
بسم الله الرحمن الرحيم

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر- هوش مصنوعی

بهبود الگوریتم‌های مسیریابی درشبکه‌های روی تراشه‌ی بی‌سیم به کمک یادگیری تقویتی

دانشجو: زهره هراتی

استاد راهنما

دکتر اسماعیل طحانیان

اساتید مشاور

دکتر علیرضا تجری

دکتر منصور فاتح

شهریور ماه ۹۹

شماره:

تاریخ:

باسمه تعالی

مدیریت تحصیلات تکمیلی

فرم شماره (۶)

فرم صورت جلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) نتیجه ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم / آقای ..... به شماره دانشجویی ..... رشته ..... گرایش .....

..... تحت عنوان .....

که در تاریخ ..... با حضور هیأت محترم داوران در دانشگاه شاهرود برگزار گردید به شرح ذیل اعلام می‌گردد:

قبول (با درجه : ..... امتیاز .....) ☐ دفاع مجدد ☐ مردود ☐

۱ـ عالی (۲۰ ـ ۱۹) ۲ـ بسیار خوب (۱۸/۹۹ ـ ۱۸)

۳ـ خوب (۱۷/۹۹ ـ۱۶) ۴ـ قابل قبول (۱۵/۹۹ ـ ۱۴)

۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استاد راهنما			
۲ـ استاد مشاور			
۳ـ نماینده شورای تحصیلات تکمیلی			
۴ـ استاد ممتحن			
۵ ـ استاد ممتحن			

رئیس دانشکده : امضاء

تقدیم به

مادر، پدر

و همسر عزیز و مهربانم

که در سختی‌ها و دشواری‌های زندگی، همواره یاوری دلسوز و فداکار

و پشتیبانی محکم و مطمئن برایم بوده‌اند.

از استاد گرامی جناب آقای دکتر طحانیان که راهنمایی این پایان‌نامه را به عهده

داشته‌اند؛ و با یاری‌ها و راهنمایی‌های بی‌دریغشان،

بسیاری از سختی‌ها را برایم آسان نمودند، کمال تشکر را دارم.

همچنین از

آقای دکتر تجری و آقای دکتر فاتح که مشاوره‌ی این پژوهش را تقبل کردند؛

کمال تشکر را دارم.

زهره هراتی

شهریور ۱۳۹۹

تعهدنامه

اینجانب زهره هراتی دانشجوی دوره کارشناسی ارشد رشته کامپیوتر گرایش هوش مصنوعی دانشکده کامپیوتر دانشگاه صنعتی شاهرود نویسنده پایان نامه " بهبود الگوریتم‌های مسیریابی درشبکه‌های روی تراشه بی‌سیم به کمک یادگیری تقویتی " تحت راهنمائی دکتر اسماعیل طحانیان، دکترتجری و دکتر فاتح متعهد می شوم.

- تحقیقات در این پایان نامه توسط اینجانب انجام شده و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا « «Shahrood University of Technology به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه، در مواردی که از موجود زنده (بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است.

تاریخ

امضای دانشجو

چکیده

شبکه‌های روی تراشه را می‌توان جایگزین فناوری گذرگاه در تراشه‌های با تعداد هسته‌ی زیاد دانست. در این تراشه‌ها، هسته‌ها به کمک سیم و یا از طریق بی‌سیم به هم وصل می‌شوند. علت اصلی استفاده از ارتباط بی‌سیم، کاهش تأخیر و توان مصرفی در شبکه می‌باشد. در شبکه‌های روی تراشه، مانند دیگر شبکه‌های مرسوم، مسئله‌ی مسیریابی حائز اهمیت است. الگوریتم مسیریابی مناسب در این شبکه‌ها، الگوریتمی است که ضمن کاهش تأخیر و افزایش کارایی، از پدیده‌ی بن‌بست نیز جلوگیری کند. واضح است که وقتی لینک‌های بی‌سیم به شبکه اضافه می‌شود، پیچیدگی‌های الگوریتم مسیریابی هم افزایش پیدا خواهد کرد. شایان ذکر است با استفاده از الگوریتم‌های مسیریابیِ مبتنی بر روش‌های یادگیری ماشینی، می‌توان مسیریابی را بهتر و با ازدحام کمتر انجام داد. در کارهای گذشته، الگوریتم یادگیری برای یک شبکه بر روی تراشه‌ی سیمی ارائه گردیده است. از آن‌جایی که در شبکه‌ی بی‌سیم، تقاضا برای استفاده از لینک‌های بی‌سیم زیاد است، الگوریتم یادگیری تا حد زیادی می‌تواند ازدحام در شبکه را کاهش دهد. از طرفی، اساس شبکه‌های بر روی تراشه‌ی بی‌سیم، متفاوت از شبکه‌های سیمی می‌باشد که خود منجر به تفاوت در پیاده‌سازی الگوریتم یادگیری تقویتی خواهد شد. لذا در این پایان‌نامه، تلاش می‌شود تا به کمک یادگیری تقویتی یک الگوریتم مسیریابی برای شبکه‌های روی تراشه‌ی بی‌سیم ارائه گردد. از آن‌جا که الگوریتم یادگیری تقویتی ارائه شده قابلیت تشخیص مسیر پر ازدحام و تغییر مسیر را دارد، می‌تواند در هنگام ارسال بسته از گره مبدأ به گره مقصد، تصمیم بهتری بگیرد. همچنین نتایج روش پیشنهادی با الگوریتم‌های مسیریابی گذشته مقایسه شده و در الگوریتم پیشنهادی حداقل ۸ درصد بهبودی ایجاد شده است.

کلمات کلیدی: شبکه‌ی روی تراشه (NOC)، الگوریتم یادگیری Q (Q-Learning)، یادگیری تقویتی (RL)، الگوریتم مسیریابی، بن‌بست، شبکه‌ی روی تراشه‌ی بی‌سیم، Noxim.

فهرست مطالب

فهرست جداول

فهرست اشکال

فهرست الگوریتم‌ها

فصل ۱: کلیات

۱-۱ مقدمه

با پیشرفت دنیای الکترونیک و افزایش توانایی در طراحی مدارات در مقیاس نانو، یکپارچه‌سازی در طراحی دستگاه‌های الکترونیکی به بحث روز تبدیل شده است و تلاش‌ها برای طراحی و ساخت دستگاه‌های یکپارچه ادامه دارد. این تلاش‌ها باعث ایجاد زمینه جدیدی با عنوان سیستم روی تراشه[۱] شده است. در سیستم روی تراشه سعی می‌گردد تا هسته‌های پردازشی و ارتباطی و واسط بر روی یک تراشه مجتمع شوند. یکی از معضلاتی که بر سر راه این یکپارچه‌سازی قرار دارد، نحوه اتصال و ارتباطات بین این هسته‌ها می‌باشد.

شبکه‌ی روی تراشه[۲] یک راه‌حل جدید برای ایجاد اتصالات میانی درون سیستم روی تراشه می‌باشد که در سال ۱۹۹۹ مطرح شد. در راه‌حل‌های سنتی، اتصالات میانی با استفاده از ساختار گذرگاه[۳] ایجاد می‌شدند. با فشرده‌تر شدن مدارها، راه‌حل‌های مبتنی بر گذرگاه، کارایی خود را در مقابل نیاز فناوری‌های جدید از دست دادند. گذرگاه‌ها شروع به محدود کردن و در بدترین حالت مسدود کردن ترافیک کردند. در شبکه‌ی روی تراشه ساختارهای مبتنی بر گذرگاه جای خود را به شبکه‌های شبیه به شبکه‌های کامپیوتری دادند که قسمت‌های مختلف با ارسال داده‌های بسته‌بندی شده، از طریق این شبکه با یکدیگر ارتباط برقرار می‌کنند.

همانند یک شبکه‌ی کامپیوتری، یک شبکه‌ی روی تراشه نیز شامل هسته‌ها، مسیریاب‌هایی[۴] که ترافیک را بین هسته‌ها مسیریابی می‌کنند و سیم‌هایی که دستگاه‌ها را به مسیریاب‌ها و مسیریاب‌ها را به یکدیگر وصل می‌کنند، می‌باشد.

---

۱ System On Chip (SOC)

۲ Network On Chip (NOC)

۳ Bus

۴ Router

۱-۲ شبکه‌ی روی تراشه

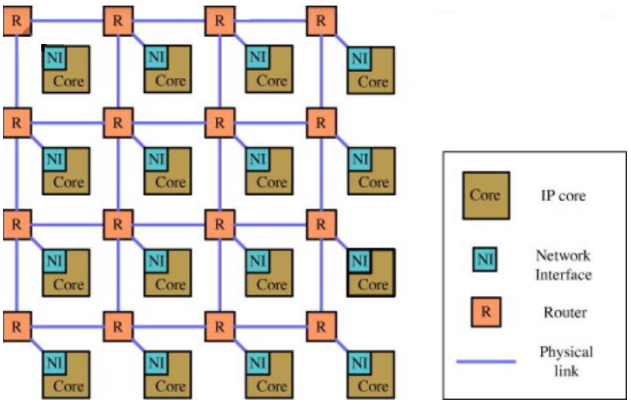
در تراشه‌های سنتی دو نوع مکانیزم ارتباطی وجود دارد که عبارت است از روش اتصالات نقطه به نقطه[۱] و روش اتصالات گذرگاه. این مکانیزم‌ها برای تراشه‌هایی استفاده می‌شود که دارای تعداد کمی عنصر پردازشی و حافظه‌ای باشند. در اتصالات نقطه به نقطه یک اتصال اختصاصی بین دو هسته‌ی پردازشی (که نیازمند به ارتباط با هم هستند) ایجاد می‌شود. در زمان کم بودن تعداد هسته‌ها، این روش بهترین کارایی و توان مصرفی را برای برقراری ارتباط دارد؛ به این دلیل که در این روش تنها از سیم‌ها و بدون استفاده از سخت‌افزار اضافه برای انتقال داده‌ها استفاده می‌شود؛ اما این روش مشکلات زیادی دارد که عبارت است از پیچیدگی زیاد طراحی و مسیریابی اتصالات، عدم مقیاس‌پذیری و هزینه‌ی پیاده‌سازی بالا. در روش اتصالات گذرگاه‌ها، هسته‌های پردازشی با استفاده از یک کانال مشترک به یکدیگر متصل می‌شوند. روش گذرگاه در مقایسه با روش نقطه به نقطه، پیچیدگی طراحی کمتری دارد و چون از کانال‌های کمتری استفاده می‌کند، هزینه‌ی پیاده‌سازی آن نیز پایین‌تر است؛ اما این روش دو مشکل اساسی دارد: یکی عدم مقیاس‌پذیری توان و دیگری کارایی پایین. با زیاد شدن تعداد دستگاه‌های متصل به گذرگاه، طول و نیز مدارات ارسال و دریافت داده‌ی متصل به آن افزایش می‌یابد که منجر به ایجاد بار خازنی زیادی می‌گردد. این بار خازنی تأخیر و همچنین توان مصرفی را افزایش می‌دهد.

علاوه بر این، تمام عناصر متصل به گذرگاه از یک مسیر واحد استفاده می‌نمایند و لذا در هر لحظه فقط دو گره با هم ارتباط دارند و سایر گره‌ها باید منتظر آزاد شدن کانال بمانند. این امر موجب کاهش شدید کارآیی سیستم (به ویژه هنگامی‌که عناصر متقاضی ارتباط زیاد باشد) می‌شود. برای مقابله با این مشکلات نیاز به بازنگری در روش‌های سنتی ارتباطی درون تراشه ایجاد شد و شبکه‌ی روی تراشه به عنوان یک طرح ارتباطی درون تراشه‌ایِ نوین برای رفع و کاهش این مشکل مورد توجه قرار گرفت. پس در واقع می‌توان گفت شبکه‌های روی تراشه جایگزین فناوری گذرگاه در تراشه‌های با هسته‌های زیاد می‌باشد [۱۷-۲۰].

[۱] Point To Point

شبکه‌ی روی تراشه شبکه‌ای m×n از مسیریاب‌ها و منابع است [۳۹، ۴۰]. ساده‌ترین ساختاری که می‌توان برای یک شبکه‌ی روی تراشه در نظر گرفت، یک شبکه دوبعدی است و اکثر تحقیقاتی که انجام شده است بر روی همین ساختار بوده است. دلیل استفاده از ساختار دوبعدی، سادگی پیاده‌سازی و کار کردن با آن است [۴۱]. منابع می‌تواند هسته‌ی پردازنده، حافظه، بلاک سخت‌افزاری سفارشی یا هر بلاک با ویژگی خاصی باشد. منابع در واقع واحدهای محاسباتی یا ذخیره‌سازی هستند. منابع و مسیریاب‌ها با استفاده از کانال به یکدیگر ارتباط دارند.

شبکه‌های روی تراشه از سه قسمت اصلی تشکیل یافته‌اند: مسیریاب، رابط شبکه، اتصالات بین مسیریاب‌ها.



شکل ۱-۱) ساختار کلی شبکه‌ی روی تراشه

همان‌طور که در شکل ۱-۱ دیده می‌شود، در هر گره از شبکه، هسته به یک مسیریاب متصل است و این مسیریاب از طریق اتصالات نقطه به نقطه در یک شبکه به سایر مسیریاب‌ها وصل شده است. ارتباط بین گره‌ها از طریق تولید بسته‌های داده و ارسال آن‌ها بر روی این زیرساخت ارتباطی انجام می‌شود. رابط شبکه، ارتباط بین هسته‌ها و مسیریاب‌ها را از طریق بسته‌بندی داده‌ها در سمت فرستنده و باز کردن بسته‌ها و سپس استخراج داده‌های آن در سمت گیرنده برقرار می‌کند. به بیان دقیق‌تر، این رابط وظیفه‌ی تبدیل یک پیام به چندین بسته در طرف فرستنده و ترکیب بسته‌ها و بازسازی پیام در سمت مقابل را به عهده دارد. رابط شبکه برای انجام این کار وظیفه تبدیل پروتکل پیاده‌سازی شده در واسط هسته به بسته‌های قابل انتقال در شبکه‌های

روی تراشه را به عهده دارد. در طرف دیگرِ انتقال، رابط شبکه بسته‌های رسیده را به داده‌های قابل فهم برای هسته (طبق پروتکل واسط آن هسته) تبدیل می‌کند.

۱-۳ فلیت

هر بسته از تعدادی فلیت تشکیل شده است که انتخاب فلیت‌هایی با ابعاد کوچک می‌تواند باعث سادگی طراحی مسیریاب شود. به طور کلی بسته از حداقل سه بخش تشکیل شده که شامل سرآیند[۱]، بدنه[۲] و انتهای بسته[۳] است. هر بسته فقط یک سرآیند، یک یا چند بدنه و یک انتهای بسته دارد. در هنگام ارسال داده، ابتدا سرآیند بسته ارسال می‌شود و مسیر توسط آن برای انتقال داده رزرو می‌شود. وقتی یک یا چند بدنه ارسال شد، در انتها با ارسال انتهای بسته، مسیر رزرو شده آزاد می‌شود تا امکان رزرو آن توسط گره‌های دیگر وجود داشته باشد.

از آن جا که در شبکه‌های روی تراشه از سیم‌های مشترک به جای سیم‌های اختصاصی برای عبور سیگنال‌ها استفاده می شود، باید هزینه‌ی آن منطقی باشد. مقدار دیتایی که در هر کلاک در شبکه‌ی روی تراشه جابه‌جا می‌شود به اندازه‌ی تعداد سیم‌ها است؛ مثلا وقتی در هر کلاک ۳۲ بیت جابه‌جا می‌شود، در واقع ۳۲ سیم بین دو گره وجود دارد. اگر بخواهیم برای ارسال داده از بسته استفاده کنیم، مشکلاتی به وجود می‌آید که عبارت است از:

- با توجه به این که اندازه‌ی بسته‌ها به نسبت فلیت‌ها زیاد است باید مقدار بسیار زیادی سیم بین دو گره ایجاد شود. از طرفی تعداد سیم‌های بین گره‌ها محدود است و در صورت محدود نبودن، ایجاد سیم‌های زیاد بین دو گره توان مصرفی را به شدت بالا می‌برد.
- از آن جا که اندازه‌ی بسته‌ها یکسان نیست، باید همیشه به اندازه‌ی بزرگ‌ترین بسته سیم بین دو گره ایجاد شود که ممکن است در اکثر موارد تعداد کمی از این سیم‌ها استفاده شود.

---

[۱] Head

[۲] Body

[۳] Tail

- برای انتقال بسته‌ها به حافظه‌ی بافر زیادی نیاز است که این خود باعث افزایش تأخیر بسته‌ها می‌شود.

پس در واقع می‌توان گفت که فلیت، راه‌حلی برای انتقال داده در شبکه‌ی روی تراشه است.

۱-۴ هدف پایان‌نامه

در روش‌های موجود در گذشته [۸-۱۰]، الگوریتم یادگیری برای یک شبکه‌ی روی تراشه‌ی سیمی ارائه گردیده است و از اتصالات بی‌سیم استفاده نشده است. از آن‌جا که در شبکه‌ی بی‌سیم[۱] تقاضا برای استفاده از لینک‌های آن به دلیل سهولت در ارسال بسته زیاد است، الگوریتم یادگیری تا حد زیادی می‌تواند ازدحام در شبکه را کاهش دهد. هدف این پایان‌نامه بهبود مسیریابی درشبکه‌های روی تراشه‌ی بی‌سیم به کمک یادگیری تقویتی است.

۱-۵ چالش‌ها و راه‌حل‌های آن‌ها

یکی از چالش‌های مهم در طراحی محیط‌های بی‌سیم، کنترل تقاضاها برای استفاده از لینک‌های بی‌سیم است؛ چون در صورت وجود لینک‌های بی‌سیم، گره‌ها تمایل به ارسال داده از طریق این لینک‌ها به گره مقصد دارند؛ و این باعث می‌شود که در این لینک‌های بی‌سیم بعد از مدتی ازدحام پیش بیاید و وجود ازدحام در شبکه باعث کاهش کارایی شبکه می‌شود. پس باید در صورت استفاده از محیط‌های بی‌سیم روشی ایجاد گردد که بتواند ازدحام مذکور را مدیریت کند. در این پایان‌نامه برای رفع این چالش از الگوریتم یادگیری تقویتی بر اساس سازوکار پاداش و مجازات استفاده شده است. سازوکار پاداش و مجازات در این الگوریتم به این صورت است که در صورت انتخاب مسیر کم ازدحام در شبکه به عمل پاداش تعلق می‌گیرد و در صورت انتخاب مسیر پر ازدحام نیز مجازات شامل حال عمل می‌شود.

[۱] Wireless

تاکنون تحقیقاتی برای این چالش با استفاده از یادگیری تقویتی در شبکه‌ی روی تراشه‌ی بی‌سیم انجام نشده است. کارهایی که در زمینه‌ی یادگیری تقویتی در شبکه‌ی روی تراشه انجام شده، همه در شبکه‌های سیمی بوده است که در فصل دوم به بررسی آن پرداخته شده است.

چالش دیگر، به وجود آمدن بن‌بست در شبکه است که در صورت استفاده از یادگیری تقویتی برای شبکه‌ی روی تراشه ایجاد می‌شود. بن‌بست در شبکه به این صورت است که بعد از گذشت زمانی، بسته‌ی جدیدی به شبکه تزریق نمی‌شود. در این پایان‌نامه با بررسی علل به وجود آمدن بن‌بست در شبکه‌ی روی تراشه، از افزایش تعداد کانال‌های مجازی برای رفع بن‌بست استفاده شده است.

۱-۶ نوآوری

در این پایان‌نامه یک الگوریتم مسیریابی مبتنی بر یادگیری تقویتی ارائه شده است که دارای قابلیت اجرا در شبکه‌ی روی تراشه‌ی بی‌سیم است و برای جابه‌جایی داده در شبکه‌ی روی تراشه از فلیت استفاده می‌کند؛ در حالی که در روش‌های قبلی، از بسته استفاده می‌شود. این الگوریتم مسیریابی قابلیت سازگاری با شبکه دارد. به این صورت که اگر در مسیر حرکت بسته از گره مبدأ به گره مقصد، مسیر پر ازدحام باشد، تغییر مسیر داده و مسیر کم ازدحام‌تر را انتخاب می‌کند. تغییر مسیر توسط الگوریتم، بر اساس اطلاعات موجود در جداول گره‌های آن صورت می‌گیرد که اطلاعات این جداول، بر اساس اطلاعاتی که در هر جابجایی گره‌ها به دست می‌آیند، به‌روزرسانی می‌شوند. در واقع یادگیری در این الگوریتمِ مسیریابی، ماحصل تغییر مقادیر جداول است.

۱-۷ ساختار پایان‌نامه

در ادامه این تحقیق، در فصل دوم، مروری بر روش‌های پیشین انجام شده است. در فصل سوم، روش پیشنهادی ارائه شده است. در فصل چهارم، نتایج و آزمایش‌ها مورد بررسی قرار گرفته است و در فصل پایانی، جمع‌بندی و نتیجه‌گیری ارائه شده است.

فصل ۲: پیشینه‌ی تحقیق

۲-۱ مقدمه

ایده‌ی شبکه‌ی روی تراشه با استفاده از لینک‌های بی‌سیم اولین بار در سال ۲۰۱۰ مطرح شده است [۱] و تاکنون از جنبه‌های مختلف از جمله الگوریتم‌های مسیریابی به آن پرداخته شده است. مهم‌ترین روش‌های مسیریابی که تاکنون ارائه شده است عبارت است از: مسیریابی توزیع شده و متمرکز[۱] [۲، ۲۳، ۳۶، ۳۷]، مسیریابی قطعی و انطباقی[۲] [۳، ۱۷، ۲۴، ۲۵]، مسیریابی حداقلی و غیرحداقلی[۳] [۴، ۲۸، ۲۹، ۳۰]، مسیریابی با توجه به دمای شبکه[۴] [۵، ۲۲، ۳۴، ۳۵] و نیز مسیریابی خطاپذیر[۵] [۶، ۳۱، ۳۲، ۳۳].

از این الگوریتم‌های مسیریابی، یکی از مهم‌ترین آن‌ها، الگوریتم مبتنی بر یادگیری است که می‌توان آن را جزء دسته‌ی انطباقی قرار داد.

به طور کلی در این فصل، در بخش ۲-۱ در مورد نحوه‌ی هم‌بندی در شبکه‌ی روی تراشه در بخش ۲-۲ الگوریتم‌های مسیریابی در شبکه‌ی روی تراشه، در بخش ۲-۳ ویژگی‌هایی که هر الگوریتم مسیریابی باید داشته باشد، در بخش ۲-۴ تعدادی از الگوریتم‌های مسیریابی که در دو دسته‌ی قطعی و تطبیقی قرار می‌گیرند و در بخش ۲-۵ به بحث و بررسی مقالاتی پرداخته خواهد شد که قبلاً از الگوریتم تقویتی در شبکه‌ی روی تراشه استفاده کرده‌اند و در بخش ۲-۶ نتیجه‌گیری کلی ارائه می‌شود.

۲-۲ هم‌بندی در شبکه‌ی روی تراشه

نحوه‌ی اتصال گره‌ها به یکدیگر در یک سیستم چند کامپیوتری، هم‌بندی آن سیستم را مشخص می‌کند. شبکه‌ی بین پردازنده‌های هر سیستمِ چند کامپیوتری را می‌توان به‌وسیله یک گراف مشخص کرد که هر گرهِ این گراف نمایانگر یک عنصر پردازشی و هر یال آن نمایانگر کانال فیزیکی بین دو گره می‌باشد.

