



دانشگاه صنعتی شریف
دانشکده‌ی مهندسی هوافضا

پروژه کارشناسی ارشد
مهندسی فضا

عنوان:

هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویای چندجسمی با پیشران کم

نگارش:

علی بنی اسد

استاد راهنما:

دکتر هادی نوبهاری

دی ۱۴۰۳



به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی هوافضا

پروژه کارشناسی ارشد

عنوان: هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویای چندجسمی
با پیشران کم

نگارش: علی بنی اسد

کمیته‌ی ممتحنین

استاد راهنما: دکتر هادی نوبهاری
امضاء:

استاد مشاور: استاد مشاور
امضاء:

استاد مدعو: استاد ممتحن
امضاء:

تاریخ:

سپاس

از استاد بزرگوارم جناب آقای دکتر نوبهاری که با کمک‌ها و راهنمایی‌های بی‌دریغشان، بنده را در انجام این پروژه یاری داده‌اند، تشکر و قدردانی می‌کنم. از پدر دلسوزم ممنونم که در انجام این پروژه مرا یاری نمود. در نهایت در کمال تواضع، با تمام وجود بر دستان مادرم بوسه می‌زنم که اگر حمایت بی‌دریغش، نگاه مهربانش و دستان گرمش نبود برگ برگ این دست نوشته و پروژه وجود نداشت.

چکیده

در این پژوهش، از یک روش مبتنی بر نظریه بازی^۱ به منظور کنترل وضعیت استند سه درجه آزادی چهارپره استفاده شده است. در این روش بازیکن اول سعی در ردگیری ورودی مطلوب می‌کند و بازیکن دوم با ایجاد اغتشاش سعی در ایجاد خطا در ردگیری بازیکن اول می‌کند. در این روش انتخاب حرکت با استفاده از تعادل نش^۲ که با فرض بدترین حرکت دیگر بازیکن است، انجام می‌شود. این روش نسبت به اغتشاش ورودی و همچنین نسبت به عدم قطعیت مدل‌سازی می‌تواند مقاوم باشد. برای ارزیابی عملکرد این روش ابتدا شبیه‌سازی‌هایی در محیط سیمولینک انجام شده است و سپس، با پیاده‌سازی روی استند سه درجه آزادی صحت عملکرد کنترل‌کننده تایید شده است.

کلیدواژه‌ها: چهارپره، بازی دیفرانسیلی، نظریه بازی، تعادل نش، استند سه درجه آزادی، مدل‌مبنا، تنظیم‌کننده مربعی خطی

¹Game Theory

²Nash Equilibrium

فهرست مطالب

۱	مقدمه	۱
۱-۱	انگیزه پژوهش	۱
۲-۱	تعریف مسئله	۲
۳-۱	یادگیری تقویتی	۳
۴-۱	یادگیری تقویتی چندعاملی	۳
۵-۱	محتوای گزارش	۴
۲	پیشینه پژوهش	۵
۱-۲	ماموریت‌های بین مداری	۵
۲-۲	یادگیری تقویتی	۷
۳-۲	پیشینه‌ی پژوهش یادگیری تقویتی چندعاملی	۸
۳	مدل‌سازی محیط یادگیری سه جسمی	۱۰
۱-۳	مسئله‌ی سه جسمی محدود دایره‌ای (CRTBP)	۱۰
۱-۱-۳	لاگرانژ و معادلات حرکت	۱۲
۲-۳	نقاط تعادل لاگرانژ	۱۲
۴	یادگیری تقویتی	۱۵
۱-۴	مفاهیم اولیه	۱۵

۱۶	۱-۱-۴	حالت و مشاهدات
۱۶	۲-۱-۴	فضای عمل
۱۶	۳-۱-۴	سیاست
۱۷	۴-۱-۴	مسیر
۱۷	۵-۱-۴	تابع پاداش و برگشت
۱۸	۶-۱-۴	ارزش در یادگیری تقویتی
۱۹	۷-۱-۴	معادلات بلمن
۲۰	۸-۱-۴	تابع مزیت
۲۱	۲-۴	عامل گرادیان سیاست عمیق قطعی
۲۱	۱-۲-۴	یادگیری Q در DDPG
۲۳	۲-۲-۴	سیاست در DDPG
۲۳	۳-۲-۴	اکتشاف و بهره‌برداری در DDPG
۲۳	۴-۲-۴	شبکه‌کد DDPG
۲۵	۳-۴	عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه
۲۶	۱-۳-۴	اکتشاف و بهره‌برداری در TD3
۲۶	۲-۳-۴	شبکه‌کد TD3
۲۸	۴-۴	عامل عملگر نقاد نرم
۲۸	۱-۴-۴	یادگیری تقویتی تنظیم‌شده با آنتروپی
۲۸	۲-۴-۴	سیاست در SAC
۲۹	۳-۴-۴	تابع ارزش در SAC
۲۹	۴-۴-۴	تابع Q در SAC
۲۹	۵-۴-۴	معادله بلمن در SAC
۳۰	۶-۴-۴	یادگیری Q
۳۰	۷-۴-۴	سیاست در SAC

۳۱	۸-۴-۴	اکتشاف و بهره‌برداری در SAC
۳۲	۹-۴-۴	شبکه‌کد SAC
۳۳	۵-۴	عامل بهینه‌سازی سیاست مجاور
۳۴	۱-۵-۴	سیاست در الگوریتم PPO
۳۵	۲-۵-۴	اکتشاف و بهره‌برداری در PPO
۳۵	۳-۵-۴	شبکه‌کد PPO
۳۷	۵	شبیه‌سازی عامل در محیط سه جسمی
۳۷	۱-۵	طراحی عامل
۳۷	۱-۱-۵	فضای حالت
۳۸	۲-۱-۵	فضای عمل
۳۹	۳-۱-۵	تابع پاداش
۴۰	۲-۵	شبیه‌سازی عامل
۴۰	۱-۲-۵	پارامترهای یادگیری الگوریتم‌های مورد استفاده
۴۳	۲-۲-۵	فرآیند آموزش
۴۵	۶	یادگیری تقویتی چندعاملی
۴۵	۱-۶	تعاریف و مفاهیم اساسی
۴۷	۲-۶	نظریه بازی‌ها
۴۷	۱-۲-۶	تعادل نش
۴۸	۲-۲-۶	بازی مجموع صفر
۴۹	۳-۶	گرادیان سیاست عمیق قطعی دوعاملی
۴۹	۱-۳-۶	چالش‌های یادگیری تقویتی در محیط‌های چندعاملی
۴۹	۲-۳-۶	معماری MA-DDPG در بازی‌های مجموع صفر
۵۰	۳-۳-۶	آموزش MA-DDPG در بازی‌های مجموع صفر

۵۱	اکتشاف در MA-DDPG ۴-۳-۶
۵۱	شبکه MA-DDPG برای بازی‌های دوعاملی مجموع صفر ۵-۳-۶
۵۳	مزایای MA-DDPG در بازی‌های مجموع صفر ۶-۳-۶
۵۳	عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی ۴-۶
۵۳	چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MATD3 ۱-۴-۶
۵۴	معماری MATD3 در بازی‌های مجموع صفر ۲-۴-۶
۵۴	آموزش MATD3 ۳-۴-۶
۵۵	اکتشاف در MATD3 ۴-۴-۶
۵۶	شبکه MATD3 برای بازی‌های چندعاملی مجموع صفر ۵-۴-۶
۵۸	مزایای MATD3 در بازی‌های مجموع صفر ۶-۴-۶
۵۸	عامل عملگر نقاد نرم چندعاملی ۵-۶
۵۸	چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MASAC ۱-۵-۶
۵۹	معماری MASAC در بازی‌های مجموع صفر ۲-۵-۶
۵۹	آموزش MASAC ۳-۵-۶
۶۱	اکتشاف در MASAC ۴-۵-۶
۶۱	شبکه MASAC برای بازی‌های چندعاملی مجموع صفر ۵-۵-۶
۶۳	مزایای MASAC در بازی‌های مجموع صفر ۶-۵-۶
۶۳	عامل بهینه‌سازی سیاست مجاور چندعاملی ۶-۶
۶۳	چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MAPPO ۱-۶-۶
۶۴	معماری MAPPO در بازی‌های مجموع صفر ۲-۶-۶
۶۴	آموزش MAPPO ۳-۶-۶
۶۶	اکتشاف در MAPPO ۴-۶-۶
۶۶	شبکه MAPPO برای بازی‌های چندعاملی مجموع صفر ۵-۶-۶
۶۷	مزایای MAPPO در بازی‌های مجموع صفر ۶-۶-۶

۶۸	۷-۶	تنظیمات آزمایشی
۶۸	۸-۶	نتایج عملکرد الگوریتم‌ها
۶۸	۹-۶	تحلیل پایداری و همگرایی
۶۸	۱۰-۶	مقایسه روش‌های تک‌عاملی و چندعاملی
۶۸	۱۱-۶	ارزیابی مقاومت الگوریتم‌ها در برابر اختلالات
۶۸	۱۲-۶	تحلیل آماری نتایج
۶۸	۱۳-۶	بحث و تفسیر نتایج
۶۸	۱۴-۶	مقایسه با معیارهای مرجع
۶۸	۱۵-۶	جمع‌بندی و پیشنهادات آتی
۶۹	۷	سخت افزار در حلقه عملکرد عامل در محیط
۷۰	۸	ارزیابی و نتایج یادگیری
۷۰	۱-۸	تنظیمات آزمایشی
۷۰	۲-۸	نتایج عملکرد الگوریتم‌ها
۷۰	۳-۸	تحلیل پایداری و همگرایی
۷۱	۴-۸	مقایسه با معیارهای مرجع

فهرست جداول

۱-۳	مقادیر عددی برای مسئله سه جسمی محدود (سیستم زمین-ماه)	۱۱
۲-۳	مقادیر عددی نقاط لاگرانژ برای مسئله سه جسمی محدود سیستم زمین-ماه	۱۴
۱-۵	ویژگی‌های الگوریتم‌های مورد استفاده در شبیه‌سازی	۴۰
۲-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم DDPG [۱]	۴۲
۳-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم TD3 [۱]	۴۲
۴-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم SAC [۱]	۴۳
۵-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم PPO [۱]	۴۳

فهرست تصاویر

۱-۳	هندسه مسئله سه بدنه محدود	۱۱
۲-۳	نقاط لاگرانژ	۱۳
۱-۴	حلقه تعامل عامل و محیط	۱۶
۱-۵	ساختار شبکه عصبی عامل	۴۱
۲-۵	ساختار شبکه عصبی نقاد	۴۱
۱-۶	حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط	۴۶
۱-۸	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی DDPG	۷۱
۲-۸	مقایسه مسیر و فرمان پیشران دو الگوریتم تک‌عاملی و چندعاملی DDPG	۷۱
۳-۸	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی PPO	۷۲
۴-۸	مقایسه مسیر و فرمان پیشران دو الگوریتم تک‌عاملی و چندعاملی PPO	۷۲
۵-۸	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی SAC	۷۲
۶-۸	مقایسه مسیر و فرمان پیشران دو الگوریتم تک‌عاملی و چندعاملی SAC	۷۳
۷-۸	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی TD3	۷۳
۸-۸	مقایسه مسیر و فرمان پیشران دو الگوریتم تک‌عاملی و چندعاملی TD3	۷۳
۹-۸	مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی DDPG در سناریوهای مختلف	۷۴
۱۰-۸	مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی PPO در سناریوهای مختلف	۷۴
۱۱-۸	مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی SAC در سناریوهای مختلف	۷۵

- ۸-۱۲ مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی TD3 در سناریوهای مختلف . ۷۶
- ۸-۱۳ مقایسه مجموع پاداش الگوریتم‌های تک‌عاملی در سناریوهای مختلف ۷۶
- ۸-۱۴ مقایسه مجموع پاداش الگوریتم‌های چندعاملی در سناریوهای مختلف ۷۷

فهرست الگوریتم‌ها

۲۴	گرایان سیاست عمیق قطعی	۱
۲۷	عامل گرایان سیاست عمیق قطعی تاخیری دوگانه	۲
۳۲	عامل عملگرد نقاد نرم	۳
۳۶	بهینه‌سازی سیاست مجاور (PPO-Clip)	۴
۵۲	گرایان سیاست عمیق قطعی چندعاملی برای بازی‌های مجموع صفر	۵
۵۷	عامل گرایان سیاست عمیق قطعی تاخیری دوگانه دوعاملی	۶
۶۲	عامل عملگرد نقاد نرم دوعاملی	۷
۶۷	عامل بهینه‌سازی سیاست مجاور دوعاملی	۸

فصل ۱

مقدمه

در سال‌های آغازین عصر فضا، فرآیند هدایت فضایی‌ها عمدتاً بر مبنای دینامیک کلاسیک و کنترل خطی استوار بوده است. با این حال، پیچیدگی روزافزون مأموریت‌های کنونی مانند سفرهای میان‌سیاره‌ای با پیشران کم و شبکه‌های انبوه ماهواره‌ای در مدار زمین، موجب دوچندان شدن ضرورت بهره‌گیری از روش‌های هوشمند و تطبیق‌پذیر شده است.

۱-۱ انگیزه پژوهش

در دو دهه‌ی اخیر، مأموریت‌های فضایی به دلیل کوچک‌سازی سامانه‌ها، توسعه‌ی وسایل الکترونیک مقرون به صرفه و افزایش ظرفیت‌های پرتابی، تحولات بنیادینی را تجربه کرده است. از پروژه‌های علمی بین‌سیاره‌ای گرفته تا منظومه‌های انبوه ماهواره‌ای در مدارهای پایین زمین، همگی با چالش فراگیر هدایت بهینه در حضور عدم قطعیت‌ها مواجه‌اند. در مسیرهای فرا-قمری^۱ و به طور خاص در ناحیه‌های ناپایدار نقاط لاگرانژ در چارچوب مسئله‌ی سه جرمی محدود و دایروی^۲، طراحی سامانه‌ی کنترل مستلزم توانایی تضمین همزمان پایداری ایستا و بهره‌وری سوخت با نیروی پیشران کم^۳ است.

هم‌راستا با این تحولات، ظهور و گسترش الگوریتم‌های یادگیری تقویتی عمیق^۴، امکانات نوینی را برای طراحی کنترل‌کننده‌های تطبیقی فراهم آورده است. با این حال، غالب رویکردهای رایج بر سناریوهای تک‌عاملی و اتکا به مدل‌های دینامیکی دقیق استوارند. غیاب یک استراتژی مقاوم در مواجهه با اغتشاشات مدل و تغییرات

¹Trans-lunar

²Circular Restricted Three-Body Problem (CRTBP)

³Low-thrust

⁴Deep Reinforcement Learning (DRL)

محیطی از جمله خطای تراست در پیشران و تأخیر در سیگنال‌های حسگر منجر به فاصله‌ی چشمگیر عملکرد واقعی از پیش‌بینی‌های حاصل از شبیه‌سازی‌های ایده‌آل می‌گردد. این پژوهش بر آن است تا گسست اشاره‌شده را با بهره‌گیری از چارچوب یادگیری تقویتی چندعاملی مقاوم مرتفع سازد و بدین وسیله، اطمینان هدایت پیشران کم در مسئله‌ی سه‌جسمی محدود دایره‌ای^۵ را افزایش دهد.

۲-۱ تعریف مسئله

در سال‌های اخیر، پیشرفت‌های فناوری در زمینه‌های مختلف، از جمله کنترل پرواز، پردازش سیگنال و هوش مصنوعی، به افزایش کاربردهای ماهواره با پیشران کم در منظومه زمین-ماه کمک کرده است. ماهواره با پیشران کم می‌تواند برای تعقیب ماهواره‌ها، انتقال مداری و استقرار ماهواره‌ها استفاده شود. روش‌های هدایت بهینه قدیمی جهت کنترل ماهواره‌ها اغلب نیازمند فرضیات ساده‌کننده، منابع محاسباتی فراوان و شرایط اولیه مناسب هستند. الگوریتم‌های مبتنی بر یادگیری تقویتی این توانایی را دارند که بدون مشکلات اشاره‌شده هدایت ماهواره را انجام دهند. به دلیل ساختار شبکه‌ای، این الگوریتم‌ها می‌توانند امکان محاسبات درونی^۶ را فراهم کنند.

هدف از این پژوهش، طراحی سیاست کنترلی برای یک فضاپیما به جرم m است که در میدان جاذبه‌ی سیستم زمین-ماه به صورت دو بُعدی مدل شده است. ویژگی‌های این سامانه در ادامه آورده شده است.

- **پویایی‌ها:** معادلات حرکت در چارچوب مرجع چرخان به صورت مجموعه غیرخطی $\dot{x} = f(x) + g(x)u$ نوشته می‌شود که $x = [x, y, \dot{x}, \dot{y}]^T$ بردار حالت و u بردار تراست با کران $u \leq u_{\max}$ است.

- **عدم قطعیت‌ها:** عوامل عدم قطعیت شامل شرایط اولیه تصادفی، اغتشاش در عملگرها، عدم تطابق مدل، مشاهده ناقص، نویز حسگر و تأخیر زمانی هستند که همگی بر عملکرد و پایداری سیستم تأثیر می‌گذارند.

- **صورت بازی دیفرانسیلی:** فضاپیما و طبیعت (اغتشاشات) به ترتیب به عنوان عامل کنترلی و حریف مزاحم مدل شده است. مسئله به عنوان یک بازی مجموع صفر^۷ در افق زمان محدود t_f فرمول‌بندی شده‌اند.

صورت کامل مسأله را می‌توان با یافتن سیاست $\pi^* : \mathcal{X} \rightarrow \mathcal{U}$ تعریف کرد که معیار بهینه‌سازی هزینه‌ی تجمعی است.

⁵CRTBP

⁶On-board Computing

⁷Zero-Sum

۳-۱ یادگیری تقویتی

یادگیری تقویتی^۸ شاخه‌ای از یادگیری ماشین است که در آن یک عامل از طریق تعامل پیاپی با محیط می‌آموزد چه توالی اقدام‌هایی $a_t \in \mathcal{A}$ را انتخاب کند تا بازده تجمعی آینده را بیشینه کند. یک فرایند تصمیم‌گیری مارکوف^۹ به صورت $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ تعریف می‌شود که در آن

• \mathcal{S} : مجموعه حالات،

• $p(s'|s, a)$: دینامیک انتقال،

• $r(s, a)$: پاداش آنی،

• $\gamma \in [0, 1)$: ضریب تنزیل.

سیاست^{۱۰} $\pi(a|s)$ احتمال انتخاب اقدام a در وضعیت s را بیان می‌کند. هدف بیشینه‌سازی برگشت^{۱۱}

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (1-1)$$

است. روش‌های RL معمولاً در دو دسته‌ی ارزش‌محور (مانند Q-learning و DQN) و سیاست‌محور (مانند REINFORCE) جای می‌گیرند؛ ترکیب این دو به چارچوب Actor-Critic منتهی می‌شود که در آن یک بازیکن (Actor) سیاست را به‌روزرسانی می‌کند و یک منتقد (Critic) ارزش یا Q را برآورد می‌نماید [۲].

در حضور فضاهای پیوسته‌ی حالت-عمل، الگوریتم‌های گرادیان سیاست عمیق مانند DDPG، TD3، SAC و PPO با تکیه بر شبکه‌های عصبی به‌عنوان تقریب‌گر توابع، کارایی بالایی نشان داده‌اند. در این پژوهش، خانواده‌ی Actor-Critic پایه‌ی توسعه‌ی کنترل‌کننده پیشنهاد شده‌است.

۴-۱ یادگیری تقویتی چندعاملی

در یادگیری تقویتی چندعاملی^{۱۲}، فضای تصمیم‌گیری به‌صورت یک بازی مارکوفی^{۱۳} با مجموعه‌ی عامل‌ها $\mathcal{N} = \{1, \dots, N\}$ مدل شده‌است که در آن هر عامل سیاست π_i مختص خود را با هدف بیشینه‌سازی پاداش تجمعی

⁸Reinforcement Learning (RL)

⁹Markov Decision Process (MDP)

¹⁰Policy

¹¹Return

¹²Multi-Agent Reinforcement Learning (MARL)

¹³Markov Games (MG)

کسب می‌کند. در سناریوهای رقابتیِ دونفره، این پژوهش از چارچوب بازی‌های Zero-Sum استفاده شده و مفهوم تعادل نش^{۱۴} به‌عنوان معیار پایداری سیاست مطرح شده‌است.

رویکرد آموزش متمرکز، اجرا توزیع‌شده^{۱۵} با جداکردن مرحله‌ی آموزش که در آن اطلاعات خصوصی همهٔ عامل‌ها در دسترس است از اجرا که در آن هر عامل صرفاً بر مشاهده‌ی محلی اتکا می‌کند، تعادل بین کارایی و مقیاس‌پذیری را برقرار می‌سازد. این معماری مخصوصاً در حضور تعامل‌های ضعیف عامل‌ها مفید است، زیرا هزینه‌ی ارتباطی در زمان اجرا را حذف می‌کند.

۵-۱ محتوای گزارش

¹⁴Nash Equilibrium

¹⁵Centralized Training with Decentralized Execution (CTDE)

فصل ۲

پیشینه پژوهش

۱-۲ ماموریت‌های بین مداری

هدایت فضاپیماها معمولاً با استفاده از ایستگاه‌های زمینی انجام می‌شود. با این حال، این تکنیک‌ها دارای محدودیت‌هایی از جمله حساسیت به قطع ارتباطات، تاخیرهای زمانی و محدودیت‌های منابع محاسباتی هستند. الگوریتم‌های یادگیری تقویتی و بازی‌های دیفرانسیلی می‌توانند برای بهبود قابلیت‌های هدایت فضاپیماها، از جمله مقاومت در برابر تغییرات محیطی، کاهش تاخیرهای ناشی از ارتباطات زمینی و افزایش کارایی محاسباتی، مورد استفاده قرار گیرند.

هدایت فضاپیماها معمولاً پیش از پرواز انجام می‌شود. این روش‌ها می‌توانند از تکنیک‌های بهینه‌سازی فراگیر [۳] یا برنامه‌نویسی غیرخطی برای تولید مسیرها و فرمان‌های کنترلی بهینه استفاده کنند. با این حال، این روش‌ها معمولاً حجم محاسباتی زیادی دارند و برای استفاده درون‌سفینه‌ای نامناسب هستند [۴]. یادگیری ماشین می‌تواند برای بهبود قابلیت‌های هدایت فضاپیماها استفاده شود. کنترل‌کننده شبکه عصبی حلقه‌بسته می‌تواند برای محاسبه سریع و خودکار تاریخچه کنترل استفاده شود. یادگیری تقویتی نیز می‌تواند برای یادگیری رفتارهای هدایت بهینه استفاده شود.

روش‌های هدایت و بهینه‌سازی مسیر فضاپیماها به‌طور کلی به راه‌حل‌های اولیه مناسب نیاز دارند. در مسائل چند جسمی، طراحان مسیر اغلب حدس‌های اولیه کم‌هزینه‌ای برای انتقال‌ها با استفاده از نظریه سیستم‌های دینامیکی و منیفولدهای ثابت [۵، ۶] ایجاد می‌کنند.

شبکه‌های عصبی ویژگی‌های جذابی برای فعال‌سازی انجام هدایت در فضاپیما دارند. به‌عنوان مثال، شبکه‌های عصبی می‌توانند به‌طور مستقیم از تخمین‌های وضعیت به دستورهای پیش‌ران کنترلی که با محدودیت‌های

مأموریت سازگار است، برسند. عملکرد هدایت شبکه‌های عصبی در مطالعاتی مانند فرود بر سیارات [۷]، عملیات نزدیکی به سیارات [۸] و کنترل فضاپیما با پیشران ازدست‌رفته [۹] نشان داده شده‌است. تازه‌ترین پیشرفت‌های تکنیک‌های یادگیری ماشین در مسائل خودکارسازی درونی به‌طور گسترده‌ای مورد مطالعه قرار گرفته‌اند؛ از پژوهش‌های اولیه تا توانایی‌های پیاده‌سازی. به‌عنوان مثال، الگوریتم‌های یادگیری ماشین ابتدایی در فضاپیماهای مریخی‌نورد برای کمک به شناسایی ویژگی‌های زمین‌شناسی تعبیه شده‌اند. الگوریتم AEGIS توانایی انتخاب خودکار هدف توسط یک دوربین در داخل فضاپیماهای Spirit، Opportunity و Curiosity را فعال دارد [۱۰]. در کامپیوتر پرواز اصلی، فرآیند دقت‌افزایی (Refinement Process) نیاز به ۹۴ تا ۹۶ ثانیه دارد [۱۱]، که به‌طور قابل توجهی کمتر از زمان مورد نیاز برای ارسال تصاویر به زمین و انتظار برای انتخاب دستی توسط دانشمندان است. برنامه‌های آینده برای کاربردهای یادگیری ماشین درون‌سفینه شامل توانایی‌های رباتیکی درون‌سفینه برای فضاپیمای Perseverance [۱۲، ۱۳] و شناسایی عیب برای Europa Clipper [۱۴] می‌شود. الگوریتم‌های یادگیری ماشین دارای پتانسیلی انجام سهم مهمی در مأموریت‌های اتوماسیون آینده دارند.

