



جای گذاری کنترل کننده ها در شبکه های تعریف شده نرم افزار

با استفاده از یک الگوریتم فراابتکاری بر پایه ی ژنتیک

مهناز خجند^(۱) کامبیز مجیدزاده^{(۲)*} محمد مصدري^(۳) یوسف فرهنگ^(۴)

(۱) گروه مهندسی کامپیوتر، واحد ارومیه، دانشگاه آزاد اسلامی، ارومیه، ایران

(۲) گروه مهندسی کامپیوتر، واحد ارومیه، دانشگاه آزاد اسلامی، ارومیه، ایران*

(۳) گروه مهندسی کامپیوتر، واحد ارومیه، دانشگاه آزاد اسلامی، ارومیه، ایران

(۴) گروه مهندسی کامپیوتر، واحد خوی، دانشگاه آزاد اسلامی، خوی، ایران

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

چکیده

شبکه های نرم افزار محور شامل جداسازی صفحه کنترل از صفحه داده است. در شبکه های نرم افزار محور کنترل شبکه توسط موجودیتی به نام کنترل کننده که در صفحه کنترل قرار دارد، تعیین می شود. تعیین تعداد و مکان بهینه کنترل کننده ها در صفحه کنترل به عنوان مسئله ی جای گذاری کنترل کننده ها شناخته می شود. در این مقاله در دو فاز مسئله ی جای گذاری کنترل کننده ها با استفاده از خوشه بندی و الگوریتم بهینه ساز گله اسب (HOA) حل شده است. در فاز اول تعداد کنترل کننده ها با استفاده از خوشه بندی تعیین می شود. الگوریتم استفاده شده برای خوشه بندی ورژنی از K-means می باشد که علاوه بر خوشه بندی سوئیچ ها، تعداد کنترل کننده ها را نیز تعیین می کند. در فاز دوم برای تعیین مکان بهینه کنترل کننده در داخل هر خوشه از الگوریتم فراابتکاری گله اسب استفاده می شود. در اینجا برای بالا بردن قدرت اکتشاف و بهره برداری الگوریتم گله اسب و همین طور بهبود همگرایی و سرعت آن از عملگرهای ژنتیک و استراتژی تضاد نخبان استفاده شده است که الگوریتم گله اسب بهبود یافته (MHOA) نام گذاری می شود. نتایج نشان می دهد که روش پیشنهادی در مقایسه با سایر الگوریتم های 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 الگوریتم های دیگر دست یافته است.

کلمات کلیدی: شبکه تعریف شده توسط نرم افزار، مسئله قرارگیری کنترل کننده، الگوریتم های فراابتکاری، الگوریتم بهینه سازی مبتنی بر تضاد نخبان، الگوریتم ژنتیک

*عهده دار مکاتبات:

کامبیز مجیدزاده

نشانی: گروه مهندسی کامپیوتر، واحد ارومیه، دانشگاه آزاد اسلامی، ارومیه، ایران

پست الکترونیکی: kambiz.majidzadeh@iau.ac.ir

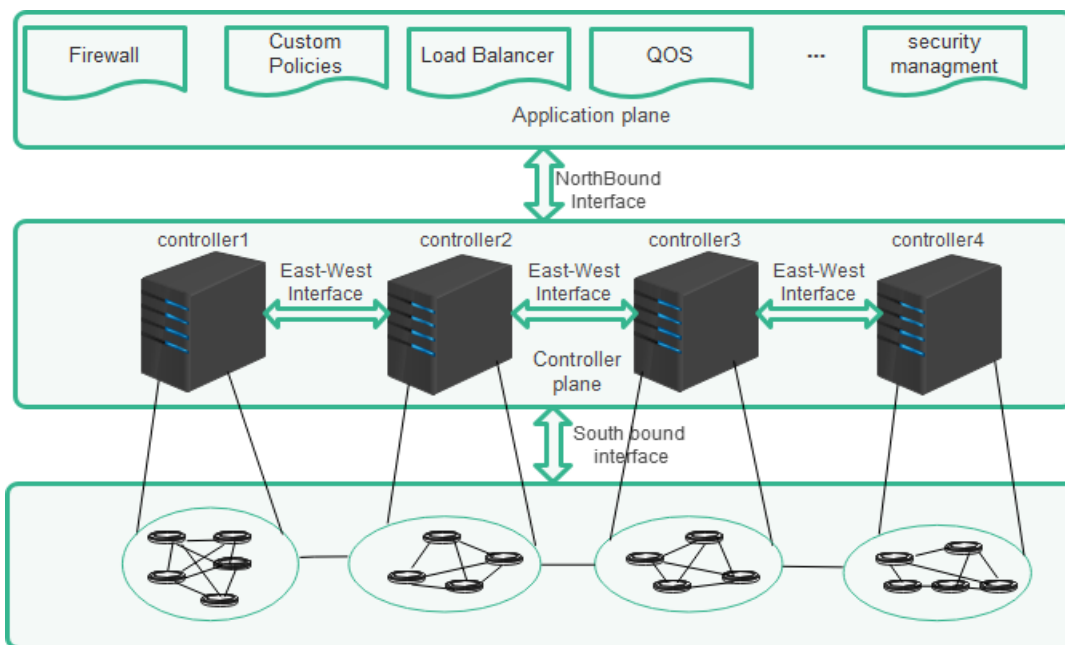
در حال حاضر، با توجه به افزایش تعداد کاربران، حجم بالای داده‌ها، پیچیدگی و ترافیک داده‌ها، شبکه‌های سنتی قادر به پاسخگویی نبوده و با چالش‌های بسیاری از لحاظ مدیریت و پاسخ‌دهی روبه‌رو هستند [4-1]. در راستای پاسخ به چالش‌های مطرح‌شده، معماری شبکه‌های نرم‌افزارمحور^۱ به عنوان یک راه‌حل پیشنهاد شده است. 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 لایه 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 صفحه 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 مسئله‌ای جای‌گذاری کنترل‌کننده‌ها^۲ 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 کارایی 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^۳ 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 بهینه‌سازی گله اسب^۱ 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 بهینه ی محلی گیر کند و دچار پایان زودهنگام شود. MHOA برای افزایش قدرت اکتشاف^۱ و بهره وری^۲ و بهبود همگرایی، از استراتژی تضاد نخبگان^۳ و عملگرهای ژنتیک استفاده می کند. در ادامه از MHOA برای حل مسئله جای گذاری کنترل کننده ها استفاده می شود. برای ارزیابی الگوریتم پیشنهادی، از مجموعه داده ی اینترنتی توپولوژی زو^۴ استفاده شده است. این مطالعه نشان می دهد که الگوریتم MHOA در مقایسه با دیگر روش های فراابتکاری به طور موثری باعث بهبود تأخیر انتها به انتها، مصرف انرژی و تعادل بار می گردد.