---

[۱] Source And Distributed Routing

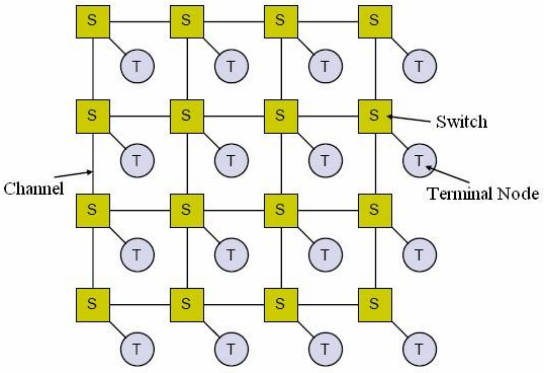
[۲] Deterministic And Adaptive Routing

[۳] Minimal And non-minimal Routing

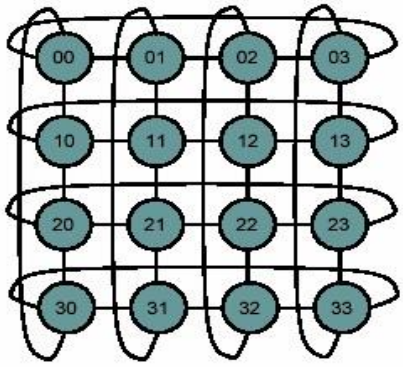
[۴] Thermal-aware Routing

[۵] Fault-Tolerant Routing

پیمایش از یک گره به گره همسایه را یک گام می‌گویند و حداقل تعداد گام‌های بین هر دو گره، فاصله‌ی بین آن دو گره را مشخص می‌کند. حداکثر مقدار (در مقیاس گام)، از حداقل فاصله‌های بین هر دو گره موجود در شبکه، قطر آن شبکه را مشخص می‌کند. شبکه‌های مورد استفاده در شبکه‌ی روی تراشه بر اساس نوع کاربرد و معماری سیستم دارای هم‌بندهای مختلفی می‌باشند. شکل ۲-۱ ساختار پیشنهاد شده برای شبکه‌ی روی تراشه که مبتنی بر شبکه‌ی دوبعدی می‌باشد را نشان می‌دهد. شکل ۲-۲ نیز هم‌بندی Torus را نشان می‌دهد که گره‌های با برچسب S مشخص کننده‌ی مسیریاب‌ها و گره‌های با برچسب T، مشخص کننده ترمینال گره‌ها می‌باشند.



شکل ۲-۱) هم‌بندی شبکه



شکل ۲-۲) هم‌بندی Torus

۲-۳ الگوریتم‌های مسیریابی در شبکه‌ی روی تراشه

در واقع ایده‌ی شبکه‌ی روی تراشه از شبکه‌های کامپیوتری و سیستم‌های توزیع شده در مقیاس بزرگ گرفته شده است. با این حال تکنیک‌های مسیریابیِ به کار رفته برای شبکه‌ی روی تراشه باید طراحی منحصر به

فردی داشته باشند و توان عملیاتی شبکه را تضمین کنند. با توجه به موانع و محدودیت‌هایی که در حافظه و منابع موجود بر روی تراشه وجود دارد، تکنیک مسیریابی به کار رفته برای شبکه‌ی روی تراشه باید منطقی ساده داشته باشد. به طور کلی کارایی شبکه‌ی روی تراشه نیز همانند هر شبکه‌ی دیگر، به طور گسترده‌ای وابسته به تکنیک مسیریابی به کار رفته در آن می‌باشد.

الگوریتم مسیریابی[۱]، مسیر بین مسیریاب مبدأ و مقصد را تعیین می‌کند. الگوریتم مسیریابیِ خوب باید عاری از بن‌بست[۲]، سرگردانی[۳] و گرسنگی[۴] باشد.

الگوریتم‌های مسیریابی را می‌توان به دو دسته‌ی قطعی[۵] و انطباقی[۶] تقسیم‌بندی کرد. در الگوریتم‌های مسیریابی قطعی، تمام مسیرها در همان ابتدا از گره مبدأ به گره مقصد مشخص شده و سپس بسته‌ها از همان مسیرهای تعیین شده به گره مقصد ارسال می‌شوند؛ ضمن این که در هنگام تعیین مسیر برای بسته‌ها، هیچ توجهی به شرایط ترافیکی شبکه نمی‌شود. در مقابل آن، الگوریتم‌های مسیریابی انطباقی قرار دارند. در الگوریتم‌های انطباقی تصمیمات مسیریابی با توجه به شرایط ترافیکی شبکه اخذ می‌شود و معمولاً در این الگوریتم‌ها سعی می‌شود تا در صورت وجود ازدحام در شبکه، بسته‌ها از مسیرهای دیگری که دارای ازدحام کمتری هستند به سمت مقصد مسیریابی شوند. الگوریتم‌های مسیریابی قطعی و انطباقی هرکدام دارای یک سری مزایا و معایب می‌باشند. یکی از مهم‌ترین مزایای الگوریتم‌های مسیریابی قطعی این است که به دلیل ساده بودن منطق مسیریابی، طراحی مسیریاب‌های آن بسیار ساده‌تر است و در زمانی که شبکه دارای ازدحام کمی باشد، دارای تأخیر مسیریابی کمتری هستند. ولی باید توجه داشت در زمانی که ازدحام شبکه بالا می‌باشد، تأخیر مسیریابی الگوریتم‌های قطعی بسیار زیاد است و این امر باعث کاهش شدید توان عملیاتی شبکه خواهد شد. این مشکل بدین دلیل به وجود می‌آید که الگوریتم‌های قطعی در هنگام پر ازدحام بودن

---

۱ Routing algorithm

۲ Dead Lock

۳ Live Lock

۴ Starvation

۵ Deterministic

۶ Adaptive

شبکه قادر نیستند برای اجتناب از ازدحام، از مسیرهای دیگری جهت ارسال بسته به مقصد استفاده کنند. در مقابل، الگوریتم‌های مسیریابی انطباقی به دلیل توانایی ارائه‌ی مسیرهای دیگر در هنگام برخورد با ازدحام در لینک‌های شبکه، می‌توانند تأخیر کمتری داشته باشند و باعث افزایش توان عملیاتی شبکه شوند. ولی الگوریتم-های انطباقی در هنگامی که شبکه دارای ترافیک کمی می‌باشد دارای تأخیر بیشتری نسبت به الگوریتم‌های قطعی خواهند بود. این امر را می‌توان به دلیل پیچیده‌تر بودن منطق مسیریابی انطباقی دانست.

۲-۴ خصوصیات مهم الگوریتم‌های مسیریابی

برخی از خصوصیات شبکه که مستقیماً به الگوریتم مسیریابی وابسته‌اند، عبارت است از:

- قابلیت اتصال[۱]: قابلیت و توانایی مسیریابی بسته‌ها از هر گره مبدأ به هر گره مقصد در شبکه.
- قابلیت تطبیق‌پذیری[۲]: توانایی ارسال بسته‌ها به گره مقصد از طریق مسیرهای متفاوت با وجود اجزا و خطوط مشغول، پرتراکم یا خطادار.
- عاری بودن از بن‌بست و سرگردانی بسته‌ها: توانایی تضمین این که بسته‌ها تا ابد در شبکه سرگردان نخواهند بود و در بن‌بست نیز گرفتار نخواهند شد. بن‌بست ممکن است به صورت یک چرخه‌ی وابسته در میان گره‌هایی که درخواست منابع را دارند، صورت گیرد.
- قابلیت تحمل‌پذیری خطا[۳]: توانایی مسیریابی بسته‌ها با وجود اجزای معیوب و خطادار.

۲-۵ الگوریتم‌های مسیریابی

معماری‌های متعددی برای طراحی مسیریاب‌های شبکه‌ی روی تراشه ارائه شده است. همچنین الگوریتم‌های مسیریابی متعددی جهت استفاده در شبکه‌ی روی تراشه مطرح گردیده است. زمانی که یک بسته از درون یک شبکه‌ی توری دوبعدی عبور می‌کند، می‌تواند به چهار جهت حرکت کند: شمال، جنوب، شرق و غرب؛ بنابراین

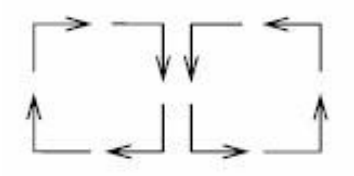
---

[۱] Connectivity

[۲] Adaptivity

[۳] Fault tolerant

هشت نوع چرخش ممکن و نیز دو دور انتزاعی وجود دارد که ممکن است توسط بسته انجام گیرد. این چرخش‌ها در شکل ۲-۳ نشان داده شده‌اند. الگوریتمی که هیچ محدودیتی در انجام این چرخش‌ها قائل نشود، الگوریتم کاملاً انطباقی نامیده می‌شود، ولی اگر در انجام هر یک از این چرخش‌ها محدودیتی اعمال گردد، به الگوریتم حاصله، انطباقیِ جزئی گفته می‌شود. الگوریتم‌های کاملاً انطباقی، مستعد بروز بن‌بست هستند، ولی اگر حداقل از دو چرخش ممانعت به عمل آید، این امکان وجود خواهد داشت که الگوریتم حاصله عاری از بن‌بست باشد.



شکل ۲-۳) هشت نوع چرخش ممکن در یک شبکه‌ی دوبعدی

در این بخش چند الگوریتم مسیریابی انطباقی جزئی و یک الگوریتم قطعی معرفی خواهند شد. این الگوریتم‌ها عبارت است از: XY، West-First، North-Last، Negative-First، East-First، South-Last، SNWE. لازم به ذکر است که الگوریتم‌های انطباقی ذکر شده، به صورت مینیمال مورد بررسی قرار خواهد گرفت، به این معنی که همواره با هر حرکت، یک قدم به هدف نزدیک می‌شوند و تحت هیچ شرایطی از هدف دور نمی‌شوند. تمام الگوریتم‌های نام برده شده در بالا به غیر از XY همگی از بن‌بست، سرگردانی و گرسنگی محفوظ هستند، یعنی تحت هیچ شرایطی با هیچ‌کدام از موارد ذکر شده مواجه نمی‌شوند، ولی الگوریتم XY که یک الگوریتم قطعی می‌باشد ممکن است با بن‌بست مواجه شود.

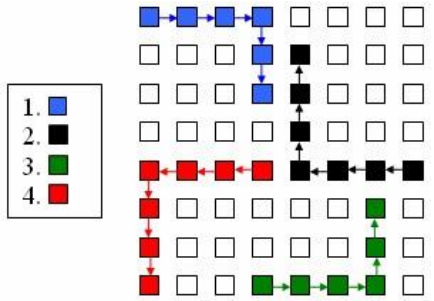
گره‌های مبدأ و مقصد با X,Y نمایش داده خواهند شد. بدین ترتیب که گره مبدأ به صورت (X_S,Y_S) و گره مقصد به صورت (X_T,Y_T) نمایش داده خواهند شد. لازم به ذکر است که در شبه کدها به جای X_S از 1x، به جای Ys از 1y، به جای X_t از tx و به جای Y_t از ty استفاده می‌شود.

۲-۵-۱ الگوریتم مسیریابی XY

این الگوریتم یک الگوریتم مسیریابی قطعی است کـه فلیت‌هـا در آن ابتدا مسیر Xها را طی می‌کنند تا زمانی که Y_T و Y_S بر روی یک خـط از X قرار گیرند. سپس فلیت‌ها در جهـت Yها حرکـت می‌کنند تا به مقصد برسند. در شکل ۲-۴، چهار مسـیریابی XY نمایش داده شده است. شبه کد مربوط به آن، در الگوریتم ۲-۱ آورده شده است.

الگوریتم ۲-۱) الگوریتم مسیریابی XY

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.  elsif lx <> tx and auxfree(dirx)='1' then
4.      Go X Dimension;
5.  elsif lx = tx and ly <> ty and auxfree(diry)='1' then
6.      Go Y Dimension;
```



شکل ۲-۴) مسیرهای پیموده شده توسط الگوریتم XY

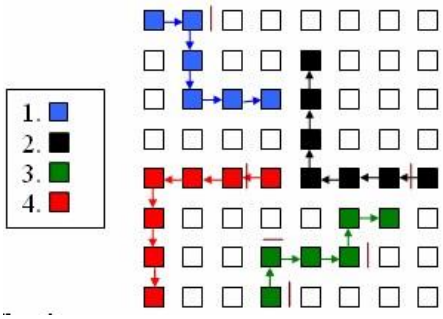
۲-۵-۲ الگوریتم مسیریابی West-First

هدف این الگوریتم اولویت‌دهی به گره‌هایی است که در سمت غرب شبکه قرار دارد. در این الگوریتم موقعی که $X_T \leq X_S$ باشد، بسته‌ها به همـان روش XY مسـیریابی مـی‌شـوند. اگـر $X_T > X_S$ باشد، بسته‌ها به صورت توافقی در جهـات شـرق یا شمال یا جنوب هدایت می‌شوند. زمان کـل بـرای انتقـال یک بسته در الگوریتم توافقی کاهش داده می‌شود، چرا کـه هنگامی که به یک حالت بلوکه می‌رسد و نمی‌تواند بـه سمت آن گره

حرکت کند، به صورت توافقی بـه مسـیر دیگری حرکت می‌کند. شبه کد مربوط به آن، در الگوریتم ۲-۲ آورده شده است.

الگوریتم ۲-۲) الگوریتم مسیریابی West-First

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.   elsif lx > tx and auxfree(WEST)='1' then
4.      Go WEST;
5.  elsif lx < tx and auxfree(EAST)='1' then
6.      Go EAST;
7.   elsif (lx = tx or lx < tx) and ly < ty and auxfree(NORTH)='1' then
8.      Go NORTH;
9.  elsif (lx = tx or lx < tx) and ly > ty and auxfree(SOUTH)='1' then
10.     Go SOUTH;
```



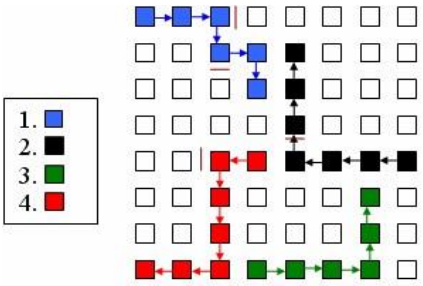
شکل ۲-۵) مسیرهای پیموده شده توسط الگوریتمWest-First

۲-۵-۳ الگوریتم مسیریابی North-Last

هدف این الگوریتم کاهش اولویت گره‌هایی است که در سمت شمال شبکه قرار دارند. در این الگوریتم موقعی که $Y_T \leq Y_S$ باشد بسته‌ها به همان روش XY مسیریابی می‌شوند و اگر $Y_T > Y_S$ باشد، بسته‌ها به صورت توافقی در جهات غرب یا شرق یا جنوب هدایت می‌شوند. شبه کد مربوط به آن، در الگوریتم ۲-۳ آورده شده است.

الگوریتم ۲-۳) الگوریتم مسیریابی North-Last

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.  elsif lx < tx and auxfree(EAST)='1' then
4.      Go EAST;
5.  elsif lx > tx and auxfree(WEST)='1' then
6.  elsif ly > ty and auxfree(SOUTH)='1' then
7.      Go SOUTH;
8.  elsif lx = tx and ly < ty and auxfree(NORTH)='1' then
9.      Go NORTH;
```



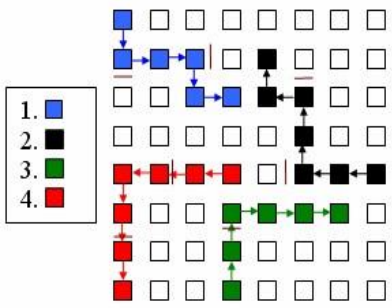
شکل ۲-۶) مسیرهای پیموده شده توسط الگوریتم North-Last

۲-۵-۴ الگوریتم مسیریابی Negative-First

هدف این الگوریتم اولویت‌دهی به گره‌هایی است که در سمت جنوب و غرب شبکه قرار دارند. در این الگوریتم بسته‌ها ابتدا در جهات منفی شمال یا غرب حرکت می‌کنند. اگر ($X_T \leq X_S$ و $Y_T \geq Y_S$) یا ($X_T \geq X_S$ و $Y_T \leq Y_S$) باشد، در این صورت بسته‌ها به صورت قطعی مسیریابی می‌شوند (مسیرهای ۳ و ۴ در شکل ۲-۷ این عمل را نشان می‌دهد). شبه کد مربوط به آن، در الگوریتم ۲-۴ آورده شده است.

الگوریتم ۲-۴) الگوریتم مسیریابی Negative-First

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.  elsif lx > tx and auxfree(WEST)='1' then
4.      Go WEST;
5.  elsif ly > ty and auxfree(SOUTH)='1' then
6.      Go SOUTH;
7.  elsif (ly = ty or ly < ty) and lx < tx and auxfree(EAST)='1' then
8.      Go EAST;
9.  elsif (lx = tx or lx < tx) and ly < ty and auxfree(NORTH)='1' then
10.     Go NORTH;
```



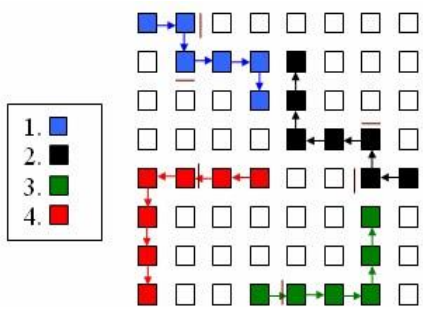
شکل ۲-۷) مسیرهای پیموده شده توسط الگوریتم Negative-First

۲-۵-۵ الگوریتم مسیریابی East-First

هدف این الگوریتم اولویت‌دهی به گره‌هایی است که در سمت شرق شبکه قرار دارد. در این الگوریتم موقعی که $X_T > X_S$ باشد، بسته‌ها به همان روش XY مسیریابی می‌شوند. اگر $X_T \leq X_S$ باشد، بسته‌ها به صورت توافقی در جهات غرب یا شمال یا جنوب هدایت می‌شوند. شبه کد مربوط به آن، در الگوریتم ۲-۵ آورده شده است.

الگوریتم ۲-۵) الگوریتم مسیریابی East-First

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.  elsif lx < tx and auxfree(EAST)='1' then
4.      Go EAST;
5.  elsif lx > tx and auxfree(WEST)='1' then
6.      Go WEST;
7.  elsif (lx = tx or lx > tx) and ly < ty and auxfree(NORTH)='1' then
8.      Go NORTH;
9.  elsif (lx = tx or lx > tx) and ly > ty and auxfree(SOUTH)='1' then
10.     Go SOUTH;
```



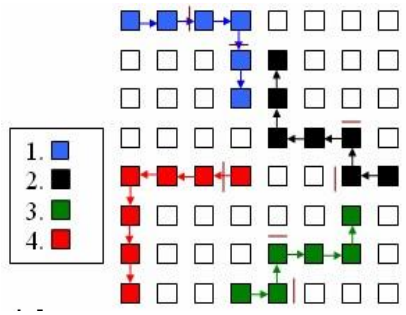
شکل ۲-۸) مسیرهای پیموده شده توسط الگوریتمEast-First

۲-۵-۶ الگوریتم مسیریابی South-Last

هدف این الگوریتم کاهش اولویت گره‌هایی است که در سمت جنوب شبکه قرار دارند. در این الگوریتم موقعی که $Y_T>Y_S$ باشد، بسته‌ها به همان روش XY مسیریابی می‌شوند. اگـر $Y_T \leq Y_S$ باشد، در این صورت بسته‌ها به صورت تـوافقی در جهـات غـرب یا شرق یا شمال هدایت می‌شوند. شـبه کـد مربـوط بـه آن، در الگوریتم ۲-۶ آورده شده است.

الگوریتم ۲-۶) الگوریتم مسیریابی South-Last