علاوه بر رباتیک سیاره‌ای، پژوهش‌های مختلفی به استفاده از تکنیک‌های مختلف یادگیری ماشین در مسائل نجومی پرداخته‌اند. در طراحی مسیر عملکرد رگرسیون معمولاً مؤثرتر هست. به‌عنوان مثال، از یک شبکه عصبی (NN) در بهینه‌سازی مسیرهای رانشگر کم‌پیشران استفاده شده‌است [۱۵]. پژوهش‌های جدید شامل شناسایی انتقال‌های هتروکلینیک [۱۶]، اصلاح مسیر رانشگر کم‌پیشران [۱۷] و تجزیه و تحلیل مشکلات ازدست‌رفتن رانشگر [۹] می‌شود.

تکنیک‌های یادگیری نظارتی می‌توانند نتایج مطلوبی تولید کنند؛ اما، دارای محدودیت‌های قابل توجهی هستند. یکی از این محدودیت‌ها این است که این رویکردها بر وجود دانش پیش از فرآیند تصمیم‌گیری متکی هستند. این امر مستلزم دقیق‌بودن داده‌های تولیدشده توسط کاربر برای نتایج مطلوب و همچنین وجود تکنیک‌های موجود برای حل مشکل کنونی و تولید داده است.

در سال‌های اخیر، قابلیت یادگیری تقویتی (RL) در دستیابی به عملکرد بهینه در بخش‌هایی با ابهام محیطی قابل توجه، به اثبات رسیده است [۱۸، ۱۹]. هدایت انجام‌شده توسط RL را می‌توان به‌صورت گسترده بر اساس فاز پرواز دسته‌بندی کرد. مسائل فرود [۲۰، ۲۱] و عملیات در نزدیکی اجسام کوچک [۷، ۸]، از حوزه‌های پژوهشی هستند که از RL استفاده می‌کنند. تحقیقات دیگر شامل مواجهه تداخل خارجی جوی [۲۲]، نگهداری ایستگاهی [۲۳] و هدایت به‌صورت جلوگیری از شناسایی [۲۴] است. مطالعاتی که فضاپیماهای رانشگر کم‌پیشران را در یک چارچوب دینامیکی چند بدنی با استفاده از RL انجام‌شده‌است، شامل طراحی انتقال با استفاده از Q-learning [۲۵]، Proximal Policy Optimization [۲۶] و هدایت نزدیکی مدار [۲۷] است.

۲-۲ یادگیری تقویتی

از نخستین صورت‌بندی‌های فرایند تصمیم‌گیری مارکوفی در یادگیری تقویتی، پژوهش بر آن بوده‌است که عامل بتواند با اجرای عمل‌ها و دریافت پاداش، سیاستی برای بیشینه‌سازی برگشت بیاموزد. تبیین جامع این چارچوب و الگوریتم‌های بنیادین در ویرایش دوم کتاب سوتون و بارتو به‌مثابه مرجع کلاسیک این حوزه ارائه شده و همچنان مبنای بسیاری از آثار معاصر است [۲].

دهه‌ی ۱۹۹۰ ملادی شاهد شکل‌گیری روش‌هایی بر پایه‌ی ارزش^۱ نظیر Q-learning و نخستین رویکردهای گرادیان سیاست بود؛ با وجود این، محدودیت توان محاسباتی و فقدان داده‌ی فراوان، سرعت رشد را کند می‌کرد. ورود شبکه‌های عصبی عمیق نقطه‌ی عطفی بود: مقاله‌ی معروف دیپ‌ماینده^۲ نشان داد که شبکه‌ی Q عمیق (DQN) می‌تواند صرفاً از پیکسل‌های بازی آتاری سیاستی نزدیک به انسان بیاموزد [۲۸].

موفقیت DQN نگاه‌ها را به‌سوی گرادیان سیاست^۳ مقیاس‌پذیر معطوف ساخت. بهینه‌سازی ناحیه‌ی اطمینان^۳ تضمین بهبود یکنواخت سیاست را فراهم کرد [۲۹]، و روش A3C با موازی‌سازی بازیگران، سرعت یادگیری را چند برابر افزایش داد [۳۰]. کمی بعد، DDPG اولین بار گرادیان سیاست قطعی را به فضاهای عمل پیوسته وارد کرد [۳۱]. سپس PPO با ساده‌سازی قیود TRPO و کاهش پارامترهای حساس، به انتخاب پیش‌فرض بسیاری از کاربردهای مهندسی بدل شد [۳۲].

با گسترش دامنه‌ی مسائل، پایداری و کارایی داده به چالش اصلی بدل گشت. TD3 نشان داد که کمینه‌کردن میان دو منتقد می‌تواند برآورد بیش‌ازحد Q را مهار کند [۳۳]، و SAC با افزودن بند آنتروپی، هم‌زمان اکتشاف و بازده را بهبود داد [۳۴].

در محیط‌های پرخطر یا گران، جمع‌آوری داده‌ی برخت ناممکن است؛ از این‌رو RL آفلاین مطرح شد. روش CQL با برقراری کران محافظه‌کارانه بر Q-value از گرایش خارج از توزیع جلوگیری می‌کند [۳۵]، و مرور اخیر پراودنسیو و همکاران طبقه‌بندی جامعی از چالش‌های باز این حوزه ارائه داده است [۳۶].

هم‌زمان، دغدغه‌ی ایمنی و تبیین در سامانه‌های واقعی پررنگ شد. مرور سال ۲۰۲۲ نشان می‌دهد که ترکیب قیده‌های سخت، توابع جریمه‌ی ریسک و شبیه‌سازی محیط‌های بدبینانه سه خط اصلی ایمنی در RL هستند [۳۷]. سلسله‌مراتب نیز با هدف انتقال دانش و تسریع یادگیری مورد توجه قرار گرفت و یک مطالعه‌ی جامع در ACM Computing Surveys چهار چالش کشف زیرکار، یادگیری اشتراک‌پذیر، انتقال و مقیاس‌پذیری را برجسته می‌کند [۳۸].

وقتی چند عامل به‌طور هم‌زمان یاد می‌گیرند، پویایی محیط از دید هر عامل غیرایستا می‌شود. مرور جامع

¹Value

²DeepMind

³Trust Region Policy Optimization (TRPO)

۲۰۲۴ نشان می‌دهد که چارچوب ناظر متمرکز - بازیگر توزیع‌شده^۴ راهکاری موثر برای این چالش است و مباحثی چون تخصیص اعتبار جمعی و کشف تعادل را معرفی می‌کند [۳۹].

پیشرفت‌های یادشده در نهایت به دستاوردهای نمادینی چون AlphaGo [۴۰] و AlphaStar [۴۱] انجامیدند که در بازی‌های Go و StarCraft II از انسان پیشی گرفتند، و معماری توزیع‌شده IMPALA نشان داد که چگونه می‌توان هزاران شبیه‌ساز را با به‌روزرسانی وزن‌های مهم ادغام کرد [۴۲].

به‌رغم این جهش‌ها، سه شکاف اساسی پابرجا مانده است: (۱) تضمین ایمنی سخت‌گیرانه در سناریوهای نزدیک‌برخورد، (۲) کاهش وابستگی به داده‌ی پرهزینه یا نایاب از طریق روش‌های مدل‌منا و آفلاین، و (۳) مقیاس‌پذیری یادگیری چندعاملی برای سامانه‌های رباتیکی یا فضایی چندگانه.

۳-۲ پیشینه‌ی پژوهش یادگیری تقویتی چندعاملی

امروز یادگیری تقویتی چندعاملی (MARL) به‌عنوان بنیاد اصلی سامانه‌های هوشمند مشارکتی شناخته می‌شود؛ مسیری که از آزمون‌های ساده‌ی دوعاملی در دهه‌ی ۱۹۹۰ آغاز شد و اکنون به معماری‌های توزیع‌شده‌ی در مقیاس هزاران بازیگر رسیده است. این بخش، به بررسی اینکه چگونه ایده‌ی آموزش متمرکز - اجرای توزیع‌شده (CTDE) به پاسخ غالب برای چالش‌های غیریستایی و انفجار بُعدی بدل شد و چه گام‌هایی هنوز برای ایمنی، ناهمگونی و مقیاس‌پذیری باقی مانده است.

دهه‌ی ۱۹۹۰ با مقاله‌ی [۴۳] آغاز شد؛ جایی که برای نخستین‌بار مقایسه‌ی عامل‌های مستقل با عامل‌های همکار انجام شد و سود ارتباط و اشتراک تجربه به‌صورت تجربی نشان داده شد. در میانه‌ی دهه‌ی بعد، مرور جامع پانایت و لوک [۴۴] چشم‌اندازی از مسائل تخصیص اعتبار و غیریستایی ترسیم کرد و دو موضوع یادگیری تیمی و یادگیری هم‌زمان را صورت‌بندی نمود. هم‌زمان، بوشونیو و همکاران [۴۵] ادبیات MARL را در قالب اهداف پایداری دینامیک یادگیری و انطباق با رفتار سایر عامل‌ها جمع‌بندی کردند و راه را برای تحلیل‌های بازی‌محور هموار ساختند.

ورود شبکه‌های عمیق در سال‌های ۲۰۱۶ و ۲۰۱۷ نقطه‌ی عطف بعدی بود؛ منتقد متمرکز - بازیگر توزیع‌شده در MADDPG [۴۶] نشان داد که می‌توان از حالت سراسری در فاز آموزش بهره برد، اما سیاست نهایی را صرفاً بر اساس مشاهدات محلی اجرا کرد. در همان سال، Value-Decomposition Networks [۴۷] ایده‌ی تجزیه‌ی خطی پاداش را برای تیم‌های کاملاً تعاونی مطرح کرد و راه را برای فاکتوربندی‌های پیش‌رفته گشود.

۲۰۱۸ شاهد جهش مهمی با QMIX بود؛ این روش با اعمال قید تک‌نوا^۵ بر ترکیب مقادیر منفرد، هم

^۴Centralized Training with Decentralized Execution (CTDE)

^۵Monotonic

امکان بهینه‌سازی آف‌پالیسی را فراهم کرد و هم تضمین سازگاری سیاست‌های محلی با ارزش مشترک را برقرار ساخت [۴۸].

سال ۲۰۱۹ به گسترش بسترهای آزمایش اختصاص یافت. چالش استاندارد StarCraft Multi-Agent Challenge (SMAC) بر مبنای StarCraft II معرفی شد و معیار مشترکی برای مقایسه‌ی الگوریتم‌ها را مهیا کرد [۴۹]. هم‌زمان، QTRAN [۵۰] نشان داد که می‌توان بدون قید خطی یا تک‌نوا، تابع ارزش مشترک را به فضای قابل تجزیه تبدیل کرد. از سوی دیگر، MAVEN با افزودن متغیر نهفته‌ی مشترک، کاوش هماهنگ و سلسله‌مراتبی را امکان‌پذیر ساخت [۵۱]. نقطه‌ی اوج همان سال، سامانه‌ی AlphaStar بود که نشان داد ترکیب خودبازی و معماری توزیع‌شده می‌تواند به رتبه‌ی استاد بزرگ^۶ انسان برساند [۴۱].

در ۲۰۲۰ مفهوم نقش‌های در حال ظهور با ROMA [۵۲] معرفی شد تا عامل‌ها بر اساس شباهت رفتاری به‌طور خودکار خوشه‌بندی و اشتراک دانش کنند؛ رویکردی که در نقشه‌های پرتراکم SMAC برتری محسوسی نشان داد. پژوهش‌های متا در ۲۰۲۱، از مرور نظری زانگ و بشار [۵۳] تا محک^۷ تطبیقی پاپوداکیس و همکاران [۵۴]، شکاف‌های باقی‌مانده در تضمین همگرایی و مقیاس را فهرست کردند.

آخرین موج مطالعات بر ناهمگونی و ایمنی تمرکز دارد. مرور جامع [۵۵] نشان می‌دهد که تفاوت در قابلیت‌ها و اطلاعات عامل‌ها، مسائلی نظیر تخصیص اعتبار و تعادل را پیچیده‌تر می‌سازد و به الگوریتم‌های سازگار با نقش‌های پویا نیاز دارد.

به‌طور خلاصه، مسیر تاریخی MARL از الگوهای مستقل دهه‌ی ۱۹۹۰ به سامانه‌های توزیع‌شده‌ی امروزی، همواره با سه دغدغه‌ی اصلی هدایت شده است: کنترل انفجار بُعدی توابع ارزش، مقابله با غیریستایی ناشی از یادگیری هم‌زمان، و انتقال مؤثر تجربه میان عامل‌ها. علی‌رغم پیشرفت‌های شتابان، تضمین ایمنی سخت‌گیرانه در محیط‌های شکست‌پذیر، مدیریت نقش‌های پویا در تیم‌های ناهمگون و کاهش نیاز به داده‌ی شبیه‌سازی پرهزینه همچنان چالش‌های باز باقی می‌مانند؛ چالش‌هایی که در این پژوهش با رویکرد ترکیبی مدل‌مبنا، مقاوم و چندعاملی پیگیری می‌شوند.

^۶Grandmaster

^۷Benchmark

فصل ۳

مدل سازی محیط یادگیری سه جسمی

مسیرهای فضایی پیشین تحت تأثیر گرانش یک جسم مرکزی (خورشید، زمین یا سیاره‌ای دیگر) شکل می‌گیرند و توسط اجسام سوم تحت تأثیر قرار می‌گیرند. در برخی موارد، مأموریت فضایی آن را در ناحیه‌ای از فضا قرار می‌دهد که به‌طور همزمان تحت تأثیر دو جسم بزرگ است. این مسیرها نمی‌توانند از تحلیل دو جسم با اختلالات جسم سوم استفاده کنند، بلکه باید تأثیرات هر دو جسم به‌طور همزمان در نظر گرفته شوند.

مسئله سه جسمی محدود که شامل دو جسم اصلی با جرم‌های بزرگ و یک جسم کوچک (فضایما) است، محیطی مناسب برای بکارگیری روش‌های یادگیری تقویتی محسوب می‌شود. در این مسئله، دینامیک غیرخطی پیچیده‌ای حاکم است که نقاط تعادل خاصی به نام نقاط لاگرانژ در آن وجود دارد

در این فصل ابتدا به مدل سازی ریاضی مسئله سه جسمی محدود و استخراج معادلات حرکت در سیستم مختصات دوران در بخش ۱-۳ پرداخته شده است. سپس، نقاط لاگرانژ و خصوصیات پایداری آنها در بخش ۲-۳ مورد بررسی قرار گرفته.

۱-۳ مسئله‌ی سه جسمی محدود دایره‌ای (CRTBP)

دو جرم اصلی (زمین با جرم m_1 و ماه با جرم m_2) روی مدارهایی دایره‌ای و هم‌صفحه پیرامون مرکز جرم مشترک حرکت می‌کنند. جرم سوم (فضایما با جرم ناچیز m_3) چنان کوچک فرض می‌شود که تأثیر گرانشی آن بر حرکت دو جسم اصلی قابل نظر است؛ بدین ترتیب مسئله‌ی سه جسمی محدود دایره‌ای شکل می‌گیرد.

جدول ۱-۳: مقادیر عددی برای مسئله سه جسمی محدود (سیستم زمین-ماه)

پارامتر	توصیف	مقدار عددی
m_1	جرم زمین	$5.972 \times 10^{24} \text{ kg}$
m_2	جرم ماه	$7.348 \times 10^{22} \text{ kg}$
μ	نسبت جرمی	0.0121505856
ω	سرعت زاویه‌ای سیستم	$2.6617 \times 10^{-6} \text{ rad/s}$

دستگاه مختصات چرخانی هم‌دوران با دو جرم اصلی انتخاب می‌شود؛ مبدأ در مرکز جرم سامانه است، محور x خطِ واصلِ دو جرم و محور y بر آن عمود (در صفحه‌ی مدارها) است. واحدِ طول برابر فاصله‌ی ثابتِ میان دو جرم و واحدِ زمان چنان تعریف می‌شود که دوره‌ی مداری سامانه 2π (و در نتیجه $\omega = 1$) گردد. همچنین جرم‌ها به گونه‌ای مقیاس می‌شود که مجموع دو جرم برابر با یک شود.

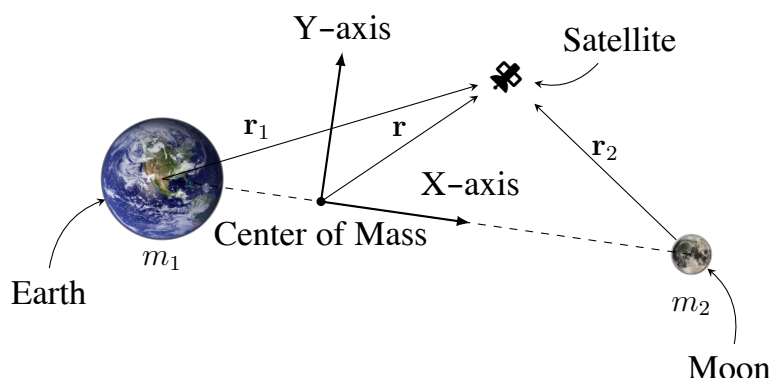
$$m_1 + m_2 = 1. \quad (1-3)$$

با نسبتِ جرمی

$$\mu \equiv \frac{m_2}{m_1 + m_2}, \quad (2-3)$$

داریم $m_1 = 1 - \mu$ و $m_2 = \mu$ و مکانِ دو جرم در دستگاه بی‌بُعد به صورت

$$\mathbf{r}_{\text{Earth}} = (-\mu, 0), \quad \mathbf{r}_{\text{Moon}} = (1 - \mu, 0). \quad (3-3)$$



شکل ۱-۳: هندسه مسئله سه بدنه محدود

۱-۱-۳ لاگرانژ و معادلات حرکت

با در نظر گرفتن $G = 1$ در حالت بی‌بعد، تابع لاگرانژ جرم سوم در دستگاه چرخان برابر است با [۵۶]

$$L = \frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + (1 - \mu) \frac{1}{r_1} + \mu \frac{1}{r_2} + \frac{1}{2}(x^2 + y^2), \quad (۴-۳)$$

که در آن

$$r_1 = \sqrt{(x + \mu)^2 + y^2 + z^2}, \quad r_2 = \sqrt{(x - 1 + \mu)^2 + y^2 + z^2}. \quad (۵-۳)$$

با به‌کارگیری رابطه‌ی اوایلر-لاگرانژ

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0, \quad q_i \in \{x, y, z\},$$

معادلات بی‌بعد حرکت جرم سوم به دست می‌آید:

$$\ddot{x} - 2\dot{y} = x - \frac{1 - \mu}{r_1^3}(x + \mu) - \frac{\mu}{r_2^3}(x - 1 + \mu), \quad (۶-۳)$$

$$\ddot{y} + 2\dot{x} = y - \frac{1 - \mu}{r_1^3}y - \frac{\mu}{r_2^3}y, \quad (۷-۳)$$

$$\ddot{z} = -\frac{1 - \mu}{r_1^3}z - \frac{\mu}{r_2^3}z. \quad (۸-۳)$$

یا به نگاشت برداری به صورت زیر است.

$$\ddot{\mathbf{r}} + 2\boldsymbol{\omega} \times \dot{\mathbf{r}} = \nabla \Omega(\mathbf{r}), \quad \Omega(x, y, z) = \frac{1}{2}(x^2 + y^2) + \frac{1 - \mu}{r_1} + \frac{\mu}{r_2}. \quad (۹-۳)$$

۲-۳ نقاط تعادل لاگرانژ

نقطه‌ی تعادل مکانی است که در چارچوب چرخان، جرم سوم بی‌حرکت بماند. این شرایط با صفرشدن مؤلفه‌های سرعت و شتاب به دست می‌آید؛ از این رو در معادلات بالا قرار می‌دهیم $\dot{x} = \dot{y} = \dot{z} = \ddot{x} = \ddot{y} = \ddot{z} = 0$. در نتیجه دستگاه جبری زیر برای مختصات نقطه‌ی تعادل حاصل می‌شود:

$$0 = x - \frac{1 - \mu}{r_1^3}(x + \mu) - \frac{\mu}{r_2^3}(x - 1 + \mu), \quad (۱۰-۳)$$

$$0 = y \left[1 - \frac{1 - \mu}{r_1^3} - \frac{\mu}{r_2^3} \right], \quad (۱۱-۳)$$

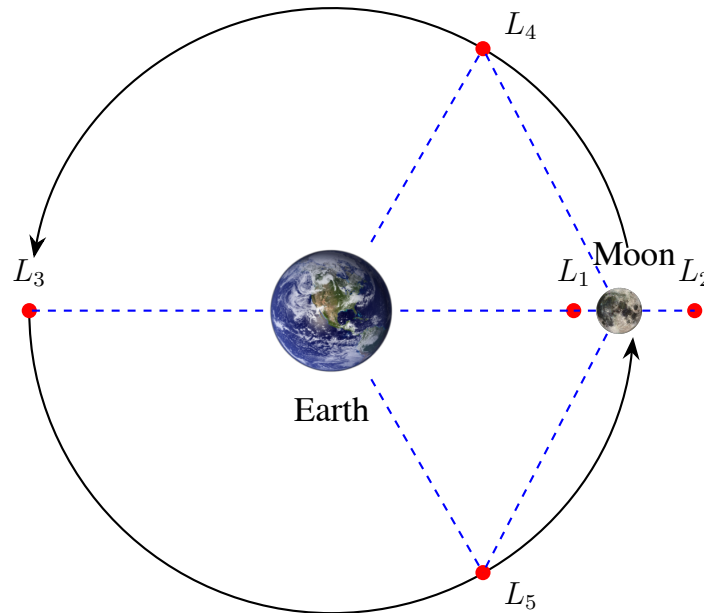
$$0 = -\frac{1 - \mu}{r_1^3}z - \frac{\mu}{r_2^3}z. \quad (۱۲-۳)$$

معادله‌ی سوم نشان می‌دهد برای حالت عمومی باید $z = 0$ باشد؛ بنابراین نقاط تعادل همگی در صفحه‌ی مدار قرار می‌گیرند.

دسته‌بندی کلی

۱. نقاط هم‌خط (Collinear). سه نقطه‌ی L_1 ، L_2 و L_3 روی خطِ واصلِ دو جرم قرار دارند و لذا $y = 0$ است.

۲. نقاط سه‌گوش (Triangular). دو نقطه‌ی L_4 و L_5 رأس‌های مثلثِ متساوی‌الاضلاع با دو جرم اصلی را تشکیل می‌دهند و در آن‌ها $y \neq 0$.



شکل ۳-۲: نقاط لاگرانژ

نقاط هم‌خط (L_1, L_2, L_3)

با اعمال $y = 0$ ، تنها معادله‌ی زیر باقی می‌ماند

$$x - \frac{1 - \mu}{|x + \mu|^3}(x + \mu) - \frac{\mu}{|x - 1 + \mu|^3}(x - 1 + \mu) = 0. \quad (۱۳-۳)$$

این معادله در سه ناحیه‌ی مجزا—بین دو جرم، بیرونِ جرمِ کوچک و بیرونِ جرمِ بزرگ—دارای یک ریشه است که به ترتیب نقاط L_1 ، L_2 و L_3 را تعیین می‌کند.

برای $\mu \ll 1$ (همچون سامانه‌ی خورشید-زمین یا زمین-ماه) می‌توان تقریب‌های شناخته‌شده را نوشت:

$$x_{L_1} \simeq (1 - \mu) - \left(\frac{\mu}{3}\right)^{1/3},$$

$$x_{L_2} \simeq (1 - \mu) + \left(\frac{\mu}{3}\right)^{1/3},$$

$$x_{L_3} \simeq -1 - \frac{5}{12}\mu; \quad y_{L_i} = 0.$$

در عمل، ریشه‌ی دقیقِ معادله‌ی (۱۳-۳) با یک روشِ عددی (نیوتن-رافسون) محاسبه می‌شود.