خلاصه ی کارهای انجام شده در این مقاله به شرح زیر است:

- پیدا کردن تعداد کنترل کننده ها.
- بهبود الگوریتم گله اسب با استفاده از استراتژی تضاد نخبگان و استفاده از عملگرهای ژنتیک (MHOA).
- استفاده از MHOA برای حل مسئله ی جای گذاری کنترل کننده ها.
- ارزیابی قابلیت مقیاس پذیری MHOA برای قرار دادن کنترل کننده ها در دو شبکه با اندازه ی متفاوت از مجموعه داده ی توپولوژی زو.
- مقایسه ی نتایج الگوریتم پیشنهادی با نتایج الگوریتم های فراابتکاری دیگر از لحاظ تأخیر انتها به انتها، مصرف انرژی و تعادل بار.

بخش ۲ این مقاله، خلاصه ای از مطالعات قبلی انجام شده در این حوزه را ارائه داده و فصل ۳ الگوریتم های استفاده شده در این مقاله، به عنوان مثال HOA و K-means بهبود یافته توضیح داده شده و با معرفی MHOA به چگونگی حل CPP می پردازد. بخش ۴ نحوه ی ارزیابی کار را نشان می دهد و در نهایت، در بخش ۶ خلاصه ای از نتایج به دست آمده از مقاله ارائه می شود.

۲- کارهای گذشته

در این بخش، ما برخی از کارهای انجام شده در حوزه CPP با استفاده از روش های فراابتکاری را معرفی می کنیم. مسئله ی جای گذاری کنترل کننده به صورت پویا برای نخستین بار توسط [16] مورد مطالعه قرار گرفت. نویسندگان مسئله ی "جای گذاری کنترل کننده به صورت پویا" را به عنوان یک برنامه ی 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 تأخیر 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 تأخیر، 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 تأخیر، 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 تأخیر سرتاسری را به عنوان یک معیار برای حل مسئله مورد بررسی قرار دادند. این الگوریتم با وجود اینکه به‌طور مؤثری تأخیر 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 نظر گرفته شد. هدف این پژوهش یافتن بهترین مکان و عملکرد برای کنترل‌کننده‌ها در یک شبکه بود. نتایج نشان دادند که CCPGWO در کمینه کردن تأخیر 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 نمی کند. در مقاله ی [22]، نویسندگان با در نظر گرفتن عواملی مانند انواع مختلف کنترل کننده، منابع محدود در موقعیت های لبه و ضرورت کاهش تاخیر، یک روش جدید برای قرارگیری کنترل کننده ها در شبکه های انتقال نوری ارائه کردند. آن ها همچنین یک روش بر پایه ی روش های یادگیری ماشین، برای کمک به پیش بینی مکان های بهینه معرفی کردند. این رویکرد دارای چندین مزیت مانند کاهش تاخیر، بهبود عملکرد و صرفه جویی در هزینه بود. با این حال، قابلیت اعمال این مطالعه محدود به شبکه های انتقال چند دامنه است، که باعث کاهش قابلیت استفاده گسترده ی آن می شود. علاوه بر این، پیاده سازی 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. آن ها فرض کردند که توپولوژی شبکه پیش تعیین شده است، که این ممکن است همیشه در سناریوهای واقعی صحیح نباشد.

نویسندگان در [24]، پروژه SQHCP را برای حل مسئله ی CPP ارائه دادند، که هدف قرار دادن کنترلرهای نامتجانس به صورت امن و با اطمینان از کیفیت بود. آن ها نیازهای امنیت و کیفیت سرویس و در عین حال خطاهای لایه کنترل و تضمین استفاده متعادل از کنترلر ها را در نظر گرفتند. با این حال، پیاده سازی این رویکرد ممکن است نسبت به روش های 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 کنترلر ها را تعیین کنند. آن ها از الگوریتم جست و جوی تابو یکپارچه پارتو برای شناسایی بهترین مکان برای این کنترلر ها استفاده کردند. آن ها همچنین یک روش فرا ابتکاری برای مدیریت سناریوهای پویا مانند خرابی های پیوند و عدم تعادل بار بر روی کنترل کننده ها ارائه دادند. در [25]، نویسندگان الگوریتم بهینه سازی نوآورانه ای به نام PHCPA معرفی کردند که از الگوریتم های بهینه سازی الهام گرفته از طبیعت و تقسیم بندی شبکه برای بهبود عملکرد شبکه استفاده می کرد. هدف آن ها بهینه سازی موقعیت چندین کنترلر در SDN بود؛ به طوری که، حداقل تاخیر بین سوئیچ ها و کنترلر ها را کاهش دهد. با این حال، این الگوریتم نیاز به زمان CPU و فضای ذخیره سازی بیشتری نسبت به الگوریتم های جایگزین داشت. در [8]، پژوهشگران یک الگوریتم ژنتیک تکاملی را برای حل مسئله ی CPP معرفی کردند. آن ها در این تحقیق، تخصیص بهینه سوئیچ ها، توازن بار و جلوگیری از خرابی ها را در نظر گرفته و در عین حال از اعمال تغییرات در نصب کنترل کننده ها و جریان ترافیک خودداری کردند. با این حال، پیاده سازی این راه حل ممکن است برای شبکه هایی با مقیاس بزرگ، منابع محاسباتی قابل توجهی را مطالبه کند.

در [26]، نویسندگان برای بهبود عملکرد و کارایی 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، پیاده‌سازی 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 تأخیر
[8]	جست‌وجوی تکاملی، الگوریتم ژنتیک	نیاز به منابع محاسباتی بیشتر	نگاشت سوئیچ به کنترل کننده‌ها، افزایش تغادل بار، کاهش تأخیر
[17]	کوله پشتی ۰-۱، مار گارتنر	نادیده گرفتن مقیاس پذیری، نیاز به منابع محاسباتی بیشتر	کاهش پیچیدگی زمانی، کاهش تأخیر
[18]	الگوریتم شبیه‌ساز حرارت	نادیده گرفتن دسترس پذیری	بهبود تعادل بار، کاهش تأخیر
[19]	الگوریتم ژنتیک رتبه بندی نامغلوب	در نظر گرفتن شبکه به صورت استاتیک	بهبود قابلیت اطمینان و تأخیر در شبکه
[20]	ازدحام ذرات	عدم مطرح نحوه گسسته سازی الگوریتم	افزایش تعادل بار و کاهش تأخیر 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 شبکه های 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

جدول ۱: مروری بر کارهای انجام شده در حوزه SDN

۳- شرح مسئله

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

۳-۱- روش K-means بهبود یافته

الگوریتم K-means یک روش خوشه بندی محبوب است که هدف آن بیشینه سازی شباهت درون خوشه ها و افزایش اختلاف میان آن ها می باشد. این روش با اینکه در زیر مجموعه ی روش های یادگیری بدون نظارت قرار دارد، اما لازم است که تعداد خوشه ها از قبل مشخص شوند. الگوریتم K-means بهبود یافته، نسخه ای اصلاح شده از الگوریتم K-means است که نیاز به پیش فرض ها و انتخاب پارامتر ندارد و می تواند به طور خودکار تعداد بهینه ی خوشه ها را نیز تعیین کند [11]. این الگوریتم از مفهوم آنتروپی استفاده می کند. در این هنگام $X = \{x_1, \dots, x_n\}$ مجموعه داده هایی است که باید خوشه بندی شوند و $A = \{a_1, \dots, a_n\}$ مراکز خوشه می باشند. 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.

$$z_{ik} = \begin{cases} 1 & \text{if } \|x_i - a_k\|^2 - \gamma \ln \alpha_k = \min_{1 \leq k \leq c} \|x_i - a_k\|^2 - \gamma \ln \alpha_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

الگوریتم K-means بهبود یافته از مفهوم آنتروپی استفاده می‌کند 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.

