



دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد
شبکه‌های کامپیوتری

عنوان:

پیشنهاد الگوریتمی برای تخصیص منابع در محاسبات لپه
با بهره‌گیری از یادگیری تقویتی عمیق

نگارش:

ایمان رحمتی

استاد راهنما:

دکتر علی موقر

شهریور ۱۴۰۱

اللهُ أَكْبَرُ

تشکر و قدردانی

شکر شایان نثار پروردگار که منجر به پایان یافتن این فصل از زندگی من شد. در ابتدا از همکاری‌های استاد اندیشمند جناب آقای دکتر موقر سپاس‌گزاری می‌نمایم که بدون زحمات و راهنمایی‌های ایشان انجام این پروژه میسر نبود، سپس از اعضاء خانواده‌ام به موجب حمایت‌ها و محبت‌های بی‌دریغشان در طی به ثمر رسیدن این پایان‌نامه بی‌اندازه قدردانم.

چکیده

توسعه دستگاه‌های هوشمند متحرک با بهبود ارتباطات و قابلیت‌های ادراکی موجب تکثیر بسیاری از برنامه‌های کاربردی پیچیده و محاسباتی شده است. دستگاه‌های هوشمند با منابع محدود بیش از هر زمان دیگری با محدودیت‌های شدید ظرفیت روبرو هستند. به نظر می‌رسد به عنوان یک مفهوم جدید از معماری شبکه و توسعه محاسبات ابری، محاسبات لبه یک راه حل امیدوارکننده برای پاسخ‌گویی به این چالش نوظهور است. محاسبات لبه با نزدیک کردن منابع محاسباتی به کاربران نهایی و توزیع محاسبات در سیستم، دستگاه‌ها را از بار سنگین محاسباتی تخلیه می‌کند و موجب بهبود عملکرد سیستم، کیفیت خدمات و کیفیت تجربه کاربران می‌شود. با توجه به پویایی سطح بار محاسباتی نامشخص در گره‌های لبه، تخصیص منابع محاسباتی و تصمیم‌گیری در مورد محل انجام محاسبات، در راستای مرتفع کردن نیازهای برنامه‌های کاربردی، امری چالش‌برانگیز خواهد بود. در این پژوهش، وظایف حساس به تأخیر و غیرقابل تقسیم را به همراه پویایی سطح بار محاسباتی سیستم در نظر می‌گیریم، و یک مسئله بارسپاری وظیفه را برای به حداقل رساندن هزینه طولانی مدت مورد انتظار فرموله می‌کنیم. در جهت بهبود کارایی سیستم، به بهینه‌سازی مشترک تأخیر و مصرف انرژی می‌پردازیم و یک الگوریتم توزیع شده مبنی بر یادگیری تقویتی عمیق بدون مدل را پیشنهاد می‌کنیم، که در آن هر دستگاه می‌تواند تصمیم بارسپاری محاسبات خود را بدون دانستن مدل‌های وظیفه در دستگاه‌های دیگر و سطح بار محاسباتی در گره‌های لبه، تعیین کند. برای بهبود برآورد هزینه بلندمدت در این الگوریتم، از حافظه کوتاه‌مدت ماندگار، شبکه Q عمیق دوئل، و تکنیک‌های شبکه Q عمیق دوگانه بهره می‌گیریم. نتایج ارزیابی نشان می‌دهد که روش پیشنهادی می‌تواند به نحوی کارآمد از ظرفیت‌های محاسباتی گره‌های لبه بهره‌برداری کند و در جهت بهبود کارایی سیستم موثر واقع شود. این روش می‌تواند علاوه بر کاهش چشمگیر میانگین تاخیر و مصرف انرژی در سیستم، به افزایش تعداد وظایف انجام‌شده در مقایسه با چندین روش موجود دست یابد.

کلیدواژه‌ها: محاسبات لبه، بارسپاری محاسباتی، تخصیص منابع، مصرف انرژی، یادگیری تقویتی عمیق

فهرست مطالب

۱	۱	مقدمه
۱	۱-۱	پیش‌زمینه
۴	۲-۱	اهداف پژوهش
۵	۳-۱	چالش‌ها
۶	۴-۱	دست‌آوردها
۶	۵-۱	ساختار پایان‌نامه
۷	۲	مفاهیم اولیه
۷	۱-۲	معماری شبکه
۸	۱-۱-۲	محاسبات ابری
۹	۲-۱-۲	محاسبات لبه متخرک
۱۱	۳-۱-۲	ابعاد محاسبات ابری و محاسبات لبه
۱۳	۲-۲	بارسپاری محاسباتی
۱۴	۱-۲-۲	تصمیم‌گیری در مورد بارسپاری محاسبات به گره لبه
۱۵	۳-۲	مدیریت منابع

۱۶	۱-۳-۲ تخمین منابع
۱۶	۲-۳-۲ کشف منابع
۱۷	۳-۳-۲ تخصیص منابع
۱۸	۴-۳-۲ اشتراک‌گذاری منابع
۱۸	۵-۳-۲ بهینه‌سازی منابع
۱۸	۴-۲ فرایند تصمیم‌گیری مارکوف
۲۰	۵-۲ تعریف فرایند تصمیم‌گیری مارکوف
۲۱	۶-۲ یادگیری تقویتی
۲۱	۷-۲ اجزا مسئله یادگیری تقویتی
۲۲	۱-۷-۲ سیگنال امتیاز
۲۲	۲-۷-۲ عامل و محیط
۲۴	۳-۷-۲ حالات و تاریخچه
۲۵	۴-۷-۲ توابع ارزش و معادله بلمن
۲۷	۸-۲ بهره‌برداری و اکتشاف
۲۸	۹-۲ روش‌های پاسخ به مسائل یادگیری تقویتی
۲۹	۱-۹-۲ روش تکرارشونده ارزش
۲۹	۲-۹-۲ روش تکرارشونده سیاست
۳۰	۱۰-۲ تقریب توابع
۳۱	۱۱-۲ شبکه‌های عصبی عمیق
۳۳	۱۲-۲ مدل محیط در روش‌های مبتنی بر یادگیری تقویتی
۳۴	۱۳-۲ جمع‌بندی و نتیجه‌گیری

۳۶	۳ پژوهش‌های مرتبط پیشین
۳۶	۱-۳ مقدمه
۳۷	۲-۳ پژوهش‌های پیشین
۴۴	۴ تعریف مسئله
۴۴	۱-۴ مدل سیستم
۴۶	۲-۴ مدل وظیفه
۴۷	۳-۴ تصمیم‌گیری بارسپاری وظیفه
۴۷	۴-۴ مدل ارتباطی
۴۹	۵-۴ مدل محاسباتی
۴۹	۱-۵-۴ محاسبات محلی
۵۰	۲-۵-۴ محاسبات در لبه
۵۳	۳-۵-۴ پردازش و انقضا وظیفه
۵۴	۵ مسئله بارسپاری وظیفه مبتنی بر یادگیری تقویتی عمیق
۵۴	۱-۵ بیان مسئله با یادگیری تقویتی عمیق
۵۵	۱-۱-۵ حالت
۵۶	۲-۱-۵ عمل
۵۶	۳-۱-۵ هزینه
۵۷	۲-۵ سیاست بهینه
۵۸	۳-۵ الگوریتم بارسپاری وظیفه مبتنی بر یادگیری تقویتی عمیق
۵۹	۴-۵ شبکه عصبی

۵-۵	لایه ورودی	۶۰
۶-۵	لایه حافظه کوتاه‌مدت ماندگار	۶۰
۷-۵	لایه‌های کاملاً متصل	۶۱
۸-۵	لایه ارزش، امتیاز-عمل و لایه خروجی	۶۱
۹-۵	الگوریتم مبتنی بر یادگیری تقویتی عمیق	۶۳
۱-۹-۵	الگوریتم اجرایی در دستگاه تلفن همراه	۶۴
۲-۹-۵	الگوریتم اجرایی در گره لبه	۶۶
۶	نتایج	۶۹
۱-۶	شبیه‌سازی	۶۹
۲-۶	ارزیابی عملکرد	۷۱
۳-۶	مقایسه نتایج	۷۳
۷	نتیجه‌گیری	۸۰

فهرست شکل‌ها

۱-۲	معماری شبکه با بهره‌گیری از محاسبات لبه	۸
۲-۲	نحوه تعامل عامل هوشمند یادگیری تقویتی در محیط	۲۳
۳-۲	نمونه‌ای از یک شبکه عصبی	۳۲
۴-۴	تصویر صفحه‌ای تشکیل شده در دستگاه تلفن همراه $m \in M$ و در گره لبه $n \in N$	۴۶
۱-۵	شبکه عصبی دستگاه تلفن همراه $M \in m$, همراه با بردار پارامترهای θ_m , که نگاشت بین حالت $s_m(t) \in S$ و مقادیر Q را برای هر عمل $a \in A$ انجام می‌دهد.	۵۹
۲-۵	شبکه حافظه کوتاه‌مدت ماندگار با T^{step} واحد حافظه.	۶۱
۳-۵	معماری شبکه تک‌جریانی Q معمولی (بالا) و معماری شبکه دوئل Q مورد استفاده (پایین)	۶۲
۱-۶	همگرایی الگوریتم پیشنهادی تحت تغییرات نرخ یادگیری.	۷۴
۲-۶	همگرایی الگوریتم پیشنهادی تحت تغییرات تناوب درخواست پارامتر به روز رسانی.	۷۵
۳-۶	مقایسه عملکرد در نرخ وظایف منقضی شده تحت: (a) تغییرات تعداد دستگاه‌های متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی.	۷۶
۴-۶	مقایسه عملکرد در میانگین تأخیر تحت: (a) تغییرات تعداد دستگاه‌های متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی.	۷۷

۵-۶ مقایسه عملکرد در مقدار انرژی مصرفی سیستم تحت: (a) تغییرات تعداد دستگاه‌های متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی
۷۸

فهرست جداول

۱-۳ جمع‌بندی پژوهش‌های مرتبط	۴۲
۱-۶ تنظیمات پارامترهای اصلی	۷۰

فصل ۱

مقدمه

در این فصل ابتدا پیش‌زمینه‌ای در مورد موضوع پژوهش بیان می‌شود، سپس به بیان انگیزه پژوهش، اهداف و چالش‌های موجود در این حوزه می‌پردازیم.

۱-۱ پیش‌زمینه

در سال‌های اخیر، با توسعه دستگاه‌های هوشمند سیار، از جمله تلفن‌های همراه هوشمند، وسایل نقلیه هوشمند، دستگاه‌های ناظرت بر سلامت و سایر موارد، شاهد طیف گسترده‌ای از تعاملات فناوری‌های هوشمند در زندگی روزمره انسانی هستیم. قابلیت‌های ارتباطی و ادراکی این دستگاه‌ها در حال پیشرفت است، در نتیجه شاهد گسترش سریع تعداد زیادی از برنامه‌های کاربردی پیچیده و محاسباتی مانند واقعیت مجازی^۱، مدل‌سازی سه‌بعدی، پردازش زبان طبیعی^۲ و بازی‌های تعاملی هستیم، که در پی آن با تنوع نیازهای برنامه‌های کاربردی از جمله نیاز به محاسبات پیچیده و نیاز به پاسخ بی‌درنگ^۳، روبرو می‌شویم.

با توجه به توان باتری و قدرت محاسباتی محدود دستگاه‌های هوشمند، انجام این محاسبات توسط

Virtual Reality^۱
Natural Language Processing^۲
Real-time^۳

خود دستگاه بهنهایی ممکن نخواهدبود. بنابراین دستگاههای متصل به شبکه، راهی بهجز اتصال به سرورهای ابری برای بارسپاری محاسبات سنگین و استفاده بهینه از منابع محدود خود ندارند. با این حال به کارگیری منابع محاسباتی ابری معمول، به علت فاصله مکانی زیاد آنها با کاربران منجر به تأخیر انتقال زیادی می‌شود، که در بسیاری از کاربردهای کنونی از جمله کاربردهایی که به تضمین بی‌درنگ بودن نیاز دارند، مشکل ایجاد می‌کند.

دستگاههای هوشمند با منابع محدود بیش از هر زمان دیگری با محدودیت‌های شدیدی در ظرفیت رو به رو هستند، و این درحالی است که اجرای تمام برنامه‌ها در سرورهای محاسباتی ابری علاوه بر تأخیر انتقال، به راحتی منجر به ازدحام شبکه و تخریب عملکرد جدی می‌شود. در این زمان، به نظر می‌رسد که به عنوان یک مفهوم جدید از معماری شبکه و توسعه محاسبات ابری^۴، محاسبات لبه^۵ یک راه حل امیدوارکننده برای پاسخگویی به این چالش نوظهور باشد. محاسبات لبه با نزدیک کردن منابع محاسباتی و ذخیره‌سازی به کاربران انتهایی نه تنها می‌تواند از ازدحام شبکه اصلی بکاهد بلکه با برآورده ساختن نیازهای سختگیرانه تأخیر در عملکرد سیستم و کیفیت تجربه مشتری نیز تا حد زیادی بهبود می‌بخشد.

جایگیری سرورهای محاسباتی در لبه شبکه و استفاده از محاسبات لبه سیار^۶، ظرفیت‌های محاسباتی جدیدی را برای کاربران در لبه شبکه بی‌سیم فراهم می‌کند. که علاوه بر ایجاد منابع محاسباتی برای کاربران، تأخیر انتقال را نیز به شدت کاهش می‌دهد و باعث افزایش کیفیت خدمات دریافتی کاربر می‌شود. به این شکل قدرت محاسباتی دستگاههای موجود در لبه شبکه با بهره‌گیری از پیشرفت‌های اخیر محاسبات لبه سیار، به مقدار قابل توجهی افزایش می‌یابد [۱].

معماری محاسبات لبه دارای مزایای مهمی در رسیدگی به تکثیر دستگاههای هوشمند و برنامه‌های کاربردی است، با این حال محدودیت‌ها و چالش‌های زیادی را نیز در بر خواهدداشت. از جمله این محدودیت‌ها می‌توان به هزینه بالای استقرار و نگهداری زیرساخت و سرورهای قدرتمند ایستگاههای پایه‌ای در لبه شبکه اشاره کرد، که هزینه غیرقابل قبولی در پی خواهدداشت. و همچنین با اینکه ظرفیت‌های محاسباتی و ذخیره‌سازی سرورهای لبه به مرتب بیشتر از تجهیزات موجود در دستگاههای هوشمند می‌باشد، اما تعداد محدودی از برنامه‌ها را می‌توان در یک سرور لبه برای ارائه خدمات به دستگاههای

Cloud Computing^۴Edge computing^۵Mobile Edge computing^۶

هوشمند متعدد مستقرکرد.

به عنوان مهم‌ترین چالش در محاسبات لبه می‌توان به تنوع فعالیت‌های انسانی در زندگی روزمره اشاره کرد، که باعث ایجاد نیازهای متنوع دستگاه‌های هوشمند و تغییر وضعیت مناطق درخواست شده و موجب بی‌ثباتی سیستم می‌شود. همچنین حجم بالای درخواست‌های پشت سر هم، می‌تواند باعث افزایش ناگهانی بار محاسبات در سرورهای لبه و همچنین عدم تعادل بار جدی در بین آن‌ها شود، که باعث تراکم درخواست‌ها در مناطق خاص و بالا رفتن نرخ تأخیر در پاسخ‌گویی خواهدشد.

با توجه به محدودیت‌ها و چالش‌های موجود در محاسبات لبه، چگونگی تخصیص منابع محاسباتی به شکلی منعطف و مقیاس‌پذیر به یک چالش بزرگ در این حوزه تبدیل شده است. بیشتر پژوهش‌های انجام‌گرفته بر تخلیه وظیفه^۴ محاسباتی، بهینه‌سازی در برنامه‌ریزی^۵ محاسبات و تخصیص منابع^۶ محاسباتی لبه تمرکز دارند، و مسئله بارسپاری محاسباتی را علت وجود متغیرهای دو حالتی تصمیم‌گیری به صورت برنامه‌ریزی عدد صحیح آمیخته^۷ بیان می‌کنند، و برای حل آن الگوریتم‌هایی همچون برنامه‌ریزی پویا^۸ ارائه شده است که برای شبکه‌های بزرگ با تعداد بالای سرورها، پیچیدگی زیادی در بر دارد [۲].

اگرچه این پژوهش‌ها زمینه را برای تحقیقات بر روی محاسبات لبه، در جنبه‌های مختلف فراهم می‌کند، اما این مطالعات به سازگاری روش‌های تخصیص منابع در محاسبات لبه در مواجهه با بار سنگین و عدم ثبات شبکه نپرداخته‌اند. هنگامی که منطق تولید درخواست و همچنین نیازهای دستگاه‌های هوشمند به طرز چشمگیری افزایش یابد، بدون در نظر گرفتن شرایط مختلف، تأثیر قابل توجهی در عملکرد سیستم از جمله توزیع بار محاسباتی خواهد داشت.

Task Offloading^۷
Scheduling^۸
Resource allocation^۹
Mixed Integer Linear Programming^{۱۰}
Dynamic programming^{۱۱}

۱-۲ اهداف پژوهش

در این پژوهش سعی داریم با هدف کاهش تأخیر متوسط محاسبات و بهینه‌سازی مصرف انرژی، به یک سیاست بهینه تخصیص منابع، در یک شبکه محاسباتی لبه دست پیدا کنیم. ما پویایی سطح بار ناشناخته را در گره‌های لبه در نظر می‌گیریم و یک الگوریتم بارسپاری توزیع شده برای سیستم محاسبات لبه متحرک، با بهره‌گیری از یادگیری تقویتی عمیق^{۱۲} ارائه می‌دهیم. در الگوریتم پیشنهادی، هر دستگاه هوشمند متحرک می‌تواند تصمیم‌گیری تخلیه وظایف را به صورت غیرمتمرکز با استفاده از اطلاعاتی که به صورت محلی مشاهده می‌شود، از جمله اندازه وظیفه، اطلاعات صفات و سطوح بار تاریخی در گره‌های لبه، اخذ کند.

ما عمل تخلیه وظیفه را برای وظایف غیر قابل تقسیم و حساس به تأخیر^{۱۳} فرموله می‌کنیم، و در جهت دستیابی به حداقل هزینه‌های موردنانتظار طولانی‌مدت، تابع هزینه را مطابق با تأخیر محاسبات و مصرف انرژی در نظر می‌گیریم. برای دستیابی به پویایی سطح بار محاسباتی سیستم از یک حافظه کوتاه‌مدت ماندگار^{۱۴} استفاده می‌کنیم، که با یادگیری اطلاعات تاریخی می‌تواند سطوح بار را در آینده نزدیک پیش‌بینی کند و این سطح بار پیش‌بینی شده، قسمتی از مشاهدات محیطی را در شبکه یادگیری تقویتی تشکیل می‌دهد.

شایان ذکر است که منابع مورد نیاز وظایف محاسباتی در هر شکاف زمانی متفاوت و پویا است، یک دستگاه هوشمند خاص ممکن است به دلیل منابع محاسباتی بیش از حد مورد نیاز تحت محدودیت حد اکثر تأخیر مجاز، قادر به اجرای کار وارد شده به صورت محلی نباشد. بنابراین، تفاوت کلیدی بین روش پیشنهادی و روش‌های پایه در این است که روش پیشنهادی می‌تواند تصمیم‌گیری‌های بارسپاری را صورت پویا انجام دهد.

^{۱۲} Deep Reinforcement Learning

^{۱۳} Delay-sensitive

^{۱۴} Long Short-term Memory

۱-۳ چالش‌ها

یکی از چالش‌های موجود در مسئله بارسپاری محاسبات، تعیین نحوه انجام این فرایند در لبه شبکه است. یک استراتژی خوب می‌تواند منجر به کاهش انرژی مصرفی در شبکه و همچنین کاهش زمان پاسخ سیستم شود. دو چالش اصلی در تخلیه وظایف محاسباتی وجود دارد، اولین چالش این است که یک دستگاه هوشمند متحرک، در چه زمانی و در مواجه با چه نوع وظایفی باید محاسبات خود را به گره‌های لبه تخلیه کند، و چالش دوم این است که، اگر تصمیم بر تخلیه محاسبات باشد، این امر به چه شکل و به کدام گره لبه باید انجام شود. برای رسیدگی به این چالش‌ها، خیل زیادی از پژوهش‌های جدید به توسعه الگوریتم‌های تخلیه محاسباتی پرداخته‌اند.

زمان پردازش یک بار محاسباتی در سرورهای مختلف با توجه به قدرت پردازش آنها متفاوت است. همچنین سرعت انتقال داده بین کاربران و سرورها و بین سرورهای مختلف با توجه به کیفیت متفاوت راههای ارتباطی بین آنها تعیین می‌شود. تمام این متغیرها در تأخیر انجام محاسبات تأثیر بسزایی دارد و می‌تواند سیاست تخصیص منابع و فرآیند کاری را کاملاً تغییر دهد. این موضوع زمانی که چندین کاربر با محاسبات مختلف داریم چالش بیشتری ایجاد می‌کند. به طور مثال کاربران به طور طبیعی سروری با قدرت محاسباتی بالاتر را انتخاب می‌کند اما در صورتی که کاربران زیادی به این سرور بارسپاری کنند زمان پاسخ آن افزایش یافته و دیگر سرور مناسبی برای بارسپاری محاسباتی محسوب نمی‌شود.

بنابراین نحوه تخصیص منابع به کاربران مختلف و پخش بار محاسباتی در شبکه به شکلی که متوسط زمان انجام محاسبات کاهش یابد از اهمیت ویژه‌ای برخوردار است.

۱-۴ دستآوردها

در این پژوهش به بررسی چالش بارسپاری وظایف در یک سیستم محاسباتی لبه پرداخته و با هدف به حداقل رساندن میانگین تأخیر و مصرف انرژی در طولانی مدت یک الگوریتم بارسپاری توزیع شده ارائه می‌دهیم، که دستگاه‌های تلفن همراه را قادر می‌سازد تا تصمیم‌گیری بارسپاری وظایف خود را به شیوه‌ای غیرمت مرکز انجام دهد، در حالی که می‌توانند پویایی سطح بار ناشناخته را در گره‌های لبه در نظر گرفته و به بهینه‌سازی کارایی سیستم در مواجهه با وظایف غیرقابل تقسیم و حساس به تأخیر پردازد. و این در حالی است که محدودیت تأخیر و ظرفیت محدود منابع نیز در نظر گرفته شده است.

نتایج شبیه سازی نشان می‌دهد که در مقایسه با چندین روش موجود، الگوریتم پیشنهادی می‌تواند نسبت کارهای منقضی شده، میانگین تأخیر و میزان مصرف انرژی را کاهش دهد. این مزیت به ویژه زمانی قابل توجه است که وظایف حساس به تأخیر باشند، یا سطوح بار در گره‌های لبه بالا باشد.

۱-۵ ساختار پایان‌نامه

ساختار ادامه پایان‌نامه در این قسمت بیان می‌شود. در فصل ۲، به بیان ادبیات موضوع و مفاهیم اساسی در حوزه پژوهش می‌پردازیم و در فصل ۳، پژوهش‌های مرتبط پیشین را بررسی و طبقه‌بندی می‌کنیم.

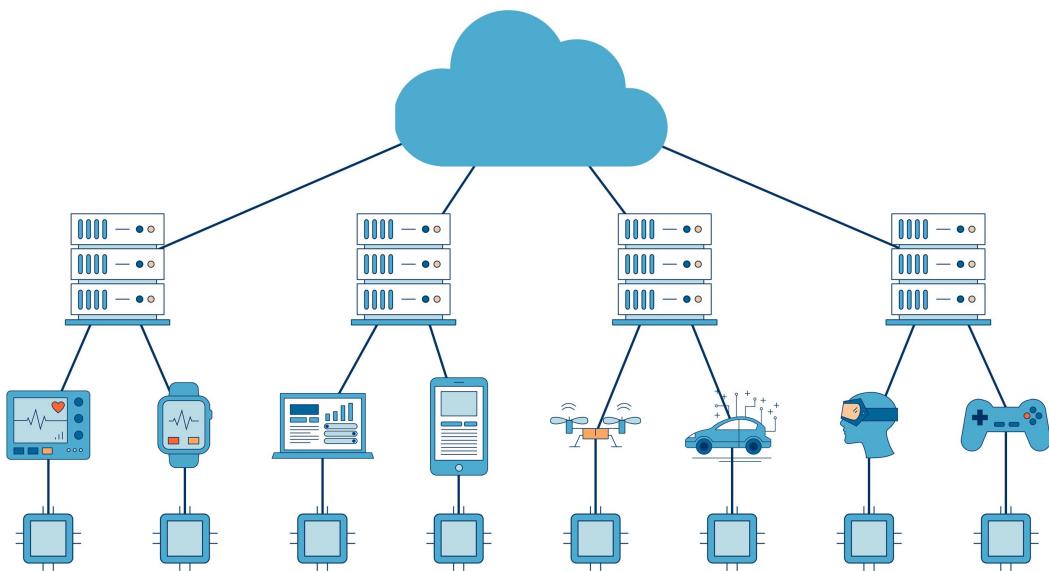
در فصل ۴، ابتدا به تعریف مسئله در محاسبات لبه پرداخته و ساختار مدل سیستم را تعریف می‌کنیم. سپس به معرفی مدل دستگاه‌های تلفن همراه و مدل گره‌های لبه می‌پردازیم. مدل دستگاه‌های تلفن همراه را تحت قالب مدل وظیفه، صفات محاسبات و صفات انتقال مورد بررسی قرار می‌دهیم و مدل گره‌های لبه را در قالب صفات محاسبات دستگاه در گره لبه بیان می‌کنیم. در فصل ۵، به بررسی چالش بارسپاری وظیفه در محاسبات لبه، بر مبنای یادگیری تقویتی عمیق می‌پردازیم. مسئله را در چهارچوب مولفه‌های یادگیری تقویتی بیان کرده و شبکه‌های عصبی استفاده شده در الگوریتم پیشنهادی را تشریح می‌کنیم. الگوریتم پیشنهادی را معرفی کرده و نحوه استقرار آن در دستگاه‌های تلفن همراه و گره‌های لبه را بیان می‌کنیم. سپس در فصل ۶ به تشریح شبیه‌سازی، ارزیابی عملکرد و مقایسه نتایج حاصل از آن می‌پردازیم. و در فصل ۷، به نتیجه‌گیری پرداخته و پژوهش‌های آینده را معرفی می‌کنیم.

فصل ۲

مفاهیم اولیه

۱-۲ معماری شبکه

همانطور که اشاره شد رشد فزاینده سیستم‌های هوشمند و نیازهای محاسباتی برنامه‌های کاربردی دیگر فقط از طریق منابع محاسباتی تجهیزات کاربر امکان‌پذیر نخواهد بود، بنابرین محاسبات ابری نقش مهمی را در تامین منابع محاسباتی ایفا می‌کند. همچنین وجود نیازهای متعدد سیستم‌های هوشمند از جمله کاربردهای بی‌درنگ باعث غیرقابل تحمل بودن تأخیر شده است. از این رو انتقال محاسبات به ابر مرکزی در برخی موارد امکان‌پذیر نخواهد بود، محاسبات لبه با جایگیری در بخش میانی دستگاه‌های هوشمند و ابر مرکزی با نزدیک کردن منابع محاسباتی نقشی همانند محاسبات ابری اما به صورت توزیع شده و در فاصله نزدیک‌تر به دستگاه‌های هوشمند را ایفا می‌کنند. بنابراین می‌توان معماری سیستم را به شکل کلی شامل چند لایه متشكل از محاسبات ابری، محاسبات لبه و دستگاه‌های هوشمند همانطور که در شکل ۱-۲ نشان داده شده است، متصور بود.



شکل ۱-۲: معماری شبکه با بهره‌گیری از محاسبات لبه

۱-۱-۲ محاسبات ابری

امروزه محاسبات ابری با فراهم کردن منابع محاسباتی قدرتمند به صورت متمرکز و در دسترس از راه دور نقش مهمی در تامین نیازهای محاسباتی سیستم‌های هوشمند را برعهده دارد. هدف اصلی محاسبات ابری میسر ساختن دسترسی به حجم عظیمی از منابع محاسباتی به صورت مجازی‌سازی شده است. این کار با استفاده از تجمعی منابع و ایجاد یک سیستم یکپارچه انجام می‌شود. خدمات ابری کاربران را قادر می‌سازند تا برنامه‌ها را در سرورهای راه دور ذخیره و اجرا کنند و سپس از طریق اینترنت به کلیه داده‌ها دسترسی پیدا کنند. این بدان معناست که کاربر برای دسترسی به اطلاعات محدودیت مکانی نخواهد داشت و از هر جایی از طریق اینترنت دسترسی خواهد داشت [۳]. سرورهای ابری از توان محاسباتی بالایی برخوردار هستند و با تجمعی اطلاعات دریافتی از نقاط محاسباتی دیگر از جمله دستگاه‌های هوشمند، سنسورها و سرورهای لبه می‌توانند سیستمی متمرکز و یکپارچه ایجاد کنند و این سیستم قادر خواهد بود در ارائه خدمات متنوع به دستگاه‌های هوشمند و بهبود عملکرد سیستم موثر واقع شود.

۲-۱-۲ محاسبات لبه متحرک

در پی رشد دستگاه های هوشمند و برنامه های کاربردی، بهره گیری از محاسبات ابری توانسته تا حدی به توزیع بار محاسبات کمک کند و باعث بهبود عملکرد سیستم شود، اما با توجه به گسترش هر روزه این نیازهای محاسباتی، اجرای تمام برنامه ها در ابر مرکزی به راحتی منجر به ازدحام شبکه و تخریب عملکرد جدی می شود. همچنین وجود نیازهای متنوع برنامه ها از جمله کاربردهای بی درنگ، نمی توانند تأخیر ارسال^۱ محاسبات به ابر را تحمل کنند، در خیلی از این موقع این تأخیر می تواند سبب تخریب سیستم و حتی صدمات جدی باشد. به عنوان مثال تأخیر در واکنش یک وسیله نقلیه خودران^۲ و یا دستگاه های مورد استفاده در پزشکی، می تواند آسیب های جدی و غیرقابل جبرانی را در پی داشته باشد. در این زمان، به نظر می رسد که به عنوان یک مفهوم جدید از معماری شبکه و توسعه محاسبات ابری، محاسبات لبه یک راه حل امیدوارکننده برای پاسخگویی به این چالش باشد.

محاسبات لبه با نزدیک کردن منابع محاسباتی و ذخیره سازی به کاربران انتهایی باعث کاهش تأخیر و بهبود عملکرد سیستم می شود و کیفیت تجربه مشتری را تا حد زیادی بهبود می بخشد. سرورهای لبه که بین دستگاه های هوشمند و مرکز داده ابری مستقر می شوند، خدمات محاسباتی، ذخیره سازی در شبکه را فراهم می کنند. محاسبات لبه همانند محاسبات ابری با فراهم کردن منابع محاسباتی در دسترس برای رفع نیازهای محاسباتی موجود در نظر گرفته می شود.