نقاطِ سه‌گوش (L_4, L_5)

در این نقاط $r_1 = r_2 = 1$ و شرط $1 - (1 - \mu)/r_1^3 - \mu/r_2^3 = 0$ به طور طبیعی برقرار است. مختصات به سادگی عبارت‌اند از

$$x_{L_4} = x_{L_5} = \frac{1}{2} - \mu, \quad y_{L_4} = +\frac{\sqrt{3}}{2}, \quad y_{L_5} = -\frac{\sqrt{3}}{2}. \quad (۱۴-۳)$$

پایداری این نقاط مستلزم نسبتِ جرمِ کافی است؛ شرطِ کلاسیک $m_1/m_2 > 24.96$ در سامانه‌های خورشید-سیاره یا زمین-ماه به خوبی برقرار است و سببِ وجودِ خانواده‌ی سیارک‌های تروجان حول L_4 و L_5 می‌شود. در مقابل، نقاطِ هم‌خط ناپایدارند و معمولاً مأموریت‌های فضایی روی مدارهای هاله‌ای یا لیسائور در پیرامونِ آن‌ها قرار می‌گیرند.

برای سامانه‌ی زمین-ماه، $\mu \simeq 0.01215$ است. جدولِ زیر مختصاتِ بی‌بعدِ هر پنج نقطه را نشان می‌دهد (واحدِ طول: فاصله‌ی زمین-ماه). موقعیتِ زمین در $(-\mu, 0)$ و ماه در $(1 - \mu, 0)$ است.

جدول ۳-۲: مقادیر عددی نقاط لاگرانژ برای مسئله سه‌جسمی محدود سیستم زمین-ماه

نقطه‌ی لاگرانژ	x (بی‌بعد)	y (بی‌بعد)
L_1	+0.83692	0
L_2	+1.15568	0
L_3	-1.00506	0
L_4	+0.48785	+0.86603
L_5	+0.48785	-0.86603

این نتایج نشان می‌دهد که L_1 در حدودِ 0.84 فاصله‌ی زمین-ماه از زمین قرار دارد (فاصله‌ی آن تا ماه در حدود 0.16 واحد طول است) و L_2 بیرونِ مدارِ ماه است. نقطه‌ی L_3 تقریباً یک واحد طول در سوی مقابلِ ماه نسبت به زمین قرار دارد. دو نقطه‌ی L_4 و L_5 در مختصات $(0.488, \pm 0.866)$ قرار گرفته و با زمین و ماه مثلثِ متساوی‌الاضلاع می‌سازند.

فصل ۴

یادگیری تقویتی

در این فصل به بررسی یادگیری تقویتی پرداخته شده است. ابتدا در فصل ۴-۱ مفاهیم اولیه یادگیری تقویتی ارائه شده است. در ادامه عامل‌های گرادیان سیاست عمیق قطعی ۴-۲، گرادیان سیاست عمیق قطعی تاخیری دوگانه ۴-۳، عملگر نقاد نرم ۴-۴ و بهینه‌سازی سیاست مجاور ۴-۵ توضیح داده شده است.

۴-۱ مفاهیم اولیه

دو بخش اصلی یادگیری تقویتی^۱ شامل عامل^۲ و محیط^۳ است. عامل در محیط قرار دارد و با آن در تعامل است. در هر مرحله از تعامل بین عامل و محیط، عامل یک مشاهده جزئی از وضعیت محیط انجام می‌دهد و سپس در مورد اقدامی که باید انجام دهد، تصمیم می‌گیرد. وقتی عامل روی محیط عمل می‌کند، محیط تغییر می‌کند؛ اما، ممکن است محیط به تنهایی نیز تغییر کند. عامل همچنین یک سیگنال پاداش^۴ از محیط دریافت می‌کند؛ سیگنالی که به عامل می‌گوید وضعیت تعامل فعلی آن با محیط چقدر خوب یا بد است. هدف عامل بیشینه‌کردن پاداش انباشته خود است که برگشت^۵ نام دارد. یادگیری تقویتی به روش‌هایی گفته می‌شود که در آن‌ها عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۴-۱ تعامل بین محیط و عامل نشان داده شده است.

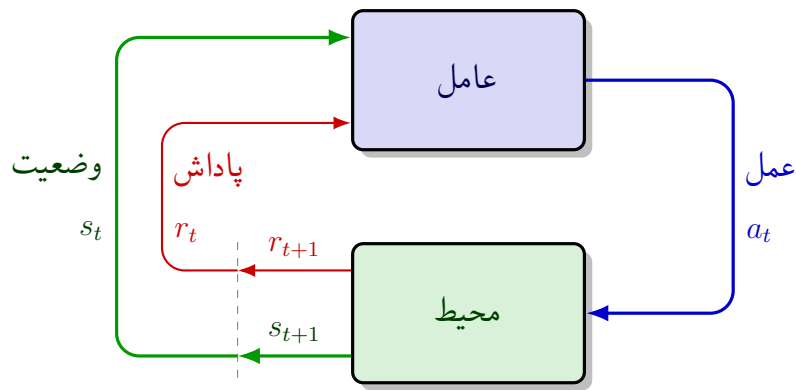
^۱ Reinforcement Learning (RL)

^۲ Agent

^۳ Environment

^۴ Reward

^۵ Return



شکل ۴-۱: حلقه تعامل عامل و محیط

۴-۱-۱ حالت و مشاهدات

حالت^۶ (s) توصیف کاملی از وضعیت محیط است. همه‌ی اطلاعات محیط در حالت وجود دارد. مشاهده^۷ (o) یک توصیف جزئی از حالت است که ممکن است شامل تمامی اطلاعات نباشد. در این پژوهش مشاهده توصیف کاملی از محیط هست؛ در نتیجه، حالت و مشاهده برابر هستند.

۴-۱-۲ فضای عمل

فضای عمل (a) در یادگیری تقویتی، مجموعه‌ای از تمام اقداماتی است که یک عامل می‌تواند در محیط انجام دهد. این فضا می‌تواند گسسته^۸ یا پیوسته^۹ باشد. در این پژوهش فضای عمل پیوسته و محدود به یک بازه مشخص است.

۴-۱-۳ سیاست

سیاست^{۱۰} قاعده‌ای است که یک عامل برای تصمیم‌گیری در مورد اقدامات خود استفاده می‌کند. در این پژوهش به تناسب الگوریتم پیاده‌سازی شده از سیاست قطعی^{۱۱} یا تصادفی^{۱۲} استفاده شده است که به دو صورت زیر نشان

^۶State

^۷Observation

^۸Discrete

^۹Continuous

^{۱۰}Policy

^{۱۱}Deterministic

^{۱۲}Stochastic

داده می‌شود:

$$a_t = \mu(s_t) \quad (۱-۴)$$

$$a_t \sim \pi(\cdot | s_t) \quad (۲-۴)$$

که زیروند t بیانگر زمان است. در یادگیری تقویتی عمیق از سیاست‌های پارامتری شده استفاده می‌شود. خروجی این سیاست‌ها تابعی پارامترهای سیاست (وزن‌ها و بایاس‌های یک شبکه عصبی) هستند که می‌توان از الگوریتم‌های بهینه‌سازی جهت تعیین مقدار بهینه این پارامترها استفاده کرد. در این پژوهش پارامترهای سیاست با θ نشان داده شده‌است و سپس نماد آن به‌عنوان زیروند سیاست مانند معادله (۳-۴) نشان داده شده‌است.

$$a_t = \mu_\theta(s_t) \quad (۳-۴)$$

$$a_t \sim \pi_\theta(\cdot | s_t)$$

۴-۱-۴ مسیر

یک مسیر^{۱۳} یک توالی از حالت‌ها و عمل‌ها در محیط است.

$$\tau = (s_0, a_0, s_1, a_1, \dots) \quad (۴-۴)$$

گذار حالت^{۱۴} به اتفاقاتی که در محیط بین زمان t در حالت s_t و زمان $t + 1$ در حالت s_{t+1} رخ می‌دهد، گفته می‌شود. این گذارها توسط قوانین طبیعی محیط انجام می‌شوند و تنها به آخرین اقدام انجام‌شده توسط عامل (a_t) بستگی دارند. گذار حالت را می‌توان به‌صورت زیر تعریف کرد.

$$s_{t+1} = f(s_t, a_t) \quad (۵-۴)$$

۵-۱-۴ تابع پاداش و برگشت

تابع پاداش^{۱۵} در حالت کلی به حالت فعلی محیط، آخرین عمل انجام‌شده و حالت بعدی محیط بستگی دارد. تابع پاداش را می‌توان به‌صورت زیر تعریف کرد.

$$r_t = R(s_t, a_t, s_{t+1}) \quad (۶-۴)$$

¹³Trajectory

¹⁴State Transition

¹⁵Reward Function

در این پژوهش، پاداش تنها تابعی از جفتِ حالت-عمل ($r_t = R(s_t, a_t)$) فرض شده‌است. هدف عامل این است که مجموع پاداش‌های به‌دست‌آمده در طول یک مسیر را به حداکثر برساند. در این پژوهش مجموع پاداش‌ها در طول یک مسیر را با نماد $R(\tau)$ نشان داده شده‌است و به آن تابع برگشت^{۱۶} گفته می‌شود. یکی از انواع برگشت، برگشت بدون تنزیل^{۱۷} با افق محدود^{۱۸} است که مجموع پاداش‌های به‌دست‌آمده در یک بازه زمانی ثابت و از مسیر τ است که در معادله (۷-۴) نشان داده شده‌است.

$$R(\tau) = \sum_{t=0}^T r_t \quad (۷-۴)$$

نوع دیگری از برگشت، برگشت تنزیل‌شده با افق نامحدود^{۱۹} است که مجموع همه پاداش‌هایی است که تا به حال توسط عامل به‌دست آمده‌است. اما، فاصله زمانی تا دریافت پاداش باعث تنزیل ارزش آن می‌شود. این معادله برگشت (۸-۴) شامل یک فاکتور تنزیل^{۲۰} با نماد γ است که عددی بین صفر و یک است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (۸-۴)$$

۴-۱-۶ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش^{۲۱} یک حالت یا جفتِ حالت-عمل ضروری است. منظور از ارزش، برگشت مورد انتظار^{۲۲} است. یعنی اگر از آن حالت یا جفتِ حالت-عمل شروع شود و سپس برای همیشه طبق یک سیاست خاص عمل شود، به‌طور میانگین چه مقدار پاداش دریافت خواهد شد. توابع ارزش تقریباً در تمام الگوریتم‌های یادگیری تقویتی به کار می‌روند. در اینجا به چهار تابع مهم اشاره شده‌است.

۱. تابع ارزش تحت سیاست^{۲۳} ($V^\pi(s)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود و همیشه طبق سیاست π عمل شود و به‌صورت زیر بیان می‌شود:

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s] \quad (۹-۴)$$

۲. تابع ارزش-عمل تحت سیاست^{۲۴} ($Q^\pi(s, a)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود، یک اقدام دلخواه a (که ممکن است از سیاست π نباشد) انجام شود و سپس

¹⁶Return

¹⁷Discount

¹⁸Finite-Horizon Undiscounted Return

¹⁹Infinite-Horizon Discounted Return

²⁰Discount Factor

²¹Value

²²Expected Return

²³On-Policy Value Function

²⁴On-Policy Action-Value Function

برای همیشه طبق سیاست π عمل شود و به صورت زیر بیان می شود:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a] \quad (۱۰-۴)$$

۳. تابع ارزش بهینه^{۲۵} ($V^*(s)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود و همیشه طبق سیاست بهینه در محیط عمل شود و به صورت زیر بیان می شود:

$$V^*(s) = \max_{\pi} (V^\pi(s)) \quad (۱۱-۴)$$

۴. تابع ارزش-عمل بهینه^{۲۶} ($Q^*(s, a)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود، یک اقدام دلخواه a انجام شود و سپس برای همیشه طبق سیاست بهینه در محیط عمل شود و به صورت زیر بیان می شود:

$$Q^*(s, a) = \max_{\pi} (Q^\pi(s, a)) \quad (۱۲-۴)$$

۷-۱-۴ معادلات بلمن

توابع ارزش اشاره شده از معادلات خاصی که به آن ها معادلات بلمن گفته می شود، پیروی می کنند. ایده اصلی پشت معادلات بلمن این است که ارزش نقطه شروع برابر است با پاداشی است که انتظار دارید از آنجا دریافت کنید، به علاوه ارزش مکانی که بعداً به آنجا می رسید. معادلات بلمن برای توابع ارزش سیاست محور به شرح زیر هستند:

$$V^\pi(s) = \mathbb{E}_{\substack{a \sim \pi \\ s' \sim P}} [r(s, a) + \gamma V^\pi(s')] \quad (۱۳-۴)$$

$$Q^\pi(s, a) = r(s, a) + \mathbb{E}_{s' \sim P} \left[\gamma \mathbb{E}_{a' \sim \pi} [Q^\pi(s', a')] \right] \quad (۱۴-۴)$$

که در آن $V^\pi(s)$ تابع ارزش حالت s تحت سیاست π است؛ $Q^\pi(s, a)$ تابع ارزش عمل a در حالت s تحت سیاست π است؛ $R(s, a)$ پاداش دریافتی پس از انجام عمل a در حالت s است؛ γ ضریب تنزیل است که ارزش پاداش های آینده را کاهش می دهد؛ $s' \sim P(\cdot | s, a)$ نشان می دهد که حالت بعدی s' از توزیع انتقال محیط P با شرط های s و a نمونه برداری می شود؛ و $a' \sim \pi(\cdot | s')$ نشان می دهد که عمل بعدی a' از سیاست

²⁵Optimal Value Function

²⁶Optimal Action-Value Function

π با شرط حالت جدید s' نمونه‌برداری می‌شود. این معادلات بیانگر این هستند که ارزش یک حالت یا عمل، مجموع پاداش مورد انتظار آن و ارزش حالت بعدی است که بر اساس سیاست فعلی تعیین می‌شود. معادلات بلمن برای توابع ارزش بهینه به شرح زیر هستند:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')] \quad (۱۵-۴)$$

$$Q^*(s, a) = r(s, a) + \mathbb{E}_{s' \sim P} \left[\gamma \max_{a'} Q^*(s', a') \right] \quad (۱۶-۴)$$

تفاوت حیاتی بین معادلات بلمن برای توابع ارزش سیاست محور و توابع ارزش بهینه، عدم حضور یا حضور عملگر max بر روی اعمال است. حضور آن منعکس‌کننده این است که هرگاه عامل بتواند عمل خود را انتخاب کند، برای عمل بهینه، باید هر عملی را که منجر به بالاترین ارزش می‌شود انتخاب کند.

۸-۱-۴ تابع مزیت

گاهی در یادگیری تقویتی، نیازی به توصیف میزان خوبی یک عمل به صورت مطلق نیست، بلکه تنها می‌خواهیم بدانیم که چه مقدار بهتر از سایر اعمال به‌طور متوسط است. به عبارت دیگر، مزیت نسبی آن عمل مورد بررسی قرار می‌گیرد. این مفهوم با تابع مزیت^{۲۷} توضیح داده می‌شود.

تابع مزیت $A^\pi(s, a)$ که مربوط به سیاست π است، توصیف می‌کند که انجام یک عمل خاص a در حالت s چقدر بهتر از انتخاب تصادفی یک عمل بر اساس $\pi(\cdot|s)$ است، با فرض اینکه شما برای همیشه پس از آن مطابق با π عمل می‌کنید. به صورت ریاضی، تابع مزیت به صورت زیر تعریف می‌شود:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

که در آن $A^\pi(s, a)$ تابع مزیت برای عمل a در حالت s است. $Q^\pi(s, a)$ تابع ارزش عمل a در حالت s تحت سیاست π است و $V^\pi(s)$ تابع ارزش حالت s تحت سیاست π است. این تابع مزیت نشان می‌دهد که انجام عمل a در حالت s نسبت به میانگین اعمال تحت سیاست π چقدر مزیت دارد. اگر $A^\pi(s, a)$ مثبت باشد، نشان‌دهنده این است که عمل a بهتر از میانگین اعمال است و اگر منفی باشد، نشان‌دهنده کمتر بودن عملکرد آن نسبت به میانگین است.

²⁷ Advantage Function

۲-۴ عامل گرادیان سیاست عمیق قطعی

گرادیان سیاست عمیق قطعی^{۲۸} الگوریتمی است که همزمان یک تابع Q و یک سیاست را یاد می‌گیرد. این الگوریتم برای یادگیری تابع Q از داده‌های غیرسیاست محور^{۲۹} و معادله بلمن استفاده می‌کند. این الگوریتم برای یادگیری سیاست نیز از تابع Q استفاده می‌کند.

این رویکرد وابستگی نزدیکی به یادگیری Q دارد. اگر تابع ارزش-عمل بهینه مشخص باشد، در هر حالت داده‌شده عمل بهینه را می‌توان با حل معادله (۱۷-۴) به دست آورد.

$$a^*(s) = \arg \max_a Q^*(s, a) \quad (۱۷-۴)$$

الگوریتم DDPG ترکیبی از یادگیری تقریبی برای $Q^*(s, a)$ و یادگیری تقریبی برای $a^*(s)$ است و به صورتی طراحی شده است که برای محیط‌هایی با فضاها عمل پیوسته مناسب باشد. آنچه این الگوریتم را برای فضای عمل پیوسته مناسب می‌کند، روش محاسبه $a^*(s)$ است. فرض می‌شود که تابع $Q^*(s, a)$ نسبت به آرگومان عمل مشتق‌پذیر است. مشتق‌پذیری این امکان را می‌دهد که یک روش یادگیری مبتنی بر گرادیان برای سیاست $\mu(s)$ استفاده شود. سپس، به جای اجرای یک بهینه‌سازی زمان‌بر در هر بار محاسبه $\max_a Q(s, a)$ ، می‌توان آن را با رابطه $\max_a Q(s, a) \approx Q(s, \mu(s))$ تقریب زد.

۱-۲-۴ یادگیری Q در DDPG

معادله بلمن که تابع ارزش عمل بهینه $(Q^*(s, a))$ را توصیف می‌کند، در پایین آورده شده است.

$$Q^*(s, a) = r(s, a) + \mathbb{E}_{s' \sim P} \left[\gamma \max_{a'} Q^*(s', a') \right] \quad (۱۸-۴)$$

عبارت $s' \sim P$ به این معنی است که وضعیت بعدی یعنی s' از توزیع احتمال $P(\cdot | s, a)$ نمونه گرفته می‌شود. در معادله بلمن نقطه شروع برای یادگیری $Q^*(s, a)$ یک مقداردهی تقریبی است. پارامترهای شبکه عصبی $Q_\phi(s, a)$ با علامت ϕ نشان داده شده است. مجموعه \mathcal{D} شامل اطلاعات جمع‌آوری شده تغییر از یک حالت به حالت دیگر (s, a, r, s', d) (که d نشان می‌دهد که آیا وضعیت s' پایانی است یا خیر) است. در بهینه‌سازی از تابع خطای میانگین مربعات بلمن^{۳۰} (MSBE) استفاده شده است که معیاری برای نزدیکی Q_ϕ به حالت بهینه برای برآورده کردن معادله بلمن است.

²⁸Deep Deterministic Policy Gradient (DDPG)

²⁹Off-Policy

³⁰Mean Squared Bellman Error

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left[\left(Q_\phi(s, a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s', a') \right) \right)^2 \right] \quad (۱۹-۴)$$

در الگوریتم DDPG دو ترفند برای عملکرد بهتر استفاده شده است که در ادامه به بررسی آن پرداخته شده است.

- بافرهای تکرار بازی

الگوریتم‌های یادگیری تقویتی جهت آموزش یک شبکه عصبی عمیق برای تقریب $Q^*(s, a)$ از بافرهای تکرار بازی^{۳۱} تجربه شده استفاده می‌کنند. این مجموعه \mathcal{D} شامل تجربیات قبلی عامل است. برای داشتن رفتار پایدار در الگوریتم، بافر تکرار بازی باید به اندازه کافی بزرگ باشد تا شامل یک دامنه گسترده از تجربیات شود. انتخاب داده‌های بافر به دقت انجام شده است چرا که اگر فقط از داده‌های بسیار جدید استفاده شود، بیش‌برازش^{۳۲} رخ می‌دهد و اگر از تجربه بیش از حد استفاده شود، ممکن است فرآیند یادگیری کند شود.

- شبکه‌های هدف

الگوریتم‌های یادگیری Q از شبکه‌های هدف استفاده می‌کنند. اصطلاح زیر به عنوان هدف شناخته می‌شود.

$$r + \gamma(1-d) \max_{a'} Q_\phi(s', a') \quad (۲۰-۴)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده است تا تابع Q شبیه‌تر به هدف یعنی رابطه (۲۰-۴) شود. اما مشکل این است که هدف بستگی به پارامترهای در حال آموزش ϕ دارد. این باعث ایجاد ناپایداری در کمینه کردن تابع خطای میانگین مربعات بلمن می‌شود. راه حل آن استفاده از یک مجموعه پارامترهایی است که با تأخیر زمانی به ϕ نزدیک می‌شوند. به عبارت دیگر، یک شبکه دوم ایجاد می‌شود که به آن شبکه هدف گفته می‌شود. شبکه هدف پارامترهای شبکه اول را با تأخیر دنبال می‌کند. پارامترهای شبکه هدف با نشان ϕ_{targ} نشان داده می‌شوند. در الگوریتم DDPG، شبکه هدف در هر به‌روزرسانی شبکه اصلی، با میانگین‌گیری پولیاک^{۳۳} به صورت زیر به‌روزرسانی می‌شود.

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi \quad (۲۱-۴)$$

در رابطه بالا ρ یک ابرپارامتر^{۳۴} است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک در نظر گرفته شده است.

³¹Replay Buffers

³²Overfit

³³Polyak Averaging

³⁴Hyperparameter

الگوریتم DDPG نیاز به یک شبکه سیاست هدف ($\mu_{\theta_{\text{targ}}}$) برای محاسبه عمل‌هایی که به‌طور تقریبی بیشینه $Q_{\phi_{\text{targ}}}$ را حاصل کند، را دارد. برای رسیدن به این شبکه سیاست هدف از همان روشی که تابع Q به دست می‌آید یعنی با میانگین‌گیری پولیاک از پارامترهای سیاست در طول زمان آموزش استفاده می‌شود.

با در نظر گرفتن موارد اشاره‌شده، یادگیری Q در DDPG با کمینه‌کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۲۲-۴) با استفاده از کاهش گرادیان تصادفی^{۳۵} انجام می‌شود.

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left[\left(Q_{\phi}(s, a) - (r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))) \right)^2 \right] \quad (22-4)$$

۲-۲-۴ سیاست در DDPG

در این بخش یک سیاست تعیین‌شده $\mu_{\theta}(s)$ یاد گرفته می‌شود تا عملی را انجام می‌دهد که بیشینه $Q_{\phi}(s, a)$ رخ دهد. از آنجا که فضای عمل پیوسته است و فرض شده‌است که تابع Q نسبت به عمل مشتق‌پذیر است، رابطه زیر با استفاده از صعود گرادیان^{۳۶} (تنها نسبت به پارامترهای سیاست) بیشینه می‌شود.

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\phi}(s, \mu_{\theta}(s))] \quad (23-4)$$

۳-۲-۴ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف^{۳۷} در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسندگان مقاله DDPG [۵۷] توصیه کرده‌اند که نویز OU^{۳۸} با هم‌بندی زمانی^{۳۹} اضافه شود. در زمان بهره‌برداری^{۴۰} سیاست، از آنچه یاد گرفته است، نویز به عمل‌ها اضافه نمی‌شود.

۴-۲-۴ شبکه‌کد DDPG

در این بخش، شبکه‌کد الگوریتم DDPG پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۱ در محیط پایتون با استفاده از کتابخانه TensorFlow [۵۸] پیاده‌سازی شده‌است.

³⁵Stochastic Gradient Descent

³⁶Gradient Ascent

³⁷Exploration

³⁸Ornstein-Uhlenbeck

³⁹Time-Correlated

⁴⁰Exploitation

ورودی: پارامترهای اولیه سیاست (θ) ، پارامترهای تابع $Q(\phi)$ ، بافر تکرار بازی خالی (\mathcal{D})

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\phi_{\text{targ}} \leftarrow \phi, \theta_{\text{targ}} \leftarrow \theta$

۲: تا وقتی همگرایی رخ دهد:

۳: وضعیت s را مشاهده کرده و عمل $a = \text{clip}(\mu_{\theta}(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید به طوری که $\epsilon \sim \mathcal{N}$ است.