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

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.

$$\alpha_k^{t+1} = \sum_{i=1}^n z_{ik} / n + (\beta / \gamma) \alpha_k^t \left[\ln \alpha_k^t - \sum_{s=1}^c \alpha_s^{(t)} \ln \alpha_s^{(t)} \right] \quad (3)$$

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

$$\beta_t = \min \left[\frac{\sum_{k=1}^c \exp(-n |\alpha_k^{(t+1)} - \alpha_k^{(t)}|)}{c}, \frac{1 - \max_{1 \leq k \leq c} (\frac{1}{n} \sum_{i=1}^n z_{ik})}{(-\max_{1 \leq k \leq c} \alpha_k^{(t)} \sum_{k=1}^c \ln \alpha_k^{(t)})} \right] \quad (4)$$

تا زمانی که 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 بهینه سازی گله اسب (HOA) یک الگوریتم فرا ابتکاری است که برای حل مسائل بهینه سازی با تعداد زیادی متغیر طراحی شده است. این الگوریتم الهام خود را از رفتار گله های اسب گرفته و رفتارهای اجتماعی آن ها در مراحل مختلف زندگی را تقلید می کند. این رفتار شامل چرا، سلسله مراتب، اجتماعیت، تقلید، مکانیزم دفاع و گردشگری است [15]. رویکرد الگوریتم شامل حرکت اسب ها در هر تکرار است که توسط رابطه ی (۵) تعیین می شود.

$$\begin{aligned} x_m^{Iter.Age} &= \vec{V}_m^{Iter.Age} + X_m^{(Iter-1).Age} \\ Age &= \alpha, \beta, \gamma, \delta \end{aligned} \quad (5)$$

در HOA، موقعیت هر اسب به عنوان $x_m^{Iter.Age}$ نمایش داده می شود که Age مربوط به بازه ی سنی اسب است و $Iter$ نشان دهنده ی تکرار کنونی است. بردار سرعت یک اسب به عنوان $v_m^{Iter.Age}$ نشان داده می شود. طول عمر اسب ها در حدود ۲۵-۳۰ است و گروه های سنی مختلف، رفتارهای متمایزی را نشان می دهند. اسب ها در بازه ی سنی ۰ تا ۵ سال به عنوان δ ، در بازه ی سنی ۵ تا ۱۰ سال به عنوان γ ، ۱۰ تا ۱۵ سال به عنوان β و بیش از ۱۵ سال با عنوان α دسته بندی می شوند. برای تعیین سن اسب ها، یک ماتریس جمعیت موجود تولید و مرتب می شود. ۱۰٪ برتر به عنوان اسب های α ، ۲۰٪ بعدی به عنوان اسب های β ، ۳۰٪ به عنوان اسب های γ و بقیه ۴۰٪ به عنوان اسب های δ مشخص می شوند. از روش های ریاضی برای شبیه سازی رفتار اسب ها و محاسبه بردارهای سرعت آن ها استفاده می شود. حرکت اسب ها در هر چرخه الگوریتم با استفاده از رابطه (۶) که الگوهای رفتار مذکور را در نظر می گیرد، توصیف می شود.

$$\begin{aligned} \vec{V}_m^{Iter.\alpha} &= \vec{G}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\beta} &= \vec{G}_m^{Iter.\alpha} + \vec{H}_m^{Iter.\alpha} + \vec{S}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\gamma} &= \vec{G}_m^{Iter.\alpha} + \vec{H}_m^{Iter.\alpha} + \vec{S}_m^{Iter.\alpha} + \vec{I}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha} + \vec{R}_m^{Iter.\alpha} \\ \vec{V}_m^{Iter.\delta} &= \vec{G}_m^{Iter.\alpha} + \vec{I}_m^{Iter.\alpha} + \vec{R}_m^{Iter.\alpha} \end{aligned} \quad (6)$$

۳-۳- استراتژی تضاد نخبگان

استراتژی تضاد نخبگان^۱، یک رویکرد نوآورانه در یادگیری ماشین است که الهام خود را از استراتژی مبتنی بر مخالف^۲ می گیرد [29]. این مفهوم روشی مبتنی بر نخبه گرایی و جهت مخالف آن در همان زمان است. در این روش به منظور افزایش احتمال کشف مناطقی که پتانسیل بیش تری برای بهینه بودن دارند، ابتدا مناطق و حدس های بهینه از بین

¹ Elite Opposition-Based Learning (EOBL)

² Opposition-Based Learning (OBL)

جمعیت اولیه شناخته شده و سپس برای اکتشاف بیش تر و عبور از بهینه های محلی نقاط کاملاً مخالف آن ها نیز انتخاب می شوند. در این روش می توانیم به سمت جهت مخالف هدایت شویم که ما را به سمت نزدیک تر شدن به راه حل بهینه ناشناخته هدایت می کند. لازم به ذکر است که نقطه مخالف از طریق انعکاس نقطه محاسبه شده از طریق نقطه مرکزی به دست می آید. پس اگر x یک عدد حقیقی باشد که در بازه $x \in [a, b]$ قرار دارد، عکس آن، $\check{x}$ توسط رابطه (۷) تعریف می شود.

$$\check{x} = a + b - x \quad (۷)$$

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

$$\check{x} = a_i + b_i - x_i \quad (۸)$$

۳-۴- فرموله سازی مسئله

تمرکز اصلی راه حل های پیشنهاد شده برای مسئله جای گذاری کنترل کننده ها، در تعداد کنترل کننده ها و محل قرارگیری آنها است [16]. در این کار شبکه نرم افزار محور به صورت یک گراف $G(S, E)$ نشان داده می شود، به طوری که S مجموعه ی گره ها و E یال های بین این گره ها در یک شبکه هستند. یال uv یک ارتباط دوطرفه بین گره های u و v است. هر یال دارای وزنی است که نمایانگر پهنای باند ارتباط است. مجموعه ی $C = \{c_1, c_2, \dots, c_k\}$ کنترل کننده های شبکه را نشان می دهد. در شبکه های نرم افزار محور، چندین معیار معمولاً برای ارزیابی کارایی سیستم استفاده می شود. در این مقاله، برخی از این معیارها به شرح زیر تعریف شده اند.

۳-۴-۱- متوسط تأخیر انتشار

نحوه ی محاسبه ی متوسط تأخیر انتشار در شبکه ی نرم افزار محور، توسط رابطه (۹) تعریف می شود [30]. این رابطه از مجموع میانگین زمانی انتشار اطلاعات بین کنترل کننده ها و میانگین زمانی انتشار اطلاعات از سوئیچ ها به کنترل کننده ها به دست می آید. میانگین زمان انتشار اطلاعات بین کنترل کننده ها با Avg_{ICL} و میانگین زمان انتشار اطلاعات از سوئیچ ها به کنترل کننده ها با Avg_{SCL} نشان داده می شود، که در ادامه هر یک توضیح داده شده است.

