاست متفاوت و احتمالاً اکتشافی پیروی می کند، یاد می گیرد و تخمین ارزش عمل خود را بر اساس حداکثر مقدار حالت بعدی به روز می کند.

یادگیری Q بر اساس ایده یادگیری تابع ارزش عمل Q(s,a) است که نشان دهنده پاداش تجمعی مورد انتظار است که با انجام اقدام a در حالت s و پیروی از سیاست بهینه پس از آن به دست می آید. این الگوریتم به طور مکرر مقادیر Q خود را بر اساس انتقال ها و پاداش های مشاهده شده به روز می کند.

الگوریتم آن بصورت زیر میباشد:

۱- *مقداردهی اولیه:*

تابع action-value Q(s,a) را به طور دلخواه برای همه جفت های حالت-عمل مقدار دهی میکنیم.

۲- *انتخاب عمل (سیاست):*

یک عمل a را با استفاده از استراتژی اکتشاف-بهره برداری، مانند ε-greedy، بر اساس وضعیت فعلی s انتخاب می‌کنیم.

۳- *تعامل با محیط:*

عمل a را در حالت s انجام میدهیم، حالت بعدی s' و پاداش R را دریافت میکنیم.

۴- *به روز رسانی تابع Action-Value:*

تابع action-value را با استفاده از انتقال بین حالت ها و پاداش مشاهده شده به روز میکنیم:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

۵- *به روز رسانی حالت ها:*

s را روی s' قرار میدهیم و مراحل ۲-۴ را تکرار میکنیم تا شرط پایان برآورده شود.

مقایسه

SARSA به طور کلی محافظه کارتر و ایمن تر است زیرا برآوردهای خود را بر اساس سیاستی که دنبال می کند به روز می کند. این می تواند منجر به یادگیری پایدارتر شود، به ویژه در محیط هایی با واریانس بالا یا پاداش های غیر ثابت.

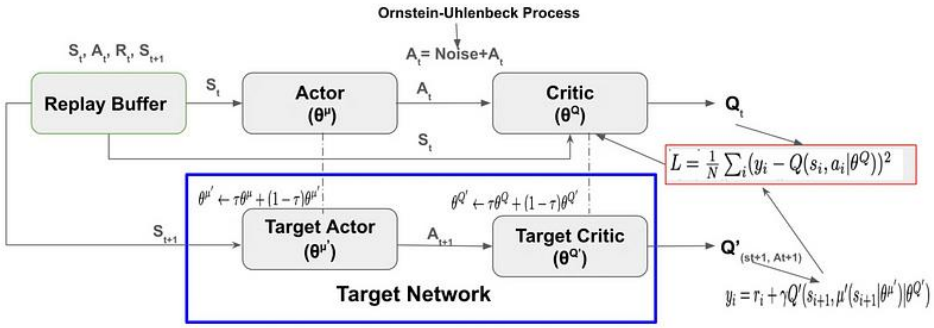
یادگیری Q، که خارج از سیاست است، گاهی اوقات می‌تواند سریع‌تر یاد بگیرد و با یادگیری سیاست بهینه بدون توجه به سیلست که در طول کاوش دنبال می‌کند، به عملکرد مجانبی بهتری دست یابد. با این حال، ممکن است بیشتر مستعد سوگیری های تخمین بیش از حد باشد، به ویژه در محیط هایی با واریانس بالا یا پاداش های غیر ثابت.

۲.۳. الگوریتم های پیوسته و ACTOR-CRITIC

معماری Actor-Critic با جداسازی توابع policy و value امکان یادگیری کارآمد را فراهم می کند. این جداسازی، عامل را قادر می‌سازد تا همزمان سیاست بهینه و تابع مقدار متناظر آن را بیاموزد. در زیر به چند نمونه از آن ها اشاره میکنیم و بر محیط خود تست میکنیم.

DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

DDPG یک الگوریتم یادگیری تقویتی off-policy و بصورت Actor-Critic است که می تواند برای حل مسائل با فضای عمل پیوسته استفاده شود. این الگوریتم از هر دو روش مبتنی بر سیاست و مبتنی بر ارزش را ترکیب می کند و از شبکه های عصبی عمیق برای تقریب توابع سیاست و ارزش استفاده می کند.



یک شبکه Actor را ایجاد می کند که سیاست بهینه را به طور قطعی می آموزد و یک شبکه Critic که ارزش اقدامات انجام شده توسط Actor را ارزیابی می کند. با استفاده از شبکه‌های عصبی عمیق، DDPG می‌تواند سیاست‌های پیچیده و توابع ارزش را تقریب بزند و آن را قادر می‌سازد تا فضاهای حالت و عمل با ابعاد بالا را مدیریت کند.

شبکه Actor:

شبکه Actor که با θ^{μ} پارامترگذاری شده است و بصورت $\mu(s|\theta^{\mu})$ مشخص میشود ، با هدف به حداکثر رساندن بازده مورد انتظار[1]، به طور مستقیم حالت ها را به عمل خاص تبدیل می‌کند. با تقریب قطعی سیاست بهینه، این شبکه فرآیند تصمیم‌گیری عامل را بدون نیاز به کاوش هدایت می کند.