تفاوت محاسبات لبه با محاسبات ابری را می توان در نحوه جایگیری منابع تعریف کرد. ساختار محاسبات ابری به شکلی متتمرکز^۳ و در نقاط مکانی محدود تشکیل شده است اما محاسبات لبه سرورها را در نقاط مختلف و در نزدیکی دستگاه های هوشمند به شکلی توزیع شده^۴ استقرار می دهد. همچنین سرورهای لبه در مقایسه با ابر مرکزی از توان محاسباتی کمتری برخوردار هستند. سرورهای نزدیک به کاربر می توانند از دستگاه های هوشمند همراه برای ارتباطات بی وقهه پشتیبانی کنند. از ویژگی های متمایز کننده محاسبات لبه متحرک می توان به موارد زیر اشاره کرد:

Transmission Delay^۱
Autonomous Vehicles^۲
Centralize^۳
Distributed^۴

پشتیبانی از تحرک کاربران همانطور که تعداد دستگاه‌های هوشمند متحرک به سرعت در حال افزایش است، محاسبات لبه متحرک با استفاده از روش‌های غیر مت مرکز، از تحرک کاربران پشتیبانی می‌کند، بهره‌گیری از روش‌های توزیع شده، برای برقراری ارتباط مستقیم با دستگاه‌های متحرک، با جداسازی هویت میزبان از هویت محل، اصل کلیدی را در پشتیبانی تحرک در محاسبات لبه متحرک ایفا می‌کند.

توزيع جغرافیایی متراکم محاسبات لبه خدمات ابری را به کاربر نزدیک‌تر می‌کند و با استفاده از سیستم عامل‌های متعدد می‌تواند به توزیع جغرافیایی متراکم زیرساخت‌ها کمک‌کند، مدیران شبکه قادر خواهند بود خدمات متحرک مبتنی بر مکان را تسهیل کنند. همچنین تجزیه و تحلیل داده‌های بزرگ را می‌توان به سرعت با دقت بهتری انجام داد، سیستم‌های لبه، تجزیه و تحلیل بی‌درنگ را در مقیاس بزرگ ممکن می‌کنند.

آگاهی از موقعیت مکانی آگاهی از موقعیت مکانی کاربران در محاسبات لبه به کاربران متحرک این امکان را می‌دهد تا خدمات را از سرورهای لبه در نزدیک‌ترین مکان به موقعیت فیزیکی خود در دسترس داشته باشند. کاربران می‌توانند فناوری‌های مختلفی مانند زیرساخت‌های موقعیت‌یاب در تلفن همراه، و یا نقاط دسترسی بی‌سیم را برای اشتراک‌گذاری موقعیت مکانی خود به کار گیرند. این آگاهی از موقعیت مکانی می‌تواند در توزیع مناسب بار محاسباتی در شبکه و کاهش زمان انتقال محاسبات بسیار تاثیرگذار و مفید باشد.

نzdیکی به کاربران در محاسبات لبه متحرک، منابع محاسباتی و خدمات در نزدیکی کاربران در دسترس است و از این طریق کاربران می‌توانند تجربه خود را بهبود بخشنند. در دسترس بودن منابع محاسباتی و خدمات در مجاورت محلی، کاربران را قادر می‌سازد اطلاعات مربوط به شبکه را برای تصمیم‌گیری تخلیه^۵ و استفاده از خدمات، مورد استفاده قرار دهند. به طور مشابه، ارائه دهنده خدمات می‌تواند اطلاعات کاربران را استخراج کرده و با تجزیه و تحلیل رفتار کاربران در جهت بهبود خدمات و تخصیص منابع خود، استفاده کند.

^۵ Offloading Decision

زمان تأخیر کم پارادایم‌های محاسباتی لبه، منابع محاسباتی و خدمات را به کاربران نزدیک‌تر می‌کنند که به موجب آن تأخیر در دسترسی به خدمات را کاهش می‌دهد. تأخیر کم محاسبات لبه، کاربران را قادر می‌سازد تا محاسبات برنامه‌های کاربردی حساس به تأخیر خود را بر روی سرورهای لبه با منابع محاسباتی قدرتمند انجام‌دهند و تأخیر را کاهش دهند.

ناهمگونی ناهمگونی در محاسبات لبه به وجود سیستم‌عامل‌ها، ساختار، فرایнд، فناوری، محاسبات و فن‌آوری‌های ارتباطی متنوع استفاده شده توسط عناصر محاسباتی لبه و دستگاه‌های پایانی و شبکه‌ها اشاره دارد. وجود چنین تفاوت‌هایی منجر به مسائل مربوط به قابلیت‌های عملیاتی شده‌است و آن را یک چالش اصلی در استقرار موفقیت‌آمیز محاسبات لبه ارائه می‌دهد. ناهمگونی شبکه به تنوع فن‌آوری‌های ارتباطی اشاره دارد که بر تحویل خدمات لبه تاثیر می‌گذارد.

هدف اصلی محاسبه لبه پیش‌پردازش داده‌ها، کاهش تأخیر و ارائه نیازهای برنامه‌های کاربردی با تأخیر کم و پاسخ در زمان کوتاه است [۴]. در حال حاضر، معماری رایج برای محاسبات لبه در ساختار یک شبکه چند لایه‌ی تشکیل شده از ابر مرکزی، سرورهای لبه و دستگاه‌های هوشمند انتهایی، همانطور که در شکل ۱-۲ نشان داده شده است، در نظر گرفته می‌شود.

۳-۱-۲ ابعاد محاسبات ابری و محاسبات لبه

با توجه به ماهیت مشترک این مفاهیم در تامین نیازهای محاسباتی دستگاه‌های هوشمند و برنامه‌های کاربردی، در این بخش به تفکیک مشخصات و بررسی دامنه و ابعاد هر یک از این تعاریف می‌پردازیم. به طور کلی می‌توان تفاوت بین محاسبات ابری، محاسبات مه^۶ و محاسبات لبه را در محل محاسبات، توان محاسبات و هدف از محاسبات در نظر گرفت [۵]:

محل محاسبات تفاوت اصلی بین محاسبات ابری، محاسبات مه و محاسبات لبه، مکانی است که پردازش داده در آن انجام می‌شود. در محاسبات ابری، داده‌ها بر روی یک سرور ابری مرکزی پردازش می‌شوند، که معمولاً^۷ بسیار دور از منبع اطلاعات قرار دارند. در محاسبات لبه معمولاً محاسبات مستقیماً

Fog Computing^۹

روی دستگاه‌هایی که سنسورها به آنها متصل شده‌اند یا سرورهای دروازه^v شبکه که در مجاورت سنسورها قرار دارد، رخ می‌دهد، و از طرف دیگر، محاسبات مه وظایف محاسباتی لبه را به منابع محاسباتی که مستقیماً به سرورهای محلی متصل شده‌اند منتقل می‌کنند، که از نظر فیزیکی از عملگرها و سنسورها فاصله بیشتری دارند.

بنابراین، برای محاسبات لبه، داده‌ها بر روی خود دستگاه‌ها و یا دستگاه نزدیک، پردازش می‌شوند بدون اینکه به جای دیگری منتقل شوند. و در محاسبات مه، داده‌ها درون یک دروازه که معمولاً در شبکه محلی قرار دارد پردازش می‌شوند. اگرچه از نظر فاصله مکانی این دسته‌بندی معتبر خواهد بود اما این مفاهیم در هر معماری بخصوصی می‌تواند به شکل دیگری در نظر گرفته شود. به عنوان مثال یک سنسور که در درون بدن یک بیمار وظیفه ثبت اطلاعات را بر عهده دارد گوشی همراه در نزدیکی بیمار برای سنسور، گره لبه محسوب می‌شود. بنابراین این تعاریف با توجه به موقعیت و معماری مورد استفاده شکل دقیق‌تری به خواهد خواهد گرفت.

توان محاسبات توان محاسباتی منابع مورد استفاده در معماری محاسبات لبه متحرک با دورشدن از دستگاه‌های انتهایی بیشتر شده و با نزدیکی به آن‌ها کاهش می‌یابد. در بالاترین سطح و دورترین فاصله، محاسبات ابری قرار دارد که بیشترین توان پردازشی را دارا می‌باشد. در لایه پایین‌تر محاسبات لبه که نسبت به محاسبات ابری از توان محاسباتی کمتری برخوردار هستند، قراردارند. به طور کلی می‌توان نتیجه گرفت با نزدیک شدن به دستگاه‌های هوشمند توان محاسباتی کاهش خواهد یافت، اما زمان پاسخ درخواست کمتر می‌شود و همچنین با دور شدن از نقاط انتهایی توان پردازشی افزایش یافته و زمان پاسخ نیز افزایش می‌یابد.

هدف محاسبات بعد دیگری از تفاوت‌های ساختاری محاسبات را می‌توان دلیل انجام آن محاسبات در نظر گرفت. به طور کلی محاسبات ابری برای تحلیل عمیق و طولانی مدت داده‌ها مناسب خواهد بود، و محاسبات لبه برای پردازش‌های کوتاه و تحلیل سریع و پاسخ در زمان کوتاه مناسب هستند. در اینجا ذکر این نکته نیز لازم است که محاسبات ابری به اتصال همیشگی اینترنت وابسته است، در حالی که سرورها در محاسبات لبه معمولاً حتی بدون اتصال به اینترنت نیز می‌توانند فعالیت کنند. همچنین به

لطف ساختار توزیع شده آن‌ها می‌توان محاسبات لبه را ایمن دانست. نسخه‌های مختلف از داده‌ها بین گره‌ها توزیع می‌شوند، و دستکاری داده‌ها در مقایسه با ساختار مرکز محاسبات ابری بسیار دشوار خواهدبود. بنابراین، در مواردی که امنیت یک نگرانی اساسی است، محاسبات لبه ترجیح داده‌می‌شوند.

از آنجاکه داده‌ها در میان گره‌ها توزیع می‌شوند، خرابی یک گره باعث خرابی سیستم نخواهدشد، در نتیجه تحمل‌پذیری اشکال^۴ در سیستم نسبت به محاسبات ابری بسیار بالا خواهدبود. از این رو در مواردی که توقف سیستم غیرقابل قبول باشد استفاده از محاسبات لبه مناسب خواهدبود. با توجه به مفاهیم ذکر شده و نقش آن‌ها در یک سیستم اینترنت اشیا، در راستای بهبود عملکرد سیستم وجود عماری چند لایه متشکل از محاسبات ابری، محاسبات لبه و دستگاه‌های هوشمند لازمه‌ی کارایی سیستم خواهدبود. این عماری به فراخور سیستم مورد نیاز در پژوهش‌های انجام شده به صورت‌های مختلفی در نظر گرفته شده است. به عنوان مثال در [۶] این عماری در سه سطح محاسبات ابری، محاسبات لبه و دستگاه‌های هوشمند ارائه شده است و در آن تعریف محاسبات لبه و محاسبات مه به صورت مشترک و تحت عنوان محاسبات لبه در نظر گرفته شده است. همچنین در [۷] این عماری به سه لایه محاسبات ابری، محاسبات مه و دستگاه‌های هوشمند خلاصه می‌شود.

۲-۲ بارسپاری محاسباتی

از منظر کاربر، یک مورد استفاده حیاتی از محاسبات لبه امر بارسپاری محاسباتی است، تخلیه بار محاسباتی می‌تواند زمینه را برای مواردی مانند کاهش تأخیر و بهینه‌سازی مصرف انرژی در تجهیزات کاربر و یا کلیت سیستم، فراهم‌آورد و در نتیجه موجب بهبود کیفیت خدمات^۹ و تجربه مشتری^{۱۰} گردد. به شکل کلی، واحد جدیدی در مورد بارسپاری محاسباتی، تصمیم می‌گیرد، که آیا بارسپاری انجام شود یا نشود، و اینکه چقدر و چه چیزی باید بارسپاری شود. اساساً، تصمیم‌گیری در مورد بارسپاری محاسبات ممکن است منجر به مواردی از جمله موارد زیر شود [۸]:

- محاسبات محلی: کل محاسبات به صورت محلی در تجهیزات کاربر انجام می‌شود. به عنوان

Fault Tolerance^۸
Quality of Service^۹
Quality of Experience^{۱۰}

مثال، وقتی که به دلیل عدم دسترسی به منابع محاسبات لبه و یا وجود هزینه بالا در بارسپاری محاسبات، محاسبات به شکل محلی انجام می‌شوند.

- **بارسپاری کامل:** کل محاسبات مورد نیاز به سرورهای لبه تخلیه می‌شوند. به عنوان مثال وقتی نیاز به محاسبات زیاد وجود دارد و تأخیر ارسال قابل تحمل است.

- **بارسپاری جزئی:** در این نوع از تخلیه محاسباتی، در صورتی که محاسبات قابل تقسیم باشند، بخشی از محاسبات به صورت محلی انجام می‌شود، در حالی که مابقی به سرورهای لبه تخلیه می‌شوند.

تخلیه محاسباتی در عمل یک فرآیند بسیار پیچیده و حساس است، چرا که می‌تواند تحت تأثیر عوامل مختلفی مانند ترجیحات کاربران، قابلیت اتصال، قابلیت های تجهیزات کاربران و دسترسی به ابر باشد. یک جنبه مهم در تخلیه محاسبات نیز، نوع برنامه کاربردی است، زیرا تعیین می‌کند که آیا تخلیه کامل و یا پراکنده قابل اجرا است، چه چیزی و به چه شکل می‌تواند تخلیه شود. نحوه استفاده و مدیریت فرآیند تخلیه در عمل بسیار دشوار است. اساساً تجهیزات کاربر باید یه یک واحد تصمیم‌گیری مجهز شود که مسئولیت آن، این است که با توجه به نوع برنامه، نوع داده و هدف از تخلیه، تعیین کند که چه محاسباتی می‌تواند بارسپاری شود، و چه محاسباتی می‌بایست توسط تجهیزات کاربر انجام شود.

۱-۲-۲ تصمیم‌گیری در مورد بارسپاری محاسبات به گره لبه

هدف اصلی در تصمیم‌گیری تخلیه محاسباتی، معمولاً به حداقل رساندن تأخیر و مصرف انرژی را شامل می‌شود. همچنین به حداقل رساندن مصرف انرژی می‌بایست در حالی که محدودیت‌های تأخیر از پیش تعریف شده است، برآورده شود. به شکل کلی اهداف اصلی از بارسپاری محاسبات را می‌توان طبق زیر طبقه‌بندی نمود.

کمینه‌سازی تأخیر اجرا یکی از مزایایی که با تخلیه محاسبات به سرورهای لبه معرفی می‌شود، امکان کاهش تأخیر در اجراست. در صورتی که تجهیزات کاربر تمام محاسبات را به تنها یی انجام دهد (یعنی

تخلیه بار انجام نشود)، تأخیر اجرا صرفاً نشان دهنده زمان صرف شده توسط اجرای محلی در تجهیزات کاربر است، اما در صورت تخلیه محاسباتی به سرورهای لبه، تأخیر اجرا شامل سه بخش زیر خواهد بود:

- مدت زمان انتقال داده‌های تخلیه شده به سرورهای لبه
- زمان انجام محاسبات در سرور لبه
- مدت زمان بازگشت و پذیرش داده‌های پردازش شده

کمینه‌سازی مصرف انرژی یکی دیگر از اهداف در بارسپاری محاسبات این است که در حالی که محدودیت‌های زمانی مورد قبول در برنامه‌های کاربردی ارضا می‌شوند، بتوان با بهره‌گیری از سیاست‌های مناسب در تخلیه محاسبات، مصرف انرژی در تجهیزات کاربر را به حداقل رساند. بنابراین با تخلیه محاسبات به سرورهای لبه می‌توان بدون انجام محاسبات محلی موجب کاهش مصرف انرژی در تجهیزات کاربر شد، اما از طرفی تخلیه و انتقال محاسبات به سرورهای لبه، موجب مصرف انرژی برای انتقال محاسبات به سرورهای لبه و همچنین دریافت نتایج در تجهیزات کاربر خواهد بود.

۳-۲ مدیریت منابع

ادغام محاسبات در تجهیزات کاربر و منابع محاسباتی لبه، درک کامل مدیریت منابع^{۱۱} را به امری ضروری بدل می‌کند. تأخیر محاسبات و کمبود منابع بهشدت تحت تاثیر تراکم شبکه قرار می‌گیرند، استفاده از توان بیشتر پردازشی در محاسبات لبه، به عنوان نزدیکترین منبع محاسبات و ذخیره‌سازی، می‌تواند یک راه حل امیدوارکننده برای کاهش زمان تأخیر و مصرف انرژی دستگاهها تلقی شود، و منابع غیر متتمرکز نقش مهمی در به اشتراک گذاری این منابع ایفا می‌کنند.

همانطور که اشاره شد، امید می‌رود محاسبات لبه، به عنوان مکمل محاسبات ابری، مشکلات منابع محدود دستگاه‌های هوشمند و مهلت^{۱۲} وظایف حساس به تأخیر را مرتفع نماید، و البته در پی آن چالش‌های زیادی در مسیر دست‌یافتن به این امر مطرح خواهد شد. منابع محاسباتی محدود و رشد

فزاینده دستگاه‌های هوشمند در محیطی متغیر و غیرقابل پیش‌بینی بر اهمیت بسیار زیاد مدیریت بهینه منابع به صورتی منعطف و مقیاس‌پذیر تأکید می‌ورزد. طبقه‌بندی مدیریت منابع، بر اساس تحقیقات فعلی در این زمینه، در [۹] ارائه شده است. طبق این طبقه‌بندی، در مجموع پنج دسته مختلف با هدف مدیریت منابع به شکل زیر در نظر گرفته شده است.

۱-۳-۲ تخمین منابع

یکی از اولین الزامات در مدیریت منابع، توانایی تخمین میزان منابع مورد نیاز برای انجام یک وظیفه است. تخمین منابع^{۱۳} امری مهم است، به ویژه برای توانایی کنترل نوسانات تقاضا در حالی که کیفیت خوبی از خدمات را برای کاربر حفظ می‌کند. منابع موجود در گره‌های لبه می‌توانند متحرک باشند و بنابراین ممکن است غیرقابل دسترسی شوند، که باعث می‌شود قابلیت اطمینان به آنها نسبت به منابع موجود در یک مرکز محاسباتی کمتر شود. از طرفی تحرک کاربر نیز به این عدم اطمینان می‌افزاید. از این رو درخواست‌های مربوطه باید توسط گره‌های لبه کنترل شود. به عنوان مثال نویسنده‌گان در [۱۰] از میانگین داده‌های تاریخی برای پیش‌بینی ویژگی‌های توزیع بار و استفاده از منابع برای دوره‌های بعدی استفاده می‌کنند.

۲-۳-۲ کشف منابع

برخلاف مسئله تخمین که مربوط به سمت تقاضای درخواست است، کشف منابع^{۱۴} مربوط به سمت عرضه منابع می‌شود. یک سیستم مدیریتی باید بداند چه منابعی برای استفاده در دسترس است، در کجا قرار دارد و چه مدت برای استفاده در دسترس خواهد بود (مخصوصاً اگر دستگاه تأمین‌کننده منبع در حال حرکت باشد). این دسته در محاسبات لبه از اهمیت بسیاری برخوردار است، زیرا هیچ منبعی همیشه تحت کنترل سیستم نیست، و بنابراین بی ثبات خواهد بود. در [۱۱] نویسنده‌گان الگوریتمی برای خوشبندی^{۱۵} دستگاه‌ها و کشف منابع در خوش ارائه می‌دهند. استراتژی آن‌ها این است که هر یک از

^{۱۳} Resource Estimation

^{۱۴} Resource Discovery

^{۱۵} Clustering

اعضای خوشه منابع خوشه خود را به سر خوشه اطلاع می‌دهد و کلیه درخواست‌ها توسط سر خوشه انجام می‌شود.

۳-۳-۲ تخصیص منابع

دسته سوم با هدف تخصیص برنامه‌ها در نزدیکی کاربران ظاهر می‌شود. تخصیص منابع^{۱۶}، از دانش منابع موجود برای نقشه‌برداری از بخش‌هایی از برنامه‌ها در دستگاه‌های مختلف استفاده می‌کند تا نیازهای آن‌ها برآورده شود. چند دیدگاه مختلف از تخصیص منابع وجود دارد اینکه در چه مرحله‌ای تخصیص انجام شود، در صورت نیاز چه زمانی و چه مقدار تخصیص یابد. از میان رویکردهای غالب تخصیص منابع، سه دیدگاه عمدۀ به شکل زیر تقسیم می‌شوند.

- **جایگذاری:** بیشتر پژوهش‌های انجام‌شده بر این دیدگاه تأکید دارند، به این شکل که وظیفه در کجا باید اجرا شود و چگونه منبعی را برای بهترین اجرای ممکن اختصاص داد.

- **مهاجرت:** با در نظر گرفتن اینکه تخصیص منابع باید کجا و به چه شکل انجام شود، صحبت از نهادهای مجازی مانند سرویس‌ها، برنامه‌ها و ماشین‌های مجازی به میان می‌آید، از این رو تمرکز می‌تواند بر این باشد که چگونه می‌توان آن‌ها را در حین اجرا و بر حسب نیاز برای بهبود عملکرد جابه‌جا کرد. به عنوان مثال در [۱۲] نویسنده‌گان با هدف کاهش تأخیر یک استراتژی مهاجرت بین منابع محاسباتی لبه در بارهای سنگین محاسباتی را ارائه می‌دهند.

- **برنامه‌ریزی:** در حالی که تحقیقات گسترده‌ای در مورد زمان و تعداد منابع برای تخصیص منابع در شبکه‌ها و مناطق خاص ابر وجود دارد، برنامه‌ریزی به اولویت‌دهی و زمان‌بندی فعالیت‌های محاسباتی در منابع می‌پردازد.

۴-۳-۲ اشتراک‌گذاری منابع

منابع موجود در دستگاه‌های انتهايی، ناهمگن و در اكثراً اوقات فاقد توان محاسباتی کافی است، منابع محاسباتی لبه نيز در مقایسه با منابع موجود در ابر محدوديت بيشتری دارند. اشتراک‌گذاری منابع^{۱۷} بين خود دستگاه‌ها يا بين دستگاه‌های انتهايی و گره‌های لبه، با هدف تأمین منابع محاسباتی مورد نياز از اهمیت زيادي برخوردار است. اشتراک‌گذاری منابع معمولاً با تجمیع منابع در مجاورت محلی گره‌ها محقق می‌شود. اين امر می‌تواند به دامنه محاسبات لبه گسترش يابد و يا در دستگاه‌های انتهايی باقی‌بماند.

۵-۳-۲ بهينه‌سازی منابع

هدف پنجم مورد بررسی در پژوهش‌های انجام‌شده، بهینه‌سازی منابع^{۱۸} است. اين امر معمولاً يك هدف مشترک و همراه با يكى يا بيشتر از اهداف توصيف شده قبلی است، كه به بررسی جنبه‌های مختلف در بهینه‌سازی می‌پردازد. بهینه‌سازی با ترکیب رویکردهای مدیریت منابع فوق الذکر به دست می‌آید. هدف اصلی بهینه‌سازی استفاده از منابع موجود در لبه با توجه به محدودیت‌های موجود در شبکه است.

۴-۲ فرایند تصمیم‌گیری مارکوف

در يك شهود سطح بالا، فرایند تصمیم‌گیری مارکوف^{۱۹} نوعی مدل ریاضی است برای بیان رفتار و تغییر وضعیت احتمالی محیط با توجه به تصمیمات پی‌درپی در حالات^[۲۰]، به شکلی که اگر به زنجیره گسته‌زمان مارکوف^{۲۱} عدم قطعیت افزوده شود، حاصل فرایند تصمیم‌گیری مارکوف نامیده می‌شود که علاوه بر خواص زنجیره گسته‌زمان مارکوف می‌توان به کمک این مدل هم رفتارهای احتمالی و هم رفتارهای دارای عدم قطعیت را مدل‌سازی نمود. در يك زمان مشخص مثل t ، عامل (تصمیم‌گیرنده)، حالت سیستم را مشاهده می‌کند. بر اساس این حالت و سیاست تصمیم‌گیری π ، عملی را انتخاب می‌کند. انتخاب این عمل دو نتیجه به همراه دارد: عامل پاداش فوری دریافت می‌کند (یا يك هزینه

Resource Sharing^{۱۷}

Resource Optimization^{۱۸}

Markov Decision Process^{۱۹}

Discrete-time Markov Chain^{۲۰}

فوری را متحمل می‌شود) و همچنین با توجه به توزیع احتمال تعیین شده توسط انتخاب عمل P ، به حالت جدیدی وارد می‌شود. در این گام از زمان، عامل مجددًا باید تصمیم‌گیری کند، اما اکنون ممکن است سیستم در حالت دیگری باشد و مجموعه اعمال دیگری را برای انتخاب پیش‌رو داشته باشد.

در واقع فرایند تصمیم‌گیری مارکوف، فرایندی جهت مدل‌سازی تصمیم‌گیری در شرایطی است که نتایج تا حدودی تصادفی و تا حدودی تحت کنترل یک تصمیم‌گیرنده است. این فرآیند برای طیف گسترده‌ای از مسائل بهینه‌سازی که از طریق برنامه‌ریزی پویا و یادگیری تقویتی حل می‌شوند، بسیار خاص و منحصر به فرد است.

این مدل به عوامل هوشمند این امکان را می‌دهد تا رفتار ایده‌آل را در یک محیط خاص تعیین کنند، تا در جهت آنچه در پی آن هستند توانایی مدل را برای دستیابی به یک حالت خاص، در یک محیط به‌حداکثر برسانند.

به طور دقیق‌تر فرایندهای تصمیم‌گیری مارکوف فرایندهای کنترل تصادفی زمان‌گسسته است و می‌توان نتیجه گرفت در فرآیند تصمیم‌گیری مارکوف، تصمیم‌گیرنده می‌تواند بر روی حالتی از سیستم با انجام عمل تاثیر بگذارد که موجب می‌شود کارایی از پیش تعریف شده را بهینه کند.

این هدف با توجه به سیاست‌ها و خواسته‌ی مورد نظر، تحت عنوان قوانینی برای رفتار سیستم تعریف می‌شود که وابسته به نوع محیط، در رفتار عامل در محیط اعمال می‌شوند.

فرایند تصمیم‌گیری مارکوف به دنبال بهینه‌سازی اقدامات اخذ شده تا دستیابی به حالت نهایی است. این بهینه‌سازی با یک سیستم بازخورد پاداش انجام می‌شود، به شکلی که اقدامات مختلف بسته به حالت پیش‌بینی شده‌ای که این اقدامات آن را ایجاد می‌کنند، وزن دهنده می‌شوند.

فرآیند تصمیم‌گیری مارکوف توسط مجموعه‌ای از حالت‌ها، اقدامات، یک مدل انتقال و یک تابع پاداش تعریف می‌شود. پاداش و انتقال بر طبق خواص زنجیره مارکوفی، همواره فقط به وضعیت فعلی، عمل انتخاب شده و وضعیت بعدی بستگی دارند، و حالت‌های قبلی اهمیتی نخواهند داشت.

این مدل تصویری از حالت‌های متنوع محیط و اقدامات ممکن در هر حالت را تشکیل می‌دهد، که در هر وضعیت، هر عمل موجب تغییر احتمالی به وضعیت جدید خواهد بود. در هر حالت عامل یک وضعیت را مشاهده می‌کند و عملی را انجام می‌دهد، که باعث می‌شود پاداشی از تابع پاداش تعریف شده

به دست آورده، و یا در سناریوی دیگری هزینه به حداقل برسد، و در همین حال، حالت جانشین فقط به وضعیت فعلی و عمل انتخابی بستگی دارد. همچنین ممکن است حالت جانشین، براساس عدم اطمینان ما در محیطی که جستجو در آن انجام می شود، احتمالاتی باشد. در ادامه به بررسی دقیق‌تر و اجزا تشکیل‌دهنده یک مدل فرایند تصمیم‌گیری مارکوف می‌پردازیم.

۵-۲ تعریف فرایند تصمیم‌گیری مارکوف

از آنجا که سنگ بنای یادگیری تقویتی، فرایندهای تصمیم‌گیری مارکوف هستند؛ هر مسئله یادگیری تقویتی قابل فرموله شدن به کمک فرایند تصمیم‌گیری مارکوف است. ویژگی بارز فرایند تصمیم‌گیری مارکوف که از آن تحت عنوان "خاصیت مارکوف" یاد می‌شود این است که گذار از یک وضعیت به وضعیت بعدی تنها به وضعیت فعلی وابسته است، و نسبت به وضعیتهای قبل از آن بی‌تفاوت است که در زیر بیان ریاضی آن ذکر شده است.

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, X_{n-2} = i_{n-2}, \dots, X_1 = i_1) = P(X_{n+1} = j \mid X_n = i)$$

فرایند تصمیم‌گیری مارکوف، یک مدل ریاضی احتمالی برای یک سناریوی تصمیم‌گیری است. در هر مرحله، تصمیم‌گیرنده یا همان عامل، عملی را انتخاب می‌کند که بخشی از نتایج آن تصادفی و بخشی دیگر نیز نتیجه عمل است. این فرایندها برای مدل‌سازی انواع مسائل بهینه‌سازی به خصوص مسائلی که نیاز به تصمیم‌گیری‌های متوالی و پی‌درپی دارند مورد استفاده قرار می‌گیرند.

فرایند تصمیم‌گیری مارکوف شامل پنج عنصر است که به صورت چندتایی $(\mathcal{S}, \bar{s}, Act, Step, L)$ در زیر مشخص می‌شود:

- \mathcal{S} مجموعه متناهی از حالت‌های سیستم است.

- $\bar{s} \in \mathcal{S}$ حالت اولیه سیستم است.

- Act مجموعه عمل‌ها در سیستم است.

- $Step : \mathcal{S} \rightarrow Dist(s)$ تابع احتمال گذار است.

$\mathcal{S} \rightarrow 2^{AP}$ • تابع برچسب‌گذاری است. برای هر حالت $s \in \mathcal{S}$ ، مجموعه $L(s)$ از گزاره‌های

اتمی نسبت داده می‌شود. در واقع این تابع برای هر حالت مجموعه‌ای از گزاره‌های اتمی را مشخص می‌کند که در آن حالت برقرار هستند.

همانطور که مشخص است، در هر حالت ممکن است چندین عمل قابل اخذ باشد. رفتار یک فرایند تصمیم‌گیری مارکوفی به این شکل است که ابتدا در هر حالت $s \in \mathcal{S}$ ، از میان اعمال قابل اخذ، یک عمل مانند $a \in Act$ با عدم قطعیت انتخاب می‌شود، سپس حالت بعدی با استفاده از تابع توزیع $Step(s, a)$ ، مشخص می‌شود.

