دوره هفدهم، شماره بهار و تابستان ۱۴۰۳

مجله فناوری اطلاعات در طراحی مهندسی

Information Technology in Engineering Design

http://sanad.iau.ir/journal/ited

جای‌گذاری کنترل‌کننده‌ها در شبکه‌های تعریف شده‌ی نرم‌افزار با استفاده از یک الگوریتم فراابتکاری بر پایه‌ی ژنتیک

مهناز خجند[۱] کامبیز مجیدزاده*[۲] محمد مصدری[۳] یوسف فرهنگ[۴]

(۱) دانشکده فنی مهندسی، دانشگاه آزاد اسلامی، واحد ارومیه، ارومیه، ایران

(۲) دانشکده فنی مهندسی، دانشگاه آزاد اسلامی، واحد ارومیه، ارومیه، ایران*

(۳) دانشکده فنی مهندسی، دانشگاه آزاد اسلامی، واحد ارومیه، ارومیه، ایران

(۴) دانشکده فنی مهندسی، دانشگاه آزاد اسلامی، واحد خوی، خوی، ایران

(تاریخ دریافت :۱۴۰۳/۰۴/۲۹ تاریخ پذیرش:۱۴۰۳/۰۶/۰۵)

چکیده

شبکه‌های نرم‌افزارمحور شامل جداسازی صفحه کنترل از صفحه داده است. در شبکه‌های نرم‌افزارمحور کنترل شبکه توسط موجودیتی به نام کنترل‌کننده که در صفحه کنترل قرار دارد، تعیین می‌شود. تعیین تعداد و مکان بهینه کنترل‌کننده‌ها در صفحه کنترل به عنوان مسئله‌ی جای‌گذاری کنترل‌کننده‌ها شناخته می‌شود. در این مقاله در دو فاز مسئله‌ی جای‌گذاری کنترل‌کننده‌ها با استفاده از خوشه‌بندی و الگوریتم بهینه‌ساز گله اسب (HOA) حل شده است. در فاز اول تعداد کنترل‌کننده‌ها با استفاده از خوشه‌بندی تعیین می‌شود. الگوریتم استفاده‌شده برای خوشه‌بندی ورژنی از K-means می‌باشد که علاوه بر خوشه‌بندی سوئیچ‌ها، تعداد کنترل‌کننده‌ها را نیز تعیین می‌کند. در فاز دوم برای تعیین مکان بهینه کنترل‌کننده در داخل هر خوشه از الگوریتم فراابتکاری گله اسب استفاده می‌شود. در اینجا برای بالا بردن قدرت اکتشاف و بهره‌برداری الگوریتم گله اسب و همین‌طور بهبود همگرایی و سرعت آن از عملگرهای ژنتیک و استراتژی تضاد نخبگان استفاده شده است که الگوریتم گله اسب بهبود یافته (MHOA) نام‌گذاری می‌شود. نتایج نشان می‌دهد که روش پیشنهادی در مقایسه با سایر الگوریتم‌های مطرح شده از نظر تأخیر انتها به انتها، عدم توازن بار و مصرف انرژی عملکرد بهتری دارد. روش پیشنهادی با کاهش ۹,۶۶٪ عدم توازن بار، ۱۹,۶۵٪ تأخیر انتها به انتها و ۸,۴۳٪ میانگین مصرف انرژی به نتایج بهتری نسبت به الگوریتم‌های دیگر دست یافته است.

کلمات کلیدی: شبکه تعریف‌شده توسط نرم‌افزار، مسئله قرارگیری کنترل‌کننده، الگوریتم های فراابتکاری، الگوریتم بهینه‌سازی مبتنی بر تضاد نخبگان، الگوریتم ژنتیک

*عهده‌دار مکاتبات:

کامبیز مجیدزاده

نشانی: دانشکده فنی مهندسی، دانشگاه آزاد اسلامی، واحد ارومیه ارومیه، ایران

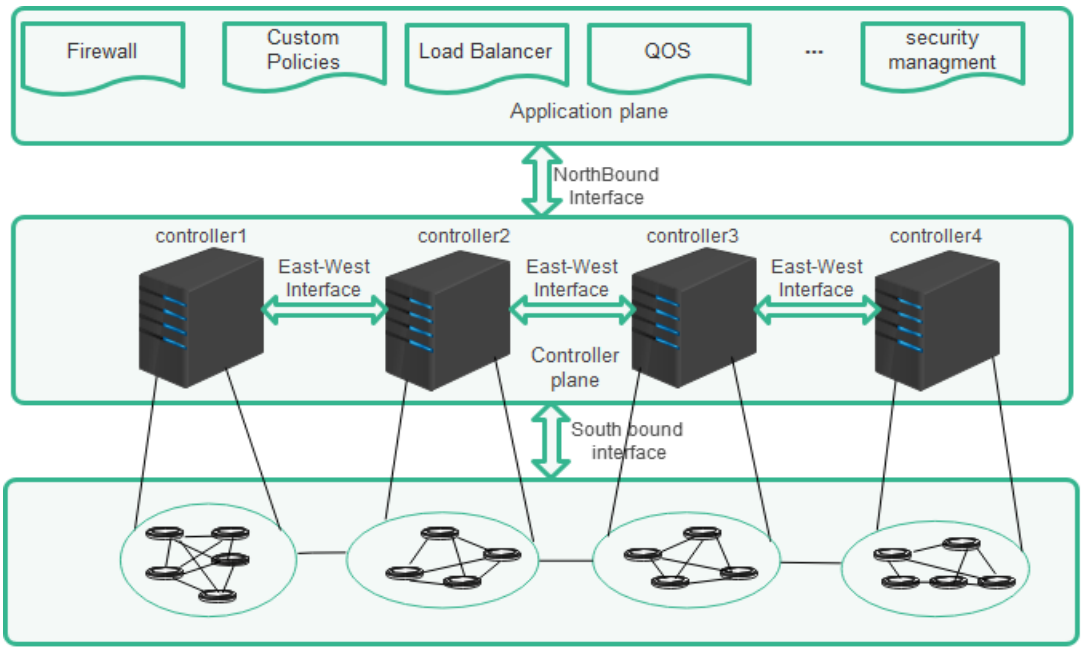
پست الکترونیکی:**Kambiz.majidzadeh@iau.ac.ir**

۱- مقدمه

در حال حاضر، با توجه به افزایش تعداد کاربران، حجم بالای داده‌ها، پیچیدگی و ترافیک داده‌ها، شبکه‌های سنتی قادر به پاسخگویی نبوده و با چالش‌های بسیاری از لحاظ مدیریت و پاسخ‌دهی روبه‌رو هستند [4-1]. در راستای پاسخ به چالش‌های مطرح‌شده، معماری شبکه‌های نرم‌افزارمحور[1] به عنوان یک راه‌حل پیشنهاد شده است. شبکه‌های نرم‌افزارمحور یک معماری شبکه است که مدیریت (صفحه کنترل) را از پردازش داده (صفحه داده) جدا می‌کند و در نتیجه بهبود مدیریت شبکه‌های گسترده را فراهم می‌سازد. شکل ۱ نشان‌دهنده معماری شبکه‌های نرم‌افزارمحور است که شامل سه لایه‌ی داده، کنترل و برنامه می‌باشد. ارتباط بین این لایه‌ها از طریق رابط‌های جنوبی و شمالی صورت می‌گیرد. همچنین، رابط‌های شرقی و غربی نیز پروتکل ارتباطی بین کنترل‌کننده‌های مختلف در شبکه را تعریف می‌کنند [5,6]. صفحه داده می‌تواند با اختصاص وظایف مسیریابی و سیگنال‌دهی به یک موجودیت جداگانه به نام کنترل‌کننده که در صفحه کنترل قرار دارد، امکان ارسال سریع داده‌ها را داشته باشد. استفاده از فقط یک کنترل‌کننده ممکن است منجر به ایجاد محدودیت‌هایی در شبکه شود، زیرا آن کنترل‌کننده به عنوان یک گلوگاه عمل کرده و بار کاری کل شبکه بر روی آن تمرکز می‌یابد؛ بنابراین، اگر کنترل‌کننده از کار افتد، کل شبکه با شکست مواجه خواهد شد. از این‌رو، داشتن چندین کنترل‌کننده در صفحه‌ی کنترل بسیار حیاتی است. این کنترل‌کننده‌ها در سراسر شبکه توزیع می‌شوند و باعث کاهش تاخیر در شبکه، افزایش قابلیت اطمینان، بهبود زمان پاسخ و تعادل بار می‌شوند. هر سوئیچ به حداقل یک کنترل‌کننده اختصاص می‌یابد و هر کنترل‌کننده مسئول مدیریت یک مجموعه از سوئیچ‌ها [7,8] است.

تعیین تعداد و موقعیت بهینه کنترل‌کننده‌ها و اختصاص سوئیچ‌ها به آن‌ها مسئله‌ی جای‌گذاری کنترل‌کننده‌ها[2] نامیده می‌شود. حل بهینه‌ی مسئله‌ی جای‌گذاری کنترل‌کننده‌ها برای عملکرد شبکه بسیار حیاتی است زیرا تعداد و موقعیت کنترل‌کننده‌ها باید قابل تطبیق با تغییرات اندازه و بار شبکه باشد، به‌طوری‌که بتواند باعث بهبود کارآیی شبکه از لحاظ تاخیر، مصرف انرژی، توزیع یکنواخت ترافیک شبکه و قابلیت اعتماد شود. با این حال، مسئله‌ی جای‌گذاری کنترل‌کننده‌ها مسئله‌ای پیچیده است و حل آن آسان نمی‌باشد و به عنوان یک مسئله‌ی ان-پی سخت[3] شناخته می‌شود [9,10].

[1] Software Defined Networks (SDN)
[2] Controller Placement Problem
[3] NP-hard



شکل ۱: معماری پیشنهاد شده برای SDN

در فاز اولیه‌ی این مقاله، ورژنی از K-means برای حل مسئله‌ی جای‌گذاری کنترل‌کننده‌ها استفاده شده است. K-means به دلیل سرعت، پیچیدگی کم و کارایی آن‌ها یکی از روش‌های پرطرفدار در خوشه‌بندی است. یکی از محدودیت‌های اساسی K-means نیاز به مشخص کردن تعداد خوشه‌ها به عنوان ورودی الگوریتم است. روش استفاده شده در این مقاله ورژنی از K-means است که نیاز به تعیین تعداد خوشه‌ها از قبل ندارد و با تعیین خودکار تعداد بهینه‌ی خوشه‌ها، K-means را گسترش می‌دهد [11,12]. این الگوریتم می‌تواند به عنوان یک گزینه ایده‌آل برای خوشه‌بندی و یافتن تعداد کنترل‌کننده‌ها در مسئله جای‌گذاری کنترل‌کننده‌ها استفاده شود. در این مرحله، گره‌ها با در نظر گرفتن مکان گره‌ها و ترافیک شبکه خوشه‌بندی می‌شوند. تعداد کنترل‌کننده‌ها برابر با تعداد خوشه‌های تعیین‌شده می‌باشد و به عنوان ورودی فاز بعدی مقاله است. در فاز دوم، یک الگوریتم فراابتکاری به نام الگوریتم بهینه‌سازی اسب بهبودیافته (MHOA) برای یافتن مکان کنترل‌کننده در داخل هر خوشه معرفی می‌شود. تکنیک‌های فراابتکاری روش‌هایی هستند که برای جست‌وجو، از یک زیرمجموعه‌ی خاص از فضای جست‌وجو به صورت کارآمد استفاده می‌کنند تا به راه‌حل‌های نزدیک به بهینه برسند. این روش‌ها در حل مسائل پیچیده و ان-پی سخت مانند مسئله جای‌گذاری کنترل‌کننده‌ها بسیار کارآمد هستند [13,14]. الگوریتم MHOA ورژن بهبود یافته‌ی الگوریتم بهینه‌ساز گله اسب[1] است [15]. الگوریتم **گله اسب** در حل بسیاری از مسائل دنیای واقعی به خوبی عمل می‌کند. با این حال، در

[1] Horse Herd Optimization Algorithm

برخی مسائل پیچیده ممکن است که در بهینه‌ی محلی گیر کند و دچار پایان زودهنگام شود. MHOA برای افزایش قدرت اکتشاف[1] و بهره‌وری[2] و بهبود همگرایی، از استراتژی تضاد نخبگان[3] و عملگرهای ژنتیک استفاده می‌کند. در ادامه از MHOA برای حل مسئله جای‌گذاری کنترل‌کننده‌ها استفاده می‌شود. برای ارزیابی الگوریتم پیشنهادی، از مجموعه داده‌ی اینترنتی توپولوژی زو[4] استفاده شده است. این مطالعه نشان می‌دهد که الگوریتم MHOA در مقایسه با دیگر روش‌های فراابتکاری به‌طور موثری باعث بهبود تأخیر انتها به انتها، مصرف انرژی و تعادل بار می‌گردد.

خلاصه‌ی کارهای انجام شده در این مقاله به شرح زیر است:

- پیدا کردن تعداد کنترل‌کننده‌ها.
- بهبود الگوریتم گله اسب با استفاده از استراتژی تضاد نخبگان و استفاده از عملگرهای ژنتیک(MHOA).
- استفاده از MHOA برای حل مسئله‌ی جای‌گذاری کنترل‌کننده‌ها.
- ارزیابی قابلیت مقیاس‌پذیری MHOA برای قرار دادن کنترل‌کننده‌ها در دو شبکه با اندازه‌ی متفاوت از مجموعه داده‌ی توپولوژی زو.
- مقایسه‌ی نتایج الگوریتم پیشنهادی با نتایج الگوریتم‌های فراابتکاری دیگر از لحاظ تاخیر انتها به انتها، مصرف انرژی و تعادل بار.

بخش ۲ این مقاله، خلاصه‌ای از مطالعات قبلی انجام‌شده در این حوزه را ارائه داده و فصل ۳ الگوریتم‌های استفاده‌شده در این مقاله، به‌عنوان مثال HOA و K-means بهبودیافته توضیح داده شده و با معرفی MHOA به چگونگی حل CPP می‌پردازد. بخش ۴ نحوه‌ی ارزیابی کار را نشان می‌دهد و درنهایت، در بخش ۶ خلاصه‌ای از نتایج به‌دست آمده از مقاله ارائه می‌شود.

۲- کارهای گذشته

در این بخش، ما برخی از کارهای انجام شده در **حوزه** CPP با استفاده از روش‌های فراابتکاری را معرفی می‌کنیم. مسئله‌ی جای‌گذاری کنترل‌کننده به‌صورت پویا برای نخستین بار توسط [16] مورد مطالعه قرار گرفت. نویسندگان مسئله‌ی "جای‌گذاری کنترل‌کننده به‌صورت پویا" را به عنوان یک برنامه‌ی خطی عدد صحیح تعریف کردند و دو

[1] Exploration
[2] Exploitation
[3] Elite Opposition-Based Learning
[4] Internet Topology ZOO

الگوریتم ترکیبی برای حل آن در موارد بزرگ‌تر پیشنهاد دادند. با این حال، آن‌ها اعتبارپذیری را در نظر نگرفتند و تنها بر کاهش زمان راه‌اندازی جریان و هزینه ارتباطات تمرکز کردند. نویسندگان در [17] یک استراتژی جدید برای قرار دادن کنترل‌کننده‌های SDN در شبکه با استفاده از مسئله‌ی کوله‌پشتی ۰-۱ و الگوریتم بهینه‌سازی مار گارتنر معرفی کردند. آزمایشات آن‌ها نشان داد که این روش به‌طور مؤثری تأخیرها را هم در WAN و هم در SDN کاهش می‌دهد. این رویکرد گرچه توانست استفاده از منابع شبکه و خدمات‌رسانی به مشتریان را بهبود بخشد، ولی باید توجه کرد که اجرای آن ممکن است نیازمند منابع محاسباتی قابل توجهی باشد که ممکن است محدودیت‌هایی در محیط‌های شبکه داشته باشد. علاوه بر این، در این مقاله به مسئله مقیاس‌پذیری پرداخته نشده است.

کیلی و همکاران در [18]، یک الگوریتم قرارگیری کنترل‌کننده ارائه کردند که با استفاده از تطبیق پلی-پایدار، توانایی مدیریت شبکه‌های بزرگ را داشت و سوئیچ‌ها را بین کنترل‌کننده‌ها توزیع می‌کرد. این الگوریتم عواملی مانند تاخیر و بار را در نظر گرفته بود تا سوئیچ‌های باقی‌مانده را تخصیص دهد. این الگوریتم در مقایسه با الگوریتم‌های موجود، از نظر توازن بار و کاهش تاخیر، عملکرد بهتری داشت. قابلیت مقیاس‌پذیری و کارآمدی این الگوریتم در شبکه‌های WAN بر روی سه شبکه ISP واقعی با مقیاس متوسط و بزرگ آزمایش شد. در این مقاله، تأثیر قرارگیری کنترل‌کننده بر محدودیت‌های در دسترسی و زمان همگرایی شبکه‌های SD-WAN مشخص نشده است.

نویسندگان در [19]، یک تکنیک به نام MHNSGA-II پیشنهاد دادند که عوامل مختلفی مانند تاخیر، قابلیت اطمینان و زمان محاسبه را در نظر می‌گرفت. آن‌ها روشی جدید برای ایجاد راه‌حل‌های اولیه پیشنهاد دادند که امکان بررسی مناطق مختلف در فضای راه‌حل را فراهم کرده و کیفیت راه‌حل را افزایش می‌دهد. با این حال، در این کار فرض شد که توپولوژی شبکه و ترافیک ثابت و شناخته شده است، که ممکن است در شرایط واقعی صحت نداشته باشد. در [20]، یک الگوریتم جدید برای حل مسئله‌ی CPP معرفی شده است. آن‌ها از رویکرد فراابتکاری بهینه‌سازی گروه ذرات (PSO) استفاده کردند و تاخیر سرتاسری را به عنوان یک معیار برای حل مسئله مورد بررسی قرار دادند. این الگوریتم با وجود اینکه به‌طور موثری تاخیر انتشار را کاهش داد، اثر بار ترافیک پایدار بر روی کنترل‌کننده‌ها را نادیده گرفت.