سیاست قطعی ایجاد شده توسط شبکه Actor، مبادله اکتشاف و بهره برداری[2] را ساده می کند، زیرا عامل می تواند مستقیماً از سیاست آموخته شده بدون نیاز به استراتژی های اکتشافی صریح استفاده کند. همچنین ماهیت قطعی سیاست می‌تواند منجر به یادگیری پایدارتر و ویژگی‌های همگرایی بهتر در مقایسه با سیاست‌های تصادفی، به ویژه در محیط‌هایی با فضاهای عمل پیوسته شود.

شبکه Critic:

شبکه Critic که توسط θ^{Q} پارامتر شده است و بصورت $Q(s,a|\theta^{Q})$ مشخص میشود، تابع مقدار عمل بهینه را ارزیابی می کند و ارزش انجام یک اقدام خاص را در یک حالت مشخص تخمین می زند. شبکه منتقد با یادگیری تقریبی Q-value، بازخوردی را به شبکه Actor ارائه می‌کند و آن را به سمت اقداماتی هدایت می‌کند که منجر به بازدهی بالاتر می‌شود.

شبکه Critic با ارائه بازخورد به شبکه Actor در قالب مقادیر Q، که نشان دهنده بازده مورد انتظار انجام یک اقدام خاص در یک وضعیت معین است، نقش مهمی در DDPG ایفا می کند. همچنین با یادگیری تقریب Q-value، شبکه Critic عامل را قادر می سازد تا کیفیت اقدامات خود را ارزیابی کند و تصمیمات آگاهانه ای در مورد اقداماتی که در حالات مختلف انجام دهد اتخاذ کند.

توابع هدف:

هدف شبکه Actor، به حداکثر رساندن بازده مورد انتظار با انتخاب اقداماتی است که مقدار Q برآورد شده را به حداکثر می‌رسانند.

$$J(\theta^{\mu}) = \mathbb{E}_{s_t \sim \rho^{\beta}}[Q(s, \mu(s \mid \theta^{\mu}) \mid \theta^{Q})]$$

[1] Expected return
[2] Exploration & Exploitation

از سوی دیگر، هدف شبکه Critic این است که خطای اختلاف زمانی (TD) را با یادگیری تقریب دقیق‌تر مقدار Q واقعی به حداقل برساند.

$$J(\theta^Q) = \mathbb{E}_{s_t,a_t\sim\rho^\beta}[(Q(s,a \mid \theta^Q) - y)^2]$$

که در آن $y = r + \gamma Q'\left(s', \mu'\left(s' \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$ می‌باشد.

s' حالت بعدی، μ' سیاست هدف (target policy (Actor) network) و Q' تابع ارزش Q هدف (target Q-value (Critic) network) می‌باشد.

الگوریتم DDPG بصورت زیر می‌باشد:

۱- مقدار دهی اولیه شبکه Actor و Critic

۲- مقدار دهی اولیه شبکه های Actor target و Critic target بر اساس شبکه های آنلاین Actor و Critic

۳- مقدار دهی اولیه replay buffer برای ذخیره تجربه ها

۴- برای هر اپیزود:

a. مقدار دهی حالت اولیه

b. برای هر timestep:

i. انتخاب عمل با توجه به سیاست کنونی و نویز بصورت $a_t = \mu(s_t|\theta^\mu) + N$

ii. اجرای عمل انتخاب شده و دریافت پاداش و مشاهده حالت بعد s_{t+1}

iii. ذخیره حالت انتقال (s_t, a_t, r_t, s_{t+1}) در بافر

iv. آپدیت شبکه Critic با کمینه کردن خطای TD : $J(\theta^Q)$

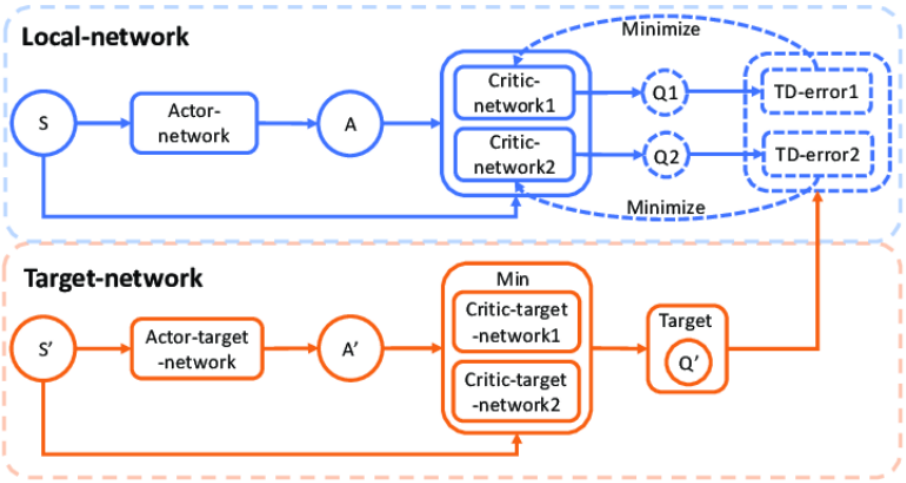
v. آپدیت شبکه Actor با گرادیان سیاست

vi. آپدیت شبکه های target

با ترکیب عناصر روش‌های مبتنی بر سیاست و ارزش و استفاده از شبکه‌های عصبی عمیق برای تقریب توابع، DDPG می‌تواند به طور موثر سیاست‌های پیچیده را از فضاهای حالت با ابعاد بالا بیاموزد. علیرغم چالش های آن، مانند واریانس بالا و همگرایی کند آن.

TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENT (TD3)

TD3 یک الگوریتم یادگیری تقویتی خارج از سیاست (off-policy) است که بر اساس DDPG بنا شده و برخی از محدودیت های آن را برطرف می کند. DDPG به دلیل توانایی خود در مدیریت فضاهای پیوسته مستمر در وظایف یادگیری تقویتی شناخته شده است، اما از تخمین بیش از حد[1] برآورد و مسائل بی ثباتی رنج می برد.



TD3 از معماری Actor-Critic استفاده می‌کند، جایی که شبکه Actor سیاست را می‌آموزد و شبکه Critic تابع ارزش عمل را تقریب می‌زند.در این الگوریتم دو شبکه Critic وجود دارد، که این دو با در نظر گرفتن حداقل دو تخمین Q-value به کاهش سوگیری تخمین بیش از حد کمک می کنند. هموارسازی سیاست هدف، به روز رسانی سیاست ها را منظم می کند و آموزش را تثبیت می کند. به روز رسانی های تاخیری همبستگی بین به روز رسانی های متوالی را کاهش می دهد و کاوش را بهبود می دهد.

شبکه Actor (π):

شبکه Actor سیاست عامل را تعیین می کند و حالت ها را به مقادیر عمل پیوسته نگاشت می کند. آموزش داده شده است تا با تقریب مستقیم اقدام بهینه با توجه به وضعیت فعلی، پاداش‌های تجمعی مورد انتظار را به حداکثر برساند.

تابع ضرر Actor نیز بصورت روبرو می‌باشد: $\mathcal{L}_{\text{actor}} = -\mathbb{E}[Q(s,\pi(s))]$

[1] Overestimated

شبکه های Critic (Q1, Q2):

شبکه‌های Critic، عملکرد ارزش عمل را تخمین می‌زنند و بازخوردی را در مورد خوب بودن کنش‌ها به Actor ارائه می‌کنند. TD3 از دو شبکه Critic برای کاهش تخمین بیش از حد برآورد استفاده می کند. این شبکه ها برای به حداقل رساندن خطای اختلاف زمانی (TD) بین مقادیر Q پیش بینی شده و هدف آموزش داده شده اند.

تابع ضرر Critic بصورت روبرو تعریف میشود: $\mathcal{L}_{\text{critic}} = \frac{1}{2}\mathbb{E}[(Q(s,a)-y)^2]$

که در آن $y = r + \gamma \min_{i=1,2} Q_{\text{target}}\left(s', \pi_{\text{target}}(s') + \epsilon\right)$ اختلاف زمانی TD هدف است و $\epsilon \sim \text{clip}\left(\mathcal{N}(0,\sigma), -c, c\right)$ نویز برای هموار کردن هدف میباشد.

شبکه های هدف:

برای تثبیت آموزش، TD3 از شبکه های هدف هم برای Actor و هم برای Critic استفاده می کند. این شبکه ها به آرامی با استفاده از مکانیزم به روز رسانی نرم به روز می شوند، که در آن پارامترهای هدف میانگین وزنی پارامترهای فعلی و پارامترهای هدف هستند.

TD3 در طول آموزش، نویز را به سیاست هدف اضافه می کند تا به روز رسانی ها را منظم کند. این به جلوگیری از بهره برداری Actor از سیاست های اولیه غیربهینه کمک می کند و استحکام را بهبود می‌بخشد. همچنین برای کاهش همبستگی بین به‌روزرسانی‌های متوالی و تثبیت آموزش، TD3 شبکه‌های Actor و Critic را کمتر از الگوریتم‌های سنتی به‌روزرسانی می‌کند.

TD3 چندین مزیت را نسبت به DDPG ارائه می دهد. با کاهش تخمین بیش از حد برآورد از طریق twin-Critic و هموارسازی سیاست های هدف، ثبات را بهبود می بخشد. با یادگیری سیاست های قوی‌تر، کارایی نمونه بالاتری را به دست می آورد. علاوه بر این، TD3 عملکرد بهبود یافته ای را در محیط های پیچیده با فضاهای عملی با ابعاد بالا نشان می دهد.