```
1.  if lx = tx and ly = ty and auxfree(LOCAL)='1' then
2.      Go Core;
3.  elsif lx < tx and auxfree(EAST)='1' then
4.      Go EAST;
5.  elsif lx > tx and auxfree(WEST)='1' then
6.      Go WEST;
7.  elsif ly < ty and auxfree(NORTH)='1' then
8.      Go NORTH;
9.  elsif lx = tx and ly > ty and auxfree(SOUTH)='1' then
10.     Go SOUTH;
```



شکل ۲-۹) مسیرهای پیموده شده توسط الگوریتمSouth-Last

۲-۵-۷ الگوریتم مسیریابی Wireless

این الگوریتم با استفاده از اتصالات بی‌سیم، بسته را سریع‌تر و با تأخیر کمتر از گره مبدأ به گره مقصد ارسال می‌کند. در این الگوریتم موقعی که بسته‌ای به گره‌ای می‌رسد دو حالت بررسی می‌شود:

- ارسال بسته از گره مبدأ به گره مقصد به همان روش XY.
- ارسال بسته از گره مبدأ به گره مقصد از طریق ارسال بسته به نزدیک‌ترین مسیریاب بی‌سیم مبدأ، دریافت آن در نزدیک‌ترین مسیریاب بی‌سیم مقصد و سپس ارسال بسته به گره مقصد.

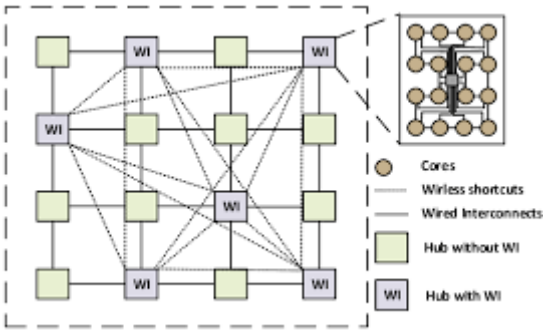
انتخاب هر یک از حالات فوق، از طریق محاسبه‌ی فاصله‌ی دکارتی بین مبدأ و مقصد، یک بار بدون در نظر گرفتن گره بی‌سیم و یک بار با در نظر گرفتن گره بی‌سیم انجام می‌شود. اگر فاصله‌ی اول از فاصله‌ی دوم کمتر باشد، از روش الگوریتم مسیریابی XY استفاده خواهد شد؛ در غیر این صورت، از الگوریتم مسیریابی WirelessXY استفاده خواهد شد.

در صورت عدم گرهِ بی‌سیم، فاصله‌ی دکارتی برابر است با:

|فاصله ستون مبدأ و ستون مقصد| + |فاصله سطر مبدأ و سطر مقصد|

و در صورت وجود گرهِ بی‌سیم، فاصله‌ی دکارتی برابر است با:

|فاصله دکارتی بین مقصد و نزدیک‌ترین گره بی‌سیم به مقصد| + |فاصله دکارتی بین مبدأ و نزدیک‌ترین گره بی‌سیم به مبدأ| + هزینه استفاده از بی‌سیم [۵۸].



شکل ۲-۱۰) مسیریابی توسط الگوریتمWireless

۲-۶ کارهای مرتبط

روش‌های مسیریابی ذکر شده در بالا، بر اساس یک الگوریتم مشخص عمل می‌کنند که مخصوصاً برای ترافیک‌های Hotspot و Transpose دچار ازدحام می‌شوند. بنابراین لازم است از یادگیری تقویتی استفاده شود که یکی از مهم‌ترین الگوریتم‌های مسیریابی است و می‌توان آن را جزء دسته‌ی انطباقی قرار داد. مراجع زیر از الگوریتم‌های مبتنی بر یادگیری تقویتی استفاده کرده‌اند:

در مرجع [۱۲] برای مسیریابی در یک شبکه‌ی سیمی از روش یادگیری تقویتی استفاده شده است. در مدل یادگیری تقویتی شبکه، وضعیت[۱] همان گره، عامل[۲] همان بسته و عمل[۳] انتخاب پورت خروجی است. در این مقاله از تکنیک یادگیری Q[۴] استفاده شده است؛ به این صورت که برای هر گره یک جدول قرار داده که ستون‌هایش عمل و سطرهایش وضعیت است و مقادیر این جدول معادل تأخیری است که صرف می‌شود تا گره از مبدأ به مقصد برسد؛ در نتیجه در هر حالت کمترین مقدار انتخاب می‌شود، با توجه به اینکه مقدار تأخیر مثبت در نظر گرفته شده است. وقتی بسته‌ای به گره‌ای می‌رسد، این جدول بررسی می‌شود و با توجه به نوع عمل که مشخص شده، پورت خروجی انتخاب می‌شود، با توجه به آن پورت وضعیت بعدی هم مشخص می‌شود. اگر پورتِ انتخاب شده در جهت کاهش تأخیر باشد، پاداش[۵] داده می‌شود تا مورد تأیید قرار گیرد؛ و اگر پورتِ انتخاب شده در جهت افزایش تأخیر باشد، برایش تنبیه[۶] در نظر گرفته می‌شود. وقتی بسته‌ای به یک گره ارسال می‌شود، گرهِ فرستنده اطلاعاتی در مورد وضعیت گره‌ها یا جدولش را به گره گیرنده ارسال می‌کند تا گره گیرنده بتواند جدول خودش را به‌روزرسانی کند. گره گیرنده هم پس از دریافتِ بسته، اطلاعاتی در مورد وضعیت خود، با بسته‌ی یادگیری به گره فرستنده ارسال می‌کند تا گره فرستنده هم از این اطلاعات استفاده کند. پس در واقع می‌توان گفت یک ارتباط دوطرفه ایجاد شده است و هر دو گره گیرنده و فرستنده جداول خود را با اطلاعات ارسالی به‌روز می‌کنند. از مدل یادگیری تقویتیِ دوطرفه اولین بار در ارتباطات ماهواره‌ای استفاده شده است که در آن، دو انتهای سیستم ارتباطی با استفاده از سیگنال تقویت کننده برای انتهای دیگر، خود را سازگار می‌کنند.

در مرجع [۹] به حل مشکل اساسی شبکه‌های روی تراشه پرداخته شده است که این مشکل اصلی عبارت است از ازدحام در شبکه، به طوری که وقتی یک بسته ارسال می‌شود اگر در این مسیرهای پر ازدحام

---

۱ State

۲ Agent

۳ Action

۴ Q-Learning

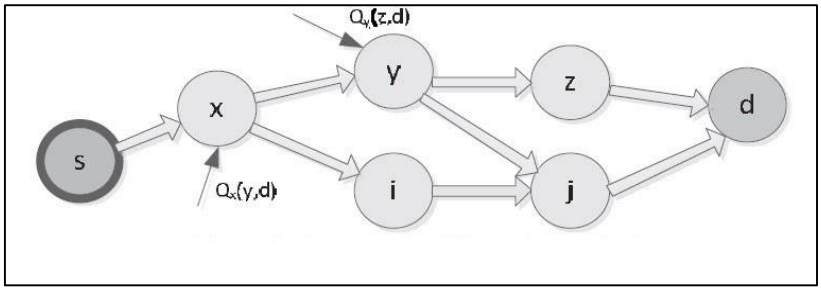
۵ Reward

۶ Punishment

قرار بگیرد ممکن است به مقصد نرسد و یا دیر برسد. این خیلی مهم است که بتوانیم مسیرهای کم ازدحام‌تر را برای ارسال بسته به گره مقصد انتخاب کنیم.

برای این کار راه‌های مختلفی وجود دارد. یکی از این راه‌ها استفاده از یادگیری تقویتی است که جزء الگوریتم‌های مسیریابی انطباقی است. در روش ارائه شده در این مقاله، همان‌طور که در مرجع [۱۲] گفته شد، برای هر گره یک جدول وجود دارد که حالت‌های مختلف حرکت بسته را از آن گره در شبکه نشان می‌دهد و مقادیر این جدول معادل تأخیری است که صرف می‌شود تا گره از مبدأ به مقصد برسد.

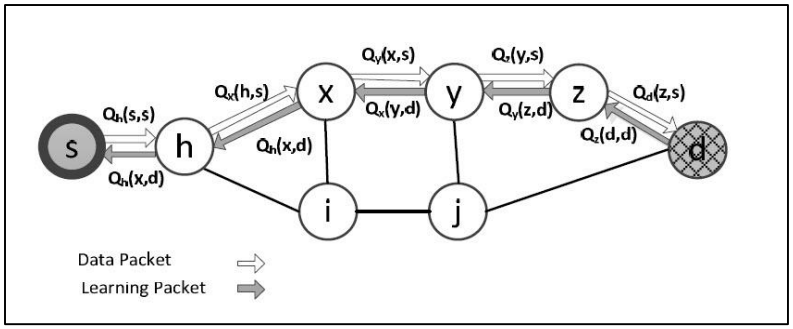
وقتی بسته‌ای به یک گره می‌رسد جدول را بررسی می‌کند، چون هدف انتخاب کم ازدحام‌ترین مسیر است. در صورتی که مقدار تأخیر مثبت فرض شود، پایین‌ترین مقدار را انتخاب می‌کند و بسته را در آن مسیر ارسال می‌کند. وقتی بسته از طریق این گره ارسال شد و گره گیرنده آن را دریافت کرد، گره گیرنده اطلاعات خودش را در مورد وضعیت پورت‌هایش، از طریق بسته یادگیری برای گره فرستنده می‌فرستد و گره فرستنده با استفاده از این اطلاعات رسیده، جدولش را به‌روزرسانی می‌کند. این مسیریابی یک مسیریابی یک‌طرفه است، چرا که وقتی بسته‌ای از گره x به گره y ارسال می‌شود فقط جدول گره فرستنده یعنی گره x به‌روزرسانی می‌شود. در شکل ۲-۱۱ نمونه‌ای از مسیریابی یک‌طرفه نشان داده شده است.



شکل ۲-۱۱) نمونه‌ای از مسیریابی یک‌طرفه [۹]

در مرجع [۱۰] برای حل مشکل ترافیک در شبکه‌های روی تراشه راه‌حل دیگری ارائه شده است که این راه‌حل بر خلاف راه‌حل مرجع [۹] (که مبتنی بر یادگیری تقویتی به صورت یک‌طرفه است) عبارت است از استفاده

از یادگیری تقویتی به صورت دوطرفه. در این روش پس از انتخاب مسیر، در هنگام ارسال بسته، گره فرستنده اطلاعاتی از وضعیت ازدحام در پورت‌های خود را نیز ارسال می‌کند. این اطلاعات برای به‌روزرسانی جدول یادگیری گره گیرنده استفاده می‌شود. وقتی بسته با این اطلاعات اضافه شده ارسال می‌شود، گره گیرنده این اطلاعات را دریافت می‌کند و اطلاعات مورد نیاز را از بسته‌ی رسیده شده استخراج می‌کند و اطلاعات مربوط به خود را با بسته‌ی یادگیری به گره فرستنده ارسال می‌کند. با این اطلاعاتی که گره گیرنده و گره فرستنده دریافت می‌کنند، جداول مربوط به خودشان را به‌روزرسانی می‌کنند. به همین دلیل به این روش، دوطرفه گفته می‌شود. روش ارائه شده در این مقاله مسیر بهینه‌تر را سریع‌تر و بهتر پیدا می‌کند و کارائی شبکه را بیشتر بهبود می‌بخشد. در شکل ۲-۱۲ نمونه‌ای از مسیریابی دوطرفه نشان داده شده است.



شکل ۲-۱۲) نمونه‌ای از مسیریابی دوطرفه [۱۰]

در مرجع [۸]، با استفاده از یادگیری تقویتی به بهینه‌سازی عملکرد شبکه‌ی روی تراشه پرداخته شده است.

در مرجع [۴۴]، یک الگوریتم مسیریابی تطبیقی مبتنی بر مسیریابی Q ارائه شده که با استفاده از یک روش یادگیری در کل شبکه، ترافیک را توزیع می‌کند.

لازم به ذکر است که در تمامی الگوریتم‌های مسیریابی بر اساس یادگیری تقویتیِ ذکر شده، بین گره مبدأ و گره مقصد، به جای فلیت، بسته جابجا شده است و نیاز به روشی برای جابجایی فلیت است و همچنین انتقال داده در یک شبکه‌ی سیمی بوده است .

در مرجع [۵۹]، یک الگوریتم مسیریابیِ آگاه از ازدحام و مبتنی بر یادگیری Q ارائه شده است که این الگوریتم شبکه را به چندین خوشه تقسیم می‌کند و هر خوشه یک جدول را حفظ می کند.
این جدول اطلاعات مربوط به ازدحام محلی و کلی را در مورد مسیرهای جایگزین برای ارسال بسته به خوشه‌ی مقصد ذخیره می کند. هر خوشه می تواند کانال خروجی کم تراکم را بر اساس اطلاعات جدول انتخاب کند. علاوه بر این، برای به روزرسانی بیشتر جدول‌ها، هر دو بسته یادگیری و داده‌ها در انتشار اطلاعات ازدحام شرکت می کنند. در این مرجع برای انتقال داده از فلیت استفاده شده است، ولی از لینک‌های بی‌سیم استفاده نشده است.

۲-۷ نتیجه‌گیری

با بررسیِ منطق الگوریتم‌های مسیریابیِ موجود و بررسی مقالاتی که مسیریابی در شبکه‌ی روی تراشه را بر اساس یادگیری تقویتی انجام داده‌اند و درک مزایا و معایب هر روش به این موضوع پی برده می‌شود که مشکلات به کمک الگوریتم یادگیری تقویتی تا حدودی کاهش یافته است، اما در این روش‌ها هنوز مشکلاتی به چشم می‌خورد که از آن جمله می‌توان به استفاده از بسته‌ها به جای فلیت، استفاده از روش سیمی به جای بی‌سیم و به وجود آمدن بن‌بست نام برد. به همین دلیل در این پایان‌نامه سعی شده تا این مشکلات در الگوریتم مسیریابی برطرف شود. به این منظور در فصل بعد به بررسی الگوریتم پیشنهادی پرداخته شده است.

فصل ۳: روش پیشنهادی

۳-۱ مقدمه

در این فصل ابتدا به توضیح یادگیری تقویتی و انواع روش‌های یادگیری تقویتی پرداخته شده است. سپس به دلیل استفاده از الگوریتم یادگیری Q در پیاده‌سازی، این الگوریتم و کد آن مورد بررسی قرار گرفته است. در انتهای این بخش، مزایا و معایب این روش بیان شده است. در بخش بعدی این فصل، روش پیشنهادی مورد بحث و بررسی قرار گرفته است و روش‌های پیاده‌سازی و توابع مورد استفاده در آن به طور مختصر توضیح داده شده است و سعی شده است تا روال پیاده‌سازی به صورت گام به گام بیان شود.

۳-۲ یادگیری تقویتی

هوش مصنوعی، علم مطالعه و بررسی ماشین‌هایی است که رفتار هوشمندانه دارند. مفهوم هوشمندی، چیزی است که به سختی قابل تعریف است، اما معمولاً هوشمندی با توانایی یادگیری از طریق تجربه مرتبط است. یادگیری ماشین، زمینه‌ای مطالعاتی در هوش مصنوعی است که به دنبال ایجاد عواملی همچون برنامه‌های کامپیوتری است که بتوانند با استفاده از تجربیات خود، چیزها را یاد بگیرند. ایده‌ی ایجاد ماشین‌هایی که رفتار هوشمندانه‌ی انسان را تقلید کنند، بدون الهام گرفتن از هوشمندی انسان هیچ‌گاه کامل نخواهد بود و نتیجه‌ای نخواهد داشت.

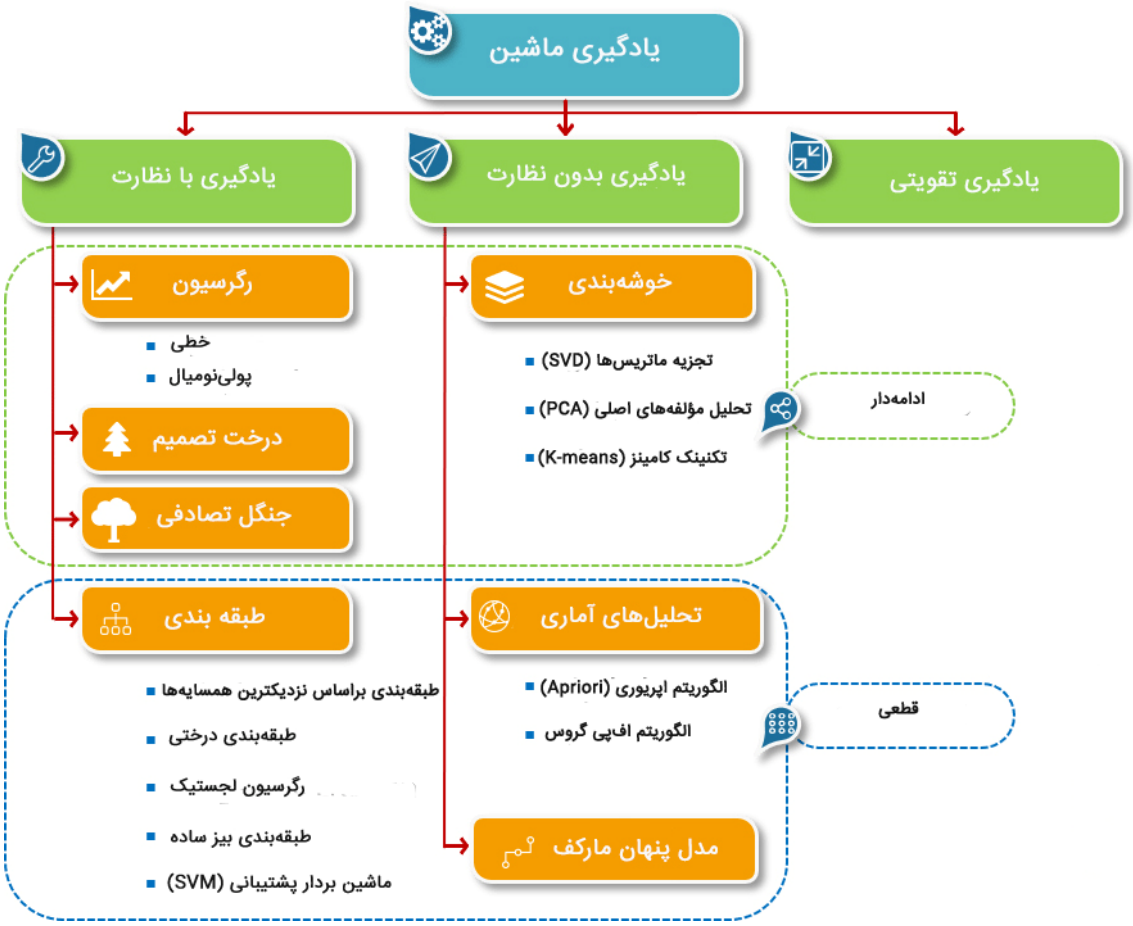
با گذشت زمان، هوش مصنوعی و یادگیری ماشین در حال تبدیل شدن به رشته‌های مهندسی هستند و به ندرت به عنوان علوم طبیعی به آن‌ها پرداخته می‌شود. در عصر صنعت، ما به دنبال ساخت ماشین‌هایی هستیم که کار فیزیکی انجام دهند، اما در عصر اطلاعات، در پی ساخت ماشین‌هایی هستیم که فکر کنند. ایجاد چنین ماشین‌هایی بسیار دشوار است، زیرا اطلاع دقیقی از منابع هوشمندی انسان (که مورد نیاز ساخت چنین ماشین‌هایی است) در دست نیست و طبعا نمی‌توان ماشین‌هایی هوشمند و نظیر انسان تولید کرد.
مهم‌ترین کاری که غالباً یک سیستم هوشمند انجام می‌دهد، یادگیری است و اگر یک سیستم هوشمند بتواند یادگیری انسان را شبیه‌سازی کند، تا حدود زیادی در پیاده‌سازی هوش مصنوعی موفق خواهد بود. هدف اصلی از یادگیری، یافتن شیوه‌ای برای عملکرد حالات مختلف است که این شیوه در مقایسه با سایر روش‌های

یادگیری با در نظر گرفتن معیارهایی بهتر است. معمولاً این شیوه‌ی عملکرد از نظر ریاضی به صورت نگاشتی از فضای حالات به فضای اعمال، قابل بیان است. هنگامی می‌توان گفت یادگیری اتفاق افتاده است که عامل بر اساس تجربیاتی که کسب می‌کند، بهتر عمل کند. در این صورت باید نحوه‌ی عملکردِ عامل در اثر کسب اطلاعات جدید، متفاوت از نحوه‌ی عملکرد آن قبل از کسب این اطلاعات و تجارب باشد.

۳-۳ انواع یادگیری ماشین

زیرمجموعه‌ها و انواع سیستم‌های یادگیری ماشین به سه دسته تقسیم می‌شوند:

- یادگیری با نظارت
- یادگیری بدون نظارت
- یادگیری تقویتی



شکل ۳-۱) انواع یادگیری ماشین

یادگیری با نظارت[۱] [۱۳]: این نوع یادگیری، مرسوم‌ترین نوع یادگیری در سیستم‌های آموزشی انسان‌ها است؛ اما با این حال، چیزی نیست که در طبیعت رایج باشد. در این نوع یادگیری، حضور یک خبره، معلم، ناظر و یا داده‌هایی حاوی دانش وی، ضروری است؛ به عبارتی این نوع یادگیری زمانی رخ می‌دهد که مقادیر ورودی ماشین به خوبی برچسب‌گذاری۲ شده باشد؛ یعنی پاسخ‌های درست متناظر با آن‌ها از قبل مشخص شده باشد. به عنوان مثال یک مجموعه‌ی ورودی عددی (مثلاً اعداد طبیعی) و یک مجموعه‌ی جواب متناظر (مثلاً ریشه‌ی سوم متناظر با تک تک آن اعداد طبیعی) به ماشین داده می‌شود. سپس با آموزش، ماشین یاد می‌گیرد که هر کدام از جواب‌ها ریشه‌ی سوم ورودی‌ها است. حال اگر یک ورودی کاملاً جدید به ماشین داده شود، بر اساس تجربه‌ی یادگیری قبلی، او با احتمالی از صحت۳ می‌تواند جواب متناظر را تشخیص دهد. واضح است که ارائه‌ی داده‌های آموزشیِ بیشتر به ماشین، باعث نمی‌شود ماشین با احتمال صحت بیشتری جواب ورودی‌های جدید را مشخص کند.

یادگیری بدون نظارت[۴]: این شیوه‌ی یادگیری، در بسیاری از موجودات و در برهه‌های مختلف زندگی انسان‌ها دیده می‌شود و یکی از سخت‌ترین نوع روش‌های یادگیری است. در این نوع یادگیری، نیازی به حضور ناظر، معلم یا خبره نیست. مثلاً فرض کنید که به طور مرتب در معرض اطلاعاتی (مثل واژه‌های یک زبان خارجی) قرار دارید که هیچ دانشِ قبلی از آن‌ها ندارید و معلم و آموزنده‌ای هم نیست که در خصوص آن‌ها چیزی یاد دهد. در روزهای اول از مفاهیم و معانی واژه‌ها هیچ چیز تشخیص داده نشود، ولی به مرور زمان مغز شروع به یادگیری می‌کند. به طوری که اگر یک واژه را ببیند می‌تواند صوت متناسب با آن را حدس بزند. این دقیقاً چیزی است که در یادگیری بدون نظارت اتفاق می‌افتد.

---

۱ Supervised Learning

۲ Labeled

۳ Accuracy

٤ Unsupervised Learning

بنابراین می‌توان گفت که این نوع یادگیری زمانی رخ می‌دهد که داده‌های ورودی برچسب‌گذاری نشده‌اند؛ یعنی جواب متناظر با آن‌ها برای ماشین تعریف نشده است و ماشین فقط از طریق تحلیل ورودی‌ها و یافتن الگوهایی در بین آن‌ها، می‌تواند آن‌ها را تحلیل کند.

یادگیری تقویتی[1]: شاید علم بیش از هر چیزی توسط یادگیری تقویتی توسعه پیدا کرده باشد. مثلاً یک مهندس را در نظر بگیرید که می‌خواهد یک موتور را طراحی کند که سرعت چرخش موتور آن به مقدار مشخصی مثلا ۱۰۰ تا ۱۵۰ دور در دقیقه (RPM) برسد. مهندس با آزمودن روش‌های مختلف به‌مرور می‌فهمد که چه اقدامات و طراحی‌هایی را نباید انجام دهد و چه اقدامات و طراحی‌هایی را باید انجام دهد تا به هدف مورد نظرش برسد. این چیزی است که در روح یادگیریِ تقویتی نهفته است. در بسیاری از حیوانات، یادگیری تقویتی، تنها شیوه‌ی یادگیری است. همچنین یادگیری تقویتی، بخش اساسی از رفتار انسان‌ها را تشکیل می‌دهد. هنگامی که دست در مواجهه با حرارت می‌سوزد، به سرعت یاد می‌گیریم که چه کاری بکند و چه کاری نکند تا سوختن دست برایش تکرار نشود. لذت و درد مثال‌های خوبی از پاداش‌ها هستند که الگوهای رفتاری انسان و بسیاری از حیوانات را تشکیل می‌دهد. در یادگیری تقویتی، هدف اصلی از یادگیری، انجام دادن کاری و یا رسیدن به هدفی است، بدون آن‌که عامل یادگیرنده، با اطلاعات مستقیم بیرونی تغذیه شود. در یادگیری تقویتی، وقتی عامل، کاری را انجام می‌دهد که او را به هدفش نزدیک‌تر می‌سازد، پاداش می‌گیرد و هدف این است که اقداماتی را انجام دهد تا در دراز مدت پاداش او را حداکثر سازد [۱۴، ۱۵]. اگرچه، هم یادگیری نظارت شده و هم یادگیری تقویتی از نگاشت بین ورودی و خروجی استفاده می‌کنند، اما در یادگیری تقویتی بر خلاف یادگیری نظارت شده از پاداش‌ها و تنبیه‌ها به عنوان سیگنال‌هایی برای بهبود عملکرد نهایی سیستم استفاده می‌شود [۲۵، ۲۶]. ضمن این‌که تفاوت اصلی میان یادگیری تقویتی با روش‌های نظارتی یادگیری ماشین، در این است که در یادگیری تقویتی، هیچ‌گاه به عامل گفته نمی‌شود که عمل صحیح در هر وضعیت چیست و فقط به وسیله‌ی معیاری، عامل متوجه می‌شود که یک عمل چقدر خوب و چقدر بد است. این وظیفه‌ی عاملِ یادگیرنده است که با در دست داشتن این اطلاعات، یاد بگیرد که بهترین عمل در هر

[1] Reinforcement Learning

وضعیت کدام است. این موضوع، بخشی از نقاط قوت خاص یادگیری تقویتی است. به کمک یادگیری تقویتی، غالباً پیچیدگی‌های تصمیم‌گیری می‌تواند با فراهم شدن کمترین میزان اطلاعاتِ مورد نیاز حل شود.

۳-۴ اجزای اصلی یادگیری تقویتی

یادگیری تقویتی چهار جزء اصلی و اساسی دارد که عبارت است از سیاست، تابع پاداش، تابع ارزش‌گذاری و محیط. این اجزا ذیلاً بررسی می‌شود:

۳-۴-۱ سیاست[۱]

شیوه‌ی رفتار کردن عامل را سیاست تعریف می‌کند. در واقع سیاست، نگاشتی بین وضعیت دریافت شده از محیط و عمل‌های قابل انجام در آن وضعیت است. این نگاشت به عامل می‌گوید که در مواجهه با حالات مختلف، چه عمل یا اعمالی را انجام دهد. پیروی از یک سیاست خوب، قطعاً عامل را به نتیجه‌های مناسب خواهد رساند. به بیانی دیگر، سیاست (π) تابعی است که احتمال انتخاب شدن هر عمل را در هر حالت و با توجه به گام زمانی محاسبه می‌کند. به طور مثال $\pi_t (s,a) = p$ به این معناست که اگر عامل در زمان t و در حالت s قرار گرفته باشد، با احتمال p، عمل a را انتخاب می‌کند. علاوه بر این، در بعضی موارد، سیاست می‌تواند یک جدول ساده‌ی جستجو یا فرآیندهای سنگین جستجو باشد. سیاست، هسته‌ی یادگیری تقویتی بوده و به تنهایی برای تعیین رفتار عامل کافی است.

۳-۴-۲ تابع پاداش[۲]

این تابع، حالت (حالت-عمل) دریافت شده از محیط را توسط یک نگاشت به یک سیگنال عددی به نام پاداش مرتبط می‌سازد. این تابع به کمک پاداش یا جریمه تعیین می‌کند که کدام عمل برای عامل، خوب و کدام عمل بد است. هدف اصلیِ عامل این است که پاداش‌هایی که در طولانی مدت به دست می‌آورد، بیشینه شود. همچنین ممکن است از این تابع به عنوان مبنایی برای تعویض سیاست استفاده کند؛ مثلاً اگر یک عمل انتخاب