در مقاله [21]، نویسندگان از الگوریتم بهینه‌سازی گرگ خاکستری (GWO) برای حل CPP استفاده کردند (CCPGWO). در این کار ظرفیت کنترل‌کننده‌ها نیز در نظرگرفته شد. هدف این پژوهش یافتن بهترین مکان و عملکرد برای کنترل‌کننده‌ها در یک شبکه بود. نتایج نشان دادند که CCPGWO در کمینه کردن تاخیر بدترین حالت، اجتناب از نقاط بهینه محلی و دستیابی به نتایجی که نزدیک به بهینه جهانی هستند، موفق بوده است. در این مقاله

فرض شده است که هر گره در شبکه قابلیت OpenFlow را دارد، ولی این شرایط همیشه در واقعیت صدق نمی‌کند. در مقاله‌ی [22]، نویسندگان با در نظر گرفتن عواملی مانند انواع مختلف کنترل‌کننده، منابع محدود در موقعیت‌های لبه و ضرورت کاهش تاخیر، یک روش جدید برای قرارگیری کنترل‌کننده‌ها در شبکه‌های انتقال نوری ارائه کردند. آن‌ها همچنین یک روش بر پایه‌ی روش‌های یادگیری ماشین، برای کمک به پیش‌بینی مکان‌های بهینه معرفی کردند. این رویکرد دارای چندین مزیت مانند کاهش تاخیر، بهبود عملکرد و صرفه‌جویی در هزینه بود. با این حال، قابلیت اعمال این مطالعه محدود به شبکه‌های انتقال چند دامنه است، که باعث کاهش قابلیت استفاده گسترده‌ی آن می‌شود. علاوه بر این، پیاده‌سازی و مدیریت این چارچوب ممکن است نیاز به منابع قابل توجه و دانش تخصصی داشته باشد. در مقاله [23]، کاستین و همکاران با استفاده از تئوری بازی توانستند بین تعادل بار در شبکه و تاخیر توازن برقرار کنند. آن‌ها فرض کردند که توپولوژی شبکه پیش‌تعیین شده است، که این ممکن است همیشه در سناریوهای واقعی صحیح نباشد.

نویسندگان در [24]، پروژه SQHCP را برای حل مسئله‌ی CPP ارائه دادند، که هدف قرار دادن کنترلرهای نامتجانس به صورت امن و با اطمینان از کیفیت بود. آن‌ها نیازهای امنیت و کیفیت سرویس و در عین حال خطاهای لایه کنترل و تضمین استفاده متعادل از کنترلرها را در نظر گرفتند. با این حال ، پیاده‌سازی این رویکرد ممکن است نسبت به روش‌های موجود منابع و پیچیدگی بیشتری را مورد نیاز داشته باشد. در [1]، رامایا و همکاران یک تکنیک نوآورانه بر اساس نظریه گراف ارائه دادند تا تعداد لازم کنترلرها را تعیین کنند. آن‌ها از الگوریتم جست‌وجوی تابو یکپارچه پارتو برای شناسایی بهترین مکان برای این کنترلرها استفاده کردند. آن‌ها همچنین یک روش فراابتکاری برای مدیریت سناریوهای پویا مانند خرابی‌های پیوند و عدم تعادل بار بر روی کنترل‌کننده‌ها ارائه دادند. در [25]، نویسندگان الگوریتم بهینه‌سازی نوآورانه‌ای به نام PHCPA معرفی کردند که از الگوریتم‌های بهینه‌سازی الهام‌گرفته از طبیعت و تقسیم‌بندی شبکه برای بهبود عملکرد شبکه استفاده می‌کرد. هدف آن‌ها بهینه‌سازی موقعیت چندین کنترلر در SDN بود؛ به‌طوری که، حداقل تاخیر بین سوئیچ‌ها و کنترلرها را کاهش دهد. با این حال، این الگوریتم نیاز به زمان CPU و فضای ذخیره‌سازی بیشتری نسبت به الگوریتم‌های جایگزین داشت. در [8]، پژوهشگران یک الگوریتم ژنتیک تکاملی را برای حل مسئله‌ی CPP معرفی کردند. آن‌ها در این تحقیق، تخصیص بهینه سوئیچ‌ها ، توازن بار و جلوگیری از خرابی‌ها را در نظر گرفته و در عین حال از اعمال تغییرات در نصب کنترل‌کننده‌ها و جریان ترافیک خودداری کردند. با این حال، پیاده‌سازی این راه‌حل ممکن است برای شبکه‌هایی با مقیاس بزرگ، منابع محاسباتی قابل توجهی را مطالبه کند.

در [26]، نویسندگان برای بهبود عملکرد و کارآیی انرژی در شبکه‌های ماهواره‌ای (STIN)، قرار دادن استراتژیک کنترل‌کننده‌های SDN را ارائه دادند و مزایای استفاده از SDN در STIN را مانند پوشش جهانی و اتصال پایدار برجسته ساختند. آن‌ها الگوریتم MSAP را معرفی کردند که در مقایسه با الگوریتم‌های دیگر در مورد تأخیر متوسط هنگام استقرار چندین کنترل‌کننده در STIN عملکرد بهتری داشت. در [27]، نویسندگان الگوریتم ژنتیک جدیدی به **نام** IGCPA را برای حل مسئله‌ی CPP معرفی کردند**.** آن‌ها بر اهمیت در نظر گرفتن مصرف انرژی در هنگام انتخاب مکان کنترل‌کننده تأکید کردند و مزایای این کار را توضیح دادند. با این حال، پیچیدگی محاسباتی مدل BIP ممکن است آن را برای شبکه‌های بسیار بزرگ مناسب نکند. در [28]، نویسندگان بررسی‌هایی در مورد قرار دادن پویای چندین کنترل‌کننده و توزیع منابع محاسباتی از طریق الگوریتم‌های فرابتکاری مورچه انجام دادند. این رویکرد به سیستم امکان تطبیق با شرایط متغیر را می‌دهد و عملکرد شبکه را با تسهیل انتقال داده سریع‌تر کرده و موجب کاهش تأخیر و بهبود کارایی کلی شبکه می‌شود. با این حال، پیاده سازی این سیستم ممکن است نیاز به منابع محاسباتی قابل توجه، افزایش هزینه و پیچیدگی داشته باشد و خطر شکست در کنترل‌کننده‌های متمرکز را به همراه داشته باشد.

مراجع	الگوریتم	کمبودها	منافع
[1]	جست‌وجوی ممنوعه پترو(PITS)	نادیده گرفتن تعادل بار، نادیده گرفتن هزینه	مدیریت شرایط پویای شبکه، کاهش تاخیر
[8]	جست‌وجوی تکاملی، الگوریتم ژنتیک	نیاز به منابع محاسباتی بیشتر	نگاشت سوئیچ به کنترل کننده ها، افزایش تغادل بار، کاهش تاخیر
[17]	کوله پشتی ۰-۱، مار گارتنر	نادیده گرفتن مقیاس پذیری، نیاز به منابع محاسباتی بیشتر	کاهش پیچیدگی زمانی، کاهش تاخیر
[18]	الگوریتم شبیه‌ساز حرارت	نادیده گرفتن دسترس پذیری	بهبود تعادل بار، کاهش تاخیر
[19]	الگوریتم ژنتیک رتبه بندی نامغلوب	در نظر گرفتن شبکه به صورت استاتیک	بهبود قابلیت اطمینان و تاخیر در شبکه
[20]	ازدحام ذرات	عدم مطرح نحوه گسسته سازی الگوریتم	افزایش تعادل بار و کاهش تاخیر در شبکه
[21]	گرگ خاکستری	عدم در نظر گرفتن تعادل بار و هزینه	افزایش قابلیت اطمینان

[23]	تئوری بازی	در نظر گرفتن شبکه به صورت استاتیک	افزایش تعادل بار و کاهش تاخیر
[24]	الگوریتم ژنتیک	پیچیدگی بالا	افزایش تعادل بار و کاهش تاخیر
[25]	الگوریتم سفره ماهی، بهینه ساز عنکبوت	نیاز به زمان محاسباتی بالا	کاهش تاخیر
[27]	الگوریتم ژنتیک	مناسب نبودن برای شبکه‌های بسیار بزرگ.	کاهش مصرف انرژی
[26]	شبیه ساز حرارت بهبود یافته	عدم در نظر گرفتن تعادل بار و هزینه	کاهش مصرف انرژی
[28]	شبیه ساز حرارت	نیاز به زمان محاسباتی بالا	افزایش کارایی شبکه و افزایش تعادل بار

جدول۱: مروری بر کارهای انجام شده در حوزه SDN

۳- شرح مسئله

در این قسمت توضیح مختصری از روش‌های استفاده شده در این مقاله آورده شده است.

۳-۱- روش K-means بهبودیافته

الگوریتم K-means یک روش خوشه‌بندی محبوب است که هدف آن بیشینه‌سازی شباهت درون خوشه‌ها و افزایش اختلاف میان آن‌ها می‌باشد. این روش با اینکه در زیرمجموعه‌ی روش‌های یادگیری بدون نظارت قرار دارد، اما لازم است که تعداد خوشه‌ها از قبل مشخص شوند. الگوریتم K-means بهبودیافته، نسخه‌ای اصلاح‌شده از الگوریتم K-means، است که نیاز به پیش‌فرض‌ها و انتخاب پارامتر ندارد و می‌تواند به طور خودکار تعداد بهینه‌ی خوشه‌ها را نیز تعیین کند [11]. این الگوریتم از مفهوم آنتروپی استفاده می‌کند. در این هنگام $X = \{x_1, \dots, x_n\}$ مجموعه داده‌هایی است که باید خوشه‌بندی شوند و $A = \{a_1, \dots, a_n\}$ مراکز خوشه می‌باشند. فرض کنید $Z = [z_{ik}]_{n\times c}$

که در آن z_{ik} یک متغیر دودویی، $z_{ik} \in \{0,1\}$، است. همان‌طور که در رابطه (۱) نشان داده شده است اگر $K = 1 \dots c$ باشد، z_{ik} تعیین می‌کند که آیا داده‌ی x_i به خوشه k ام تعلق دارد یا نه.

$$z_{ik} = \begin{cases} 1 & if \|x_i - a_k\|^2 - \gamma \ln \alpha_k = min\|x^i - a_k\|^2_{1\le k\le c} - \gamma \ln \alpha_k \\ 0 & otherwise \end{cases} \quad (۱)$$

الگوریتم K-means بهبود یافته از مفهوم آنتروپی استفاده می کند تا الگوریتم خوشه بندی K-means را بدون نیاز به مقداردهی اولیه استفاده کرده و به‌طور خودکار تعداد خوشه ها را تعیین کند. همانطور که در رابطه (۲) نشان داده شده است، $J_{UKM}(Z, A, \alpha)$ تابع هدف را تعیین می‌کند.

$$J_{UKM}(Z, A, \alpha) = \sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik}\|x_i - a_k\|^2 - \beta n \sum_{k=1}^{c} a_k \ln a_k - \gamma \sum_{i=1}^{n}\sum_{k=1}^{c} Z_{ik} \ln a_k \quad (۲)$$

در اینجا $\|x_i - a_k\|$ فاصله اقلیدسی بین نقطه داده x_i و مرکز خوشه a_k است. مقدار a_k احتمال تعلق یک نقطه داده به خوشه k ام است. این احتمال بر اساس رابطه (۳) به‌روزرسانی می‌شود.

$$\alpha_k^{t+1} = \sum_{i=1}^{n} z_{ik}/n + (\beta/\gamma)\alpha_k^t \left[\ln \alpha_k^t - \sum_{s=1}^{c} \alpha_s^{(t)} \ln \alpha_s^{(t)}\right] \quad (۳)$$

مقدار γ_t به عنوان $e^{-c^t/250}$ تعریف شده است، و β بر اساس رابطه (۴) به‌دست می‌آید.

$$\beta_t = min\left[\frac{\sum_{k=1}^{c} \exp(-n\left|\alpha_k^{(t+1)} - \alpha_k^{(t)}\right|)}{c}, \frac{1 - \max_{1\le k\le c}(\frac{1}{n}\sum_{i=1}^{n} z_{ik})}{(-\max_{1\le k\le c} \alpha_k^{(t)} \sum_{k=1}^{c} \ln \alpha_k^{(t)}}\right] \quad (۴)$$

تا زمانی‌که مقدار c به یک مقدار پایدار برسد، مقادیر α، β و γ در هر تکرار به‌طور پیوسته به‌روز می‌شوند.

۳-۲- الگوریتم بهینه‌سازی گله اسب

الگوریتم بهینه‌سازی گله اسب (HOA) یک الگوریتم فراابتکاری است که برای حل مسائل بهینه‌سازی با تعداد زیادی متغیر طراحی شده است. این الگوریتم الهام خود را از رفتار گله‌های اسب گرفته و رفتارهای اجتماعی آن‌ها در مراحل مختلف زندگی را تقلید می‌کند. این رفتار شامل چرا، سلسله مراتب، اجتماعیت، تقلید، مکانیزم دفاع و گردشگری است [15]. رویکرد الگوریتم شامل حرکت اسب‌ها در هر تکرار است که توسط رابطه‌ی (۵) تعیین می‌شود.

$$x_m^{Iter.Age} = \vec{V}_m^{Iter.Age} + X_m^{(Iter-1).Age} \qquad Age = \alpha, \beta, \gamma, \delta \qquad (۵)$$

در HOA، موقعیت هر اسب به عنوان $x_m^{Iter.Age}$ نمایش داده می‌شود که Age مربوط به بازه‌ی سنی اسب است و $Iter$ نشان‌دهنده‌ی تکرار کنونی است. بردار سرعت یک اسب به عنوان $v_m^{Iter.Age}$ نشان داده می‌شود. طول عمر اسب‌ها در حدود ۳۰-۲۵ است و گروه‌های سنی مختلف، رفتارهای متمایزی را نشان می‌دهند. اسب‌ها در بازه‌ی سنی ۰ تا ۵ سال به عنوان δ، در بازه‌ی سنی ۵ تا ۱۰ سال به عنوان γ، ۱۰ تا ۱۵ سال به عنوان β و بیش از ۱۵ سال با عنوان α دسته‌بندی می‌شوند. برای تعیین سن اسب‌ها، یک ماتریس جمعیت موجود تولید و مرتب می‌شود. ۱۰٪ برتر به عنوان اسب‌های α، ٪۲۰ بعدی به عنوان اسب‌های β، ٪۳۰ به عنوان اسب‌های γ و بقیه ۴۰٪ به عنوان اسب‌های δ مشخص می‌شوند. از روش‌های ریاضی برای شبیه‌سازی رفتار اسب‌ها و محاسبه بردارهای سرعت آن‌ها استفاده می‌شود. حرکت اسب‌ها در هر چرخه الگوریتم با استفاده از رابطه (۶) که الگوهای رفتار مذکور را در نظر می‌گیرد، توصیف می‌شود.

$$\begin{aligned} \vec{V}_m^{Iter.\alpha} &= \vec{G}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\beta} &= \vec{G}_m^{Iter.\alpha} + \vec{H}_m^{Iter.\alpha} + \vec{S}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\gamma} &= \vec{G}_m^{Iter.\alpha} + \vec{H}_m^{Iter.\alpha} + \vec{S}_m^{Iter.\alpha} + \vec{I}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} + \vec{R}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\delta} &= \vec{G}_m^{Iter.\alpha} + \vec{I}_m^{Iter.\alpha} + \vec{R}_m^{Iter.\alpha} \end{aligned} \qquad (۶)$$

۳-۳- استراتژی تضاد نخبگان

استراتژی تضاد نخبگان[1]، یک رویکرد نوآورانه در یادگیری ماشین است که الهام خود را از استراتژی مبتنی بر مخالف[2] می‌گیرد [29]. این مفهوم روشی مبتنی بر نخبه‌گرایی و جهت مخالف آن در همان زمان است. در این روش به منظور افزایش احتمال کشف مناطقی که پتانسیل بیش‌تری برای بهینه بودن دارند، ابتدا مناطق و حدس‌های بهینه از بین

[1] Elite Opposition-Based Learning (EOBL)
[2] Opposition-Based Learning (OBL)

جمعیت اولیه شناخته‌شده و سپس برای اکتشاف بیش‌تر و عبور از بهینه‌های محلی نقاط کاملا مخالف آن‌ها نیز انتخاب می‌شوند. در این روش می‌توانیم به سمت جهت مخالف هدایت شویم که ما را به سمت نزدیک‌تر شدن به راه‌حل بهینه ناشناخته هدایت می‌کند. لازم به ذکر است که نقطه مخالف از طریق انعکاس نقطه محاسبه‌شده از طریق نقطه مرکزی به‌دست می‌آید. پس اگر x یک عدد حقیقی باشد که در بازه‌ی $x \in [a,b]$ قرار دارد، عکس آن، $\breve{x}$ توسط رابطه (۷) تعریف می‌شود.

$$\breve{x} = a + b - x \tag{۷}$$

رابطه (۷) در بعدهای بالاتر به‌صورت رابطه‌ی (۲) تعریف می‌شود. فرض کنید $x(x_1,..,x_D)$ یک نقطه در فضای D بعدی باشد و متغیر x به‌صورت $x_i \in [a_i,b_i]$, $i = 1,2,3,\dots,D$ تعریف شود، معکوس x توسط $\breve{x}$ $(\breve{x}_1,\dots x_D)$ به صورت رابطه‌ی (۸) تعریف می‌شود.

$$\breve{x} = a_i + b_i - x_i \tag{۸}$$

۳-۴- فرموله‌سازی مسئله

تمرکز اصلی راه‌حل‌های پیشنهادشده برای مسئله جای‌گذاری کنترل‌کننده‌ها، در تعداد کنترل‌کننده‌ها و محل قرارگیری آن‌ها است [16]. در این کار شبکه نرم‌افزارمحور به صورت یک گراف $G(S,E)$ نشان داده می‌شود، به‌طوری که S مجموعه‌ی گره‌ها و E یال‌های بین این گره‌ها در یک شبکه هستند. یال uv یک ارتباط دوطرفه بین گره‌های u و v است. هر یال دارای وزنی است که نمایانگر پهنای باند ارتباط است. مجموعه‌ی $C = \{c_1, c_2, \dots, c_k\}$ کنترل‌کننده‌های شبکه را نشان می‌دهد. در شبکه‌های نرم‌افزارمحور، چندین معیار معمولاً برای ارزیابی کارایی سیستم استفاده می‌شود. در این مقاله، برخی از این معیارها به شرح زیر تعریف شده‌اند.