الگوریتم TD3 بصورت زیر می‌باشد:

۱- مقدار دهی اولیه شبکه Actor یعنی π

۲- مقدار دهی اولیه شبکه های Twin-Critic یعنی Q_2 و Q_1

۳- مقدار دهی اولیه شبکه های Actor target و Twin-Critic target بر اساس شبکه های آنلاین Actor و Critic یعنی Q_2' و Q_1'

۴- مقدار دهی اولیه replay buffer برای ذخیره تجربه ها

۵- برای هر اپیزود:

a. مقدار دهی حالت اولیه

b. برای هر timestep:

i. برداشت نمونه حالت انتقال (s_t, a_t, r_t, s_{t+1}) از بافر

ii. محاسبه TD target بصورت $y = r + \gamma \min_{i=1,2} Q_{target}(s', \pi_{target}(s') + \epsilon$

که ϵ نویز بریده شده از توزیع هموارسازی سیاست هدف میباشد.

iii. آپدیت شبکه Critic با کمینه کردن خطای ضرر بیان شده

iv. آپدیت شبکه Actor با بیشینه کردن مقدار ارزش Q پیشبینی شده توسط Critic

v. آپدیت شبکه های target بصورت آپدیت نرم: $\theta_{target} \leftarrow \tau\theta + (1-\tau)\theta_{target}$

SOFT ACTOR CRITIC (SAC)

Soft Actor-Critic (SAC) یک الگوریتم یادگیری تقویتی پیشرفته است که به دلیل کارایی و اثربخشی در حل انواع مسئله های کنترل پیوسته شناخته شده است. SAC به دلیل توانایی آن در یادگیری کارآمد از فضاهای عمل پیوسته با ابعاد بالا و در عین حال بهینه سازی همزمان برای تابع های سیاست و ارزش متمایز است.

SAC در چارچوب فرآیندهای تصمیم گیری مارکوف (MDPs) عمل می کند، جایی که یک عامل با انجام عمل ها و دریافت پاداش با یک محیط تعامل می کند. هدف، به حداکثر رساندن پاداش تجمعی مورد انتظار در طول زمان است. به طور رسمی، الگوریتم SAC یک سیاست پارامتری شده را که با π (a|s;θ) مشخص می شود، بهینه می کند، جایی که θ نشان دهنده پارامترهای سیاست است.

تابع هدف اصلی SAC شامل سه جزء اصلی است:

I. *Entropy-Regularized Expected Return*

تابع هدف شامل یک عبارت آنتروپی است که کاوش را تشویق می کند. اصطلاح آنتروپی به صورت زیر تعریف می شود:

$$H(\pi(\cdot \mid s)) = -\mathbb{E}_{a\sim\pi(\cdot|s)}[\log(\pi(a \mid s))]$$

این عبارت، سیاست‌هایی را که بیش از حد قطعی هستند، جریمه می‌کند و تصادفی بودن را در انتخاب عمل ترویج می‌کند.

II. *Soft Q-Function*

تابع Q نرم، پاداش تجمعی مورد انتظار انجام یک عمل a با پیروی از سیاست در یک حالت را تخمین می زند. به صورت $Q^{\pi}(s,a)$ نشان داده می شود و به عنوان مجموع مورد انتظار پاداش ها تعریف می‌شود:

$$Q^{\pi}(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a) + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^{\pi}(s',a')]\right]$$

بطوری که $r(s,a)$ پاداش فوری پس از انجام عمل a در حالت s است و $P(\cdot \mid s,a)$ احتمال انتقال از حالت s به حالت بعدی s' است.

III. *Soft Value Function*

تابع ارزش نرم، پاداش تجمعی مورد انتظار قابل دستیابی از یک وضعیت تحت سیاست فعلی را تخمین می‌زند. به صورت $V^{\pi}(s)$ نشان داده می شود و به عنوان مجموع مورد انتظار پاداش ها تعریف می شود:

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi}(s,a) - \alpha\log\,\pi(a \mid s)]$$

که α یک پارامتر تنظیم است و تنظیم آنتروپی را کنترل می کند.

تابع هدف SAC این مولفه ها را ترکیب می کند و معمولاً مجموع تابع Q نرم، عبارت آنتروپی و تابع مقدار نرم را به حداکثر می رساند.

اجزا SAC را میتوان بصورت زیر بیان کرد:

Actor Network:

شبکه Actor تابع سیاست $\pi(a|s;\theta)$ را پارامتر می کند، جایی که θ نشان دهنده پارامترهای سیاست است که توزیع احتمال را بر روی عمل های انجام شده در حالت های داده‌شده خروجی می دهد.

Critic Networks:

SAC از دو شبکه Critic استفاده می کند: یکی برای تخمین تابع Q نرم و دیگری برای تخمین تابع ارزش نرم. این شبکه‌ها یاد می‌گیرند که پاداش‌های تجمعی مورد انتظار از انجام عمل ها با توجه به پیروی از سیاست‌های موجود تقریبی بزنند.

Target Networks:

برای تثبیت آموزش، SAC از شبکه‌های هدف استفاده می‌کند که به آرامی به‌روزرسانی می‌شوند تا مقادیر هدف را برای تابع Q-soft و تابع ارزش نرم ارائه کنند. این به کاهش مسائل مربوط به سوگیری برآورد ارزش هدف کمک می کند.