۴: عمل a را در محیط اجرا کنید.

۵: وضعیت بعدی s' ، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا خیر.

۶: اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

۷: اگر زمان بهروزرسانی فرا رسیده است:

۸: به ازای هر تعداد بهروزرسانی:

۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$ ، از \mathcal{D} نمونه‌گیری شود.

۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

۱۱: تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر بهروزرسانی کنید:

$$\nabla_{\phi} \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_{\phi}(s, a) - y(r, s', d))^2$$

۱۲: سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر بهروزرسانی کنید:

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} Q_{\phi}(s, \mu_{\theta}(s))$$

۱۳: شبکه‌های هدف را با استفاده از معادلات زیر بهروزرسانی کنید:

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta$$

۳-۴ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه^{۴۱} یکی از الگوریتم‌های یادگیری تقویتی است که برای حل مسائل کنترل در محیط‌های پیوسته طراحی شده است. این الگوریتم بر اساس الگوریتم DDPG توسعه یافته و با استفاده از تکنیک‌های مختلف، پایداری و کارایی یادگیری را بهبود می‌بخشد. در حالی که DDPG گاهی اوقات می‌تواند عملکرد بسیار خوبی داشته باشد، اما اغلب نسبت به ابرپارامترها و سایر انواع تنظیمات یادگیری حساس است. یک حالت رایج شکست عامل DDPG در یادگیری این است که تابع Q یادگرفته شده شروع به بیش‌برآورد مقادیر Q می‌کند که منجر به واگرایی سیاست می‌شود. واگرایی به این دلیل رخ می‌دهد که در فرآیند یادگیری سیاست از تخمین تابع Q استفاده می‌شود که افزایش خطای تابع Q منجر به ناپایداری در یادگیری سیاست می‌شود.

الگوریتم TD3 (Twin Delayed DDPG) از دو ترفند زیر جهت بهبود مشکلات اشاره شده استفاده می‌کند.

- یادگیری دوگانه‌ی محدود شده^{۴۲}: الگوریتم TD3 به جای یک تابع Q ، دو تابع Q_{ϕ_1} و Q_{ϕ_2} را یاد می‌گیرد (از این رو دوگانه^{۴۳} نامیده می‌شود) و از کوچک‌ترین مقدار این دو Q_{ϕ_1} و Q_{ϕ_2} در تابع بلمن استفاده می‌شود. نحوه محاسبه هدف بر اساس دو تابع Q اشاره شده در رابطه (۴-۲۴) آورده شده است.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_i, \text{targ}}(s', a'(s')) \quad (۴-۲۴)$$

سپس، در هر دو تابع Q_{ϕ_1} و Q_{ϕ_2} یادگیری انجام می‌شود.

$$L(\phi_1, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left(Q_{\phi_1}(s, a) - y(r, s', d) \right)^2 \quad (۴-۲۵)$$

$$L(\phi_2, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left(Q_{\phi_2}(s, a) - y(r, s', d) \right)^2 \quad (۴-۲۶)$$

- به‌روزرسانی‌های تاخیری سیاست^{۴۴}: الگوریتم TD3 سیاست را با تاخیر بیشتری نسبت به تابع Q به‌روزرسانی می‌کند. در مرجع [۵۹] توصیه شده است که برای هر دو به‌روزرسانی تابع Q ، یک به‌روزرسانی سیاست انجام شود.

^{۴۱}Twin Delayed Deep Deterministic Policy Gradient (TD3)

^{۴۲}Clipped Double-Q Learning

^{۴۳}twin

^{۴۴}Delayed Policy Updates

این دو ترفند منجر به بهبود قابل توجه عملکرد TD3 نسبت به DDPG پایه می‌شوند. در نهایت سیاست با پیشنهاد کردن Q_{ϕ_1} آموخته می‌شود:

$$\max_{\theta} E_{s \sim \mathcal{D}} [Q_{\phi_1}(s, \mu_{\theta}(s))] \quad (27-4)$$

۱-۳-۴ اکتشاف و بهره‌برداری در TD3

الگوریتم TD3 یک سیاست قطعی را به صورت غیرسیاست محور آموزش می‌دهد. از آنجایی که سیاست قطعی است، در ابتدا عامل تنوع کافی از اعمال را برای یافتن روش‌های مفید امتحان نمی‌کند. برای بهبود اکتشاف سیاست‌های TD3، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. در این پژوهش، نویز گاوسی با میانگین صفر بدون هم‌بندی زمانی اعمال شده است. شدت نویز جهت بهره‌برداری بهتر در طول زمان کاهش می‌یابد.

۲-۳-۴ شبه‌کد TD3

در این بخش الگوریتم TD3 پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

الگوریتم ۲ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

ورودی: پارامترهای اولیه سیاست (θ) ، پارامترهای تابع Q (ϕ_1, ϕ_2) ، بافر بازی خالی (\mathcal{D})

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\phi_{\text{targ},2} \leftarrow \phi_2, \phi_{\text{targ},1} \leftarrow \phi_1, \theta_{\text{targ}} \leftarrow \theta$

۲: تا وقتی همگرایی رخ دهد:

۳: وضعیت (s) را مشاهده کرده و عمل $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به طوری که $\epsilon \sim \mathcal{N}$ است.

۴: عمل a را در محیط اجرا کنید.

۵: وضعیت بعدی s' ، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا خیر.

۶: اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

۷: اگر زمان به روزرسانی فرا رسیده است:

۸: به ازای j در هر تعداد به روزرسانی:

۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$ ، از \mathcal{D} نمونه‌گیری شود.

۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

۱۱: تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر به روزرسانی کنید:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \quad \text{for } i = 1, 2$$

۱۲: اگر باقیمانده j بر تاخیر سیاست برابر ۰ باشد :

۱۳: سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر به روزرسانی کنید:

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

۱۴: شبکه‌های هدف را با استفاده از معادلات زیر به روزرسانی کنید:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta$$

۴-۴ عامل عملگر نقاد نرم

عملگر نقاد نرم^{۴۵} الگوریتمی است که یک سیاست تصادفی را به صورت غیرسیاست محور بهینه می‌کند و پلی بین بهینه‌سازی سیاست تصادفی و رویکردهای غیرسیاست محور مانند DDPG ایجاد می‌کند. این الگوریتم جانشین مستقیم TD3 نیست (زیرا تقریباً همزمان منتشر شده است)؛ اما، ترفند یادگیری دوگانه محدود شده را در خود جای داده است و به دلیل سیاست تصادفی SAC، از روشی به نام صاف کردن سیاست هدف^{۴۶} استفاده شده است. یکی از ویژگی‌های اصلی SAC، تنظیم آنتروپی است. آنتروپی معیاری از تصادفی بودن انتخاب عمل در سیاست است. آموزش سیاست در جهت تعادل بهینه بین آنتروپی و بیشینه‌سازی بازده مورد انتظار است. این شرایط ارتباط نزدیکی با تعادل اکتشاف-بهره‌برداری دارد. افزایش آنتروپی منجر به اکتشاف بیشتر می‌شود که می‌تواند یادگیری را در مراحل بعدی تسریع کند. همچنین، می‌تواند از همگرایی زودهنگام سیاست به یک بهینه محلی بد جلوگیری کند. برای توضیح SAC، ابتدا باید به بررسی یادگیری تقویتی تنظیم شده با آنتروپی^{۴۷} پرداخته شود. در RL تنظیم شده با آنتروپی، روابط تابع ارزش کمی متفاوت است.

۱-۴-۴ یادگیری تقویتی تنظیم شده با آنتروپی

آنتروپی معیاری برای سنجش میزان عدم قطعیت یا تصادفی بودن یک متغیر تصادفی یا توزیع احتمال آن است. به عبارت دقیق‌تر، آنتروپی برای یک توزیع احتمال، میانگین اطلاعات حاصل از نمونه‌برداری از آن توزیع را اندازه‌گیری می‌کند. در زمینه یادگیری تقویتی، تنظیم با آنتروپی تکنیکی است که با افزودن یک ترم متناسب با آنتروپی سیاست به تابع هدف، عامل را تشویق به اکتشاف بیشتر و اتخاذ سیاست‌های تصادفی‌تر می‌کند. این امر می‌تواند به بهبود پایداری فرآیند یادگیری و جلوگیری از همگرایی زودهنگام به بهینه‌های محلی کمک کند. فرض کنید X یک متغیر تصادفی پیوسته با تابع چگالی احتمال $p(x)$ باشد. آنتروپی $H(X)$ این متغیر تصادفی به صورت امید ریاضی لگاریتم منفی چگالی احتمال آن تعریف می‌شود:

$$H(X) = \mathbb{E}_{x \sim p} [-\log p(x)] \quad (۲۸-۴)$$

۲-۴-۴ سیاست در SAC

در یادگیری تقویتی تنظیم شده با آنتروپی، عامل در هر مرحله زمانی متناسب با آنتروپی سیاست در آن مرحله زمانی پاداش دریافت می‌کند. بر اساس توضیحات اشاره شده روابط یادگیری تقویتی به صورت زیر می‌شود.

^{۴۵}Soft Actor Critic (SAC)

^{۴۶}Target Policy Smoothing

^{۴۷}Entropy-Regularized Reinforcement Learning

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi} \sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)) \right) \quad (29-4)$$

که در آن $(\alpha > 0)$ ضریب مبادله^{۴۸} است.

۳-۴-۴ تابع ارزش در SAC

اکنون می‌توان تابع ارزش کمی متفاوت را بر اساس این مفهوم تعریف کرد. V^π به گونه‌ای تغییر می‌کند که پاداش‌های آنتروپی را از هر مرحله زمانی شامل می‌شود.

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)) \right) \middle| s_0 = s \right] \quad (30-4)$$

۴-۴-۴ تابع Q در SAC

تابع Q^π به گونه‌ای تغییر می‌کند که پاداش‌های آنتروپی را از هر مرحله زمانی به جز مرحله اول شامل می‌شود.

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot|s_t)) \middle| s_0 = s, a_0 = a \right] \quad (31-4)$$

با این تعاریف رابطه V^π و Q^π به صورت زیر است.

$$V^\pi(s) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)] + \alpha H(\pi(\cdot|s)) \quad (32-4)$$

۵-۴-۴ معادله بلمن در SAC

معادله بلمن در حالت تنظیم‌شده با آنتروپی به صورت زیر ارائه می‌شود.

$$Q^\pi(s, a) = \mathbb{E}_{\substack{s' \sim P \\ a' \sim \pi}} [R(s, a, s') + \gamma (Q^\pi(s', a') + \alpha H(\pi(\cdot|s')))] \quad (33-4)$$

$$= \mathbb{E}_{s' \sim P} [R(s, a, s') + \gamma V^\pi(s')] \quad (34-4)$$

⁴⁸Trade-Off

۶-۴-۴ یادگیری Q

با در نظر گرفتن موارد اشاره شده، یادگیری Q در SAC با کمینه کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۳۵-۴) با استفاده از کاهش گرادیان انجام می شود.

$$L(\phi_i, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left[\left(Q_{\phi_i}(s, a) - y(r, s', d) \right)^2 \right] \quad (۳۵-۴)$$

در معادله (۳۵-۴) تابع هدف برای روش یادگیری تقویتی SAC به صورت زیر تعریف می شود.

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s', \tilde{a}') - \alpha \log \pi_{\theta}(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_{\theta}(\cdot|s') \quad (۳۶-۴)$$

نماد عمل بعدی را به جای a' به \tilde{a}' تغییر داده شده تا مشخص شود که عمل های بعدی باید از آخرین سیاست نمونه برداری شوند در حالی که r و s باید از بافر تکرار بازی آمده باشند.

۷-۴-۴ سیاست در SAC

سیاست باید در هر وضعیت برای به حداکثر رساندن بازگشت مورد انتظار آینده به همراه آنتروپی مورد انتظار آینده عمل کند. یعنی باید $V^{\pi}(s)$ را به حداکثر برساند، بسط تابع ارزش در ادامه آمده است.

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi} [Q^{\pi}(s, a)] + \alpha H(\pi(\cdot|s)) \quad (۳۷-۴)$$

$$= \mathbb{E}_{a \sim \pi} [Q^{\pi}(s, a) - \alpha \log \pi(a|s)] \quad (۳۸-۴)$$

در بهینه سازی سیاست از ترفند پارامترسازی مجدد^{۴۹} استفاده می شود، که در آن نمونه ای از $\pi_{\theta}(\cdot|s)$ با محاسبه یک تابع قطعی از وضعیت، پارامترهای سیاست و نویز مستقل استخراج می شود. در این پژوهش مانند نویسندگان مقاله SAC [۶۱]، از یک سیاست گاوسی فشرده^{۵۰} استفاده شده است. بر اساس این روش نمونه ها مطابق با رابطه زیر بدست می آیند:

$$\tilde{a}_{\theta}(s, \xi) = \tanh(\mu_{\theta}(s) + \sigma_{\theta}(s) \odot \xi), \quad \xi \sim \mathcal{N} \quad (۳۹-۴)$$

در رابطه بالا \odot نماد ضرب داخلی است. تابع \tanh در سیاست SAC تضمین می کند که اعمال در یک محدوده متناهی محدود شوند. این مورد در سیاست های VPG، TRPO و PPO وجود ندارد. همچنین اعمال این تابع توزیع را از حالت گاوسی تغییر می دهد.

⁴⁹Reparameterization

⁵⁰Squashed Gaussian Policy

در الگوریتم SAC با استفاده از ترفند پارامتری‌سازی مجدد، عمل‌ها از یک توزیع نرمال به‌وسیله نویز تصادفی تولید شده و به این ترتیب امکان محاسبه مشتق‌ها به‌طور مستقیم از طریق تابع توزیع فراهم می‌شود، که باعث ثبات و کارایی بیشتر در آموزش می‌شود. اما در حالت بدون پارامتری‌سازی مجدد، عمل‌ها مستقیماً از توزیع سیاست نمونه‌برداری می‌شوند و محاسبه گرادیان نیازمند استفاده از ترفند نسبت احتمال^{۵۱} است که معمولاً باعث افزایش واریانس و ناپایداری در آموزش می‌شود.

$$\mathbb{E}_{a \sim \pi_\theta} [Q^{\pi_\theta}(s, a) - \alpha \log \pi_\theta(a|s)] = \mathbb{E}_{\xi \sim \mathcal{N}} [Q^{\pi_\theta}(s, \tilde{a}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s, \xi)|s)] \quad (۴۰-۴)$$

برای به‌دست آوردن تابع هزینه سیاست، گام نهایی این است که باید Q^{π_θ} را با یکی از تخمین‌زننده‌های تابع خود جایگزین کنیم. برخلاف TD3 که از Q_{ϕ_1} (فقط اولین تخمین‌زننده Q) استفاده می‌کند، SAC از $\min_{j=1,2} Q_{\phi_j}$ (کمینه‌ی دو تخمین‌زننده Q) استفاده می‌کند. بنابراین، سیاست طبق رابطه زیر بهینه می‌شود:

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}, \xi \sim \mathcal{N}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s, \xi)|s) \right] \quad (۴۱-۴)$$

که تقریباً مشابه بهینه‌سازی سیاست در DDPG و TD3 است، به جز ترفند min-double-Q، تصادفی‌بودن و عبارت آنتروپی.

۸-۴-۴ اکتشاف و بهره‌برداری در SAC

الگوریتم SAC یک سیاست تصادفی با تنظیم‌سازی آنتروپی آموزش می‌دهد و به صورت سیاست محور به اکتشاف می‌پردازد. ضریب تنظیم آنتروپی α به طور صریح تعادل بین اکتشاف و بهره‌برداری را کنترل می‌کند، به‌طوری‌که مقادیر بالاتر α به اکتشاف بیشتر و مقادیر پایین‌تر α به بهره‌برداری بیشتر منجر می‌شود. مقدار بهینه α (که به یادگیری پایدارتر و پاداش بالاتر منجر می‌شود) ممکن است در محیط‌های مختلف متفاوت باشد و نیاز به تنظیم دقیق داشته باشد. در زمان آزمایش، برای ارزیابی میزان بهره‌برداری سیاست از آنچه یاد گرفته است، تصادفی بودن را حذف کرده و از عمل میانگین به جای نمونه‌برداری از توزیع استفاده می‌کنیم. این روش معمولاً عملکرد را نسبت به سیاست تصادفی بهبود می‌بخشد.

⁵¹Likelihood Ratio Trick

۹-۴-۴ شبکه‌کد SAC

در این بخش الگوریتم SAC پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۳ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

الگوریتم ۳ عامل عملگرد نقاد نرم

ورودی: پارامترهای اولیه سیاست (θ) ، پارامترهای تابع Q (ϕ_1, ϕ_2) ، بافر بازی خالی (\mathcal{D})

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\theta_{\text{targ}} \leftarrow \theta$ ، $\phi_{\text{targ},1} \leftarrow \phi_1$ ، $\phi_{\text{targ},2} \leftarrow \phi_2$

۲: تا وقتی همگرایی رخ دهد:

۳: وضعیت (s) را مشاهده کرده و عمل $a \sim \pi_{\theta}(\cdot|s)$ را انتخاب کنید.

۴: عمل a را در محیط اجرا کنید.

۵: وضعیت بعدی s' ، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا

خیر.

۶: اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

۷: اگر زمان به‌روزرسانی فرا رسیده است:

۸: به ازای j در هر تعداد به‌روزرسانی:

۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$ ، از \mathcal{D}

نمونه‌گیری شود.

۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_{\theta}(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_{\theta}(\cdot|s')$$

۱۱: تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \quad \text{for } i = 1, 2$$

۱۲: سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_{\theta}(s)) - \alpha \log \pi_{\theta}(\tilde{a}_{\theta}(s)|s) \right)$$

۱۳: شبکه‌های هدف را با استفاده از معادلات زیر به‌روزرسانی کنید:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

۵-۴ عامل بهینه‌سازی سیاست مجاور

الگوریتم بهینه‌سازی سیاست مجاور^{۵۲} یک الگوریتم بهینه‌سازی سیاست مبتنی بر گرادینان است که برای حل مسائل کنترل مسئله‌های یادگیری تقویتی استفاده می‌شود. این الگوریتم از الگوریتم TRPO^{۵۳} الهام گرفته شده است و با اعمال تغییراتی بر روی آن، سرعت و کارایی آن را افزایش داده است. در این بخش به بررسی این الگوریتم و نحوه عملکرد آن می‌پردازیم. الگوریتم PPO همانند سایر الگوریتم‌های یادگیری تقویتی، به دنبال یافتن بهترین گام ممکن برای بهبود عملکرد سیاست با استفاده از داده‌های موجود است. این الگوریتم تلاش می‌کند تا از گام‌های بزرگ که می‌توانند منجر به افت ناگهانی عملکرد شوند، اجتناب کند. برخلاف روش‌های پیچیده‌تر مرتبه دوم مانند TRPO، PPO از مجموعه‌ای از روش‌های مرتبه اول ساده‌تر برای حفظ نزدیکی سیاست‌های جدید به سیاست‌های قبلی استفاده می‌کند. این سادگی در پیاده‌سازی، PPO را به روشی کارآمدتر تبدیل می‌کند، در حالی که از نظر تجربی نشان داده شده است که عملکردی حداقل به اندازه TRPO دارد. از جمله ویژگی‌های مهم این الگوریتم می‌توان به سیاست محور بودن آن اشاره کرد. این الگوریتم برای عامل‌های یادگیری تقویتی که سیاست‌های پیوسته و گسسته دارند، مناسب است.

الگوریتم PPO دارای دو گونه اصلی PPO-Clip و PPO-Penalty است. در ادامه به بررسی هر یک از این دو گونه پرداخته شده است.

- **روش PPO-Penalty:** روش PPO-Penalty به دنبال حل تقریبی و به‌روزرسانی با محدودیت واگرایی کولباک-لیبلر^{۵۴} است، مشابه روشی که در الگوریتم TRPO استفاده شده است. با این حال، به جای اعمال یک محدودیت سخت^{۵۵}، PPO-Penalty واگرایی KL را در تابع هدف جریمه می‌کند. این جریمه به طور خودکار در طول آموزش تنظیم می‌شود تا از افت ناگهانی عملکرد جلوگیری کند.

- **روش PPO-Clip:** در این روش، هیچ عبارت واگرایی KL در تابع هدف وجود ندارد و هیچ محدودیتی اعمال نمی‌شود. در عوض، PPO-Clip از یک عملیات بریدن^{۵۶} خاص در تابع هدف استفاده می‌کند تا انگیزه سیاست جدید برای دور شدن از سیاست قبلی را از بین ببرد.

در این پژوهش از روش PPO-Clip برای آموزش عامل‌های یادگیری تقویتی استفاده شده است.

⁵²Proximal Policy Optimization (PPO)

⁵³Trust Region Policy Optimization

⁵⁴Kullback-Leibler (KL) Divergence

⁵⁵Hard Constraint

⁵⁶Clipping

۴-۵-۱ سیاست در الگوریتم PPO

تابع سیاست در الگوریتم PPO به صورت یک شبکه عصبی پیاده‌سازی شده‌است. این شبکه عصبی ورودی‌های محیط را دریافت کرده و اقدامی را که باید عامل انجام دهد را تولید می‌کند. این شبکه عصبی می‌تواند شامل چندین لایه پنهان با توابع فعال‌سازی مختلف باشد. در این پژوهش از یک شبکه عصبی با سه لایه پنهان و تابع فعال‌سازی ReLu استفاده شده‌است. تابع سیاست در الگوریتم PPO به صورت زیر به‌روزرسانی می‌شود:

$$\theta_{k+1} = \arg \max_{\theta} E_{s,a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \quad (۴۲-۴)$$

در این پژوهش برای به حداکثر رساندن تابع هدف، چندین گام بهینه‌سازی گرادیان کاهشی تصادفی^{۵۷} اجرا شده‌است. در معادله بالا L به‌صورت زیر تعریف شده‌است:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \quad (۴۳-۴)$$

که در آن ϵ یک ابرپارامتر است که مقدار آن معمولاً کوچک است. این ابرپارامتر مشخص می‌کند که چقدر اندازه گام بهینه‌سازی باید محدود شود. در این پژوهش مقدار $\epsilon = 0.2$ انتخاب شده‌است. جهت سادگی در پیاده‌سازی معادله (۴۳-۴) به معادله تغییر داده شده‌است.

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a)) \right) \quad (۴۴-۴)$$

که تابع g به صورت زیر تعریف شده‌است.

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0 \end{cases} \quad (۴۵-۴)$$

در حالی که این نوع محدود کردن (PPO-Clip) تا حد زیادی به اطمینان از به‌روزرسانی‌های معقول سیاست کمک می‌کند، همچنان ممکن است سیاستی به‌دست آید که بیش از حد از سیاست قدیمی دور باشد. برای جلوگیری از این امر، پیاده‌سازی‌های مختلف PPO از مجموعه‌ای از ترفندها استفاده می‌کنند. در پیاده‌سازی این پژوهش، از روشی ساده به نام توقف زودهنگام^{۵۸} استفاده شده‌است. اگر میانگین واگرایی کولباک-لیبلر (KL) خط‌مشی جدید از خط‌مشی قدیمی از یک آستانه فراتر رود، گام‌های گرادیان (بهینه‌سازی) را متوقف می‌شوند.

⁵⁷Stochastic Gradient Descent (SGD)

⁵⁸Early Stopping

۲-۵-۴ اکتشاف و بهره‌برداری در PPO

الگوریتم PPO از یک سیاست تصادفی به صورت سیاست‌محور برای آموزش استفاده می‌کند. این به این معنی است که اکتشاف محیط با نمونه‌گیری عمل‌ها بر اساس آخرین نسخه از این سیاست تصادفی انجام می‌شود. میزان تصادفی بودن انتخاب عمل به شرایط اولیه و فرآیند آموزش بستگی دارد.

در طول آموزش، سیاست به طور کلی به تدریج کمتر تصادفی می‌شود، زیرا قانون به‌روزرسانی آن را تشویق می‌کند تا از پاداش‌هایی که قبلاً پیدا کرده است، بهره‌برداری کند. البته این موضوع می‌تواند منجر به رسیدن سیاست به بهینه‌های محلی^{۵۹} شود.