۳-۴-۱- متوسط تاخیر انتشار

نحوه‌ی محاسبه‌ی متوسط تأخیر انتشار در شبکه‌ی نرم افزارمحور، توسط رابطه (۹) تعریف می‌شود [30]. این رابطه از مجموع میانگین زمانی انتشار اطلاعات بین کنترل‌کننده‌ها و میانگین زمانی انتشار اطلاعات از سوئیچ‌ها به کنترل‌کننده‌ها به‌دست می‌آید. میانگین زمان انتشار اطلاعات بین کنترل‌کننده‌ها با Avg_{ICL} و میانگین زمان انتشار اطلاعات از سوئیچ‌ها به کنترل‌کننده‌ها با Avg_{SCL} نشان داده می‌شود، که در ادامه هر یک توضیح داده شده است.

کوتاه‌ترین مسیر بین سوئیچ s، $s \in S$ و کنترل‌کننده‌ی c، $c \in C$ توسط $d(s,c)$ نمایش داده می‌شود و کوتاه‌ترین مسیر بین کنترل‌کننده‌ها ی c_i و c_j به صورت $d(c_i,c_j)$ نمایش داده می‌شود. کوتاه‌ترین مسیر بین دو گره i و j از طریق فاصله‌ی اقلیدسی بر طبق رابطه‌ی (۱۰) به‌دست می‌آید [31].

$$Del_{ave}\ (\mathrm{C}) = S2C_{Avg} + C2C_{Avg} \quad (۹)$$

$$d(i.j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \quad (۱۰)$$

الف) تأخیر انتشار بین کنترل‌کننده‌ها

تاخیر بین کنترل‌کننده‌ها اهمیت بسیاری دارد زیرا ارتباط بین کنترل‌کننده‌ها برای رسیدن به همگامی بین اجزای شبکه ضروری است. تاخیر بین کنترل‌کننده‌ها می‌تواند به عملکرد شبکه تأثیر بگذارد، زیرا اگر جداول مسیریابی یا اطلاعات کنترلی به موقع به‌روزرسانی نشوند، کنترل‌کننده‌ها ممکن است تصمیمات نادرستی بگیرند. در صورت وجود تاخیر بین کنترل‌کننده‌ها، که توسط دورترین کنترل‌کننده محدود شده است، ممکن است به عملکرد شبکه آسیب وارد شود [32].

متوسط تأخیر بین کنترل‌کننده‌ها در شبکه نرم‌افزارمحور را می‌توان با استفاده از رابطه‌ی (۱۱) به‌دست آورد [33] .

$$C2C_{Avg} = \frac{1}{|C|.(|C|-1)} \sum_{i.j \in C} d(i.j)\ x_{ij} \quad (۱۱)$$

ب) تأخیرانتشار بین سوئیچ تا کنترل‌کننده

معادله (۱۲) میانگین تأخیر سوئیچ به کنترل‌کننده را در سیستم‌های شبکه نرم افزارمحور نشان می‌دهد که در آن $d\ (i,j)$ کوتاه‌ترین مسیر از سوئیچ j تا کنترل‌کننده i است که با فاصله اقلیدسی محاسبه می‌شود [33].

$$S2C_{Avg} = \frac{1}{|S|} \sum_{j \in V} \sum_{i \in C} d(i.j)\ x_{i.j} \quad (۱۲)$$

۳-۴-۲- عدم تعادل بار

هر کنترل‌کننده در شبکه مسئول مدیریت مجموعه‌ای از سوئیچ‌ها است. اما وقتی که سوئیچ‌ها به نزدیک‌ترین کنترل‌کننده اختصاص داده می‌شوند، توزیع بار بر روی کنترل‌کننده‌ها نامتوازن می‌شود. به این معنی که برخی از کنترل‌کننده‌ها به شدت دچار بار کاری زیاد بوده، در حالی که از برخی دیگر به‌طور کامل استفاده نمی‌شود. به عبارت دیگر، عدم تعادل بار بین دو کنترل‌کننده توسط تفاوت در تعداد سوئیچ های اختصاص داده شده به هر کنترل‌کننده تعیین می‌شود. عدم تعادل بار بروی یک کنترل‌کننده با تعداد عناصری که به آن ارسال می‌شوند تعیین می‌شود که با نماد FE نشان داده شده است. در این زمینه، عدم تعادل بار یک کنترل‌کننده با استفاده از رابطه (۱۳) محاسبه می‌شود که s تعداد گره‌هایی است که قادر به ارتباط با کنترل‌کننده هستند [34].

$$AvgLI = \max(maxn^{s}_{|FE|} - minn^{s}_{|FE|}) \qquad (۱۳)$$

۳-۴-۳- مصرف انرژی

در این مقاله، مصرف انرژی شبکه، طبق رابطه (۱۴) تعریف شده است. در این رابطه، E_{base} مصرف انرژی کنترل‌کننده‌ی c_i را وقتی که روشن است، نشان می‌دهد. Ed_{ci} مصرف انرژی پویای کنترل‌کننده c_i است که بر اساس بار کاری کنترل‌کننده در هر لحظه تغییر می‌کند و N_Sتعداد سوییچ‌های تحت کنترل c_i را نشان می‌دهد. e_2 یک مقدار تعریف‌شده توسط کاربر است. EL_{sc}مجموع مصرف انرژی بین کنترل‌کننده و سوییچ‌های تحت کنترل آن را نمایش می‌دهد. در حالی که bw_{cs} پهنای باند یال بین کنترل‌کننده c_i و سوییچ s_i است.

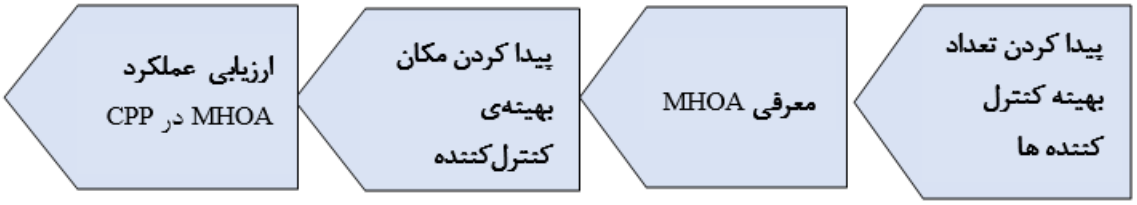
$$EC = \sum_{i=1}^{n}\left(Eb_{c_i} + Ed_{c_i}) + \sum_{c=1}^{c=cl} EL_{sc}\right)_{sc}$$
$$EL_{sc} = \frac{d(c,s)}{bw_{cs}} \times e_1 \qquad (۱۴)$$
$$Ed_{c_i} = N_S \times e_2$$

۴- بیان مسئله

هدف این مقاله حل مسئله CPP با پیدا کردن تعداد و مکان بهینه‌ی کنترل‌کننده‌ها و در عین حال بهبود عملکرد شبکه و دستیابی به اهدافی مانند کاهش تأخیر انتها به انتها، کاهش مصرف انرژی و بهینه‌سازی تعادل بار است. مسئله‌ی

CPP به عنوان یک مسئله‌ی NP-hard معرفی شده است. روش‌های فراابتکاری یک گزینه مناسب برای حل مسائل پیچیده و NP-hard مانند CPP هستند که با بررسی بخش کوچکی از فضای جست‌وجو می‌توانند راه‌حل نزدیک به بهینه را پیدا کنند. با این حال، این روش‌ها ممکن است پیچیده و زمان‌بر باشند. از طرف دیگر در مورد مسائل خوشه‌بندی، K-means و روش‌های چندمعیاری مشابه، اغلب به دلیل سرعت، سادگی و کارآمدی می‌تواند گزینه مناسبی باشند. به همین دلیل در این مقاله در فاز اول برای یافتن تعداد کنترل‌کننده‌ها و خوشه‌بندی از نسخه‌ی بهبودیافته‌ی K-mean استفاده شده است که می‌تواند با پیدا کردن تعداد مناسب خوشه‌ها بر اساس فاصله اقلیدسی گره‌ها، عمل خوشه‌بندی را انجام دهد. در این مقاله با تغییر تابع هدف در این الگوریتم علاوه بر فاصله اقلیدسی گره‌ها، ترافیک شبکه نیز در نظر گرفته می‌شود. در فاز دوم مقاله، الگوریتمی بر پایه‌ی ژنتیک به نام الگوریتم گله اسب بهبودیافته (MHOA) تعریف می‌شود و از آن برای پیدا کردن مکان کنترل‌کننده در هر خوشه استفاده می‌شود. MHOA ورژن بهبودیافته‌ی الگوریتم گله اسب (HOA) است. HOA یکی از الگوریتم‌های اخیرا پیشنهاد شده است که در حل بسیاری از مسائل دنیای واقعی به خوبی عمل می‌کند. با این حال، در برخی مسائل پیچیده ممکن است که در بهینه‌ی محلی گیر کند. MHOA تا حدی زیاد می‌تواند این مشکلات ر ا حل کند.

استفاده از یک جمعیت اولیه تصادفی در HOA منجر به بی‌ثباتی در عملکرد آن می شود. اگر بتوانیم جمعیت اولیه را به گونه‌ای انتخاب کنیم که تنوع بیشتری داشته باشند مشکل بهینه محلی تا حد زیادی می‌تواند حل شود و سرعت همگرایی الگوریتم و توانایی آن برای رسیدن به راه‌حل جهانی بهینه را افزایش می‌یابد. استفاده از استراتژی تضاد نخبگان می تواند به عنوان نقطه شروع مفیدی برای فرآیند جست‌وجو عمل کند و به‌طور بالقوه عملکرد کلی الگوریتم را افزایش دهد. در ابتدا، الگوریتم یک جمعیت اولیه را انتخاب می کند و برازندگی هر راه‌حل را ارزیابی می کند. سپس استراتژی EOLB برای انتخاب N جمعیت اعمال می شود به‌طوری که بهترین جمعیت انتخاب شده و برای اکتشاف بالاتر دقیقا نقطه‌ی مقابل این افراد نیز انتخاب می‌شود. این کار باعث بالا رفتن قدرت اکتشاف الگوریتم می‌گردد. MHOA از استراتژی EOBL برای اکتشاف بیشتر استفاده می‌کند. استفاده از عملگرهای ژنتیک قدرت اکتشاف و بهره‌برداری الگوریتم را بالا می‌برد و باعث همگرایی بهتر و سرعت بیشتر الگوریتم می‌شود. در فاز بعدی مقاله از الگوریتم MHOA برای پیدا کردن مکان کنترل‌کننده در داخل هر خوشه استفاده می‌شود. هر گره در داخل خوشه می‌تواند به عنوان کنترل‌کننده در نظر گرفته می‌شود. در این سناریو فرض شده است که هر سوئیچ فقط می‌تواند به یک کنترل‌کننده متصل شود و دو کنترل‌کننده با مختصات یکسان در شبکه وجود ندارند. در ادامه، جزئیات بیشتری در مورد نحوه انجام کار ارائه می‌شود. شکل ۲ ساختار این مقاله را نشان می‌دهد.



شکل ۲: ساختار کلی مقاله

۴-۱- پیدا کردن تعداد کنترل‌کننده‌ها با استفاده ازK-means بهبودیافته

هر چه‌قدر تعداد کنترل‌کننده‌ها زیاد باشد شبکه در حالت ایده‌آل‌تری قرار خواهد گرفت. با این حال، استفاده از کنترل‌کننده هزینه‌بر بوده و از طرف دیگر بیشتر از یک تعداد مشخص عملا تاثیر چندانی بر کارایی شبکه نخواهد داشت. تعداد کم کنترل‌کننده‌ها نیز می‌تواند به‌طور منفی بر کیفیت شبکه تأثیر بگذارد و منجر به ازدحام در شبکه شود. بنابراین تعیین تعداد بهینه‌ی کنترل‌کننده‌ها در شبکه حیاتی است. در این مقاله برای تعیین تعداد کنترل‌کننده‌ها از الگوریتم K-means بهبود یافته و مفهوم خوشه‌بندی استفاده می‌کنیم. دراین نسخه از K-means نیازی به وارد کردن تعداد خوشه‌ها به عنوان ورودی نیست و تعداد خوشه‌ها نیز در حین اجرای الگوریتم تعیین می‌شود. در هر خوشه یک کنترل‌کننده برای کنترل سوئیچ‌های خوشه قرار می‌گیرد. تعداد کنترل‌کننده‌ها برابر با تعداد خوشه‌ها است. در این فاز علاوه بر تعیین تعداد کنترل‌کننده‌ها، سوئیچ‌های تحت کنترل یک کنترل‌کننده نیز تعیین می‌شوند. بهتر است که سوئیچ‌های نزدیک به هم و سوئیچ‌هایی که بیشتر با هم در ارتباط هستند تحت نظارت یک کنترل‌کننده باشند تا مجبور به بارگذاری اطلاعات شبکه بر روی کنترل‌کننده‌های متفاوت نباشیم. بنابراین سوئیچ‌های نزدیک به هم را در یک خوشه قرار می‌دهیم تا تحت نظارت یک کنترل‌کننده واحد باشند. از طرف دیگر اگر تعداد سوئیچ‌های یک خوشه و میزان پیام‌های ارسالی در یک شبکه زیاد باشد باعث ازدحام در آن کنترل‌کننده خواهد شد. بنابراین زمانی که ترافیک در شبکه زیاد باشد باید تعداد خوشه‌ها افزایش یابد تا ترافیک بین کنترل‌کننده‌ها تقسیم شود. از طرف دیگر، اگر بار شبکه کاهش یابد، توصیه می شود تعداد کنترل‌کننده‌ها کاهش یابد تا از هزینه‌های غیرضروری جلوگیری شود. در ابتدا حداکثر تعداد کنترل‌کننده‌های موجود به تعداد سوئیچ‌ها، به عنوان ورودی به الگوریتم K-means بهبودیافته داده می‌شود. تعداد خوشه‌ها تعیین‌کننده‌ی تعداد کنترل‌کننده‌هاست. الگوریتم اصلی به گونه‌ای تعریف شده است که فقط از فاصله اقلیدسی برای خوشه‌بندی استفاده می‌کند. ما در این مقاله علاوه بر فاصله اقلیدسی، ترافیک شبکه را نیز در نظر گرفته‌ایم. در اینجا، برای خوشه‌بندی سوئیچ‌ها از فاصله اقلیدسی آن‌ها و ترافیک شبکه استفاده می‌کنیم. احتمال اختصاص یک سوئیچ

به خوشه‌ی k_ام توسط α_{ik} نمایش داده می‌شود. برای هر سوئیچ i، یک آرایه به نام $strscl$ در نظر گرفته می‌شود که طول آن برابر با تعداد کنترل‌کننده‌هاست، $|strscl| = k$. هر سلول از $strscl$ احتمال نگاشت سوئیچ i به یک کنترل‌کننده k را نشان می‌دهد. ابتدا، مقدار α_{ik} به $1/c$ تنظیم می‌شود که c تعداد کل کنترل‌کننده‌هاست که در ابتدا به تعداد سوئیچ‌ها در نظر گرفته شده است. زمانی که α_{ik} برابر با $1/c$ است، یعنی احتمال تخصیص آن سوئیچ به هر یک از کنترل‌کننده‌ها برابر است که نشان‌دهنده‌ی عدم وجود اطلاعات قابل دسترس در مورد α_{ik} است. در هر تکرار، α_{ik} بر اساس معادله (۳) محاسبه می‌شود. برای در نظر گرفتن ترافیک هر خوشه علاوه بر فاصله‌ی اقلیدسی هر سوئیچ، رابطه‌ی ۱ به صورت رابطه (۱۵) تعریف می‌شود. به طوریکه $load_{clusi}$ میزان بار در خوشه‌ی i و $AvgLi$ نشان دهنده متوسط بار شبکه می‌باشد.

$$z_{ik} = \begin{cases} 1 & if \|x_i - a_k\|^2 - \gamma \ln \alpha_k = min\|x^i - a_k\|^2_{1\le k\le c} - \gamma \ln \alpha_k \ \ amd \ \ load_{clusi} < AvgLi \\ 0 & otherwise \end{cases} \qquad (۱۵)$$

مقادیر α_{ik} بر اساس بار کنترل‌کننده به‌روزرسانی می‌شوند. حداکثر باری که یک کنترل‌کننده می‌تواند مدیریت کند، مربوط به ویژگی فیزیکی آن کنترل‌کننده است که از قبل مشخص است. زمانی که بار یک کنترل‌کننده به حداکثر نزدیک می‌شود، کنترل‌کننده در وضعیت بحرانی قرار می‌گیرد و کارایی آن کاهش می‌یابد. در چنین مواردی، توصیه می‌شود تا کنترل‌کننده‌های بیشتری به شبکه اضافه شوند تا بار متعادل شود. بنابراین، احتمال انتخاب این کنترل‌کننده برای همه گره‌ها کاهش می‌یابد در حالی که احتمال کنترل‌کننده‌های دیگر بر روی گره‌ها با مقدار مشابه افزایش می‌یابد. اگر بار بر روی یک کنترل‌کننده خیلی کم باشد، بهتر است که آن را غیرفعال شود. در این صورت، سوئیچ‌های متصل به این کنترل‌کننده باید مجدداً به یک کنترل‌کننده دیگر اختصاص یابند. بر اساس این اطلاعات، برخی از کنترل‌کننده‌ها روشن شده و دیگری خاموش می‌شوند. پس از به‌روزرسانی بردارهای احتمال، سوئیچ‌ها از روش چرخ رولت برای انتخاب کنترل‌کننده مناسب استفاده می‌کنند. در این مرحله، تعداد کنترل‌کننده‌ها تعیین می‌شود و سوئیچ‌های تحت کنترل هر کنترل‌کننده تعیین می‌شوند.