Replay Buffer:

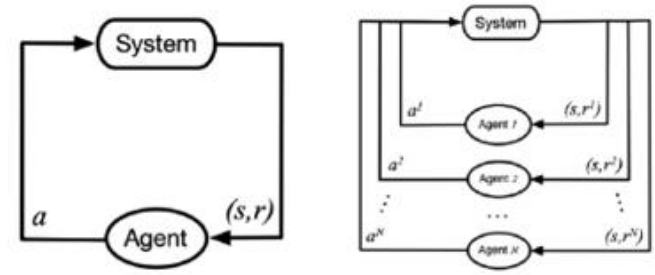
SAC از یک بافر برای ذخیره تجربیات (state-action-reward-next tuples) برای استفاده موثر در آموزش استفاده می کند. در طول هر تکرار آموزشی، تجربیات از بافر بازپخش برای به‌روزرسانی شبکه‌های Actor و Critic نمونه‌برداری می‌شود.

فرآیند آموزش شامل نمونه‌برداری از تجربیات از بافر پخش مجدد، محاسبه گرادیان تابع هدف با توجه به پارامترهای سیاست، و به‌روزرسانی شبکه‌ها با استفاده از روش‌های نزولی گرادیان تصادفی است.

۳.۳. الگوریتم های یادگیری تقویت شده چند عامله

MULTI AGENT DDPG

یادگیری تقویتی چند عاملی (MARL) شامل عامل های متعددی است که در یک محیط مشترک با یکدیگر تعامل دارند، جایی که عمل هر عامل بر محیط و مشاهدات سایر عوامل تأثیر می گذارد. MADDPG به چالش‌های هماهنگی و همکاری بین چندین عامل می‌پردازد و آنها را قادر می‌سازد تا سیاست‌های غیرمتمرکز را در حین در نظر گرفتن تعامل جهانی بیاموزند.



(a) Markov decision process (b) Markov game

MADDPG با معرفی شبکه‌های Actor غیرمتمرکز و شبکه‌های Critic متمرکز، DDPG را به تنظیمات چند عاملی گسترش می‌دهد. در MADDPG، هر عاملی شبکه Actor خود را دارد که سیاست خود را تنها بر اساس مشاهدات خود می‌آموزد. با این حال، همه عامل ها یک شبکه Critic متمرکز را به اشتراک می گذارند، که اقدامات مشترک و مشاهدات همه عوامل را به عنوان ورودی برای تخمین تابع ارزش عمل می گیرد.

فرض کنید فضای حالت را با S، فضای عمل را با A، و فضای مشاهده را با O نشان دهیم. برای یک محیط چند عاملی با N عامل، عمل هر عامل به صورت a_i، و مشاهده با o_i، که i نشان دهنده هر عامل است، نشان داده می شود.

شبکه Actor برای عامل i سیاست $\mu_i(o_i;\theta_i)$ را می آموزد، بصورتی که θ_i پارامترهای شبکه Actor را نشان می دهد. شبکه Critic تابع action-value، $Q(o_1,\dots,o_N,a_1,\dots,a_N)$ را با استفاده از مشاهدات و عمل های همه عوامل تخمین می زند.

شبکه Actor برای به حداکثر رساندن بازده مورد انتظار آموزش بر اساس تابع هدف زیر آموزش می‌بینند:

$$J(\theta_i) = \mathbb{E}_{o_i\sim O, a_i\sim\mu_i}[Q(o_1,\dots,o_N,a_1,\dots,a_N)]$$

شبکه Critic برای به حداقل رساندن خطای TD بر اساس تابع هدف زیر آموزش می‌بینند:

$$L = \mathbb{E}_{(o_1,\dots,o_N),(a_1,\dots,a_N),r,(o'_1,\dots,o'_N)}\left[\left(y - Q(o_1,\dots,o_N,a_1,\dots,a_N)\right)^2\right]$$

که در آن $y = r + \gamma Q'\left(o'_1,\dots,o'_N,\mu'_1(o'_1),\dots,\mu'_N(o'_N)\right)$ می‌باشد و Q' و μ' نشان دهنده به ترتیب شبکه های target Critic و target Actor می‌باشد.

عامل ها در طول آموزش یک Critic متمرکز را به اشتراک می گذارند اما در طول تعامل سیاست های غیرمتمرکز را اجرا می کنند. همچنین عامل ها در طول آموزش، تجربه های خود را با یکدیگر به اشتراک می گذارند.

برای اطمینان از اینکه هر عامل فقط اقدامات معتبر را انتخاب می کند، از پوشش عمل[1] استفاده می شود. هر عامل یک ماسک عمل دریافت می کند که عمل های معتبر را بر اساس مشاهده محلی خود نشان می‌دهد. این مانع از انتخاب عمل هایی می شود که در وضعیت فعلی امکان پذیر نیستند.

الگوریتم MADDPG شامل مراحل زیر است:

۱. شبکه های Actor و Critic برای هر عامل مقدار دهی اولیه میشود.
۲. یک مجموعه کوچک از تجربیات را از بافر پخش مجدد نمونه برداری میشود.
۳. مقادیر target Q را با استفاده از معادله بلمن که در بالا اشاره شد محاسبه میشود.
۴. شبکه‌های Critic با استفاده از گرادیان نزولی به‌روزرسانی میشود تا خطای اختلاف زمانی به حداقل برسد.
۵. شبکه های Actor با استفاده از قضیه گرادیان سیاست قطعی به روز میشود.
۶. به طور دوره ای شبکه های target با استفاده از به روز رسانی های هدف نرم[2] به روز رسانی میشود. $\theta_{target} \leftarrow \tau\theta + (1-\tau)\theta_{target}$
۷. مراحل ۲-۶ را تا زمان همگرایی تکرار میشود.