۶-۲ یادگیری تقویتی

یادگیری تقویتی^{۲۱} همانند روش‌های یادگیری بانظارت^{۲۲} و یادگیری بدون نظارت^{۲۳}، شاخه‌ای از علم یادگیری ماشین است که عامل طراحی شده برای این نوع از یادگیری بر اساس سطحی از خطا سعی در شناخت و مدل‌سازی محیط و تصمیم‌گیری بر اساس آن می‌کند.

در این شاخه، ورودی شامل بردار ویژگی‌ها به همراه سیگنال امتیاز عمل انجام‌شده در حالت فعلی است، که عامل به کمک آن اقدام به تعیین کیفیت عمل انجام‌شده خود می‌نماید. همانطور که در شکل ۲-۲ مشاهده می‌شود، عامل با انجام اعمال مختلف به ازای حالت‌های مختلف و دریافت بازخورد از محیط، اقدام به شناخت ویژگی‌های محیط نموده و از این طریق به تصمیم‌گیری می‌پردازد.

۷-۲ اجزا مسئله یادگیری تقویتی

در این بخش به تعریف عواملی می‌پردازیم که باعث شکل‌گرفتن و تعریف یک مسئله یادگیری تقویتی می‌گردد. یک مسئله یادگیری تقویتی استاندارد به طور کلی شامل بخش‌های زیر می‌باشد:

^{۲۱} Reinforcement Learning

^{۲۲} Supervised Learning

^{۲۳} Unsupervised Learning

۱-۷-۲ سیگنال امتیاز

یکی از بارزترین مشخصه‌های هر مسئله یادگیری تقویتی وجود سیگنال امتیاز است سیگنال امتیاز یک سیگنال عددی بازخورد^{۴۴} است که عامل از طریق آن متوجه می‌شود با انجام دنباله‌ای از اعمال، چه میزان قابل قبول بوده است. عامل در هر بازه زمانی یک عمل از اعمال تعریف شده خود را بر طبق سیاست اتخاذ شده، انجام داده و نتیجه آن را از طریق سیگنال بازخورد دریافت می‌کند. هدف نهایی عامل در مسئله یادگیری تقویتی بیشینه کردن مجموع امتیازهای به دست آمده از ابتدا تا انتها است. این مهم بر اساس نظریه امتیاز شکل گرفته است که پایه و اساس روش یادگیری تقویتی است.

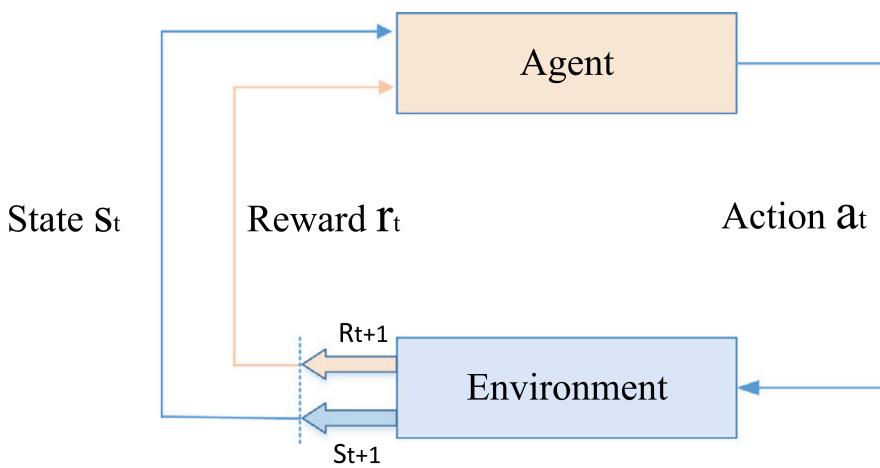
نظریه امتیاز به طور کلی بیان می‌کند که می‌توان اهداف را به صورت بیشینه کردن امتیاز کسب شده در طول مراحل رسیدن به هدف تعریف کرد، و مسائلی که بر پایه یادگیری تقویتی هستند همگی باید شرایط موجود در نظریه جایزه را داشته باشند تا بتوان از روش‌های مرسوم یادگیری تقویتی در آن‌ها استفاده کرد.

سیگنال امتیاز در مسائل یادگیری تقویتی عمدتاً دارای تأخیر زیادی است و معمولاً پس از طی قالب‌های زمانی زیادی نتیجه عمل انجام شده در قالب ۲ به دست می‌آید. ضمن اینکه به دلیل تاثیر مستقیم و وابسته بودن حالت‌های پیش‌آمده به یکدیگر، باید تاثیر تصمیمات فعلی بر آینده دور یا نزدیک را در نظر داشته باشیم که مسئله ما را به یک مسئله تصمیم‌گیری ترتیبی تبدیل می‌کند. در این مسئله هدف بیشینه کردن مجموع امتیاز به دست آمده است.

۲-۷-۲ عامل و محیط

یکی دیگر از تعاریف مهم در یادگیری تقویتی تعریف عامل و محیط فعالیت است. در زیر به تعریف این دو می‌پردازیم. عامل هوشمند عاملی خودمختار است که توسط حسگرهایش محیط را درک کرده، و بر روی محیط تغییر دلخواه را برای رسیدن به هدف تعیین شده، که بیشینه کردن امتیاز اکتسابی است، انجام می‌دهد.

محیط فعالیت: محیطی که عامل در آن قرارگرفته و به فعالیت در آن می‌پردازد محیط فعالیت نامیده می‌شود. جهان پیرامون عامل در یک شبیه‌ساز یک حالت داخلی دارد که حالت فعلی محیط را مشخص



شکل ۲-۲: نحوه تعامل عامل هوشمند یادگیری تقویتی در محیط

می‌کند که بر حسب دسترسی عامل به آن محیط به دسته‌های زیر تقسیم می‌گردد:

- اگر تعامل با حالت محیط دسترسی داشته باشد محیط آشکار^{۲۵} است.
- اگر حالت محیط برای عامل کاملاً قابل دسترس نباشد محیط قسمتی آشکار^{۲۶} است.

یکی از وظایف محیط، دریافت عمل عامل و بهروزرسانی حالت داخلی خود بر حسب آن و محاسبه امتیاز و ارسال آن به عامل است، و این چرخه تا رسیدن به هدف تعیین شده ادامه می‌یابد.

به طور کلی در هر مرحله ارتباط بین عامل و محیط به نحو زیر است:

- در هر بازه زمانی t ، عامل روند زیر را طی می‌دهد:

۱. اجرای عمل a_t

۲. دریافت مشاهده محیط O_t

۳. دریافت امتیاز R_t از محیط

- در هر بازه زمانی t ، محیط روند زیر را طی می‌دهد:

Observable^{۲۵}
Partially Observable^{۲۶}

۱. دریافت عمل a_t از عامل

۲. ارسال ویژگی‌های متناسب با حالت داخلی فعلی O_{t+1}

۳. ارسال امتیاز عمل R_{t+1} به عامل

۳-۷-۲ حالات و تاریخچه

عامل برای فعالیت کارآمد نیازمند کسب اطلاعات جامعی از محیط پیرامون خود است. عامل از این اطلاعات استفاده کرده تا بتواند موقعیت خود را در محیط پیرامون بسنجد و بداند چه اعمالی را در گذشته انجام داده و چه تاثیراتی بر محیط داشته است، تا به کمک آن برای حالات آینده تصمیم‌گیری صحیح انجام دهد. بدین منظور نیازمند مطالعه و بررسی نحوه ذخیره‌سازی تاریخچه اعمال عامل هستیم که مفاهیم زیر به ما در این زمینه کمک می‌نمایند:

- **تاریخچه:** تاریخچه^{۲۷} دخیره‌شده شامل دنباله‌ای از مشاهدات، اعمال و امتیازات دریافت شده توسط عامل است.

$$\mathbf{H}_t = o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t \quad (1-2)$$

- **حالت:** به دلیل پیچیدگی تاریخچه و نیاز به فضای ذخیره‌سازی بالا، نیازمند معرفی یک ساختار ساخت‌یافته‌تر برای ذخیره تاریخچه عامل هستیم که ما را به مفهوم حالت^{۲۸} می‌رساند.

حالت در مفهوم کلی به معنی تابعی از تاریخچه است با این تفاوت که اطلاعات ضروری آن را استخراج کرده و از آن‌ها برای تشخیص موقعیت خود و تصمیم‌گیری استفاده می‌کنند، بدون آنکه درگیر پردازش و ذخیره سازی اطلاعات غیرضروری شود.

$$\mathcal{S}_t = f(\mathbf{H}_t) \quad (2-2)$$

به طور کلی حالت‌های تعریف شده در مسائل یادگیری تقویتی شامل دو دسته کلی زیر است:

۱. حالت درونی:^{۲۹} در داخل عامل ذخیره می‌شود و عامل به کمک آن وضعیت فعلی خویش را درک می‌کند.

۲. حالت بروونی:^{۳۰} یک خصیصه در محیط فعالیت عامل در صورت استفاده از شبیه‌ساز است و وضعیت فعلی محیط را مشخص می‌کند، و از طریق یک بازنمایی از آن عامل می‌تواند درک مناسبی از جهان پیرامون خود به دست آورد و معمولاً حالت داخلی محیط از دید عامل مخفی است.

یکی از مهم‌ترین خصیصه‌های حالت در مسائل یادگیری تقویتی، دارا بودن خاصیت مارکوف درجه اول است. بدین معنی که تنها با داشتن حالت فعلی بتوان اطلاعات لازم برای درک وضعیت فعلی محیط به دست آورد و نیازی به حالت پیشین نباشد. به حالتی که این شرط را ارضاء نماید حالت اطلاعاتی^{۳۱} گوییم. طراحی حالت اطلاعاتی که خاصیت مارکوف درجه اول را ارضاء کند نیازمند برقرار بودن رابطه ۴.۲ است:

$$P[\mathcal{S}_{t+1} | \mathcal{S}_t] = P[\mathcal{S}_{t+1} | \mathcal{S}_1, \dots, \mathcal{S}_t] \quad (3-2)$$

با دانستن حالت دیگر نیازی به نگهداری تاریخچه نیست و می‌توان تاریخچه را کنار گذاشت. به همین دلیل تعریف صحیح حالت تاثیر مهمی در شناسایی محیط توسط تعامل دارد و می‌توان با مشاهده یکسان ولی با نحوه متفاوت طراحی حالت‌ها عامل را به انجام اعمال متفاوت و گاهی اشتباه وادران نمود. به این دلیل طراحی حالت صحیح، یکی از مهم‌ترین اجزاء تعریف مسئله یادگیری تقویتی است.

۴-۷-۲ توابع ارزش و معادله بلمن

تقریباً تمامی الگوریتم‌های یادگیری تقویتی به دنبال تخمین توابع ارزش هستند. این توابع می‌خواهند میزان خوب بودن یک حالت برای بودن عامل در آن که به آن تابع ارزش حالت می‌گوییم، یا میزان خوب بودن انجام عمل a در یک حالت که به آن تابع ارزش عمل می‌گوییم را، تخمین بزنند.

Internal State^{۲۹}

Environment State^{۳۰}

Information State^{۳۱}

از این رو در این بخش قبل از معرفی سیاست و سیاست بهینه باید این دو مفهوم اصلی را تعریف کنیم. تابع ارزش حالت و تابع ارزش اعمال در حالت، توابعی هستند که ما با کمک آنها سیاست بهینه را استخراج می‌کنیم. حال یک سیاست فرضی مثل π را در نظر بگیرید، طبق تعریف ارزش حالت s تحت سیاست π برابر است با:

$$V_\pi(s) = E_\pi[G_t | s_t = s] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \quad (4-2)$$

که به معنای امید ریاضی پاداشهای دریافتی تا به حال با شروع از حالت s و طی کردن مسیرها تحت سیاست π به دست آورده است. به رابطه با معادله بلمن برای $V_\pi(s)$ نیز می‌گویند، که رابطه بین ارزش یک حالت با ارزش حالت‌های بعدی را بیان می‌کند. در فصل‌های بعدی به دنبال تخمین این تابع ارزش با کمک الگوریتم‌های یادگیری تقویتی هستیم. مفهوم دیگری تحت عنوان تابع ارزش اعمال نیز به صورت زیر تعریف می‌شود:

$$Q_\pi(s, a) = E_\pi[R_t | s_t = s, a_t = a] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \quad (5-2)$$

که به معنای امید ریاضی پادashهای دریافتی تا به حال با شروع از حالت s و انجام عمل a و طی کردن مسیرها تحت سیاست π است. حال می‌توانیم از این تعاریف برای تشکیل معادله بلمن استفاده کنیم. لازم به ذکر است که این تبدیل در []، در دسترس است و در این قسمت صرفا فرم نهایی معادله بلمن را نمایش می‌دهیم. معادله بلمن برای ارزش حالت و ارزش اعمال در حالت به صورت زیر نوشته می‌شود:

$$V_\pi(s) = \sum_a \pi(s, a) \sum_{s'} P(s, a, s') [R(s, a, s') + \gamma V_\pi(s')] \quad (6-2)$$

$$Q_\pi(s, a) = \sum_{s'} P(s, a, s') \left[R(s, a, s') + \gamma \sum_{a'} \pi(s', a') Q(s', a') \right] \quad (7-2)$$

۸-۲ بهره‌برداری و اکتشاف

یکی از مسائل مهمی که همیشه در یادگیری تقویتی با آن مواجه بوده‌ایم مسئله سبک‌سنگین کردن^{۳۲} بین بهره‌برداری^{۳۳} و جستجو^{۳۴} است. در ابتدا به تعریف این دو مفهوم می‌پردازیم:

- **بهره‌برداری:** در بهره‌برداری هدف اصلی بیشینه کردن سود ما به ازای هر انتخاب است، به نوعی می‌خواهیم با استفاده از دانش کسب شده، حداکثر بهره ممکن را به دست آوریم. پس در هر زمانی که نیاز به تصمیم‌گیری توسط عامل احساس می‌شود، در صورتی که عامل هدف اصلی اش بهره‌برداری باشد باید عملی را انجام دهد که امتیاز وی را بیشینه نماید.
- **اکتشاف:** در روش اکتشاف هدف اصلی کشف حالات جدید است. در هنگامی که عامل اقدام به کاوش در محیط می‌نماید، بسیاری از محیط‌ها برایش ناشناخته هستند و تاکنون آن‌ها را مشاهده نکرده است. در حالت جست و جو، هدف اصلی مشاهده بیشترین حالت ناشناخته ممکن است، پس باید عملی را انتخاب کنیم که ما را به حالتی جدید ببرد.

در مسائل یادگیری تقویتی هر دو روش بهره‌برداری و اکتشاف را باید به صورت همزمان استفاده کرد، تا بتوانیم به جواب بهینه برسیم. بهره‌برداری را به این دلیل نیاز داریم که امتیاز خود را بیشینه کنیم و به این دلیل از جستجو بهره می‌گیریم تا حالت‌های جدید را به امید اینکه از حالت‌های فعلی امتیاز بیشتری دارند، مشاهده و کشف نماییم. باید بین این دو روش تناسبی برقرار نمود. اصولاً در ابتدا میزان جستجو زیاد است و با مشاهده حالت‌های بیشتر میزان بهره‌وری را زیاد می‌کنیم. با پیگیری این روش می‌توان به جواب بهینه مناسبی دست پیدا کرد.

یکی از ویژگی‌های کلیدی یادگیری تقویتی چالش وزن‌دهی به دو عامل بهره‌برداری و اکتشاف است. اگر عامل بخواهد اعمال بهتری را بیاموزد یا به عبارتی دیگر اعمالی که در نهایت به پاداش انباسته بیشتری منجر شود، باید اعمال جدید را به صورت غیر حریصانه امتحان کند. همچنین اگر عامل از دانش فعلی خود بهره برد و اقدامات قبل شناخته شده برای بازخورد پاداش‌های خوب را دنبال کرده باشد، تضمین نمی‌شود که بازدهی بالاتر از پاداشی که عامل می‌تواند دریافت کند داشته باشد.

Trade-off^{۳۲}

Exploitation^{۳۳}

Exploration^{۳۴}

بنابراین این موضوعی است که یک عامل هنگام تصمیم‌گیری برای اعمال بعدی با آن روبرو می‌شود. یا اعمال تصادفی را امتحان کند و حدس بزند که پاداش بیشتری می‌گیرد اما با خطر بدتر شدن پاداش مواجه است، یا اینکه با عملکرد مطابق با شرایط فعلی خود به پاداش احتمالاً پایین‌تر اما مطمئن دست پیدا کند. به عبارت دیگر اگر عامل تنها اکتشاف را انجام دهد ممکن است در این کار به امتیازات بالاتری نرسد و اعمال خود را بهبود ندهد. در طرف دیگر اگر فقط از استخراج استفاده شود ممکن است در خطمشی فعلی خود با دیدن تمام خط سیرهای احتمالی گیر کند. از همین رو عامل احتمالاً خطمشی بهینه را از دست خواهد داد. بنابراین باید یک تعادل مناسب بین اکتشاف و بهره برداری وجود داشته باشد.

این معضل از آنجایی ناشی می‌شود که معمولاً فرایند یادگیری در یادگیری تقویتی به صورت برخط صورت می‌گیرد. به عبارت دیگر به یادگیری تقویتی هیچ داده‌ای همانند یادگیری با نظارت داده نمی‌شود بنابراین عامل خود به نوعی به دنبال جمع‌آوری داده‌ها است و از طریق اعمالی که انجام می‌دهد بر داده‌های مشاهده شده اثر می‌گذارد، و از همین رو گاهی ارزش دارد که اعمال مختلفی را برای به دست آوردن داده‌های جدید انجام دهد.

از این رو برای ایجاد توازن جهت یادگیری شبکه‌های عصبی، می‌توان از استراتژی انتخاب حریصانه – اپسیلون استفاده کرد، که روشی ساده و در عین حال کاربردی برای انتخاب عمل در هر مرحله به عنوان استراتژی انتخاب، انتخاب حریصانه – اپسیلون است. در این روش یک پارامتر بین صفر تا یک انتخاب می‌شود که اقدام عامل مبنی بر اینکه بهره برداری یا اکتشاف انجام دهد را کنترل می‌کند. با استفاده از این روش در هر زمان عامل به طور احتمالی بین بهره برداری و استخراج یکی را انتخاب می‌کند و با احتمال (ϵ) انتخاب تصادفی از بین تمام اعمال موجود و با احتمال ($1 - \epsilon$) بهره برداری را انجام می‌دهد. و مقدار (ϵ) در طی مراحل یادگیری همواره کاهش می‌یابد. عبارت زیر این امر را تعریف می‌نماید.

۹-۲ روش‌های پاسخ به مسائل یادگیری تقویتی

در این قسمت می‌خواهیم به بررسی روش‌هایی که برای حل مسائل یادگیری تقویتی به کار می‌روند پپردازیم. روش‌های مختلفی برای این منظور ارائه شده است که به تفصیل به توضیح آنها پرداخته و

نقاط ضعف و قوت آنها را بررسی می‌کنیم.

۱-۹-۲ روش تکرارشونده ارزش

در این روش نیازی به داشتن Π که نمایانگر سیاست تعامل در مواجهه با حالت‌های مختلف است، نیست و به جای آن مقدار $(s)\Pi$ در هر جا که نیاز باشد، از روی $V(s)$ به صورت زیر محاسبه می‌شود.

$$V_{i+1}(s) = \max_a \left[\sum_{s'} P_a(s, s')[R(s, a, s') + \gamma V_i(s)] \right] \quad (8-2)$$

مقدار V_i نمایش‌دهنده تکرار i است و مقدار اولیه V_0 که بیانگر نقطه شروع اعمال روش است، را به صورت تخمینی از مقادیر ممکن تعیین می‌کنیم. با اجرای الگوریتم و محاسبه مقدار V_i قدم به قدم به جواب اصلی همگرا می‌شویم.

۲-۹-۲ روش تکرارشونده سیاست

این روش با یک سیاست تصادفی آغاز می‌شود و عامل در هر دور اجرای الگوریتم سعی در بهینه‌تر کردن سیاست اتخاذ شده می‌کند. این روش از برنامه‌نویسی پویا^{۳۵} استفاده می‌کند و هدف اصلی آن به دست آوردن سیاست بهینه است. روش تکرارشونده سیاست از دو مرحله کلی زیر تشکیل شده است:

۱. ارزیابی سیاست:^{۳۶}

در این مرحله عامل سعی می‌کند به ازای سیاست داده شده مقدار $V_{\Pi}(s)$ که نمایانگر تخمین امتیاز به دست آمده با پیگیری سیاست π است، محاسبه و بهروزرسانی نماید.

۲. بهبود سیاست:^{۳۷}

در این مرحله عامل سعی می‌کند با توجه به مقدار $V_{\Pi}(s)$ محاسبه شده در مرحله قبل، سیاست جدید را بهروزرسانی کند.

Dynamic Programming^{۳۵}

Policy Evaluation^{۳۶}

Policy Improvement^{۳۷}

با استفاده از روش‌های توضیح داده شده، به راحتی می‌توان سیاست بهینه را به دست آورد و بر اساس آن امتیاز عامل را بیشینه نمود.

مشکل اصلی روش‌های بیان شده، نیاز به ذخیره مقدار امتیاز هر حالت در یک جدول^{۳۸} برای مشاهده مقادیر از روی آن است که در مسائل با تعداد حالت‌های بالا عملاً به دلیل نیاز به حافظه فراوان غیرقابل استفاده است. بدین منظور از روش‌های تقریب تابع^{۳۹} استفاده می‌کنیم.

۱۰-۲ تقریب توابع

همانطور که پیشتر اشاره شد یکی از مشکلات اصلی روش‌های مبتنی بر برنامه نویسی پویا نیاز فراوان آن‌ها به حافظه است، که عملاً استفاده از آن‌ها را برای مسائل پیچیده منتفی می‌نماید [۱۴]. به همین دلیل از روش‌های دیگری برای محاسبه امتیاز استفاده می‌شود.

این روش‌ها سعی در تقریب تابع‌ای دارند که حالت فعلی را به صورت ورودی دریافت کرده و امتیاز آن را محاسبه می‌نماید. این روش‌ها توانایی خوبی در زمینه تشخیص حالت‌های نزدیک به یکدیگر و گرفتن تصمیم مناسب در این حالت‌ها ندارند. در حالت کلی دو نوع تابع زیر را برای تصمیم‌گیری استفاده می‌کنیم:

۱. **تابع ارزش:**^{۴۰} این تابع حالت فعلی را به عنوان ورودی گرفته، و با تخمین امتیاز کسب شده از حالت فعلی تا حالت نهایی، با رفتن به حالت جدید امکان تصمیم‌گیری برای اینکه کدامیک از حالت‌های ممکن بهترین حالت برای ادامه مسیر هستند را برای ما ممکن می‌کند.

۲. **تابع امتیاز-عمل:**^{۴۱} این تابع با دریافت حالت فعلی و عمل دلخواه، امتیازت تخمینی در صورت انجام عمل ورودی و مسیر از حالت فعلی به حالت نهایی را محاسبه می‌کند، و می‌توان بر اساس آن برای پیدا کردن سیاست بهینه تصمیم‌گیری نمود.

Look Up Table ^{۳۸}	Function Approximation ^{۳۹}	Value Function ^{۴۰}
Action-value Function ^{۴۱}		

این دو تابع، توابع اصلی در روش یادگیری تقویتی هستند، که ما سعی در تخمین آنها برای تصمیم‌گیری صحیح داریم. روش‌های متفاوتی برای تخمین تابع وجود دارد که یکی از بهترین روش‌های فعلی برای تخمین تابع استفاده از شبکه‌های عصبی عمیق^{۴۲} است.

۱۱-۲ شبکه‌های عصبی عمیق

شبکه‌های عصبی، سیستم‌ها و روش‌های نوین محاسباتی هستند، که در تخمین پاسخ‌های مسئله‌های پیچیده در یادگیری ماشین استفاده می‌شوند [۱۵]. این شبکه‌ها از ترکیب واحدهای محاسباتی کوچکی به نام نورون^{۴۳} با آرایش‌های خاصی همانطور که در شکل ۳-۲ مشاهده می‌شود، ساخته شده‌اند که توانایی حل دامنه وسیعی از مسائل را دارا می‌باشند. مشکل اصلی شبکه‌های عصبی کم‌عمق^{۴۴} این است که این شبکه‌ها، تخمین‌گر خوبی در فضای محلی داده‌های ورودی هستند. یک تصمیم محلی تنها در صورتی می‌تواند برای داده‌های جدید ورودی به خوبی تصمیم‌گیری نماید که داده‌های آموزشی در همسایگی X بتوانند به خوبی مقدار تابع در X را توصیف نمایند. در شکل ۲-۲ یک شبکه عصبی در حالت کلی نشان داده شده است.

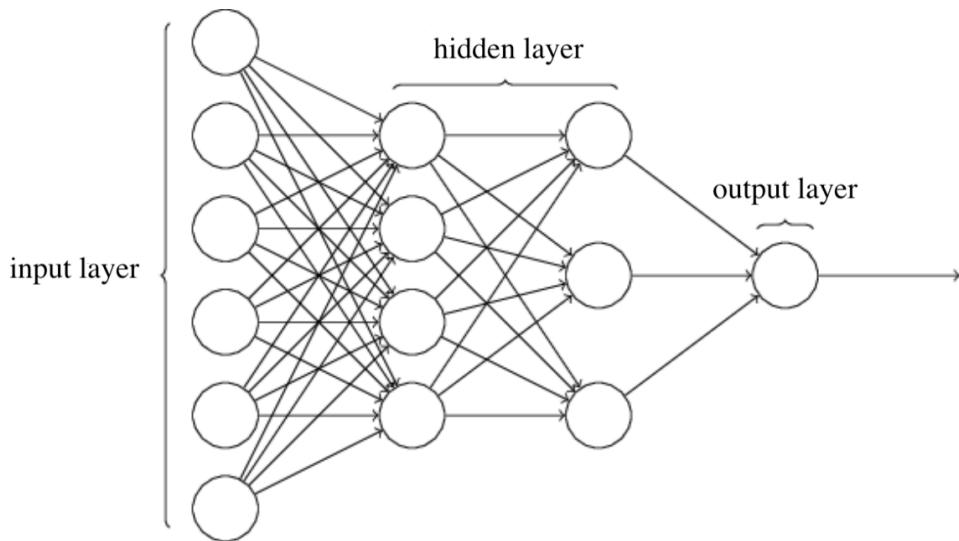
شبکه‌های عصبی کم‌عمق توانایی افزایش فضای ورودی به ناحیه‌های مختلفی را دارند، و می‌توانند مشخصات و پارامترهای لازم برای توصیف هر کدام از این نواحی را به دست آورند. وقتی تابع پیچیده باشد تعداد نواحی زیاد شده و به همین دلیل شبکه نیاز به تخمین پارامترهای بسیاری دارد. برای یادگیری و تنظیم این پارامترها نیازمند تعداد بسیار زیادی داده آموزشی هستیم، که در عمل در بسیاری از مسائل به تعداد مورد نیاز، داده آموزشی در اختیار نداریم. به همین دلیل توانایی شبکه‌های عصبی کم‌عمق در حل مسائل پیچیده به شدت کاهش می‌یابد و برای حل مسائل پیچیده به استفاده از شبکه‌های عصبی عمیق روی می‌آوریم [۱۶].

شبکه عصبی عمیق دارای تعریف روشی نیست و اصولاً^{۴۵} به شبکه‌ای گفته می‌شود که دارای تعداد لایه‌های بیش از سه عدد است و در اتصالات بین این لایه‌ها از تابع غیر خطی استفاده شود. این مهم به

Deep Neural Networks^{۴۲}

Neuron^{۴۳}

Shallow^{۴۴}



شکل ۲-۳: نمونه‌ای از یک شبکه عصبی

ما این توانایی را می‌دهد که بتوانیم مسائل بسیار پیچیده را تحلیل کرده و پاسخ مناسب آن‌ها را پیش‌بینی کنیم، به همین دلیل مسائل بسیاری را می‌توان ذیل شبکه‌های عصبی عمیق تحلیل نمود.

استفاده از شبکه‌های عصبی عمیق در سال‌های اخیر فراگیر گشته است. در سال‌های پیشین با توجه به قدرت محدود محاسباتی و عدم وجود روش‌های بهینه‌سازی کارا، شبکه‌های عصبی عمیق توانایی محاسبه پاسخ مناسب در بسیاری از مسائل را نداشتند. علیرغم وجود روش بازنثر خطای^{۴۵} برای یادگیری مولفه‌های شبکه‌های چند لایه همچنان لایه‌های شبکه نمی‌تواند از تعداد مشخصی بیشتر شود. مشکل اصلی در الگوریتم پساننتشار خطای و نحوه بهینه‌سازی است، که وقتی تعداد لایه‌های شبکه از حدی بیشتر می‌شود، در عمل توانایی خود را برای پیدا کردن بهینه مناسب از دست می‌دهد [۱۶].

یادگیری تقویتی عمیق به استفاده از شبکه‌های عصبی عمیق به عنوان تقریب توابع در تابع مقدار یا خط‌مشی در چهارچوب یادگیری تقویتی اشاره دارد و در روش‌های گرادیان سیاست^{۴۶}، یادگیری Q و عامل-منتقد^{۴۷} با موفقیت اعمال شده است. در ادامه به تشریح الگوریتم‌های مدرن یادگیری تقویتی عمیق که در یادگیری تقویتی عادی نیز کاربرد دارند اشاره خواهد شد.

Error Back Propagation^{۴۵}

Policy Gradient^{۴۶}

Actor-Critic^{۴۷}

در ابتدا نیاز است تا توضیحاتی راجع به مفهوم شبکه‌های عصبی و ساختار آن ارائه شود، شبکه‌های عصبی مصنوعی یک شبیه‌سازی از ساختار مغز انسان است و بر پایه این فرضیات می‌باشد این شبکه‌های عصبی از تعداد زیادی واحد ساده به نام نورون تشکیل شده است:

- پردازش اطلاعات در ساختارهایی ساده با تعداد زیاد به نام نورون‌ها انجام می‌گیرد.
- سیگنال‌ها از طریق اتصالات بین نورون‌های شبکه منتقل می‌شوند.
- هر اتصال وزن مربوط به خود را دارد که در یک شبکه عصبی این وزن‌ها در سیگنال انتقالی ضرب می‌شوند.
- هر نورون یک تابع فعال‌سازی^{۴۸} را بر ورودی‌های خود که جمع وزن‌دار سیگنال‌های ورودی می‌باشد، اعمال کرده تا سیگنال خروجی را ایجاد نماید.