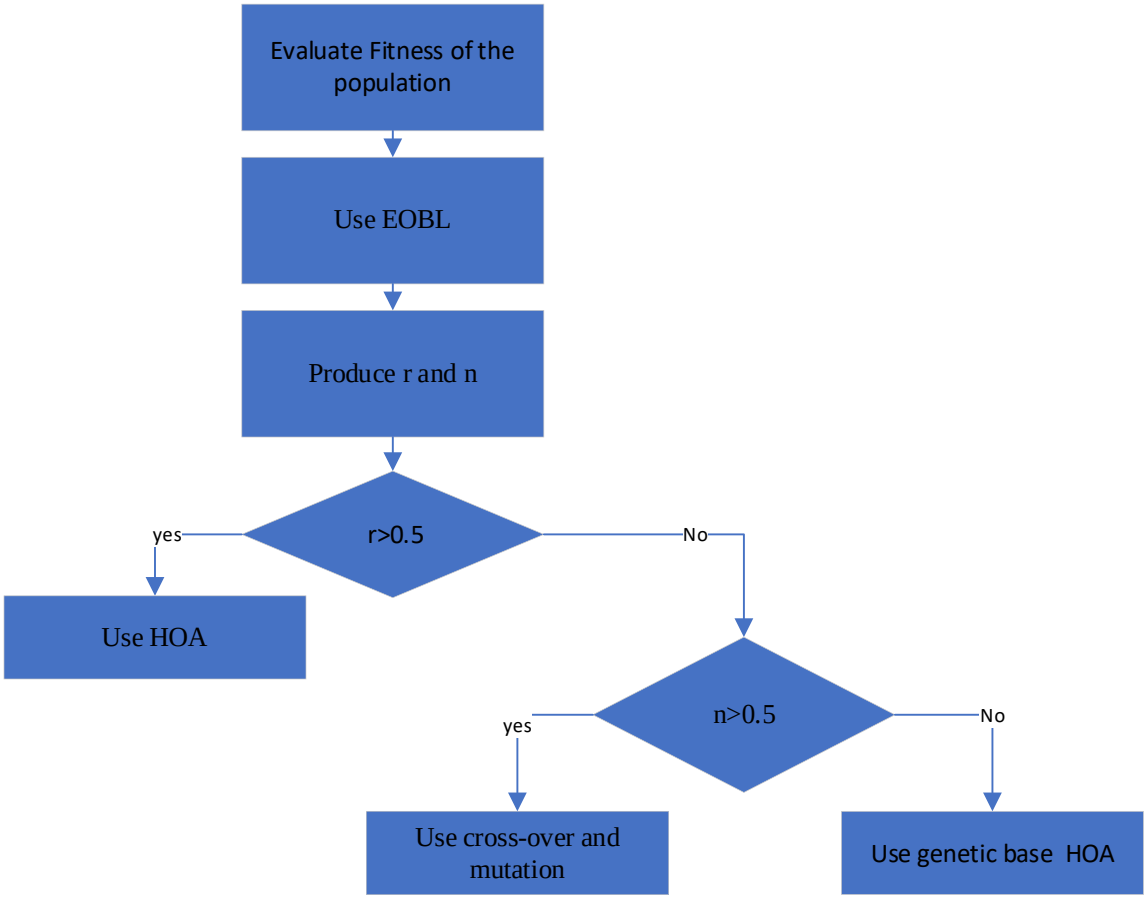
۴-۲- تعیین مکان کنترل‌کننده‌ها

در این مرحله باید مکان بهینه‌ی کنترل‌کننده در هر خوشه تعیین شود. برای این کار تعداد کنترل‌کننده‌ها و مختصات سوئیچ‌های هر خوشه که در فاز اول تعیین شد به عنوان ورودی به این فاز انتقال یافته و مورد استفاده قرار می‌گیرند. خروجی مختصات مکانی کنترل‌کننده‌ها می‌باشد. مکان هر یک از سوئیچ‌ها می‌تواند به عنوان مکان یک کنترل‌کننده باشد. از آن‌جایی‌که یافتن مکان بهینه کنترل‌کننده یک مسئله ان-پی سخت است ما برای حل آن از یک الگوریتم فراابتکاری به نام MHOA استفاده می‌کنیم که ورژن بهبودیافته‌ی الگوریتم گله اسب می‌باشد. شکل ۳ نحوه عمل MHOA ر ا نشان می‌دهد. ابتدا بعد از تولید جمعیت اولیه با استفاده از تابع ارزیابی مقدار برازندگی هر فرد جمعیت برآورد می‌شود. سپس براساس استراتژی تضاد نخبگان بهترین جمعیت انتخاب شده و برای اکتشاف بالاتر دقیقا نقطه‌ی مقابل این افراد نیز انتخاب می‌شود. این کار باعث بالا رفتن قدرت اکتشاف الگوریتم می‌گردد. MHOA، از عملگرهای ژنتیک برای اکتشاف و بهره‌برداری بیشتر و بهبود همگرایی و سرعت الگوریتم استفاده می‌کند. در MHOA ابتدا دو عدد تصادفی n و r بین ۰ و ۱ تولید می‌شوند. چون الگوریتم HOA، خود الگوریتمی قدرتمند می‌باشد، برای برخورداری از مزایای آن زمانی که r بزرگتر از ۰,۵ از HOA استفاده می‌شود. اگر r کوچک‌تر از ۰,۵ و n بزرگ‌تر از ۰,۵ باشد از عملگرهای ساده‌ی ژنتیک شامل عملگر برش[1] و جهش[2] استفاده می‌شود. اگر r کوچک‌تر از ۰,۵ و n کوچک‌تر از ۰,۵ باشد از HOA بهبودیافته استفاده می‌شود که در ادامه کامل شرح داده شده است.

در استفاده از عملگرهای ژنتیک باید در نظر داشت که در HOA اسب‌های جوان، تاکید بیشتری بر روی اکتشاف وجود دارد، در حالی که اسب‌های مسن تمایل به بهره‌برداری بیشتر دارند. بنابراین، برای اسب‌های جوان‌تر بیش‌تر از عملگرهای ژنتیکی که جست‌وجوی سراسری را ترویج می‌کنند، مانند جهش و ترکیب استفاده می‌شود. برای اسب‌های مسن بیشتر از عملگرهایی که تغییرات وابسته به والدین دارند و برای جست‌وجوی محلی مناسب‌تر هستند، استفاده می‌شود. عملگر کپی[3] شامل کپی کردن یک قسمت از یک راه‌حل و الصاق آن در یک راه‌حل دیگر است. تعداد سلول‌های کپی شده می‌تواند درصد مشخصی از کل تعداد کنترل‌کننده‌ها باشد. عملگر جهش[4]، یک بخش تصادفی از راه‌حل را تغییر می‌دهد. تعداد سلول‌های تغییر داده شده توسط ضرب تعداد کنترل‌کننده‌ها در یک عدد تصادفی بین ۰ و ۱ تعیین می‌شود. عملگر برش تک‌نقطه‌ای[5]، یک نقطه‌برش و دو والد را انتخاب می‌کند. اطلاعات در یکی از طرفین نقطه برش

[1] crossover
[2] mutation
[3] Duplicate
[4] Mutation
[5] One-point crossover

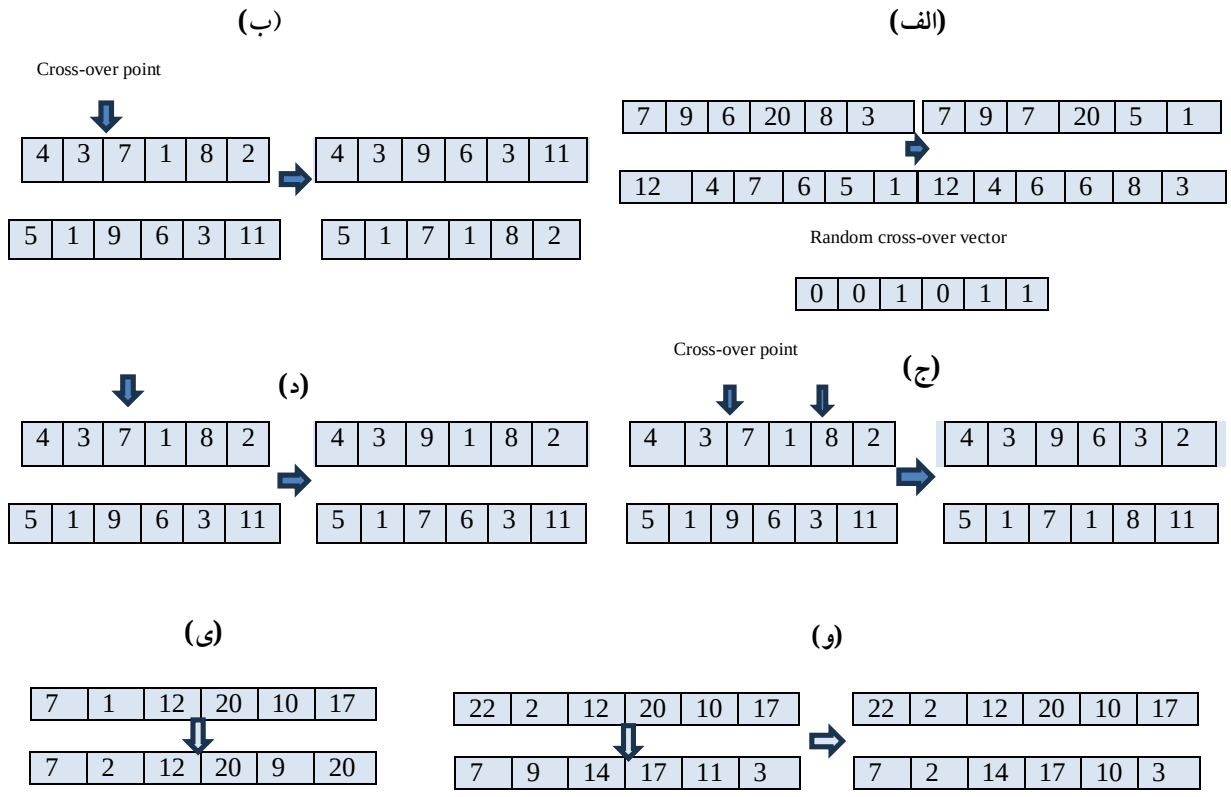
بین والدین معاوضه می‌شود و دو فرزند جدید تولید می‌شوند. در عملگر برش دونقطه‌ای[1] دو نقطه‌ی برش و دو والد انتخاب می‌شوند و اطلاعات بین این نقاط در دو والد معاوضه می‌شوند.



شکل ۳: نحوه عملکرد الگوریتم **MHOA**

در عملگر برش یکنواخت[2] یک بردار برش شامل مقادیر دودویی به صورت تصادفی تولید می‌شود که تعیین می‌کند کدام سلول‌ها بین والدین جابه‌جا شوند. در عملگر تعویض[3]، دو والد انتخاب شده و ژن‌های آن به صورت تصادفی معاوضه می‌شوند. در این عملگر، تعداد ژن‌های معاوضه شده درصدی از کل تعداد کنترل‌کننده‌ها است[35-37].

[1] Two-point crossover
[2] Uniform cross-over
[3] Swap



شکل ۴: نحوه عملکرد عملگرهای ژنتیک. (الف) برش یکنواخت. (ب) برش تک نقطه‌ای. (ج) برش دونقطه‌ای. (د) تعویض. (و) عملگر جهش. (ی)عملگر کپی

در ادامه رفتارهای اسب‌ها با استفاده از عملگرهای ژنتیک در MHOA به روزرسانی شده‌اند. مقدار r یک عدد تصادفی بین ۰ و ۱ می‌باشد. برای اسب‌هایی که بیش از ۱۵ سال هستند، نحوه‌ی رفتار اسب‌ها بر اساس رابطه‌ی (۱۶) تعیین می‌شود. اسب‌های مسن‌تر بیشتر رفتار چرا و دفاع را از خود نشان می‌دهند.

$$\vec{V}_m^{Iter.\alpha} = single\ cross\ over\ (\vec{G}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha}) \qquad (۱۶)$$

اسب‌هایی که ۱۰ تا ۱۵ سال سن دارند، بردار سرعت خود را با استفاده از رابطه (۱۷) تعیین می‌کنند. بر اساس این رابطه، اگر مقدار r بیشتر از ۰/۷۵ باشد، از عملگر برش تک‌نقطه‌ای استفاده می‌شود. اگر مقدار r بین ۰/۲۵ و ۰/۵ باشد، از یک برش یکنواخت و اگر مقدار r کمتر از ۰/۵ باشد، از عملگر برش دونقطه‌ای استفاده می‌شود. اسب‌ها در گروه β رفتارهای چرا، تقلید سلسله‌مراتبی، مکانیزم دفاع و اجتماعیت را نشان می‌دهند.

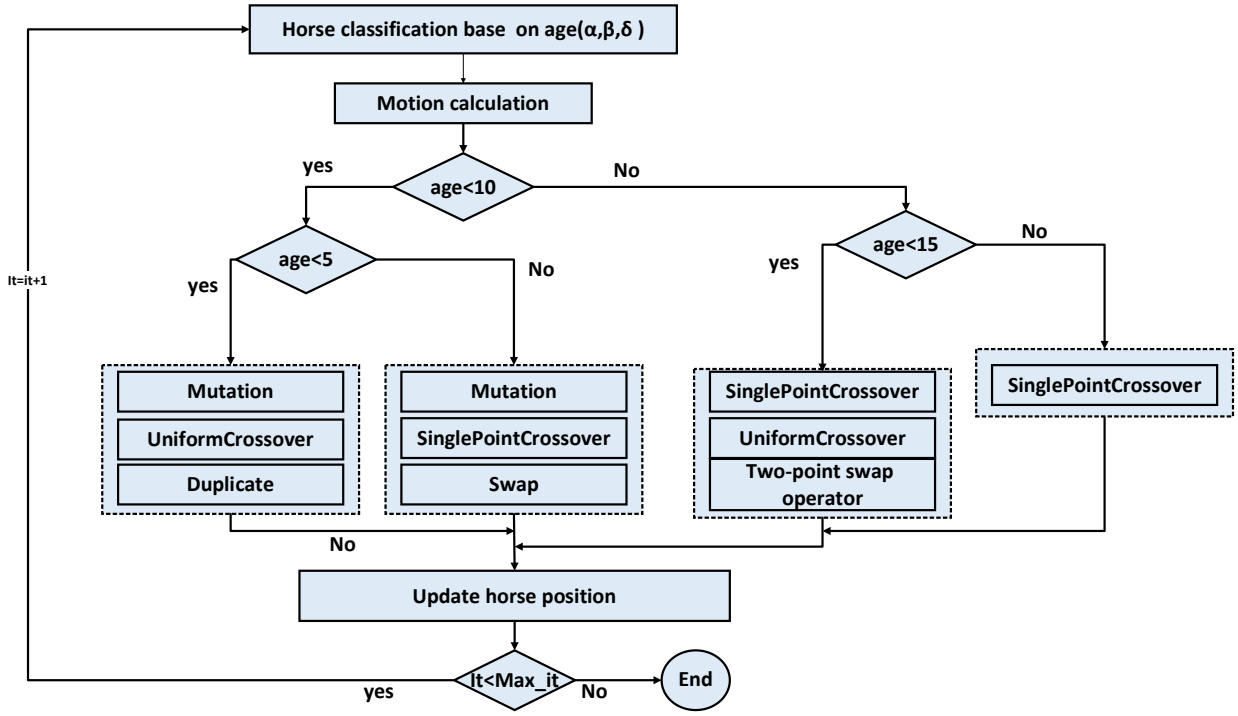
$$\vec{V}_m^{Iter.\beta} = \begin{cases} single\ cross\ over\ (\vec{G}_m^{Iter.\beta} + \vec{D}_m^{Iter.\beta}) \\ ,r \geq 0.75 \\ uniform\ crossover\left(\vec{H}_m^{Iter.\beta} + \vec{S}_m^{Iter.\beta}\right) \\ ,0.5 \leq r < 0.75 \\ two - point\ crossover\left(\vec{D}_m^{Iter.\beta},\ \vec{H}_m^{Iter.\beta}\right) \\ ,r < 0.5 \end{cases} \quad (۱۷)$$

اسب‌های گروه گاما رفتارهای مختلفی مانند چرا، تقلید سلسله‌مراتبی، مکانیزم دفاع و اجتماعیت را نشان می‌دهند. با توجه به رفتار آن‌ها می‌توان فهمید که این گروه اسب‌ها هم تمایل به اکتشاف و هم بهره‌برداری دارند. رابطه (۱۸) بردار سرعت را برای این اسب‌ها تعیین می‌کند. اگر مقدار r بیشتر از ۰٫۷۵ باشد، عملیات جهش روی بردار رفتار تقلید انجام می‌شود. اگر مقدار r بین ۰/۵ و ۰/۷۵ باشد، عملیات جابه‌جایی روی بردارهای سلسله‌مراتبی و اجتماعیت انجام می‌شود. به‌طور مشابه، اگر مقدار r بین ۰/۲۵ و ۰/۵ باشد، یک برش دونقطه‌ای و جهش بر روی بردارهای مکانیزم دفاع و رفتارهای چرا انجام خواهد شد.

$$\vec{V}_m^{Iter.\gamma} = \begin{cases} Mutation\ (\vec{I}_m^{Iter.\gamma}) \\ ,r \geq 0.75 \\ Swap\ (\vec{G}_m^{Iter.\alpha}, \vec{R}_m^{Iter.\gamma}) \\ ,0.5 \leq r < 0.75 \\ Swap(\vec{S}_m^{Iter.\alpha},\ \vec{H}_m^{Iter.\gamma}) \\ 0.25 <\ r < 0.5 \\ Mutation(one - pointCrossover\ (\vec{R}_m^{Iter.\gamma}, \vec{D}_m^{Iter.\gamma})\ ) \\ r < 0.25 \end{cases} \quad (۱۸)$$

برای اسب‌های جوان زیر ۵ سال در گروه δ، به صورت رابطه (۱۹) پیش می‌رویم. اسب های گروه δ تمایل بیشتری به اکتشاف دارند، به همین دلیل از عملگرهایی استفاده می‌کنیم که می‌توانند امکان کاوش را در این اسب ها افزایش دهند. فلوچارت نحوه‌ی بهبود HOA با عملگرهای ژنتیک در شکل ۲ نشان داده شده است.

$$\vec{V}_m^{Iter.\delta} = \begin{cases} Mutation(\vec{G}_m^{Iter.\delta}) & ,r \geq 0.75 \\ niform\ crossover\ (\vec{R}_m^{Iter.\delta} + \vec{I}_m^{Iter.\delta}) & ,0.5 \leq r < 0.75 \\ Duplicate(\vec{G}_m^{Iter.\delta},\ \vec{R}_m^{Iter.\delta}) & 0.25 <\ r < 0.5 \end{cases} \quad (۱۹)$$