به طور کلی، MADDPG مزایای تصمیم گیری غیرمتمرکز را با یادگیری متمرکز ترکیب می کند تا هماهنگی موثر بین چندین عامل در محیط های پیچیده را فراهم کند. استفاده از Policy gradients و Central critics یادگیری سیاست های قوی و مشارکتی را تسهیل می کند.

[1] Action masking
[2] Soft target updates

(۴) شبیه سازی و نتیجه گیری

در این بخش، ابتدا پارامتر های سیستم مدل و شرایط شبیه سازی را بیان کرده، سپس نتایج و حل مسئله توسط الگوریتم های توضیح داده شده را بیان میکنیم و در آخر مقایسه جزیی بین آن ها ارائه میدهیم.

۱.۴. پارامتر های شبیه سازی و سیستم مدل

برای شبیه سازی و حل مسئله، ما $K_t = 1$، $K_r = 2$، $L_t = 2$، $L_r = 1$، و $M_q = 20$ و $N_t = 4$ را تنظیم کردیم. همچنین برای پارامتر های محدودیت ها را بصورت، $P^{BS}_{max} = 3.5$ W، $\alpha = 0.5$ ، $P^{l}_{max} =$ 1W، و $T_{kb}^{DL} = T_{kb}^{UL} = 1$ms قرار دادیم.

با قرار دادن پارامتر های بالا اعم از تعداد کاربران UL و DL تعداد STAR-RIS به همراه تعداد المنت های آن ها و همچنین تعداد آنتن های ایستگاه پایه، بُعد فضای اکشن و حالت های ما بصورت زیر میشود:

- تعداد اکشن ها: ۱۳۱
- تعداد حالت ها: ۴۳۴

۲.۴. نتایج TEMPORAL-DIFFERENCE

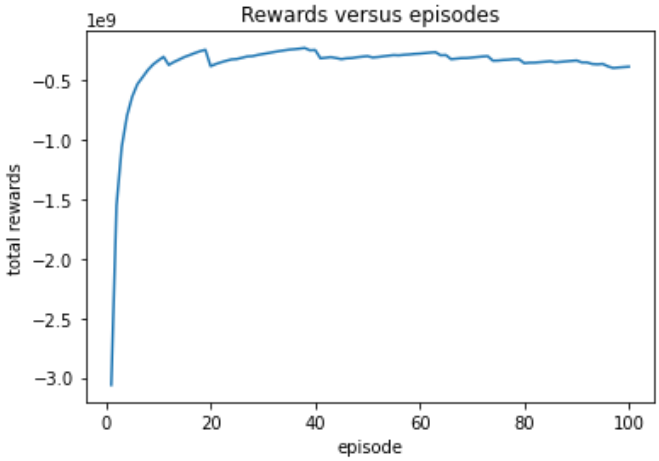
با توجه به این موضوع که این الگوریتم ها برای فضای اکشن گسسته طراحی شده اند، بنابراین امکان استفاده از آن ها در این مسئله وجود نداشت. علاوه بر این موضوع نیز به دلیل حجم بالای اکشن ها و تغییرات گسترده آن ها و اثرات زیاد آن ها با توجه به تغییر کوچک اکشن ها، امکان گسسته سازی اکشن ها نیز وجود ندارد.

۳.۴. نتایج DDPG

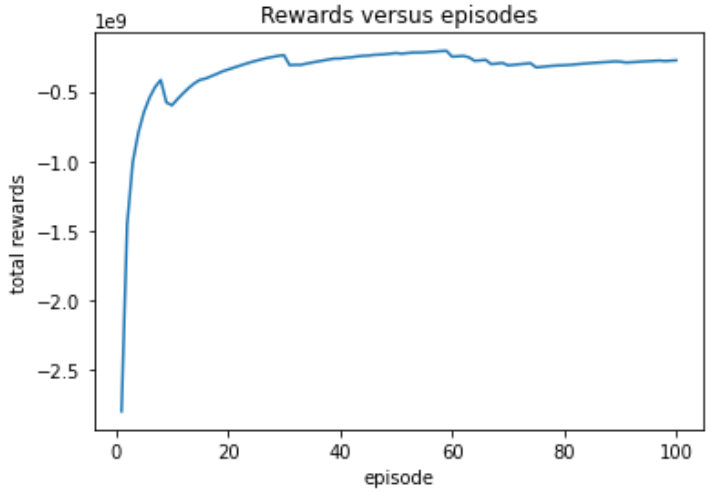
مسئله را با الگوریتم DDPG به تعداد ۲۰۰ اپیزود و حداکثر گام ۵۰۰ آموزش دادیم و در نهایت مدل را ذخیره کردیم، سپس مدل ذخیره شده را در ۱۰۰ اپیزود تست کردیم که نتایج آن بصورت زیر قابل مشاهده می‌باشد:

نمودار های زیر بر اساس مجموع میزان پاداش دریافتی در هر اپیزود رسم شده است.