کوتاه ترین مسیر بین سوئیچ $s, s \in S$ و کنترل کننده ی $c, c \in C$ توسط $d(s, c)$ نمایش داده می شود و کوتاه ترین مسیر بین کنترل کننده ها ی c_i و c_j به صورت $d(c_i, c_j)$ نمایش داده می شود. کوتاه ترین مسیر بین دو گره i و j از طریق فاصله ی اقلیدسی بر طبق رابطه ی (۱۰) به دست می آید [31].

$$Del_{ave}(C) = S2C_{Avg} + C2C_{Avg} \quad (9)$$

$$d(i, j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \quad (10)$$

الف) تأخیر انتشار بین کنترل کننده ها

تأخیر بین کنترل کننده ها اهمیت بسیاری دارد زیرا ارتباط بین کنترل کننده ها برای رسیدن به همگامی بین اجزای شبکه ضروری است. تأخیر بین کنترل کننده ها می تواند به عملکرد شبکه تأثیر بگذارد، زیرا اگر جداول مسیریابی یا اطلاعات کنترلی به موقع به روزرسانی نشوند، کنترل کننده ها ممکن است تصمیمات نادرستی بگیرند. در صورت وجود تأخیر بین کنترل کننده ها، که توسط دورترین کنترل کننده محدود شده است، ممکن است به عملکرد شبکه آسیب وارد شود [32].

متوسط تأخیر بین کنترل کننده ها در شبکه نرم افزار محور را می توان با استفاده از رابطه ی (۱۱) به دست آورد [33].

$$C2C_{Avg} = \frac{1}{|C|. (|C| - 1)} \sum_{i, j \in C} d(i, j) \quad (11)$$

ب) تأخیر انتشار بین سوئیچ تا کنترل کننده

معادله (۱۲) میانگین تأخیر سوئیچ به کنترل کننده را در سیستم های شبکه نرم افزار محور نشان می دهد که در آن

$d(i, j)$ کوتاه ترین مسیر از سوئیچ j تا کنترل کننده i است که با فاصله اقلیدسی محاسبه می شود [33].

$$S2C_{Avg} = \frac{1}{|S|} \sum_{j \in V} \sum_{i \in C} d(i, j) \quad (12)$$

۳-۴-۲- عدم تعادل بار

هر کنترل کننده در شبکه مسئول مدیریت مجموعه‌ای از سوئیچ‌ها است. اما وقتی که سوئیچ‌ها به نزدیک‌ترین کنترل کننده اختصاص داده می‌شوند، توزیع بار بر روی کنترل کننده‌ها نامتوازن می‌شود. به این معنی که برخی از کنترل کننده‌ها به شدت دچار بار کاری زیاد بوده، در حالی که از برخی دیگر به طور کامل استفاده نمی‌شود. به عبارت دیگر، عدم تعادل بار بین دو کنترل کننده توسط تفاوت در تعداد سوئیچ‌های اختصاص داده شده به هر کنترل کننده تعیین می‌شود. عدم تعادل بار بروی یک کنترل کننده 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 کنترل کننده با استفاده از رابطه (۱۳) محاسبه می‌شود که S تعداد گره‌هایی است که قادر به ارتباط با کنترل کننده هستند [34].

$$AvgLI = \max(\max_{|FE|}^S - \min_{|FE|}^S) \quad (13)$$

۳-۴-۳- مصرف انرژی

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

$$EC = \sum_{i=1}^n \left(Eb_{c_i} + Ed_{c_i} \right) + \sum_{c=1}^{c=cl} \sum_{sc} EL_{sc} \quad (14)$$

$$EL_{sc} = \frac{d(c,s)}{bw_{cs}} \times e_1$$

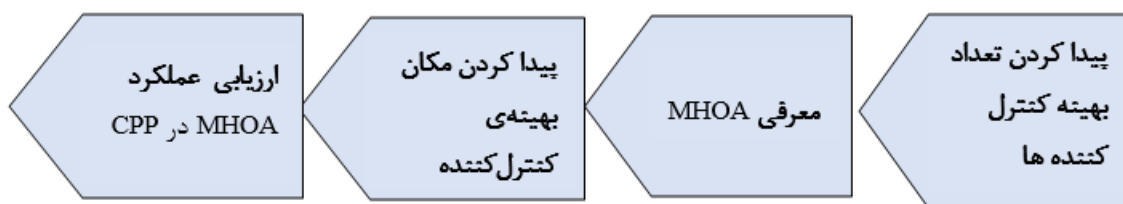
$$Ed_{c_i} = N_S \times e_2$$

۴- بیان مسئله

هدف این مقاله حل مسئله CPP با پیدا کردن تعداد و مکان بهینه‌ی کنترل کننده‌ها 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 مسئله ی NP-hard معرفی شده است. روش های فرا ابتکاری یک گزینه مناسب برای حل مسائل پیچیده و NP-hard مانند CPP هستند که با بررسی بخش کوچکی از فضای جست و جو می توانند راه حل نزدیک به بهینه را پیدا کنند. با این حال، این روش ها ممکن است پیچیده و زمان بر باشند. از طرف دیگر در مورد مسائل خوشه بندی، K-means و روش های چند معیاری مشابه، اغلب به دلیل سرعت، سادگی و کارآمدی می تواند گزینه مناسبی باشند. به همین دلیل در این مقاله در فاز اول برای یافتن تعداد کنترل کننده ها و خوشه بندی از نسخه ی بهبود یافته ی K-mean استفاده شده است که می تواند با پیدا کردن تعداد مناسب خوشه ها بر اساس فاصله اقلیدسی گره ها، عمل خوشه بندی را انجام دهد. در این مقاله با تغییر تابع هدف در این الگوریتم علاوه بر فاصله اقلیدسی گره ها، ترافیک شبکه نیز در نظر گرفته می شود. در فاز دوم مقاله، الگوریتمی بر پایه ی ژنتیک به نام الگوریتم گله اسب بهبود یافته (MHOA) تعریف می شود و از آن برای پیدا کردن مکان کنترل کننده در هر خوشه استفاده می شود. MHOA ورژن بهبود یافته ی الگوریتم گله اسب (HOA) است. HOA یکی از الگوریتم های اخیرا پیشنهاد شده است که در حل بسیاری از مسائل دنیای واقعی به خوبی عمل می کند. با این حال، در برخی مسائل پیچیده ممکن است که در بهینه ی محلی گیر کند. MHOA تا حدی زیاد می تواند این مشکلات را حل کند.