۳-۵-۴ شبه‌کد PPO

در این بخش الگوریتم PPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم^۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

⁵⁹Local Optima

الگوریتم ۴ بهینه‌سازی سیاست مجاور (PPO-Clip)

ورودی: پارامترهای اولیه سیاست (θ_0) ، پارامترهای تابع ارزش (ϕ_0)

۱: به ازای $k = 0, 1, 2, \dots$:

۲: مجموعه‌ای از مسیرها به نام $\mathcal{D}_k = \{\tau_i\}$ با اجرای سیاست $\pi_k = \pi(\theta_k)$ در محیط جمع‌آوری شود.

۳: پاداش‌های باقی‌مانده (\hat{R}_t) محاسبه شود.

۴: برآوردهای مزیت را محاسبه کنید، \hat{A}_t (با استفاده از هر روش تخمین مزیت) بر اساس تابع ارزش فعلی V_{ϕ_k} .

۵: سیاست را با به حداکثر رساندن تابع هدف PPO-Clip به‌روزرسانی کنید:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

معمولاً از طریق گرادیان افزایشی تصادفی Adam.

۶: برازش تابع ارزش با رگرسیون بر روی میانگین مربعات خطا:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

معمولاً از طریق برخی از الگوریتم‌های کاهشی گرادیان.

فصل ۵

شبیه‌سازی عامل در محیط سه جسمی

در این فصل، فرآیند شبیه‌سازی عامل هوشمند کنترل‌کننده فضاپیما در محیط دینامیکی سه جسمی بررسی شده است. در بخش ۱-۵ به طراحی و در بخش ۲-۵ به شبیه‌سازی عامل هدایت‌کننده مبتنی بر یادگیری تقویتی است پرداخته شده است. این عامل طراحی و شبیه‌سازی شده باید توانایی این را داشته باشد که فضاپیما را به‌طور مؤثر به سمت اهداف تعیین‌شده هدایت کند، در حالی که محدودیت‌هایی نظیر مصرف سوخت و وجود اغتشاش دارد.

۱-۵ طراحی عامل

در این زیربخش، معماری عامل هوشمند کنترل‌کننده فضاپیما در محیط سه جسمی شرح داده شده است. این معماری شامل تعریف فضای حالت، عمل و تابع پاداش است.

۱-۱-۵ فضای حالت

فضای حالت^۱ در این پژوهش به‌گونه‌ای طراحی شده است که وضعیت دینامیکی فضاپیما را نسبت به یک مسیر و سرعت مرجع مشخص می‌کند. این فضا شامل اختلاف‌های موقعیت و سرعت از مسیر و سرعت مرجع است و به‌صورت زیر تعریف می‌شود:

$$S = \{\delta x, \delta y, \delta \dot{x}, \delta \dot{y}\}$$

که در آن:

¹State Space

- $\delta x, \delta y$: اختلاف موقعیت فضایی نسبت به مسیر مرجع در محورهای x, y .
- $\delta \dot{x}, \delta \dot{y}$: اختلاف سرعت فضایی نسبت به سرعت مرجع در محورهای x, y .

هر یک از این متغیرها به طور مستقل وضعیت فضایی را در یک جهت خاص توصیف می‌کنند و امکان تحلیل دقیق انحرافات را فراهم می‌سازند. استفاده از اختلاف‌های موقعیت و سرعت به جای مقادیر مطلق، به دلایل زیر انجام شده است:

- **تمرکز بر انحرافات:** هدف اصلی سیستم کنترلی، کاهش انحرافات از مسیر و سرعت مطلوب است. با استفاده از اختلاف‌ها، کنترلر می‌تواند به طور مستقیم بر این انحرافات اثر بگذارد و نیازی به محاسبه مقادیر مطلق موقعیت و سرعت ندارد.
- **سازگاری با یادگیری تقویتی:** در الگوریتم‌های یادگیری تقویتی، فضاهای حالت مبتنی بر اختلاف معمولاً دامنه محدودتری دارند که فرآیند یادگیری را سریع‌تر و پایدارتر می‌کند.

۵-۱-۲ فضای عمل

فضای عمل^۲ فضایی با پیشران کم مجموعه‌ای از عمل‌های پیوسته است که فضایی می‌تواند در محیط شبیه‌سازی انجام دهد. این فضا به گونه‌ای طراحی شده که امکان اعمال نیرو در جهت‌های مشخص و با مقادیر متناسب با توان واقعی فضاییها فراهم شود. به طور خاص، فضای اقدام شامل موارد زیر است:

- **نیروی اعمال شده در جهت x :** این متغیر پیوسته، مقدار نیرویی را که در جهت محور x به فضایی وارد می‌شود، تعیین می‌کند. دامنه این نیرو بر اساس توان پیشران‌های موجود در فضاییها و واقعی انتخاب شده است. به عبارت دیگر، اگر حداکثر نیروی قابل اعمال در جهت x برابر با $f_{x,\max}$ باشد، این متغیر می‌تواند مقادیری در بازه $[-f_{x,\max}, f_{x,\max}]$ داشته باشد.

- **نیروی اعمال شده در جهت y :** این متغیر پیوسته، مقدار نیرویی را که در جهت محور y به فضایی وارد می‌شود، مشخص می‌کند. مشابه جهت x ، دامنه این نیرو نیز بر اساس توان پیشران‌های موجود تعیین شده و می‌تواند در بازه $[-f_{y,\max}, f_{y,\max}]$ قرار گیرد.

انتخاب این نیروها بر اساس ویژگی‌های واقعی فضاییها، به‌ویژه توان و محدودیت‌های پیشران‌های آنها، صورت گرفته است. این امر اطمینان می‌دهد که شبیه‌سازی تا حد ممکن به شرایط واقعی نزدیک باشد و نتایج

²Action Space

به دست آمده قابلیت تعمیم به کاربردهای عملی را داشته باشند. همچنین، تعریف فضای اقدام به صورت پیوسته، امکان کنترل دقیق و انعطاف پذیر بر حرکت فضاپیما را فراهم می کند، که برای دستیابی به اهداف کنترلی در محیط های دینامیکی پیچیده ضروری است. به طور خلاصه، فضای اقدام به صورت زیر تعریف می شود:

$$a = \{f_x, f_y \mid f_x \in [-f_{x,\max}, f_{x,\max}], f_y \in [-f_{y,\max}, f_{y,\max}]\}$$

۳-۱-۵ تابع پاداش

تابع پاداش^۳ به منظور هدایت رفتار عامل طراحی شده و شامل دو مؤلفه اصلی است:

- پاداش برای دستیابی به هدف: تشویق عامل برای نزدیک شدن به مدار هدف.
- جریمه برای مصرف سوخت: تنبیه برای استفاده بیش از حد از پیشران.
- جریمه برای انحراف از مسیر مرجع: تنبیه برای خروج از مسیر مرجع.

تابع پاداش به صورت زیر تعریف می شود:

$$r(s, a) = r_{\text{target}}(s) + r_{\text{thrust}}(a) + r_{\text{divergence}}(s)$$

که در آن مؤلفه های تابع پاداش به صورت زیر تعریف شده اند:

$$r_{\text{target}}(s) = -k_1 \cdot d(s, s_{\text{target}}) \quad (۱-۵)$$

$$r_{\text{thrust}}(a) = -k_2 \cdot |0a|0 \quad (۲-۵)$$

$$r_{\text{divergence}}(s) = \begin{cases} -k_3 & \text{if } d(s, s_{\text{reference}}) > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (۳-۵)$$

تابع $d(s, s')$ فاصله بین دو وضعیت s و s' را نشان می دهد که معمولاً به صورت فاصله اقلیدسی محاسبه می شود. ضرایب k_1, k_2, k_3 از طریق آزمایش و خطا تنظیم شده اند تا تعادل مناسبی بین دستیابی به هدف، بهینه سازی مصرف سوخت، و حفظ مسیر مرجع برقرار شود. علاوه بر این، این ضرایب تأثیر مستقیمی بر پایداری و فرآیند یادگیری عامل دارند. به عنوان مثال، انتخاب مقادیر بیش از حد بزرگ برای k_1 ممکن است باعث شود عامل به سرعت به سمت هدف حرکت کند اما پایداری مسیر را از دست بدهد، در حالی که مقادیر بزرگ k_3 می تواند عامل را بیش از حد محافظه کار کرده و فرآیند یادگیری را کند نماید. تنظیم دقیق این ضرایب، نه تنها عملکرد عامل را بهینه می کند، بلکه پایداری عددی و سرعت همگرایی الگوریتم یادگیری تقویتی را نیز تضمین می نماید.

^۳Reward Function

۲-۵ شبیه‌سازی عامل

در این زیربخش، فرآیند شبیه‌سازی و آموزش عامل با استفاده از الگوریتم‌های یادگیری تقویتی پیشرفته شرح داده می‌شود. الگوریتم‌های مورد استفاده، مراحل آموزش، و نتایج حاصل از شبیه‌سازی ارائه می‌گردند.

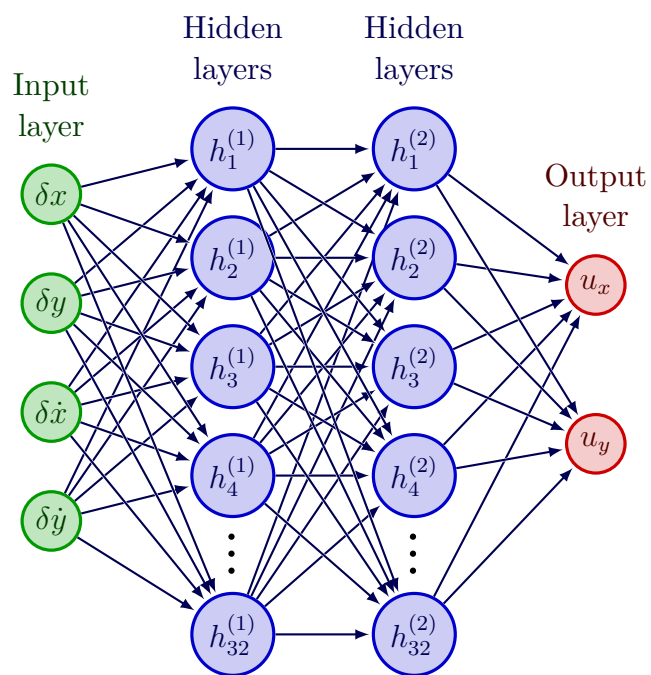
۱-۲-۵ پارامترهای یادگیری الگوریتم‌های مورد استفاده

برای آموزش عامل، الگوریتم‌های زیر به‌کار گرفته شده‌اند:

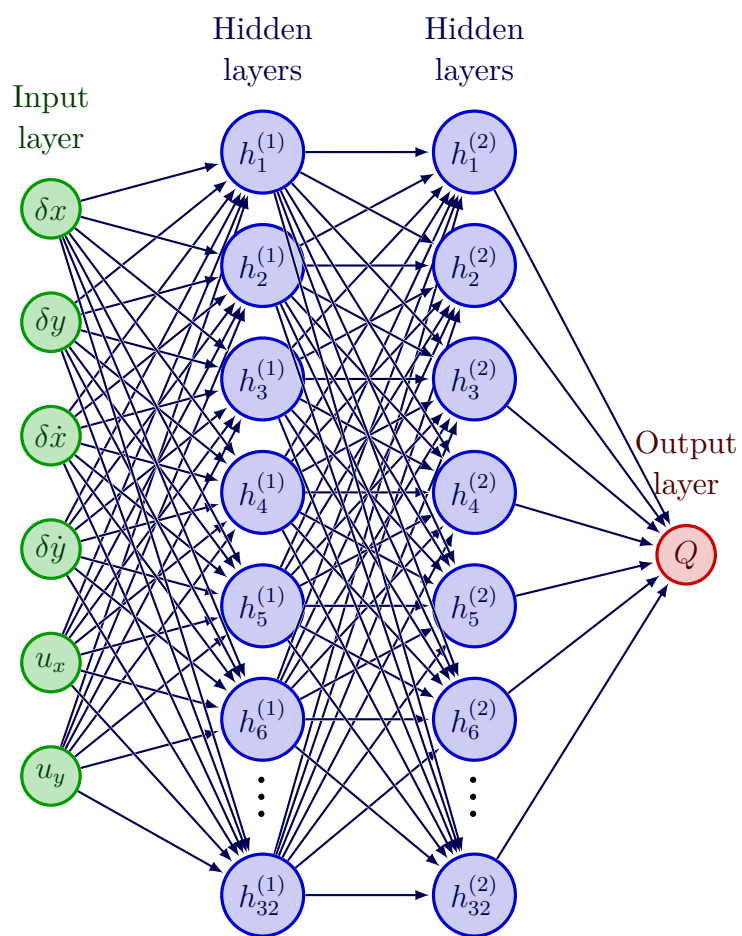
جدول ۱-۵: ویژگی‌های الگوریتم‌های مورد استفاده در شبیه‌سازی

تعداد پارامترها	شبکه Critic		شبکه Actor		الگوریتم
	نودها	لایه‌ها	نودها	لایه‌ها	
150×10^3	$(2^8, 2^5)$	3	$(2^8, 2^7, 2^6)$	3	DDPG
50×10^3	$(2^7, 2^6)$	2	$(2^7, 2^6)$	2	PPO
160×10^3	$(2^8, 2^7, 2^6)$	3	$(2^8, 2^7, 2^6)$	3	SAC
200×10^3	$(2^8, 2^7, 2^7, 2^6)$	4	$(2^8, 2^7, 2^6)$	3	TD3

این الگوریتم‌ها به دلیل توانایی در مدیریت فضاها پیوسته و عملکرد مؤثر در محیط‌های پیچیده انتخاب شده‌اند. در شکل‌های ۱-۵ و ۲-۵ ساختار شبیه‌سازی شده شبکه عصبی عامل و نقاد آورده شده است.



شکل ۵-۱: ساختار شبکه عصبی عامل



شکل ۵-۲: ساختار شبکه عصبی نقاد

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزن تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخ یادگیری سیاست	10^{-3}
نرخ یادگیری Q	10^{-3}	اندازه‌ی دسته	1024
گام شروع استفاده از سیاست	5 000	گام شروع به‌روزرسانی	1 000
فاصله‌ی به‌روزرسانی	2 000	نویز عمل	0.1
حداکثر طول رخداد	6 000	دستگاه	Cuda
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

جدول ۵-۲: جدول پارامترها و مقادیر پیش‌فرض الگوریتم DDPG [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزن تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخ یادگیری سیاست	10^{-3}
نرخ یادگیری Q	10^{-3}	اندازه‌ی دسته	1024
گام شروع استفاده از سیاست	5 000	گام شروع به‌روزرسانی	1 000
فاصله‌ی به‌روزرسانی	2 000	نویز عمل	0.1
نویز هدف	0.2	برش نویز	0.5
تأخیر در به‌روزرسانی سیاست	2	حداکثر طول رخداد	30 000
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

جدول ۵-۳: جدول پارامترها و مقادیر پیش‌فرض الگوریتم TD3 [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزن تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخ یادگیری	10^{-3}
نرخ دمای آلفا	0.2	اندازه‌ی دسته	1024
گام شروع استفاده از سیاست	5 000	گام شروع به‌روزرسانی	1 000
تعداد به‌روزرسانی در هر مرحله	10	فاصله‌ی به‌روزرسانی	2 000
تعداد اپیزودهای آزمون	10	حداکثر طول رخداد	30 000
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

جدول ۴-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم SAC [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
ضریب تنزیل (γ)	0.99	ضریب برش ratio clip	0.2
نرخ یادگیری سیاست	3×10^{-4}	نرخ یادگیری تابع ارزش	10^{-3}
تعداد تکرار آموزش سیاست	80	تعداد تکرار آموزش ارزش	80
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

جدول ۵-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم PPO [۱]

۲-۲-۵ فرآیند آموزش

آموزش عامل به‌صورت کلی در چند مرحله انجام شده است. ابتدا، کاوش اولیه در محیط با استفاده از یک سیاست تصادفی صورت گرفته و تجربه‌های اولیه جمع‌آوری شده‌اند. سپس، شبکه‌های عصبی الگوریتم‌ها با بهره‌گیری از این تجربه‌ها به‌روزرسانی شده‌اند. در نهایت، پارامترهای کلیدی مانند نرخ یادگیری و اندازه بافر تجربه تنظیم شده‌اند تا پایداری فرآیند تضمین شود.

برای پیاده‌سازی این فرآیند، از چارچوب PyTorch استفاده شده است. همچنین، به‌منظور جلوگیری از بیش‌برازش، تکنیک Noise Exploration به‌کار گرفته شده است. آموزش تا زمانی ادامه یافته که موفقیت عامل در بیش از ۹۰ درصد موارد به‌دست آمده باشد. در این راستا، برای بهینه‌سازی پارامترهای شبکه‌های

عصبی، از روش Backpropagation استفاده شده است. این روش بر اساس گرادیان تابع خطا نسبت به پارامترها عمل می‌کند که به صورت زیر بیان می‌شود:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w} \quad (۴-۵)$$

که در آن L تابع خطا، w وزن‌های شبکه، و y خروجی شبکه عصبی است. به روزرسانی وزن‌ها با استفاده از روش گرادیان نزولی انجام شده است:

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w} \quad (۵-۵)$$

که η نرخ یادگیری است و به عنوان یک پارامتر کلیدی تنظیم شده است.

فصل ۶

یادگیری تقویتی چندعاملی

کاربردهای پیچیده در یادگیری تقویتی نیازمند اضافه کردن چندین عامل^۱ برای انجام همزمان وظایف مختلف هستند. با این حال، افزایش تعداد عامل‌ها چالش‌هایی در مدیریت تعاملات میان آن‌ها به همراه دارد. در این فصل، بر اساس مسئله بهینه‌سازی برای هر عامل، مفهوم تعادل^۲ معرفی شده تا رفتارهای توزیعی چندعاملی را تنظیم کند. رابطه رقابت میان عامل‌ها در سناریوهای مختلف تحلیل شده و آن‌ها با الگوریتم‌های معمول یادگیری تقویتی چندعاملی ترکیب شده‌اند. بر اساس انواع تعاملات، یک چارچوب نظریه بازی برای مدل‌سازی عمومی در سناریوهای چندعاملی استفاده شده است. با تحلیل بهینه‌سازی و وضعیت تعادل برای هر بخش از چارچوب، سیاست بهینه یادگیری تقویتی چندعاملی برای هر عامل بررسی شده است.

۶-۱ تعاریف و مفاهیم اساسی

یادگیری تقویتی چندعاملی^۳ به بررسی چگونگی یادگیری و تصمیم‌گیری چندین عامل مستقل در یک محیط مشترک پرداخته می‌شود. برای تحلیل دقیق و درک بهتر این حوزه، اجزای اصلی آن شامل عامل، سیاست و مطلوبیت^۴ در نظر گرفته می‌شوند که در ادامه به صورت مختصر و منسجم تشریح می‌گردند.

- عامل: یک موجودیت مستقل به عنوان عامل تعریف می‌شود که به صورت خودمختار با محیط تعامل کرده و بر اساس مشاهدات رفتار سایر عامل‌ها، سیاست‌هایش انتخاب می‌گردند تا سود حداکثر یا ضرر حداقل حاصل شود. در سناریوهای مورد بررسی، چندین عامل به صورت مستقل عمل می‌کنند؛ اما اگر

¹Multi-Agent

²Equilibrium

³Multi-Agent Reinforcement Learning (MARL)

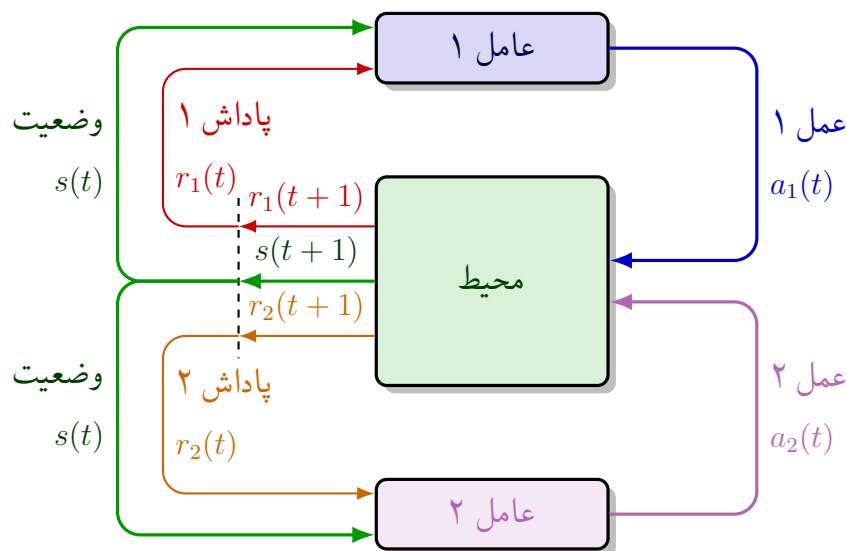
⁴Utility

تعداد عامل‌ها به یک کاهش یابد، MARL به یادگیری تقویتی معمولی تبدیل می‌شود.

- سیاست: برای هر عامل در MARL، سیاستی خاص در نظر گرفته می‌شود که به عنوان روشی برای انتخاب اقدامات بر اساس وضعیت محیط و رفتار سایر عامل‌ها تعریف می‌گردد. این سیاست‌ها با هدف به حداکثر رساندن سود و به حداقل رساندن هزینه طراحی شده و تحت تأثیر محیط و سیاست‌های دیگر عامل‌ها قرار می‌گیرند.

- مطلوبیت: مطلوبیت هر عامل بر اساس نیازها و وابستگی‌هایش به محیط و سایر عامل‌ها تعریف شده و به صورت سود منهای هزینه، با توجه به اهداف مختلف محاسبه می‌شود. در سناریوهای چندعاملی، از طریق یادگیری از محیط و تعامل با دیگران، مطلوبیت هر عامل بهینه می‌گردد.

در این چارچوب، برای هر عامل در MARL تابع مطلوبیت خاصی در نظر گرفته شده و بر اساس مشاهدات و تجربیات حاصل از تعاملات، یادگیری سیاست به صورت مستقل انجام می‌شود تا ارزش مطلوبیت به حداکثر برسد، بدون اینکه مستقیماً به مطلوبیت سایر عامل‌ها توجه شود. این فرآیند ممکن است به رقابت یا همکاری میان عامل‌ها منجر گردد. با توجه به پیچیدگی تعاملات میان چندین عامل، تحلیل نظریه بازی‌ها به عنوان ابزاری مؤثر برای تصمیم‌گیری در این حوزه به کار گرفته می‌شود. بسته به سناریوهای مختلف، این بازی‌ها در دسته‌بندی‌های متفاوتی قرار داده شده که در بخش‌های بعدی بررسی خواهند شد.



شکل ۶-۱: حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط

۲-۶ نظریه بازی‌ها

نظریه بازی‌ها شاخه‌ای از ریاضیات است که به مطالعه تصمیم‌گیری در موقعیت‌هایی می‌پردازد که نتیجه انتخاب‌های هر فرد به تصمیمات دیگران وابسته است. این نظریه چارچوبی برای تحلیل تعاملات میان بازیکنان ارائه می‌دهد و در حوزه‌های مختلفی مانند اقتصاد، علوم سیاسی، زیست‌شناسی و علوم کامپیوتر کاربرد دارد. در این فصل، دو مفهوم کلیدی نظریه بازی‌ها یعنی تعادل نش و بازی‌های مجموع صفر بررسی شده است.

۱-۲-۶ تعادل نش

تعادل نش^۵ یکی از بنیادی‌ترین مفاهیم در نظریه بازی‌ها است که توسط جان نش در سال ۱۹۵۰ معرفی شد. این مفهوم به مجموعه‌ای از بازی‌ها اشاره دارد که در آن هیچ بازیکنی نمی‌تواند با تغییر یک‌جانبه سیاست خود، سود بیشتری به دست آورد، به شرطی که سیاست‌های سایر بازیکنان ثابت بماند.

- تعریف تعادل نش: فرض کنید یک بازی با n بازیکن داریم. هر بازیکن i دارای مجموعه سیاست‌های Π_i و تابع مطلوبیت $u_i: \Pi_1 \times \Pi_2 \times \dots \times \Pi_n \rightarrow \mathbb{R}$ است. یک مجموعه سیاست $\pi^* = (\pi_1^*, \pi_2^*, \dots, \pi_n^*)$ تعادل نش نامیده می‌شود اگر برای هر بازیکن i و هر سیاست $\pi_i \in \Pi_i$ در وضعیت s داشته باشیم:

$$u_i(\pi_i^*, \pi_{-i}^*, s) \geq u_i(\pi_i, \pi_{-i}^*, s) \quad (۱-۶)$$

در اینجا، π_{-i}^* نشان‌دهنده سیاست‌های همه بازیکنان به جز بازیکن i است. در ادامه پژوهش جهت استفاده از چارچوب نظریه بازی در یادگیری تقویتی تابع مطلوبیت به‌گونه‌ای تعریف شده است که برابر با تابع ارزش $u_i(\pi_i, \pi_{-i}, s) = V_i^{\pi_i, \pi_{-i}}(s)$ باشد.