شکل ۵: نحوه بهبود الگوریتم HOA با عملگرهای ژنتیک

۴-۲-۱- تابع ارزیابی

هدف اصلی این مقاله شناسایی تعداد بهینه کنترل‌کننده‌های شبکه نرم‌افزار محور و موثرترین تخصیص سوئیچ‌های توزیع‌شده به این کنترل‌کننده‌ها است. این کار برای کاهش تاخیر انتها به انتها، به حداقل رساندن عدم تعادل شبکه و کاهش مصرف انرژی انجام می‌شود. در این زمینه، تابع تناسب با رابطه‌های (۲۰) و (۲۱) تعریف می‌شود. در اینجا، تابع f یک تابع غیر خطی از تعداد کنترل‌کننده‌های مستقر N است. $AvgD_{e2e}$ نشان‌دهنده‌ی متوسط تأخیر سرتاسری، WD_{e2e} نشان‌دهنده‌ی بدترین تأخیر سرتاسری بوده و همچنین میانگین درجه عدم تعادل بار با $AvgLi$ و بدترین درجه عدم تعادل بار Wli در نظر گرفته شده‌اند. علاوه بر این، میانگین مصرف انرژی به صورت $AvgEc$ و بدترین مصرف انرژی به صورت WEc در شبکه تعریف شده‌اند. این مقادیر بر اساس روابط ارائه شده در بخش ۳ محاسبه می شوند.

$$Min\ f(N, AvgD_{e2e}, AvgLi, AvgEc) \qquad (۲۰)$$

$$f(N, AvgD_{e2e}, AvgLi, AvgEc) = \frac{AvgD_{e2e}}{WD_{e2e}} + \frac{AvgLi}{Wli} + \frac{AvgEc}{WEc} \quad (۲۱)$$

۴-۲-۲- پیچیدگی محاسباتی

برای ارزیابی کارایی یک الگوریتم هنگام حل مسائل بهینه‌سازی، محاسبه پیچیدگی محاسباتی بسیار مفید است. پیچیدگی محاسباتی کل MHOA می‌تواند به صورت رابطه‌ی (۲۱) بیان شود. که n تعداد جمعیت d، بعد مسئله و T حداکثر تعداد تکرار را نشان می‌دهند. چون نصف جمعیت توسط الگوریتم اسب اجرا می‌شوند، پیچیدگی الگوریتم در این قسمت به صورت $O\left(T \times \frac{n}{2} \times d\right)$ تعریف می شود. از طرف دیگر $O\left(\mathrm{T} \times \frac{n}{4} \times d\right)$ جمعیت فقط از الگوریتم ژنتیک و $O\left(T \times \frac{n}{4} \times d\right)$ از الگوریتم اسب بر پایه‌ی ژنتیک استفاده می‌کنند. رابطه (۲۲) پیچیدگی محاسباتی MHOA را نشان می‌دهد.

$$\begin{aligned} O(MHOA) &= O\,(problem\ definition) + O(initialization) \\ &+ O\,(function\ evaluation) + O\,(Updating\ MHOA) \\ &= O\left(1 + n + Tn + T\frac{n}{2}\,d + T\frac{n}{4}\,d + T\frac{n}{4}\,d\right) \cong O(Tnd + Tn + n) \end{aligned} \quad (۲۲)$$

۵- تنظیمات شبیه‌سازی

برای شبیه‌سازی این کار از یک سیستم دارای یک پردازنده Intel Core i7 با سرعت ۲/۸۰ گیگاهرتز، ۱۶ گیگابایت رم و سیستم عامل ویندوز ده ۶۴ بیتی استفاده است. الگوریتم‌ها در متلب ۲۰۱۶ پیاده‌سازی شده و به صورت موازی ۴۰ بار اجرا شده‌اند. برای ارزیابی نتایج استفاده‌شده از دیتاست واقعی توپولوژی زو [38] استفاده شده است. برای ارزیابی مقیاس‌پذیری الگوریتم پیشنهادی از سه شبکه‌ی Arnet، Bics و Colt که از لحاظ اندازه و توپولوژی کاملا متفاوت هستند، استفاده شده است. Aarnet، یک شبکه کوچک است که در لایه IP با ۱۹ گره و ۲۴ یال استرالیا را تحت پوشش خود قرار داده است. Bics شبکه‌ای متوسط در سطح اروپا بوده و دارای ۴۶ گره و ۸۵ یال می‌باشد. شبکه‌ی Colt، یک شبکه بزرگ است که در سراسر اروپا گسترده شده است و شامل ۱۴۹ گره و ۱۹۱ پیوند در سطح IP است. نتایج نشان می‌دهند که MHOA در هر دو شبکه نسبت به الگوریتم‌های دیگر بهتر عمل کرده است. پارامترهای شبیه‌سازی در جدول ۲ نشان داده شده است، و جدول ۳ پارامترهای استفاده شده در MHOA را نشان می‌دهد. در این تحقیق برای پیاده سازی شبکه‌ی پویا، ترافیک شبکه با بسته‌های با سایز متفاوت ایجاد شده است و برای

هر بسته در ابتدای کار مشخص شده است که چه تایمی وارد شبکه خواهد شد. در هر راند بررسی می‌شود که کدام بسته‌ها باید در شبکه باشند و بر اساس تعداد و اندازه‌ی بسته‌ها ترافیک تعیین می‌شود.

پارامترها	مقادیر
سایز بسته	۱۰۰ تا ۲۰۰ بایت
بیشینه تعدادکنترل‌کننده‌ها	۱۲
بیشینه تکرار	۴۰
سایز جمعیت	۵۰
ترافیک	تصادفی(۰-۲۵۰)

جدول۲:پارامترهای شبیه‌سازی

به ترتیب ۰/۹ و ۰/۵	فاکتور سلسله مراتب	$h\beta\ , h\gamma$
به ترتیب ۰/۹ و ۰/۵	فاکتور اجتماعیت	$s\beta\ , s\gamma$
۰/۳	فاکتور تقلید	$i\gamma$
به ترتیب ۰/۵ و ۰/۲ و ۰/۱	فاکتور دفاع	$d\alpha\ , d\beta\ , d\gamma$
به ترتیب ۰/۰۵ و ۰/۱	فاکتور گشت و گذار	$r\beta\ , r\gamma$
۰/۹۵	کمترین حد فضای چرا	L
۱/۰۵	بیشترین حد فضای چرا	U
۱۰٪ اسب‌ها	تعداد اسب‌های با بهترین مکان	P
۲۰٪ اسب‌ها	تعداد اسب های با بدترین مکان	Q

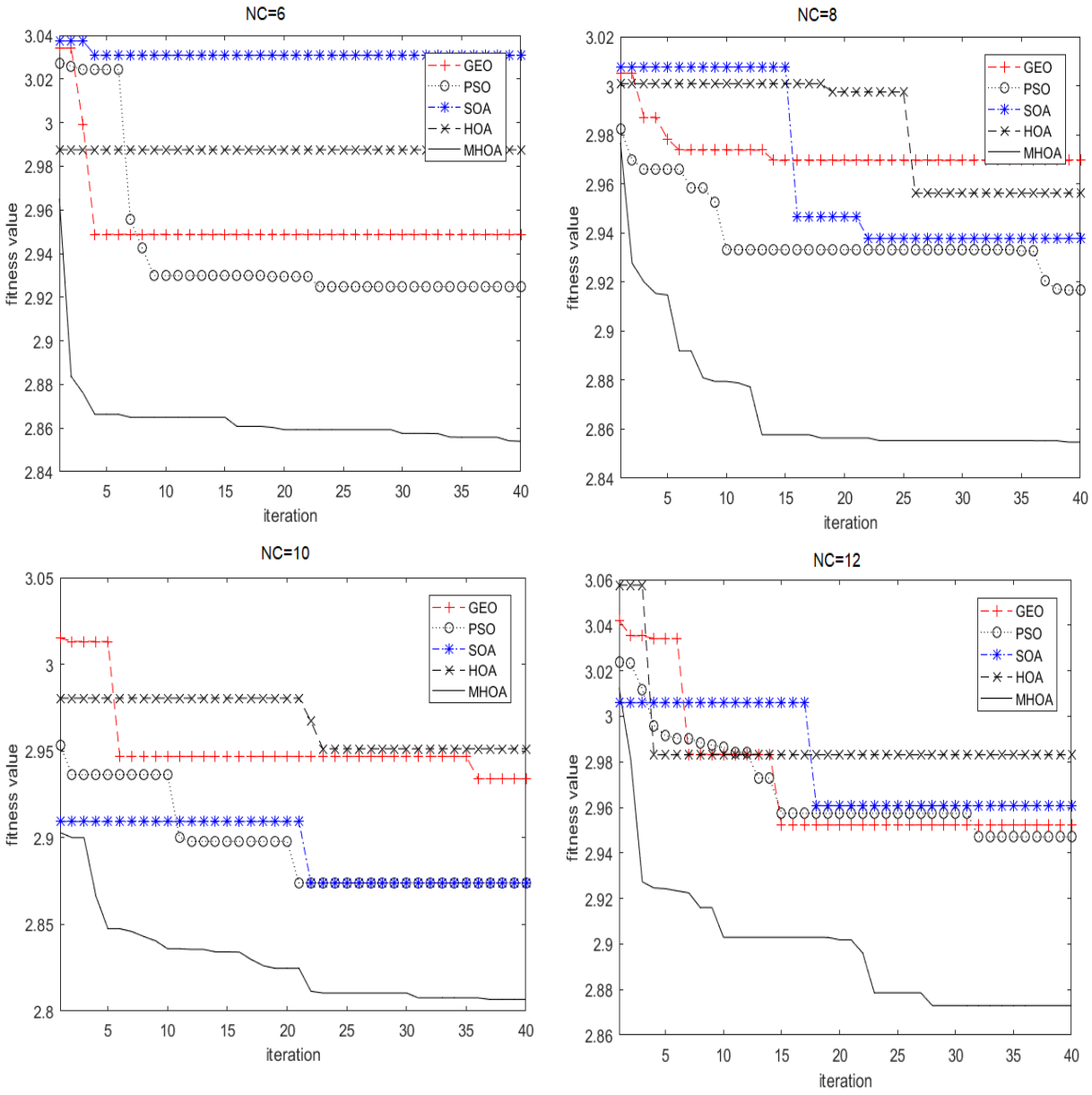
جدول۳: پارامترهای استفاده شده در MHOA

در این بخش، ابتدا کارایی الگوریتم MHOA از لحاظ میزان همگرایی بررسی می‌شود. نتایج نشان می‌دهد که الگوریتم MHOA نسبت به دیگر الگوریتم‌های فراابتکاری بهتر عمل می‌کند. در مرحله‌ی بعد MHOA برای حل مسئله‌ی CPP استفاده می‌شود و باعث کاهش عدم تعادل بار تا میزان ۹,۶۶٪، تاخیر انتها به انتها تا میزان ۱۹,۶۵٪ و میانگین مصرف انرژی تا میزان ۸,۴۳٪ می‌شود.

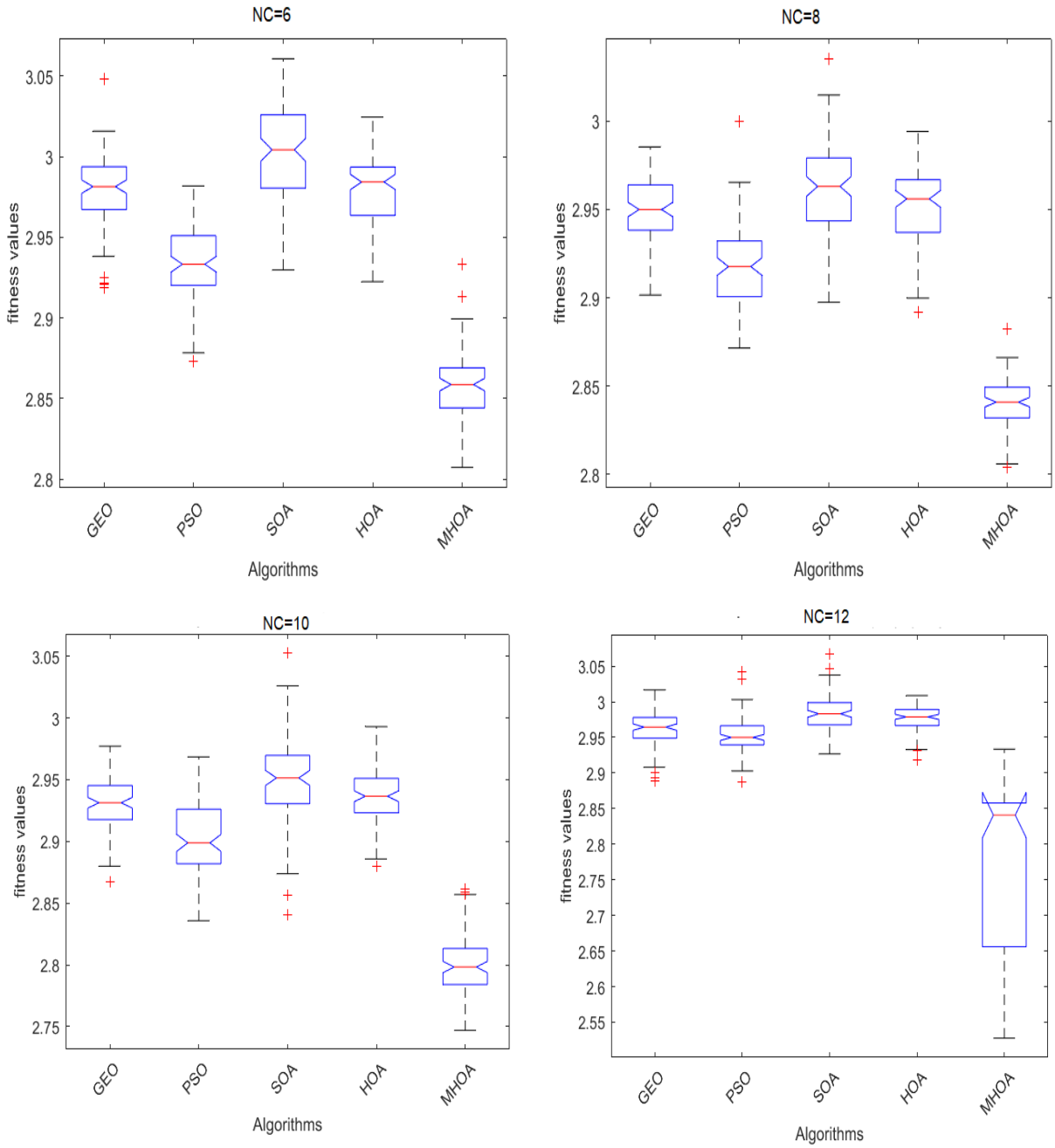
۵-۱- نتایج آنالیزها

این بخش شامل یک مجموعه جامع از نتایج تجربی است که به ارزیابی عملکرد الگوریتم پیشنهادی می‌پردازد. نتایج هر آزمایش به صورت جداگانه در ادامه ارائه می‌شوند. در این مرحله، توانایی MHOA در بهبود همگرایی با چهار الگوریتم فراابتکاری مطرح شامل الگوریتم بهینه سازی عقاب طلایی[1](GEO) [39]، الگوریتم ازدحام ذرات[2](PSO) [40]، الگوریتم مرغ دریایی[3](SOA) [41] و HOA مقایسه شده‌اند. در این کار برای ارزیابی کارایی الگوریتم پیشنهادی از نمودارهای همگرایی و ANOVA استفاده شده است. آزمون ANOVA برای نشان دادن میزان انحراف معیار در خروجی‌های به کار می‌رود. در این نمودار پراکندگی کم در نتایج نشان‌دهنده عملکرد و پایداری بهتر الگوریتم است. از نمودار همگرایی برای تجزیه و تحلیل نرخ همگرایی الگوریتم در به‌دست آوردن نتایج استفاده می‌شود. برای سنجش کارایی الگوریتم در حل CPP الگوریتم با چهار روش موفق، CCPGWO[23]، GEWO[35]، PHCPA[14] که در بخش ۲ شرح داده شده‌اند مقایسه می‌شود. برای تعیین کارآیی روش MHOA ارزیابی‌ها بر روی سه شبکه‌ی واقعی از دیتاست واقعی زو شامل Aarnet، Bics و Colt انجام شده است. شکل‌های ۶ و ۷ عملکرد MHOA را از نظر نرخ همگرایی و آزمون ANOVA در شبکه Aarnet نشان می‌دهند. نتایج نشان می‌دهند که MHOA نتایج بهتری نسبت به الگوریتم‌های دیگر دارد. آزمایش با تعداد کنترل‌کننده‌های مختلف، ۶، ۸، ۱۰ و ۱۲ انجام شده است. نمودارهای ANOVA نشان می‌دهند که MHOA به‌طور مداوم نتایج پایدار با کمترین انحراف معیار نسبت به سایر الگوریتم‌ها تولید می‌کند. علاوه بر این، MHOA از ابتدا با عملکردی قوی، بهترین همگرایی را نسبت به روش‌های دیگر دارد.

[1] Golden Eagle Optimization
[2] Partial swarm Optimization
[3] Seagull optimization algorithm



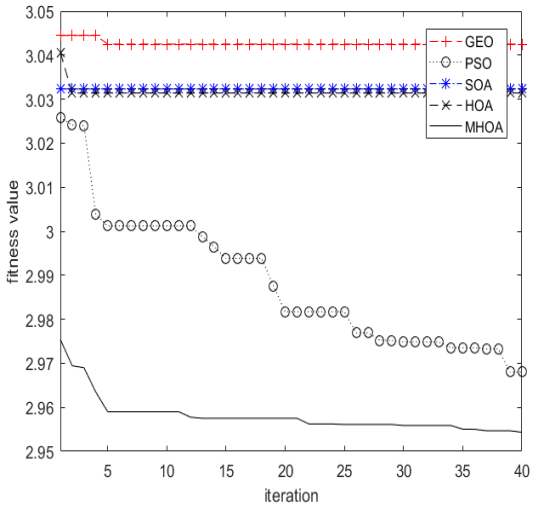
شکل ۶: نرخ همگرایی **MHOA** در شبکه‌ی **Aarnet** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**



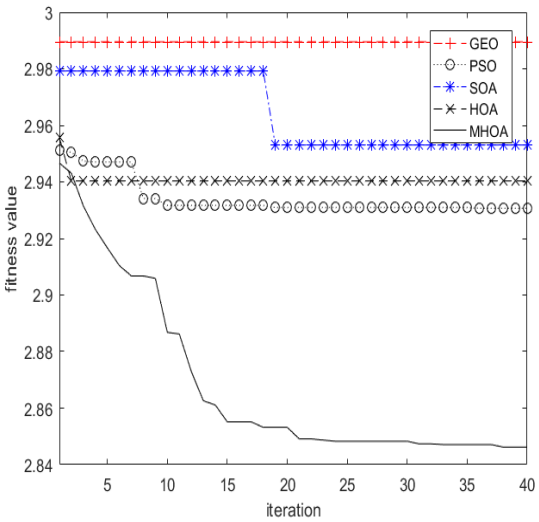
شکل ۷: تست **ANOVA** برای **MHOA** در شبکه‌ی **Aarnet** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**

تصاویر ۸ و ۹ عملکرد MHOA را از نظر نرخ همگرایی و نتایج آزمون ANOVA بر روی شبکه Bic نشان می‌دهند. یافته‌ها نشان می‌دهند که MHOA به‌طور مداوم از الگوریتم‌های دیگر پیشی گرفته است. آزمایش‌ها با تعداد کنترل‌کننده‌های متفاوت ۶، ۸، ۱۰ و ۱۲، انجام شده‌اند. نمودارهای Anova نشان می‌دهد که MHOA با کمترین انحراف معیار نسبت به الگوریتم‌های دیگر نتایج پایداری تولید می‌کند.