استفاده از یک جمعیت اولیه تصادفی در HOA منجر به بی ثباتی در عملکرد آن می شود. اگر بتوانیم جمعیت اولیه را به گونه ای انتخاب کنیم که تنوع بیشتری داشته باشند مشکل بهینه محلی تا حد زیادی می تواند حل شود و سرعت همگرایی الگوریتم و توانایی آن برای رسیدن به راه حل جهانی بهینه را افزایش می یابد. استفاده از استراتژی تضاد نخبگان می تواند به عنوان نقطه شروع مفیدی برای فرآیند جست و جو عمل کند و به طور بالقوه عملکرد کلی الگوریتم را افزایش دهد. در ابتدا، الگوریتم یک جمعیت اولیه را انتخاب می کند و برازندگی هر راه حل را ارزیابی می کند. سپس استراتژی EOLB برای انتخاب N جمعیت اعمال می شود به طوری که بهترین جمعیت انتخاب شده و برای اکتشاف بالاتر دقتا نقطه ی مقابل این افراد نیز انتخاب می شود. این کار باعث بالا رفتن قدرت اکتشاف الگوریتم می گردد. MHOA از استراتژی EOBL برای اکتشاف بیشتر استفاده می کند. استفاده از عملگرهای ژنتیک قدرت اکتشاف و بهره برداری الگوریتم را بالا می برد و باعث همگرایی بهتر و سرعت بیشتر الگوریتم می شود. در فاز بعدی مقاله از الگوریتم MHOA برای پیدا کردن مکان کنترل کننده در داخل هر خوشه استفاده می شود. هر گره در داخل خوشه می تواند به عنوان کنترل کننده در نظر گرفته می شود. در این سناریو فرض شده است که هر سوئیچ فقط می تواند به یک کنترل کننده متصل شود و دو کنترل کننده با مختصات یکسان در شبکه وجود ندارند. در ادامه، جزئیات بیشتری در مورد نحوه انجام کار ارائه می شود. شکل ۲ ساختار این مقاله را نشان می دهد.



شکل ۲: ساختار کلی مقاله

۴-۱- پیدا کردن تعداد کنترل کننده‌ها با استفاده از K-means بهبودیافته

هر چه قدر تعداد کنترل کننده‌ها زیاد باشد شبکه در حالت ایده‌آل تری قرار خواهد گرفت. با این حال، استفاده از کنترل کننده هزینه‌بر بوده و از طرف دیگر بیشتر از یک تعداد مشخص عملاً تأثیر چندانی بر کارایی شبکه نخواهد داشت. تعداد کم کنترل کننده‌ها نیز می‌تواند به‌طور منفی بر کیفیت شبکه تأثیر بگذارد و منجر به ازدحام در شبکه شود. بنابراین تعیین تعداد بهینه‌ی کنترل کننده‌ها در شبکه حیاتی است. در این مقاله برای تعیین تعداد کنترل کننده‌ها از الگوریتم K-means بهبود یافته و مفهوم خوشه‌بندی استفاده می‌کنیم. در این 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 کنترل کننده برای کنترل سوئیچ‌های خوشه قرار می‌گیرد. تعداد کنترل کننده‌ها برابر با تعداد خوشه‌ها است. در این فاز علاوه بر تعیین تعداد کنترل کننده‌ها، سوئیچ‌های تحت کنترل یک کنترل کننده نیز تعیین می‌شوند. بهتر است که سوئیچ‌های نزدیک به هم و سوئیچ‌هایی که بیشتر با هم در ارتباط هستند تحت نظارت یک کنترل کننده باشند تا مجبور به بارگذاری اطلاعات شبکه بر روی کنترل کننده‌های متفاوت نباشیم. بنابراین سوئیچ‌های نزدیک به هم را در یک خوشه قرار می‌دهیم تا تحت نظارت یک کنترل کننده واحد باشند. از طرف دیگر اگر تعداد سوئیچ‌های یک خوشه و میزان پیام‌های ارسالی در یک شبکه زیاد باشد باعث ازدحام در آن کنترل کننده خواهد شد. بنابراین زمانی که ترافیک در شبکه زیاد باشد باید تعداد خوشه‌ها افزایش یابد تا ترافیک بین کنترل کننده‌ها تقسیم شود. از طرف دیگر، اگر بار شبکه کاهش یابد، توصیه می‌شود تعداد کنترل کننده‌ها کاهش یابد تا از هزینه‌های غیرضروری جلوگیری شود. در ابتدا حداکثر تعداد کنترل کننده‌های موجود به تعداد سوئیچ‌ها، به عنوان ورودی به الگوریتم K-means بهبود یافته داده می‌شود. تعداد خوشه‌ها تعیین کننده‌ی تعداد کنترل کننده‌هاست. 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 خوشه ی k ام توسط α_{ik} نمایش داده می شود. برای هر سوئیچ i ، یک آرایه به نام $strscl$ در نظر گرفته می شود که طول آن برابر با تعداد کنترل کننده ها است، $|strscl| = k$. هر سلول از $strscl$ احتمال نگاشت سوئیچ i به یک کنترل کننده k را نشان می دهد. ابتدا، مقدار α_{ik} به $1/c$ تنظیم می شود که c تعداد کل کنترل کننده ها است که در ابتدا به تعداد سوئیچ ها در نظر گرفته شده است. زمانی که α_{ik} برابر با $1/c$ است، یعنی احتمال تخصیص آن سوئیچ به هر یک از کنترل کننده ها برابر است که نشان دهنده ی عدم وجود اطلاعات قابل دسترس در مورد α_{ik} است. در هر تکرار، α_{ik} بر اساس معادله (۳) محاسبه می شود. برای در نظر گرفتن ترافیک هر خوشه علاوه بر فاصله ی اقلیدسی هر سوئیچ، رابطه ی ۱ به صورت رابطه (۱۵) تعریف می شود. به طوریکه $load_{ctusi}$ میزان بار در خوشه ی i و $AvgLi$ نشان دهنده متوسط بار شبکه می باشد.

$$z_{ik} = \begin{cases} 1 & \text{if } \|x_i - a_k\|^2 - \gamma \ln \alpha_k = \min_{1 \leq k \leq c} \|x^i - a_k\|^2 - \gamma \ln \alpha_k \text{ and } load_{ctusi} < AvgLi \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

مقادیر α_{ik} بر اساس بار کنترل کننده به روزرسانی می شوند. حداکثر باری که یک کنترل کننده می تواند مدیریت کند، مربوط به ویژگی فیزیکی آن کنترل کننده است که از قبل مشخص است. زمانی که بار یک کنترل کننده به حداکثر نزدیک می شود، کنترل کننده در وضعیت بحرانی قرار می گیرد و کارایی آن کاهش می یابد. در چنین مواردی، توصیه می شود تا کنترل کننده های بیشتری به شبکه اضافه شوند تا بار متعادل شود. بنابراین، احتمال انتخاب این کنترل کننده برای همه گره ها کاهش می یابد در حالی که احتمال کنترل کننده های دیگر بر روی گره ها با مقدار مشابه افزایش می یابد. اگر بار بر روی یک کنترل کننده خیلی کم باشد، بهتر است که آن را غیرفعال شود. در این صورت، سوئیچ های متصل به این کنترل کننده باید مجدداً به یک کنترل کننده دیگر اختصاص یابند. بر اساس این اطلاعات، برخی از کنترل کننده ها روشن شده و دیگری خاموش می شوند. پس از به روزرسانی بردارهای احتمال، سوئیچ ها از روش چرخ رولت برای انتخاب کنترل کننده مناسب استفاده می کنند. در این مرحله، تعداد کنترل کننده ها تعیین می شود و سوئیچ های تحت کنترل هر کنترل کننده تعیین می شوند.