- اهمیت تعادل نش: تعادل نش نقطه‌ای را در بازی مشخص می‌کند که هر بازیکن بهترین پاسخ را نسبت به انتخاب‌های دیگران ارائه داده است. این مفهوم به‌ویژه در بازی‌های غیرهمکارانه، به‌عنوان پیش‌بینی رفتار منطقی بازیکنان استفاده می‌شود و در زمینه‌هایی مانند یادگیری تقویتی چند عامله کاربرد گسترده‌ای دارد.

⁵Nash Equilibrium

۶-۲-۲ بازی مجموع صفر

بازی‌های مجموع صفر^۶ دسته‌ای از بازی‌ها هستند که در آن‌ها تابع ارزش یک بازیکن دقیقاً برابر با ضرر بازیکن دیگر است؛ بنابراین، مجموع ارزش‌های همه بازیکنان در هر مرحله صفر خواهد بود.

• تعریف بازی مجموع صفر:

در یک بازی دو نفره، اگر تابع ارزش حالت (value) بازیکن اول $V_1^{(\pi_1, \pi_2)}(s)$ و بازیکن دوم $V_2^{(\pi_1, \pi_2)}(s)$ برای هر مجموعه سیاست (π_1, π_2) به گونه‌ای باشند که:

$$V_1^{(\pi_1, \pi_2)}(s) + V_2^{(\pi_1, \pi_2)}(s) = 0 \implies V_1^{(\pi_1, \pi_2)}(s) = -V_2^{(\pi_1, \pi_2)}(s), \quad (۲-۶)$$

آنگاه آن بازی را بازی مجموع صفر می‌نامیم.

به‌طور مشابه، اگر تابع ارزش-عمل برای دو بازیکن را با $Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2)$ و $Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2)$ نشان دهیم، باید برقرار باشد:

$$(۳-۶)$$

$$Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2) + Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2) = 0 \implies Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2) = -Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2).$$

• سیاست بهینه در بازی مجموع صفر:

در این بازی‌ها، هر بازیکن سیاستی را برمی‌گزیند که تابع ارزش خود را در برابر بهترین پاسخ حریف بیشینه کند؛ این انتخاب در نهایت به تعادل نش منجر می‌شود.

به‌صورت تابع ارزش حالت:

$$V_1^*(s) = \max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1, \pi_2)}(s), \quad (۴-۶)$$

$$V_2^*(s) = \max_{\pi_2} \min_{\pi_1} V_2^{(\pi_1, \pi_2)}(s). \quad (۵-۶)$$

و به‌صورت تابع ارزش-عمل:

$$Q_1^*(s, a_1, a_2) = \max_{\pi_1} \min_{\pi_2} Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2), \quad (۶-۶)$$

$$Q_2^*(s, a_1, a_2) = \max_{\pi_2} \min_{\pi_1} Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2). \quad (۷-۶)$$

^۶Zero-Sum Games

۳-۶ گرادیان سیاست عمیق قطعی دوعاملی

گرادیان سیاست عمیق قطعی چندعاملی^۷ توسعه‌ای از الگوریتم DDPG برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های دوعاملی مجموع صفر می‌پردازیم که در آن مجموع پاداش‌های دو عامل همواره صفر است (آنچه یک عامل به دست می‌آورد، عامل دیگر از دست می‌دهد).

۱-۳-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی

در محیط‌های چندعاملی، سیاست هر عامل مدام در حال تغییر است، که باعث می‌شود محیط از دید هر عامل غیرایستا^۸ شود. این مسئله چالش بزرگی برای الگوریتم‌های یادگیری تقویتی تک‌عاملی مانند DDPG ایجاد می‌کند، زیرا فرض ایستایی محیط را نقض می‌کند.

MA-DDPG با استفاده از رویکرد آموزش متمرکز، اجرای غیرمتمرکز^۹ این مشکل را حل می‌کند. در این رویکرد، هر عامل در زمان آموزش به اطلاعات کامل محیط دسترسی دارد، اما در زمان اجرا تنها از مشاهدات محلی خود استفاده می‌کند.

۲-۳-۶ معماری MA-DDPG در بازی‌های مجموع صفر

در یک بازی دوعاملی مجموع صفر، دو عامل با نمادهای ۱ و ۲ نشان داده می‌شوند. هر عامل دارای شبکه‌های منحصراً به فرد خود است:

- شبکه‌های بازیگر: $\mu_{\theta_1}(o_1)$ و $\mu_{\theta_2}(o_2)$ که مشاهدات محلی o_1 و o_2 را به اعمال a_1 و a_2 نگاشت می‌کنند.
- شبکه‌های منتقد: $Q_{\phi_1}(o_1, a_1, a_2)$ و $Q_{\phi_2}(o_2, a_2, a_1)$ که ارزش حالت-عمل را با توجه به مشاهدات و اعمال تمام عامل‌ها تخمین می‌زنند.
- شبکه‌های هدف: مشابه DDPG، برای پایدار کردن آموزش از شبکه‌های هدف استفاده می‌شود.

در بازی‌های مجموع صفر، پاداش‌ها رابطه $r_1 + r_2 = 0$ دارند که در آن r_1 و r_2 پاداش‌های دریافتی عامل‌ها هستند. در نتیجه، $r_2 = -r_1$ است که نمایانگر تضاد کامل منافع بین عامل‌هاست.

⁷Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG)

⁸Non-stationary

⁹Centralized Training, Decentralized Execution

۳-۳-۶ آموزش MA-DDPG در بازی‌های مجموع صفر

فرایند آموزش MA-DDPG برای بازی‌های مجموع صفر به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1, 2\}$ ، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_i, \mathcal{D}) = \mathbb{E}_{(o, a, r_i, o', d) \sim \mathcal{D}} \left[\left(Q_{\phi_i}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (۸-۶)$$

که در آن $o = (o_1, o_2)$ بردار مشاهدات، $a = (a_1, a_2)$ بردار اعمال، و y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1 - d)Q_{\phi_i, \text{targ}}(o'_i, \mu_{\theta_1, \text{targ}}(o'_1), \mu_{\theta_2, \text{targ}}(o'_2)) \quad (۹-۶)$$

توجه کنید که منتقد هر عامل به اعمال همه عامل‌ها دسترسی دارد، اما در بازی‌های مجموع صفر، عامل شماره ۲ جهت مخالف هدف عامل ۱ را دنبال می‌کند.

یادگیری سیاست

سیاست هر عامل با بیشینه کردن تابع Q مربوط به آن عامل به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{o \sim \mathcal{D}} [Q_{\phi_i}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i}))] \quad (۱۰-۶)$$

که در آن $-i$ نشان‌دهنده عامل مقابل است. با توجه به ماهیت بازی مجموع صفر، هر عامل تلاش می‌کند تا مطلوبیت خود را افزایش دهد، در حالی که مطلوبیت عامل دیگر به طور همزمان کاهش می‌یابد.

شبکه‌های هدف و بافر تجربه

مشابه DDPG، برای پایدار کردن آموزش، شبکه‌های هدف با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند:

$$\phi_{i,\text{targ}} \leftarrow \rho \phi_{i,\text{targ}} + (1 - \rho) \phi_i$$

$$\theta_{i,\text{targ}} \leftarrow \rho \theta_{i,\text{targ}} + (1 - \rho) \theta_i$$

همچنین، از یک بافر تکرار بازی مشترک برای ذخیره تجربیات استفاده می‌شود که شامل وضعیت‌ها، اعمال و پاداش‌های همه عامل‌هاست.

۴-۳-۶ اکتشاف در MA-DDPG

اکتشاف در MA-DDPG مشابه DDPG است، اما برای هر عامل به طور جداگانه اعمال می‌شود. در طی آموزش، به اعمال هر عامل نویز اضافه می‌شود:

$$a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \quad (۱۱-۶)$$

که در آن ϵ_i نویز اضافه شده به عامل i است.

۵-۳-۶ شبکه‌کد MA-DDPG برای بازی‌های دو عاملی مجموع صفر

در این بخش، شبکه‌کد الگوریتم MA-DDPG پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۵ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

الگوریتم ۵ گرادیان سیاست عمیق قطعی چندعاملی برای بازی‌های مجموع صفر

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2) ، پارامترهای تابع Q (ϕ_1, ϕ_2) ، بافر تکرار بازی خالی (\mathcal{D})

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید: $\phi_{i,\text{targ}} \leftarrow \phi_i, \theta_{i,\text{targ}} \leftarrow \theta_i$ برای $i \in \{1, 2\}$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات (o_1, o_2) را دریافت کنید

۴: برای هر عامل i ، عمل $a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به‌طوری‌که $\epsilon_i \sim \mathcal{N}$ است

۵: اعمال (a_1, a_2) را در محیط اجرا کنید

۶: مشاهدات بعدی (o'_1, o'_2) ، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

۷: تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر \mathcal{D} ذخیره کنید

۸: اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان به‌روزرسانی فرا رسیده است:

۱۰: به ازای هر تعداد به‌روزرسانی:

۱۱: یک دسته تصادفی از تجربیات، $B = \{(o, a, r_1, r_2, o', d)\}$ ، از \mathcal{D} نمونه‌گیری کنید اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1 - d)Q_{\phi_1, \text{targ}}(o'_1, \mu_{\theta_1, \text{targ}}(o'_1), \mu_{\theta_2, \text{targ}}(o'_2))$$

$$y_2 = r_2 + \gamma(1 - d)Q_{\phi_2, \text{targ}}(o'_2, \mu_{\theta_2, \text{targ}}(o'_2), \mu_{\theta_1, \text{targ}}(o'_1))$$

۱۲: توابع Q را با نزول گرادیان به‌روزرسانی کنید:

$$\nabla_{\phi_1} \frac{1}{|B|} \sum_{(o, a, r_1, r_2, o', d) \in B} (Q_{\phi_1}(o_1, a_1, a_2) - y_1)^2$$

$$\nabla_{\phi_2} \frac{1}{|B|} \sum_{(o, a, r_1, r_2, o', d) \in B} (Q_{\phi_2}(o_2, a_2, a_1) - y_2)^2$$

۱۳: سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_1}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_2}(o_2, \mu_{\theta_2}(o_2), a_1)$$

۱۴: شبکه‌های هدف را به‌روزرسانی کنید:

$$\phi_{1, \text{targ}} \leftarrow \rho \phi_{1, \text{targ}} + (1 - \rho) \phi_1$$

$$\phi_{2, \text{targ}} \leftarrow \rho \phi_{2, \text{targ}} + (1 - \rho) \phi_2$$

$$\theta_{1, \text{targ}} \leftarrow \rho \theta_{1, \text{targ}} + (1 - \rho) \theta_1$$

$$\theta_{2, \text{targ}} \leftarrow \rho \theta_{2, \text{targ}} + (1 - \rho) \theta_2$$

۶-۳-۶ مزایای MA-DDPG در بازی‌های مجموع صفر

MA-DDPG چندین مزیت برای یادگیری در بازی‌های دو عاملی مجموع صفر ارائه می‌دهد:

- **مقابله با غیرایستایی:** با استفاده از منتقدهایی که به اطلاعات کامل دسترسی دارند، مشکل غیرایستایی محیط از دید هر عامل حل می‌شود.
- **همگرایی بهتر:** در بازی‌های مجموع صفر، MA-DDPG معمولاً همگرایی بهتری نسبت به آموزش مستقل عامل‌ها با DDPG نشان می‌دهد.
- **یادگیری استراتژی‌های متقابل:** عامل‌ها می‌توانند استراتژی‌های متقابل پیچیده را یاد بگیرند که در آموزش مستقل امکان‌پذیر نیست.

در بازی‌های دو عاملی مجموع صفر، این رویکرد به رقابت کامل بین عامل‌ها منجر می‌شود، که هر یک تلاش می‌کند بهترین استراتژی را در برابر استراتژی رقیب پیدا کند.

۶-۴ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی^{۱۰} توسعه‌ای از الگوریتم TD3 برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملی مجموع صفر می‌پردازیم که در آن ترکیب ویژگی‌های TD3 با رویکرد چندعاملی MADDPG به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۶-۴-۱ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MATD3

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، بیش‌برآورد تابع Q که در DDPG دیده می‌شود، در محیط‌های چندعاملی می‌تواند تشدید شود.

MATD3 هر دو چالش را با ترکیب رویکردهای زیر حل می‌کند:

- **آموزش متمرکز، اجرای غیرمتمرکز:** مشابه MADDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.

¹⁰Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3)

- منتقد‌های دوگانه: برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش‌برآورد تابع Q را کاهش دهد.

- به‌روزرسانی‌های تاخیری سیاست: سیاست‌ها را با تواتر کمتری نسبت به منتقد‌ها به‌روزرسانی می‌کند.

۶-۴-۲ معماری MATD3 در بازی‌های مجموع‌صفر

در یک بازی چندعاملی مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- شبکه بازیگر: $\mu_{\theta_i}(o_i)$ که مشاهدات محلی o_i را به اعمال a_i نگاشت می‌کند.
 - شبکه‌های منتقد دوگانه: $Q_{\phi_{i,1}}(o_i, a_1, a_2)$ و $Q_{\phi_{i,2}}(o_i, a_1, a_2)$ که ارزش حالت-عمل را تخمین می‌زنند.
 - شبکه‌های هدف: برای پایدارسازی آموزش، از نسخه‌های هدف بازیگر و منتقد‌ها استفاده می‌شود.
- در بازی‌های مجموع‌صفر، پاداش‌ها رابطه $r_1 + r_2 = 0$ دارند، بنابراین $r_2 = -r_1$ است.

۶-۴-۳ آموزش MATD3

فرایند آموزش MATD3 به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1, 2\}$ و هر منتقد $j \in \{1, 2\}$ ، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \mathbb{E}_{(o, a, r_i, o', d) \sim \mathcal{D}} \left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (۱۲-۶)$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1 - d) \min_{j=1,2} Q_{\phi_{i,j}, \text{targ}}(o'_i, \mu_{\theta_{1, \text{targ}}}(o'_1), \mu_{\theta_{2, \text{targ}}}(o'_2)) \quad (۱۳-۶)$$

استفاده از عملگر حداقل روی دو منتقد، بیش‌برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

یادگیری سیاست با تاخیر

سیاست هر عامل با تاخیر (معمولاً پس از هر دو به‌روزرسانی منتقدها) و با بیشینه کردن تابع Q اول به‌روزرسانی می‌شود:

$$\max_{\theta_i} E_{o \sim D} [Q_{\phi_{i,1}}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i}))] \quad (۱۴-۶)$$

به‌روزرسانی تاخیری سیاست اجازه می‌دهد تا منتقدها قبل از تغییر سیاست به مقادیر دقیق‌تری همگرا شوند.

شبکه‌های هدف

مشابه TD3، شبکه‌های هدف با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند:

$$\begin{aligned} \phi_{i,j,\text{targ}} &\leftarrow \rho \phi_{i,j,\text{targ}} + (1 - \rho) \phi_{i,j} \quad \text{برای } j = 1, 2 \\ \theta_{i,\text{targ}} &\leftarrow \rho \theta_{i,\text{targ}} + (1 - \rho) \theta_i \end{aligned}$$

۴-۴-۶ اکتشاف در MATD3

اکتشاف در MATD3 با افزودن نویز به اعمال هر عامل انجام می‌شود:

$$a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \quad (۱۵-۶)$$

که در آن $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$ است و مقدار σ_i به مرور زمان کاهش می‌یابد.

۵-۴-۶ شبکه‌کد MATD3 برای بازی‌های چندعاملی مجموع صفر

در این بخش، شبکه‌کد الگوریتم MATD3 پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۶ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

الگوریتم ۶ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه دو عاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2) ، پارامترهای توابع Q $(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$ ، بافر تکرار بازی خالی (D)

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$$j \in \{1, 2\} \text{ و } i \in \{1, 2\} \text{ برای } \phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}, \theta_{i,\text{targ}} \leftarrow \theta_i$$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات (o_1, o_2) را دریافت کنید

۴: برای هر عامل i ، عمل $a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به طوری که $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$ است

۵: اعمال (a_1, a_2) را در محیط اجرا کنید

۶: مشاهدات بعدی (o'_1, o'_2) ، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

۷: تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر D ذخیره کنید

۸: اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان به روزرسانی فرا رسیده است:

۱۰: به ازای j در هر تعداد به روزرسانی:

۱۱: یک دسته تصادفی از تجربیات، $B = \{(o, a, r_1, r_2, o', d)\}$ ، از D نمونه‌گیری کنید

۱۲: اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1 - d) \min_{k=1,2} Q_{\phi_{1,k,\text{targ}}}(o'_1, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2))$$

$$y_2 = r_2 + \gamma(1 - d) \min_{k=1,2} Q_{\phi_{2,k,\text{targ}}}(o'_2, \mu_{\theta_{2,\text{targ}}}(o'_2), \mu_{\theta_{1,\text{targ}}}(o'_1))$$

۱۳: توابع Q را با نزول گرادیان به روزرسانی کنید:

$$\nabla_{\phi_{1,k}} \frac{1}{|B|} \sum_B (Q_{\phi_{1,k}}(o_1, a_1, a_2) - y_1)^2 \quad \text{برای } k = 1, 2$$

$$\nabla_{\phi_{2,k}} \frac{1}{|B|} \sum_B (Q_{\phi_{2,k}}(o_2, a_2, a_1) - y_2)^2 \quad \text{برای } k = 1, 2$$

۱۴: اگر باقیمانده j بر تاخیر سیاست برابر ۰ باشد:

۱۵: سیاست‌ها را با صعود گرادیان به روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_{1,1}}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_{2,1}}(o_2, \mu_{\theta_2}(o_2), a_1)$$

۱۶: شبکه‌های هدف را به روزرسانی کنید:

$$\phi_{i,k,\text{targ}} \leftarrow \rho \phi_{i,k,\text{targ}} + (1 - \rho) \phi_{i,k} \quad \text{برای } i, k \in \{1, 2\}$$

$$\theta_{i,\text{targ}} \leftarrow \rho \theta_{i,\text{targ}} + (1 - \rho) \theta_i \quad \text{برای } i \in \{1, 2\}$$

۶-۴-۶ مزایای MATD3 در بازی‌های مجموع صفر

MATD3 مزایای زیر را نسبت به MADDPG در بازی‌های چندعاملی مجموع صفر ارائه می‌دهد:

- پایداری بیشتر: با استفاده از منتقدهای دوگانه، بیش‌برآورد تابع Q که در محیط‌های غیرایستای چند-عاملی شدیدتر است، کاهش می‌یابد.
 - یادگیری کارآمدتر: به‌روزرسانی‌های تاخیری سیاست اجازه می‌دهد منتقدها به تخمین‌های دقیق‌تری دست یابند، که منجر به بهبود کیفیت یادگیری سیاست می‌شود.
 - مقاومت در برابر نویز: ترکیب منتقدهای دوگانه با رویکرد آموزش متمرکز، مقاومت الگوریتم در برابر نویز و تغییرات محیط را افزایش می‌دهد.
 - همگرایی بهتر: بهبودهای TD3 در کنار رویکرد چندعاملی، به همگرایی سریع‌تر و پایداری در بازی‌های رقابتی منجر می‌شود.
- در مجموع، MATD3 ترکیبی از بهترین ویژگی‌های TD3 و MADDPG را ارائه می‌دهد که آن را به گزینه‌ای مناسب برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملی مجموع صفر تبدیل می‌کند.

۶-۵ عامل عملگر نقاد نرم چندعاملی

عامل عملگر نقاد نرم دوعاملی^{۱۱} توسعه‌ای از الگوریتم SAC برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملی مجموع صفر می‌پردازیم که در آن ترکیب ویژگی‌های SAC با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۶-۵-۱ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه‌حل MASAC

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، چالش‌های مربوط به تعادل اکتشاف-بهره‌برداری در محیط‌های چندعاملی پیچیده‌تر است.

MASAC این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

¹¹Multi-Agent Soft Actor-Critic (MASAC)

- آموزش متمرکز، اجرای غیرمتمرکز: مشابه MADDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.
- سیاست‌های تصادفی: برخلاف MADDPG و MATD3 که سیاست‌های قطعی دارند، MASAC از سیاست‌های تصادفی استفاده می‌کند.
- تنظیم آنتروپی: با استفاده از تنظیم آنتروپی، اکتشاف و همگرایی به سیاست‌های بهتر را بهبود می‌بخشد.
- منتقد‌های دوگانه: برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش‌برآورد تابع Q را کاهش دهد.

۲-۵-۶ معماری MASAC در بازی‌های مجموع‌صفر

در یک بازی چندعاملی مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- شبکه بازیگر: $\pi_{\theta_i}(a_i|o_i)$ که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.
- شبکه‌های منتقد دوگانه: $Q_{\phi_{i,1}}(o_i, a_1, a_2)$ و $Q_{\phi_{i,2}}(o_i, a_1, a_2)$ که ارزش حالت-عمل را تخمین می‌زنند.
- شبکه‌های هدف: برای پایداری آموزش، از نسخه‌های هدف منتقد‌ها استفاده می‌شود.
- در بازی‌های مجموع‌صفر، پاداش‌ها رابطه $r_1 + r_2 = 0$ دارند، بنابراین $r_2 = -r_1$ است.

۳-۵-۶ آموزش MASAC

فرایند آموزش MASAC به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1, 2\}$ و هر منتقد $j \in \{1, 2\}$ ، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \mathbb{E}_{(o, a, r_i, o', d) \sim \mathcal{D}} \left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (۱۶-۶)$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1 - d) \left(\min_{j=1,2} Q_{\phi_{i,j},\text{targ}}(o'_i, \tilde{a}'_1, \tilde{a}'_2) - \alpha_i \log \pi_{\theta_i}(\tilde{a}'_i | o'_i) \right) \quad (۱۷-۶)$$

که در آن $\tilde{a}'_i \sim \pi_{\theta_i}(\cdot | o'_i)$ است. استفاده از عملگر حداقل روی دو منتقد، بیش‌برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

یادگیری سیاست

سیاست هر عامل با بیشینه کردن ترکیبی از تابع Q و آنتروپی به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{\mathbf{o} \sim \mathcal{D}} \left[\min_{j=1,2} Q_{\phi_{i,j}}(o_i, \tilde{a}_i, a_{-i}) - \alpha_i \log \pi_{\theta_i}(\tilde{a}_i | o_i) \right] \quad (۱۸-۶)$$

که در آن $\tilde{a}_i \sim \pi_{\theta_i}(\cdot | o_i)$ است و از ترفند پارامترسازی مجدد برای استخراج گرادیان استفاده می‌شود:

$$\tilde{a}_{i,\theta_i}(o_i, \xi_i) = \tanh(\mu_{\theta_i}(o_i) + \sigma_{\theta_i}(o_i) \odot \xi_i), \quad \xi_i \sim \mathcal{N} \quad (۱۹-۶)$$

شبکه‌های هدف

مشابه SAC، شبکه‌های هدف منتقد با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند:

$$\phi_{i,j,\text{targ}} \leftarrow \rho \phi_{i,j,\text{targ}} + (1 - \rho) \phi_{i,j} \quad \text{برای } j = 1, 2 \quad (۲۰-۶)$$

تنظیم ضریب آنتروپی

یکی از مزایای MASAC، توانایی تنظیم خودکار ضریب آنتروپی α_i برای هر عامل است که می‌تواند با استفاده از یک تابع هزینه مجزا بهینه شود:

$$\min_{\alpha_i} \mathbb{E}_{\mathbf{o} \sim \mathcal{D}, \tilde{a}_i \sim \pi_{\theta_i}} \left[-\alpha_i \left(\log \pi_{\theta_i}(\tilde{a}_i | o_i) + H_{\text{target}} \right) \right] \quad (۲۱-۶)$$

که در آن H_{target} آنتروپی هدف است که به عنوان یک ابرپارامتر تعیین می‌شود.