---

۱ Policy

۲ Reward

شده توسط سیاستی پاداش کمی به همراه داشته باشد، این سیاست ممکن است عوض شود تا در آینده عمل دیگری در آن وضعیت انتخاب گردد.

۳-۴-۳ تابع ارزش‌گذاری[۱]

تابع ارزش‌گذاری برای هر حالت مقداری را مشخص می‌کند که هر چه این مقدار بیشتر باشد، مفهومش این است که عامل به هدف نزدیک‌تر شده است. این تابع نگاه بلند مدت دارد؛ مانند این‌که در یک بازی به حریف اجازه دهید مهره‌ی شما را بزند. در این حالت پاداش نمی‌گیرید، ولی به حالت دیگری می‌روید که بهتر است.

ارزش یک حالت برابر است با مجموع مقادیر پاداش دریافتی با شروع از آن حالت و پیروی از سیاست مشخصی که به یک حالت پایانی (هدف) ختم می‌شود که در رابطه‌ی (۳-۱) نشان داده شده است.

$$V^{\pi}(s)=E_{\pi}\{R_t|S_t=s\}=E_{\pi}\{\sum\gamma^k r_{t+k+1}|S_t=s\} \qquad (۱-۳)$$

در رابطه‌ی (۳-۱)، γ عامل تخفیف و R_t مجموع پاداش‌های دریافتی با شروع از حالت s می‌باشد. تابع ارزش عبارت است از نگاشتی میان حالت و ارزش که توسط هر تقریب زننده‌ی تابع، نظیر یک شبکه‌ی عصبی تخمین زده شود. یکی از خواص بنیادی توابع ارزش، به دست آوردن رابطه‌ی (۳-۲) به صورت بازگشتی می‌باشد که تحت عنوان معادله بلمن شناخته می‌شود.

$$Q(s_t,a_t)\leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}}+\underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}}\times\left[\overbrace{\underbrace{R(s_t)}_{\text{reward}}+\underbrace{\gamma}_{\text{discount factor}}\underbrace{\max_{a_{t+1}}Q(s_{t+1},a_{t+1})}_{\text{max future value}}}^{\text{learned value}}-\underbrace{Q(s_t,a_t)}_{\text{old value}}\right] \qquad (۲-۳)$$

در رابطه‌ی (۳-۲)، s_t حالت فعلی، s_{t+1} حالت بعدی، a_t عمل فعلی و a_{t+1} عمل بعدی می‌باشد. این رابطه نشان دهنده‌ی رابطه‌ی میان ارزش حالت فعلی و ارزش حالت بعدی است. پارامترهای این تابع در انتهای فصل توضیح داده شده است.

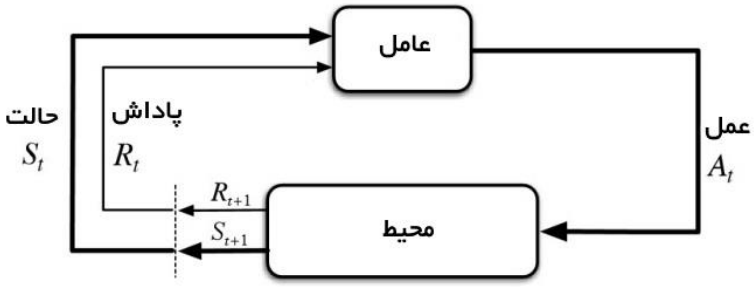
[۱] Value Function

تابع دیگری به نام $Q^{\pi}(s, a)$ وجود دارد که بیان‌گر مجموع پاداش‌هایی است که عامل با شروع از حالت s و انجام عمل a و در پیش گرفتن سیاست π به دست می‌آورد. این تابع، تابع ارزش-عمل برای سیاست π نام دارد.

۳-۴-۴ محیط[۱]

عامل یادگیرنده، با سعی و خطا با یک محیطِ پویا درگیر شده و یاد می‌گیرد که برای هر وضعیت چه عملی انجام دهد. در واقع، هر چیزی که خارج از عامل است و عامل با آن تعامل دارد، محیط نامیده می‌شود. محیط باید کاملاً یا لااقل قسمتی از آن قابل مشاهده باشد.

مشاهده‌ی محیط ممکن است از طریق خواندن اطلاعات یک حس‌گر و توضیح نمادین و ... باشد. در حالت ایده‌آل، عامل باید به طور کامل قادر به مشاهده‌ی محیط باشد و اغلب تئوری‌ها نیز بر اساس همین فرض بنا شده‌اند. شکل ۳-۲ شمای کلی یادگیری تقویتی را نشان می‌دهد.



شکل ۳-۲) شمای کلی یادگیری تقویتی

۳-۵ خاصیت مارکوف

در یادگیری تقویتی، عامل بر اساس سیگنال دریافتی از محیط که به آن، حالت محیط گفته می‌شود تصمیم می‌گیرد. فرض کنید سیگنال s_t نشان دهنده‌ی حالت محیط در لحظه t است.

مطلوب این است که s_t تمام داده‌های مفید مربوط به حال و گذشته را در خود خلاصه کند. برای رسیدن به این هدف، چیزی فراتر از یک درک آنی لازم است، ولی به معنای دارا بودن تمام تاریخ گذشته نیز

[۱] Environment

نمی‌باشد. به حالتی که چنین باشد، خاصیت مارکوف گفته می‌شود. به عنوان مثال، چیدمان مهره‌ها روی صفحه‌ی شطرنج دارای خاصیت مارکوف است؛ زیرا با وجود این که تمام حرکت‌هایی که از اول بازی انجام شده است را به بازیکن نشان نمی‌دهد، ولی تمام داده‌های مفید برای ادامه‌ی بازی را در خود دارد؛ از سیگنالِ حالت نباید انتظار داشت تمام داده‌های محیط را برای تصمیم‌گیری عامل نشان دهد. برای مثال، اگر عامل به تماس-های دریافتی پاسخ می‌دهد، نباید انتظار داشت اطلاعاتی در مورد تماس گیرنده داشته باشد.

۳-۶ فرآیند تصمیم‌گیری مارکوف[۱]

به مسئله‌ی یادگیری تقویتی که در آن خاصیت مارکوف برقرار باشد، فرآیند تصمیم‌گیری مارکوف گفته می‌شود. اگر فضای حالات و مجموعه عمل‌های ممکن متناهی باشد، به فرآیند تصمیم‌گیری، مارکوف متناهی گفته می‌شود.

به عبارت دیگر، فرآیند تصمیم‌گیری مارکوف، مدلی برای تصمیم‌گیری‌های ترتیبی می‌باشد و زمانی مورد استفاده قرار می‌گیرد که خروجی‌ها غیرقطعی باشد. منظور از تصمیم‌گیری‌های ترتیبی این است که کارایی عامل به مجموعه‌ای از تصمیمات که به صورت متوالی گرفته شده، بستگی دارد و تنها وابسته به تصمیم فعلی او نخواهد بود.

فرآیند تصمیم‌گیری مارکوف، در حالت کلی به صورت یک مجموعه‌ی شش‌تایی ($S, A, T, Next, R, \alpha$) نشان داده می‌شود که S مجموعه‌ی وضعیت‌های ممکن در محیط، A مجموعه‌ی عمل‌های ممکن برای عامل در هر مرحله از تصمیم‌گیری، T نمایان‌گر احتمال عبور عامل از وضعیت s به s' به شرط انتخاب عمل a از مجموعه اعمال ممکن، $Next$ نشان دهنده‌ی مجموعه‌ی وضعیت‌هایی که می‌توان در یک قدم با احتمال غیرصفر و انتخاب عمل a به آن رسید، $R(s,a)$ بیانگر مقدار پاداشی که عامل به ازای انجام عمل a در وضعیت s از محیط دریافت می‌کند و α فاکتور کاهنده می‌باشد.

۱ Markov Decision Processes

۳-۷ کاربردهای یادگیری تقویتی

از گذشته تا به حال از یادگیری تقویتی در زمینه‌های مختلفی استفاده شده است که نمونه‌هایی از آن ارائه می‌شود:

- **سامانه‌های چندعامله**: این سامانه‌ها نمونه‌هایی از هوش مصنوعی توزیع شده هستند که با ساختاری متشکل از چند عامل که رفتارهای عامل‌ها باید در یک محیط به سمت یک هدف جامع با هم هماهنگ شود، بیان می‌شوند. همچنین کنترل سامانه‌های چندعامله به صورت متمرکز و توزیع‌شده است. فرآیند تصمیم‌گیری هر عامل باید به دست خودش انجام شود که البته این تصمیم‌گیری روی مشاهدات و دانش‌هایی که درباره‌ی محیط و عامل‌های دیگر بنا شده است، انجام می‌گیرد.
- **بازی‌ها:** از یادگیری تقویتی برای چندین بازی اصلی مثل شطرنج و تخته نرد که در جهان مطرح هستند استفاده شده است. در سال ۱۹۵۹ یک سامانه ارائه شد که به چکرزها یاد داده می‌شود که از راه بازی کردن با خودشان بازی کنند. این روش بعدها با ایده‌ی یادگیری به نام یادگیری TD ترکیب شد و توانست قدرت بازی چکرزها را افزایش دهد.
- **هدایت کردن ربات:** هدایت خودکار ربات‌ها در محیط‌های ناشناخته یک بستر آزمایش برای تحقیقات زیاد در زمینه‌ی یادگیری است. در سال ۱۹۸۳ یک ربات قادر شد به کمک این یادگیری به اهداف تعیین شده با فاصله گرفتن از موانع برسد.
- **زمان‌بندی:** یک استفاده‌ی دیگر از یادگیری تقویتی، زمان‌بندی در کارها است که نخست در سال ۱۹۹۶ مورد بررسی قرار گرفت. با موفقیت دیدگاه یادگیری TD در مسائل زمان‌بندی، این روش یادگیری در این زمینه هم مطرح شد.
- **مسیریابی:** مسیریابی در شبکه‌هایی با شرایط پویا می‌تواند با مسیریابی Q[۱] به‌خوبی انجام گیرد که یک روش یادگیری تقویتی توزیع شده است.

[۱] Q-Routing

۳-۸ روش‌های یادگیری تقویتی

- **برنامه‌ریزی پویا:** به مجموعه‌ی الگوریتم‌هایی گفته می‌شود که با در دست داشتن مدل کاملی از محیط به عنوان فرآیند تصمیم‌گیری مارکوف، می‌توانند سیاست بهینه را محاسبه کنند.
- **مونت کارلو**: این روش نیازی به شناخت کامل محیط ندارد و مسئله‌ی یادگیری تقویتی را بر اساس میانگین بازده‌ها حل می‌کند. برای این منظور (به دست آوردن بازده مناسب و درست)، مونت کارلو فقط بر روی وظایف دوره‌ای تعریف می‌شود؛ بنابراین یک وظیفه به چند دوره‌ی زمانی تقسیم شده و هر دوره در نهایت به پایان می‌رسد و فقط در انتهای یک دوره‌ی زمانی است که تخمین مقادیر و ارزش‌ها صورت گرفته و سیاست‌ها انتخاب می‌شود.
- **تفاضل زمانی**: این روش، هسته‌ی اصلی یادگیری تقویتی است. این روش ترکیبی از برنامه‌ریزی پویا و روش مونت کارلو است؛ یعنی عامل می‌تواند مانند روش مونت کارلو، بدون نیاز به مدلی از محیط پویا، به صورت مستقیم از تجربه‌ها یاد بگیرد و همانند برنامه‌ریزی پویا بر اساس یادگیری‌های دیگرش و بدون انتظار برای رسیدن به یک نتیجه‌ی نهایی برآورد خود را به‌روزرسانی نماید.

۳-۹ الگوریتم‌های یادگیری تقویتی

۳-۹-۱ الگوریتم سارسا[1]

این الگوریتم در سال ۱۹۹۴ مطرح گردید. همان‌طور که از نام الگوریتم (حالت، عمل، پاداش، حالت، عمل) مشخص است، عامل در حالت جاری s_t، عمل a_t را انتخاب می‌کند و با دریافت پاداش r، به حالت جدید s_{t+1} می‌رود و عمل a_{t+1} را انتخاب می‌کند $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$

محاسبه‌ی تابع ارزش-عمل و به‌روزرسانی آن مطابق رابطه‌ی ۳-۳ صورت می‌گیرد.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \qquad (۳-۳)$$

[1] State-Action-Reward-State-Action

به‌روزرسانی پس از هر حالت غیرپایانی s_t صورت می‌گیرد. اگر حالت s_{t+1} حالت پایانی باشد، مقدار تابع $Q(s_{t+1},a_{t+1})$ تغییری نکرده و مقدار اولیه‌ی خود را حفظ خواهد کرد. این قانون برای هر پنج‌تایی $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ که در نهایت منجر به انتقال از حالت-عمل به حالت-عمل بعدی می‌گردد، استفاده می‌شود. در این رابطه، متغیر نرخ یادگیری α، تعیین می‌کند که تا چه میزان اطلاعات به دست آمده‌ی جدید بر اطلاعات قدیمی ترجیح داده شود. مقدار صفر باعث می‌شود که عامل چیزی یاد نگیرد و مقدار یک باعث می‌شود که عامل فقط اطلاعات جدید را ملاک قرار دهد. همچنین متغیر عامل تخفیف γ، اهمیت پاداش‌های آینده را تعیین می‌کند. مقدار صفر باعث می‌شود که عامل ماهیت فرصت طلبانه گرفته و فقط پاداش‌های فعلی را مد نظر قرار دهد. وقتی که نرخ تخفیف معادل یک قرار گیرد، عامل ترغیب شده و در یک دوره‌ی زمانی طولانی برای کسب پاداش تلاش می‌کند. الگوریتم ۳-۱ این روال را نشان می‌دهد.

الگوریتم ۳-۱) الگوریتم سارسا

1. Initialize Q(s,a) arbitarily
2. Repeat(for each episode)
3. Initialize s
4. Choose a from s using policy derived from Q(e.g, €-greedy)
5. Repeat (for each step of episode)
6. Take a action, observe r, s'
7. Choose a' from s' using policy derived from Q(e.g, €-greedy)
8. $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$.
9. s←s'; a←a';
10. Until s is terminal[4];

شرح الگوریتم:

- به ازای هر عمل a، به Q(s,a) مقدار اولیه اختصاص بده (معمولاً صفر).

- در هر بازه‌ی زمانی تکرار کن.
  - حالت فعلی s را مشاهده کن.
  - عمل a را از حالت فعلی s انتخاب کن (با سیاست جاری).
  - تا مشاهده‌ی حالت پایانی تکرار کن.
    - عمل a را اجرا کن، پاداش r و حالت جدید s' را مشاهده کن.
    - یک عمل ممکن از حالت جدید s' را انتخاب کن (با سیاست جاری).
    - مقدار Q را برای s و a به‌روز کن.
    - حالت جدید s' را به عنوان حالت فعلی s و عمل جدید a' را به‌عنوان عمل فعلی a در نظر بگیر.

باید توجه داشت که الگوریتم سارسا با احتمال یک، به سیاست بهینه همگرا خواهد شد و تابع ارزش-عمل $Q(s,a)$ برای هر جفت حالت-عمل در تعداد تکرار نامتناهی به دست خواهد آمد [۵۳-۵۵].

۳-۹-۲ الگوریتم یادگیری Q

یادگیری Q یک تکنیک یادگیری تقویتی است که با یادگیری یک تابع حالت-عمل، سیاست مشخصی را برای انجام حرکات مختلف در وضعیت‌های مختلف دنبال می‌کند. یکی از نقاط قوت این روش، توانایی یادگیری تابع مذکور بدون داشتن مدل معینی از محیط است.

در الگوریتم یادگیری Q، به هر زوج حالت-عمل یک مقدار$Q(s,a)$ نسبت داده می‌شود که این مقدار عبارت است از مجموع پاداش‌های دریافتی. وقتی عامل از حالت s شروع می‌کند، عمل a را انجام می‌دهد و سیاست موجود را پیروی می‌کند، تا زمانی که به مقدار بهینه همگرا شود، از رابطه‌ی (۳-۴) برای به‌روزرسانی استفاده می‌کند.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R(s_t)}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{max future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right] \quad (۴-۳)$$

هسته‌ی الگوریتم از یک به‌روزرسانی تکراری ساده تشکیل شده است. به این ترتیب که مقادیر قبلی بر اساس اطلاعات جدید اصلاح می‌شود. همچنین در هر بازه‌ی زمانی، زمانی‌که s_{t+1}، یک حالت نهایی باشد، مقدار تابع Q برای آن هیچ‌گاه به‌روز نمی‌شود و مقدار اولیه‌ی خود را حفظ می‌کند.

در یادگیری Q، از جداول برای ذخیره‌ی داده استفاده می‌شود. ضمن این که در یادگیری Q به‌جای انجام یک نگاشت از حالت‌ها به مقادیر حالت‌ها، نگاشتی از زوج حالت-عمل به مقداری که «مقدار Q» نامیده می‌شود ایجاد می‌شود. در واقع این الگوریتم نوعی از یادگیری تقویتی بدون مدل است که بر پایه‌ی برنامه‌ریزی پویا عمل می‌کند. این روش با پیچیده شدن سیستم، به سرعت کارایی خود را از دست می‌دهد [۲۶]. الگوریتم ۳-۲ این روال را نشان می‌دهد.

الگوریتم ۳-۲) الگوریتم یادگیری Q

1. Initialize Q(s,a) arbitarily
2. Repeat(for each episode)
3. Initialize s
4. Repeat (for each step of episode)
5. Choose a from s using policy derived from Q(e.g, €-greedy)
6. Take action a, observe r, s'
7. $Q(s_t, a_t) \leftarrow Q(s_{t+1}, a_{t+1}) + a[r_{t+1} + \gamma max_a * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
8. s←s';
9. Until s is terminal;

شرح الگوریتم:

- به ازای هر عمل a، به $Q(s,a)$ مقدار اولیه اختصاص بده (معمولاً صفر).
- در هر بازه زمانی تکرار کن.
  - حالت فعلی s را مشاهده کن.
  - تا مشاهده‌ی حالت پایانی تکرار کن.
    - عمل a را از حالت فعلی s انتخاب کن (با سیاست جاری).
    - عمل a را اجرا کن، پاداش r و حالت جدید s' را مشاهده کن.
    - مقدار Q را برای s و a به‌روز کن.
    - حالت جدید s' را حالت فعلی s در نظر بگیر.

۳-۱۰ جمع‌بندی الگوریتم‌های یادگیری

الگوریتم یادگیری Q و سارسا دو الگوریتم محبوب و مستقل از مدل برای یادگیری تقویتی هستند. تمایز این الگوریتم‌ها با یکدیگر در استراتژی‌های جست‌وجوی آن‌ها است، در حالی که استراتژی‌های استخراج آن‌ها مشابه است. الگوریتم یادگیری Q یک روش مستقل از سیاست است که عامل در آن، ارزش‌ها را بر اساس عمل a^* که از سیاست دیگری مشتق شده، می‌آموزد. سارسا یک روش مبتنی بر سیاست محسوب می‌شود که در آن ارزش‌ها را بر اساس عمل کنونی a که از سیاست کنونی آن مشتق شده، می‌آموزد. پیاده‌سازی این دو روش آسان است، اما فاقد تعمیم‌پذیری هستند؛ زیرا دارای توانایی تخمین ارزش‌ها برای حالت‌های مشاهده نشده نیستند.

۳-۱۱ رویکرد پیشنهادی

همان‌طور که گفته شد، در مدل یادگیری تقویتیِ شبکه، وضعیت همان گره، عامل همان بسته و عمل، انتخاب پورت خروجی است. در این رویکرد پیشنهادی از یادگیری Q استفاده شده است به این صورت که برای هر

گره یک جدول قرار داده که ستون‌هایش عمل و سطرهایش وضعیت است و مقادیر این جدول معادل تأخیری است که صرف می‌شود تا گره از مبدأ به مقصد برسد، در نتیجه در هر حالت کمترین مقدار انتخاب می‌شود، با توجه به اینکه مقدار تأخیر مثبت در نظر گرفته شده است.

مرحله‌ی اول پیاده‌سازی شامل ایجاد صفحه‌ای به نام QTable.cpp است که در این صفحه جدولی برای هر گره موجود در شبکه m*n ایجاد شده است. لازم به ذکر است که در این شبکه‌ی m*n ممکن است یک‌سری اتصالات بی‌سیم ایجاد شده باشد؛ یعنی بعضی گره‌ها قابلیت بی‌سیم داشته باشد. بعد از اضافه شدنِ صفحه‌ی QTable از این صفحه در روال مسیریابی استفاده شده است. کار اصلی و خروجی این پروژه، نوشتن یک الگوریتم مسیریابی است که این الگوریتم مبتنی بر یادگیری تقویتی است و نحوه‌ی عملکرد آن به این صورت است که وقتی بسته‌ای به گره‌ای می‌رسد، با بررسی مقادیر تأخیر جدول گره، پایین‌ترین مقدار تأخیر را انتخاب کند و جهت آن را جهت حرکت بعدی خود قرار دهد تا در مسیری کم ازدحام‌تر قرار گیرد [۴۲، ۴۳] و بتواند به مقصد مورد نظر سریع‌تر برسد.

نحوه پر کردن جدول هر گره با توجه به گره‌های بی‌سیم عبارت است از:

- برای هر گره یک جدول ایجاد می‌شود که شامل یک فیلد برای مبدأ، یک فیلد برای مقصد، چهار فیلد برای گره‌های همسایه (که نشان‌دهنده‌ی گره‌هایی است که در حرکت از مبدأ به سمت مقصد باید ابتدا به آن‌ها برویم) چهار فیلد برای جهت و چهار فیلد برای تأخیر است.

جدول ۳-۱) مقادیر جدول در یک شبکه‌ی ۳*۳ با گره‌های بی‌سیم ۱ و ۶

مقصد	تأخیرها				جهت‌ها				گره‌های همسایه				منبع
گره ۰				۰				۳				۰	گره ۱
گره ۱				۰				۴				۱	گره ۱
گره ۲				۰				۱				۲	گره ۱
گره ۳		۰	۰	۰		۵	۲	۳	۶	۴		۰	گره ۱
گره ۴				۰				۲				۴	گره ۱
گره ۵			۰	۰			۲	۱		۴		۲	گره ۱
گره ۶		۰	۰	۰		۵	۲	۳	۶	۴		۰	گره ۱
گره ۷		۰		۰		۵		۲	۶			۴	گره ۱
گره ۸		۰	۰	۰		۵	۲	۱	۶	۴		۲	گره ۱

- در آن جدول، به ازای همه‌ی مقصدها یک سطر وجود دارد. به‌طور مثال در یک شبکه‌ی ۳*۳ در واقع ۹ سطر برای هر گره مبدأ داریم که از ۰ تا ۸ می‌باشد.
- در هر سطر اطلاعاتی در رابطه با ارسال داده از گره مبدأ به گره مقصد وجود دارد.

پر کردن چهار فیلد گره‌های همسایه به صورت زیر است:

- اگر در شبکه گره بی‌سیم وجود نداشته باشد، در صورتی که گره مقصد در همان سطر یا ستون گره مبدأ باشد یک گره همسایه وجود دارد و گره مبدأ برای ارسال داده به گره مقصد، یک جهت دارد؛ ولی اگر گره مبدأ و گره مقصد در یک سطر یا ستون نباشد، برای ارسال داده، دو جهت وجود دارد. پس در واقع گره مبدأ دارای دو گره همسایه است. الگوریتم ۳-۳ روال به دست آوردن گره‌های همسایه بدون وجود گره بی‌سیم را نشان می‌دهد.

الگوریتم ۳-۳) الگوریتم به دست آوردن گره‌های همسایه و جهت‌ها بدون وجود گره بی‌سیم