۴-۲- تعیین مکان کنترل کننده ها

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 آنجایی که 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 را 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 دقیقاً 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^۱ و جهش^۲ 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^۳ 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^۴، 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^۵، یک نقطه برش و دو والد را انتخاب می‌کند. اطلاعات در یکی از طرفین نقطه برش

¹ crossover

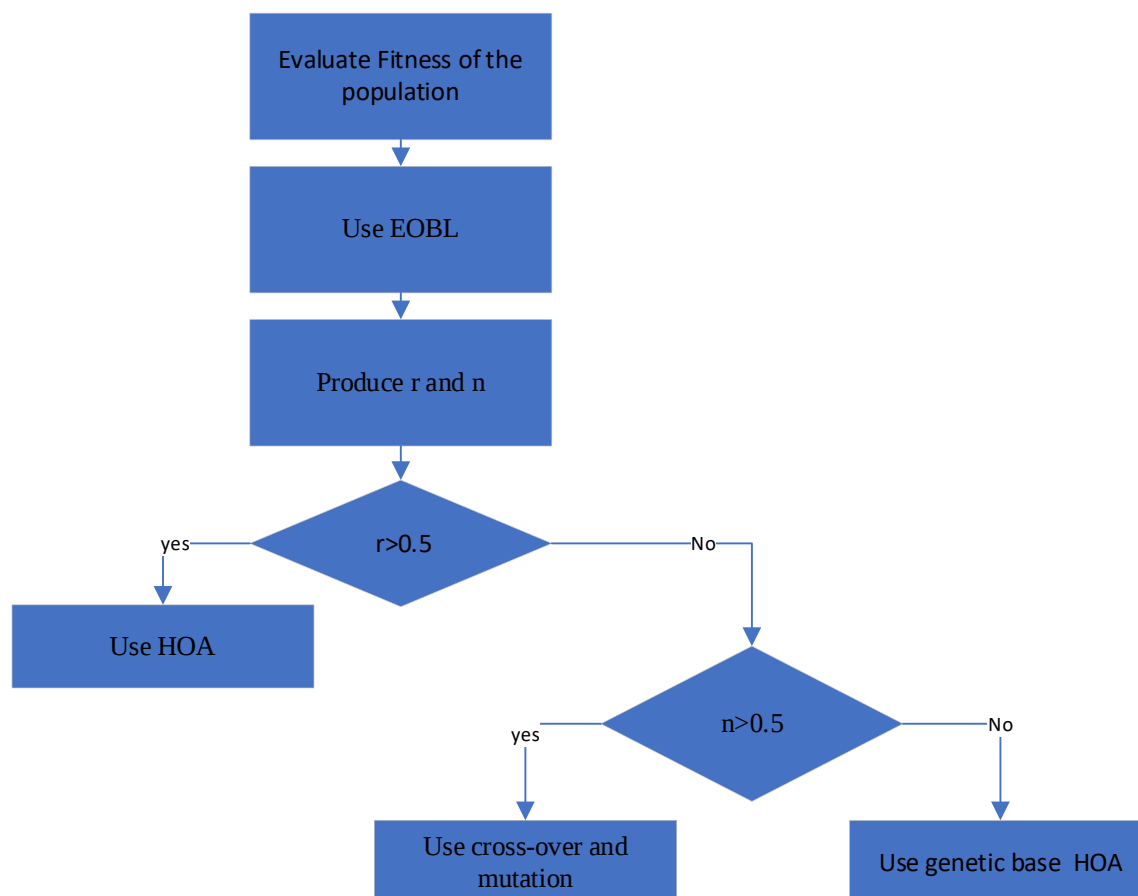
² mutation

³ Duplicate

⁴ Mutation

⁵ One-point crossover

بین والدین معاوضه می شود و دو فرزند جدید تولید می شوند. در عملگر برش دو نقطه ای^۱ دو نقطه ی برش و دو والد انتخاب می شوند و اطلاعات بین این نقاط در دو والد معاوضه می شوند.



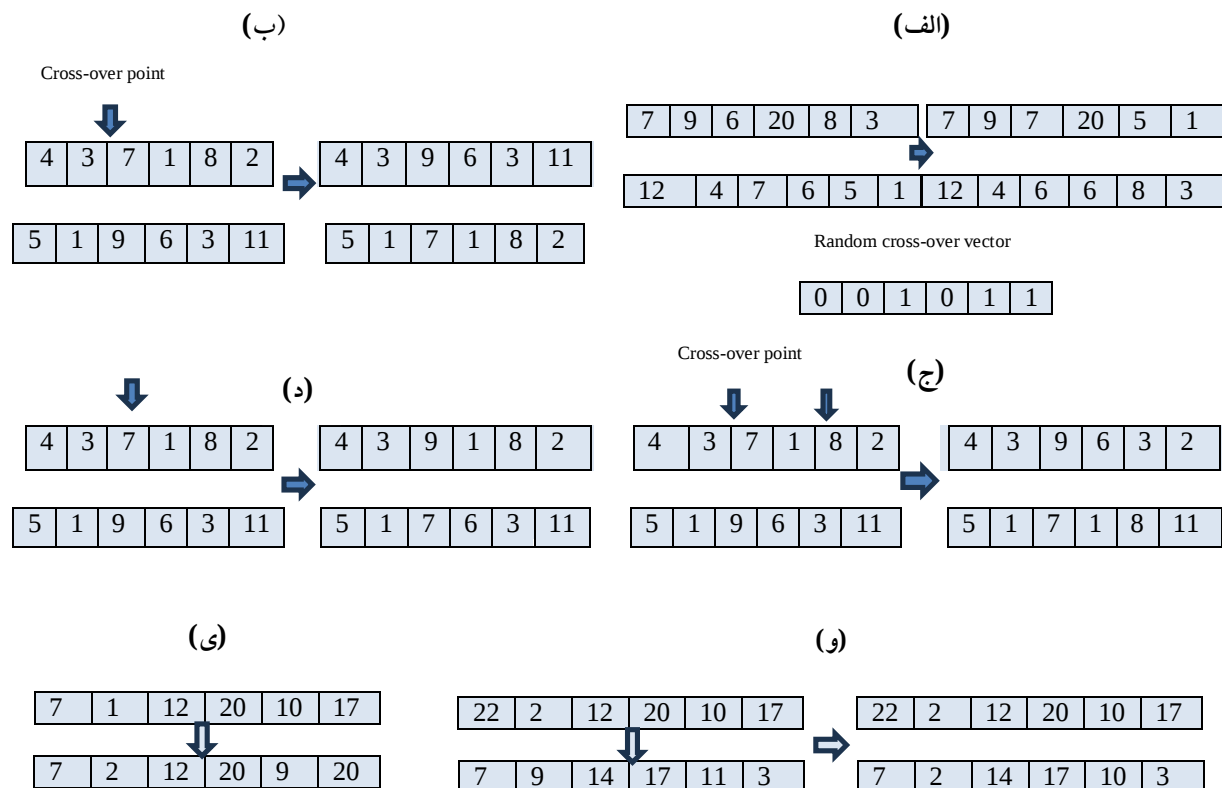
شکل ۳: نحوه عملکرد الگوریتم MHOA

در عملگر برش یکنواخت^۲ یک بردار برش شامل مقادیر دودویی به صورت تصادفی تولید می شود که تعیین می کند کدام سلول ها بین والدین جابه جا شوند. در عملگر تعویض^۳، دو والد انتخاب شده و ژن های آن به صورت تصادفی معاوضه می شوند. در این عملگر، تعداد ژن های معاوضه شده درصدی از کل تعداد کنترل کننده ها 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.

$$\vec{V}_m^{Iter.\alpha} = single\ cross\ over (\vec{G}_m^{Iter.\alpha} + \vec{D}_m^{Iter.\alpha}) \quad (16)$$

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 تک نقطه‌ای 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.