نمودار آموزش مدل:



نمودار تست مدل:



همانطور که از روند نمودار آموزش پیداست، الگوریتم به خوبی در این محیط همگرا شده و نمودار تست نیز نشان دهنده، همگرایی میباشد.

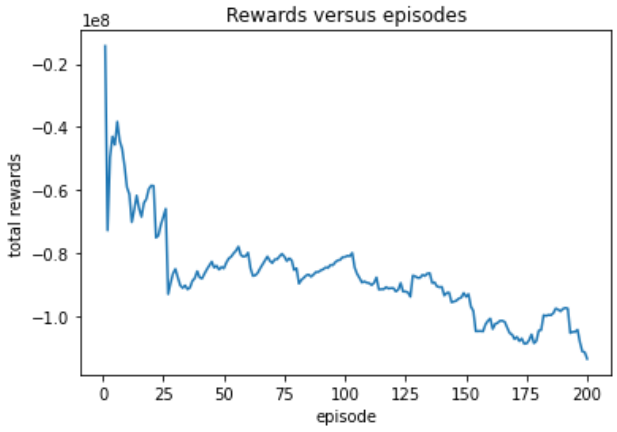
با مقایسه نتایج میتوان به بهبود نسبت به نتایج مقاله نیز پی برد.

۴.۴. نتایج TD3

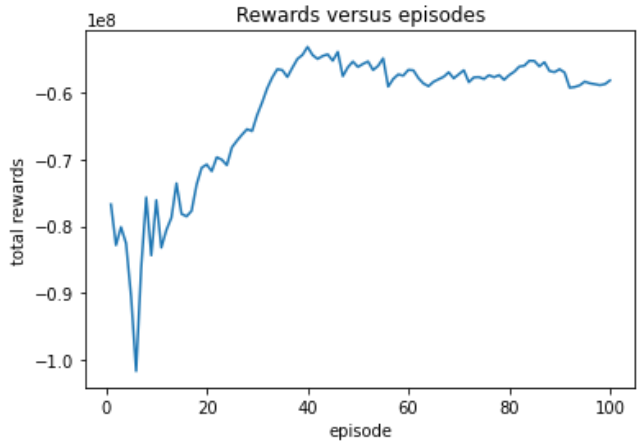
مسئله را با الگوریتم TD3 به تعداد ۲۰۰ اپیزود و حداکثر گام ۵۰۰ آموزش دادیم و در نهایت مدل را ذخیره کردیم، سپس مدل ذخیره شده را در ۱۰۰ اپیزود تست کردیم که نتایج آن بصورت زیر قابل مشاهده می‌باشد:

نمودار های زیر بر اساس مجموع میزان پاداش دریافتی در هر اپیزود رسم شده است.

نمودار آموزش مدل:



نمودار تست مدل:



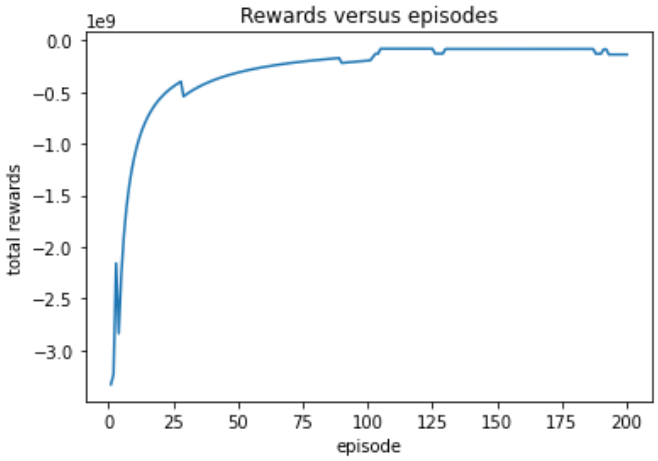
در این الگوریتم در نمودار آموزش، بر خلاف نمودار تست روند کاهشی دارد که میتوان آن را به دلیل آموزش کم با توجه به پیچیدگی بیشتر نسبت به DDPG نسبت داد.

۵.۴. نتایج SAC

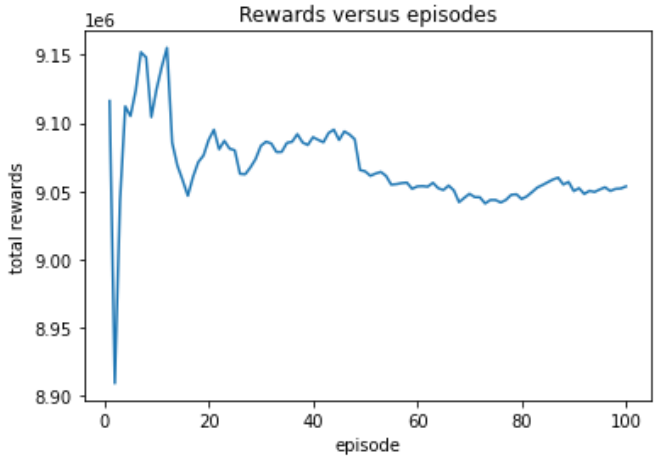
مسئله را با الگوریتم SAC به تعداد ۲۰۰ اپیزود و حداکثر گام ۵۰۰ آموزش دادیم و در نهایت مدل را ذخیره کردیم، سپس مدل ذخیره شده را در ۱۰۰ اپیزود تست کردیم که نتایج آن بصورت زیر قابل مشاهده می‌باشد:

نمودار های زیر بر اساس مجموع میزان پاداش دریافتی در هر اپیزود رسم شده است.