```
1.  myR=SourceNumber/NumberOfColums;
2.  myC=SourceNumber%NumberOfColums;
3.  dR=DestinitionNumber/NumberOfColums;
4.  dC=DestinitionNumber%NumberOfColums;
5.  if  myR=dR then
6.          if myC=dC then
7.                  Port1=local;
8.                  NextNumber1=SourceNumber  ;
9.          elseif  myC<dC then
10.                 Port1=east;
11.                 NextNumber1=SourceNumber+1;
12.         else
13.                 Port1=west;
14.                 NextNumber1=SourceNumber-1;
15.         Endif
16. elseif  myC=dC then
17.         if myR<dR then
18.                 Port1=south;
19.                 NextNumber1=SourceNumber+NumberOfColums;
20.         else
21.                  Port1=north;
22.                  NextNumber1=SourceNumber-NumberOfColums;
23.         Endif
24. else
25.         if myR>dR and  myC>dC then
26.                 Port1=north;
27.                 NextNumber1=SourceNumber-NumberOfColums;
28.                 Port2=west;
```

```
29.             NextNumber2=SourceNumber-1;
30.         Endif
31.         if myR>dR and  myC<dC then
32.             Port1=north;
33.             NextNumber1=SourceNumber-NumberOfColums;
34.             Port2=east;
35.             NextNumber2=SourceNumber+1;
36.         Endif
37.         if myR<dR and  myC>dC then
38.             Port1=south;
39.             NextNumber1=SourceNumber+NumberOfColums;
40.             Port2=west;
41.             NextNumber2=SourceNumber-1;
42.         Endif
43.         if myR<dR and myC<dC then
44.             Port1=south;
45.             NextNumber1=SourceNumber+NumberOfColums;
46.             Port2=east;
47.             NextNumber2=SourceNumber+1;
48.         Endif
49. Endif
```

- اگر در شبکه گره بی‌سیم وجود داشته باشد، علاوه بر به دست آوردن گره‌های همسایه مطابق با الگوریتم ۳-۳، استفاده از قابلیت بی‌سیم بررسی می‌شود.

برای بررسی لازم است مشخص شود که آیا نزدیک‌ترین مسیریاب بی‌سیم گره مبدأ و گره مقصد یکی است یا خیر. در صورت یکی بودن نزدیک‌ترین مسیریاب بی‌سیمِ گره مبدأ و گره مقصد، امکان استفاده از قابلیت بی‌سیم وجود ندارد؛ ولی اگر امکان ارسال بی‌سیم وجود داشته باشد، دو حالت پیش می‌آید: اول این‌که مبدأ بی‌سیم باشد و دیگری این‌که مبدأ بی‌سیم نباشد. اگر گره مبدأ بی‌سیم باشد، باید شماره‌ی نزدیک‌ترین گره بی‌سیم به مقصد را به گره‌های همسایه اضافه کرد؛ به این دلیل که گره بعدی، گره بی‌سیم است که باید

فیلدها برایش ارسال شود و جهت بی‌سیم نیز به جهت‌های آن اضافه شود. اگر مبدأ بی‌سیم نباشد، باید برای استفاده از قابلیت بی‌سیم، فیلدها به نزدیک‌ترین مسیریاب بی‌سیم ارسال شود. پس در واقع از الگوریتم ۳-۳ دو بار استفاده می‌شود؛ با این توضیح که یک‌بار مبدأ و مقصد همان مبدأ و مقصد اولیه است و در بار دیگر، مبدأ همان مبدأ ولی مقصد، نزدیک‌ترین گره بی‌سیم به مبدأ است.

اگر نزدیک‌ترین مسیریاب بی‌سیم مبدأ و مقصد یکی نباشد همچنین باید صرفه‌ی زمانیِ استفاده از آن بررسی شود. برای این کار، باید فاصله‌ی دکارتی بین مبدأ و مقصد را بدون در نظر گرفتن گره بی‌سیم و با در نظر گرفتن گره بی‌سیم به دست آورد. اگر فاصله‌ی اول از فاصله‌ی دوم کمتر باشد، از قابلیت بی‌سیم استفاده نخواهد شد؛ در غیر این صورت، از قابلیت بی‌سیم استفاده خواهد شد.

برای به دست آوردن فاصله‌ی دکارتی به این صورت عمل می‌شود که در صورت عدم وجود گره بی‌سیم این مقدار برابر است با:

|فاصله ستون مبدأ و ستون مقصد| + |فاصله سطر مبدأ و سطر مقصد|

و در صورت وجود گره بی‌سیم این مقدار برابر است با:

|فاصله دکارتی بین مقصد و نزدیک‌ترین گره بی‌سیم به مقصد| + |فاصله دکارتی بین مبدأ و نزدیک‌ترین گره بی‌سیم به مبدأ| + هزینه استفاده از بی‌سیم.

مقادیر تأخیر نیز در هنگام ایجاد جدول برای هر گره به‌طور پیش‌فرض برابر صفر می‌باشد. برای پر کردن فیلد جهت‌ها از الگوریتم ۳-۳ استفاده می‌شود.

پس از پر کردن مقادیر جداول، نکته‌ی مهم این است که وقتی بسته‌ای ارسال می‌شود، جداول در این جابجا شدن بسته‌ها به‌روزرسانی شود تا گره در زمان انتخاب مسیر، تصمیم‌گیری درستی انجام دهد. برای محاسبه و به‌روزرسانی مقادیر تأخیر، از رابطه (۳-۵) استفاده می‌شود.

$$Q_x(y,d) = Qx\,(y,d)_{old} + \alpha\left[\left(\gamma * Q_y(z,d)\right) + q_y + \delta - Qx\,(y,d)_{old}\right] \quad (۵-۳)$$

این رابطه مقدار تأخیر جدید را در زمان ارسال بسته از گره x به گره d با واسطه‌ی گره همسایه‌ی y محاسبه می‌کند. در این رابطه α نرخ یادگیری است که در واقع نرخ اطلاعات جدیدتر را تعیین می‌کند.

q_y نشان‌دهنده‌ی مدت‌زمانی است که بسته ارسال شده از گره x در بافر ورودی گره y منتظر می‌ماند.

δ نشان‌دهنده‌ی مدت‌زمانی است که بسته از گره x به گره y ارسال می‌شود.

$Q_y(z,d)$ نشان‌دهنده‌ی مدت‌زمانی است که بسته از گره y با کمک گره همسایه‌ی z به مقصد d می‌رود.

$Qx\ (y,d)_{old}$ نشان‌دهنده‌ی مقدار تأخیری است که در جدول قبل از به‌روزرسانی وجود دارد.

γ یا عامل تخفیف[1]، اهمیت پاداش‌های آینده را تعیین می‌کند و به همین علت در مقدار $Q_y(z,d)$ تأثیر می‌گذارد.

برای محاسبه‌ی هر کدام از مقادیر بالا به این صورت عمل شده است:

مقدار $Qx\ (y,d)_{old}$ در جدول وجود دارد.

برای به دست آوردن مقدار $Q_y(z,d)$ اولین کاری که باید انجام شود این است که جدول مجزایی ایجاد شود که در هر خانه‌ی آن، جداول گره‌های موجود قرار داشته باشد و این جدول به صورت عمومی در دسترس قرار گیرد. البته با این کار، دسترسی فقط به جدول گره بعدی ممکن است؛ بنابراین باید بررسی شود که مقصد کدام سطر با مقصد مورد نظر برابری می‌کند. وقتی سطر مورد نظر به دست آمد، تأخیری که مقدار آن از همه کمتر و مخالف ۱- است به عنوان تأخیر از گره y به مقصد d انتخاب می‌شود.

برای محاسبه‌ی δ و q_y به این صورت عمل شده که وقتی بسته‌ای روی یک جهت قرار می‌گیرد، با استفاده از تابع sc_time_stamp() زمان قرارگیری بسته در جهت مورد نظر به دست می‌آید و وقتی بسته دریافت شد و پیغام تأیید آن نیز صادر شود، زمان آن محاسبه می‌شود. اختلاف این دو زمان، نشان‌دهنده‌ی مقدار تأخیر است.

برای مقدار نرخ یادگیری و عامل تخفیف مقادیر بهینه گذاشته می‌شود. برای محاسبه‌ی مقدار تأخیر جدید از تابع calculate که در الگوریتم ۳-۴ می‌باشد، استفاده شده است.

[1] Discount Factor

الگوریتم ۳-۴) محاسبه و به روزرسانی تأخیر

```
1.  function  calculate(float myOld_value, float learning_rate,float discount_factor, float nextNew_value, int wait_delay, int transmission_delay)
2.  {
3.  sum=myOld_value+learning_rate*((discount_factor*nextNew_value)+wait_delay+ transmission_delay- myOld_value);
4.   return sum;
5.}
```

پس از ایجاد جدول و توضیح درباره‌ی به‌روزرسانی آن، به بررسی عملکرد الگوریتم پیشنهادی که در الگوریتم ۳-۵ می‌باشد، پرداخته شده است.

الگوریتم ۳-۵) الگوریتم مسیریابی پیشنهادی

```
1.  function routingAlgorithmRLBased(int destinationId){
2.      vector directions;
3.      value=getLowestLatency(Latency1(),Latency2(), Latency3(),Latency4());
4.      if  Latency1() = value  then
5.          directions+=Port1();
6.      elseif  Latency2() = value then
7.          directions+=Port2();
8.      elseif  Latency3() = value  then
9.         directions+=Port3();
10.     elseif  Latency4() = value  then
11.         Directions+=Port4() ;
12.     Endif
13.  return directions;
14.   }
```

الگوریتم به این صورت عمل می‌کند که مقدار تأخیر را بررسی کرده و هر کدام از مقادیر تأخیر که کمتر باشد، جهت آن به عنوان جهت بعدیِ حرکت بسته در نظر گرفته می‌شود.

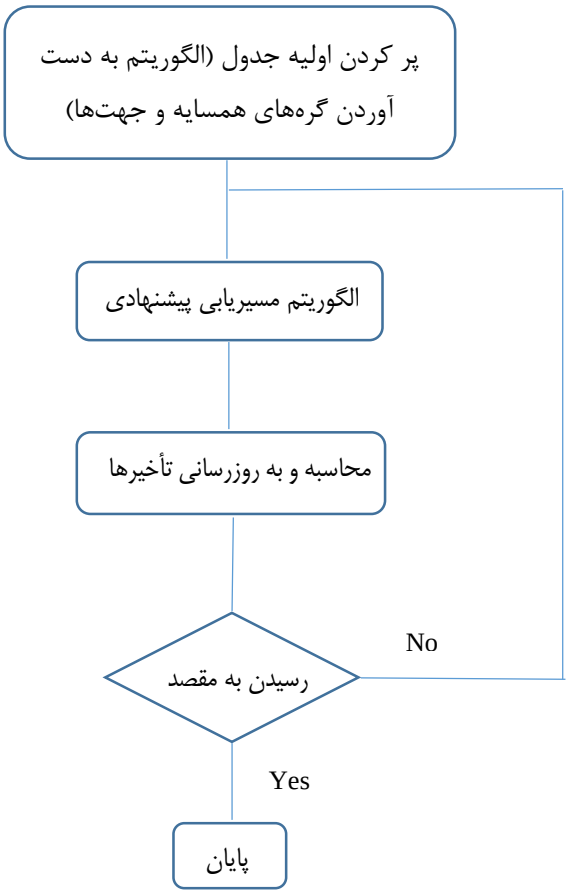
در الگوریتم ۳-۶ نحوه‌ی به دست آوردن کمترین تأخیر نشان داده شده است.

الگوریتم ۳-۶) تابع به دست آوردن کمترین تأخیر