۱۲-۲ مدل محیط در روش‌های مبتنی بر یادگیری تقویتی

همان‌طور که پیش‌تر گفته شد، هدف اصلی عامل در یادگیری تقویتی جمع‌آوری بیش‌ترین میزان پاداش در بلندمدت است. در جهت انجام این کار عامل باید یک خط‌مشی بهینه برای رفتار در محیط پیدا کند. حال این محیط می‌تواند قطعی و یا تصادفی باشد. به همین منظور در یادگیری تقویتی مسائل اغلب از منظر ریاضی به عنوان یک فرایند تصمیم‌گیری مارکوف بیان می‌شوند، چرا که این فرایندها روشی برای نمایش پویایی محیط هستند. تابع پاداش با توجه به وضعیت فعلی عامل در محیط و عملی که توسط آن در محیط انجام می‌شود، تعریف می‌شود. توابع پاداش و انتقال اغلب "الگوی محیط" نامیده می‌شوند. با این حال گاهی اوقات توابع پاداش و انتقال در دسترس نخواهد بود، از همین رو نمی‌توانیم خط‌مشی را تخمین بزنیم، زیرا ناشناخته است. در غیاب این توابع، در جهت تخمین خط‌مشی بهینه نیاز به تعامل با محیط و مشاهده پاسخ‌های آن است که اغلب به عنوان مشکل یادگیری تقویتی از آن یاد می‌شود.

با گذشت زمان عامل شروع به درک نحوه واکنش محیط به اقدامات خود می‌کند و می‌تواند خط‌مشی بهینه را تخمین بزند. بنابراین در مسائل یادگیری تقویتی عامل خط‌مشی بهینه را برای رفتار در یک محیط

ناشناخته با تعامل با آن با استفاده از روش "آزمون و خطا" تخمین می‌زند. براین اساس الگوریتم‌های یادگیری تقویتی را می‌توان به دو دسته کلی الگوریتم‌های مبتنی بر مدل و یا بدون مدل تقسیم‌بندی کرد:

- **روش‌های مبتنی بر مدل:** در الگوریتم‌های مبتنی بر مدل عامل به یک مدل کامل از محیط دسترسی دارد یا سعی می‌کند از طریق تعامل بیاموزد و بهمنظور برآورد درست خط‌مشی بهینه از تابع انتقال و پاداش استفاده می‌کند. عامل می‌داند که با توجه به وضعیت فعلی و عمل موردنظر چقدر احتمال دارد به یک وضعیت خاص وارد شود. با اینحال باید توجه داشت که توابع انتقال و پاداشی که عامل برای بهبود تخمین خط‌مشی بهینه خود استفاده می‌کند ممکن است فقط تقریبی از توابع واقعی باشد. از این رو خط‌مشی بهینه به دلیل این تقریب‌ها ممکن است هرگز یافت نشود.
- **روش‌های بدون مدل:** در مقابل الگوریتم‌های مبتنی بر مدل، الگوریتم‌های بدون مدل هیچ دانش اولیه در مورد تابع انتقال ندارند و باید ضمن یادگیری در یافتن مسیرهای کارآمد آن را بیاموزند. به عبارت دیگر یک الگوریتم بدون مدل "تابع مقدار" و یا "خط‌مشی" را مستقیماً از طریق تجربه یعنی با تعامل بین عامل و محیط تخمین می‌زند، بدون اینکه از توابع انتقال و پاداش اطلاعی داشته باشد.

هر دو روش دارای نقاط قوت و ضعف هستند. روش‌های بدون مدل تا حدودی تضمین می‌کنند که در نهایت خط‌مشی بهینه را پیدا می‌کنند. با اینحال آنها از داده‌ها در طول آزمایشات بسیار ناکارآمد استفاده می‌کنند و بنابراین اغلب برای دست‌یابی به عملکرد خوب به تجربه زیادی نیاز دارند. در مقابل الگوریتم‌های مبتنی بر مدل می‌توانند بر این مشکل فائق بیایند، اما عامل تنها برای یک مدل خاص یاد می‌گیرد و گاهی اوقات برای برخی از مدل‌های دیگر مناسب نیست.

۱۳-۲ جمع‌بندی و نتیجه‌گیری

با مطالعه ادبیات موجود در زمینه پژوهشی مورد مطالعه به ارائه روشی نوین جهت تخصیص بهینه منابع در یک محیط محاسباتی لبه می‌پردازیم. با توجه به این که توسعه دستگاههای هوشمند متحرک و بهبود ارتباطات و قابلیت‌های ادراکی جدید موجب تکثیر بسیاری از برنامه‌های کاربردی پیچیده و محاسباتی

شده است. از این رو دستگاه‌های هوشمند با منابع محدود بیش از هر زمان دیگری با محدودیت‌های شدید ظرفیت روبرو هستند. بنابراین می‌توان سیر رشد سریع تحقیقات در این حوزه را شاهد بود.

پردازش وظایف محاسباتی تولید شده هر روزه افزایش خواهد یافت و نیاز به روش‌هایی هوشمند و پویا در مدیریت سیستم بسیار اهمیت خواهد داشت. با مطالعه پژوهش‌های انجام‌شده می‌توان دریافت که در نظرگرفتن شرایط دنیای واقعی اعم از پویایی اجتناب ناپذیر نیازهای برنامه‌های کاربردی، باعث ازکارافتدگی روش‌های پیشین در مواجهه با مسائل تخصیص منابع شده است. از طرفی عملکرد درخشنان یادگیری ماشین و به ویژه یادگیری تقویتی در مسائل دنیای واقعی، ما را به سمت انجام پژوهشی در این زمینه سوق داد.

دنیای امروز، دنیای عدم قطعیت در تمامی ابعاد است. در پژوهش پیش رو سعی شده است تا یکی از انواع عدم قطعیت در مسائل تخصیص منابع را شبیه‌سازی کرده و به کمک یادگیری تقویتی یک رویکرد کارا برای حل آن ارائه کنیم.

فصل ۳

پژوهش‌های مرتبط پیشین

در این فصل پژوهش‌های مرتبط پیشین در این حوزه را مورد بررسی قرار می‌دهیم.

۱-۳ مقدمه

در مسائل مربوط به تخصیص منابع در محاسبات لبه‌ای و ابری، منابع محاسباتی مانند پردازشگرهای حافظه و پهنه‌ای باند باید به کاربران تخصیص داده شود. با توجه به محدود بودن این منابع در سرورهای لبه شبکه و فاصله زیاد سرورهای ابری، تخصیص منابع به منظور توزیع بهینه منابع محدود و همچنین بهبود عملکرد سیستم از اهمیت ویژه‌ای برخوردار است. از جمله محدودیت‌هایی که برای تخصیص منابع وجود دارد کمبود منابع موجود، محدودیت‌های انرژی مصرفی و تأخیر است.

ایده اصلی در پژوهش‌های پیشین، تخصیص منابع به منظور بهینه‌سازی یک تابع هدف است، که این بهینه‌سازی معمولاً کمینه کردن تأخیر یا انرژی مصرفی و یا ترکیب خطی از هر دوی آنها است. همچنین پژوهش‌های پیشین را می‌توان از این منظور که تخصیص دهنده منابع و تصمیم‌گیرنده بارسپاری به صورت متمرکز یا غیر متمرکز عمل می‌کند، طبقه‌بندی کرد.

محوریت پژوهش‌های موجود پیرامون تخصیص منابع در محاسبات لبه سیار، تخلیه و بارسپاری محاسبات می‌باشد. دو سوال اصلی در مورد بارسپاری محاسبات وجود دارد. اولین سوال این است

که یک دستگاه هوشمند سیار، در مواجهه با شرایط مختلف باید محاسبات خود را به یک گره لبه بارسپاری کند یا به صورت محلی انجام دهد. سوال دوم این است که اگر یک دستگاه تصمیم به بارسپاری محاسبات گیرد، آنگاه باید محاسبات خود را به کدام یک از گره‌های لبه تخلیه کند. برای پرداختن به این سوالات، برخی از آثار موجود، الگوریتم‌های بارسپاری محاسبات را پیشنهاد کردند.

۲-۳ پژوهش‌های پیشین

پژوهش‌های بسیاری وجود دارد که مسئله تخصیص منابع محاسباتی در لبه شبکه را به صورت مسئله برنامه‌ریزی عدد صحیح آمیخته مدل می‌کنند. به عنوان مثال در [۱۷] بعد از مدل کردن مسئله به صورت برنامه‌ریزی عدد صحیح آمیخته یک الگوریتم جستجوی ابتکاری را برای بارسپاری چندین محاسبه در یک شبکه شامل چند سرور لبه سیار ارائه می‌دهد، و به صورت بازگشتی متغیرهای دو حالتی مربوط به تصمیم‌گیری بارسپاری را به دست می‌آورد.

یکی دیگر از روش‌هایی که در بسیاری از پژوهش‌ها استفاده شده است، ضعیف کردن شرایط مسئله برای رسیدن به یک مسئله بهینه‌سازی محدب است. این کار معمولاً با تضییف شرط دو حالتی بودن متغیرهای تصمیم‌گیری به متغیرهای پیوسته بین صفر و یک انجام می‌شود [۱۸]. همچنین در [۱۹] علاوه بر ضعیف کردن شرط دو حالتی متغیرهای تصمیم‌گیری، توان محاسباتی سرور ابری، بی‌نهایت در نظر گرفته شده است.

در [۲۰] بارسپاری به صورت یک مسئله بهینه‌سازی با تابع هدف تأخیر متوسط مدل شده است، و یک الگوریتم برای حل کردن موثر مسئله بهینه‌سازی ارائه شده است. در این مقاله ورود بارهای محاسباتی که به صورت واحد مدل شده‌اند به صورت پواسون^۱ در نظر گرفته شده است. در نهایت برای رسیدن به متوسط بهینه تأخیر محاسبات، یک مسئله بهینه‌سازی غیر چندجمله‌ای^۲ بیان شده و یک الگوریتم ابتکاری برای به دست آوردن سیاست بهینه بارسپاری ارائه شده است.

نویسنده‌ان در [۲۱] الگوریتمی را برای تعیین تصمیمات تخلیه دستگاه‌های تلفن همراه در جهت

Poisson^۱
Polynomial^۲

به حداکثر رساندن بازده شبکه پیشنهاد کرده‌اند. در [۲۲] بر روی یک سناریوی محاسبات لبه متحرک بی‌سیم تمرکز کرده‌اند و الگوریتمی برای بهینه‌سازی مشترک تصمیمات تخلیه و انتقال نیرو پیشنهاد کرده‌اند. در آثار [۲۱] و [۲۲]، ظرفیت محاسباتی موجود در گره‌های لبه برای هر دستگاه تلفن همراه، مستقل از تعداد وظایف تخلیه‌شده‌ی آن دستگاه به گره لبه درنظر گرفته شده است.

در عمل گره‌های لبه ممکن است ظرفیت پردازش محدودی داشته باشند، بنابراین ظرفیت پردازشی که یک گره لبه به دستگاه تلفن همراه اختصاص می‌دهد به سطح بار در گره لبه بستگی دارد. هنگامی که تعداد زیادی از دستگاه‌های تلفن همراه وظایف خود را در یک گره لبه به خصوص تخلیه کنند، بار محاسباتی در آن گره لبه می‌تواند بیش از حد افزایش یابد و از این‌رو آن وظایف بارگذاری شده ممکن است با تأخیر پردازش زیادی مواجه شوند. حتی ممکن است برخی از وظایف با اتمام مهلت آنها، منقضی شوند.

برخی از پژوهش‌های موجود به سطوح بار محاسباتی در گره‌های لبه پرداخته‌اند و الگوریتم‌های تخلیه وظیفه متمرکز را پیشنهاد کرده‌اند. نویسنده‌گان در [۲۲] الزامات محاسباتی نامشخص دستگاه‌های تلفن همراه را در نظر گرفته‌اند و الگوریتی را پیشنهاد کرده‌اند که تصمیمات تخلیه دستگاه‌های تلفن همراه و تخصیص منابع محاسباتی گره لبه را بهینه می‌کند. در [۲۴] بر روی وظایف حساس به تأخیر متمرکز شده‌اند و الگوریتمی را برای به حداقل رساندن مصرف انرژی تخلیه محاسبات با توجه به محدودیت مهلت وظیفه پیشنهاد کرده‌اند. در [۱۲] یک شبکه فوق متراکم تعریف شده توسط نرم‌افزار در نظر گرفته شده و یک الگوریتم متمرکز برای به حداقل رساندن تأخیر پردازش وظیفه طراحی کرده‌اند. به همین شکل در [۲۵] بهینه‌سازی مشترک تخلیه و مسیریابی وظیفه را با درنظر گرفتن الزامات نامتقارن وظایف، مورد مطالعه قرار داده‌اند. با این حال، این الگوریتم‌های متمرکز در [۲۳] - [۲۵] ممکن است به اطلاعات کلی سیستم (به عنوان مثال، ورود و اندازه وظایف همه دستگاه‌های تلفن همراه) نیاز داشته باشند و سربار بالایی داشته باشند.

آثاری دیگر الگوریتم‌های تخلیه وظیفه توزیع شده را با درنظر گرفتن سطوح بار محاسباتی در گره‌های لبه پیشنهاد کرده‌اند، به شکلی که هر دستگاه تلفن همراه تصمیم تخلیه وظیفه خود را به شیوه‌ای غیرمتمرکز می‌گیرد. طراحی چنین الگوریتم توزیع شده‌ای چالش‌های زیادی به همراه دارد، به این دلیل که وقتی یک دستگاه تصمیم به تخلیه وظیفه می‌گیرد، وضعیت سطوح بار محاسباتی در گره‌های لبه را نمی‌داند، چراکه

سطح بار، به تصمیم‌گیری‌های تخلیه و مدل‌های وظیفه (مثلًاً اندازه و زمان رسیدن) در سایر دستگاه‌های تلفن همراه نیز بستگی دارد.

برای رسیدگی به این چالش‌ها، نویسنده‌گان در [۲۶] بر روی وظایف قابل تقسیم مرکز شده‌اند و یک الگوریتم مبتنی بر لیاپانوف^۳ را برای اطمینان از پایداری صفاتی وظیفه پیشنهاد کرده‌اند. در [۲۷] تعامل بارگذاری استراتژیک بین دستگاه‌های تلفن همراه را در نظر گرفته و یک الگوریتم توزیع شده مبتنی بر هزینه پیشنهاد کرده‌اند. نویسنده‌گان در [۲۸] یک الگوریتم بارگذاری بالقوه مبتنی بر نظریه بازی طراحی کرده‌اند تا کیفیت تجربه هر دستگاه را به حداقل برسانند. در [۲۹] یک الگوریتم تخلیه توزیع شده برای رسیدگی به رقابت کانال‌های بی‌سیم در بین دستگاه‌های تلفن همراه پیشنهاد کرده‌اند. در [۳۰] یک روش مبتنی بر برآورد، پیشنهاد کرده‌اند که در آن هر دستگاه تصمیم‌گیری تخلیه خود را بر اساس ظرفیت‌های محاسبات و قدرت انتقال خود انجام می‌دهد.

در این کار، ما بر چالش‌های موجود در تخلیه وظایف محاسباتی در یک سیستم محاسبات لبه متحرك تمرکز می‌کنیم و یک الگوریتم توزیع شده را پیشنهاد می‌کنیم که به شکلی منعطف، وظایف محاسباتی ناشناخته را در گره‌های لبه قرار می‌دهد. به شکلی که در مقایسه با پژوهش‌های فوق الذکر [۲۶] - [۳۰]، ما یک سناریوی محاسبات لبه متحرك متفاوت و واقع گرایانه را در نظر می‌گیریم. در پژوهش [۲۶] وظایف محاسباتی قابل توجهی در نظر گرفت شده‌است، اما از آنجایی که وظایف می‌توانند به صورت خودسرانه تقسیم شود، ممکن است به دلیل وابستگی بین بخش‌های تقسیم شده، این امر به دور از یک شرایط واقع گرایانه تلقی شود.

اگرچه آثار [۲۷] - [۳۰]، وظائف محاسباتی را به عنوان وظایف غیرقابل تقسیم در نظر گرفته‌اند، اما این سیستم‌ها، صفاتی اساسی را در نظر نمی‌گیرند. به عنوان یک نتیجه، محاسبات و انتقال هر وظیفه، همواره باید در یک واحد زمانی انجام شود، که در عمل ممکن است این امر همیشه تضمین نشود. متفاوت از این آثار [۲۶] - [۳۰]، ما وظایف غیرقابل تقسیم را با سیستم‌های صفت‌بندی در نظر می‌گیریم و سناریوی عملی را که در آن محاسبات و انتقال یک وظیفه می‌توانند برای چندین واحد زمان ادامه یابد را گسترش می‌دهیم. این سناریو چالش برانگیز خواهد بود، زیرا زمانی که محاسبات جدید وارد می‌شوند، تأخیر آن‌ها را می‌توان تحت تأثیر تصمیمات وظایف دستگاه‌های دیگر قرار داد.

متفاوت از کارهای مرتبط [۲۶] - [۳۰] که وظایف متحمل تأخیر را در نظر می‌گرفتند، ما وظایف حساس به تأخیر را با مهلت‌های زمانی پردازش درنظر می‌گیریم. پرداختن به این موضوع بسیار جذاب است، زیرا مهلت‌های پردازش بر پویایی سطح بار محاسباتی در گره‌های لبه و در نتیجه بر تأخیر وظایف بارگذاری شده تأثیر می‌گذارند.

همچنین میزان تحمل پذیری تأخیر در وظایف حساس به تأخیر را می‌توان چالشی اساسی در سناریوهای عملی در نظر گرفته شده دانست، زیرا مهلت‌های وظایف می‌توانند سطح بار محاسبات در گره‌های لبه را تحت تأثیر قرار دهند، و از این‌رو موجب تأخیر وظایف تخلیه شده شوند. تحت سیستم محاسبات لبه متحرک فوق‌الذکر، استفاده از روش‌های سنتی مانند نظریه‌بازی و بهینه‌سازی آنلاین به دلیل تعامل پیچیده در میان وظایف، دشوار است. برای رسیدگی به این چالش‌ها، تکنیک‌های یادگیری تقویتی عمیق (به عنوان مثال یادگیری Q عمیق [۳۱]) روش‌های مناسبی در مواجهه با چالش تخلیه وظیفه در سیستم محاسبات لبه متحرک تلقی می‌شوند، زیرا این روش‌ها عوامل را قادر می‌سازند تا بر اساس مشاهدات محلی و بدون اطلاعات محیط تصمیم‌گیری نمایند.

برخی از آثار موجود مانند [۳۲] - [۳۳]، از الگوریتم‌های مبتنی بر یادگیری تقویتی عمیق در سیستم محاسبات لبه متحرک به شکل متمرکز بهره گرفته‌اند. در حالی که آنها بر روی الگوریتم‌های تخلیه متمرکز، تمرکز کرده‌اند، نویسنده‌گان در [۳۴] یک الگوریتم تخلیه توزیع شده مبتنی بر یادگیری تقویتی عمیق را پیشنهاد کرده‌اند، که به رقابت کانال‌های بی‌سیم بین دستگاه‌های تلفن همراه می‌پردازد، و این در حالی است که الگوریتم پیشنهادی در هر دستگاه تلفن همراه به اطلاعات کیفیت خدمات در سایر دستگاه‌های تلفن همراه نیاز دارد.

نویسنده‌گان در [۳۵] یک رویکرد مبتنی بر یادگیری تقویتی عمیق را برای تخصیص منابع در محاسبات لبه پیشنهاد کرده‌اند. این پژوهش مسئله تخصیص منابع را به عنوان یک فرآیند تصمیم‌گیری مارکوف فرموله می‌کند، و همچنین یک الگوریتم شبکه Q عمیق بهبودیافته را برای یادگیری خط‌مشی سیستم پیشنهاد می‌کند، که در آن حافظه‌ای برای ذخیره جداگانه تجربیات و تأثیر متقابل اعمال در نظر گرفته می‌شود. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم پیشنهادی از نظر هم‌گرایی از الگوریتم اصلی شبکه Q عمیق بهتر عمل می‌کند و سیاست مربوطه بهتر از سایر سیاست‌ها در مورد زمان تکمیل وظیفه عمل می‌کند.

در [۶] نویسنده‌گان به موضوع تخصیص منابع در اینترنت اشیا متمرکز شده‌اند و از سطح رضایتمندی تجربه مشتری برای دستیابی به خدمات محتوامحور^۴ استفاده می‌کنند. در این پژوهش با بهره‌گیری از مکانیسم یادگیری تقویتی در راستای دست‌یافتن به عملکرد بهینه در تخصیص منابع، یک رویکرد جدید ارائه شده است. این کار ترکیبی از کیفیت تجربه مشتری و یادگیری تقویتی را برای ایجاد جداول نگاشت هزینه‌ی از پیش ذخیره‌شده استفاده می‌کند. مقادیر جدول هزینه توسط بازخورد از طرف کاربر و از طریق معیار کیفیت تجربه مشتری تعیین می‌شود.تابع ارزش در یادگیری تقویتی از سطح تجربه مشتری به عنوان پاداش برای بروزرسانی تصمیمات استفاده می‌کند. این مطالعه به پیاده‌سازی شبکه‌ای محتوامحور برای تحقق تخصیص منابع بهینه تأکید دارد. تابع هزینه مورد استفاده در یادگیری، هم هزینه انرژی و هم تأخیر پاسخ را در نظر می‌گیرد، که قابل توسعه و گسترش برای سایر برنامه‌ها است.

در [۳۶] یک سیستم محاسبات لبه چندکاربره در نظر گرفته شده‌است، که در آن تجهیزات کاربر می‌توانند بارگیری محاسباتی را از طریق کانال‌های بی‌سیم به یک سرور لبه انجام‌دهند. جمع هزینه تأخیر و مصرف انرژی به عنوان هدف بهینه‌سازی در نظر گرفته شده‌اند. در این پژوهش به طور مشترک بارگیری و تخصیص منابع محاسباتی برای بهینه‌سازی تعیین شده‌اند. به همین منظور نویسنده‌گان در این مطالعه از یادگیری تقویتی عمیق در جهت کشف سیاست بهینه استفاده کرده‌اند.

یادگیری تقویتی یک هدف بلند مدت را در نظر می‌گیرد، که برای سیستم‌های پویا از نوع محاسبات لبه متحرک چندکاربره بسیار مورد توجه است. به همین شکل در [۳۷] مدلی هوشمند برای تخصیص بهینه منابع بر اساس چارچوب یادگیری تقویتی عمیق برای تخصیص تطبیقی منابع شبکه و نیازهای محاسباتی طراحی شده‌است. این مدل برای آموزش عوامل یادگیری تقویتی عمیق به صورت توزیع شده، چارچوب یادگیری انجمنی^۵ را با محاسبات لبه متحرک تلفیق می‌کند. این مدل به خوبی می‌تواند مشکلات بارگذاری حجم زیاد داده، محدودیت‌های شرایط ارتباطی و حریم خصوصی داده‌ها را حل کند. نتایج تجربی نشان می‌دهد که مدل پیشنهادی از الگوریتم‌های سنتی تخصیص منابع با بهره‌گیری تنها از یادگیری تقویتی عمیق از سه جنبه‌ی به حداقل رساندن متوسط مصرف انرژی سیستم، به حداقل رساندن تأخیر متوسط خدمات و تعادل تخصیص منابع برتر است.

در پژوهش [۳۸] به بررسی تجمع متمرکز کاربران برای گروه‌بندی کاربران اینترنت اشیا در خوش‌های

منبع	سال انتشار	تاخیر	انرژی	مهلت	فرآیند تصمیم‌گیری مارکوف	یادگیری تقویتی عمیق
[۲۹]	۲۰۱۸	✓	✓			
[۳۰]	۲۰۱۸	✓	✓			
[۳۲]	۲۰۱۹	✓				✓
[۳۳]	۲۰۱۹	✓			✓	
[۳۵]	۲۰۲۰	✓			✓	
[۳۷]	۲۰۲۰	✓			✓	
[۳۸]	۲۰۲۰	✓			✓	
[۳۹]	۲۰۲۰	✓		✓		
[۴۰]	۲۰۲۱	✓				✓

جدول ۳-۱: جمع‌بندی پژوهش‌های مرتبط

مختلف بر اساس اولویت‌های کاربر پرداخته شده است. خوش بala اترین اولویت بارسپاری می‌شود و بر روی سرور لبه اجرا می‌شود، در حالی که خوش بala کمترین اولویت محاسبات را به صورت محلی انجام می‌دهد. برای خوش‌های دیگر، توسعه سیاست‌های جابه‌جایی توزیع شده کاربران از طریق فرآیند تصمیم‌گیری مارکوف مدل‌سازی می‌شود، که در آن هر کاربر اینترنت اشیا به عنوان عاملی در نظر گرفته می‌شود که مجموعه‌ای از تصمیم‌های جابه‌جایی را در محاسبه هزینه‌های سیستم بر اساس پویایی محیط اتخاذ می‌کند. برای پرداختن به نفرین ابعاد، از یک شبکه Q عمیق شامل یک شبکه عصبی عمیق برای تقریب تابع Q در یادگیری Q استفاده شده است.

بنابراین با توجه به پژوهش‌های مرتبط پیشین بررسی شده، می‌توان دریافت که در بسیاری از کارها جهت ارزیابی و بهبود عملکرد سیستم به معیارهای تاخیر و انرژی پرداخته شده است. علاوه بر این در برخی از پژوهش‌ها مشاهده می‌شود که به عنوان یک محدودیت اصلی در سیستم به تعریف وظایف با محدودیت زمان اجرا توجه شده است. از این رو در این پژوهش سعی شده است که معیارهای اصلی عملکرد و محدودیت مهلت زمانی برای وظایف غیر قابل تقسیم در نظر گرفته شود. همچنین با تحقیق و بررسی روش‌های به کار گرفته شده جهت بهبود عملکرد در کارهای پیشین به توجه فراوان به فرایند

تصمیم‌گیری مارکوف به عنوان یک ابزار بسیار قدرتمند جهت مدل‌سازی سیستم و در پی آن روش‌های گوناگون یادگیری تقویتی برای بهینه‌سازی مدل، بر می‌خوریم. این رویکرد در بسیاری از پژوهش‌های اخیر، کارامد و پویا ارزیابی شده است. مقایسه معیارهای مورد بررسی در کارهای پیشین در جدول ۱-۳ ارائه شده است.

فصل ۴

تعریف مسئله

۱-۴ مدل سیستم

ما مدل سیستم را بر اساس مدل ارائه شده در [۳۹]، بر مبنای مجموعه‌ای از گره‌های لبه به شکل مجموعه $N = \{1, 2, \dots, N\}$ و مجموعه‌ای از دستگاه‌های تلفن همراه را به شکل $M = \{1, 2, \dots, M\}$ در یک سیستم محاسبات لبه متحرک در نظر می‌گیریم و جهت گسترش روش ارایه شده به توسعه محیط سیستم و چالش‌های موجود، از جمله تأخیر و مصرف انرژی مطابق مدل انرژی در [۴۰] می‌پردازیم. علاوه بر این، ما بر روی هر قسمت، متعدد از مجموعه‌ای از دوره‌های زمانی $T = \{1, 2, \dots, T\}$ مرکز می‌کنیم، به شکلی که هر دوره زمانی، یک ثانیه طول خواهد کشید. در زیر، ما مدل اجزا موجود در دستگاه‌های تلفن همراه و گره‌های محاسباتی لبه را همانطور که در شکل ۱-۴ نشان داده شده است، ارائه می‌دهیم.

ما بر وظایف محاسباتی دستگاه‌های تلفن همراه مرکز می‌کنیم، به نحوی که هر وظیفه غیرقابل تقسیم بوده و می‌تواند به صورت محلی پردازش شود و یا به یک گره لبه برای پردازش تخلیه شود. ما فرض می‌کنیم که در ابتدای هر دوره زمانی^۱، هر دستگاه تلفن همراه، دارای یک وظیفه جدید، با احتمال وقوع مشخص است. این فرضیه با برخی از آثار موجود سازگار است (به عنوان مثال [۴۱]).

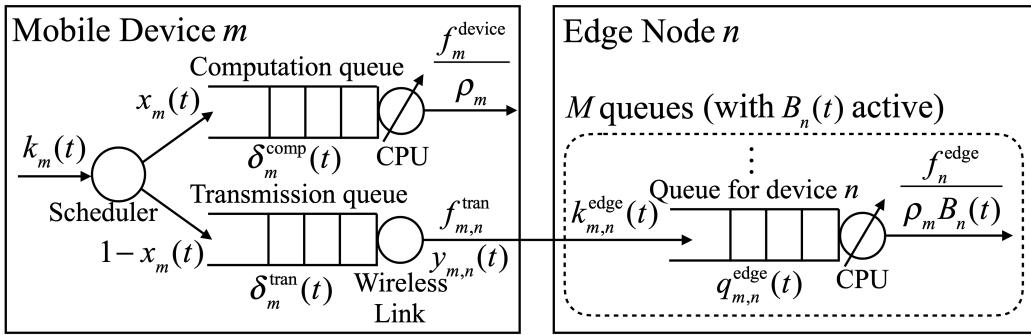
هنگامی که یک دستگاه تلفن همراه دارای یک وظیفه جدید است، ابتدا باید تصمیم بگیرد که

Time Slot^۱

آیا این وظیفه را به صورت محلی انجام دهد و یا به یک گره لبه تخلیه کند. اگر دستگاه تلفن همراه تصمیم به پردازش محاسبات به صورت محلی گیرد، پس از زمان‌بندی (شکل ۱-۴)، آن وظیفه در صفحه محاسبات^۲، برای پردازش محلی قرار می‌گیرد. در غیر این صورت، دستگاه تلفن همراه باید تصمیم بگیرد که محاسبات را به کدامیک از گره‌های لبه تخلیه کند. سپس برنامه زمان‌بندی، محاسبات را به صفحه انتقال^۳ برای تخلیه قرار می‌دهد. سپس محاسبات تخلیه شده به گره لبه انتخاب شده، از طریق لینک بی‌سیم ارسال می‌شود. برای صفحه انتقال فرض می‌کنیم که اگر پردازش (یا انتقال) یک وظیفه در یک واحد زمانی تکمیل شود، وظیفه بعدی موجود در صفحه، در ابتدای واحد زمانی بعد پردازش (یا منتقل) می‌شود. این فرضیه با برخی از آثار موجود، از نظر پویایی صفحه‌بندی در یک سیستم محاسبات لبه متحرک سازگار است. (به عنوان مثال [۴۱]).