نمودار آموزش مدل:



نمودار تست مدل:



در این الگوریتم که بهترین نتیجه از آن گرفته شده است، همانطور که مشاهده میشود، در فرایند آموزش همگرایی رخ داده است و حتی به مقدار پاداش های بیشتر از الگوریتم های قبل دست یافته است.

نکته قابل توجه دیگری که میتوان به آن اشاره کرد، این است که در هنگام تست همانطور که از نمودار آن نیز پیداست، پاداش منفی وجود ندارد که نشان دهنده ارضا شدن قید های مسئله در گام اول می‌باشد و در تمامی اپیزود ها، در گام اول مسئله حل شده و پاداش مثبت دریافت میشود.

میزان تابع هدف مسئله نیز همانطور که از مقایسه با الگوریتم های دیگر میتوان تشخیص داد، به مقدار بزرگتری دست یافته است.

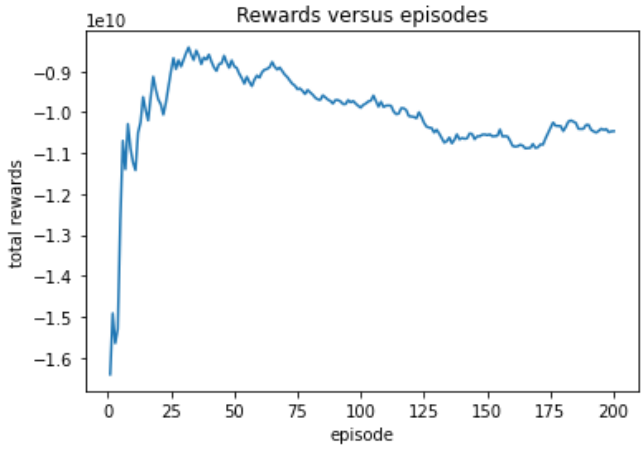
علت بهتر شدن نتیجه در این الگوریتم این است که در SAC عامل را مجبور به کاوش بیشتر نسبت به تصادفی بودن اکشن ها میکند و از طرفی چون مسئله ما حالت های تصادفی از توزیع های مختلف را داراست، بنابراین این الگوریتم بهترین نتیجه را همانطور که انتظار داشتیم میدهد.

۶.۴ نتایج MADDPG

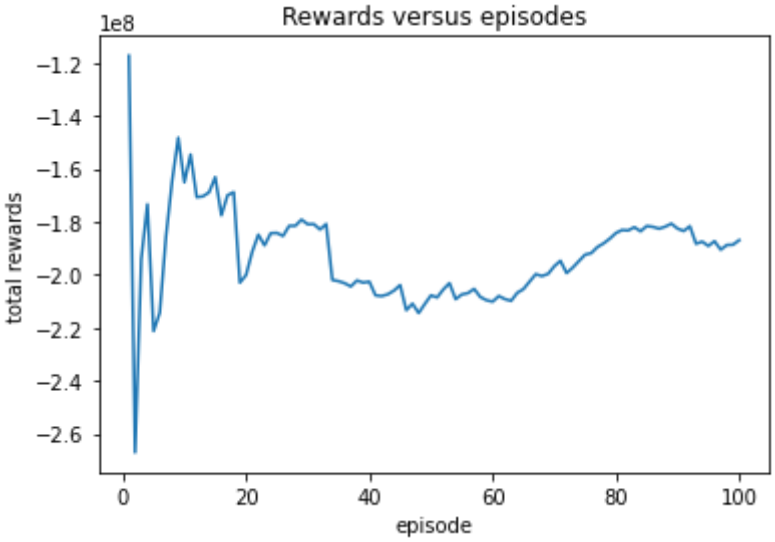
مسئله را با الگوریتم MADDPG به تعداد ۲۰۰ اپیزود و حداکثر گام ۵۰۰ آموزش دادیم و در نهایت مدل را ذخیره کردیم، سپس مدل ذخیره شده را در ۱۰۰ اپیزود تست کردیم که نتایج آن بصورت زیر قابل مشاهده می‌باشد:

نمودار های زیر بر اساس مجموع میزان پاداش دریافتی در هر اپیزود رسم شده است.

نمودار آموزش مدل:



نمودار تست مدل:



تمامی کد ها بصورت پابلیک در گیتهاب بنده به آدرس زیر موجود است:

https://github.com/amirhosein-prdv/Multi-agent-Reinforcement-learning-in-wireless-network-system.git