```
1.  Function getLowestLatency(float Latency1,float Latency2,float Latency3,float Latency4){
2.      float output1, output2, output;
3.        if   Latency1<=Latency2   and   Latency1!=-1   then
4.                output1=Latency1;
5         else if    Latency2!=-1
6.                output1=Latency2;
7.        Endif
8.        if   Latency3<=Latency4   and   Latency3!=-1   then
9.                output2=Latency3;
10.       else if    Latency4!=-1
11.               output2=Latency4;
12.       Endif
13.       if   output1<=output2   and   output1!=-1    then
14.               output=output1;
15.       else if    output2!=-1
16.               output=output2;
17.       Endif
18.   return output;
19.}
```

به طور کلی می‌توان گفت روال برنامه به این صورت است که برای هر گره یک جدول ایجاد شده و با استفاده از الگوریتم ۳-۳ تکمیل می‌شود و برای فیلدهای تأخیر در ابتدا مقادیر صفر قرار داده می‌شود. سپس با استفاده از الگوریتم مسیریابی پیشنهادی (الگوریتم ۳-۵) ، مسیر حرکت بسته انتخاب می‌شود. با توجه به این که در

ابتدا مقادیر تأخیرها به طور پیش‌فرض برابر صفر است، طبق تابع GetLowestLatency در الگوریتم ۳-۶ ، ابتدا جهت اولین گره همسایه به عنوان جهت حرکت بسته انتخاب می‌شود. پس از انتخاب مسیر، مقدار تأخیر با استفاده از الگوریتم۳-۴ محاسبه و به روز رسانی می‌شود و داده در مسیر انتخاب‌شده به حرکت خود ادامه می‌دهد و این روال تا رسیدن به مقصد ادامه پیدا می‌کند. فلوچارت زیر مراحل کار را نشان می‌دهد.



شکل ۳-۳) شمای کلی روش پیشنهادی

پس از نوشتن کد برنامه‌نویسی و گرفتن خروجی‌ها و بررسی آن‌ها به این نتیجه رسیدیم که روند اجرای برنامه بعد از مدتی دچار بن‌بست می‌شود و داده‌ای ارسال نمی‌شود.

بن‌بست به دلیل حلقه به وجود آمده است. برای از بین بردن بن‌بست، روش پیشنهادی این است که از تعداد بیشتری کانال مجازی استفاده شود. به این منظور، ابتدا مقدار کانال مجازی را در فایل تنظیمات[۱] افزایش

[۱] Config

دادیم. سپس زمانی که یک فلیت شروع به حرکت می‌کند، یکی از جهت‌ها را به عنوان جهت اولیه در نظر می‌گیریم و مقدار کانال مجازی را معادل مقدار جهت اولیه قرار می‌دهیم. در حرکت بعدی، اگر فلیت دچار تغییر جهت شد، مقدار کانال مجازی را یک واحد اضافه می‌کنیم و این روال به همین ترتیب ادامه می‌یابد تا بسته به گره مقصد برسد. با انجام این کار، مشکل بن‌بست برطرف گردید.

در این روش پیشنهادی به کمک الگوریتم‌های بهینه‌سازی مانند الگوریتم تبرید شبیه‌سازی شده[1] موقعیت مناسب گره‌های بی‌سیم تعیین می‌شود [۴۹، ۵۰].

الگوریتم تبرید شبیه‌سازی شده یک الگوریتم بهینه‌سازی فراابتکاری ساده و اثربخش در حل مسائل بهینه‌سازی در فضاهای جستجوی بزرگ است. این الگوریتم بیشتر زمانی استفاده می‌شود که فضای جستجو گسسته باشد. کاربرد آن برای مسائلی است که پیدا کردن یک پاسخ تقریبی برای بهینه کلی مهم‌تر از پیدا کردن یک پاسخ دقیق برای بهینه محلی در زمان محدود و مشخصی است. در این روش، جواب‌های یک مسئله به خوبی گرم می‌شوند و با نوسانات زیادی تغییر می‌کنند؛ سپس، به تدریج دامنه تغییرات کم می‌شود و در واقع یک سری قسمت‌ها به سمت جواب بهینه می‌رود. در واقع در این الگوریتم، از روش احتمالاتی برای حل مسئله بهینه‌سازی استفاده می‌شود [۵۱، ۵۲].

۳-۱۲ جمع‌بندی

هدف از روش پیشنهادی، ایجاد الگوریتم مسیریابی جدیدی است که در هر مرحله از مسیر، قابلیت تشخیص مسیر پر ازدحام و تغییر مسیر را داشته باشد که در الگوریتم‌های مسیریابی دیگر، این امکان وجود ندارد. این قابلیت الگوریتم مسیریابی جدید، باعث شد که بسته‌ها سریع‌تر به مقصد برسند و تأخیر در ارسال بسته کاهش یابد.

[1] Simulated Annealing

فصل ٤: نتایج و آزمایش‌ها

۴-۱ مقدمه

در این فصل به بررسی عملکرد روش پیشنهادی با استفاده از نمودارها و جداول پرداخته شده است و با آزمایش‌ها توانسته‌ایم بهینه‌ترین مقادیر را شناسایی و محاسبه کنیم و از آن برای خروجی بهتر استفاده نماییم. همچنین مسیریابی پیشنهادی را با الگوریتم‌های مسیریابی دیگر مقایسه کرده و نتایج را گزارش نماییم. نتایج شبیه‌سازی نشان می‌دهد که الگوریتم مسیریابی با رویکرد یادگیری تقویتی در حالات مختلف ترافیک باعث کاهش تأخیر بیشتری در انتقال بسته‌ها، نسبت به بقیه‌ی الگوریتم‌های مسیریابی شده است.

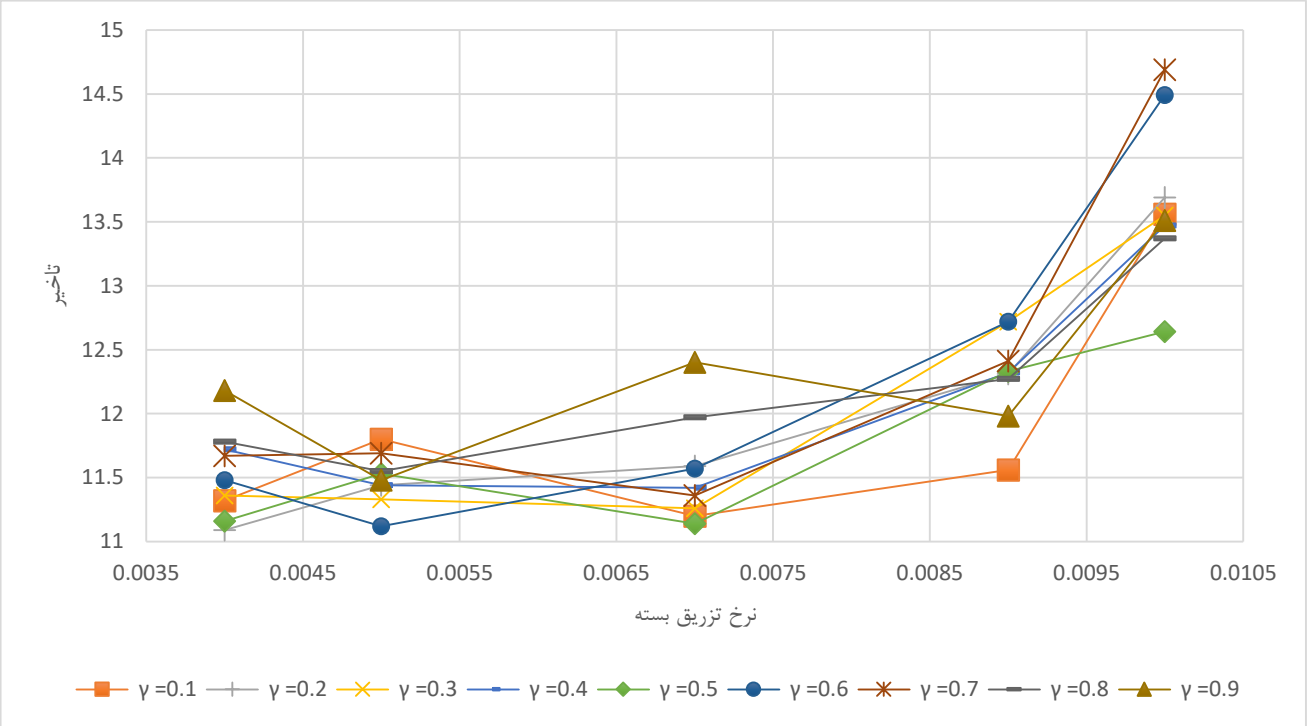
۴-۲ محیط آزمایش

برای پیاده‌سازی الگوریتم مسیریابی مبتنی بر یادگیری و نیز اجرای شبیه‌سازی آن، از شبیه‌ساز Noxim [۱۱] استفاده شده است. این شبیه‌سازِ شبکه‌ی روی تراشه، به وسیله‌ی گروه معماری کامپیوتر دانشگاه کاتانیا واقع در کشور ایتالیا طراحی شده است. در طراحی شبیه‌ساز Noxim از System C استفاده شده که System C نیز بر اساس زبان برنامه‌نویسی C++ پایه‌ریزی شده است. Noxim دارای یک خط فرمان می‌باشد که می‌توان چندین پارامتر مربوط به شبکهِ روی تراشه را تعریف و مقداردهی کند و به عنوان ورودی به شبیه‌ساز دهد. ازجمله این مقادیر ورودی می‌توان به اندازه‌ی بافر، نوع الگوریتم مسیریابی، موقعیت گره‌های بی‌سیم، تعداد کانال‌های مجازی، مدت‌زمان شبیه‌سازی شده، نوع ترافیک و ... اشاره کرد. این شبیه‌ساز خروجی‌هایی از جمله توان، تأخیر ارسال بسته‌ها، تعداد فیلدهای تزریق شده، درصد بسته‌هایی که از طریق بی‌سیم دریافت شده‌اند و ... را به کاربر می‌دهد که برای مقایسه طراح‌های مختلف بسیار سودمند هستند [۴۶-۴۸]. پیکربندی شبکه‌ی دوبعدی برای شبکه‌ی روی تراشه استفاده شده است. در این آزمایش‌ها به بررسی عملکرد روش الگوریتم مسیریابی پیشنهادی بر اساس یادگیری تقویتی در مقایسه با الگوریتم مسیریابی WirelessXY و الگوریتم مسیریابی XY پرداخته شده است. الگوریتم مسیریابی شبیه‌سازی‌ها در شبکه‌ی روی تراشه‌ی ۴*۴ و ۵*۵ انجام می‌شود و عملکرد انواع الگوریتم‌های مسیریابی بر اساس منحنی‌های تأخیر ارزیابی می‌شود. فرض شده است که بسته‌های داده و بسته‌های یادگیری دارای طول متفاوت می‌باشد. زمان شبیه‌سازی معادل ۵۰۰۰

نانوثانیه قرار داده شده است و اندازه‌ی بافر روی مقدار ۴ تنظیم شده است. همچنین از سه الگوی ترافیکی مختلفِ Transpose، Hotspot و تصادفی، برای نمایش و مقایسه نتایج استفاده کرده‌ایم.

۴-۳ خلاصه‌ی آزمایش‌ها

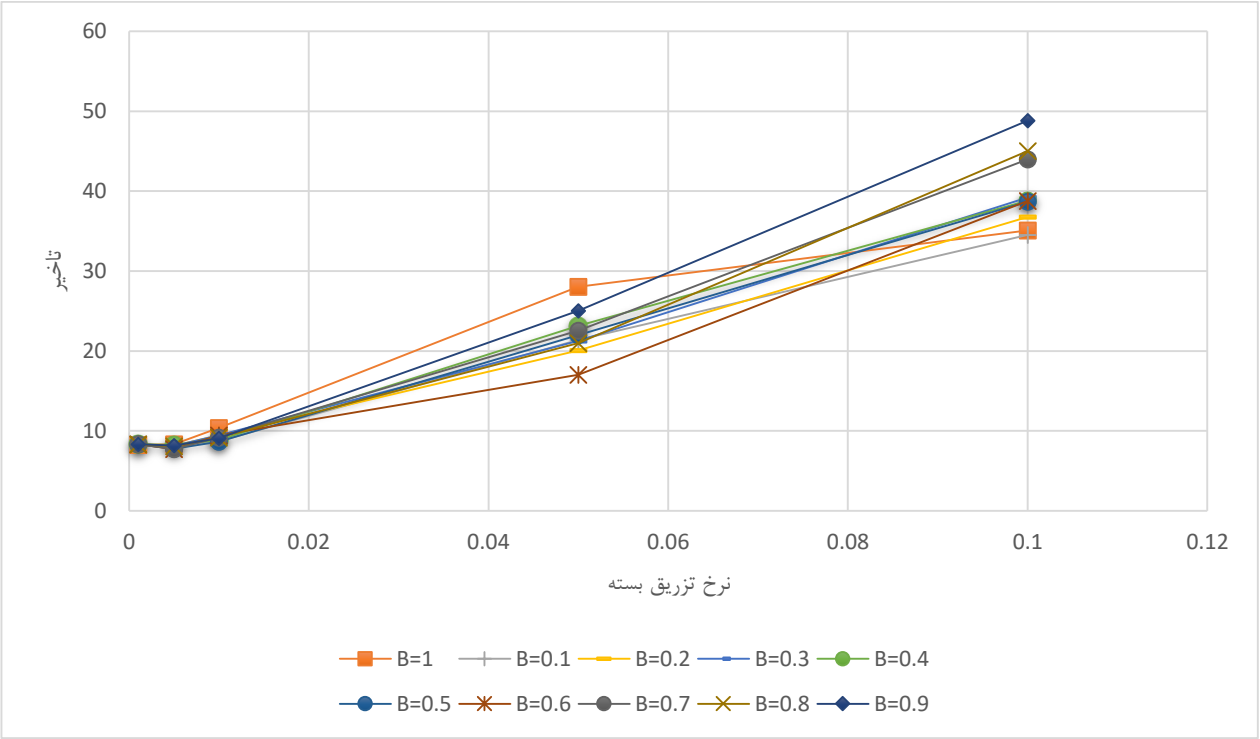
در الگوریتم مسیریابی پیشنهادی بر اساس یادگیری تقویتی، دو پارامتر نرخ یادگیری و عامل تخفیف می‌تواند دارای مقادیری متفاوتی باشد به همین دلیل تأثیر این دو پارامتر، در ابتدا مورد بررسی قرار خواهند گرفت.



شکل ۴-۱) نمودارهای تأخیر با استفاده از نرخ یادگیری‌های متفاوت

شکل ۴-۱ نشان‌دهنده‌ی نمودارهای تأخیر الگوریتم مسیریابی با استفاده از نرخ یادگیری‌های متفاوت می‌باشد. محور افقی نمودار (نرخ تزریق بسته) نشان‌دهنده‌ی مقدار احتمال تزریق بسته به شبکه در هر کلاک است. هر چه این نرخ بیشتر باشد بار شبکه هم بیشتر می‌شود. محور عمودی (تأخیر) نشان‌دهنده‌ی مدت‌زمان ایجاد بسته تا رسیدن بسته به مقصد است. با فرض مثبت بودن مقادیر تأخیر و اینکه مقدار تأخیر هر چه قدر کمتر باشد، بهتر است با بررسی نمودارها به این نتیجه می‌رسیم که نرخ یادگیریِ برابر ۰٫۵ عملکرد بهتری داشته

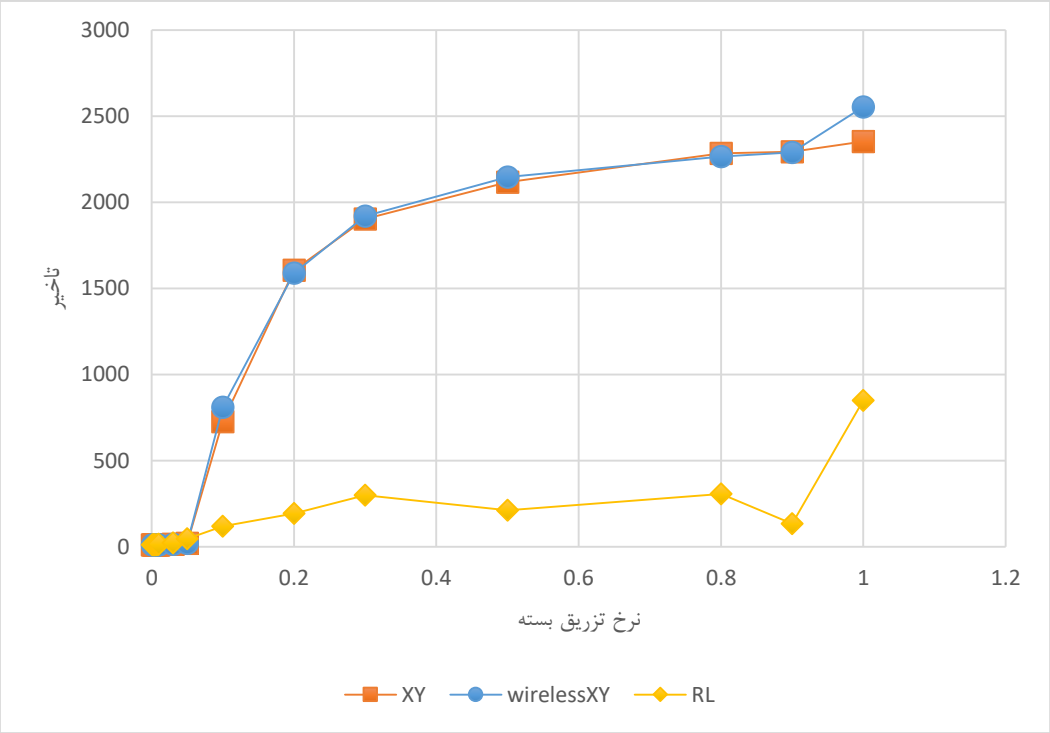
است و از این مقدار در مراحل بعدی استفاده می‌شود. همان‌طور که قبلاً گفته شد نرخ یادگیری در واقع نرخ اطلاعات جدیدتر را تعیین می‌کند یا به عبارت دیگر، نرخی را نشان می‌دهد که در آن اطلاعات جدیدتر، اطلاعات قدیمی‌تر را بازنویسی می‌کند.



شکل ۴-۲) نمودارهای تأخیر با استفاده از عامل‌های تخفیف متفاوت

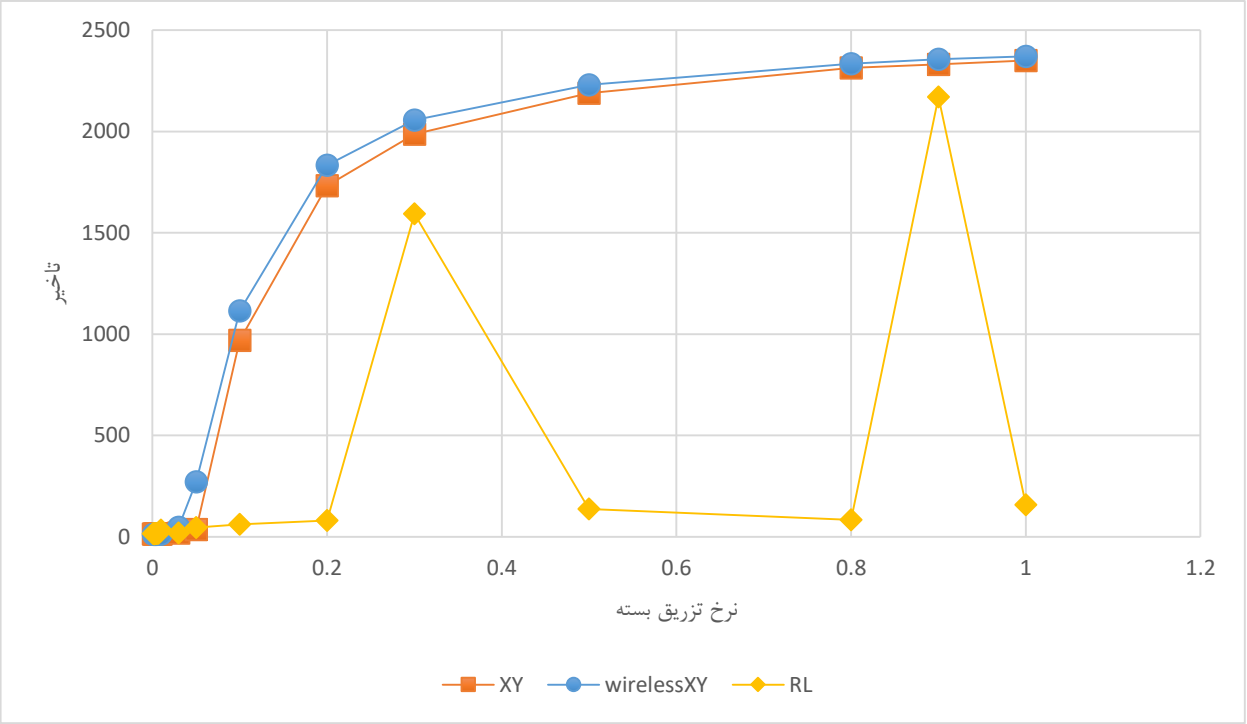
شکل ۴-۲ نمودارهای تأخیر را در الگوریتم مسیریابی با استفاده از نرخ یادگیری ۰,۵ و عامل‌های تخفیف متفاوت نشان داده است. با بررسی این نمودارها به این نتیجه رسیدیم که عامل تخفیف وقتی برابر یک گردد، الگوریتم مسیریابی دارای خروجی بهتری می‌باشد.

حال الگوریتم پیشنهادی را در سه نوع ترافیک Hotspot، Transpose و تصادفی، با الگوریتم مسیریابی WirelessXY و الگوریتم مسیریابی XY در شبکه‌ی ۴*۴ و شبکه‌ی ۵*۵ مقایسه می‌کنیم. اکنون به بررسی نمودارها، در انواع ترافیک در شبکه‌ی ۴*۴ می‌پردازیم.



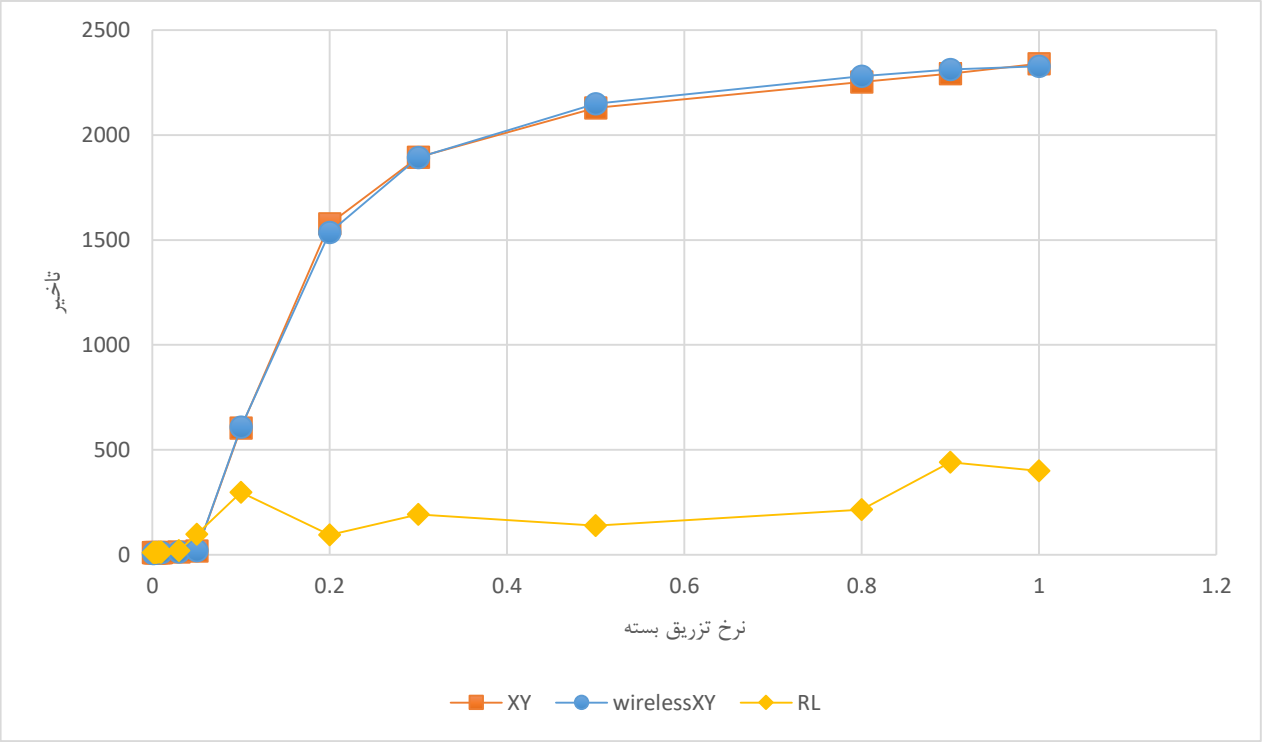
شکل ۴-۳) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک تصادفی در شبکه ۴*۴

شکل ۴-۳ میانگین تأخیر را به عنوان تابعی از میانگین نرخ تزریق داده در ترافیک تصادفی نشان می‌دهد. در ترافیک تصادفی [۴۵]، هر گره با یک احتمال تصادفی یک بسته را به گره دیگر ارسال می‌کند. مقصد بسته‌های مختلف به صورت تصادفی با استفاده از توزیع یکنواخت تعیین می‌شود. همان‌طور که از نتایج مشاهده می‌شود الگوریتم پیشنهادی منجر به تأخیر کمتری نسبت به سایر الگوریتم‌ها شده است و در این حالت الگوریتم پیشنهادی تأخیر را حداقل ۱۹ درصد بهبود بخشیده است.



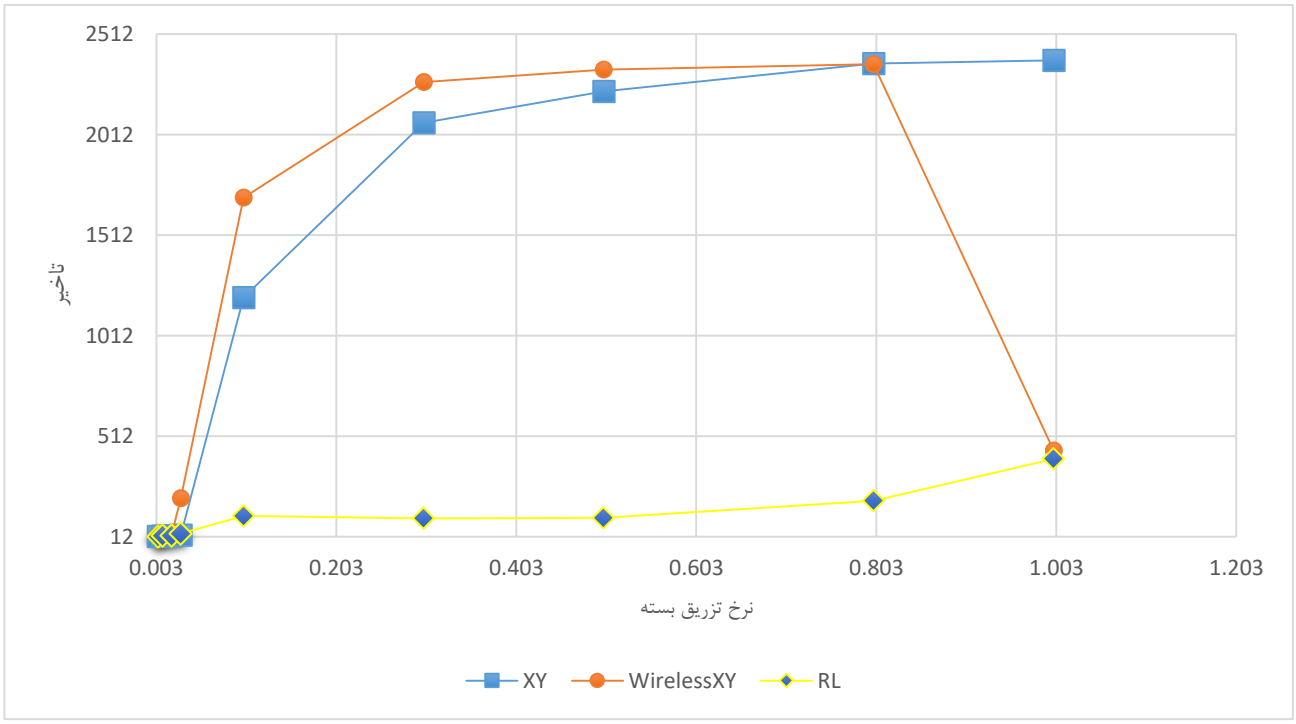
شکل ۴-۴) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک Transpose در شبکه ۴*۴

شکل ۴-۴ میانگین تأخیر را به عنوان تابعی از میانگین نرخ تزریق داده در ترافیک Transpose نشان می‌دهد. در ترافیک Transpose یک گره (i،j) فقط می‌تواند به یک گره (j،i) بسته ارسال کند. همان‌طور که از نتایج مشاهده می‌شود در این نوع ترافیک هم الگوریتم پیشنهادی نسبت به سایر الگوریتم‌ها منجر به تأخیر کمتری شده است و در این حالت الگوریتم پیشنهادی تأخیر را حداقل ۸ درصد بهبود بخشیده است.

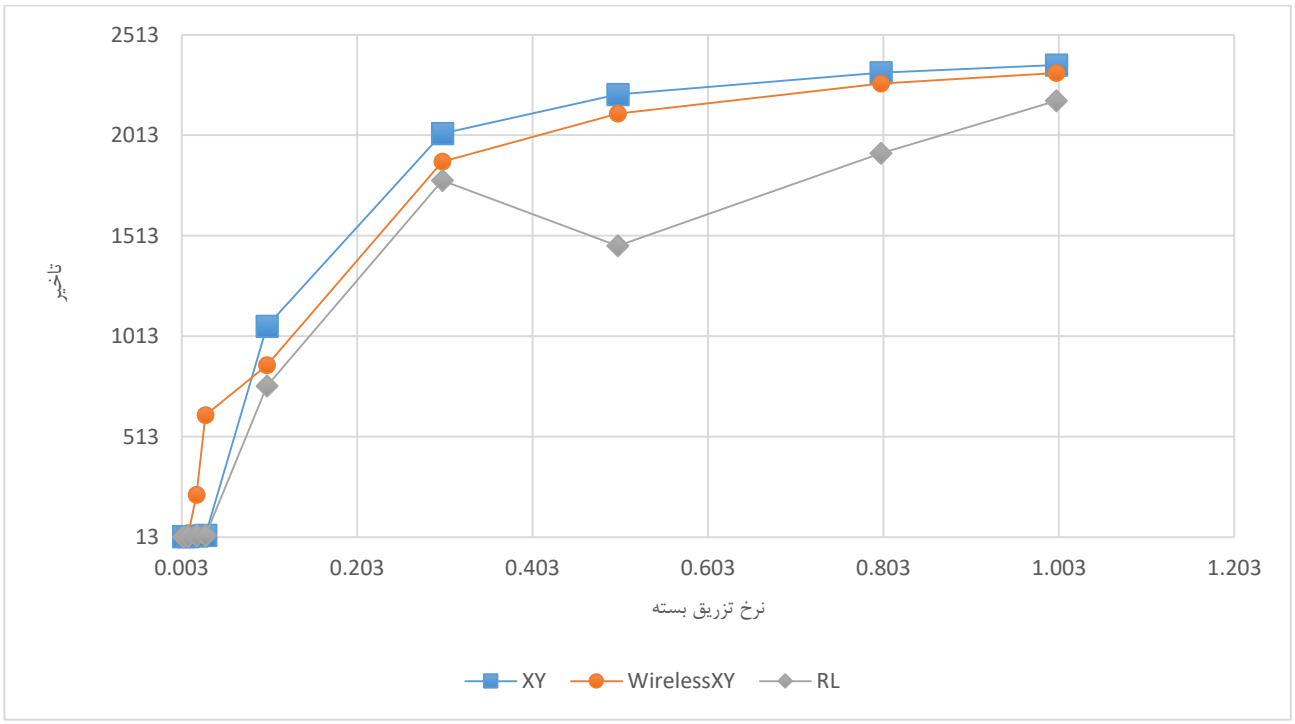


شکل ۴-۵) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک Hotspot در شبکه ۴*۴

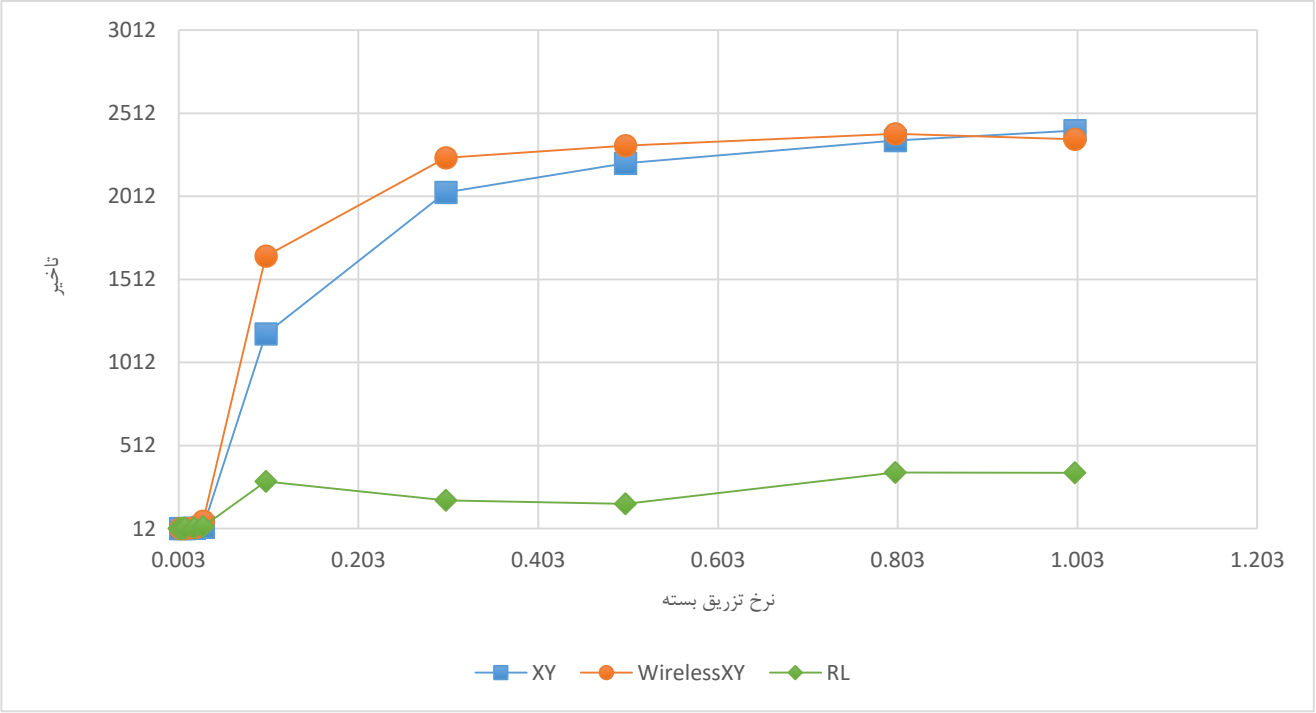
شکل ۴-۵ میانگین تأخیر را به عنوان تابعی از میانگین نرخ تزریق داده در ترافیک Hotspot نشان می‌دهد. در ترافیک Hotspot یک یا چند گره به عنوان نقاط انتخاب شده که علاوه بر ترافیک یکنواخت منظم، بخش بیشتری از ترافیک را دریافت می‌کنند وجود دارد. در این نوع ترافیک مانند سایر ترافیک‌ها، همان‌طور که از نتایج مشاهده می‌شود الگوریتم پیشنهادی منجر به تأخیر کمتر نسبت به سایر الگوریتم‌ها شده است و در این حالت الگوریتم پیشنهادی تأخیر را حداقل ۱۲ درصد بهبود بخشیده است. اکنون به بررسی نمودارها، در انواع ترافیک در شبکه‌ی ۵*۵ می‌پردازیم



شکل ۴-۶) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک تصادفی در شبکه ۵*۵



شکل ۴-۷) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک Transpose در شبکه ۵*۵



شکل ۴-۸) نمودارهای تأخیر در الگوریتم پیشنهادی، XY و WirelessXY در ترافیک Hotspot در شبکه ۵*۵

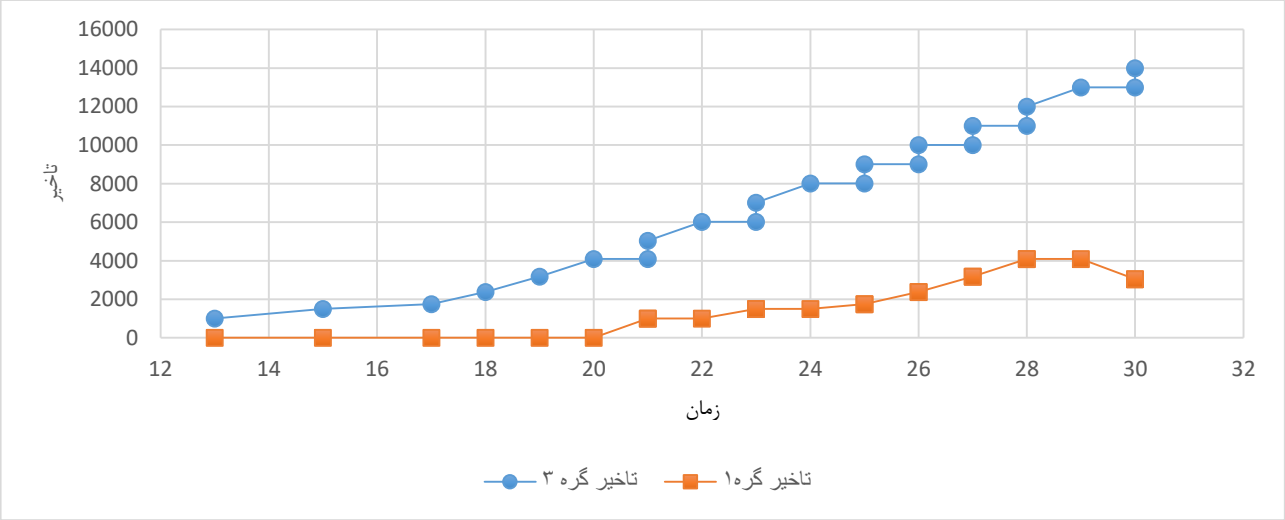
همان‌طور که از نتایج نمودارها در شبکه‌ی ۵*۵ در حالات مختلف ترافیک مشاهده می‌شود الگوریتم پیشنهادی در هر سه نوع ترافیک منجر به تأخیر کمتری نسبت به سایر الگوریتم‌ها شده است. این الگوریتم تأخیر را در شبکه‌ی ۵*۵ در ترافیک تصادفی حداقل ۱۰ درصد بهبود، در ترافیک Transpose حداقل ۶ درصد بهبود و در ترافیک Hotspot، حداقل ۱۳ درصد بهبود بخشیده است.

علت نتایج خوب الگوریتم پیشنهادی در همه حالات نسبت به الگوریتم WirelessXY و الگوریتم XY این است که این الگوریتم یک الگوریتم «قابل تطبیق با شرایط ازدحام» است؛ یعنی اگر بسته در مسیر رسیدن به مقصد به مسیر پر ازدحام برخورد کند، مسیرش را عوض می‌کند و منتظر آزاد شدن مسیر نمی‌ماند و این باعث می‌شود که تأخیر کمتری نسبت به سایر الگوریتم‌ها داشته باشد. در حالی که سایر الگوریتم‌ها دارای این قابلیت نیستند.

در تمام نمودارها در شبکه‌ی ۴*۴ و شبکه‌ی ۵*۵ نمودارهای الگوریتم پیشنهادی در ابعاد مختلف شبکه تأخیر کمتری نسبت به سایر الگوریتم‌ها داشته است و این نشان‌دهنده‌ی این است که الگوریتم پیشنهادی به ابعاد شبکه وابسته نیست و در ابعاد مختلف درست کار می‌کند.

در تمام نمودارها در شبکه‌ی ۴*۴ و شبکه‌ی ۵*۵ نمودارهای الگوریتم XY و الگوریتم WirelessXY تقریباً دارای تأخیر یکسانی هستند و این نشان‌دهنده‌ی این است که الگوریتم WirelessXY در این بازه‌ی نرخ تزریق، نتوانسته به خوبی از قابلیت بی‌سیم خود استفاده کند و عملکردی در حد الگوریتم XY داشته است.

برای نشان دادن این که یک بسته در مسیر حرکت خود چگونه به پیش می‌رود، نمودارهای زیر نمایش داده شده است. این نمودارها در یک شبکه‌ی ۳*۳ که گره مبدأ آن گره صفر و گره مقصد آن گره ۵ است، برای گره‌های وسط مقادیر تأخیرها را نشان می‌دهد.



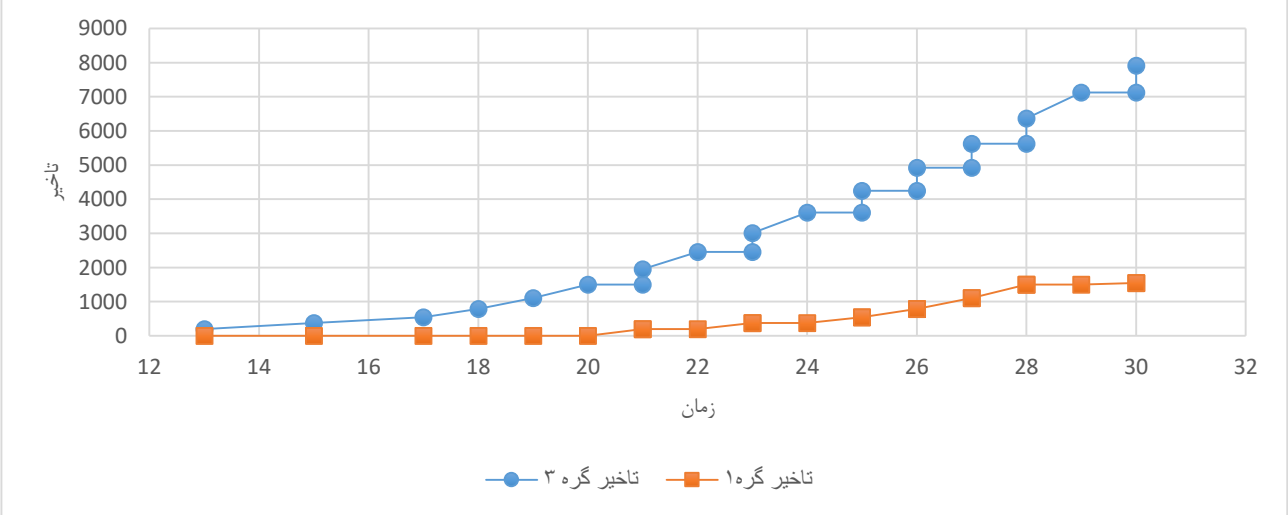
شکل ۴-۹) نمودار نحوه‌ی حرکت بسته در مسیر بر اساس تأخیرها در نرخ یادگیری ۰٫۵ برای گره ۱ و ۳

در این نمودارها گره‌های وسط یک و سه بررسی شده‌اند. با بررسی شکل ۴-۹ متوجه خواهید شد که در این دو نمودار در هر بازه زمانی یک نمودار به صورت مستقیم و در همان بازه، نمودار دیگر به صورت شیب‌دار

می‌باشد که نمودار شیب‌دار نشان‌دهنده‌ی این است که بسته در این مسیر در حال حرکت است و نمودار مستقیم نشان‌دهنده‌ی عدم وجود بستهِ در حال حرکت در این مسیر است.