هر گره لبه $M, n \in \mathcal{N}$ صفحه را شامل می‌شود، که هر صفحه برای دستگاه تلفن همراه $m \in M$ در نظر گرفته شده است. ما فرض می‌کنیم که بعد از تخلیه وظیفه و دریافت توسط گره لبه در دوره زمانی، آن وظیفه در صفحه محاسبات مربوط به دستگاه تلفن همراه ارسال کننده در گره لبه، قرار می‌گیرد. اگر یک وظیفه از دستگاه تلفن همراه $m \in M$ در صفحه محاسبات آن دستگاه در گره لبه $n \in \mathcal{N}$ و در دوره زمانی $T \in \mathcal{T}$ قرار گیرد، متغیر $K_{m,n}^{\text{edge}}(t) \in \mathbb{Z}_{++}$ را تعریف می‌کنیم که به یک شناسه منحصر به فرد برای آن وظیفه اشاره دارد. به طور مشخص اگر وظیفه $K_m(t')$ برای $t' \in \{1, 2, \dots, t - 1, t\}$ به گره لبه n در دوره زمانی $1 - t$ فرستاده شود، آنگاه $K_{m,n}^{\text{edge}}(t) = K_m(t')$. لازم به ذکر است اگر وظیفه‌ای برای انجام وجود نداشته باشد، در این صورت $K_{m,n}^{\text{edge}}(t) = 0$. همچنین متغیر $\lambda_{m,n}^{\text{edge}}(t) \in \Lambda \cup \{0\}$ (بیت) را تعریف می‌کنیم که به اندازه وظیفه ورودی در صفحه محاسبات دستگاه m در گره لبه n در شروع دوره زمانی t اشاره دارد. در ادامه، ابتدا مدل وظیفه و تصمیم تخلیه وظیفه را ارائه می‌دهیم. سپس، صفحه‌های انتقال و محاسبات را معرفی می‌کنیم.

۲ Computation Queue
۳ Transmission Queue



شکل ۴-۱: تصویر صفحه‌ای تشکیل شده در دستگاه تلفن همراه $m \in \mathcal{M}$ و در گره لبه $n \in \mathcal{N}$.

۲-۴ مدل وظیفه

در ابتدای دوره زمانی $t \in T$ ، اگر دستگاه تلفن همراه $m \in M$ یک وظیفه جدید محاسباتی داشته باشد، ما متغیر $K_m(t) \in \mathbb{Z}_{++}$ را تعریف می‌کنیم، که به شناسه منحصر به فرد برای آن وظیفه اشاره دارد. اگر دستگاه تلفن همراه m در ابتدای دوره زمانی t وظیفه جدیدی در اختیار نداشته باشد، $(K_m(t))$ را برابر با صفر قرار می‌دهیم.

متغیر $(\lambda_m(t))$ (بیت) را به منظور اشاره به اندازه وظیفه ورودی در دوره زمانی t تعریف می‌کنیم. اگر وظیفه جدید $K_m(t)$ در ابتدای دوره زمانی وجود داشته باشد، $\lambda_m(t)$ برابر است با اندازه وظیفه $(K_m(t))$. در غیر این صورت، $\lambda_m(t)$ برابر با صفر خواهد بود.

ما اندازه وظیفه را از میان مقادیر مجموعه گسسته $\{\lambda_1, \lambda_2, \dots, \lambda_{|\Lambda|}\}$ ، همراه با $|\Lambda|$ مقدار فعلی قرار می‌دهیم. از این رو، $\{\lambda_m(t)\} \subseteq \Lambda$. علاوه بر این، وظیفه $(K_m(t))$ نیاز به یک میزان پیچیدگی محاسباتی دارد که با ρ_m (سیکل پردازش پردازنده^۴ برای هر بیت) نمایش داده می‌شود که تعداد سیکل مورد نیاز پردازنده، برای پردازش یک واحد از محاسبات استفاده می‌شود. وظیفه $(K_m(t))$ دارای یک مهلت^۵ انجام است، که با τ_m (در دوره زمانی) نشان داده می‌شود. یعنی اگر کار $(K_m(t))$ باشد، تا پایان واحد زمانی $1 - \tau_m$ به طور کامل پردازش نشده باشد، پس از آن بلافاصله منقضی^۶ می‌شود.

CPU Cycle^۴
Deadline^۵
Drop^۶

۳-۴ تصمیم‌گیری بارسپاری وظیفه

اگر دستگاه تلفن همراه $m \in M$ ، در ابتدای دوره زمانی $T \in \mathcal{T}$ ، وظیفه محاسباتی $K_m(t)$ را در دست داشته باشد، سپس نیاز به تصمیم‌گیری تخلیه برای وظیفه $K_m(t)$ به شرح زیر است.

در ابتدا، متغیر دو حالتی $\{0, 1\} = \{x_m(t)\}$ را که اشاره به تصمیم‌گیری محل پردازش وظیفه (t) دارد، تعریف می‌کنیم، که بیان‌کننده این است که این وظیفه باید به صورت محلی و یا با تخلیه به یک گره لبه پردازش شود. در حالی که تصمیم بر پردازش محلی یا (تخلیه محاسباتی) باشد، $x_m(t)$ را برابر با ۱ (یا ۰) قرار می‌دهیم. در ابتدای دوره زمانی t ، $\lambda_m(t)x_m(t)$ برابر است با تعداد بیت‌های ورودی در صف محاسبات محلی در دستگاه تلفن همراه m ، و همچنین $(1 - x_m(t))\lambda_m(t)$ برابر است با تعداد بیت‌های قرارگرفته در صف انتقال دستگاه m .

اگر وظیفه $K_m(t)$ به یک گره لبه تخلیه شود، متغیر باینتری $\{0, 1\} = \{y_{m,n}(t)\}$ تعریف می‌شود که به محل تخلیه در بین گره‌های لبه اشاره دارد. اگر تصمیم بر تخلیه محاسبات به گره لبه n باشد، متغیر $y_{m,n}(t)$ را برابر با ۱ قرار می‌دهیم، و در غیر این صورت برابر با ۰ خواهد بود. همچنین متغیر $\mathcal{Y}_m(t) = (y_{m,n}(t), n \in \mathcal{N})$ را در اشاره به مجموعه گره‌های لبه تعریف می‌کنیم. شایان ذکر است، بطبق رابطه زیر امکان بارسپاری وظیفه فقط به یک گره از محیط لبه فراهم خواهد بود.

$$\sum_{n \in \mathcal{N}} y_{m,n}(t) = \mathbb{1}(x_m(t) = 1), m \in M, t \in \mathcal{T} \quad (1-4)$$

۴-۴ مدل ارتباطی

همانطور که اشاره شد، دستگاه‌های تلفن همراه و سرورهای لبه از طریق صف انتقال دستگاه‌ها در ارتباط هستند و به شیوه ورود اول خروج اول فعالیت می‌کند. ورودی‌های این صف وظایفی هستند که برای انجام محاسبات مورد نیازشان تصمیم بر بارسپاری آنها شده است. وظایف در انتظار انتقال به گره‌های لبه هستند، که از طریق یک اتصال بی‌سیم در تجهیزات کاربر میسر می‌شود. وظیفه از صف انتقال در دستگاه تلفن همراه به صف محاسبات در گره لبه منتخب منتقل می‌شود. ما یک مدل شبکه بی‌سیم را در نظر می‌گیریم که در آن دستگاه‌های تلفن همراه در کانال‌های متعامد در ارتباط هستند.

در ابتدای دوره زمانی $T \in \mathcal{T}$ ، اگر وظیفه $K_m(t)$ در صفت انتقال برای تخلیه محاسباتی قرار داشته باشد، ما متغیر $\mathcal{T} \in l_m^{tran}(t)$ را تعریف می‌کنیم که به دوره زمانی که وظیفه $K_m(t)$ ارسال و یا منقضی می‌شود، اشاره دارد. اگر صفت انتقال خالی بود و یا $l_m^{tran}(t) = K_m(t) = 0$ بود، آنگاه متغیر $\delta_m^{tran}(t)$ را برای اشاره به تعداد دوره زمانی که وظیفه $K_m(t)$ باید در انتظار ارسال در صفت انتقال m بماند، تعریف می‌کنیم. مقدار متغیر $\delta_m^{tran}(t)$ دستگاه تلفن همراه m قبل از تصمیم‌گیری برای جایگذاری وظیفه در هر یک از صفات، محاسبه می‌کند. متغیر $l_m^{tran}(t')$ را برای $t' < t$ در نظر می‌گیریم، و سپس بر طبق رابطه زیر مقدار $\delta_m^{tran}(t)$ محاسبه می‌شود.

$$\delta_m^{tran}(t) = \left[\max_{t' \in \{0, 1, \dots, t-1\}} l_m^{tran}(t') - t + 1 \right]^+ \quad (2-4)$$

برای سادگی (0) را برابر با صفر در نظر می‌گیریم. اگر دستگاه $m \in \mathcal{M}$ وظیفه $K_m(t)$ را در دوره زمانی $T \in \mathcal{T}$ در صفت انتقال قرار دهد $(0) = x_m(t)$ ، آنگاه وظیفه $K_m(t)$ در دوره زمانی (t) بر طبق رابطه زیر محاسبه می‌شود.

$$l_m^{tran}(t) = \min \left\{ t + \delta_m^{tran}(t) + \lceil D_m^{tran}(t) \rceil - 1, t + \tau_m - 1 \right\} \quad (3-4)$$

در رابطه بالا متغیر $D_m^{tran}(t)$ ، در اشاره به زمان مورد نیاز برای انتقال وظیفه $K_m(t)$ از دستگاه $m \in \mathcal{M}$ به گره لبه $n \in \mathcal{N}$ ، به شکل رابطه زیر تعریف شده است.

$$D_m^{tran}(t) = \sum_{n \in \mathcal{N}} \frac{y_{m,n}(t) \lambda_m(t)}{r_{m,n}^{tran} \Delta} \quad (4-4)$$

از این رو بخش اول عملگر کمینه برابر است با دوره زمانی که وظیفه $K_m(t)$ کاملاً منتقل شده باشد در حالی که مهلتش به پایان نرسد، و بخش دوم نیز به دوره زمانی اشاره دارد که وظیفه $K_m(t)$ منقضی و دور ریخته می‌شود. در نتیجه، $l_m^{tran}(t)$ دوره زمانی را که وظیفه $K_m(t)$ به طور کامل به گره لبه $n \in \mathcal{N}$ منتقل شود را بیان می‌کند.

ما همچنین جهت ارزیابی مصرف انرژی سیستم، متغیر E_m^{tran} را در اشاره به انرژی مصرفی مربوط به انتقال وظیفه $K_m(t)$ از دستگاه $m \in \mathcal{M}$ به گره لبه $n \in \mathcal{N}$ بر طبق رابطه زیر تعریف می‌کنیم.

$$E_m^{tran}(t) = D_m^{tran}(t) \mathcal{P}^{tran} \Delta = \sum_{n \in \mathcal{N}} \frac{y_{m,n}(t) \lambda_m(t)}{r_{m,n}^{tran}} \mathcal{P}^{tran} \quad (5-4)$$

در رابطه بالا مقدار ثابت \mathcal{P}^{tran} را به عنوان نیروی انتقال مصرفی در زمان استفاده کامل از ظرفیت لینک ارتباطی دستگاه $M \in m$ در نظر می‌گیریم.

۵-۴ مدل محاسباتی

۱-۵-۴ محاسبات محلی

همانطور که اشاره شد، صفت محاسبات در دستگاه تلفن همراه محل قرارگیری وظایفی خواهد بود که نتیجه تصمیم‌گیری برای محل پردازش آنها به صورت محلی و با استفاده از منابع موجود در تجهیزات کاربر تعیین شده است. سیاست جایگیری در صفت محاسبات به شکل اولین ورود، اولین خروج^۵ در نظر گرفته شده و ورودی‌ها وظایفی هستند که به صورت محلی پردازش می‌شوند. ما یک پردازنده در جهت محاسبات وظایف موجود در صفت در نظر می‌گیریم. متغیر f_m^{device} (سیکل پردازنده در هر ثانیه) به ظرفیت محاسباتی پردازنده در دستگاه $M \in m$ اشاره می‌کند، و مقداری ثابت خواهد بود.

در شروع دوره زمانی $T \in t$ ، اگر وظیفه $(K_m(t)$ در صفت محاسبات قرار داشته باشد، ما متغیر $l_m^{comp}(t) \in T$ را تعریف می‌کنیم که نشان‌دهنده دوره زمانی خواهد بود که وظیفه $K_m(t)$ انجام و یا دور ریخته شده باشد. اگر در ابتدای دوره زمانی $T \in t$ ، صفت محاسبات خالی باشد و یا $l_m^{comp}(t) = 0$ ، فرار می‌دهیم.

متغیر $\delta_m^{comp}(t)$ (در دوره زمانی) را به منظور اشاره به تعداد دوره زمانی باقی‌مانده تا پردازش برای وظیفه $K_m(t)$ در صفت محاسبات تعریف می‌کنیم. لازم به ذکر است که دستگاه m مقدار متغیر (t) را قبل از تصمیم‌گیری و جایگیری وظیفه در صفت، محاسبه می‌کند. متغیر $l_m^{comp}(t')$ را در $t' > t$ در نظر می‌گیریم، و مطابق با رابطه زیر مقدار $\delta_m^{comp}(t)$ محاسبه می‌شود.

$$\delta_m^{comp}(t) = \left[\max_{t' \in \{0, 1, \dots, t-1\}} l_m^{comp}(t') - t + 1 \right]^+$$
 (۶-۴)

در رابطه بالا عملگر $[z]^+ = \max\{0, z\}$ ، و متغیر $0 = l_m^{comp}(0)$ است. به طور مشخص عبارت $\max_{t' \in \{0, 1, 2, \dots, t-1\}} l_m^{comp}(t')$ First-in First-out^۶

را تا قبل از دوره زمانی t تعیین می‌کند. از این رو، $\delta_m^{comp}(t)$ تعداد دوره زمانی که وظیفه $K_m(t)$ باید در انتظار پردازش باشد را نشان می‌دهد. به عنوان مثال، فرض کنید وظیفه $(1) K_m$ در صفحه محاسبات محلی قرار دارد، و پردازش آن در دوره زمانی ۵ کامل خواهد شد. بنابراین $l_m^{comp} = (1)$ یعنی آن وظیفه باید $3 = \delta_m^{comp}(3) = [\max\{5, 0\} - 3 + 1]^+$ در انتظار اتمام فرآیند، بماند.

اگر دستگاه $m \in \mathcal{M}$ در ابتدای دوره زمانی T ، وظیفه $(K_m(t))$ را در صفحه محاسبات محلی قرار دهد $(1) x_m(t) = 1$ ، آنگاه وظیفه $(K_m(t))$ در دوره زمانی $l_m^{comp}(t)$ پردازش شده و یا از بین رفته است:

$$l_m^{comp}(t) = \min \{t + \delta_m^{comp}(t) + \lceil D_m^{comp}(t) \rceil - 1, t + \tau_m - 1\} \quad (7-4)$$

در رابطه بالا متغیر $D_m^{comp}(t)$ در اشاره به زمان مورد نیاز برای پردازش کامل وظیفه $(K_m(t))$ در دستگاه $m \in \mathcal{M}$ ، مطابق با رابطه زیر تعریف می‌شود.

$$D_m^{comp}(t) = \frac{\lambda_m(t)}{f_m^{device} \Delta / \rho_m} \quad (8-4)$$

به طور مشخص، پردازش وظیفه $(K_m(t))$ در دوره زمانی $t + \delta_m^{comp}(t)$ آغاز خواهد شد. از این رو بخش اول عملگر کمینه برابر است با دوره زمانی که وظیفه $(K_m(t))$ کامل انجام شده باشد در حالی که مهلتش به پایان نرسد، و بخش دوم نیز به دوره زمانی اشاره دارد که وظیفه $(K_m(t))$ منقضی و دور ریخته می‌شود. در نتیجه، $(l_m^{comp}(t))$ دوره زمانی را که وظیفه $(K_m(t))$ به سرانجام خواهد رسید را بیان می‌کند.

همچنین جهت ارزیابی مصرف انرژی سیستم، مقدار ثابت \mathcal{P}^{comp} را به عنوان نیروی مصرفی در حالت استفاده کامل از ظرفیت پردازنده دستگاه $m \in \mathcal{M}$ در نظر می‌گیریم و متغیر $E_m^{loc}(t)$ را در اشاره به انرژی مصرفی مربوط به پردازش کامل وظیفه $(K_m(t))$ در دستگاه $m \in \mathcal{M}$ بطبق رابطه زیر تعریف می‌کنیم.

$$E_m^{loc}(t) = D_m^{comp}(t) \mathcal{P}^{comp} \Delta = \frac{\mathcal{P}^{comp} \lambda_m(t) \rho_m}{f_m^{device}} \quad (9-4)$$

۲-۵-۴ محاسبات در لبه

در اینجا نیز صفحه در ارتباط با یک دستگاه تلفن همراه که در یک گره لبه مستقر است، به شیوه ورود اول خروج اول عمل می‌کند. ورودی صفحه، وظایف تخلیه شده توسط دستگاه تلفن همراه به آن گره

لبه است. متغیر $q_{m,n}^{\text{edge}}(t)$ (بیت) را در اشاره به طول صفت دستگاه تلفن همراه $m \in \mathcal{M}$, در گره لبه $n \in \mathcal{N}$, و در انتهای دوره زمانی $T \in \mathcal{T}$, تعریف می‌کنیم. در میان صفحه‌های گره لبه اگر در ابتدای دوره زمانی t , وظیفه ورودی در صفت دستگاه تلفن همراه m وجود داشته باشد ($\lambda_{m,n}^{\text{edge}}(t) > 0$) و یا وظیفه‌ای از دوره زمانی قبلی باقی مانده باشد ($q_{m,n}^{\text{edge}}(t-1) > 0$), به آن صفت به عنوان یک صفت فعال در آن گره در دوره زمانی t اشاره می‌کنیم. همچنین متغیر $B_n(t)$ را برای اشاره به مجموعه صفحه‌های فعال در گره لبه n در دوره زمانی t را بر طبق رابطه زیر تعریف می‌کنیم.

$$B_n(t) = \{m \mid m \in \mathcal{M}, \lambda_{m,n}^{\text{edge}}(t) > 0 \text{ or } q_{m,n}^{\text{edge}}(t-1) > 0\} \quad (10-4)$$

متغیر $B_n(t)$ را نیز در اشاره به تعداد صفحه‌های فعال در گره لبه $n \in \mathcal{N}$, در دوره زمانی T به شکل $|B_n(t)|$ تعریف می‌کنیم.

در هر دوره زمان $T \in \mathcal{T}$, صفحه‌های فعال در گره لبه $n \in \mathcal{N}$, که در مجموعه $B_n(t)$ قرار دارند، ظرفیت محاسباتی آن گره لبه را با یکدیگر بر اساس روش عمومی اشتراک‌گذاری پردازنده [۴۳] تقسیم می‌کنند. متغیر f_n^{edge} را در اشاره به ظرفیت محاسباتی گره لبه n تعریف می‌شود. از این رو گره لبه n می‌تواند به مقدار $(f_n^{\text{edge}} / (\rho_m B_n(t)))$, ظرفیت محاسباتی به هر دستگاه فعال متصل $m \in B_n(t)$ در دوره زمانی t تخصیص دهد. برای هر صفت در گره لبه، فرض می‌کنیم که اگر پردازش یک وظیفه در یک دوره زمانی تکمیل شود، وظیفه بعدی در صفت، در ابتدای دوره زمانی بعدی پردازش می‌شود. برای محاسبه طول صفت محاسبات دستگاه $m \in \mathcal{M}$ در گره لبه $n \in \mathcal{N}$, متغیر $e_{m,n}^{\text{edge}}(t)$ (بیت) را در اشاره به تعداد وظایف منقضی شده توسط این صفت در پایان دوره زمانی $T \in \mathcal{T}$, بیان می‌کنیم. در نتیجه طول صفت از رابطه زیر به روز خواهد شد.

$$q_{m,n}^{\text{edge}}(t) = \left[q_{m,n}^{\text{edge}}(t-1) + \lambda_{m,n}^{\text{edge}}(t) - \frac{f_n^{\text{edge}}}{\rho_m B_n(t)} \mathbb{1}(m \in B_n(t)) - e_{m,n}^{\text{edge}}(t) \right]^+ \quad (11-4)$$

به طور خاص اندازه صفت $q_{m,n}^{\text{edge}}(t)$ برابر است با طول صفت در انتهای دوره زمانی $(t-1)$ به اضافه تفاوت بین وظایف ورودی و وظایف به پایان رسیده (انجام یا منقضی شده) در دوره زمانی t .

همچنین جهت محاسبه انرژی مصرفی مربوط به وظایف بارسپاری شده، متغیر $D_{m,n}^{\text{edge}}(t)$ را در اشاره

به زمان پردازش وظیفه $K_m(t)$ در گره لبه $n \in \mathcal{N}$ مطابق با رابطه زیر تعریف می‌کنیم.

$$D_{m,n}^{\text{edge}}(t) = \frac{\lambda_{m,n}^{\text{edge}}(t)\rho_m}{\mathcal{B}_n(t)f_n^{\text{edge}}\Delta} \quad (12-4)$$

در پی آن مقدار ثابت $\mathcal{P}^{\text{edge}}$ را برابر با مقدار انرژی مصرفی پردازنده لبه در حالت استفاده کامل از طرفیت در نظر می‌گیریم و متغیر $E_{m,n}^{\text{edge}}(t)$ را در اشاره به انرژی مصرفی پردازش وظیفه $K_m(t)$ در گره لبه $n \in \mathcal{N}$ طبق رابطه (۲۴-۴) تعریف می‌کنیم.

$$E_{m,n}^{\text{edge}}(t) = D_{m,n}^{\text{edge}}(t)\mathcal{B}_n(t)\mathcal{P}^{\text{edge}}\Delta = \frac{\mathcal{P}^{\text{edge}}\lambda_{m,n}^{\text{edge}}(t)\rho_m}{f_n^{\text{edge}}} \quad (13-4)$$

علاوه بر انرژی مصرفی مربوط به پردازش وظیفه در گره لبه، ما انرژی مصرفی رابط کاربری دستگاه را در زمان انتظار برای تکمیل وظیفه در لبه در نظر می‌گیریم. از این رو مقدار ثابت \mathcal{P}^w ، برابر با مقدار مصرف انرژی رابط کاربری در زمان انتظار برای دستگاه M را بیان کرده و متغیر $E_{m,n}^{\text{idle}}(t)$ را در اشاره به مقدار مصرف انرژی مربوط به رابط کاربری دستگاه $m \in \mathbb{M}$ در زمان انتظار برای تکمیل و بازگشت وظیفه $K_m(t)$ از گره لبه بر طبق رابطه (۲۵-۴) تعریف می‌کنیم.

$$E_m^{\text{idle}}(t) = D_{m,n}^{\text{edge}}(t)\mathcal{P}^w\Delta = \frac{\mathcal{P}^w\lambda_{m,n}^{\text{edge}}(t)\rho_m}{\mathcal{B}_n(t)f_n^{\text{edge}}} \quad (14-4)$$

بر این اساس ما برای محاسبه کامل انرژی مصرفی مربوط به بارسپاری وظیفه $K_m(t)$ ، انرژی مصرفی محاسبه شده در (۶-۴)، (۱۴-۴) و (۱۵-۴) را تجمعی می‌کنیم، که شامل انرژی مصرفی برای جابه‌جایی وظیفه از دستگاه M به گره لبه $n \in \mathcal{N}$ به گره لبه $m \in \mathcal{M}$ (با $E_m^{\text{tran}}(t)$)، انرژی مصرفی محاسبات مربوطه در گره لبه $m \in \mathcal{M}$ و انرژی مصرفی رابط کاربری $(E_m^{\text{idle}}(t))$ می‌باشد. از این رو متغیر $E_m^{\text{offl}}(t)$ را در اشاره به انرژی مصرف شده در کل فرآیند بارسپاری وظیفه $K_m(t)$ در زمان $T \in \mathcal{T}$ ، بر طبق رابطه زیر تعریف می‌کنیم.

$$\begin{aligned} E_m^{\text{offl}}(t) &= E_{m,n}^{\text{edge}}(t) + E_m^{\text{trans}}(t) + E_m^{\text{idle}}(t) \\ &= \frac{\mathcal{P}^{\text{edge}}\lambda_{m,n}^{\text{edge}}(t)\rho_m}{f_n^{\text{edge}}} + \frac{\mathcal{P}^{\text{tran}}\lambda_m(t)}{r_{m,n}^{\text{tran}}} + \frac{\mathcal{P}^w\lambda_{m,n}^{\text{edge}}(t)\rho_m}{\mathcal{B}_n(t)f_n^{\text{edge}}} \end{aligned} \quad (15-4)$$

بنابراین ما مقدار انرژی مصرفی در هر لحظه از زمان را برای هر یک از دستگاه‌های هوشمند به شکل $E_m(t) = x_m(t)E_m^{\text{loc}}(t) + (1 - x_m(t))E_m^{\text{offl}}(t)$ زیر محاسبه می‌کنیم.

$$x_m(t) \left(\frac{\mathcal{P}^{comp} \lambda_m(t) \rho_m}{f_m^{device}} \right) + (1 - x_m(t)) \left(\frac{\mathcal{P}^{edge} \lambda_{m,n}^{edge}(t) \rho_m}{f_n^{edge}} + \frac{\mathcal{P}_m \lambda_m(t)}{r_{m,n}^{tran}} + \frac{\mathcal{P}^w \lambda_{m,n}^{edge}(t) \rho_m}{\mathcal{B}_n(t) f_n^{edge}} \right) \\ (16-4)$$

۳-۵-۴ پردازش و انقضا وظیفه

اگر وظیفه $K_{m,n}^{edge}(t)$ از دستگاه تلفن همراه $m \in \mathcal{M}$ در ابتدای دوره زمانی T در صف محاسبات متناظر آن دستگاه، در گره لبه $n \in \mathcal{N}$ قرار داشته باشد، متغیر $T \in \mathcal{T}$ برای اشاره به دوره زمانی که این وظیفه به سرانجام می‌رسد، تعریف می‌شود. با توجه به بار محاسباتی نامعلوم آینده در گره لبه n ، مقدار متغیر $K_{m,n}^{edge}(t)$ تا زمانی که وظیفه $K_{m,n}^{edge}(t)$ به سرانجام رسد، نامشخص خواهد بود. اگر $\hat{l}_{m,n}^{edge}(t) = 0$ باشد، آنگاه $K_{m,n}^{edge}(t) = 0$ خواهد بود. برای تعریف متغیر $\hat{l}_{m,n}^{edge}(t)$ ، متغیر $\hat{l}_{m,n}^{edge}(t)$ را در اشاره به دوره زمانی که پردازش وظیفه $K_{m,n}^{edge}(t)$ آغاز می‌شود، بر طبق رابطه زیر در نظر می‌گیریم.

$$\hat{l}_{m,n}^{edge}(t) = \max \left\{ t, \max_{t' \in \{0, 1, \dots, t-1\}} l_{m,n}^{edge}(t') + 1 \right\} \quad (17-4)$$

به طور مشخص می‌دانیم دوره زمانی که وظیفه $K_{m,n}^{edge}(t)$ آغاز می‌شود، باید از دوره زمانی که وظیفه در صف مستقر می‌شود، نزدیک‌تر باشد. همچنین می‌دانیم دوره زمانی که هر وظیفه وارد سیستم می‌شود باید قبل از زمان پردازش و یا انقضا آن وظیفه باشد. با توجه به تحقق سطوح بار در گره لبه n ، دوره زمانی را بر طبق محدودیت‌های زیر بیان خواهد کرد.

$$\sum_{t'=\hat{l}_{m,n}^{edge}(t)}^{l_{m,n}^{edge}(t)} \frac{f_n^{edge}}{\rho_m B_n(t')} \mathbb{1}(m \in \mathcal{B}_n(t')) \geq \lambda_{m,n}^{edge}(t) \quad (18-4)$$

$$\sum_{t'=\hat{l}_{m,n}^{edge}(t)}^{l_{m,n}^{edge}(t)-1} \frac{f_n^{edge}}{\rho_m B_n(t')} \mathbb{1}(m \in \mathcal{B}_n(t')) < \lambda_{m,n}^{edge}(t) \quad (19-4)$$

فصل ۵

مسئله بارسپاری وظیفه مبتنی بر یادگیری تقویتی عمیق

در این بخش، چالش تخلیه محاسباتی را مورد بررسی قرار می‌دهیم و مدل بارسپاری وظیفه مبتنی بر یادگیری تقویتی عمیق را در سیستم محاسبات لبه متحرک ارائه می‌دهیم.

۱-۵ بیان مسئله با یادگیری تقویتی عمیق

به طور خاص، در ابتدای هر دوره زمانی، هر دستگاه تلفن همراه حالت وضعیت خود را (به عنوان مثال، اندازه وظیفه، اطلاعات صفحه) را مشاهده می‌کند. اگر یک وظیفه جدید تازه وارد شده برای پردازش، وجود داشته باشد، دستگاه تلفن همراه اقداماتی را برای انجام این وظیفه باید انتخاب کند. حالت مشاهده شده و اقدام انتخاب شده باعث ایجاد هزینه (به عنوان مثال، تأخیر وظیفه و مصرف انرژی) برای دستگاه تلفن همراه خواهد شد. هدف هر دستگاه تلفن همراه این است که هزینه‌های طولانی مدت خود را با بهینه‌سازی سیاست تصمیم‌گیری و نقشه‌برداری از مدل سیستم، به حداقل برساند. در زیر، ابتدا حالت، عمل و هزینه را معرفی می‌کنیم، و سپس تابع هزینه را برای هر دستگاه فرموله می‌کنیم.

۱-۱-۵ حالت

در شروع دوره زمانی $t \in \mathcal{T}$ ، هر دستگاه $m \in \mathcal{M}$ اطلاعات حالت جاری شامل اندازه وظیفه ورودی، اطلاعات صفات و اطلاعات تاریخی از سطح بار گرهای لبه را مشاهده می‌کند. به طور خاص، دستگاه تلفن همراه m دارای بردار حالت زیر است:

$$\mathbf{s}_m(t) = \left(\lambda_m(t), \delta_m^{comp}(t), \delta_m^{tran}(t), \mathbf{q}_m^{edge}(t-1), \mathbf{H}(t) \right) \quad (1-5)$$

در رابطه بالا بردار $\mathbf{N} \in \mathcal{N}$ بوده، و ماتریس $\mathbf{H}(t)$ شامل اطلاعات تاریخی از سطح بار (تعداد صفات فعال) گرهای لبه در طول T^{step} (از دوره زمانی $t - T^{step}$ تا دوره زمانی $1 - t$) دوره زمانی گذشته است. همراه با اطلاعات تاریخی از سطح بار محاسباتی گرهای لبه می‌توانیم وضعیت سطح بار آنها را در آینده نزدیک تخمین بزنیم. این اطلاعات، تحت ماتریسی به اندازه $N \times T^{step}$ در نظر گرفته می‌شوند. عبارت $\{\mathbf{H}(t)\}_{i,j}$ را در اشاره به هر یک از مقادیر ماتریس $\{\mathbf{H}(t)\}$ تعریف می‌کنیم، که متناظر با سطح بار محاسباتی هر گره لبه در یک دوره زمانی است. مقادیر ماتریس $\{\mathbf{H}(t)\}$ بر طبق رابطه زیر محاسبه می‌شوند.

$$\{\mathbf{H}(t)\}_{i,j} = B_j(t - T^{step} + i - 1) \quad (2-5)$$

متغیر S را به شکل مجموعه فضای حالت گسته متناهی برای هر دستگاه تلفن همراه تعریف می‌کنیم. مجموعه $S = \Lambda \times \{0, 1, \dots, T\}^{\mathcal{Q}^N} \times \{0, 1, \dots, M\}^{T^{step} \times N}$ است به نحوی که \mathcal{Q} به مجموعه مقادیر فعال برای طول صفت در گره لبه و در طول دوره‌های زمانی اشاره دارد. دستگاه تلفن همراه $m \in \mathcal{M}$ می‌تواند اطلاعات حالت $(\lambda_m(t), \delta_m^{comp}(t), \delta_m^{tran}(t), \mathbf{q}_m^{edge}(t-1))$ را به صورت محلی در ابتدای دوره زمانی t محاسبه کند. همچنین دستگاه تلفن همراه m می‌تواند بردار اطلاعات حالت (1) را از طریق تعداد بیت‌های ارسال شده از دستگاه m به هر گره لبه در هر دوره زمانی و تعداد بیت‌های پردازش و یا منفیت شده در هر دوره زمانی، مطابق با رابطه (8) محاسبه کند. برای ماتریس $\mathbf{H}(t)$ فرض می‌کنیم که هر گره لبه تعداد صفات فعال خود را در انتهای هر دوره زمانی منتشر می‌کند. از آنجا که تعداد صفات فعال همیشه یک عدد کوچک است، و می‌تواند توسط چندین بیت نشان داده شود، انتشار این مقادیر تنها سربار کوچکی بر سیستم تحمیل خواهد کرد.