۴-۵-۶ اکتشاف در MASAC

اکتشاف در MASAC به صورت ذاتی از طریق سیاست‌های تصادفی و تنظیم آنتروپی انجام می‌شود. برخلاف MADDPG و MATD3 که به افزودن نویز به اعمال نیاز دارند، MASAC اعمال را مستقیماً از توزیع احتمال سیاست نمونه‌گیری می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot | o_i) \quad (۲۲-۶)$$

این رویکرد امکان اکتشاف ساختاریافته‌تر و کارآمدتر را فراهم می‌کند که در محیط‌های چندعاملی پیچیده مفید است.

۵-۵-۶ شبکه‌کد MASAC برای بازی‌های چندعاملی مجموع صفر

در این بخش، شبکه‌کد الگوریتم MASAC پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۷ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2) ، پارامترهای توابع Q $(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$ ، ضرایب

آنتروپی (α_1, α_2) ، بافر تکرار بازی خالی (\mathcal{D})

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$$j \in \{1, 2\} \text{ و } i \in \{1, 2\} \text{ برای } \phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}$$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات (o_1, o_2) را دریافت کنید

۴: برای هر عامل i ، عمل $a_i \sim \pi_{\theta_i}(\cdot|o_i)$ را انتخاب کنید

۵: اعمال (a_1, a_2) را در محیط اجرا کنید

۶: مشاهدات بعدی (o'_1, o'_2) ، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

۷: تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر \mathcal{D} ذخیره کنید

۸: اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان به‌روزرسانی فرا رسیده است:

۱۰: به ازای هر تعداد به‌روزرسانی:

۱۱: یک دسته تصادفی از تجربیات، $B = \{(o, a, r_1, r_2, o', d)\}$ ، از \mathcal{D} نمونه‌گیری کنید

۱۲: اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1 - d) \left(\min_{j=1,2} Q_{\phi_{1,j,\text{targ}}}(o'_1, \tilde{a}'_1, \tilde{a}'_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}'_1|o'_1) \right)$$

$$y_2 = r_2 + \gamma(1 - d) \left(\min_{j=1,2} Q_{\phi_{2,j,\text{targ}}}(o'_2, \tilde{a}'_2, \tilde{a}'_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}'_2|o'_2) \right)$$

۱۳: توابع Q را با نزول گرادیان به‌روزرسانی کنید:

$$\nabla_{\phi_{1,j}} \frac{1}{|B|} \sum_B (Q_{\phi_{1,j}}(o_1, a_1, a_2) - y_1)^2 \quad \text{برای } j = 1, 2$$

$$\nabla_{\phi_{2,j}} \frac{1}{|B|} \sum_B (Q_{\phi_{2,j}}(o_2, a_2, a_1) - y_2)^2 \quad \text{برای } j = 1, 2$$

۱۴: سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} \left[\min_{j=1,2} Q_{\phi_{1,j}}(o_1, \tilde{a}_{1,\theta_1}(o_1, \xi_1), a_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1)|o_1) \right]$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} \left[\min_{j=1,2} Q_{\phi_{2,j}}(o_2, \tilde{a}_{2,\theta_2}(o_2, \xi_2), a_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2)|o_2) \right]$$

۱۵: ضرایب آنتروپی را با نزول گرادیان به‌روزرسانی کنید (اختیاری):

$$\nabla_{\alpha_1} \frac{1}{|B|} \sum_{o \in B} -\alpha_1 \left(\log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1)|o_1) + H_{\text{target}} \right)$$

$$\nabla_{\alpha_2} \frac{1}{|B|} \sum_{o \in B} -\alpha_2 \left(\log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2)|o_2) + H_{\text{target}} \right)$$

۱۶: شبکه‌های هدف را به‌روزرسانی کنید:

$$\phi_{i,j,\text{targ}} \leftarrow \rho \phi_{i,j,\text{targ}} + (1 - \rho) \phi_{i,j} \quad \text{برای } i, j \in \{1, 2\}$$

۶-۵-۶ مزایای MASAC در بازی‌های مجموع‌صفر

MASAC مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی در بازی‌های چندعاملی مجموع‌صفر ارائه می‌دهد:

- **اکتشاف بهتر:** استفاده از سیاست‌های تصادفی و تنظیم آنتروپی، اکتشاف فضای حالت-عمل را بهبود می‌بخشد که برای یافتن راه‌حل‌های بهینه در بازی‌های دوعاملی ضروری است.
- **ثبات بیشتر:** ترکیب منتقد‌های دوگانه با تنظیم آنتروپی، یادگیری را پایدارتر می‌کند و از همگرایی زودهنگام به سیاست‌های ضعیف جلوگیری می‌کند.
- **سازگاری با محیط‌های پیچیده:** توانایی تنظیم خودکار تعادل بین اکتشاف و بهره‌برداری، MASAC را برای محیط‌های چندعاملی پیچیده مناسب می‌سازد.
- **عملکرد بهتر در مسائل با چندین بهینه محلی:** سیاست‌های تصادفی می‌توانند از دام‌های بهینه محلی فرار کنند و به راه‌حل‌های بهتر برسند.

در مجموع، MASAC ترکیبی از ویژگی‌های مثبت SAC و رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملی مجموع‌صفر تبدیل می‌کند، به‌ویژه در محیط‌هایی که اکتشاف کارآمد و سیاست‌های تصادفی اهمیت دارند.

۶-۶ عامل بهینه‌سازی سیاست مجاور چندعاملی

عامل بهینه‌سازی سیاست مجاور دوعاملی^{۱۲} توسعه‌ای از الگوریتم PPO برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملی مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های PPO با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۱-۶-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه‌حل MAPPO

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. این چالش با پیچیدگی‌های ذاتی الگوریتم‌های مبتنی بر گرادینان سیاست مانند PPO ترکیب می‌شود.

¹²Multi-Agent Proximal Policy Optimization (MAPPO)

MAPPO این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

- آموزش متمرکز، اجرای غیرمتمرکز: مشابه سایر الگوریتم‌های چندعاملی، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند، اما بازیگران تنها به مشاهدات محلی خود دسترسی دارند.
- به‌روزرسانی کلیپ‌شده: استفاده از مکانیسم کلیپ شده PPO برای محدود کردن به‌روزرسانی‌های سیاست، که به پایداری بیشتر در یادگیری چندعاملی کمک می‌کند.
- بافر تجربه مشترک: استفاده از یک بافر تجربه مشترک که تعاملات بین عامل‌ها را ثبت می‌کند.

۲-۶-۶ معماری MAPPO در بازی‌های مجموع‌صفر

در یک بازی چندعاملی مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- شبکه بازیگر: $\pi_{\theta_i}(a_i|o_i)$ که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.
- شبکه منتقد: $V_{\phi_i}(o_i, a_1, a_2)$ که ارزش حالت را تخمین می‌زند و برای محاسبه تابع مزیت استفاده می‌شود.

در بازی‌های مجموع‌صفر، پاداش‌ها رابطه $r_1 + r_2 = 0$ دارند، بنابراین $r_2 = -r_1$ است.

۳-۶-۶ آموزش MAPPO

فرایند آموزش MAPPO به شرح زیر است:

جمع‌آوری تجربیات

در هر تکرار، عامل‌ها با استفاده از سیاست‌های فعلی خود در محیط تعامل می‌کنند و مجموعه‌ای از مسیرها را جمع‌آوری می‌کنند:

$$\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\} \quad (۲۳-۶)$$

محاسبه مزیت

برای هر عامل $i \in \{1, 2\}$ ، تابع مزیت با استفاده از تابع ارزش فعلی محاسبه می‌شود. روش‌های مختلفی برای محاسبه مزیت وجود دارد؛ یک روش متداول استفاده از تخمین‌زننده مزیت تعمیم‌یافته (GAE) است:

$$\hat{A}_i^t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{i,t+l} \quad (24-6)$$

که در آن $\delta_{i,t} = r_i^t + \gamma V_{\phi_i}(o_i^{t+1}) - V_{\phi_i}(o_i^t)$ است.

به‌روزرسانی سیاست

سیاست هر عامل با بیشینه کردن تابع هدف PPO-Clip به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{(o_i, a_i) \sim \mathcal{D}_k} \left[\min \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, \text{clip} \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_i \right) \right] \quad (25-6)$$

یا با استفاده از همان فرمول‌بندی ساده‌تر:

$$\max_{\theta_i} \mathbb{E}_{(o_i, a_i) \sim \mathcal{D}_k} \left[\min \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, g(\epsilon, \hat{A}_i) \right) \right] \quad (26-6)$$

که تابع g به صورت زیر تعریف شده است:

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0 \end{cases} \quad (27-6)$$

به‌روزرسانی منتقد

تابع ارزش هر عامل با کمینه کردن خطای میانگین مربعات به‌روزرسانی می‌شود:

$$\min_{\phi_i} \mathbb{E}_{(o_i, \hat{R}_i) \sim \mathcal{D}_k} \left[\left(V_{\phi_i}(o_i) - \hat{R}_i \right)^2 \right] \quad (28-6)$$

که در آن \hat{R}_i بازده تنزیل شده برای عامل i است.

۴-۶-۶ اکتشاف در MAPPO

اکتشاف در MAPPO به صورت ذاتی از طریق سیاست‌های تصادفی انجام می‌شود. برخلاف الگوریتم‌های مبتنی بر DDPG که به افزودن نویز به اعمال نیاز دارند، MAPPO از توزیع احتمال سیاست برای اکتشاف استفاده می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot | o_i) \quad (۲۹-۶)$$

این رویکرد اکتشاف سیاست‌محور، در ترکیب با مکانیسم کلیپ PPO که از به‌روزرسانی‌های بزرگ سیاست جلوگیری می‌کند، به ثبات بیشتر در یادگیری چندعاملی کمک می‌کند.

۵-۶-۶ شبکه‌کد MAPPO برای بازی‌های چندعاملی مجموع صفر

در این بخش، شبکه‌کد الگوریتم MAPPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۸ در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

الگوریتم ۸ عامل بهینه‌سازی سیاست مجاور دوعاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2) ، پارامترهای تابع ارزش (ϕ_1, ϕ_2)

۱: به ازای $k = 0, 1, 2, \dots$

۲: مجموعه‌ای از مسیرها به نام $\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\}$ با اجرای سیاست‌های π_{θ_1} و π_{θ_2} در محیط جمع‌آوری شود.

۳: برای هر عامل i ، پاداش‌های باقی‌مانده \hat{R}_i^t را محاسبه کنید.

۴: برای هر عامل i ، برآوردهای مزیت \hat{A}_i^t را با استفاده از تابع ارزش فعلی V_{ϕ_i} محاسبه کنید.

۵: برای هر عامل i ، سیاست را با به حداکثر رساندن تابع هدف PPO-Clip به‌روزرسانی کنید:

$$\theta_{i,k+1} = \arg \max_{\theta_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i, a_i) \in \mathcal{D}_k} \min \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, g(\epsilon, \hat{A}_i) \right)$$

۶: برای هر عامل i ، تابع ارزش را با رگرسیون بر روی میانگین مربعات خطا به‌روزرسانی کنید:

$$\phi_{i,k+1} = \arg \min_{\phi_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i) \in \mathcal{D}_k} (V_{\phi_i}(o_i) - \hat{R}_i)^2$$

۶-۶-۶ مزایای MAPPO در بازی‌های مجموع‌صفر

MAPPO مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی در بازی‌های چندعاملی مجموع‌صفر ارائه می‌دهد:

- **پایداری یادگیری:** مکانیسم کلیپ PPO از به‌روزرسانی‌های بزرگ سیاست جلوگیری می‌کند که به پایداری بیشتر در محیط‌های غیرایستای چندعاملی منجر می‌شود.
- **کارایی نمونه:** نسبت به الگوریتم‌های خارج از سیاست مانند MATD3 و MASAC، MAPPO معمولاً کارایی نمونه بهتری دارد و به داده‌های کمتری برای یادگیری نیاز دارد.
- **اکتشاف سیاست‌محور:** اکتشاف ذاتی از طریق سیاست‌های تصادفی به جای افزودن نویز به اعمال، به اکتشاف کارآمدتر فضای حالت-عمل کمک می‌کند.
- **مقیاس‌پذیری:** MAPPO به راحتی به سیستم‌های با تعداد بیشتری از عامل‌ها قابل گسترش است، اگرچه در این پژوهش بر بازی‌های دوعاملی تمرکز شده‌است.

در مجموع، MAPPO ترکیبی از سادگی و کارایی PPO با رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری در بازی‌های چندعاملی مجموع صفر تبدیل می‌کند.

فصل ۷

سخت افزار در حلقه عملکرد عامل در محیط

فصل ۸

ارزیابی و نتایج یادگیری

در این فصل، نتایج حاصل از فرآیند یادگیری تقویتی در محیط سه‌جسمی ارائه و تحلیل شده است. هدف، بررسی عملکرد الگوریتم‌های استفاده‌شده و ارزیابی توانایی آن‌ها در دستیابی به اهداف تعیین‌شده می‌باشد.

۸-۱ تنظیمات آزمایشی

تنظیمات شبیه‌سازی، شامل پارامترهای محیط، نرخ یادگیری، و اندازه بافر تجربه، در این بخش تشریح شده است.

۸-۲ نتایج عملکرد الگوریتم‌ها

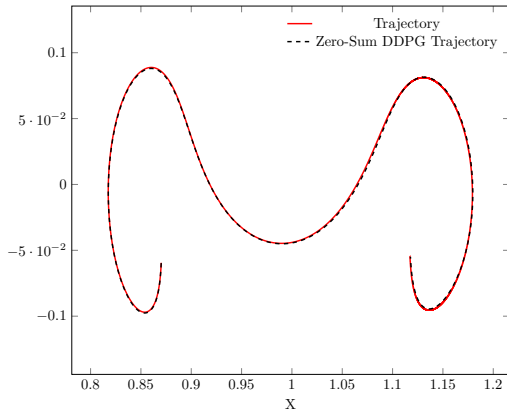
نتایج عملکرد الگوریتم‌های TD3، SAC، PPO، و DDPG با معیارهایی نظیر زمان رسیدن به هدف و مصرف سوخت گزارش شده است.

۸-۳ تحلیل پایداری و همگرایی

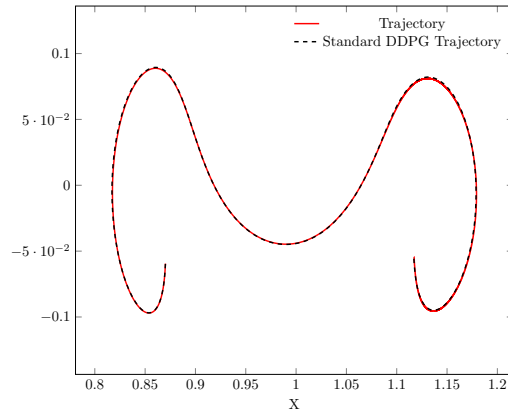
پایداری و سرعت همگرایی فرآیند یادگیری با استفاده از نمودارهای پاداش و معیارهای عددی مورد بررسی قرار گرفته است.

۴-۸ مقایسه با معیارهای مرجع

عملکرد الگوریتم‌ها با روش‌های مرجع مقایسه شده تا برتری‌ها و محدودیت‌های آن‌ها مشخص گردد.