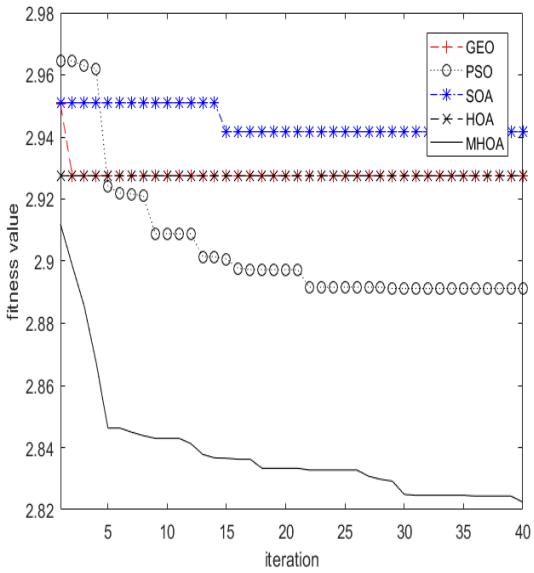
NC=6



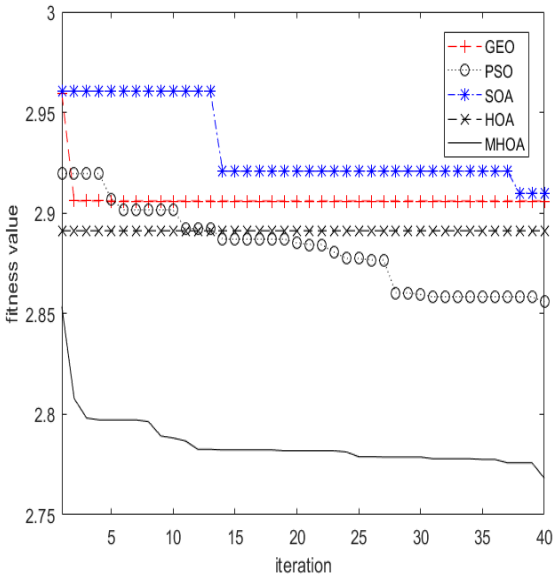
NC=8



NC=10



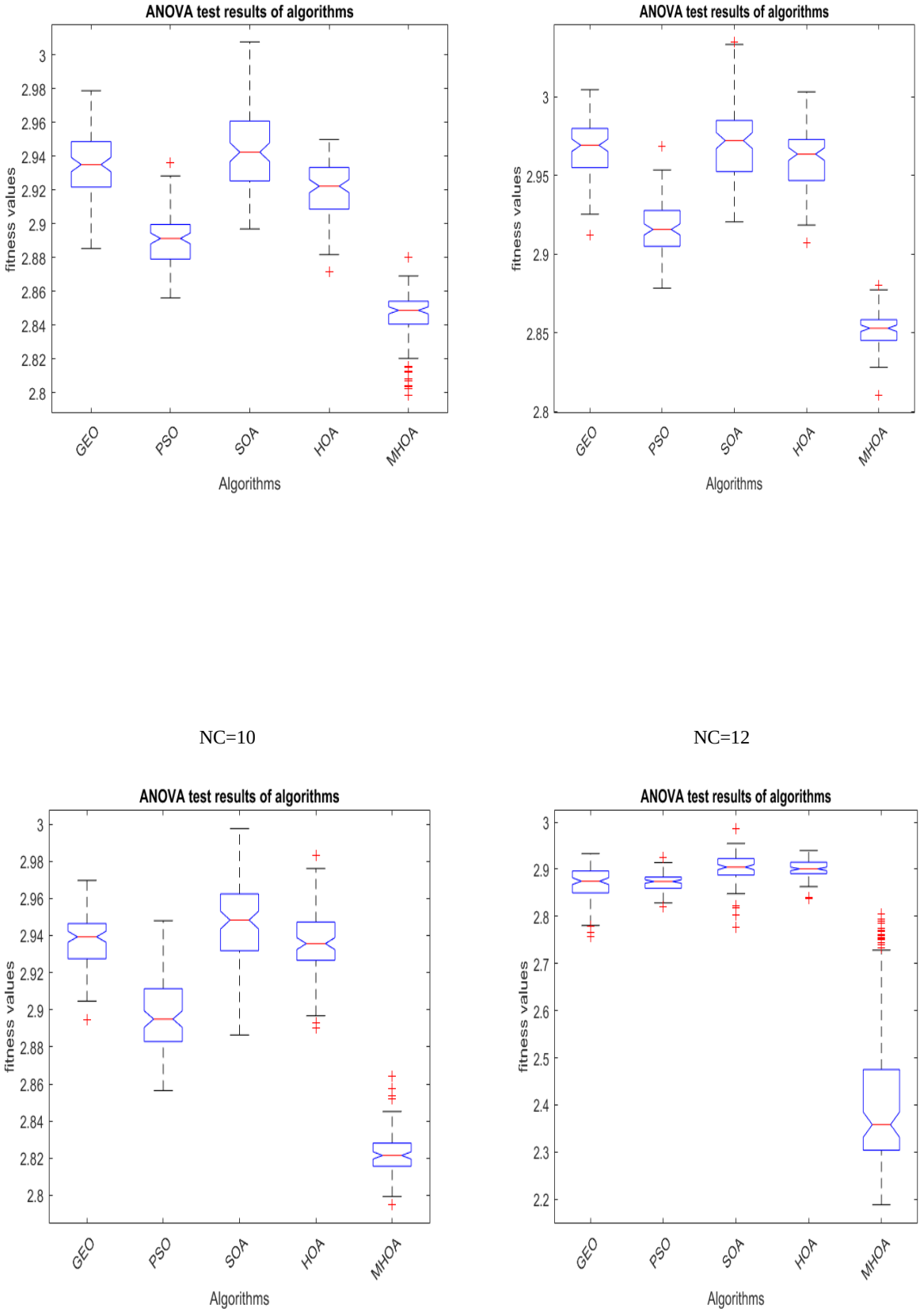
NC=12



شکل ۸: نمودار همگرایی **MHOA** در شبکه‌ی **BIC** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**

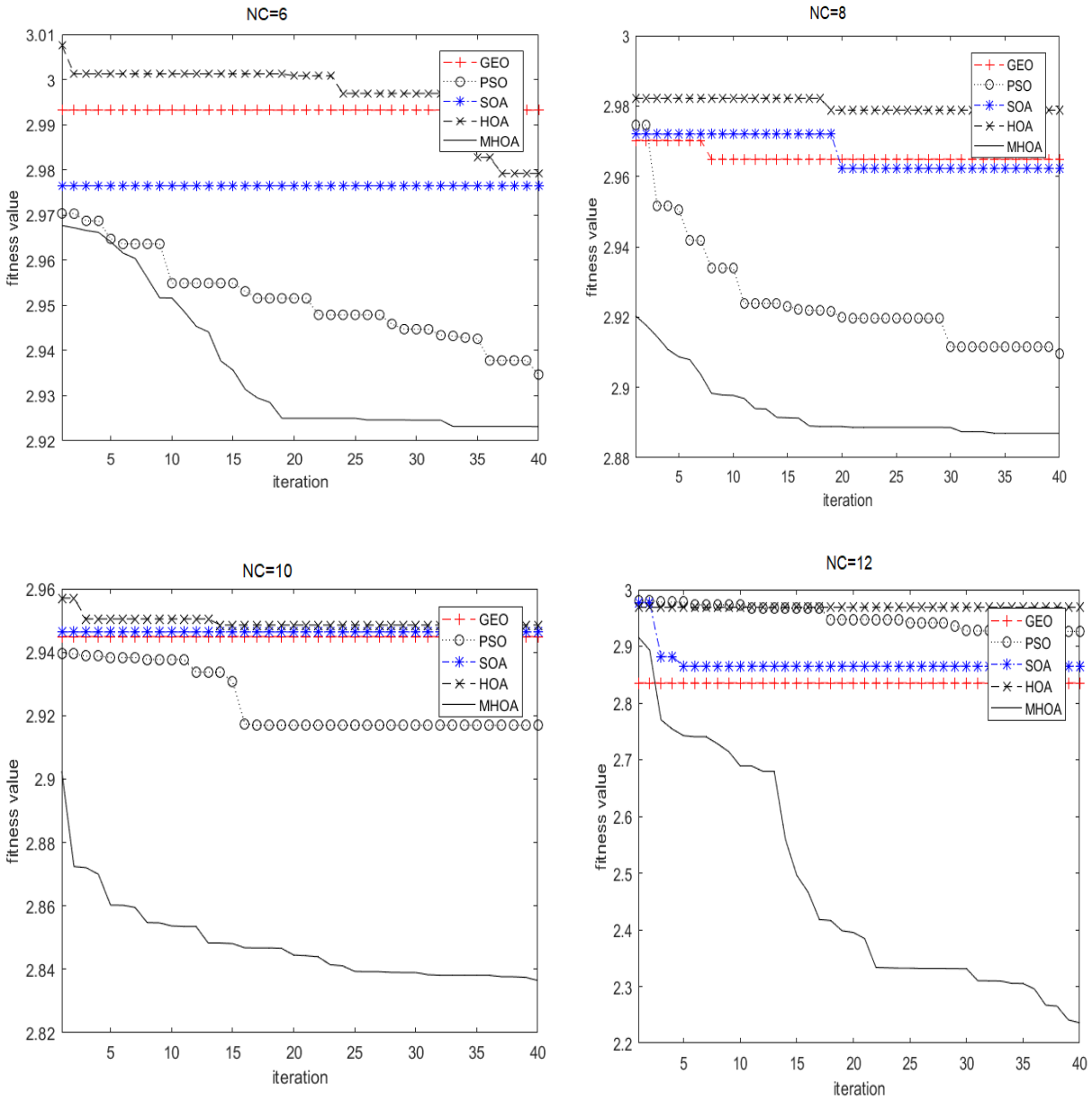
NC=6

NC=8

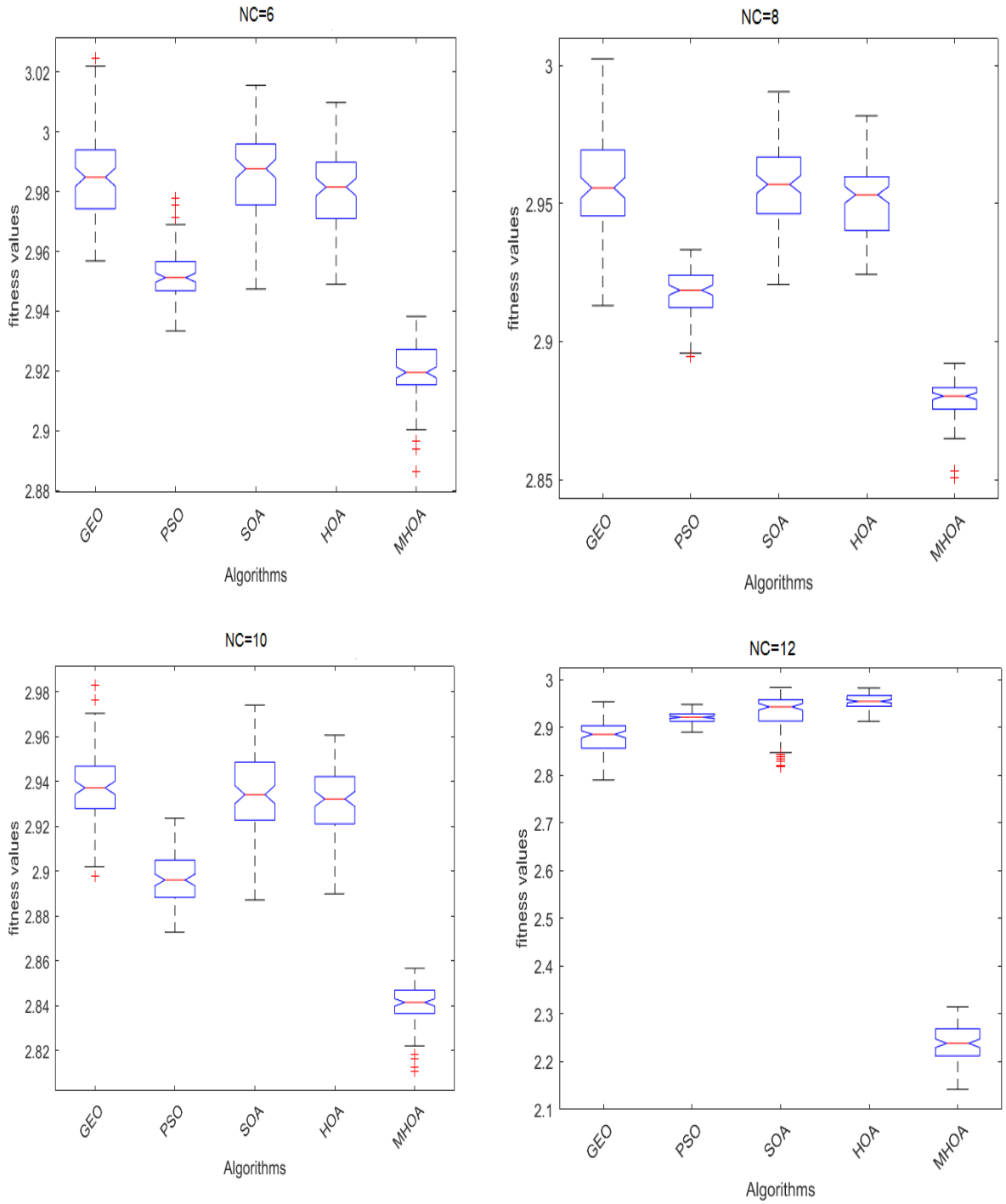


شکل ۹: تست **ANOVA** برای **MHOA** در شبکه‌ی **Bics** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**

تصویر ۱۰ نرخ همگرایی الگوریتم MHOA و شکل ۱۱ نتایج آزمون ANOVA برای الگوریتمی که بر روی شبکه Colt اعمال شده است را نشان می‌دهد. بر اساس آزمون ANOVA، MHOA نتایج پایداری با کمترین انحراف معیار نسبت به الگوریتم‌های دیگر تولید می‌کند. در این شبکه نیز، MHOA عملکردی قوی داشته و به نرخ همگرایی بهتری نسبت به روش‌های دیگر دست یافته است.



شکل ۱۰: نمودار همگرایی **MHOA** در شبکه‌ی **Colt** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**

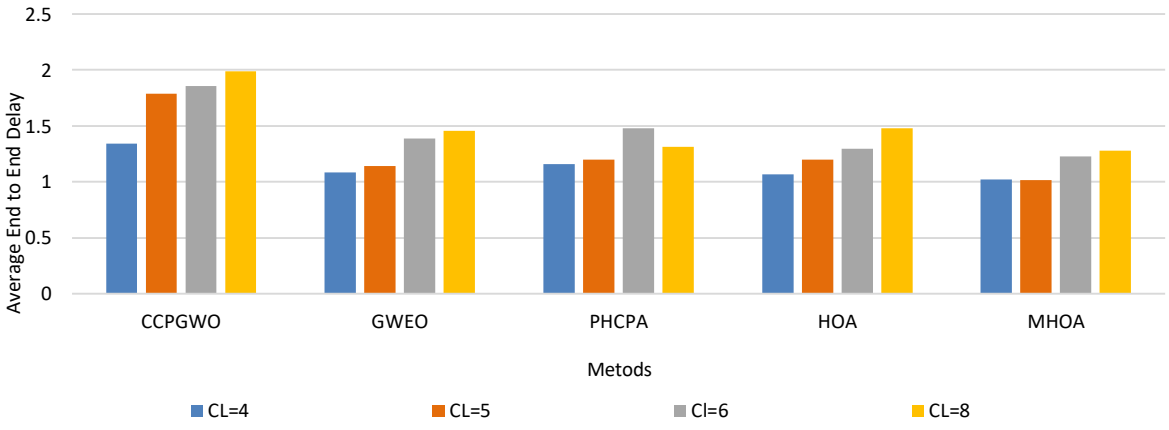


شکل ۱۱ : تست **ANOVA** برای **MHOA** در شبکه‌ی **Colt** با تعداد کنترل‌کننده‌های متفاوت **NC=6,8,10,12**

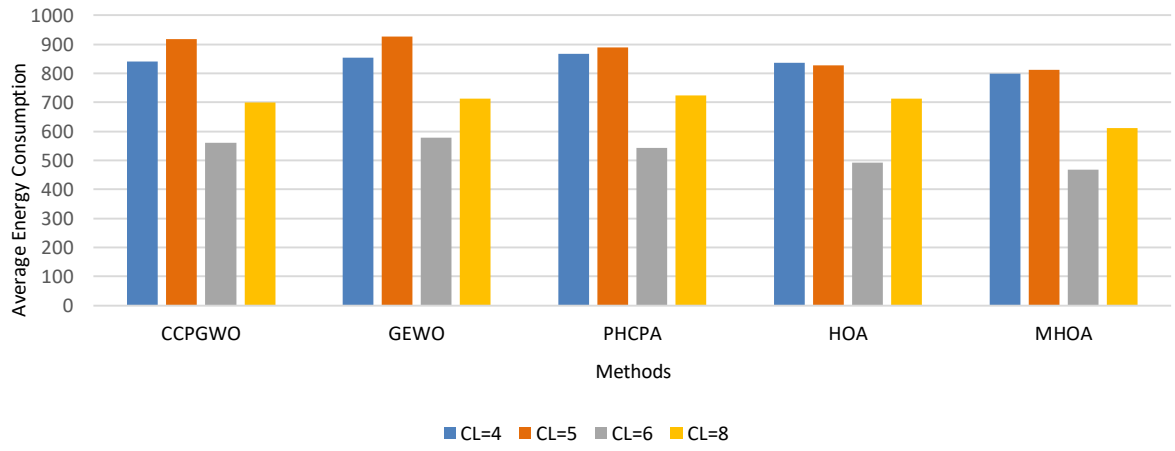
۵-۲- ارزیابی MHOA در پیدا کردن مکان بهینه کنترل‌کننده در داخل هر خوشه

در این قسمت کارایی MHOA در رابطه با بهبود تعادل شبکه، مصرف انرژی و تاخیر انتها به انتها ارزیابی می‌شود. یافته‌ها نشان می‌دهند که MHOA به‌طور موثر بار کاری را بین کنترل‌کننده‌ها توزیع می‌کند و ترافیک شبکه را کمینه کرده و تاخیر انتها به انتها و مصرف انرژی را کاهش می‌دهد. در این مرحله نیز، ارزیابی ها شامل تعداد مختلفی از کنترل‌کننده‌ها بوده و در سه شبکه‌ی SDN از توپولوژی زو با اندازه و مقیاس متفاوت، Aarnet، Bics و Colt، انجام شده‌اند. سعی شده است که بهبود MHOA را از نظر میانگین عدم تعادل بار، میانگین تاخیر انتها به انتها و میانگین مصرف انرژی از طریق نمودارهای میله‌ای و جداول آماری در مقایسه با CCPGWO[23]، GEWO[35]، PHCPA[14] و HOA نشان داده شود. با پیدا کردن تعداد بهینه کنترل‌کننده‌ها و انتساب سوئیچ ها به هر خوشه در فاز اول تعادل بار در شبکه افزایش می‌یابد. با پیدا کردن مکان بهینه کنترل‌کننده‌ها، MHOA نیاز به افزودن مداوم اطلاعات جدید به هر کنترل‌کننده را کاهش می‌دهد، که منجر به تاخیر کلی بهتر نسبت به الگوریتم‌های دیگر می‌شود. به‌طور کلی انرژی مصرفی ناشی از روشن بودن کنترل‌کننده‌های اضافی و لود اضافی اطلاعات ارسالی به کنترل‌کننده‌ها کاهش می‌یابد.