۲-۱-۵ عمل

در شروع دوره زمانی $T \in \mathcal{T}$ ، اگر دستگاه تلفن همراه $m \in \mathcal{M}$ یک وظیفه جدید ورودی $(K_m(t))$ داشته باشد، در این صورت دستگاه تلفن همراه باید در مورد جایگاه پردازش وظیفه، در موارد زیر تصمیم‌گیری انجام دهد

- محاسبات وظیفه به صورت محلی انجام‌گیرد و یا به گره‌های لبه بارسپاری شود ($x_m(t)$).
- در صورت بارسپاری وظیفه، وظیفه مورد نظر به کدامیک از گره‌های لبه انتقال داده شود ($y_m(t)$).

از این‌رو، عمل در دستگاه تلفن همراه m در دوره زمانی t به صورت بردار زیر تعریف می‌شود:

$$a_m(t) = (x_m(t), y_m(t)) \quad (3-5)$$

همچنین متغیر \mathcal{A} را در اشاره مجموعه فضای عمل تعریف می‌کنیم، $\mathcal{A} = \{0, 1\}^{1+N}$.

۳-۱-۵ هزینه

اگر یک وظیفه به طور کامل پردازش شده باشد، آنگاه تأخیر آن وظیفه، اختلاف زمانی بین ورود و تکمیل محاسبات آن، می‌باشد. متغیر $Delay_m(s_m(t), a_m(t))$ (دوره زمانی) را در اشاره به تأخیر وظیفه، در حالت $s_m(t)$ و با عمل $a_m(t)$ ، بر طبق رابطه زیر تعریف می‌کنیم.

$$:x_m(t) = 1 \text{ اگر } \quad (4-5)$$

$$Delay_m(s_m(t), a_m(t)) = l_m^{comp}(t) - t + 1 \quad (4-5)$$

$$Energy_m(s_m(t), a_m(t)) = E_m^{loc}(t) \quad (5-5)$$

$$:x_m(t) = 0 \text{ اگر } \quad (6-5)$$

$$Delay_m(s_m(t), a_m(t)) = \sum_{n \in \mathcal{N}} \sum_{t' = t}^T \mathbb{1}(k_{m,n}^{edge}(t') = k_m(t)) l_{m,n}^{edge}(t') - t + 1 \quad (6-5)$$

$$Energy_m(s_m(t), a_m(t)) = \sum_{n \in \mathcal{N}} \sum_{t' = t}^T \mathbb{1}(k_{m,n}^{edge}(t') = k_m(t)) E_{m,n}^{offl}(t') \quad (7-5)$$

به طور مشخص، فرض می‌کنیم، وظیفه $K_m(t)$ در ابتدای دوره زمانی t ، وارد سیستم شود. اگر وظیفه $K_m(t)$ در صفت محاسبات محلی قرارگیرد، آنگاه l_m^{comp} دوره زمانی را نشان می‌دهد که وظیفه پردازش می‌شود. همچنین اگر وظیفه $K_m(t)$ در صفت انتقال برای تخلیه محاسباتی قرارگیرد، در این صورت $\sum_{n \in \mathcal{N}} \sum_{t' = t}^T \mathbb{1}(k_{m,n}^{edge}(t') = k_m(t)) l_{m,n}^{edge}(t')$ دوره زمانی را نشان می‌دهد که وظیفه $K_m(t)$ پردازش می‌شود. در این رابطه عبارت $\mathbb{1}(K_{m,n}^{edge}(t') = K_m(t)) = 1$ نشان‌دهنده این است که وظیفه $K_m(t)$ در صفت مربوطه در گره لبه $n \in \mathcal{N}$ در ابتدای دوره زمانی t' قرار دارد، و متغیر $l_{m,n}^{edge}(t')$ دوره زمانی است که وظیفه در صفت دستگاه تلفن همراه m در گره لبه n در ابتدای دوره زمانی t' پردازش می‌شود. تابع هزینه مربوط به وظیفه $K_m(t)$ را در صورتی که به شکل کامل پردازش شود، به شکل زیر تعریف می‌شود:

$$C_m(s_m(t), a_m(t)) = \alpha Delay_m(s_m(t), a_m(t)) + (1 - \alpha) Energy_m(s_m(t), a_m(t)) \quad (8-5)$$

و از طرفی اگر وظیفه $K_m(t)$ منقضی شود، تابع هزینه به صورت زیر خواهد بود:

$$C_m(s_m(t), a_m(t)) = \Omega \quad (9-5)$$

متغیر $\Omega > 0$ را به عنوان یک مقدار ثابت جرمیه برای وظایف از دست رفته در نظر می‌گیریم. اگر وظیفه $C_m(s_m(t), a_m(t)) = K_m(t) = 0$ باشد، در این صورت متغیر Ω قرار می‌دهیم. در ادامه برای سادگی از عبارت $C_m(t)$ در اشاره به $C_m(s_m(t), a_m(t))$ استفاده می‌کنیم.

۲-۵ سیاست بهینه

سیاست دستگاه $M \in \mathcal{M}$ نقشه‌برداری از حالات و اعمال در محیط می‌باشد ($\pi_m : \mathcal{S} \rightarrow \mathcal{A}$). متغیر $\gamma \in [0, 1]$ را که در اشاره به عامل تنزیل^۱ تابع هزینه در آینده بیان می‌شود، تعریف می‌کنیم. هدف

Discount Factor^۱

پیدا کردن سیاست بهینه برای هر دستگاه است، به نحوی که هزینه تأخیر و انرژی طولانی مدت در آن به حداقل برسد.

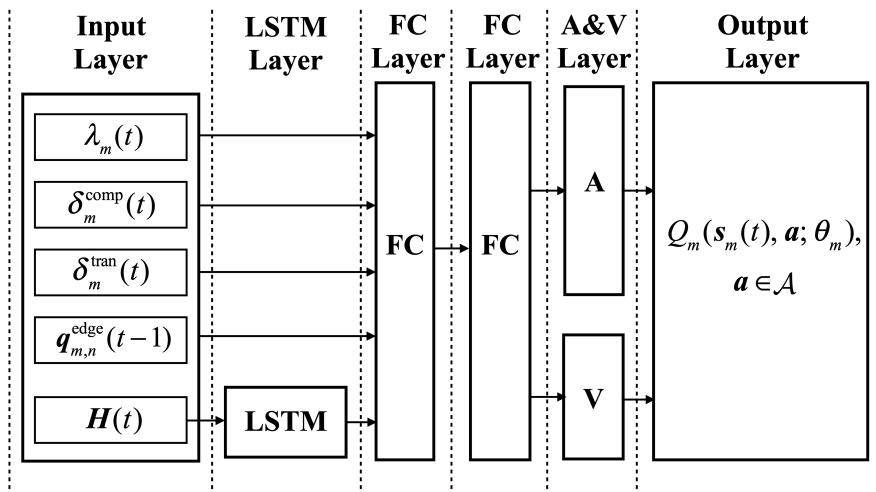
$$\pi_m^* = \arg \min_{\pi_m} \mathbb{E} \left[\sum_{t \in \mathcal{T}} \gamma^{t-1} c_m(t) \middle| \pi_m \right] \quad (10-5)$$

عملگر $\mathbb{E}[\cdot]$ با توجه به پارامترهای سیستمی متغیر زمانی، به عنوان مثال، ورود وظیفه و الزامات محاسباتی وظایف در تمام دستگاه‌های تلفن همراه اشاره دارد.

۳-۵ الگوریتم بارسپاری وظیفه مبتنی بر یادگیری تقویتی عمیق

در این بخش، یک الگوریتم تخلیه مبتنی بر یادگیری تقویتی عمیق را ارائه می‌دهیم که هر دستگاه را قادر به تصمیم‌گیری تخلیه به شکل توزیع شده می‌سازد. این الگوریتم بر اساس یادگیری Q عمیق، سیاست بهینه در تصمیم‌گیری تخلیه را تقریب می‌زنند. از آنجایی که یادگیری Q عمیق یک روش بدون مدل است، الگوریتم پیشنهادی می‌تواند تنظیمات پیچیده سیستم و تعامل بین دستگاه‌های تلفن همراه را بدون دانش قبلی از محیط سیستم و پویایی تعامل، بررسی کند. در همین حال، الگوریتم پیشنهادی می‌تواند فضای حالت بالقوه بزرگ سیستم را مدیریت کند.

در الگوریتم پیشنهادی، هر دستگاه تلفن همراه با هدف یادگیری یک نگاشت از هر جفت عمل و حالت به یک مقدار ارزش Q فعالیت می‌کند، که مشخص کننده هزینه بلندمدت مورد انتظار جفت عمل و حالت است. نگاشت توسط یک شبکه عصبی آموزش داده می‌شود. براساس این نگاشت، هر دستگاه تلفن همراه می‌تواند اقدامی را انتخاب کند تا مقدار Q را در حالت خود القا کند تا هزینه طولانی مدت مورد انتظار خود را به حداقل برساند. در ادامه به ترتیب شبکه‌های عصبی و الگوریتم مبتنی بر یادگیری تقویتی عمیق را ارائه می‌دهیم.



شکل ۱-۵: شبکه عصبی دستگاه تلفن همراه $m \in M$ ، همراه با بردار پارامترهای θ_m ، که نگاشت بین حالت $s_m(t) \in \mathcal{S}$ و مقادیر Q را برای هر عمل $a \in \mathcal{A}$ انجام می‌دهد.

۴-۵ شبکه عصبی

هدف شبکه عصبی یافتن یک نگاشت از هر حالت به مجموعه‌ای از مقادیر Q توسط اقدامات در دسترس است. شکل ۱-۵ تصویری از شبکه عصبی دستگاه تلفن همراه $m \in M$ را نشان می‌دهد. به طور خاص، ما اطلاعات وضعیت سطح بار گره‌های لبه را از طریق یک لایه ورودی به شبکه عصبی ارسال می‌کنیم، سپس، از یک لایه حافظه کوتاه‌مدت ماندگار برای پیش‌بینی سطوح بار (در گره‌های لبه) در آینده نزدیک و بر اساس تاریخچه سطح بار محاسباتی گره‌ها استفاده می‌کنیم. پس از آن، نگاشت از همه حالت‌ها (به جز تاریخچه سطح بار) و سطوح بار پیش‌بینی شده به مقادیر Q از طریق دو لایه شبکه عصبی کاملاً متصل آموزش داده می‌شود. در همین حال، از تکنیک دوئل Q [۴۴] برای بهبود کارایی یادگیری نگاشت از حالت‌ها به مقادیر Q از طریق یک لایه امتیاز-عمل^۲ و ارزش^۳ بهره می‌گیریم.

در نهایت مقادیر Q عملکردها در لایه خروجی تعیین می‌شوند، متغیر θ_m در اشاره به بردار پارامترهای شبکه عصبی دستگاه m تعریف می‌کنیم، که شامل وزن تمام اتصالات و بایاس تمام نوروون‌ها، از لایه ورودی تا لایه خروجی می‌باشد. جزئیات هر لایه به شرح زیر است:

^۲ Advantage
^۳ Value

۵-۵ لایه ورودی

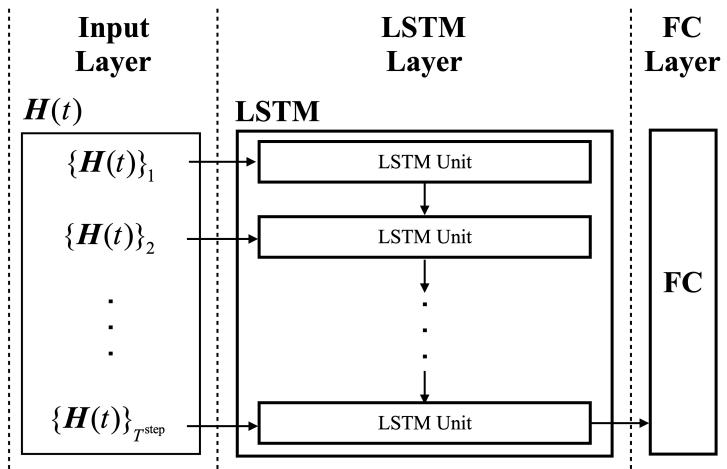
این لایه وظیفه دارد که حالت را به عنوان ورودی گرفته و آن را به لایه‌های بعدی ارسال کند. به عنوان مثال در دستگاه $M \in m$ ، اطلاعات محیطی $q_m^{edge}(t), \delta_m^{tran}(t), \lambda_m(t)$ و $(1 - \delta_m^{comp}(t))$ به لایه شبکه کاملاً متصل ارسال می‌شود و $\mathbf{H}^{(t)}$ نیز به لایه حافظه کوتاه مدت ماندگار انتقال داده می‌شود.

۵-۶ لایه حافظه کوتاه مدت ماندگار

این لایه مسئول یادگیری پویایی سطوح بار در گره‌های لبه و پیش‌بینی سطوح بار در آینده نزدیک است. این امر با بهره‌گیری از یک شبکه حافظه کوتاه مدت ماندگار^۴ بدست می‌آید، که رویکردی پرکاربرد برای یادگیری وابستگی‌های سری زمانی در مشاهدات متوالی و پیش‌بینی تغییرات آینده نزدیک است.

به طور خاص، شبکه حافظه کوتاه مدت ماندگار، ماتریس $\mathbf{H}^{(t)}$ را به عنوان ورودی دریافت می‌کند تا پویایی سطح بار را یاد بگیرد. شکل ۲-۵ ساختار شبکه حافظه کوتاه مدت ماندگار را نشان می‌دهد. این شبکه شامل T^{step} واحد حافظه است. هر واحد حافظه یک بردار از ماتریس $\mathbf{H}^{(t)}$ را به عنوان ورودی دریافت می‌کند، همانطور که در شکل ۳ مشاهده می‌شود، متغیر i در ماتریس $\mathbf{H}^{(t)}$ به بردار i^{th} در ماتریس $\mathbf{H}^{(t)}$ اشاره دارد. این واحدهای حافظه به ترتیب از $\{\mathbf{H}^{(t)}\}_{T^{step}}$ تا $\{\mathbf{H}^{(t)}\}_1$ به یکدیگر متصل می‌شوند تا تغییرات توالی‌ها را به یاد آورند، که می‌تواند تغییرات سطوح بار را در گره‌های لبه در میان شکاف‌های زمانی آشکار کند. شبکه حافظه کوتاه مدت ماندگار اطلاعاتی را که پویایی سطوح بار محاسباتی را در آینده نزدیک نشان می‌دهند در آخرین واحد حافظه، به عنوان خروجی می‌دهد، که این خروجی برای یادگیری بیشتر به نورون‌ها در لایه بعدی متصل می‌شود.

^۴ Long Short Term Memory



شکل ۵-۵: شبکه حافظه کوتاه‌مدت ماندگار با T^{step} واحد حافظه.

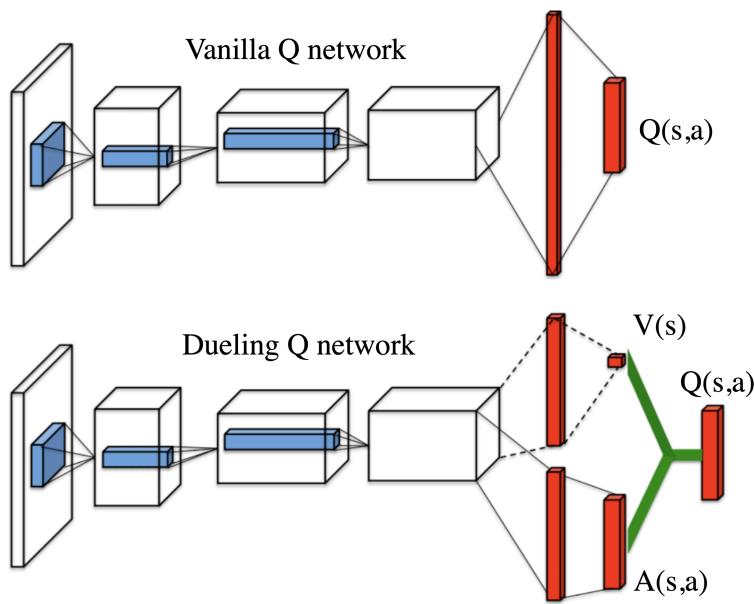
۷-۵ لایه‌های کاملاً متصل

دو لایه کاملاً متصل مسئول یادگیری نگاشت از حالت و پویایی سطح بار محاسباتی آموخته شده به مقادیر Q اقدامات هستند. هر لایه شبکه کاملاً متصل شامل مجموعه‌ای از نورون‌ها با واحد خطی اصلاح شده^۵ است. برای اولین لایه کاملاً متصل، ورودی هر نورون به نورون‌های لایه ورودی مربوط به همه حالت‌ها (به جز ماتریس \mathbf{H}) و شبکه حافظه کوتاه‌مدت ماندگار متصل می‌شود. خروجی هر نورون به هر یک از نورون‌های لایه دوم کاملاً متصل، متصل می‌شود. برای لایه دوم کاملاً متصل، خروجی هر نورون به هر یک از نورون‌های لایه ارزش و مزیت متصل می‌شود.

۸-۵ لایه ارزش، امتیاز-عمل و لایه خروجی

لایه مزیت و ارزش، نقش خروجی تکنیک شبکه Q عمیق دوئل^۶ [۴۴] را پیاده‌سازی می‌کند، و مقدار Q هر عمل را به عنوان خروجی تعیین می‌کند. ایده اصلی شبکه دوئل Q عمیق این است که ابتدا یک ارزش حالت (بخشی از Q ناشی از حالات) و مقادیر مزیت عمل (بخشی از مقدار Q ناشی از اقدامات) را به طور جداگانه یاد بگیریم و سپس برای تعیین مقادیر Q جفت‌های عمل حالت، از مقادیر حالت و

ReLU^۶
Duelling Q Network^۹



شکل ۳-۵: معماری شبکه تک جریانی Q معمولی (بالا) و معماری شبکه دوثفل Q مورد استفاده (پایین)

مزیت عمل استفاده کنیم. این تکنیک می‌تواند تخمین مقادیر کیو را از طریق ارزیابی جداگانه هزینه‌های بلندمدت مورد انتظار ناشی از یک وضعیت و یک اقدام، بهبود بخشد.

لایه مزیت و ارزش شامل دو شبکه است که با دو شبکه جداگانه تحت عنوان مزیت و ارزش نشان داده می‌شود (شکل ۳-۵ را ببینید). شبکه مزیت شامل یک شبکه کاملاً متصل با مجموعه‌ای از نورون‌ها است، که مسئول یادگیری مقدار مزیت عمل برای هر عمل $a \in A$ در هر حالت است.

در هر دستگاه تلفن همراه $m \in M$ ، متغیر $A_m(s_m(t), a; \theta_m)$ در جهت اشاره به ارزش مزیت عمل عمل a در حالت $s \in S$ همراه با بردار مقادیر θ_m شبکه، تعریف می‌کنیم. شبکه ارزش حاوی یک شبکه کاملاً متصل با مجموعه‌ای از نورون‌ها است، که به عنوان مسئول یادگیری ارزش حالت عمل می‌کند. برای هر دستگاه تلفن همراه m ، متغیر $V_m(s_m(t); \theta_m)$ به عنوان ارزش حالت در حالت $s_m(t)$ همراه با بردار وزنی شبکه θ_m ، تعریف می‌کنیم. مقادیر $V_m(s_m(t); \theta_m)$ و $A_m(s_m(t); \theta_m)$ توسط پارامترهای بردار وزنی θ_m و ساختار شبکه عصبی از لایه ورودی تا لایه ارزش مزیت، تعیین می‌شود، به شکلی که بردار θ_m قابل تنظیم است و در الگوریتم مبتنی بر یادگیری تقویتی عمیق آموخته داده می‌شود.

بر اساس ساختار لایه ارزش مزیت، برای دستگاه تلفن همراه $M \in M$ ، مقدار Q حاصل از عمل

$a \in \mathcal{A}$ تحت حالت $s_m(t) \in \mathcal{S}$ در لایه خروجی به صورت زیر محاسبه می‌شود:

$$Q_m(s_m(t), a; \theta_m) = V_m(s_m(t); \theta_m) + \left(A_m(s_m(t), a; \theta_m) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} (A_m(s_m(t), a'; \theta_m)) \right) \quad (11-5)$$

که مجموع مقادیر ارزش حالت، تحت حالت مربوطه و مزیت عمل، طبق عمل مربوطه است (یعنی تفاوت بین ارزش مزیت عمل مربوطه نسبت به میانگین همه اقدامات).

به طور خلاصه، از لایه ورودی تا لایه خروجی، شبکه عصبی دستگاه تلفن همراه $m \in \mathcal{M}$ با بردار پارامتر θ_m یک نگاشت از جفت عملکرد حالت و مقادیر Q را تشکیل می‌دهد (یعنی تحت هر حالت مشاهده شده $s_m(t) \in \mathcal{S}$ ، یک مقدار Q برای هر اقدام $a \in \mathcal{A}$ وجود دارد که با $(Q_m(s_m(t), a; \theta_m))$ نشان داده می‌شود)، که مشخص کننده هزینه طولانی مدت مورد انتظار در حالت مشاهده شده و هر یک از اقدامات در فضای عمل است.

۹-۵ الگوریتم مبتنی بر یادگیری تقویتی عمیق

در الگوریتم مبتنی بر یادگیری تقویتی عمیق پیشنهادی، ما از گره‌های لبه برای کمک به دستگاه‌های تلفن همراه استفاده می‌کنیم، تا در جهت کاهش بارهای محاسباتی در دستگاه‌های تلفن همراه، شبکه عصبی را آموزش دهنده. به طور خاص برای هر دستگاه تلفن همراه $m \in \mathcal{M}$ ، یک گره لبه $n_m \in \mathcal{N}$ وجود دارد که به دستگاه m در آموزش شبکه کمک می‌کند. این گره لبه می‌تواند گره با بیشترین ظرفیت محاسباتی در نظر گرفته شود. برای راحتی در ارائه، متغیر $\mathcal{M}_n \subset \mathcal{M}$ را در اشاره به مجموعه دستگاه‌های تلفن همراه که از گره لبه $n \in \mathcal{N}$ در نظر می‌گیریم، به شکلی که $\mathcal{M}_n = \{m \in \mathcal{M} | n_m = n\}$.

لازم به توجه است که منطقی است که از گره‌های لبه مستقیماً در جهت آموزش شبکه بهره‌گیریم، به این علت که تبادل اطلاعات درگیر در آموزش (شامل اطلاعات وضعیت و پارامترهای شبکه عصبی) محدود است و علاوه بر این، ظرفیت محاسباتی مورد نیاز آموزش در هر دوره زمانی بسیار کمتر از وظایف دستگاه‌های تلفن همراه می‌باشد.

الگوریتم مبتنی بر یادگیری تقویتی عمیق در دستگاه تلفن همراه $m \in \mathcal{M}$ و گره لبه $n \in \mathcal{N}$ بر

طبق الگوریتم ۱ و ۲ اجرا می‌شود. ایده کلیدی این الگوریتم آموزش شبکه عصبی با استفاده از تجربه دستگاه تلفن همراه (به عنوان مثال، حالت، عمل، هزینه و حالت بعدی)، برای به دست آوردن نگاشتی از جفت عمل حالت به مقادیر Q است، که براساس آن دستگاه می‌تواند اقدامی را انتخاب کند که منجر به کمینه کردن مقدار Q در حالت مشاهده شده شود، تا هزینه طولانی مدت مورد انتظار آن به حداقل برسد.

در الگوریتم مبتنی بر یادگیری تقویتی عمیق، در گره لبه $\mathcal{N} \in \mathcal{N}$ ، یک حافظه پخش مجدد D_m و دو شبکه عصبی برای دستگاه متصل $m \in \mathcal{M}_n$ نگه داشته می‌شود. حافظه بازپخش D_m تجربه مشاهده شده $(s_m(t), a_m(t), c_m(t), s_m(t+1))$ دستگاه تلفن همراه m را برای دوره زمانی T ذخیره می‌کند، و این تجربه در حافظه بازپخش برای آموزش شبکه‌های عصبی استفاده می‌شود. همچنین دو شبکه عصبی شامل شبکه هدف ($Target_{Net_m}$) و شبکه ارزیابی ($Eval_{Net_m}$) است و مقادیر Q توسط $Q_m^{Target}(s_m(t), a; \theta_m^{Target})$ و $Q_m^{Eval}(s_m(t), a; \theta_m^{Eval})$ و عمل $s_m(t) \in \mathcal{S}$ تحت مشاهده حالت $a \in \mathcal{A}$ بیان می‌شوند. توجه داشته باشید که $Target_{Net_m}$ و $Eval_{Net_m}$ ساختار شبکه عصبی مشابهی دارند، در حالی که آنها به ترتیب بردارهای پارامتر شبکه θ_m^{Eval} و θ_m^{Target} را دارند. شبکه ارزیابی برای انتخاب عمل و شبکه هدف برای مشخص کردن یک مقدار هدف Q استفاده می‌شود، که هزینه طولانی مدت مورد انتظار یک اقدام تحت وضعیت مشاهده شده را تقریب می‌زند. این مقدار هدف Q برای به روز رسانی بردار پارامتر شبکه θ_m^{Eval} در $Eval_{Net_m}$ با به حداقل رساندن اختلاف بین مقادیر Q در $Eval_{Net_m}$ و مقادیر Q هدف استفاده می‌شود. مقداردهی اولیه حافظه بازپخش D_m و دو شبکه عصبی در مراحل ۱ تا ۳ در الگوریتم ۲ آورده شده است. در ادامه، الگوریتم مبتنی بر یادگیری تقویتی عمیق را به ترتیب در دستگاه تلفن همراه $m \in \mathcal{M}$ و گره لبه $\mathcal{N} \in \mathcal{N}$ ارائه می‌کنیم.

۱-۹-۵ الگوریتم اجرایی در دستگاه تلفن همراه

متغیر $Episodes$ را در اشاره به تعداد قسمت‌ها تعریف می‌کنیم. در ابتدای هر قسمت، دستگاه تلفن همراه $m \in \mathcal{M}$ ، وضعیت حالت را مطابق زیر مقداردهی اولیه می‌کند.

$$\mathbf{s}_m(1) = (\lambda_m(1), \delta_m^{comp}(1), \delta_m^{tran}(1), \mathbf{q}_m^{edge}(0), \mathbf{H}(1)) \quad (12-5)$$

متغیر (0) $q_{m,n}^{edge}$ را برای همه گره‌های لبه برابر با صفر قرار می‌دهیم و همچنین مقادیر ماتریس (1) \mathbf{H}

Algorithm 1 DRL-based Algorithm at Device $m \in \mathcal{M}$

```

1: for episode from 1 to #_of_Episodes do
2:   Initialize  $s_m(1)$ ;
3:   for time slot  $t \in \mathcal{T}$  do
4:     if device  $m$  has a new task arrival  $k_m(t)$  then
5:       Send a parameter_request to edge node  $n_m$ ;
6:       Receive network parameter vector  $\theta_m^{\text{Eval}}$ ;
7:       Select an action  $a_m(t)$  according to (21);
8:     end if
9:     Observe the next state  $s_m(t + 1)$ ;
10:    Observe a set of costs  $\{c_m(t'), t' \in \tilde{\mathcal{T}}_{m,t}\}$ ;
11:    for each task  $k_m(t')$  with  $t' \in \tilde{\mathcal{T}}_{m,t}$  do
12:      Send  $(s_m(t'), a_m(t'), c_m(t'), s_m(t' + 1))$  to  $n_m$ ;
13:    end for
14:   end for
15: end for

```

را نیز در ابتدا برابر با صفر قرار می‌دهیم. هر قسمت شامل مجموعه‌ای از دوره‌های زمانی \mathcal{T} است. در شروع دوره زمانی $T \in \mathcal{T}$ ، اگر دستگاه تلفن همراه m یک وظیفه ورودی $K_m(t)$ داشته باشد، آنگاه درخواست پارامتر (parameter_request) را به گره لبه n_m ارسال می‌کند. پس از دریافت درخواست توسط گره لبه، بردار پارامتر θ_m^{Eval} شبکه Eval-Net_m برای دستگاه ارسال می‌شود. سپس دستگاه m بر طبق رابطه زیر برای وظیفه $K_m(t)$ ، به انتخاب عمل می‌پردازد.

$$a_m(t) = \begin{cases} \arg \min_{a \in \mathcal{A}} Q_m^{\text{Eval}}(s_m(t), a; \theta_m^{\text{Eval}}), & \text{با احتمال } \epsilon \\ \text{انتخاب عمل به صورت تصادفی}, & \text{با احتمال } 1 - \epsilon \end{cases}$$

در رابطه بالا، ϵ احتمال اکتشاف تصادفی را نشان می‌دهد. مقدار $Q_m^{\text{Eval}}(s_m(t), a; \theta_m^{\text{Eval}})$ برابر است با مقدار Q تحت پارامترهای جاری θ_m^{Eval} از شبکه عصبی Eval-Net_m . به طور شهودی، با احتمال $\epsilon - 1$ ، دستگاه تلفن همراه عملی را انتخاب می‌کند که با حداقل مقدار Q در حالت مشاهده شده $s_m(t)$ ، بر اساس Eval-Net_m مطابقت دارد.

در ابتدای دوره زمانی بعدی (یعنی دوره زمانی $t + 1$)، دستگاه تلفن همراه m حالت بعدی $(t + 1)$ را مشاهده می‌کند. از سوی دیگر، از آنجایی که پردازش و انتقال یک وظیفه ممکن است برای چندین دوره زمانی ادامه‌یابد، هزینه $c_m(t)$ که به تأخیر وظیفه $K_m(t)$ بستگی دارد، ممکن است در ابتدای دوره

زمانی جدید مشاهده نشود.