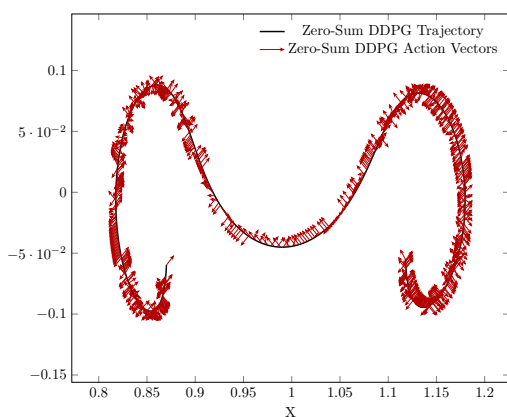


(ب) DDPG بازی مجموع صفر

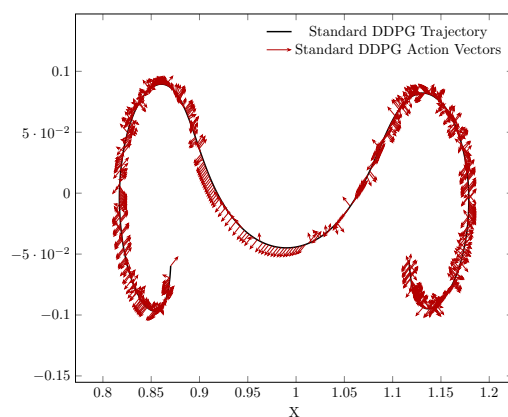


(آ) DDPG استاندارد

شکل ۸-۱: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی DDPG

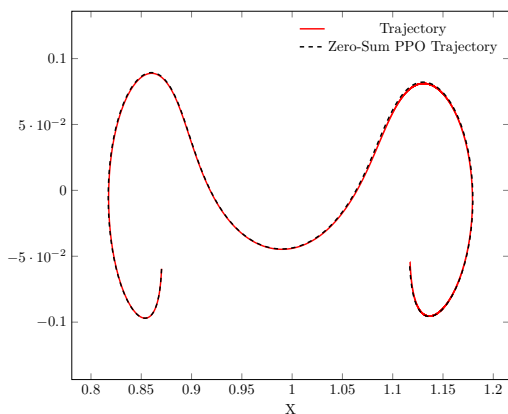


(ب) DDPG بازی مجموع صفر

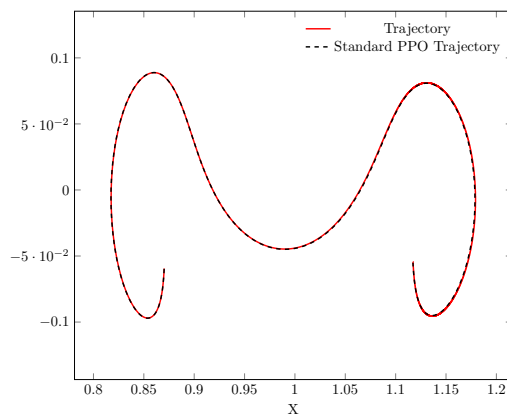


(آ) DDPG استاندارد

شکل ۸-۲: مقایسه مسیر و فرمان پیشران دو الگوریتم تک‌عاملی و چندعاملی DDPG

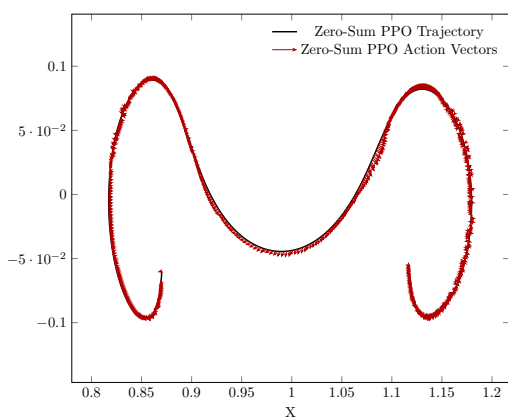


(ب) PPO بازی مجموع صفر

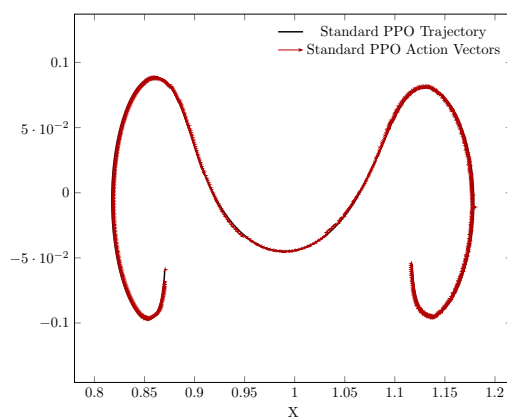


(آ) PPO استاندارد

شکل ۸-۳: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی PPO

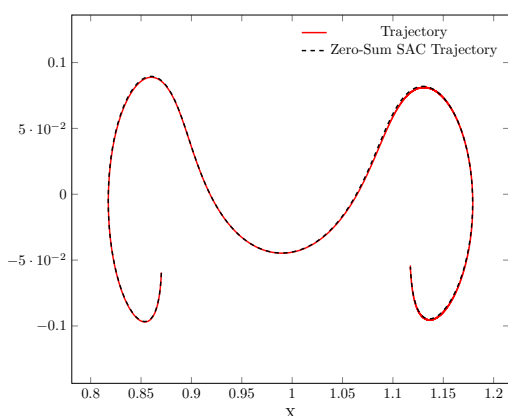


(ب) PPO بازی مجموع صفر

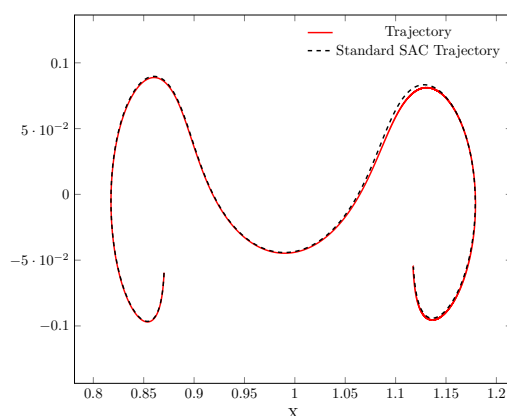


(آ) PPO استاندارد

شکل ۸-۴: مقایسه مسیر و فرمان پیش‌ران دو الگوریتم تک‌عاملی و چندعاملی PPO

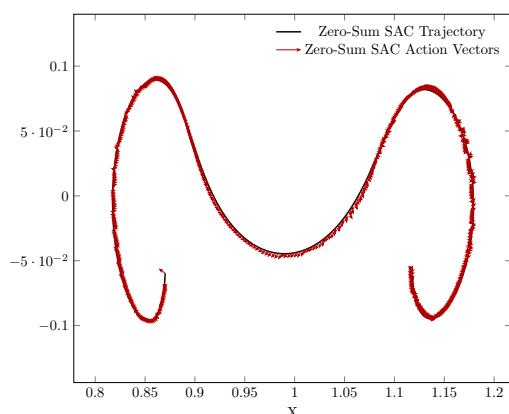


(ب) SAC بازی مجموع صفر

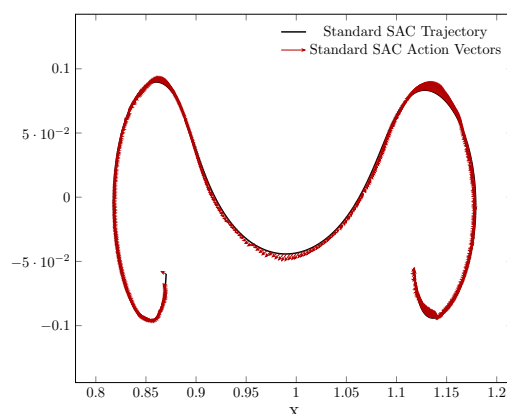


(آ) SAC استاندارد

شکل ۸-۵: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی SAC

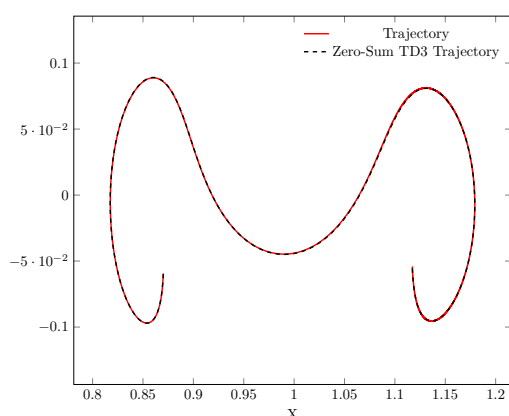


(ب) SAC بازی مجموع صفر

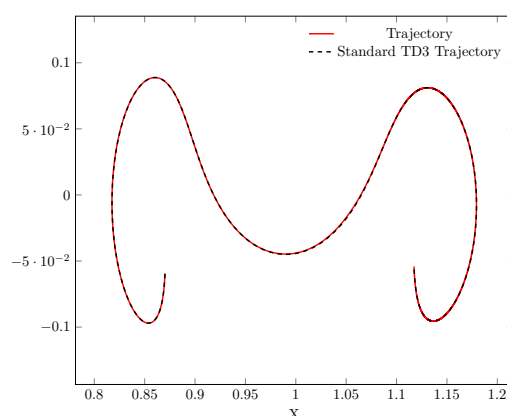


(آ) SAC استاندارد

شکل ۸-۶: مقایسه مسیر و فرمان پیشران دو الگوریتم تک عاملی و چند عاملی SAC

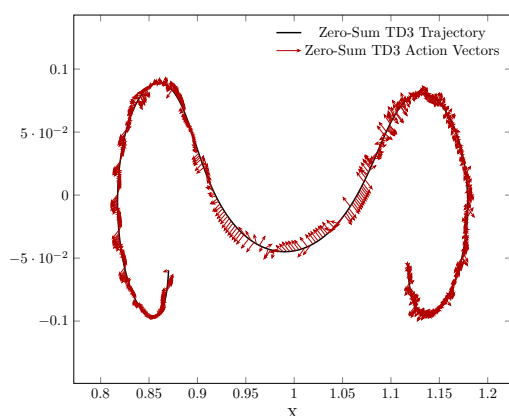


(ب) TD3 بازی مجموع صفر

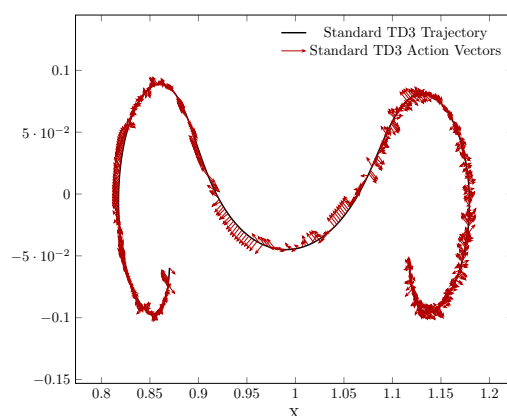


(آ) TD3 استاندارد

شکل ۸-۷: مقایسه مسیر طی شده در دو الگوریتم تک عاملی و چند عاملی TD3

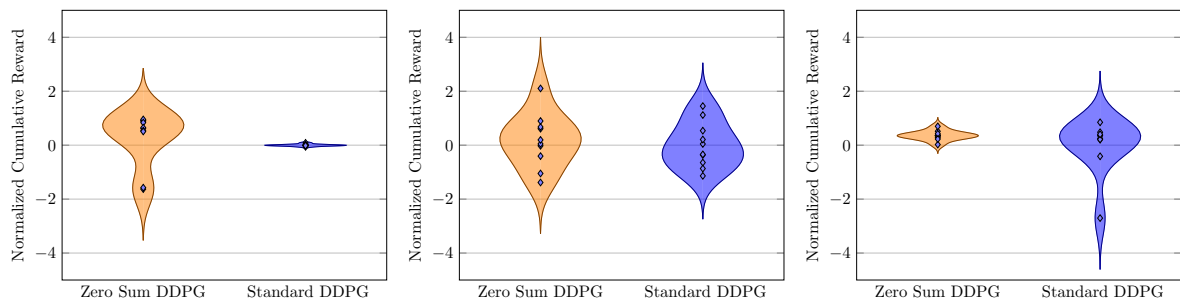


(ب) TD3 بازی مجموع صفر



(آ) TD3 استاندارد

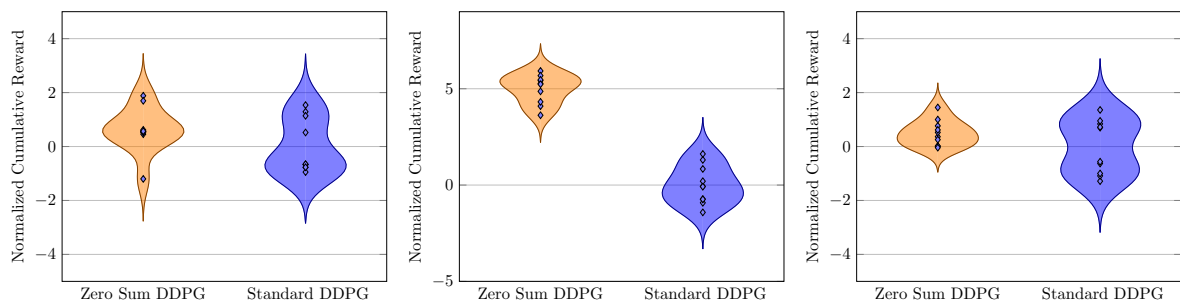
شکل ۸-۸: مقایسه مسیر و فرمان پیشران دو الگوریتم تک عاملی و چند عاملی TD3



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی

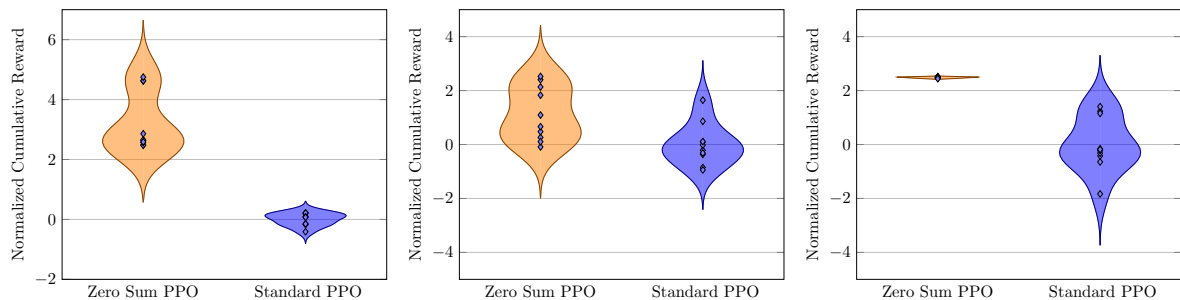


(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

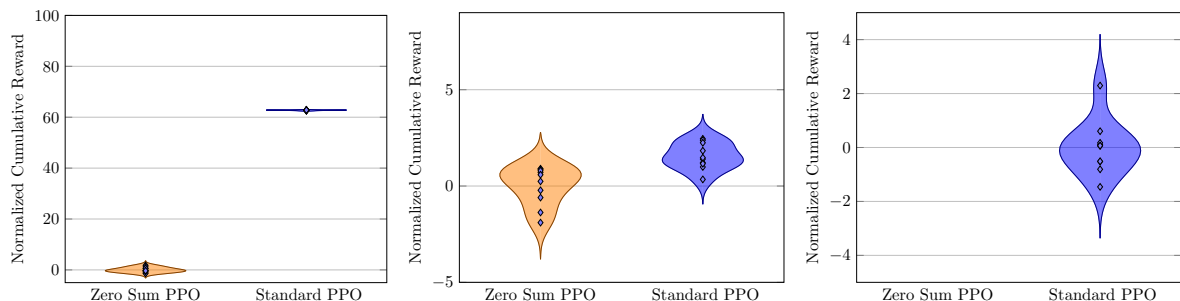
شکل ۸-۹: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی DDPG در سناریوهای مختلف



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی

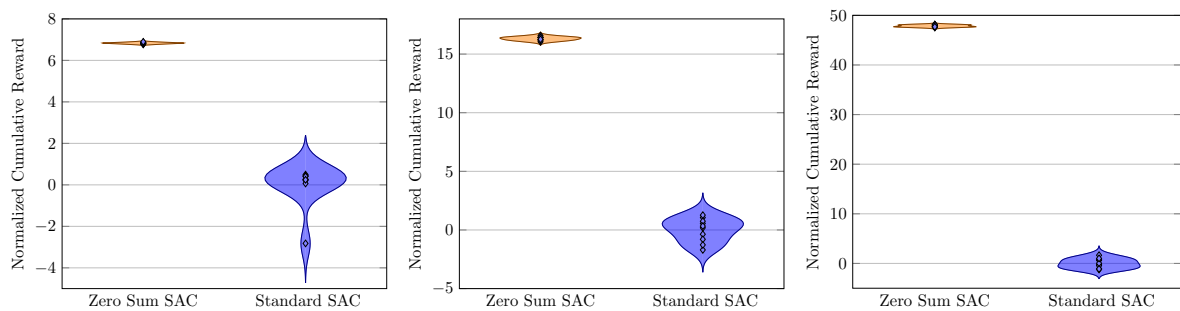


(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

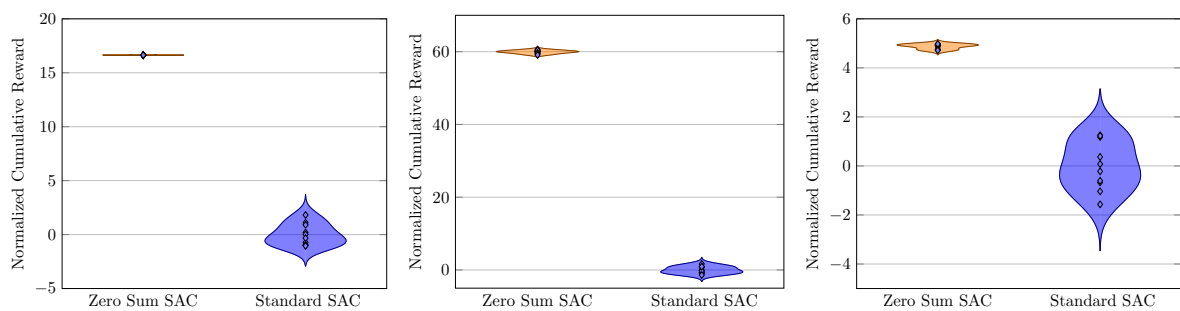
شکل ۸-۱۰: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی PPO در سناریوهای مختلف



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی

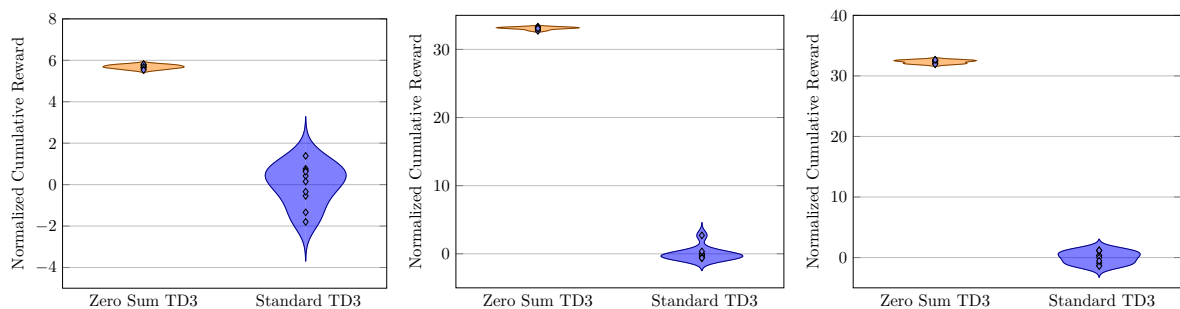


(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

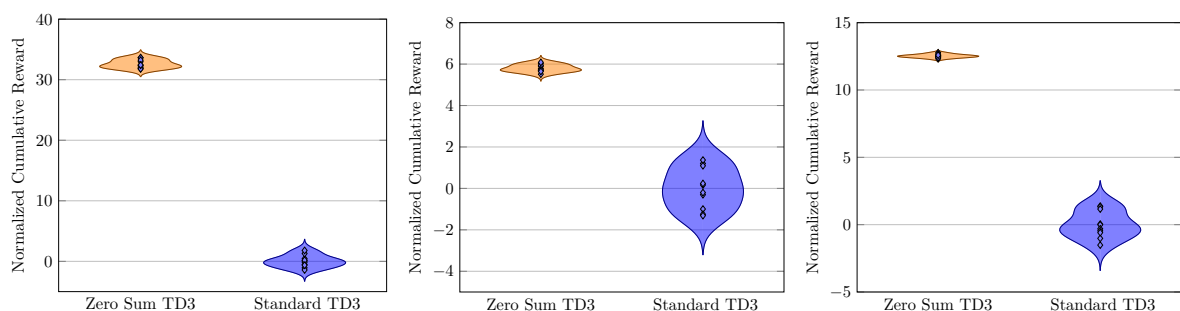
شکل ۸-۱۱: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی SAC در سناریوهای مختلف



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی

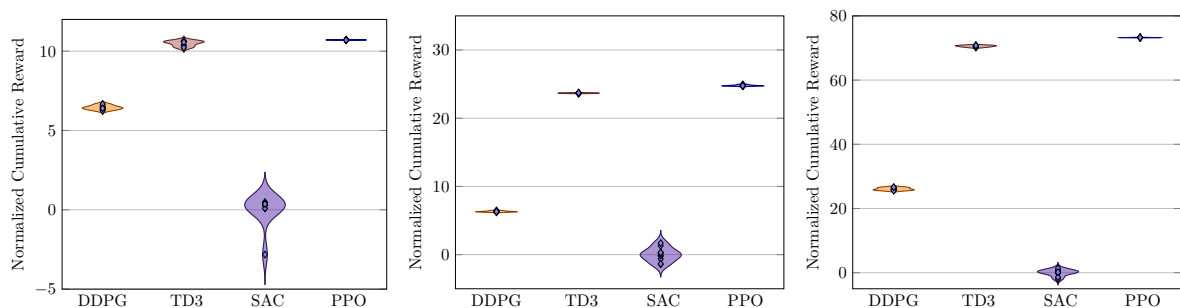


(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

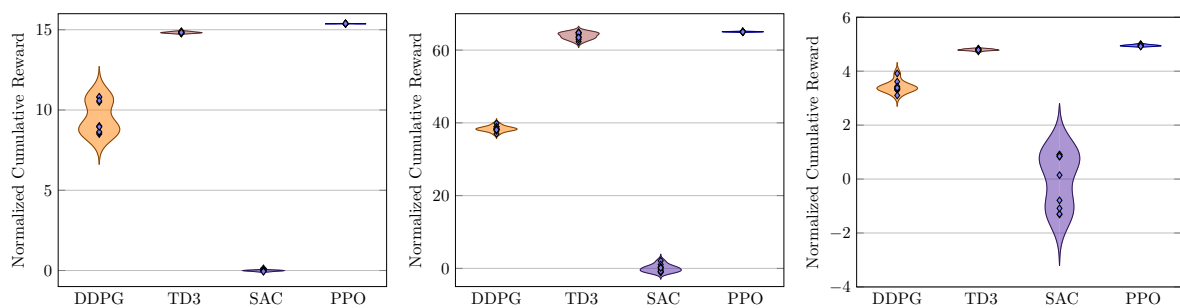
شکل ۸-۱۲: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی TD3 در سناریوهای مختلف



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی

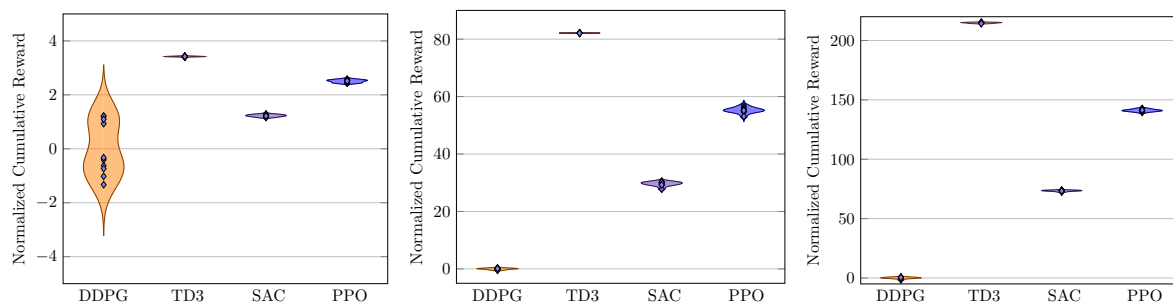


(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

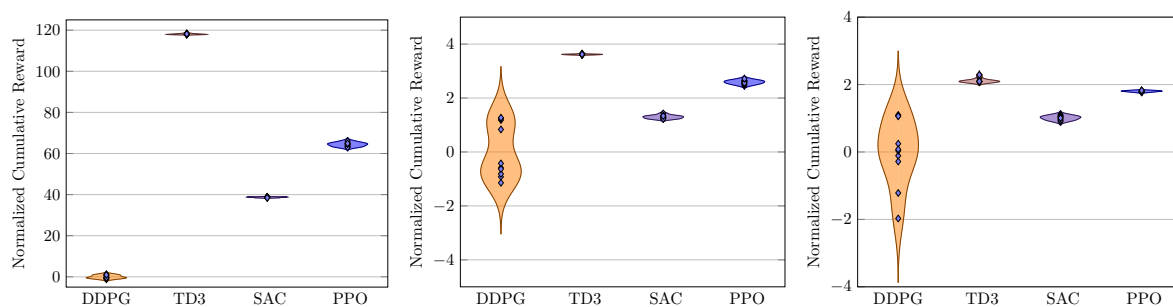
شکل ۸-۱۳: مقایسه مجموع پاداش الگوریتم‌های تک‌عاملی در سناریوهای مختلف



(ج) عدم تطابق مدل

(ب) اغتشاش در عملگرها

(آ) شرایط اولیه تصادفی



(و) تأخیر زمانی

(ه) نویز حسگر

(د) مشاهده ناقص

شکل ۸-۱۴: مقایسه مجموع پاداش الگوریتم‌های چندعاملی در سناریوهای مختلف

Bibliography

- [1] J. Achiam. Spinning Up in Deep Reinforcement Learning. 2018.
- [2] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, second edition, 2018.
- [3] M. A. Vavrina, J. A. Englander, S. M. Phillips, and K. M. Hughes. Global, multi-objective trajectory optimization with parametric spreading. In *AAS AIAA Astrodynamics Specialist Conference 2017*, 2017. Tech. No. GSFC-E-DAA-TN45282.
- [4] C. Ocampo. Finite burn maneuver modeling for a generalized spacecraft trajectory design and optimization system. *Annals of the New York Academy of Sciences*, 1017:210–233, 2004.
- [5] B. G. Marchand, S. K. Scarritt, T. A. Pavlak, and K. C. Howell. A dynamical approach to precision entry in multi-body regimes: Dispersion manifolds. *Acta Astronautica*, 89:107–120, 2013.
- [6] A. F. Haapala and K. C. Howell. A framework for constructing transfers linking periodic libration point orbits in the spatial circular restricted three-body problem. *International Journal of Bifurcation and Chaos*, 26(05):1630013, 2016.
- [7] B. Gaudet, R. Linares, and R. Furfaro. Six degree-of-freedom hovering over an asteroid with unknown environmental dynamics via reinforcement learning. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
- [8] B. Gaudet, R. Linares, and R. Furfaro. Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations. *Acta Astronautica*, 171:1–13, 2020.
- [9] A. Rubinsztein, R. Sood, and F. E. Laipert. Neural network optimal control in astrodynamics: Application to the missed thrust problem. *Acta Astronautica*, 176:192–203, 2020.

- [10] T. A. Estlin, B. J. Bornstein, D. M. Gaines, R. C. Anderson, D. R. Thompson, M. Burl, R. Castaño, and M. Judd. Aegis automated science targeting for the mer opportunity rover. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3:1–19, 2012.
- [11] R. Francis, T. Estlin, G. Doran, S. Johnstone, D. Gaines, V. Verma, M. Burl, J. Frydenvang, S. Montano, R. Wiens, S. Schaffer, O. Gasnault, L. Deflores, D. Blaney, and B. Bornstein. Aegis autonomous targeting for chemcam on mars science laboratory: Deployment and results of initial science team use. *Science Robotics*, 2, 2017.
- [12] S. Higa, Y. Iwashita, K. Otsu, M. Ono, O. Lamarre, A. Didier, and M. Hoffmann. Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics. *IEEE Robotics and Automation Letters*, 4:3876–3883, 2019.
- [13] B. Rothrock, J. Papon, R. Kennedy, M. Ono, M. Heverly, and C. Cunningham. Spoc: Deep learning-based terrain classification for mars rover missions. In *AIAA Space and Astronautics Forum and Exposition, SPACE 2016*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016.
- [14] K. L. Wagstaff, G. Doran, A. Davies, S. Anwar, S. Chakraborty, M. Cameron, I. Daubar, and C. Phillips. Enabling onboard detection of events of scientific interest for the europa clipper spacecraft. In *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2191–2201, Anchorage, Alaska, 2019.
- [15] B. Dachwald. Evolutionary neurocontrol: A smart method for global optimization of low-thrust trajectories. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pages 1–16, Providence, Rhode Island, 2004.
- [16] S. D. Smet and D. J. Scheeres. Identifying heteroclinic connections using artificial neural networks. *Acta Astronautica*, 161:192–199, 2019.
- [17] N. L. O. Parrish. *Low Thrust Trajectory Optimization in Cislunar and Translunar Space*. PhD thesis, University of Colorado Boulder, 2018.
- [18] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.
- [19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den

- Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550, 2017.
- [20] R. Furfaro, A. Scorsoglio, R. Linares, and M. Massari. Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, 171:156–171, 2020.
- [21] B. Gaudet, R. Linares, and R. Furfaro. Deep reinforcement learning for six degrees of freedom planetary landing. *Advances in Space Research*, 65:1723–1741, 2020.
- [22] B. Gaudet, R. Furfaro, and R. Linares. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerospace Science and Technology*, 99, 2020.
- [23] D. Guzzetti. Reinforcement learning and topology of orbit manifolds for station-keeping of unstable symmetric periodic orbits. In *AAS/AIAA Astrodynamics Specialist Conference*, Portland, Maine, 2019.
- [24] J. A. Reiter and D. B. Spencer. Augmenting spacecraft maneuver strategy optimization for detection avoidance with competitive coevolution. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
- [25] A. Das-Stuart, K. C. Howell, and D. C. Folta. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies. *Acta Astronautica*, 171:172–195, 2020.
- [26] D. Miller and R. Linares. Low-thrust optimal control via reinforcement learning. In *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka’anapali, Hawaii, 2019.
- [27] C. J. Sullivan and N. Bosanac. Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
- [28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.
- [29] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

- [30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016. arXiv:1602.01783.
- [31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2019.
- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017.
- [33] S. Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.
- [34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.
- [35] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 1179–1191, 2020.
- [36] K. Prudencio, J. L. Xiang, and A. T. Cemgil. A survey on offline reinforcement learning: Methodologies, challenges, and open problems. *arXiv preprint*, arXiv:2203.01387, 2022.
- [37] J. Garc a and F. Fern ndez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.
- [38] F. Ghazalpour, S. Samangouei, and R. Vaughan. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys*, 54(12):1–35, 2021.
- [39] K. Song, J. Zhu, Y. Chow, D. Psomas, and M. Wainwright. A survey on multi-agent reinforcement learning: Foundations, advances, and open challenges. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. In press, arXiv:2401.01234.
- [40] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach,

- K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [41] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [42] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1407–1416, 2018.
- [43] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 330–337, 1993.
- [44] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Robots*, 8(3):355–377, 2005.
- [45] L. Buşoniu, R. Babuška, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.
- [46] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6379–6390, 2017.
- [47] P. Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018. arXiv:1706.05296.
- [48] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4292–4301, 2018.
- [49] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, J. Foerster, N. Nardelli, T. G. J. Rudner, and et al. The starcraft multi-agent challenge. *arXiv preprint*, arXiv:1902.04043, 2019.

- [50] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5887–5896, 2019.
- [51] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 7611–7622, 2019.
- [52] T. Wang, Y. Jiang, T. Da, W. Zhang, and J. Wang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9876–9886, 2020.
- [53] K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of RL and Control*, 2021. arXiv:2106.05230.
- [54] A. Mitriakov, P. Papadakis, J. Kerdreux, and S. Garlatti. Reinforcement learning based, staircase negotiation learning: Simulation and transfer to reality for articulated tracked robots. *IEEE Robotics & Automation Magazine*, 28(4):10–20, 2021.
- [55] Y. Yu et al. Heterogeneous-agent reinforcement learning: An overview. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. In press, arXiv:2203.00596.
- [56] D. Vallado and W. McClain. *Fundamentals of Astrodynamics and Applications*. Fundamentals of Astrodynamics and Applications. Microcosm Press, 2001.
- [57] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.
- [58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

- [59] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods, 2018.
- [60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.
- [61] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

Abstract

In this study, a quadcopter stand with three degrees of freedom was controlled using game theory-based control. The first player tracks a desired input, and the second player creates a disturbance in the tracking of the first player to cause an error in the tracking. The move is chosen using the Nash equilibrium, which presupposes that the other player made the worst move.. In addition to being resistant to input interruptions, this method may also be resilient to modeling system uncertainty. This method evaluated the performance through simulation in the Simulink environment and implementation on a three-degree-of-freedom stand.

Keywords: Quadcopter, Differential Game, Game Theory, Nash Equilibrium, Three Degree of Freedom Stand, Model Base Design, Linear Quadratic Regulator



Sharif University of Technology
Department of Aerospace Engineering

Master Thesis

Robust Reinforcement Learning Differential Game Guidance in Low-Thrust, Multi-Body Dynamical Environments

By:

Ali BaniAsad

Supervisor:

Dr.Hadi Nobahari

December 2024