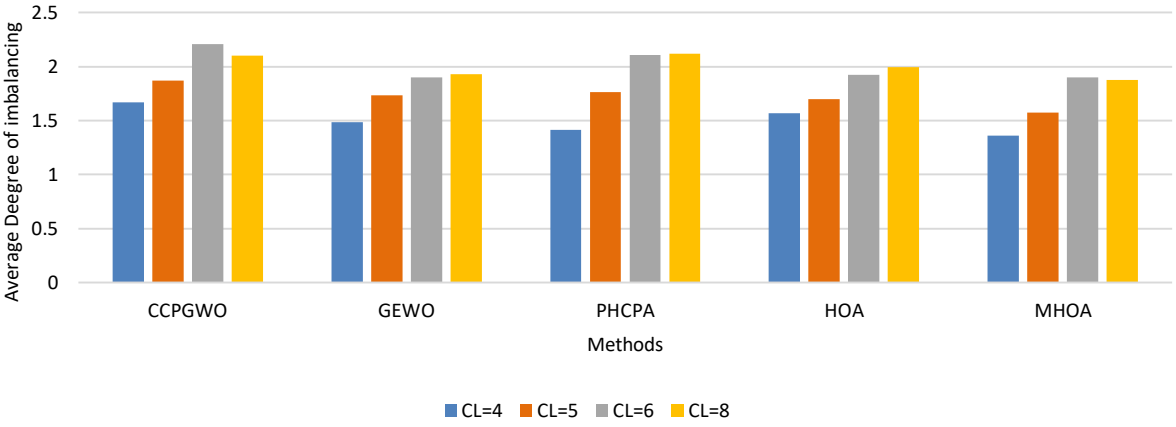
شکل ۱۲ تحلیل مقایسه‌ای از الگوریتم MHOA نسبت به الگوریتم‌هایCCPGW، GEWO، PHCPA و HOA را نشان می‌دهد. یافته‌ها برتری عملکرد الگوریتم MHOA در دستیابی به توازن بار، کاهش مصرف انرژی و کمینه کردن تاخیر انتها به انتها نسبت به سایر الگوریتم‌ها را بیان می‌کنند.



(الف)



(ب)



(ج)

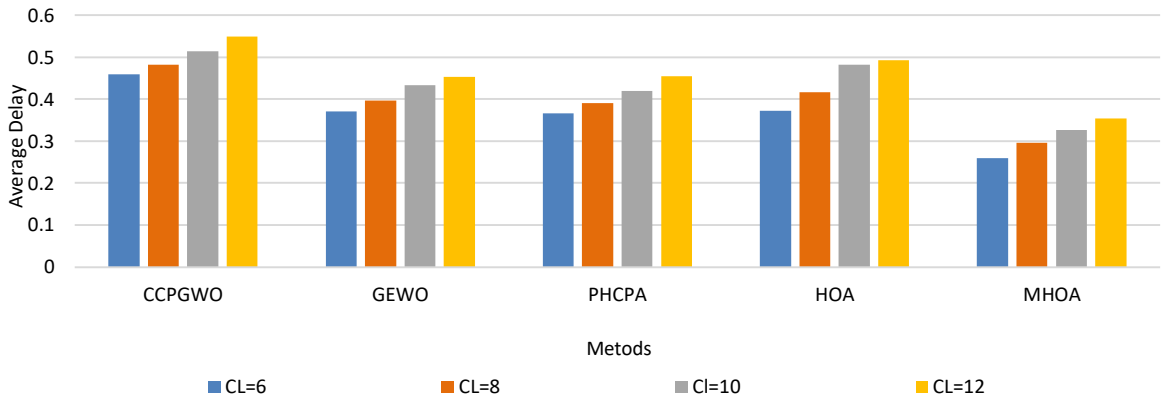
شکل ۱۲: نمودار میله‌ای نتایج به‌دست‌آمده بر روی شبکه Aarnet. الف) میانگین تاخیر انتها به انتها ب) میانگین مصرف انرژی ج) میانگین عدم تعادل بار

به همین ترتیب، نتایج آماری الگوریتم‌ها در جدول ۴ خلاصه شده است، که درصد بهبودهای MHOA نسبت به سایر الگوریتم‌ها در شبکه Aarnet را نشان می‌دهد. MHOA به طور میانگین در مصرف انرژی، ۱۲,۹۷٪ نسبت به CCPGWO، ۱۵,۱۹٪ نسبت به GEWO، ۱۳,۰۲٪ نسبت به PHCPA ، و ۶,۹۵۴۵٪ نسبت به HOA می‌نماید. به طور کلی MHOA، ۱۲٫۳% در مصرف انرژی نسبت به سایر الگوریتم‌ها بهتر عمل می‌کند. علاوه بر این، در بهبود تاخیر انتها به انتها در شبکه‌ی Aarnet، MHOA به‌طور میانگین، ۵۳,۷۹۹۶۳٪ نسبت به CCPGWO، ۱۱,۵۶۰۲۴٪ نسبت به GEWO، ۱۳,۷۶۰٪ نسبت به PHCPA ، و ۱۰,۹۸٪ نسبت به HOA بهتر عمل می‌کند. نتایج نشان می‌دهند که MHOA تاخیر انتها به انتها را ۲۲,۵۲٪ به طور میانگین بهبود می‌دهد. MHOA با موفقیت عدم توازن بار را

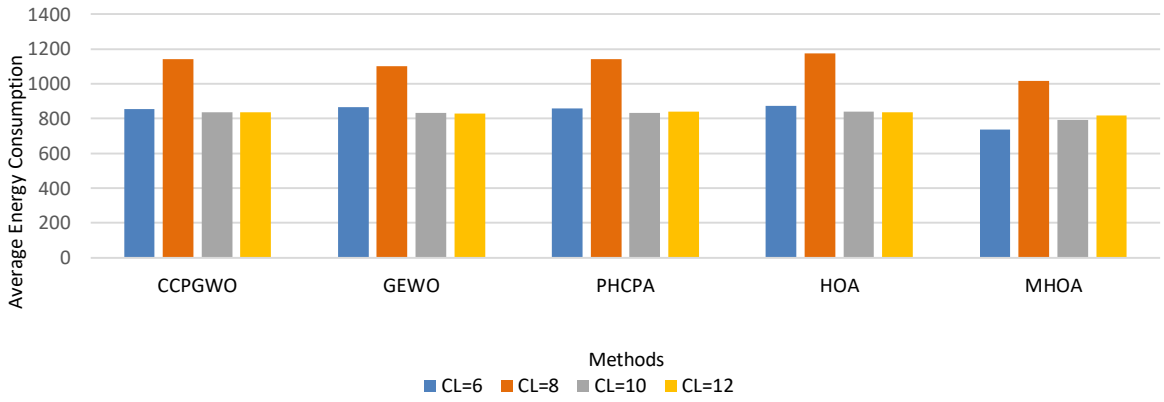
مدیریت می‌کند و به‌طور میانگین ۹,۲۲٪ بهتر از سایر الگوریتم‌های مورد بررسی عمل می‌کند. به‌طور خاص، این الگوریتم در مدیریت عدم توازن بار با بهترین عملکرد ۱۶,۰۴٪ نسبت به CCPGWO، ۶,۸۶٪ نسبت به GEWO، ۸,۴۷٪ نسبت به PHCPA و ۳۶,۳۳٪ نسبت به HOA بهتر عمل می‌کند.

درصد بهبود	HOA	PHCPA	GEWO	CCPGWO	
			میانگین عدم تعادل بار		
۱۲/۷۱	۱۵/۰۹	۳/۷۹	۹/۳۶	۲۲/۵۹	**Cl=۶**
۱۲/۱۲	۷/۷۵	۱۱/۹۷	۱۰/۰۸	۱۸/۶۸	**Cl=۸**
۸/۴۸	۱/۲۹	۱۰/۹۲	۵/۲۹	۱۶/۴۲	**Cl=۱۰**
۳/۵۹	-۲/۰۶	۷/۲۱	۲/۷۲	۶/۴۹	**Cl=۱۲**
میانگین=۹/۲۲	۵/۵۱	۸/۴۷	۶/۸۶	۱۶/۰۴	**درصد بهبود**
			میانگین مصرف انرژی		
۶/۳۴	۴/۷	۸/۶۹	۶/۸۱	۵/۲۳	**Cl=۶**
۹/۵۵	۱/۸۹	۹/۳۴	۱۴/۰۹	۱۲/۸۶	**Cl=۸**
۱۵/۸۷	۴/۷۶	۱۵/۸۴	۲۳/۲۹	۱۹/۵۸	**Cl=۱۰**
۱۶/۳۸	۱۶/۴۵	۱۸/۲۹	۱۶/۵۶	۱۴/۲۱	**Cl=۱۲**
میانگین=۱۲/۰۳	۶/۹۶	۱۳/۰۲	۱۵/۱۹	۱۲/۹۷	**درصد بهبود**
			میانگین تاخیر انتشار		
۱۴/۰۶	۴/۵۹	۱۳/۵۱	۶/۴۸	۳۱/۶۷	**Cl=۶**
۳۱/۳۴	۱۸/۰۷	۱۸/۲۶	۱۲/۵۲	۷۶/۵۱	**Cl=۸**
۲۲/۷۵	۵/۸	۲۰/۴۹	۱۳/۲۰	۵۱/۵۴	**Cl=۱۰**
۲۱/۹۳	۱۵/۴۷	۲/۷۶	۱۴/۰۳	۵۵/۴۶	**Cl=۱۲**
میانگین=۲۲/۵۲	۱۰/۹۸	۱۳/۷۶	۱۱/۵۶	۵۳/۷۹	**درصد بهبود**

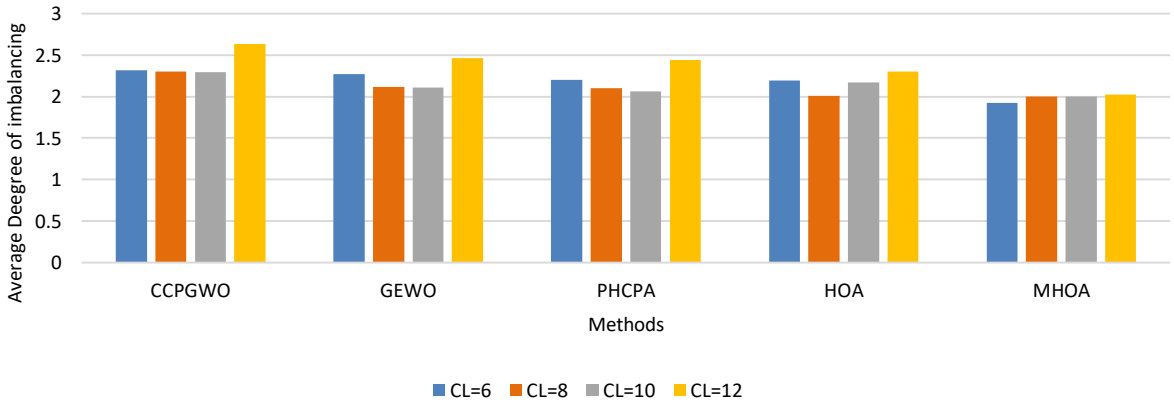
جدول۴: مقایسه آماری الگوریتم‌ها در Aarnet با۳۰ اجرای متفاوت



(الف)



(ب)



(ج)

شکل ۱۳: نمودار میله‌ای نتایج به‌دست‌آمده بر روی شبکه Bics. الف) میانگین تاخیر انتها به انتها ب) میانگین مصرف انرژی ج) میانگین عدم تعادل بار

شکل ۱۳ MHOA را با چهار الگوریتم CCPGWO، GEWO، PHCPA و HOA در شبکه Bics مقایسه می‌کند. ارزیابی با تعداد مختلف کنترل‌کننده‌ها انجام شده است. نتایج نشان می‌دهد که MHOA با توجه به توازن بار، کارایی انرژی و کاهش تاخیر انتها به انتها، نسبت به الگوریتم‌های دیگر عملکرد بهتری دارد. MHOA میانگین عدم تعادل بار، تاخیر انتها به انتها و مصرف انرژی در شبکه را را کاهش می‌دهد.

همان‌طور که در جدول ۵ نشان داده شده است، MHOA به‌طور ثابت نسبت به مصرف انرژی در شبکه Bics عملکرد بهتری نسبت به سایر الگوریتم‌ها دارد و به‌طور میانگین نسبت به الگوریتم‌های CCPGWO، GEWO، PHCPA و HOA، ۹,۷۵٪ مصرف انرژی را کاهش می‌دهد. علاوه بر این، MHOA نسبت به سایر الگوریتم‌ها، به‌طور میانگین، تاخیر انتها به انتها را ۴۰,۳۹٪ بهبود می‌بخشد. به علاوه، MHOA به‌طور موثر نابرابری بار را مدیریت می‌کند و به‌طور میانگین ۱۰,۳۷٪ بهتر از سایر الگوریتم‌ها در مدیریت این مسئله عمل می‌کند.

شکل ۱۴ تجزیه و تحلیل مقایسه ای MHOA را در برابر الگوریتم های CCPGWO، GEWO، PHCPA و HOA در شبکه Colt نشان می دهد. ارزیابی با تعداد متفاوتی از کنترل کننده ها (۶، ۸، ۱۲، و ۱۲) انجام می شود. یافته‌ها عملکرد برتر MHOA را در متعادل‌سازی بار، بهره‌وری انرژی و کاهش تاخیر انتها به انتها در مقایسه با چهار الگوریتم دیگر نشان می‌دهند. این تجزیه و تحلیل نشان می دهد که الگوریتم MHOA به‌طور مداوم عدم تعادل بار را به حداقل می رساند، و در عین حال تاخیر انتها به انتها و مصرف انرژی در شبکه را کاهش می دهد.

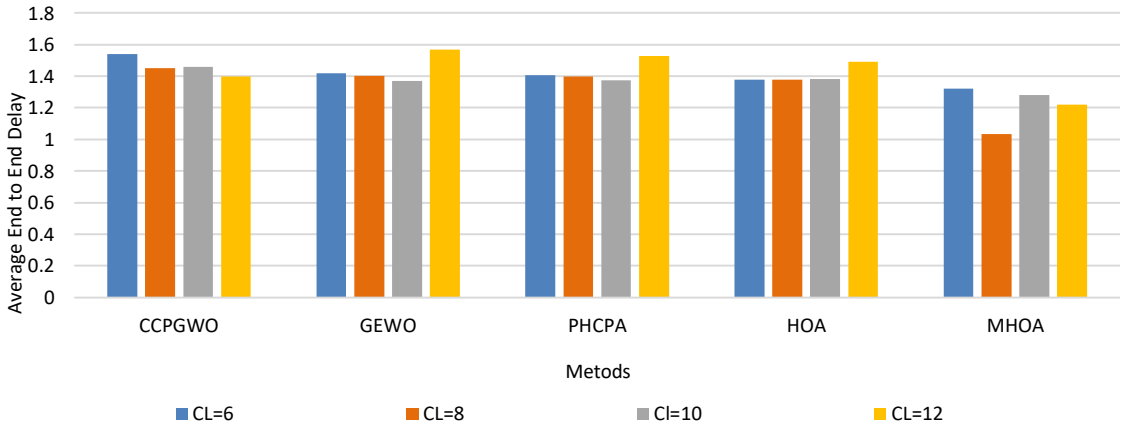
بر طبق جدول ۵، نتایج نشان می‌دهند که MHOA از نظر تاخیر انتها به انتها CCPGWO، GEWO، PHCPA و HOA به ترتیب ۱۹/۶۶٪، ۱۷/۴۹٪، ۱۵/۶۰٪ و ۱۴/۳۷٪ با میانگین کاهش ۱۶/۷۸٪ عملکرد بهتری بر روی شبکه‌ی Colt دارد. علاوه بر این، MHOA در مقایسه با چهار الگوریتم دیگر به‌طور متوسط ۴/۸۳٪ بهبود در مصرف انرژی نشان می دهد.