همچنین دستگاه تلفن همراه m ممکن است، مجموعه‌ای از هزینه‌های مربوط به برخی وظایف گذشته متعلق به وظیفه $K_m(t)$ را برای دوره زمانی $t \in \mathcal{T}$ مشاهده کند. در جهت رفع این مشکل، برای دستگاه $\tilde{\mathcal{T}}_{m,t} \subset \mathcal{T}$ ، m را به عنوان مجموعه‌ای از دوره‌های زمانی تعریف می‌کنیم، به شکلی که هر وظیفه $\tilde{\mathcal{T}}_{m,t}$ در دوره زمانی $t' \in \tilde{\mathcal{T}}_{m,t}$ پردازش شده یا در دوره زمانی t ، منقضی شده است. مجموعه $\tilde{\mathcal{T}}_{m,t}$ در زیر تعریف شده است.

$$\tilde{\mathcal{T}}_{m,t} = \left\{ t' \mid t' = 1, 2, \dots, t, \lambda_m(t') > 0, x_m(t') l_m^{comp}(t') + 1 - x_m(t') \sum_{n \in \mathcal{N}} \sum_{i=t'}^t \mathbb{1}(K_{m,n}^{edge}(i) = K_m(t')) l_{m,n}^{edge}(i) = t \right\} \quad (13-5)$$

در رابطه بالا، > 0 به این معنی است که وظیفه ورودی جدید $K_m(t')$ در دوره زمانی t' وجود دارد. به طور خاص مجموعه $\tilde{\mathcal{T}}_{m,t}$ شامل دوره‌های زمانی $\{1, 2, \dots, t\}$ است. اگر وظیفه $K_m(t')$ در دوره زمانی t تکمیل شود، در ابتدای دوره زمانی $t+1$ ، دستگاه تلفن همراه m مجموعه هزینه‌های $\{c_m(t'), t' \in \tilde{\mathcal{T}}_{m,t}\}$ را مشاهده می‌کند. سپس دستگاه m برای هر وظیفه $K_m(t')$ در دوره زمانی $t' \in \tilde{\mathcal{T}}_{m,t}$ مشاهده خود را به گره لبه n_m ارسال می‌کند.

۲-۹-۵ الگوریتم اجرایی در گره لبه

در ادامه پس از مقداردهی اولیه حافظه پخش مجدد D_m و همچنین شبکه‌های عصبی $Eval_Net_m$ و $Target_Net_m$ برای دستگاه $m \in \mathcal{M}$ ، گره لبه $n \in \mathcal{N}$ منتظر پیام‌های درخواست از دستگاه‌های تلفن همراه در مجموعه M_n می‌شود.

اگر گره لبه $n \in \mathcal{N}$ درخواست پارامتر را از دستگاه تلفن همراه $m \in \mathcal{M}$ دریافت کند، آنگاه بردار پارامتر فعلی $Eval_Net_m$ را به دستگاه m ارسال خواهد کرد. از سوی دیگر، اگر گره لبه n ، تجربه $(s_m(t), a_m(t), c_m(t), s_m(t+1))$ را از دستگاه تلفن همراه $m \in \mathcal{M}_n$ دریافت کند، تجربه را در حافظه ذخیره می‌کند. حافظه به صورت اولین ورود، اولین خروج کار می‌کند.

Algorithm 2 DRL-Based Algorithm at Edge Node $n \in \mathcal{N}$

```

1: Initialize replay memory  $D_m$  for each device  $m \in \mathcal{M}_n$  and
   set Count := 0;
2: Initialize  $\text{Eval\_Net}_m$  with random parameter  $\theta_m^{\text{Eval}}$  for each
   device  $m \in \mathcal{M}_n$ ;
3: Initialize  $\text{Target\_Net}_m$  with random parameter  $\theta_m^{\text{Target}}$  for
   each device  $m \in \mathcal{M}_n$ ;
4: while True do
5:   if receive a parameter_request from device  $m \in \mathcal{M}_n$  then
6:     Send  $\theta_m^{\text{Eval}}$  to device  $m$ ;
7:   end if
8:   if receive an experience  $(\mathbf{s}_m(t), \mathbf{a}_m(t), c_m(t), \mathbf{s}_m(t+1))$ 
   from device  $m \in \mathcal{M}_n$  then
9:     Store  $(\mathbf{s}_m(t), \mathbf{a}_m(t), c_m(t), \mathbf{s}_m(t+1))$  in  $D_m$ ;
10:    Sample a set of experiences (denoted by  $\mathcal{I}$ ) from  $D_m$ ;
11:    for each experience  $i \in \mathcal{I}$  do
12:      Obtain experience  $(\mathbf{s}_m(i), \mathbf{a}_m(i), c_m(i), \mathbf{s}_m(i+1))$ ;
13:      Compute  $\hat{Q}_{m,i}^{\text{Target}}$  according to (24);
14:    end for
15:    Set vector  $\hat{Q}_m^{\text{Target}} := (\hat{Q}_{m,i}^{\text{Target}}, i \in \mathcal{I})$ ;
16:    Update  $\theta_m^{\text{Eval}}$  to minimize  $L(\theta_m^{\text{Eval}}, \hat{Q}_m^{\text{Target}})$  in (23);
17:    Count := Count + 1;
18:    if mod(Count, Replace_Threshold) = 0 then
19:       $\theta_m^{\text{Target}} := \theta_m^{\text{Eval}}$ ;
20:    end if
21:  end if
22: end while

```

توجه داشته باشید که ما نیازی به همگام‌سازی بین دستگاه تلفن همراه m و گره لبه مرتبط با آن n_m نداریم. گره لبه شبکه عصبی را (در مراحل ۱۰ تا ۲۰ در الگوریتم ۲) آموزش می‌دهد تا بردار پارامتر θ_m^{Eval} از Eval_Net_m را به صورت زیر بهروزرساند.

گره لبه به صورت تصادفی مجموعه‌ای از تجربیات را از حافظه نمونه‌برداری می‌کند (در مرحله ۱۰) که با \mathcal{I} نشان داده شده است. بر اساس این نمونه‌های تجربه، ایده اصلی بهروزرسانی Eval_Net_m کمینه‌کردن تفاوت بین مقادیر Q در Eval_Net_m و مقادیر \hat{Q} هدف براساس نمونه‌های تجربه تحت شبکه هدف Target_Net_m است.

به طور خاص برای یک تجربه نمونه در مجموعه \mathcal{I} ، گره لبه $\hat{Q}_m^{\text{Target}} = (\hat{Q}_{m,i}^{\text{Target}}, i \in \mathcal{I})$ را

محاسبه کرده و پارامتر θ_m^{Eval} در شبکه $Eval_Net_m$ برای کمینه کردنتابع زیان زیر به روزرسانی می کند.

$$L(\theta_m^{Eval}, \hat{Q}_m^{Target}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left(\hat{Q}_m^{Eval}(s_m(i), a_m(i); \theta_m^{Eval}) - \hat{Q}_m^{Target} \right)^2 \quad (14-5)$$

در رابطه بالا $|\mathcal{I}|$ کار دینالیتی مجموعه \mathcal{I} است. تابع زیان، شکاف بین مقدار Q با عمل $a_m(i)$ در حالت (i) در تحت بردار شبکه \hat{Q}_m^{Target} و مقدار Q هدف $\hat{Q}_{m,i}^{Target}$ را برای هر تجربه در مجموعه \mathcal{I} مشخص می کند. کمینه سازی تابع زیان با انجام یک مرحله گرادیان نزولی بر روی شبکه عصبی $Eval_Net_m$ با استفاده از انتشار پسانداز انجام می شود (به بخش ۶ در [۴۵] مراجعه کنید).

مقدار \hat{Q}_m^{Target} برای هر تجربه $i \in \mathcal{I}$ ، بر اساس تکنیک شبکه Q عمیق دوگانه [۴۴] تعیین می شود. تکنیک شبکه Q عمیق دوگانه می تواند تخمین هزینه بلندمدت مورد انتظار را در مقایسه با روش های سنتی بهبود ببخشد. مقدار \hat{Q}_m^{Target} برای تجربه i ، مجموع هزینه متناظر در تجربه i و مقدار Q تنزیل شده عملی است که احتمالاً با توجه به وضعیت بعدی در تجربه i تحت شبکه هدف $Target_Net_m$ به شکل زیر انتخاب می شود.

$$\hat{Q}_{m,i}^{Target} = c_m(i) + \gamma Q_m^{Target}(s_m(i+1)), a_i^{Next}; \theta_m^{Target}) \quad (15-5)$$

در رابطه بالا مقدار a_i^{Next} نشان دهنده عملی است که کمترین مقدار Q در حالت $(i+1)$ ، تحت شبکه $Target_Net_m$ را به همراه خواهد داشت.

$$a_i^{Next} = \arg \min_{a \in \mathcal{A}} Q_m^{Eval}(s_m(i+1), a; \theta_m^{Eval}) \quad (16-5)$$

به طور شهودی، برای تجربه i مقدار $\hat{Q}_{m,i}^{Target}$ هدف Q هدف \hat{Q}_m^{Target} ، هزینه طولانی مدت مورد انتظار عمل $a_m(i)$ در حالت (i) را نشان می دهد. این مقدار خلاصه ای از هزینه واقعی ثبت شده در تجربه i ، یعنی $c_m(i)$ و همچنین هزینه تقریبی طولانی مدت انتظار آینده بر اساس شبکه هدف $Target_Net_m$ یعنی $\gamma Q_m^{Target}(s_m(i+1)), a_i^{Next}; \theta_m^{Target})$ است.

برای هر به روزرسانی $Eval_Net_m$ با کپی برداری از شبکه $Target_Net_m$ ، شبکه $Replace_Threshold$ توسط بردار پارامتر $\theta_m^{Eval} = \theta_m^{Target}$ به روزرسانی می شود.

هدف از این مرحله، به روز نگه داشتن پارامتر شبکه θ_m^{Target} در $Target_Net_m$ است، به نحوی که بهتر می تواند هزینه طولانی مدت مورد انتظار را در محاسبه مقادیر هدف Q تخمین بزند.

فصل ۶

نتایج

در این فصل ابتدا جزیيات سیستم شبیه‌سازی شده را شرح می‌دهیم و در ادامه به تحلیل نتایج به دست آمده با توجه به معیارهای اصلی عملرد می‌پردازیم.

۱-۶ شبیه‌سازی

در این بخش نحوه پیاده‌سازی روش مبتنی بر یادگیری تقویتی عمیق پیشنهادی را شرح می‌دهیم. در جهت ارزیابی محیط مورد بررسی همانطور که در فصل ۴ اشاره شد، از شبیه‌سازی رخداد گستته استفاده می‌کنیم. در این شبیه‌سازی، نسبت وظایف منقضی شده به تعداد کل وظایف، میانگین تأخیر در وظایف تکمیل شده و انرژی مصرفی در کل سیستم را به عنوان سه معیار اصلی عملکرد در نظر می‌گیریم. و همچنین برای مقایسه عملکرد الگوریتم پیشنهادی، از سه روش معیار در سناریوهای چندگانه استفاده می‌کنیم.

همانطور که در تنظیمات پارامترهای اصلی در جدول ۱-۶ آورده شده است، سناریویی را با ۵۰ دستگاه تلفن همراه و ۵ گره لبه، شبیه‌سازی می‌کنیم. هر شکاف زمانی در شبیه‌سازی را برابر با ۰/۱ ثانیه در نظر می‌گیریم و محیط را برای ۱۰۰ شکاف زمانی، معادل با ۱۰ ثانیه از زمان کار سیستم واقعی شبیه‌سازی می‌کنیم. در هر شکاف زمانی هر یک از دستگاههای تلفن همراه با احتمال ۰/۳ وظیفه جدید

پارامتر	مقدار
M	۵۰
N	۵
Δ	۰/۱ ثانیه
$f_m^{device}, m \in \mathcal{M}$	۲/۵ GHz
$f_n^{edge}, n \in \mathcal{N}$	۴۱/۸ GHz
$f_m^{tran}, m \in \mathcal{M}, n \in \mathcal{N}$	۱۴ Mbps
$\lambda_m(t), m \in \mathcal{M}, t \in \mathcal{T}$	{۲/۰, ۲/۱, ..., ۵/۰} Mbits
$\rho_m, m \in \mathcal{M}$	۰/۲۹۷ gigacycles per Mbits
$\tau_m, m \in \mathcal{M}$	(۱۰ × Δ) واحد زمانی
\mathcal{P}^{tran}	۰/۲W
\mathcal{P}^{edge}	۵W
\mathcal{P}^{comp}	۱/۵W
\mathcal{P}^w	۰/۰۳W

جدول ۶-۱: تنظیمات پارامترهای اصلی

وروودی دریافت می‌کنند، سپس باید تصمیم بگیرند که پردازش مربوط به وظیفه ورودی را خود بر عهده بگیرند و یا به گره لبه بارسپاری کنند. در حالتی که تصمیم بر بارسپاری باشد، دستگاه هوشمند می‌بایست گره لبه مناسب را با در نظر گرفتن پویایی سطح بار موجود انتخاب کرده و وظیفه ورودی را از طریق لینک ارتباطی موجود بارسپاری نماید.

همچنین تنظیمات شبکه عصبی با اندازه دسته برابر با ۱۶ تنظیم شده است. نرخ یادگیری برابر با ۰/۰۰۱ و ضریب تنزیل برابر با ۰/۹ در نظر گرفته شده است و از طرفی احتمال کاوش تصادفی به تدریج از ۰ به ۱ کاهش می‌یابد. در این شبیه سازی‌ها ما بر روی یک سناریو با محیط ثابت تمرکز می‌کنیم،تابع انتقال (از حالت و عمل به حالت بعدی) و تابع هزینه (از حالت و عمل به هزینه) در طول زمان متغیر نیستند. در یک محیط غیر ثابت، اگر محیط تغییر کرده باشد، الگوریتم پیشنهادی می‌تواند با تنظیم

مجدداً احتمال اكتشاف تصادفی، به نحوی که اكتشاف تصادفی را دوباره فعال کند، با آن سازگار شود.

شبکه عصبی در الگوریتم پیشنهادی به صورت آنلاین آموزش داده می‌شود، به نحوی که تجربیات جمع‌آوری شده به صورت بی‌درنگ برای آموزش شبکه عصبی و بهروزرسانی تصمیم‌گیری بارسپاری وظیفه استفاده می‌شود. این شبکه با بهره‌گیری از یادگیری ژرف و تقریب توابع می‌تواند مشکل رایج در این نوع از یادگیری یعنی نفرین ابعاد را مرتفع نماید. همچنین بهره‌گیری از تکنیک‌های شبکه یادگیری تقویتی \mathcal{Q} دوئل و دوگانه می‌تواند در یادگیری سیاست بهینه و سرعت همگرایی، بهبود قابل توجهی ایجاد کند. در راستای بررسی عملکرد الگوریتم پیشنهادی، همگرایی یادگیری را تحت فرآپارامترهای مختلف شبکه عصبی و تنظیمات الگوریتم ارزیابی می‌کنیم.

۲-۶ ارزیابی عملکرد

در این بخش، به نحوه ارزیابی عملکرد روش ارایه شده پرداخته و در نهایت با تحلیل نتایج حاصل از شبیه‌سازی در شرایط متغیر به مقایسه عملکرد سیستم با سایر روش‌ها می‌پردازیم.

در راستای بررسی عملکرد شبکه یادگیری تقویتی عمیق پیشنهادی، همگرایی یادگیری را تحت نرخ‌های یادگیری مختلف شبکه عصبی و تنظیمات تناوب به روز رسانی شبکه ارزیابی می‌کنیم. جهت ارزیابی عملکرد الگوریتم پیشنهادی با توجه به معیارهای اصلی عملکرد، نتایج حاصل از روش ارایه شده را در مقایسه با روش‌های معیار می‌سنجیم. همانطور که اشاره شد معیارهای عملکرد مورد توجه در این پژوهش شامل نرخ وظایف منقضی شده، میانگین تأخیر در وظایف تکمیل شده و انرژی مصرفی در کل سیستم می‌باشند، که در مقایسه الگوریتم پیشنهادی با روش‌های معیار استفاده می‌شوند. از این رو با توجه به محیط مورد بررسی و سناریوهای شبیه‌سازی شده، روش‌های مورد بررسی، به شکل زیر در نظر گرفته می‌شوند.

- **روش محاسبات محلی:** این روش تنظیماتی از سیستم را نشان می‌دهد که در آن همه دستگاه‌های تلفن همراه موجود در سناریوی شبیه‌سازی شده، سعی می‌کنند تمامی وظایف خود را به صورت محلی و تحت حداقل تأخیر قابل تحمل اجرا کنند. این روش در بخش مقایسه نتایج به اختصار به شکل (No-Offl) نشان داده شده است.

• روش بارسپاری تصادفی: این روش تنظیماتی از سیستم را نشان می‌دهد که همه دستگاه‌های تلفن همراه متصل در شبیه‌سازی، وظایف خود را به صورت تصادفی در هر مرحله به یکی از گره‌های لبه در دسترس بارسپاری می‌کنند. در این حال کل منابع ارتباطی و محاسباتی گره لبه به طور مساوی به هر دستگاه‌های تلفن همراه اختصاص داده می‌شود. این روش در بخش مقایسه نتایج، به اختصار به شکل (Rand-Offl) نشان داده شده است.

• روش بارسپاری وظیفه برخط در سطح کاربر: روش بارسپاری وظیفه ارایه شده در [۳۰]، نسخه‌ای توسعه یافته جهت بارسپاری وظایف برخط در سطح کاربر است، به نحوی که می‌تواند سناریوهای را با چند گره لبه، برای استقرار وظایف در نظر بگیرد. این روش بر اساس برآورد ظرفیت منابع با توجه به مشاهدات تاریخی طراحی شده است، که می‌تواند با بهینه‌سازی تصمیمات بارسپاری وظیفه به بهبود عملکرد سیستم دست‌یابد. این روش را جهت مقایسه نتایج حاصل از الگوریتم پیشنهادی انتخاب می‌کنیم، زیرا بسیار مشابه پژوهش ما بوده، و وظایف را به شکلی غیرقابل تقسیم و با مهلت زمانی در نظر می‌گیرد. همچنین بدون دخالت هیچ موجودیت مرکزی سیستم را بررسی می‌نماید. این روش در بخش مقایسه نتایج به اختصار به شکل (ULOOF) نشان داده شده است.

• روش بارسپاری وظیفه مبتنی بر یادگیری Q عمیق پیشنهادی: روش پیشنهادی با بهره‌گیری از یادگیری تقویتی، به تعیین محل انجام محاسبات مربوط به هر وظیفه، در هر یک از دستگاه‌های هوشمند می‌پردازد. از این رو با هدف به حداقل رساندن معیارهای اصلی، می‌تواند با بهینه‌سازی هزینه سیستم به ایجاد تعادلی در برآورد نیازهای دستگاه‌های متصل دست‌یابد. در این روش الگوریتم پیشنهادی با استفاده از شبکه Q عمیق دوئل، و تکنیک‌های شبکه Q عمیق دوگانه می‌تواند از نفرین ابعاد حالات اجتناب کند و با بهبود هزینه‌های طولانی مدت و بهبود تصمیمات پیاپی، نیازهای وظایف ورودی را بر طرف نماید. روش پیشنهادی در بخش مقایسه نتایج به اختصار به شکل (P-DDQN) نشان داده شده است.

جهت آموزش شبکه یادگیری تقویتی عمیق، الگوریتم پیشنهادی را در ۱۰۰۰ قسمت اجرا کرده و برای هر قسمت ۱۰۰ شکاف زمانی در نظر می‌گیریم، سپس در انتهای هر قسمت مقادیر هزینه و معیارهای

اصلی عملکرد را ثبت نموده و در ادامه به تحلیل و مقایسه نتایج می‌پردازیم.

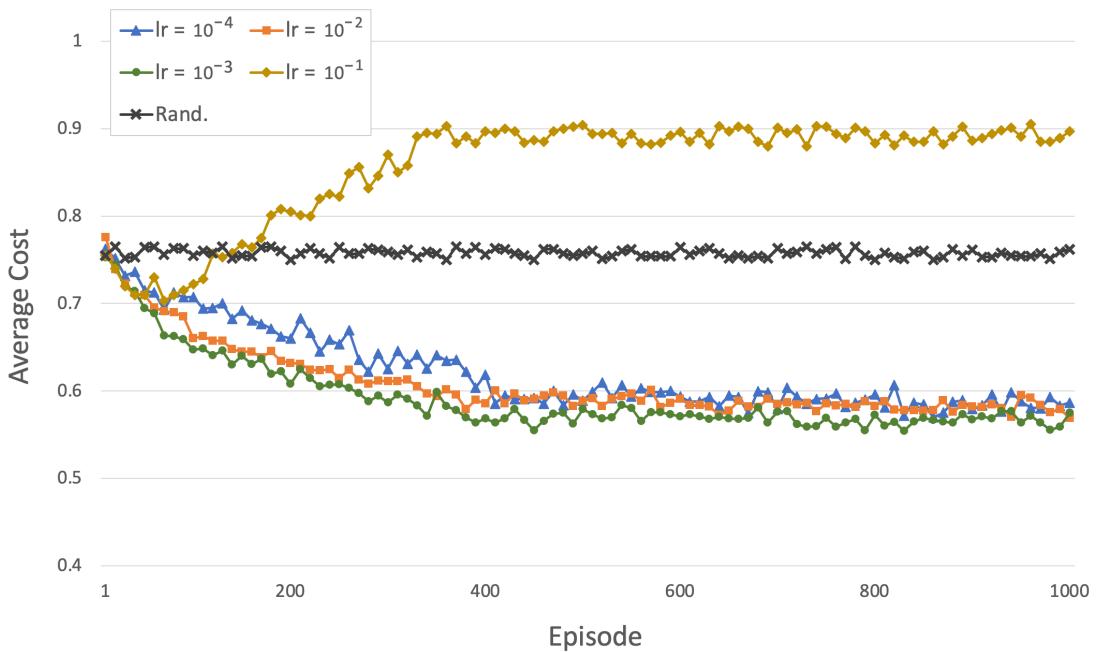
۳-۶ مقایسه نتایج

در اولین مرحله از نتایج شبیه‌سازی به بررسی رفتار تابع هزینه سیستم مطابق (۵-۸) در طی یادگیری در هر قسمت می‌پردازیم، هزینه سیستم برای هر یک از دستگاه‌های هوشمند برابر با قرینه قدر مطلق مجموع وزن‌داری از مقادیر عملکردی‌های اصلی تعریف می‌شود، که شامل نرخ وظایف منقضی شده، میانگین تأخیر وظایف تکمیل شده و انرژی مصرفی کل سیستم در شکاف‌های زمانی سیستم می‌باشد. از آنجا که انرژی مصرفی در تابع هزینه هر دستگاه شامل انرژی کل سیستم از جمله انرژی انتقال وظایف، انرژی محاسبات در تجهیزات کاربر و محاسبات در گره‌های لبه می‌باشد، شایان ذکر است با توجه به اینکه دریافت مقادیر انرژی مصرفی محاسبه شده مربوط به محاسبات وظایف هر دستگاه از گره‌های لبه بار زیادی به سیستم اعمال می‌کند، از این رو این انرژی مصرفی مطابق با (۴-۱۳) در خود دستگاه محاسبه می‌شود.

نتایج حاصل از شبیه‌سازی، به صورت میانگین مقادیر هزینه همه دستگاه‌های هوشمند در طی یادگیری در شکل ۶-۱ نشان داده شده است. محور افقی نمودار، قسمت را نشان داده و محور عمودی میانگین هزینه دستگاه‌های تلفن همراه و کل شکاف‌های زمانی در هر قسمت را نشان می‌دهد. ما عملکرد الگوریتم پیشنهادی را تحت تنظیمات مختلف و خط مشی تصادفی (که با Rand مشخص می‌شود)، رسم می‌کنیم، که در آن اقدامات به شکل تصادفی انتخاب می‌شوند.

شکل ۶-۱ همگرایی الگوریتم پیشنهادی را تحت مقادیر مختلف نرخ یادگیری نشان می‌دهد، که با (lr) مشخص می‌شود، و بیانگر اندازه گام در هر تکرار برای حرکت به سمت حداقل تابع ضرر است. در حالتی که نرخ یادگیری برابر با (10^{-3}) در نظر گرفته شود، همگرایی نسبتاً سریعی را شاهد هستیم. اما وقتی نرخ یادگیری کوچک‌تر (یعنی 10^{-4}) باشد، همگرایی با کندی همراه خواهد بود. همچنین هنگامی که نرخ یادگیری بزرگ‌تر (یعنی 10^{-2} و 10^{-1}) باشد، هزینه افزایش می‌یابد، که ممکن است حتی بیشتر از سیاست تصادفی باشد.

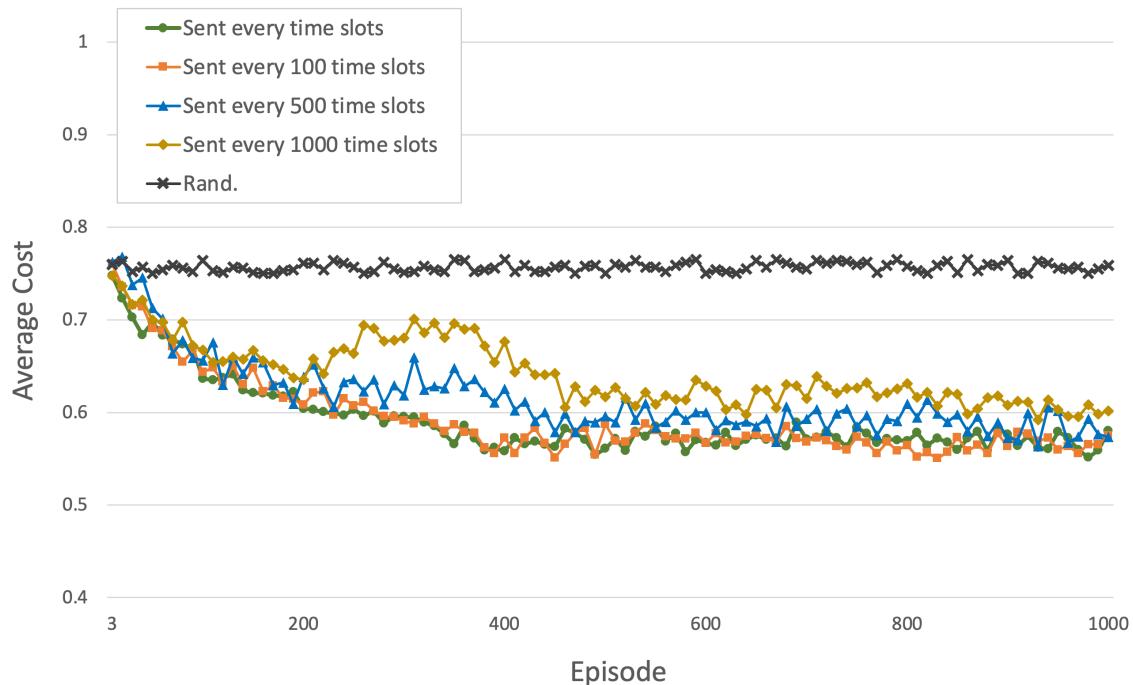
همانطور که در بخش‌های قبلی بیان شد، از آنجا که برای تسهیل در یادگیری، دستگاه‌های هوشمند



شکل ۱-۶: همگرایی الگوریتم پیشنهادی تحت تغییرات نرخ یادگیری.

شبکه‌های عصبی خود را در گره‌های لبه آموزش می‌دهند، برای به روز رسانی وزن‌های شبکه عصبی خود، درخواست بردار پارامتر خواهند داشت. این به روز رسانی می‌باشد در دوره‌های ثابت از شکاف‌های زمانی انجام شود. در شکل ۲-۶، تنظیماتی را در نظر می‌گیریم که در آن یک دستگاه تلفن همراه درخواست پارامتر را برای به روز رسانی پارامتر شبکه خود در تعداد مشخصی از شکاف‌های زمانی ارسال می‌کند (به عنوان مثال به ازای هر شکاف زمانی که دستگاه تلفن همراه یک وظیفه دریافت می‌کند). همانطور که در نمودار نشان داده شده است، ارسال درخواست در هر ۱۰۰ شکاف زمانی تاثیر زیادی بر عملکرد الگوریتم در مقایسه با ارسال در هر شکاف زمانی ندارد.

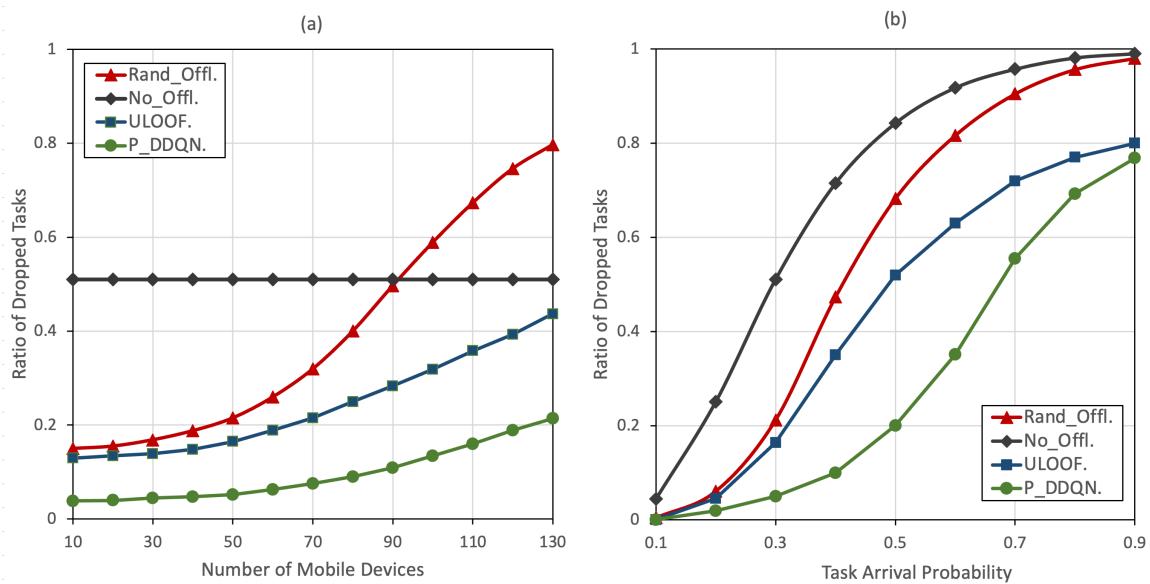
این به این دلیل است که آموزش شبکه عصبی در الگوریتم پیشنهادی بر اساس نمونه‌برداری تصادفی تجربیات در صفحه بازپخش است و تجربیات تازه به دست آمده تاثیر زیادی ایجاد نخواهند کرد. از این رو، الگوریتم می‌تواند درجه معینی از تأخیر را از نظر به روز رسانی شبکه عصبی برای انتخاب عمل تحمل کند. در نتیجه، به منظور کاهش سربار ارتباط، می‌توانیم فرکانس ارسال درخواست پارامتر را بدون تاثیر قابل توجهی بر عملکرد الگوریتم کاهش دهیم.