حرکت بسته در هر دو مسیر بر اساس الگوریتم پیشنهادی با بررسی مقادیر تأخیرها انجام می‌شود و نحوه عملکرد آن به این صورت است که در ابتدا با توجه به اینکه مقادیر تأخیرها به طور پیش فرض صفر است طبق تابع getLowestLatency در پیاده‌سازی، ابتدا جهت اولین گره همسایه به عنوان جهت حرکت بسته انتخاب می‌شود به همین دلیل ابتدا بسته در مسیر گره ۳ شروع به حرکت می‌کند تا زمان ۲۰ نانوثانیه. در این زمان الگوریتم پیشنهادی پس از بررسی مقدار تأخیرها در دو گره متوجه می‌شود که مقدار تأخیر در گره یک کمتر از گره سه می‌باشد پس جهت حرکت بسته را در مسیر گره یک قرار می‌دهد. سپس دوباره مقدار تأخیرها در زمان ۲۱ نانوثانیه بررسی شده و این دفعه گره مبدأ بسته را در مسیر گره سه قرار می‌دهد و این روال انتخاب مسیر کم ازدحام‌تر ادامه دارد.

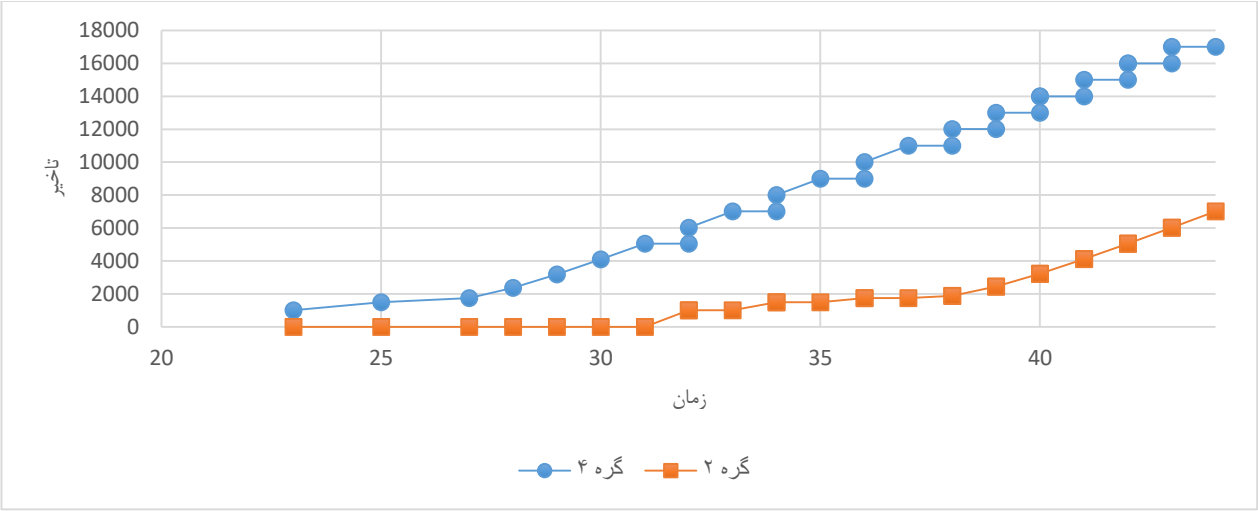
در بازه‌های زمانی که نمودار شیب‌دار است گاهی این شیب به سمت بالا و گاهی شیب نمودار به سمت پایین است. شیب بالا به این معنی است که مقدار تأخیر در حال افزایش است. علت شیب نزولی این است که گره بعدی که قرار است بسته را دریافت کند ممکن است قبلاً مسیرش توسط گره دیگری رزرو شده باشد و حالا مسیرش آزاد شده است و این باعث می‌شود که حرکت بسته با تأخیر کمتری انجام شود.



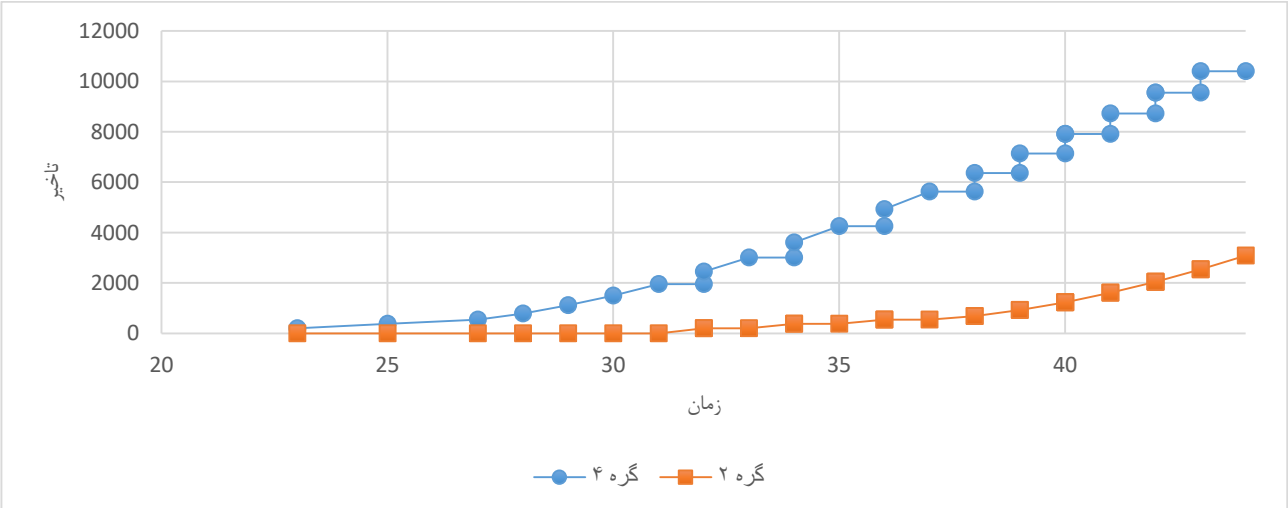
شکل ۴-۱۰) نمودار نحوه‌ی حرکت بسته در مسیر بر اساس تأخیرها در نرخ یادگیری ۰,۱ برای گره ۱ و ۳

شکل ۴-۱۰ برای همان شرایط نمودار بالاست با این تفاوت که در این نمودار نرخ یادگیری دارای مقدار ۰,۱ می‌باشد. در این نمودار حرکت بسته همان روال نمودار بالا را دارد، فقط مقدار تأخیرها در هر زمان نسبت به نمودار بالا کمتر است که این به علت مقدار پایین‌تر نرخ یادگیری می‌باشد.

اگر فرض کنیم که بسته به گره یک رسیده است گره‌های همسایه آن، گره ۴ و ۲ می‌باشند. در زیر نمودار گره‌های ۴ و ۲ وقتی بسته در گره یک است، در دو نرخ یادگیری متفاوت نمایش داده شده است.



شکل ۴-۱۱) نمودار نحوه‌ی حرکت بسته در مسیر بر اساس تأخیرها در نرخ یادگیری ۰,۵ برای گره ۲ و ۴

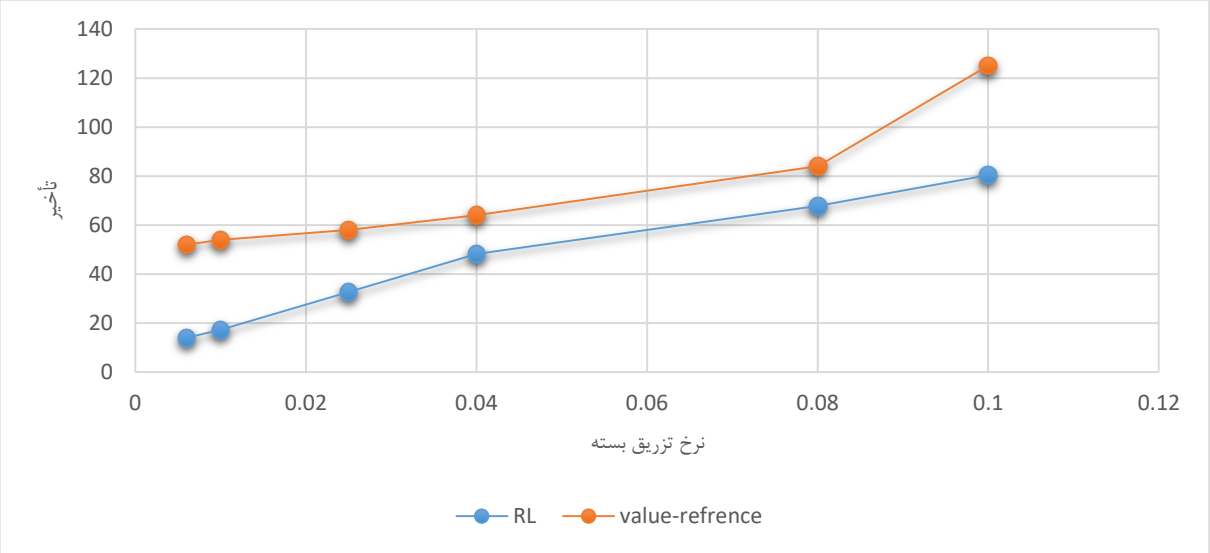


شکل ۴-۱۲) نمودار نحوه‌ی حرکت بسته در مسیر بر اساس تأخیرها در نرخ یادگیری ۰,۱ برای گره ۲ و ۴

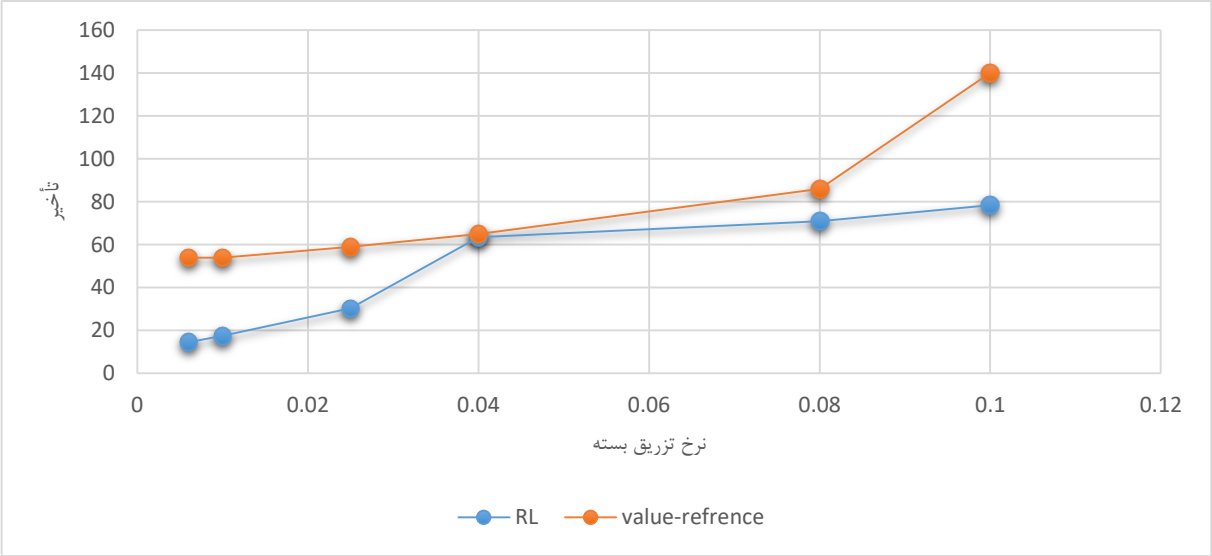
همان‌طور که در شکل ۴-۱۱ و ۴-۱۲ نشان داده شده است این دو نمودار همان روال نمودارهای قبل را دارند با این تفاوت که زمانی که بسته به این گره‌ها می‌رسند دیرتر از زمانی است که بسته به گره‌های یک و سه می‌رسد.

۴-۴ مقایسه با کارهای گذشته

در این بخش به مقایسه الگوریتم پیشنهادی با مرجع [۵۹] پرداخته‌ایم. محور افقی در شکل‌ها، نشان دهنده‌ی نرخ تزریق بسته و محور عمودی، نشان‌دهنده‌ی تأخیر یا مدت‌زمان ارسال بسته تا رسیدن بسته به مقصد است. نمودارها در دو شبکه با ابعاد ۸*۸ و ۱۴*۱۴ و در دو شرایط ترافیکی تصادفی و Hotspot مقایسه شده‌اند.

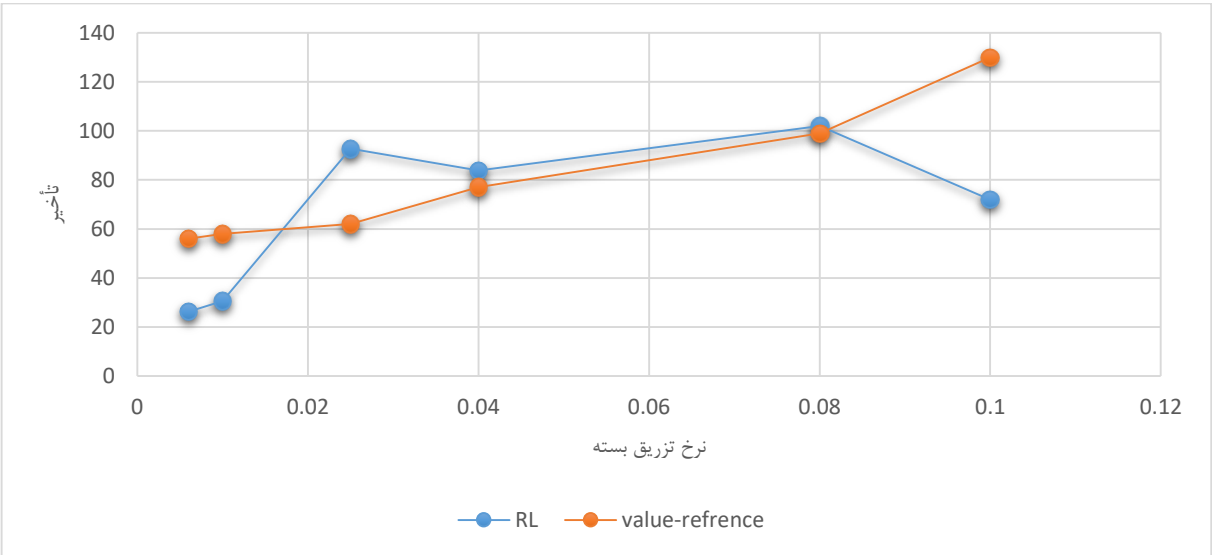


شکل ۴-۱۳) نمودارهای تأخیر در الگوریتم پیشنهادی و مرجع[۵۹] در ترافیک Random در شبکه ۸*۸

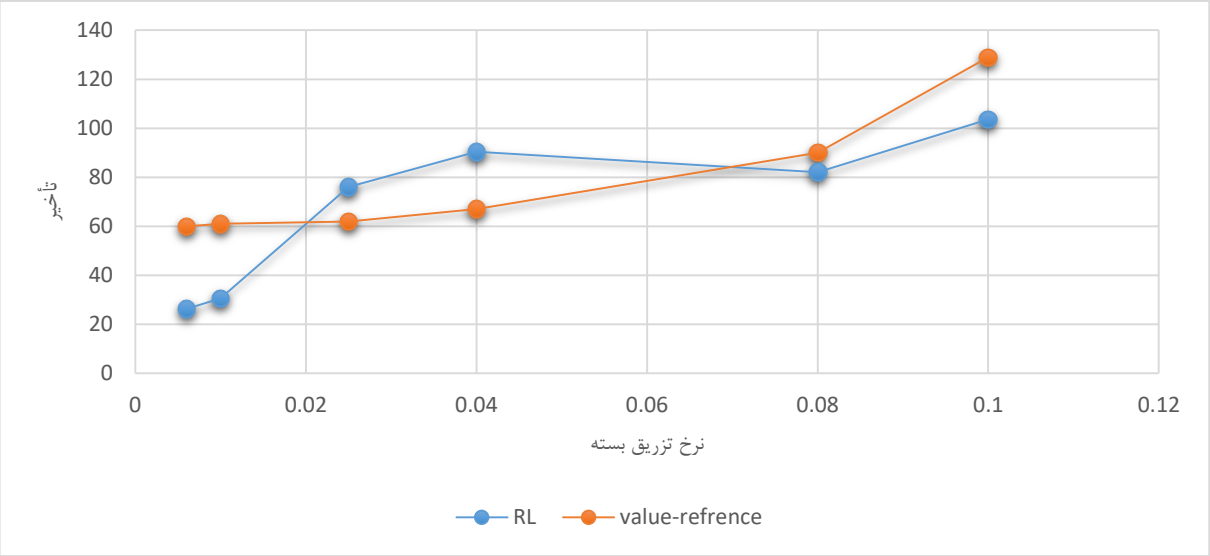


شکل ۴-۱۴) نمودارهای تأخیر در الگوریتم پیشنهادی و مرجع[۵۹] در ترافیک Hotspot در شبکه ۸*۸

همان‌طور که در شکل‌های بالا در شبکه‌ی روی تراشه ۸*۸ نشان داده شده است در هر دو ترافیک، نمودارهای الگوریتم پیشنهادی همواره نتایج بهتری داشته‌اند و تأخیرهای کمتری را نشان داده‌اند.



شکل ۴-۱۵) نمودارهای تأخیر در الگوریتم پیشنهادی و مرجع[۵۹] در ترافیک Random در شبکه ۱۴*۱۴



شکل ۴-۱۶) نمودارهای تأخیر در الگوریتم پیشنهادی و مرجع[۵۹] در ترافیک Hotspot در شبکه ۱۴*۱۴

با توجه به شکل ۴-۱۵ و ۴-۱۶ در شبکه‌ی روی تراشه ۱۴*۱۴ در هر دو ترافیک، الگوریتم پیشنهادی دارای نوسان بوده و از یک نقطه به بعد، نتایج بهتری داشته است.

علت نتایجِ بهتر در الگوریتم پیشنهادی به دلیل استفاده از لینک‌های بی‌سیم می‌باشد. وجود این لینک‌ها باعث می‌شود که گره‌ها تمایل داشته باشند تا داده‌ی خود را (به دلیل سهولت در ارسال داده) از طریق این لینک‌ها ارسال کنند و این می‌تواند تا حد زیادی ازدحام در شبکه را کاهش دهد و داده با تأخیر کمتری به مقصد برسد.

۴-۵ نتیجه‌گیری

در این فصل مقادیر بهینه برای پارامترهای الگوریتم پیشنهادی به دست آورده شده است. نحوه‌ی انتخاب مسیر توسط الگوریتم پیشنهادی با یادگیری تقویتی در حالت‌های مختلف بررسی شده است. همچنین نتایج حاصل از این الگوریتم را با الگوریتم مسیریابی XY و الگوریتم مسیریابی WirelessXY در ترافیک‌های مختلف بررسی کردیم. نتایج نشان می‌دهد که تأخیرها در همه‌ی حالات در این الگوریتم کاهش یافته است و الگوریتم پیشنهادی تأخیر را حداقل ۸ درصد بهبود بخشیده است. دلیل این امر کاملاً مشخص است؛ چون الگوریتم پیشنهادی این قابلیت را دارد که در حین ارسال بسته از گره مبدأ به گره مقصد، در صورت برخورد با مسیر پر ازدحام، تغییر مسیر دهد.

فصل ۵: نتیجه‌گیری و جمع‌بندی

۵-۱ مقدمه

در این فصل به جمع‌بندی الگوریتم ارائه شده، پرداخته شده است. همچنین پیشنهادی برای کارهای آینده ارائه کرده‌ایم.

۵-۲ جمع‌بندی و نتیجه‌گیری

در این پایان‌نامه یک الگوریتم مسیریابی مبتنی بر آموزش یادگیری تقویتی در شبکه‌ی روی تراشه‌ای که دارای لینک‌های بی‌سیم است، پیشنهاد شده است که این قابلیت را دارد که در هنگام انتقال بسته از گره مبدأ به گره مقصد، بر اساس شرایط ازدحام موجود در شبکه تصمیم‌گیری کند و در صورت پر ازدحام بودن مسیر، تغییر مسیر دهد. با این روش زمان تأخیر برای ارسال بسته از گره مبدأ به گره مقصد کاهش می‌یابد و بسته سریع‌تر به گره مقصد می‌رسد. همان‌طور که قبلاً گفته شد در این الگوریتم مسیریابی از روش یادگیری Q استفاده شده است، در این روش برای هر گره یک جدول وجود دارد که حالت‌های مختلف حرکت بسته را از آن گره، در شبکه نشان می‌دهد و مقادیر این جدول معادل تأخیری است که صرف می‌شود تا بسته از گره مبدأ به گره مقصد برسد. این جدول ابتدا با مقدار اولیه صفر پر می‌شود و سپس با حرکت بسته از گره‌ای به گره دیگر این جدول به‌روز می‌شود.

وقتی بسته‌ای به یک گره می‌رسد جدول آن گره را بررسی می‌کند، چون هدف انتخاب کم ازدحام‌ترین مسیر است در صورتی که مقدار تأخیر مثبت فرض شود، پایین‌ترین مقدار را انتخاب می‌کند و بسته را در آن مسیر ارسال می‌کند. به طور کلی می‌توان گفت یادگیری در این الگوریتم، با به‌روز کردن و تغییر دادن مقادیر جدول هر گره انجام می‌شود. در حالی که در سایر الگوریتم‌های مبتنی بر آموزش یادگیری تقویتی سیمی این امکان وجود ندارد و گره بسته‌ی خود را در مسیر از پیش تعیین شده ارسال می‌کند و قابلیت تغییر مسیر وجود ندارد. برای ارزیابی عملکرد الگوریتم، این روش با الگوریتم XY و الگوریتم WirelessXY در انواع مختلف ترافیک مقایسه گردید. ارزیابی نشان داد که الگوریتم مسیریابی مبتنی بر یادگیری تقویتی نتایج خوبی را همه حالات نسبت به سایر الگوریتم‌ها نشان می‌دهد و بهبودی حداقل ۸ درصد را در نتایج تأخیر داشته است.

۵-۳ پیشنهاد برای کارهای آتی

الگوریتم یادگیری تقویتی علی‌رغم برتری نسبت به دیگر الگوریتم‌های یادگیری، دارای معایب و محدودیت‌هایی نیز هست؛ مانند آن که در شبکه‌های بزرگ روی تراشه که جداول هر گره حاوی داده‌های بسیار زیاد است، الگوریتم یادگیری تقویتی نمی‌تواند این شبکه‌ها را تحلیل نماید.

بنابراین، به عنوان کار آینده قصد داریم که به جای یادگیری تقویتی معمولی از یادگیری تقویتی عمیق استفاده کنیم تا رویکرد خود را برای ایجاد محیط‌های پیچیده‌تر گسترش دهیم [۲۱، ۳۸، ۵۶]؛ چرا که یادگیری تقویتی عمیق می‌تواند برای شبکه‌های بزرگ و پیچیده با جداول زیاد نیز کارایی داشته باشد. استفاده از یادگیری تقویتی عمیق باعث می‌شود که مشکل محدود بودن اندازه‌ی جداول هر گره مرتفع گردد [۵۷]. همچنین با استفاده از یادگیری عمیق و با تغییر نحوه‌ی چیدمان گره‌ها در شبکه، می‌توان مفهوم مسیریابی در شبکه‌ی روی تراشه را تغییر داد به این صورت که گره‌ها به جای اینکه در یک شبکه قرار داشته باشند هر کدام در یک حلقه قرار دارند و این حلقه‌ها در یک گره‌هایی به هم می‌رسند. اگر بخواهیم یک بسته را ارسال کنیم بسته به اینکه گره مقصد در حلقه خودش باشد یا نه فرق می‌کند. اگر در حلقه خودش باشد به راحتی بسته به مقصد می‌رسد در غیر این صورت بسته به گره متصل به دو حلقه می‌رسد و وارد حلقه دیگر می‌شود در حلقه جدید اگر گره مقصد بود بسته تحویل گره مقصد داده می‌شود در غیر این صورت از طریق گره متصل دو حلقه، وارد حلقه دیگر می‌شود و این مراحل ادامه پیدا می‌کند تا بسته به گره مقصد برسد. در این روش مسیریابی وجود ندارد چون همیشه در هنگام ارسال بسته، بسته در جهت عقربه‌های ساعت فرستاده می‌شود. ما بی‌نهایت حالت برای در نظر گرفتن حلقه‌ها داریم که همه گره‌ها را پوشش دهند. این مرجع بررسی می‌کند چطوری حلقه‌ها را در نظر بگیریم، چند تا حلقه در نظر بگیریم، هر حلقه از کدام گره‌ها بگذرد، کدام حالت، حالت بهینه است و ...و این اعمال با Deep Reinforcement Learning انجام می‌شود[۱۶].

ایده‌ی دیگر برای کاهش تأخیر در ارسال بسته از گره مبدأ به گره مقصد، استفاده از مقدار هزینه‌ی بی‌سیم به صورت متغیر است؛ به این صورت که با استفاده از یادگیری تقویتی در هر مرحله مقدار هزینه‌ی بی‌سیم را (بسته به شرایط ازدحام در شبکه) مقداری متغیر انتخاب نماییم. مقداری که باعث می‌شود عملکرد شبکه بهتر، بسته‌ها سریع‌تر و از مسیر کم ازدحام‌تر به مقصد برسد.

مراجع

[1] Ganguly, Amlan, Kevin Chang, Sujay Deb, Partha Pratim Pande, Benjamin Belzer, and Christof Teuscher. "Scalable hybrid wireless network-on-chip architectures for multicore systems." *IEEE Transactions on Computers* 60, no. 10 (2010): 1485-1502.

[2] Flich, José, Samuel Rodrigo, and José Duato. "An efficient implementation of distributed routing algorithms for NoCs." In *Second ACM/ieee international symposium on networks-on-chip (NOCs 2008)*, pp. 87-96. IEEE, 2008.

[3] Chen, Zhongsheng, Ying Zhang, Zebo Peng, and Jianhui Jiang. "A Deterministic-Path Routing Algorithm for Tolerating Many Faults on Wafer-Level NoC." In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1337-1342. IEEE, 2019.

[4] Sun, Meidong, Qinrang Liu, Binghao Yan, and Xiaolong Wang. "Minimally Buffered Router and Deflection Routing Algorithm for 3D Mesh NoC." In *Recent Developments in Intelligent Computing, Communication and Devices*, pp. 515-522. Springer, Singapore, 2019.

[5] Chen, Kun-Chih. "Game-based thermal-delay-aware adaptive routing (gtdar) for temperature-aware 3d network-on-chip systems." *IEEE Transactions on Parallel and Distributed Systems* 29, no. 9 (2018).

[6] Vu, The H., Ogbodo Mark Ikechukwu, and Abderazek Ben Abdallah. "Fault-tolerant spike routing algorithm and architecture for three dimensional noc-based neuromorphic systems." *IEEE Access* 7 (2019): 90436-90452.

[7] Fu, Binzhang, Yinhe Han, Huawei Li, and Xiaowei Li. "ZoneDefense: A fault-tolerant routing for 2-D meshes without virtual channels." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, no. 1 (2013): 113-126.

[8] Kao, Sheng-Chun, Chao-Han Huck Yang, Pin-Yu Chen, Xiaoli Ma, and Tushar Krishna. "Reinforcement learning based interconnection routing for adaptive traffic optimization." In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, pp. 1-2. 2019.

[9] Farahnakian, Fahimeh, Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. "Q-learning based congestion-aware routing algorithm for on-chip network." In *2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications*, pp. 1-7. IEEE, 2011.

[10] Farahnakian, Fahimeh, Masoumeh Ebrahimi, Masoud Daneshtalab, Juha Plosila, and Pasi Liljeberg. "Adaptive reinforcement learning method for networks-on-chip." In *2012 International Conference on Embedded Computer Systems (SAMOS)*, pp. 236-243. IEEE, 2012.

[11] Norollah, Amin, Danesh Derafshi, Hakem Beitollahi, and Ahmad Patooghy. "PAT-Noxim: A Precise Power & Thermal Cycle-Accurate NoC Simulator." In *2018 31st IEEE International System-on-Chip Conference (SOCC)*, pp. 163-168. IEEE, 2018.

[12] Kumar, Shailesh, and Risto Miikkulainen. "Dual reinforcement Q-routing: An on-line adaptive routing algorithm." In *Proceedings of the artificial neural networks in engineering Conference*, pp. 231-238. 1997.

[13] Simeone, Osvaldo, (2018) "A very brief introduction to machine learning with applications to communication systems." IEEE Transactions on Cognitive Communications and Networking 4, no: 648-664.

[14] Sutton, Richard S. (1992) "Introduction: The challenge of reinforcement learning." In Reinforcement Learning, Springer, Boston, MA, pp. 1-3.