علاوه بر این، در مدیریت موثر عدم تعادل بار، MHOA در مقایسه با چهار الگوریتم دیگر به‌طور متوسط ۱۰,۱۰٪ بهبود قابل‌توجهی را نشان می دهد. به‌طوری که از CCPGWO با ۲۶/۷۸٪، از GEWO با ۶/۴۶٪، از PHCPA با ۳/۳۹٪ و HOA با ۷/۷۵٪ پیشی گرفته است.

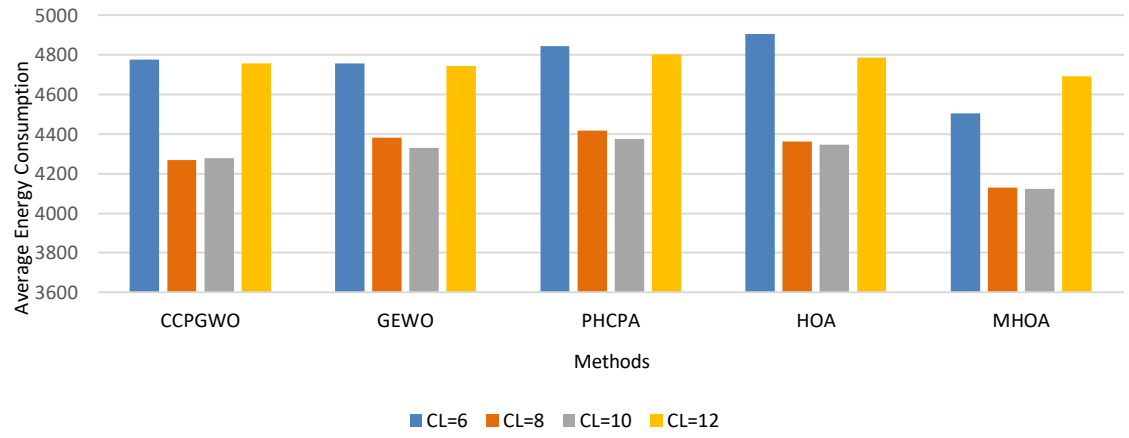
درصد بهبود	HOA	PHCPA	GEWO	CCPGWO
میانگین تعادل بار				
۱۱/۱۵	۸/۵۹	۸/۸۳	۱۲/۳۳	۱۴/۸۳
۶/۵۹	۰/۴۳	۵/۱۶	۵/۸۱	۱۴/۹۵
۷/۸۸	۸/۵۶	۳/۰۴	۵/۱۸	۱۴/۷۳
۱۵/۸۹	۸/۲۹	۱۴/۹۲	۱۶/۱۰	۲۴/۲۴

درصد بهبود	۱۷/۱۹	۹/۸۵	۷/۹۹	۶/۴۷	۱۰/۳۷
میانگین مصرف انرژی					
Cl=۶	۱۵/۹۵	۱۷/۶۷	۱۶/۶۱	۱۸/۷۱	۱۷/۲۴
Cl=۸	۱۲/۲۲	۱۵/۱۷	۱۲/۳۲	۱۵/۳۹	۱۳/۷۸
Cl=۱۰	۶/۰	۵/۵۱	۵/۳۵	۶/۳۰	۵/۷۹
Cl=۱۲	۲/۳۷	۱/۵۰	۲/۵۳	۲/۳۲	۲/۱۸
درصد بهبود	۹/۱۳	۹/۹۷	۹/۲۰	۱۰/۶۸	۹/۷۵
میانگین تاخیر					
Cl=۶	۷۶/۹۹	۴۲/۴۹	۴۱/۰۲	۳۹/۶۳	۵۰/۰۳
Cl=۸	۶۲/۶۹	۳۳/۶۷	۳۱/۹۸	۳۲/۰۶	۴۰/۱۰
Cl=۱۰	۵۷/۲۹	۳۲/۴۳	۲۸/۴۹	۲۹/۰۱	۳۶/۸۱
Cl=۱۲	۵۴/۷۸	۲۷/۷۵	۲۸/۱۸	۲۷/۷۴	۳۴/۶۱
درصد بهبود	۶۲/۹۴	۳۴/۰۹	۳۲/۴۲	۳۲/۱۱	۴۰/۳۹

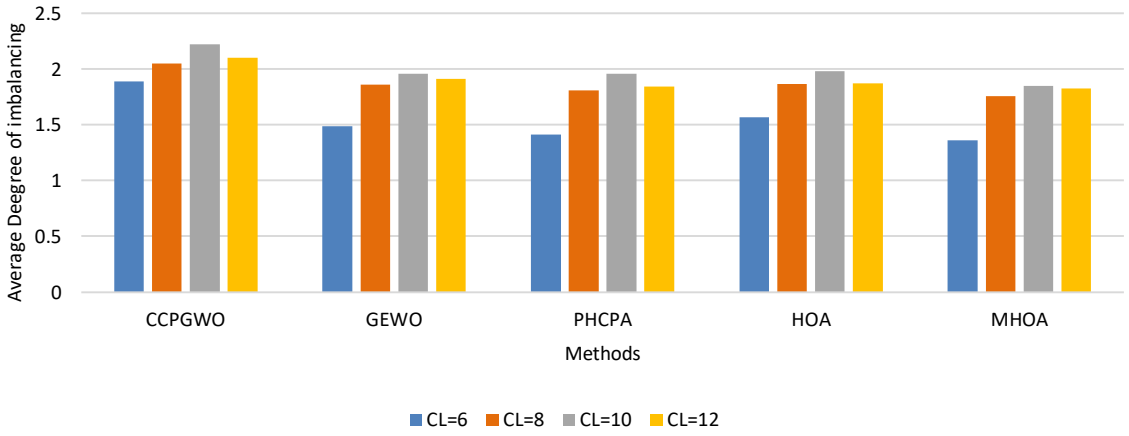
جدول ۵: مقایسه آماری الگوریتم‌ها در Bics با ۳۰ اجرای متفاوت



(الف)



(ب)



(ج)

شکل ۱۴: نمودار میله‌ای نتایج به‌دست‌آمده بر روی شبکه ی Colt. الف) میانگین تاخیر انتها به انتها ب) میانگین مصرف انرژی ج) میانگین عدم تعادل بار

درصد بهبود	HOA	PHCPA	GEWO	CCPGWO	
					میانگین عدم تعادل بار
۱۶/۷۶	۱۵/۰۹	۳/۷۹	۹/۳۶	۳۸/۸۰	**Cl=۶**
۷/۹۶	۶/۲۵	۲/۹۶	۵/۸۲	۱۶/۸۳	**Cl=۸**
۹/۸۳	۷/۱۷	۵/۸۹	۵/۹۱	۲۰/۳۵	**Cl=۱۰**
۵/۸۳	۲/۴۹	۰/۹۲	۴/۷۵	۱۵/۱۵	**Cl=۱۲**
میانگین=۱۰/۱	۷/۷۵	۳/۳۹	۴/۴۶	۲۲/۷۸	**درصد بهبود**
					میانگین مصرف انرژی
۷/۰	۸/۹۱	۷/۵۱	۵/۵۵	۶/۰۱	**Cl=۶**
۵/۴۹	۵/۵۸	۶/۹۴	۶/۰۹	۳/۳۵	**Cl=۸**
۵/۱۰	۵/۴۱	۶/۱۶	۵/۰۲	۳/۸۰	**Cl=۱۰**
۱/۷۳	۲/۰۵	۲/۳۵	۱/۱۰	۱/۴	**Cl=۱۲**
میانگین=۴/۸۳	۵/۴۹	۵/۷۴	۴/۴۴	۳/۶۴	**درصد بهبود**
					میانگین تاخیر
۸/۵۹	۴/۲۵	۶/۴۱	۷/۱۹	۱۶/۵۲	**Cl=۶**
۳۵/۸۹	۳۳/۲۲	۳۴/۸۵	۳۴/۴۴	۴۰/۰۵	**Cl=۸**
۹/۰۴	۷/۷۹	۷/۲۹	۷/۰۷	۱۴/۰۳	**Cl=۱۰**
۱۳/۶۰	۱۲/۲۵	۱۳/۸۵	۲۰/۲۷	۸/۰۳	**Cl=۱۲**
میانگین=۱۶/۷۸	۱۴/۳۷	۱۵/۶۰	۱۷/۴۹	۱۹/۶۶	**درصد بهبود**

جدول ۶: مقایسه آماری الگوریتم ها در Colt با۳۰ اجرای متفاوت

نتایج نشان‌دهنده برتری قابل توجهی از MHOA نسبت به سایر الگوریتم ها در معیارهای مختلف است و از نظر کاهش تاخیر انتها به انتها، ۳۶/۷۲٪ بهتر از CCPGWO، ۱۴/۵۲٪ بهتر از GEWO، ۱۴/۶۸٪ بهتر از PHCPA و ۱۲/۶۷٪ بهتر از HOA عمل می‌کند. علاوه بر این، از لحاظ تعادل بار، به ترتیب ۱۹/۴۱٪، ۶/۶۶٪، ۵/۹۳٪ و ۶/۶۳٪ از CCPGWO، GEWO، PHCPA و HOA پیشی می‌گیرد. از نظر مصرف انرژی، MHOA به ترتیب ۸/۳۰، ۹/۸۱، ۹/۳۸ درصد و ۶/۲۲درصد در مقایسه با CCPGWO، GEWO، PHCPA و HOA برتری دارد. به‌طور خلاصه، نتایج نشان می‌دهد که MHOA به‌طور موثر عدم تعادل بار را تا ۹/۶۶٪ کاهش می دهد، تاخیر انتها به انتها را تا ۱۹/۶۵٪ کاهش می دهد و متوسط مصرف انرژی را تا ۸/۴۳٪ کاهش می‌دهد.

۶- نتیجه‌گیری

این مقاله نسخه‌ی اصلاح‌شده‌ای از الگوریتم بهینه‌سازی HOA به نام MHOA را معرفی می‌کند که از استراتژی تضاد نخبگان و عملگرهای ژنتیک برای بهبود همگرایی و سرعت استفاده کرده است. استفاده از عملگرهای ژنتیکی قابلیت‌های اکتشاف و بهره‌برداری الگوریتم را افزایش می‌دهد. هدف این تحقیق دستیابی به توزیع بار متعادل بر روی کنترل‌کننده‌ها برای کاهش مصرف انرژی و تاخیر انتها به انتها است. الگوریتم پیشنهادی بر روی سه شبکه‌ی متفاوت SDN در دنیای واقعی با توپولوژی و اندازه‌ی متفاوت ارزیابی شده‌اند. نتایج نشان می‌دهد که الگوریتم MHOA در مقایسه با سایر الگوریتم‌های فراابتکاری، به‌طور موثر عدم تعادل بار را به حداقل می‌رساند، مصرف انرژی را کاهش می‌دهد و با بهبود اکتشاف و نرخ بهره‌برداری، تاخیر انتها به انتها کمتری را به‌دست می‌آورد. به‌طور خاص، الگوریتم MHOA عدم تعادل بار را تا ۹/۶۶٪ ، تاخیر انتها به انتها را تا ۱۹/۶۵٪ و در عین حال مصرف انرژی را تا ۸/۴۳٪ کاهش می دهد. با این حال، الگوریتم به منابع CPU و ظرفیت ذخیره سازی اضافی نیاز دارد. تلاش‌های آینده بر بهینه‌سازی استفاده از CPU و حذف نیاز به فضای ذخیره‌سازی اضافی برای کاهش این نیازها تمرکز خواهد کرد.

مراجع

[۱] G. Ramya and R. Manoharan, "Enhanced optimal placements of multi-controllers in SDN," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8187-8204, 2021.

[۲] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7545-7593, 2020.

[۳] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27, pp 51, 2014.

[۴] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, 2014.

[۵] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "SADM-SDNC: security anomaly detection and mitigation in software-defined networking using C-support vector classification," *Computing*, vol. 103, pp. 641-673, 2021.

[۶] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "A survey and classification of the security anomaly detection mechanisms in software defined networks," *Cluster Computing*, vol. 24, pp. 1235-1253, 2021.

[۷] A. Shirmarz and A. Ghaffari, "An adaptive greedy flow routing algorithm for performance improvement in software-defined network," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 33, no. 1, p. e2676, 2020.

[۸] G. D'Angelo and F. Palmieri, "A co-evolutionary genetic algorithm for robust and balanced controller placement in software-defined networks," *Journal of Network and Computer Applications,* vol. 212, p. 103583, 2023.
[۹] A. A. Ateya *et al.*, "Chaotic salp swarm algorithm for SDN multi-controller networks," *Engineering Science and Technology, an International Journal,* vol. 22, no. 4, pp. 1001-1012, 2019.
[۱۰] A. Shirmarz and A. Ghaffari, "Automatic software defined network (SDN) performance management using TOPSIS decision-making algorithm," *Journal of Grid Computing,* vol. 19, pp. 1-21, 2021.
[۱۱] K. P. Sinaga and M.-S. Yang, "Unsupervised K-means clustering algorithm," *IEEE access,* vol. 8, pp. 80716-80727, 2020.
[۱۲] N. Firouz, M. Masdari, A. B. Sangar, and K. Majidzadeh, "A hybrid multi-objective algorithm for imbalanced controller placement in software-defined networks," *Journal of Network and Systems Management,* vol. 30, no. 3, p. 51, 2022.
[۱۳] A. Ghaffari, "Designing a wireless sensor network for ocean status notification system," *Indian Journal of Science and Technology,* vol. 7, no. 6, p. 809, 2014.
[۱۴] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review,* vol. 38, no. 2, pp. 69-74, 2008.
[۱۵] F. MiarNaeimi, G. Azizyan, and M. Rashki, "Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems," *Knowledge-Based Systems,* vol. 213, p. 106711, 2021.
[۱۶] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Computer Communication Review,* vol. 42, no. 4, pp. 473-478, 2.۰۱۲
[۱۷] S. Torkamani-Azar and M. Jahanshahi, "A new GSO based method for SDN controller placement," *Computer Communications,* vol. 163, pp. 91-108, 2020.
[۱۸] B. R. Killi and S. V. Rao, "Poly-stable matching based scalable controller placement with balancing constraints in SDN," *Computer Communications,* vol. 154, pp. 82-91, 2020.
[۱۹] A. Jalili, M. Keshtgari, and R. Akbari, "Optimal controller placement in large scale software defined networks based on modified NSGA-II," *Applied Intelligence,* vol. 48, pp-۲۸۰۹ . .۲۰۱۸ ,۲۸۲۳
[۲۰] C. Gao, H. Wang, F. Zhu, L. Zhai, and S. Yi, "A particle swarm optimization algorithm for controller placement problem in software defined network," in *Algorithms and Architectures for Parallel Processing: 15th International Conference, ICA3PP 2015, Zhangjiajie, China, November 18-20, 2015, Proceedings, Part III 15*, 2015: Springer, pp. 44-54 .
[۲۱] K. Kanodia, S. Mohanty, K. Kurroliya, and B. Sahoo, "CCPGWO: A meta-heuristic strategy for link failure aware placement of controller in SDN," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020: IEEE, pp. 859-863 .
[۲۲] S. Rahman *et al.*, "Virtualized controller placement for multi-domain optical transport networks using machine learning," *Photonic Network Communications,* vol. 40, pp. 126-136, 2020.
[۲۳] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *2016 IEEE International Conference on Communications (ICC)*, 2016: IEEE, pp. 1-6 .
[۲۴] P. Yi, T. Hu, Y. Hu, J. Lan, Z. Zhang, and Z. Li, "SQHCP: Secure-aware and QoS-guaranteed heterogeneous controller placement for software-defined networking," *Computer Networks,* vol. 185, p. 107740, 2021.

[۲۵] N. Firouz, M. Masdari, A. B. Sangar ,and K. Majidzadeh, "A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multi-controller software-defined networks," *Cluster Computing,* vol. 24, pp. 2511-2544, 2021.
[۲۶] L. Wei, C. Chang, Y. Liu, and Y. Wang, "Energy-Efficient Controller Placement in Software-Defined Satellite-Terrestrial Integrated Network," *Remote Sensing,* vol. 14, no. 21, p. 5561, 2022.
[۲۷] Y. Hu, T. Luo, N. C. Beaulieu, and C. Deng, "The energy-aware controller placement problem in software defined networks," *IEEE Communications Letters,* vol. 21, no. 4, pp. 741-744, 2016.
[۲۸] C. Li, K. Jiang, and Y. Luo, "Dynamic placement of multiple controllers based on SDN and allocation of computational resources based on heuristic ant colony algorithm," *Knowledge-Based Systems,* vol. 241, p. 108330, 2022.
[۲۹] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06),* 2005, vol. 1: IEEE, pp. 695-701 .
[۳۰] "The controller placement problem. ," *ACM SIGCOMM Computer Communication Review,* vol. 42, no. 4, pp. 473-478, 2012.
[۳۱] V. B. Glen, "Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry. ," *princeton University Press, princeton,* 2013.
[۳۲] "A new GSO based method for SDN controller placement. ," *Computer Communications,* vol. 163 ,pp. 91-108, 2020.
[۳۳] "Controller placement in software defined networks: A Comprehensive survey " *Computer Networks* vol. 163, pp. 1-16, 2019.
[۳۴] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proceedings of the 2013 25th international teletraffic congress (ITC),* 2013: IEEE, pp. 1-9 .
[۳۵] M. Khojand, K. Majidzadeh, M. Masdari, and Y. Farhang, "Controller placement in SDN using game theory and a discrete hybrid metaheuristic algorithm," *The Journal of Supercomputing,* pp. 1-49, 2023.
[۳۶] M. A. Bagha, K. Majidzadeh, M. Masdari, and Y. Farhang, "ELA-RCP: An energy-efficient and load balanced algorithm for reliable controller placement in software-defined networks," *Journal of Network and Computer Applications,* vol. 225, p. 103855, 2024.
[۳۷] M. Abedini Bagha, K. Majidzadeh, M. Masdari, and Y. Farhang, "Improving delay in SDNs by metaheuristic controller placement," *International Journal of Industrial Electronics Control and Optimization,* vol. 5, no. 4, pp. 286-296, 2022.
[۳۸] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications,* vol. 29, no. 9, pp. 1765-1775, 2.۰۱۱
[۳۹] A. Mohammadi-Balani, M. D. Nayeri, A. Azar, and M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Computers & Industrial Engineering,* vol. 152, p. 107050, 2021.
[۴۰] Y. Shi, "Particle swarm optimization ,*IEEE connections,* vol. 2, no. 1, pp. 8-13, 2004.
[۴۱] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowledge-based systems,* vol. 165, pp. 169-196, 2019.