شکل ۲-۶: همگرایی الگوریتم پیشنهادی تحت تغییرات تناوب درخواست پارامتر به روز رسانی.

همانطور که اشاره شد، جهت ارزیابی عملکرد روش مبتنی بر یادگیری تقویتی عمیق پیشنهادی (که با P-DDQN مشخص می‌شود)، با سه روش از جمله بدون بارسپاری (که با No-Offl مشخص می‌شود)، بارسپاری تصادفی (که با Rand-Offl مشخص می‌شود) و روش بارسپاری مبتنی بر چارچوب برخط سطح کاربر (که با ULOOF مشخص می‌شود) مطابق با [۳۹]، در سه معیار اصلی عملکرد از جمله نرخ وظایف منقضی شده (یعنی نسبت تعداد وظایف منقضی شده به تعداد کل وظایف واردشده به سیستم)، میانگین تأخیر (یعنی متوسط تأخیر وظایفی که پردازش شده‌اند) و میزان مصرف انرژی در کل سیستم مقایسه می‌کنیم.

با توجه به معیارهای اصلی مورد ارزیابی در سیستم، تنظیمات سیستم را تحت تغییرات در تعداد دستگاه‌های متصل و تغییرات در احتمال ورود وظیفه در هر شکاف زمانی به آزمایش می‌گذاریم. در شکل ۳-۶، نمودار (a) رفتار نرخ وظایف منقضی شده را در مقایسه با تعداد دستگاه‌های متصل نشان داده، و نمودار (b) رفتار نرخ وظایف منقضی شده در مقایسه با احتمال ورود وظیفه در هر شکاف زمانی را نمایش می‌دهد.

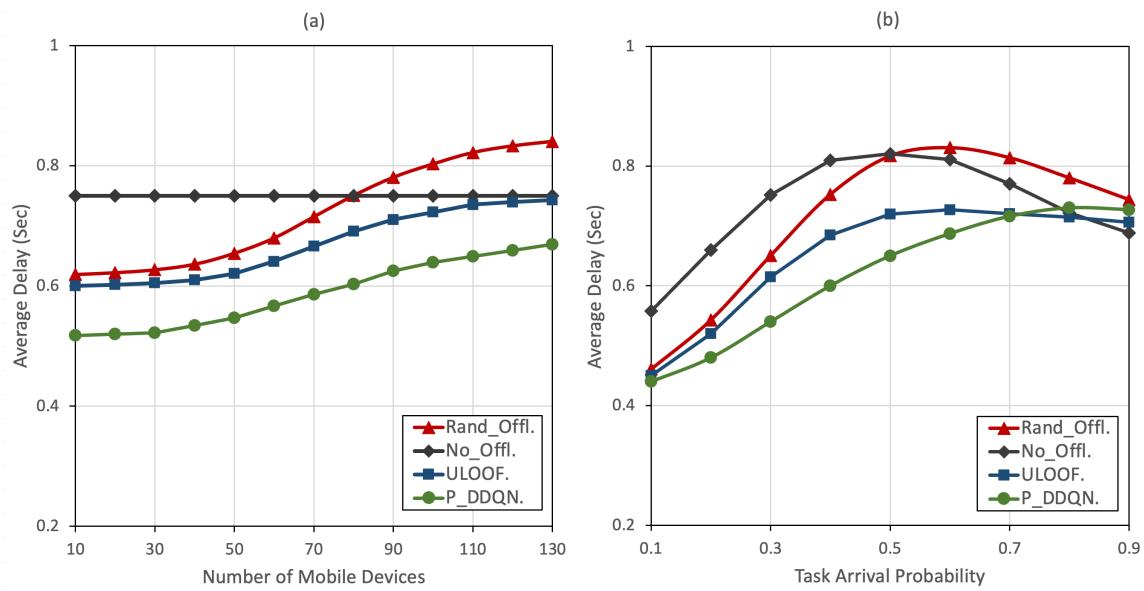


شکل ۶-۳: مقایسه عملکرد در نرخ وظایف منقضی شده تحت: (a) تغییرات تعداد دستگاههای متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی.

در شکل ۶-۳، الگوریتم پیشنهادی به نرخ پایین‌تری در وظایف منقضی شده نسبت به روش‌های دیگر دست می‌یابد، به خصوص زمانی که تعداد دستگاهها زیاد باشد. این به این دلیل است که الگوریتم پیشنهادی می‌تواند به طور موثر، بار ناشناخته را در گره‌های لبه بررسی کند. هنگامی که تعداد دستگاههای تلفن همراه به ۸۰ دستگاه افزایش می‌یابد، الگوریتم پیشنهادی نرخ وظایف منقضی شده را کمتر از ۰/۱ حفظ می‌کند. در نمودار (a) با افزایش تعداد دستگاههای تلفن همراه، میانگین تأخیر هر روش (به جز بدون بارسپاری) به دلیل افزایش بالقوه بار در گره‌های لبه افزایش می‌یابد.

همچنین در نمودار (b) با افزایش احتمال ورود وظیفه، الگوریتم مبتنی بر یادگیری تقویتی عمیق پیشنهادی در همه احتمالات ورودی می‌تواند نسبت کمتری از وظایف منقضی شده را در مقایسه با روش‌های معیار حفظ کند. هنگامی که احتمال ورود وظیفه کوچک است (به عنوان مثال ۰/۱) اکثر روش‌ها می‌توانند نسبت وظایف حذف شده را در حدود صفر به دست آورند. با افزایش احتمال ورود وظیفه از ۰/۱ به ۰/۵، نسبت وظایف منقضی شده در الگوریتم پیشنهادی کمتر از ۰/۲ باقی می‌ماند، در حالی که روش‌های معیار به بیش از ۰/۵ افزایش می‌یابد.

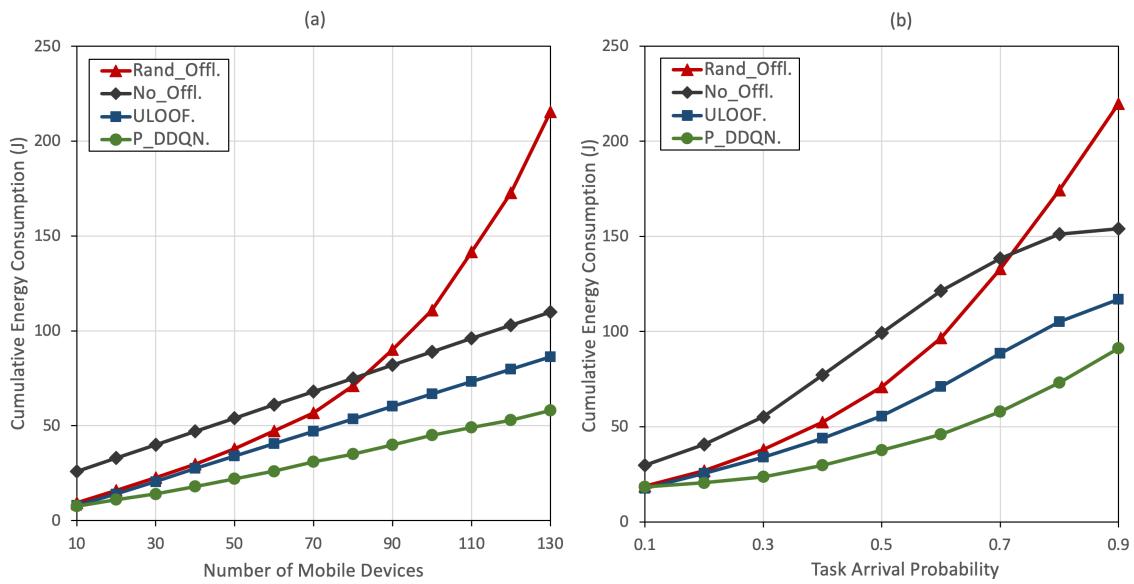
در شکل ۶-۴، نمودار (a) رفتار میانگین تأخیر را در مقایسه با تعداد دستگاههای متصل و نمودار



شکل ۶-۴: مقایسه عملکرد در میانگین تأخیر تحت: (a) تغییرات تعداد دستگاه‌های متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی.

(b) رفتار میانگین تأخیر را در مقایسه با احتمال ورود وظیفه در هر شکاف زمانی، نمایش می‌دهد. در نمودار (a) با افزایش تعداد دستگاه‌های تلفن همراه، میانگین تأخیر هر روش (به جز بدون بارسپاری) به دلیل افزایش بالقوه بار در گره‌های لب افزایش می‌یابد. از آنجایی که الگوریتم پیشنهادی می‌تواند به طور موثر با پویایی بار لب ناشناخته مقابله کند، وقتی تعداد دستگاه‌های تلفن همراه به ۱۳۰ دستگاه افزایش می‌یابد، می‌تواند به میانگین تأخیر کمتری از روش‌های معیار دست‌یابد.

در نمودار (b) نیز با افزایش احتمال ورود وظیفه از ۰/۱ به ۰/۵، میانگین تأخیر الگوریتم مبتنی بر یادگیری تقویتی عمیق پیشنهادی حدود ۲۰ درصد افزایش می‌یابد، در حالی که روش‌های معیار حداقل ۳۲ درصد افزایش می‌یابند. این نشان می‌دهد در حالی که بار سیستم افزایش می‌یابد، میانگین تأخیر الگوریتم پیشنهادی به طور چشمگیری کمتر از روش‌های معیار افزایش می‌یابد. همانطور که احتمال رسیدن کار به ۰/۶ افزایش می‌یابد، میانگین تأخیر برخی از روش‌ها کاهش می‌یابد، زیرا تعداد فزاینده‌ای از وظایف منقضی می‌شوند و بنابراین در تأخیر متوسط محاسبه نمی‌شوند. به همین دلیل، زمانی که بار سیستم زیاد است، الگوریتم پیشنهادی ممکن است تأخیر متوسط بیشتری نسبت به روش‌های دیگر داشته باشد، زیرا وظایف کمتری منقضی می‌شوند.



شکل ۶-۵: مقایسه عملکرد در مقدار انرژی مصرفی سیستم تحت: (a) تغییرات تعداد دستگاه‌های متصل و (b) تغییرات مقدار احتمال ورود وظیفه در هر شکاف زمانی.

در شکل ۶-۵، نمودار (a) میزان مصرف انرژی کلی سیستم را در مقایسه با تعداد دستگاه‌های متصل و نمودار (b) میزان مصرف انرژی کلی سیستم را در مقایسه با احتمال ورود وظیفه در هر شکاف زمانی، نمایش می‌دهد.

در نمودار (a) با افزایش تعداد دستگاه‌های تلفن همراه و در پی آن افزایش وظایف در سیستم، میزان مصرف انرژی در هر روش افزایش می‌یابد. با گسترش تعداد دستگاه‌های متصل از ۱۰ به ۱۳۰، میزان مصرف انرژی در روش بدون بارسپاری حدود ۸۰ واحد و در روش بارسپاری تصادفی حدود ۲۱۰ واحد، افزایش می‌یابد. در ابتدا با وجود کمتر از ۸۰ دستگاه متصل، میزان مصرف انرژی سیستم در روش بدون بارسپاری، بیشتر از روش بارسپاری تصادفی می‌باشد، این در حالی است که با وجود تعداد بیشتری از دستگاهها در سیستم، میزان مصرف انرژی در روش بدون بارسپاری کمتر از روش بارسپاری تصادفی می‌باشد، این به این خاطر است که افزایش بار در سیستم سبب افزایش زمان محاسبات در گره‌های لبه و در نتیجه زمان انتظاری بیشتر در رابط کاربری تجهیزات کاربر خواهد بود، که این افزایش زمان انتظار سبب افزایش میزان انرژی مصرفی در سیستم می‌شود. از این رو از آنجایی که الگوریتم پیشنهادی می‌تواند به طور موثر با پویایی بار در سیستم تطبیق یابد، می‌تواند در مواجه با افزایش تعداد دستگاه‌های تلفن

همراه، مصرف انرژی کمتری را از روش‌های معیار کسب کند.

همچنین در نمودار (b) با افزایش احتمال ورود وظایف، میزان مصرف انرژی در هر روش افزایش می‌یابد. با افزایش حتمال ورود وظیفه از $0/1$ به $0/9$ ، میزان مصرف انرژی در روش بدون بارسپاری حدود ۱۲۵ واحد و در روش بارسپاری تصادفی حدود ۲۱۰ واحد، افزایش می‌یابد. در احتمالات ورود کمتر از $0/7$ ، میزان مصرف انرژی سیستم در روش بدون بارسپاری، بیشتر از روش بارسپاری تصادفی می‌باشد، این در حالی است که با احتمال ورود بیشتر، میزان مصرف انرژی در روش بدون بارسپاری کمتر از روش بارسپاری تصادفی می‌باشد.

از آنجایی که افزایش تعداد دستگاه‌های متصل و افزایش احتمال ورود وظیفه، هر دو با افزایش تعداد وظایف همراه هستند در نتیجه رفتار نمودار (b) در روش بارسپاری تصادفی مشابه رفتار آن در نمودار (a) خواهد بود، اما در روش بدون بارسپاری، نرخ رشد انرژی مصرفی با افزایش احتمالات ورود، روند کاهشی خواهد یافت. این به این جهت است که ظرفیت محاسباتی تجهیزات کاربر اشباع خواهد شد و تعداد وظایف منقضی شده افزایش خواهد یافت. در اینجا نیز الگوریتم‌های مبتنی بر تصمیم‌گیری می‌توانند به طور موثر با پویایی بار در سیستم تطبیق یابند و در مواجه با افزایش احتمال ورود وظیفه در دستگاه‌های تلفن همراه مقابله نمایند. از این رو می‌توانند مصرف انرژی بهینه‌تری را از روش‌های معیار داشته باشند. از این رو الگوریتم یادگیری عمیق پیشنهادی توانسته است به کاهش مصرف انرژی در کل سیستم دست یابد.

فصل ۷

نتیجه‌گیری

در این پژوهش به بررسی چالش بارسپاری وظایف در یک سیستم محاسباتی لبه پرداخته و با هدف به حداقل رساندن میانگین تأخیر و مصرف انرژی در طولانی‌مدت یک الگوریتم بارسپاری توزیع شده ارائه می‌دهیم، که دستگاه‌های تلفن همراه را قادر می‌سازد تا تصمیم‌گیری بارسپاری وظایف خود را به شیوه‌ای غیر مرکزی انجام دهد، در حالی که می‌توانند پویایی سطح بار ناشناخته را در گره‌های لبه در نظر گرفته و به بهینه‌سازی کارایی سیستم در مواجهه با وظایف غیرقابل تقسیم و حساس به تأخیر بپردازد. و این در حالی است که محدودیت تأخیر و ظرفیت محدود منابع نیز در نظر گرفته شده است.

نتایج شبیه سازی نشان می‌دهد که در مقایسه با چندین روش موجود، الگوریتم پیشنهادی ما می‌تواند نسبت کارهای منقضی شده، میانگین تاخیر و میزان مصرف انرژی را کاهش دهد. این مزیت به ویژه زمانی قابل توجه است که وظایف حساس به تاخیر باشند، یا سطوح بار در گره‌های لبه بالا باشد.

در جهت بهبود کارایی سیستم به گسترش یک مسئله بهینه‌سازی در قالب یک مشکل کمینه‌سازی تأخیر، مصرف انرژی و نرخ منقضی شدن وظایف، با وجود محدودیت تأخیر می‌پردازیم، از این رو با توسعه روش ارایه شده در پژوهش [۳۹] با استفاده از فرآیند تصمیم‌گیری مارکوف توصیف رابطه بین سیاست‌های بارسپاری در محیط سیستم را انجام می‌دهیم. سپس از یک روش مبتنی بر یادگیری تقویتی برای کشف استراتژی بهینه استفاده می‌کنیم. این رو با استفاده از مدل مصرف انرژی ارایه شده در پژوهش [۴۰] به توسعه روشی ترکیبی و ایجاد تعادلی از معیارهای موجود در روش پیشنهادی می‌پردازیم.

از این جهت برای جلوگیری از نفرین ابعاد ناشی از افزایش نمایی فضای حالت یک روش یادگیری \mathcal{Q} عمیق را به کار می‌گیریم، که می‌تواند نتایج عددی اثربخشی را در سناریوهای مختلف به دست آورد.

نتایج شبیه‌سازی نشان داد که روش پیشنهادی در مقایسه با چندین روش معیار، از جمله روش چارچوب بارسپاری برخط مطابق با پژوهش [۳۰]، توانسته نتایج بهتری را کسب نماید. روش پیشنهادی می‌تواند نرخ وظایف منقضی شده، میزان مصرف انرژی در کل سیستم و میانگین تأخیر را کاهش دهد. این مزیت به ویژه زمانی قابل توجه است که وظایف به تأخیر حساس باشند یا سطوح بار در گره‌های لبه بالا باشد. با این وجود، رویکرد پیشنهادی دارای محدودیت‌هایی است که می‌تواند مورد بررسی قرار گیرند، که به عنوان پژوهش‌های آینده به شرح زیر مطرح شوند.

پژوهش‌های آینده: از محدودیت‌های موجود در الگوریتم پیشنهادی می‌توان به پیچیدگی و سربار بالای سیستم در مقیاس‌های بزرگ اشاره کرد. از آنجایی که به گره‌های لبه اجازه می‌دهیم به دستگاه‌های متصل کمک کنند تا شبکه‌های عصبی را آموزش دهند، مشکل سربار ارتباط و مقیاس‌پذیری به دلیل انتقال پارامتر شبکه عصبی ممکن است زمانی که شبکه عصبی بزرگ است نگران کننده باشد. همچنین با وجود محوریت تأخیر و مصرف انرژی در سیستم می‌توان دریافت کاهش و بهینه‌سازی مصرف انرژی زمانی بسیار تاثیرگذار خواهد بود که با محدودیت انرژی در دستگاه‌ها و گره‌های لبه مواجه باشیم. از این رو می‌توان با شبیه‌سازی بلندمدت از سیستم و با در نظر گرفتن محدودیت انرژی در دستگاه‌های متصل به ارزیابی سیستم پرداخته شود.

از جهتی می‌توان با توجه بیشتر به نیازهای هر کار به توسعه الگوریتم پیشنهادی در جهت بهبود کیفیت تجربه مشتری پرداخت. جالب است که با وجود نتایج بهبود یافته در الگوریتم پیشنهادی، می‌توان مشاهده نمود، که همواره بدء بستانی در هزینه تحمیلی سیستم متصل از سه معیار عملکرد برقرار خواهد بود، به نحوی که می‌توان با حذف هر کدام از آن‌ها، بهبودهای چشمگیری را در دو معیار دیگر مشاهده کرد، و از طرفی می‌توان دریافت که همهی دستگاه‌های متصل در سیستم می‌توانند نیازهای متفاوتی در هر زمان داشته باشند. به عنوان مثال اگر تجهیزات کاربری در مواجهه با اتمام انرژی تجهیزات خود باشد، احتمالاً می‌تواند متحمل اندکی تأخیر در مقابل انجام وظایف بیشتر باشد، اما در حالتی دیگر ممکن است حساسیت به تأخیر مهم بوده و بتوان از میزان مصرف انرژی صرفه نظر کرد. از این رو بسیار کارامد خواهد بود اگر بتوان معیارهای عملکرد را در برابر کاربران مختلف، شخصی‌سازی نمود.

می‌توان با در نظر گرفتن ترکیبی وزن دار از معیارهای اصلی عملکرد، به نحوی انحصاری، به محاسبه کیفیت خدمات پرداخت. همچنین می‌توان با وجود تاریخچه‌ای از هر کاربر و شرایط محیط در هر زمان پیش‌بینی‌ای از نیاز کاربر بدست آورد و با توجه به آن وزن هریک از معیارهای عملکرد را تعیین نمود. این امر می‌تواند با آموزش چند شبکه عصبی جداگانه برای هر معیار و بهره‌گیری از یک مکانیسم توجه^۱ پیاده‌شود.

همچنین از جهتی دیگر، کارآمد خواهد بود اگر مدل شبکه بی‌سیم ساده‌شده را گسترش دهیم تا جزئیات بیشتری را در محیط شبیه‌سازی شده دربر گیرد، به عنوان نمونه می‌توان با در نظر گرفتن خطای انتقال و تداخل بین دستگاه‌های تلفن همراه در ارتباطات به دقت واقع‌بینانه‌تری از سیستم دست‌یافت. همچنین می‌توان عملکرد الگوریتم را در یک سیستم آزمایشی ارزیابی کرد، که تحت آن بسیاری از مسائل عملی (به عنوان مثال، وظایف محاسباتی واقعی) مورد بررسی قرار گیرند. پیاده‌سازی می‌تواند با استفاده از شبیه‌سازهای مناسب انجام شود و الگوریتم پیشنهادی مورد ارزیابی قرار گیرد.

مراجع

- [1] Y. Mao, J. Zhang, and K. B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, 2016.
- [2] D. Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.
- [3] J. A. González-Martínez, M. L. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra. Cloud computing and education: A state-of-the-art survey. *Computers & Education*, 80:132–151, 2015.
- [4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.
- [5] K. Dolui and S. K. Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2017.
- [6] K. Gai and M. Qiu. Optimal resource allocation using reinforcement learning for iot content-centric services. *Applied Soft Computing*, 70:12–21, 2018.
- [7] Q. Li, J. Zhao, Y. Gong, and Q. Zhang. Energy-efficient computation offloading and resource allocation in fog computing for internet of everything. *China Communications*, 16(3):32–41, 2019.

- [8] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.
- [9] K. Toczé and S. Nadjm-Tehrani. A taxonomy for management and optimization of multiple resources in edge computing. *Wireless Communications and Mobile Computing*, 2018, 2018.
- [10] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris. Pre-fog: IoT trace based probabilistic resource estimation at fog. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 12–17. IEEE, 2016.
- [11] P. Athwani and D. P. Vidyarthi. Resource discovery in mobile cloud computing: A clustering based approach. In *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, pages 1–6. IEEE, 2015.
- [12] M. Chen and Y. Hao. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, 36(3):587–597, 2018.
- [13] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.
- [14] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [16] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [17] T. X. Tran and D. Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, 2018.

- [18] S. Guo, B. Xiao, Y. Yang, and Y. Yang. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.
- [19] S. Sundar and B. Liang. Offloading dependent tasks with communication delay and deadline constraint. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 37–45. IEEE, 2018.
- [20] X. Sun and N. Ansari. Latency aware workload offloading in the cloudlet network. *IEEE Communications Letters*, 21(7):1481–1484, 2017.
- [21] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Transactions on Wireless Communications*, 16(8):4924–4938, 2017.
- [22] S. Bi and Y. J. Zhang. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177–4190, 2018.
- [23] N. Eshraghi and B. Liang. Joint offloading decision and resource allocation with uncertain task computing requirement. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1414–1422. IEEE, 2019.
- [24] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu. Energy-efficient admission of delay-sensitive tasks for mobile edge computing. *IEEE Transactions on Communications*, 66(6):2603–2616, 2018.
- [25] K. Pouilarakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas. Joint service placement and request routing in multi-cell mobile edge computing networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 10–18. IEEE, 2019.
- [26] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj. Distributed online optimization of fog computing for selfish devices with out-of-date information. *IEEE Transactions on Wireless Communications*, 17(11):7704–7717, 2018.

- [27] L. Li, T. Q. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang. An incentive-aware job offloading control framework for multi-access edge computing. *IEEE Transactions on Mobile Computing*, 20(1):63–75, 2019.
- [28] H. Shah-Mansouri and V. W. Wong. Hierarchical fog-cloud computing for iot systems: A computation offloading game. *IEEE Internet of Things Journal*, 5(4):3246–3257, 2018.
- [29] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. Leung. A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing. *IEEE/ACM Transactions on Networking*, 26(6):2762–2773, 2018.
- [30] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci. Uloof: A user level online offloading framework for mobile edge computing. *IEEE Transactions on Mobile Computing*, 17(11):2660–2674, 2018.
- [31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [32] L. Huang, S. Bi, and Y.-J. A. Zhang. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, 19(11):2581–2593, 2019.
- [33] Y. Liu, H. Yu, S. Xie, and Y. Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, 68(11):11158–11168, 2019.
- [34] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang. Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks. *IEEE Transactions on Wireless Communications*, 18(11):5141–5152, 2019.
- [35] X. Xiong, K. Zheng, L. Lei, and L. Hou. Resource allocation based on deep reinforcement learning in iot edge computing. *IEEE Journal on Selected Areas in Communications*, 38(6):1133–1146, 2020.

- [36] J. Li, H. Gao, T. Lv, and Y. Lu. Deep reinforcement learning based computation offloading and resource allocation for mec. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.
- [37] N. Shan, X. Cui, and Z. Gao. “drl+ fl”: An intelligent resource allocation model based on deep reinforcement learning for mobile edge computing. *Computer Communications*, 160:14–24, 2020.
- [38] X. Liu, J. Yu, J. Wang, and Y. Gao. Resource allocation with edge computing in iot networks via machine learning. *IEEE Internet of Things Journal*, 7(4):3415–3426, 2020.
- [39] M. Tang and V. W. Wong. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, 2020.
- [40] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. Leung. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. *IEEE Internet of Things Journal*, 9(2):1517–1530, 2021.
- [41] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief. Delay-optimal computation task scheduling for mobile-edge computing systems. In *2016 IEEE international symposium on information theory (ISIT)*, pages 1451–1455. IEEE, 2016.
- [42] Y. Mao, J. Zhang, S. Song, and K. B. Letaief. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, 16(9):5994–6009, 2017.
- [43] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM transactions on networking*, 1(3):344–357, 1993.
- [44] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.
- [45] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

واژه‌نامه

الف

Dynamic programming	برنامه‌ریزی پویا	اشتراک‌گذاری منابع
Resource Optimization.....	بهینه‌سازی منابع	Resource Sharing.....
Feedback	بازخورد	Observable
Trade-off.....	بده‌بستان	Partially Observable.....
Exploitation	بهره‌برداری	strategy
Policy Improvement	بهبود سیاست	Exploration
Function Approximation.....	تقریب تابع	Policy Evaluation.....
Error Back Propagation.....	بازنشر خطا	First-in First-out
maximum	بیشینه	امتیاز
lyapunov optimization	بهینه‌سازی بر پایه لیاپانوف	ارزش

ب

Natural Language Processing .	پردازش زبان طبیعی .	بیدرنگ
robustness	پایداری	بارسپاری وظیفه
support	پشتیبان	Scheduling
PreTraining	پیش‌آموزش	Mixed Integer Linear Programming
Poisson	پواسون	

Internal State	حالت درونی	covering.....	پوششی
Environment State	حالت برونی		
Information State.....	حالت اطلاعاتی		
			ت
		Resource Allocation.....	تخصیص منابع
		Transmission Delay.....	تأخير ارسال
		Distributed.....	توزیع شده
		Offloading Decision.....	تصمیمگیری بارسپاری
		Fault Tolerance.....	تحمل پذیری اشکال
		Resource Estimation	تخمين منابع
		Resource Discovery	کشف منابع
		History	تاریخچه
		Resource Discovery	کشف منابع
		Value Function	تابع ارزش
		Action-value Function.....	تابع امتیاز-عمل
	Chain		ج
CPU Cycle.....	سیکل پردازش پردازنده	brute-force	جست و جوی جامع
		Depth-First Search.....	جست و جوی عمق اول
		bin	جعبه
			ح
Deep Neural Networks.....	شبکه عصبی عمیق	Delay-sensitive	حساس به تأخیر
Deep Belief Networks.....	شبکه باور عمیق	Long Short-term Memory	حافظه کوتاه مدت ماندگار
Time Slot.....	شکاف زمانی	State	حالت
Duelling Q Network	شبکه رقابتی Q		

م

Cloud Computing	محاسبات ابری
Edge Computing	محاسبات لبه
Mobile Edge Computing	محاسبات لبه متحرک
Centralize	متمرکز
Fog Computing	محاسبات مه
Resource Management	مدیریت منابع
Deadline	مهلت
Drop	منقضی

ص

Computation Queue	صف محاسبات
Transmission Queue	صف انتقال
Discount Factor	عامل تنزیل
agent-based	عامل-محور
action	عمل

ع

Virtual Reality	واقعیت مجازی
Autonomous Vehicles	وسایل نقلیه خودران
Virtual Reality	واقعیت مجازی

غ

non-Polynomial	غیر چندجمله‌ای
decentralized	غیر متمرکز
degenerate	غیرمعمول

و

Deep Reinforcement Learning	یادگیری تقویتی عمیق
Reinforcement Learning	یادگیری تقویتی
Supervised Learning	یادگیری بانظارت
Unsupervised Learning	یادگیری بدون ناظارت
Federated Learning	یادگیری انجمنی

ف

Markov Decision	فرایند تصمیم‌گیری مارکوف
Process	فرایند

ک

Quality of Service	کیفیت خدمات
Quality of Experience	کیفیت تجربه مشتری
Shallow	کم عمق

Abstract

With the explosion of mobile smart devices, many computation intensive applications have emerged, such as interactive gaming and augmented reality. Mobile edge computing (EC) is put forward, as an extension of cloud computing, to meet the low-latency requirements of the applications. In mobile edge computing systems, an edge node may have a high load when a large number of mobile devices offload their tasks to it. those offloaded tasks may experience large processing delay or even be dropped when their deadlines expire. Due to the uncertain load dynamics at the edge nodes, it is challenging for each device to determine its offloading decision (i.e., whether to offload or not, and which edge node it should offload its task to) in a decentralized manner. In this work, we studied the computational task offloading problem with non-divisible and delay-sensitive tasks in the MEC system, and formulate a task offloading problem to minimize the expected long-term cost. We propose a model-free deep reinforcement learning-based distributed algorithm, where each device can determine its offloading decision without knowing the task models and offloading decision of other devices. To improve the estimation of the long-term cost in the algorithm, we incorporate the long short-term memory (LSTM), dueling deep Q-network (DQN), and double-DQN techniques. Simulation results show that our proposed algorithm can better exploit the processing capacities of the edge nodes and significantly reduce the ratio of dropped tasks, average delay and energy consumptions in entire system, when compared with several basic method.

Keywords: Edge Computing, Computation Offloading, Resource Allocation, Energy Consumptions, Deep Reinforcement Learning.



Sharif University of Technology

Department of Computer Engineering

M.Sc. Thesis

A Novel Resource Allocation Algorithm in Edge Computing with Deep Reinforcement Learning

By:

Iman Rahmati

Supervisor:

Dr. Ali Movaghar

September 2022