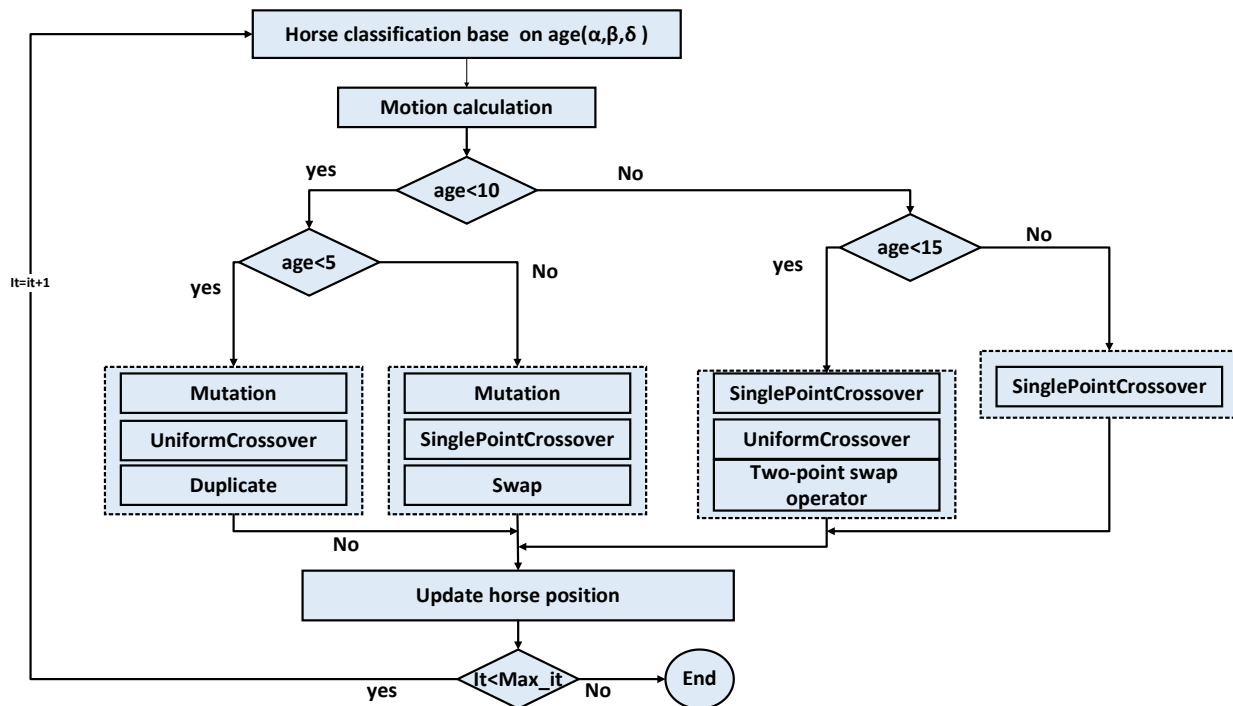
$$\vec{V}_m^{Iter.\beta} = \begin{cases} \text{single cross over } (\vec{G}_m^{Iter.\beta} + \vec{D}_m^{Iter.\beta}) \\ , r \geq 0.75 \\ \text{uniform crossover } (\vec{H}_m^{Iter.\beta} + \vec{S}_m^{Iter.\beta}) \\ , 0.5 \leq r < 0.75 \\ \text{two - point crossover } (\vec{D}_m^{Iter.\beta}, \vec{H}_m^{Iter.\beta}) \\ , r < 0.5 \end{cases} \quad (17)$$

اسب های گروه گاما رفتارهای مختلفی مانند چرا، تقلید سلسله مراتبی، مکانیزم دفاع و اجتماعیت را نشان می دهند. با توجه به رفتار آن ها می توان فهمید که این گروه اسب ها هم تمایل به اکتشاف و هم بهره برداری دارند. رابطه (۱۸) بردار سرعت را برای این اسب ها تعیین می کند. اگر مقدار r بیشتر از ۰٫۷۵ باشد، عملیات جهش روی بردار رفتار تقلید انجام می شود. اگر مقدار r بین ۰٫۵ و ۰٫۷۵ باشد، عملیات جابه جایی روی بردارهای سلسله مراتبی و اجتماعیت انجام می شود. به طور مشابه، اگر مقدار r بین ۰٫۲۵ و ۰٫۵ باشد، یک برش دونقطه ای و جهش بر روی بردارهای مکانیزم دفاع و رفتارهای چرا انجام خواهد شد.

$$\vec{V}_m^{Iter.\gamma} = \begin{cases} \text{Mutation } (\vec{I}_m^{Iter.\gamma}) \\ , r \geq 0.75 \\ \text{Swap } (\vec{G}_m^{Iter.\alpha}, \vec{R}_m^{Iter.\gamma}) \\ , 0.5 \leq r < 0.75 \\ \text{Swap } (\vec{S}_m^{Iter.\alpha}, \vec{H}_m^{Iter.\gamma}) \\ 0.25 < r < 0.5 \\ \text{Mutation } (\text{one - point Crossover } (\vec{R}_m^{Iter.\gamma}, \vec{D}_m^{Iter.\gamma})) \\ r < 0.25 \end{cases} \quad (18)$$

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

$$\vec{V}_m^{Iter.\delta} = \begin{cases} \text{Mutation } (\vec{G}_m^{Iter.\delta}) \\ , r \geq 0.75 \\ \text{niform crossover } (\vec{R}_m^{Iter.\delta} + \vec{I}_m^{Iter.\delta}) \\ , 0.5 \leq r < 0.75 \\ \text{Duplicate } (\vec{G}_m^{Iter.\delta}, \vec{R}_m^{Iter.\delta}) \\ 0.25 < r < 0.5 \end{cases} \quad (19)$$



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. $AvgD_{2e}$ نشان‌دهنده متوسط تأخیر سرتاسری، WD_{2e} نشان‌دهنده 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 می‌شوند.

$$\text{Min } f(N, AvgD_{2e}, AvgLi, AvgEc) \quad (20)$$

$$f(N, AvgD_{e2e}, AvgLi, AvgEc) = \frac{AvgD_{e2e}}{WD_{e2e}} + \frac{AvgLi}{WLi} + \frac{AvgEc}{WEc} \quad (21)$$

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 می‌شود. از طرف دیگر $O\left(T \times \frac{n}{4} \times d\right)$ 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.

$$\begin{aligned} O(MHOA) &= O(\text{problem definition}) + O(\text{initialization}) \\ &\quad + O(\text{function evaluation}) + O(\text{Updating MHOA}) \\ &= O\left(1 + n + Tn + T\frac{n}{2}d + T\frac{n}{4}d + T\frac{n}{4}d\right) \cong O(Tnd + Tn + n) \end{aligned} \quad (22)$$

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 کاملاً 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 پیاده‌سازی شبکه‌ی پویا، ترافیک شبکه با بسته‌های با سائز 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 می شود که کدام بسته ها باید در شبکه باشند و بر اساس تعداد و اندازه ی بسته ها ترافیک تعیین می شود.