[15] Mahadevan, Sridhar. (1996) "Average reward reinforcement learning: Foundations, algorithms, and empirical results." Machine learning 22, no. 1-3: 159-195.

[16] Lin, Ting-Ru, Drew Penney, Massoud Pedram, and Lizhong Chen. "Optimizing routerless network-on-chip designs: an innovative learning-based framework." *arXiv preprint arXiv:1905.04423* (2019).

[17] Kundu, Santanu, and Santanu Chattopadhyay. *Network-on-chip: the next generation of system-on-chip integration*. Taylor & Francis, 2014.

[18] Kalahroudi, Parisa Mazaheri, Elham Yaghoubi, and Behrang Barekatain. "IAM: an improved mapping on a 2-D network on chip to reduce communication cost and energy consumption." *Photonic Network Communications* (2020): 1-15.

[19] Rad, Farhad, Midia Reshadi, and Ahmad Khademzadeh. "A novel arbitration mechanism for crossbar switch in wireless network-on-chip." *Cluster Computing* (2020): 1-14.

[20] Venugopalan, Ravi, Sandeep Kumar Goel, and Yun-Han Lee. "Network-on-chip system and a method of generating the same." U.S. Patent Application 16/879,567, filed September 3, 2020.

[21] Jog, Suraj, Zikun Liu, Antonio Franques, Vimuth Fernando, Haitham Hassanieh, Sergi Abadal, and Josep Torrellas. "Millimeter Wave Wireless Network on Chip Using Deep Reinforcement Learning."

[22] Zhang, Wenfei, and Yaoyao Ye. "A Table-Free Approximate Q-Learning Based Thermal-Aware Adaptive Routing for Optical NoCs." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020).

[23] Mogharrabi-Rad, Zahra, and Elham Yaghoubi. "ADFT: an adaptive, distributed, fault-tolerant routing algorithm for 3D mesh-based networks-on-chip." *International Journal of Internet Technology and Secured Transactions* 10, no. 4 (2020): 481-490.

[24] Eltaras, Tamer Ahmed, William Fornaciari, and Davide Zoni. "Partial packet forwarding to improve performance in fully adaptive routing for cache-coherent nocs." In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 33-40. IEEE, 2019

[25] Rahaman, Munshi Mostafijur, Prasun Ghosal, and Tuhin Subhra Das. "Latency, throughput and power aware adaptive noc routing on orthogonal convex faulty region." *Journal of Circuits, Systems and Computers* 28, no. 04 (2019): 1950055

[26] C. J. Watkins and P. Dayan. (1992) “Q-learning”, Machine learning, vol. 8, no. 3-4, pp. 279– 292.

[27] Das, Tuhin Subhra, Prasun Ghosal, and Navonil Chatterjee. "Virtual circuit switch based orderly delivery of packets in adaptive NoC routing." In *Proceedings of the 12th International Workshop on Network on Chip Architectures*, pp. 1-6. 2019.

[28] Schoeberl, Martin, Luca Pezzarossa, and Jens Sparsø. "A minimal network interface for a simple network-on-chip." In *International Conference on Architecture of Computing Systems*, pp. 295-307. Springer, Cham, 2019.

[29] Song, Yang, and Bill Lin. "Uniform Minimal First: Latency Reduction in Throughput-Optimal Oblivious Routing for Mesh-Based Networks-on-Chip." *IEEE Embedded Systems Letters* 11, no. 3 (2019): 81-84.

[30] Wang, Liang, Xiaohang Wang, Ho-Fung Leung, and Terrence Mak. "A non-minimal routing algorithm for aging mitigation in 2D-mesh NoCs." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, no. 7 (2018): 1373-1377.

[31] Chen, Yu-Yin, En-Jui Chang, Hsien-Kai Hsin, Kun-Chih Chen, and An-Yeu Wu. "Path-diversity-aware fault-tolerant routing algorithm for network-on-chip systems." *IEEE Transactions on Parallel and Distributed Systems* 28, no. 3 (2016): 838-849.

[32] Zhou, Jun, Huawei Li, Tiancheng Wang, and Xiaowei Li. "LOFT: A low-overhead fault-tolerant routing scheme for 3D NoCs." *Integration* 52 (2016): 41-50.

[33] Kurokawa, Yota, and Masaru Fukushi. "Passage of Faulty Nodes: A Novel Approach for Fault-Tolerant Routing on NoCs." *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 102, no. 12 (2019): 1702-1710.

[34] Ye, Yaoyao, Wenfei Zhang, and Weichen Liu. "Thermal-Aware Design and Simulation Approach for Optical NoCs." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019).

[35] Shahabinejad, Narges, and Hakem Beitollahi. "Q-Thermal: A Q-learning based Thermal-aware Routing Algorithm for 3D Network On-Chips." *IEEE Transactions on Components, Packaging and Manufacturing Technology* (2020).

[36] Bishnoi, Rimpy. "Hybrid fault tolerant routing algorithm in NoC." *Perspectives in Science* 8 (2016): 586-588.

[37] Mubeen, Saad, and Shashi Kumar. "Designing efficient source routing for mesh topology network on chip platforms." In *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pp. 181-188. IEEE, 2010.

[38] Li, Zhiyao, and Yiwei Li. "Use Deep Reinforcement Learning for NoC Design."

[39] Murali, Srinivasan, Luca Benini, and Giovanni De Micheli. "Method to design network-on-chip (NOC)-based communication systems." U.S. Patent 8,042,087, issued October 18, 2011.

[40] Kumar, Sailesh, Amit Patankar, and Eric Norige. "System level simulation in Network on Chip architecture." U.S. Patent 10,496,770, issued December 3, 2019.

[41] Liang, Jian, Sriram Swaminathan, and Russell Tessier. "aSOC: A scalable, single-chip communications architecture." In *Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques (Cat. No. PR00622)*, pp. 37-46. IEEE, 2000.

[42] Xiang, Yande, Jianyi Meng, and De Ma. "A Q-routing based self-regulated routing scheme for network-on-chip." In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 177-181. IEEE, 2017.

[43] Zheng, Hao, and Ahmed Louri. "An energy-efficient network-on-chip design using reinforcement learning." In *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1-6. 2019.

[44] Farahnakian, Fahimeh, Masoumeh Ebrahimi, Masoud Daneshtalab, Juha Plosila, and Pasi Liljeberg. "Optimized Q-learning model for distributing traffic in on-chip networks." In *2012 IEEE 3rd International Conference on Networked Embedded Systems for Every Application (NESEA)*, pp. 1-8. IEEE, 2012.

[45] TANG, Ming-hua, and L. I. N. Jing. "A Quantitative Study on NoC Traffic Scenarios." *DEStech Transactions on Computer Science and Engineering* iece (2018).

[46] Catania, Vincenzo, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. "Noxim: An open, extensible and cycle-accurate network on chip simulator." In *2015 IEEE 26th international conference on application-specific systems, architectures and processors (ASAP)*, pp. 162-163. IEEE, 2015.

[47] Khan, Sarzamin, Sheraz Anjum, Usman Ali Gulzari, and Frank Sill Torres. "Comparative analysis of network-on-chip simulation tools." *IET Computers & Digital Techniques* 12, no. 1 (2017): 30-38.

[48] Pires, Ivan Luiz Pedroso, Marco Antonio Zanata Alves, and Luiz Carlos Pessoa Albini. "Trace-driven and processing time extensions for Noxim simulator." *Design Automation for Embedded Systems* 23, no. 1-2 (2019): 41-55.

[49] Sakamoto, Shinji, Ryoichiro Obukata, Tetsuya Oda, Leonard Barolli, Makoto Ikeda, and Admir Barolli. "Performance analysis of two wireless mesh network architectures by WMN-SA and WMN-TS simulation systems." *Journal of High Speed Networks* 23, no. 4 (2017): 311-322.

[50] Mohammadi, Robab, and Hossein Boroumand Noghabi. "SAT: Simulated Annealing and Tabu Search Based Routing Algorithm for Wireless Sensor Networks." *International Journal of Computer Networks and Communications Security* 4, no. 10 (2016): 286.

[51] Dowsland, Kathryn Anne, and Jonathan Thompson. "Simulated annealing." *Handbook of natural computing* (2012): 1623-1655.

[52] Wang, XiaoJun, Feng Shi, and Hong Zhang. "KLSAT: An Application Mapping Algorithm Based on Kernighan–Lin Partition and Simulated Annealing for a Specific WK-Recursive NoC Architecture." In *IFIP International Conference on Network and Parallel Computing*, pp. 31-42. Springer, Cham, 2019.

[53] Qiming, Fu, Hu Wen, Liu Quan, Luo Heng, Hu Lingyao, and Chen Jianping. "Residual Sarsa algorithm with function approximation." *Cluster Computing* 22, no. 1 (2019): 795-807.

[54] Jiang, Hao, Renjie Gui, Zhen Chen, Liang Wu, Jian Dang, and Jie Zhou. "An Improved Sarsa ($\lambda $) Reinforcement Learning Algorithm for Wireless Communication Systems." *IEEE Access* 7 (2019): 115418-115427.

[55] Qiang, Wang, and Zhan Zhongli. "Reinforcement learning model, algorithms and its application." In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pp. 1143-1146. IEEE, 2011.

[56] Lin, Ting-Ru, Drew Penney, Massoud Pedram, and Lizhong Chen. "A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study." In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 99-110. IEEE, 2020.

[57] Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. "Deep reinforcement learning: A brief survey." *IEEE Signal Processing Magazine* 34, no. 6 (2017): 26-38.

[58] Deb, Sujay, and Hemanta Kumar Mondal. "Wireless network-on-chip: a new era in multi-core chip design." In *2014 25nd IEEE International Symposium on Rapid System Prototyping*, pp. 59-64. IEEE, 2014.

[59] Farahnakian, Fahimeh, Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. "Bi-LCQ: A low-weight clustering-based Q-learning approach for NoCs." *Microprocessors and Microsystems* 38, no. 1 (2014): 64-75.

Abstract

Networks on chip can be considered as an alternative to bus technology in chips with a large number of cores. In these chips, the cores are connected by wire or wireless. The main reason for using wireless communication is to reduce latency and power consumption in the network. In networks on chip, as other conventional networks, the issue of routing is important. An appropriate routing algorithm in these networks is an algorithm that while avoiding latency and increasing efficiency, also avoids deadlock. It is clear that as wireless links are added to the network, the complexity of the routing algorithm will also increase. It is worth mentioning that by using routing algorithms based on machine learning methods, routing can be done better and with less congestion.

In previous works, a learning algorithm for a on-chip wired network has been proposed. Because of the high demand for wireless links in a wireless network, the learning algorithm can greatly reduce network congestion. On the other hand, the basis of networks on a wireless chip is different from wired networks, which will lead to differences in the implementation of the reinforcement learning algorithm. Therefore, in this thesis, an attempt is made to provide a routing algorithm for wireless chip networks with the help of reinforcement learning. Since the proposed reinforcement learning algorithm has the ability to detect congestive path and change the path, it can make a better decision when sending the packet from the source node to the destination node. Also, the results of the proposed method have been compared with previous routing algorithms, and in the proposed algorithm, at least 8% improvement has been made.

Keywords: Network On Chip (NOC), Q-Learning Algorithm, Reinforcement Learning (RL), Routing Algorithm, Deadlock, Wireless Network On Chip, Noxim

Shahrood University of Technology

Faculty of Computer Engineering

M.Sc. Thesis in Artificial Intelligence Engineering

Improving the Routing Algorithm of Wireless Networks on Chip using Reinforcement Learning

By: Zohreh Harati

Supervisor:
Dr. Esmaeel Tahanian

Advisor:
Dr. Alireza Tajary
Dr. Mansoor Fateh

September 2020