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

جدول ۲: پارامترهای شبیه سازی

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

جدول ۳: پارامترهای استفاده شده در MHOA

در این بخش، ابتدا کارایی الگوریتم MHOA از لحاظ میزان همگرایی بررسی می شود. نتایج نشان می دهد که الگوریتم MHOA نسبت به دیگر الگوریتم های فراابتکاری بهتر عمل می کند. در مرحله ی بعد MHOA برای حل مسئله ی CPP استفاده می شود 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٪ می شود.

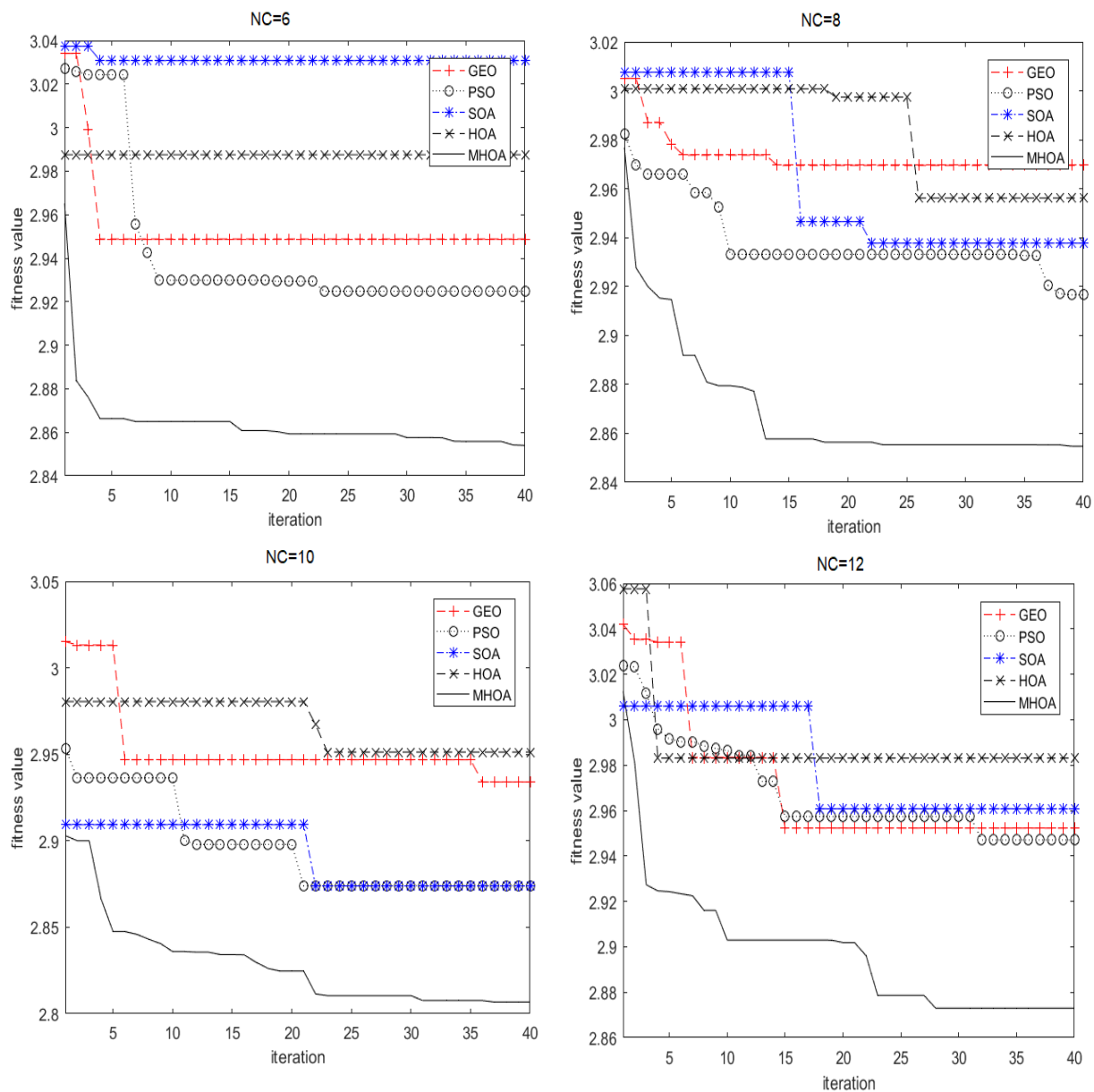
۵-۱- نتایج آنالیزها

این بخش شامل یک مجموعه جامع از نتایج تجربی است که به ارزیابی عملکرد الگوریتم پیشنهادی می پردازد. نتایج هر آزمایش به صورت جداگانه در ادامه ارائه می شوند. در این مرحله، توانایی MHOA در بهبود همگرایی با چهار الگوریتم فرا ابتکاری مطرح شامل الگوریتم بهینه سازی عقاب طلایی¹ (GEO) [39]، الگوریتم ازدحام ذرات² (PSO) [40]، الگوریتم مرغ دریایی³ (SOA) [41] و HOA مقایسه شده اند. 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 خروجی های به کار می رود. در این نمودار پراکندگی کم در نتایج نشان دهنده عملکرد و پایداری بهتر الگوریتم است. از نمودار همگرایی برای تجزیه و تحلیل نرخ همگرایی الگوریتم در به دست آوردن نتایج استفاده می شود. برای سنجش کارایی الگوریتم در حل CPP الگوریتم با چهار روش موفق، [CCPGWO[23]، GEWO[35]، PHCPA[14] که در بخش ۲ شرح داده شده اند مقایسه می شود. برای تعیین کارایی روش MHOA ارزیابی ها بر روی سه شبکه ی واقعی از دیتاست واقعی زو شامل Aarnet، Bics و Colt انجام شده است. شکل های ۶ و ۷ عملکرد MHOA را از نظر نرخ همگرایی و آزمون ANOVA در شبکه Aarnet نشان می دهند. نتایج نشان می دهند که MHOA نتایج بهتری نسبت به الگوریتم های دیگر دارد. آزمایش با تعداد کنترل کننده های مختلف، ۶، ۸، ۱۰ و ۱۲ انجام شده است. نمودارهای ANOVA نشان می دهند که MHOA به طور مداوم نتایج پایدار با کمترین انحراف معیار نسبت به سایر الگوریتم ها تولید می کند. علاوه بر این، MHOA از ابتدا با عملکردی قوی، بهترین همگرایی را نسبت به روش های دیگر دارد.

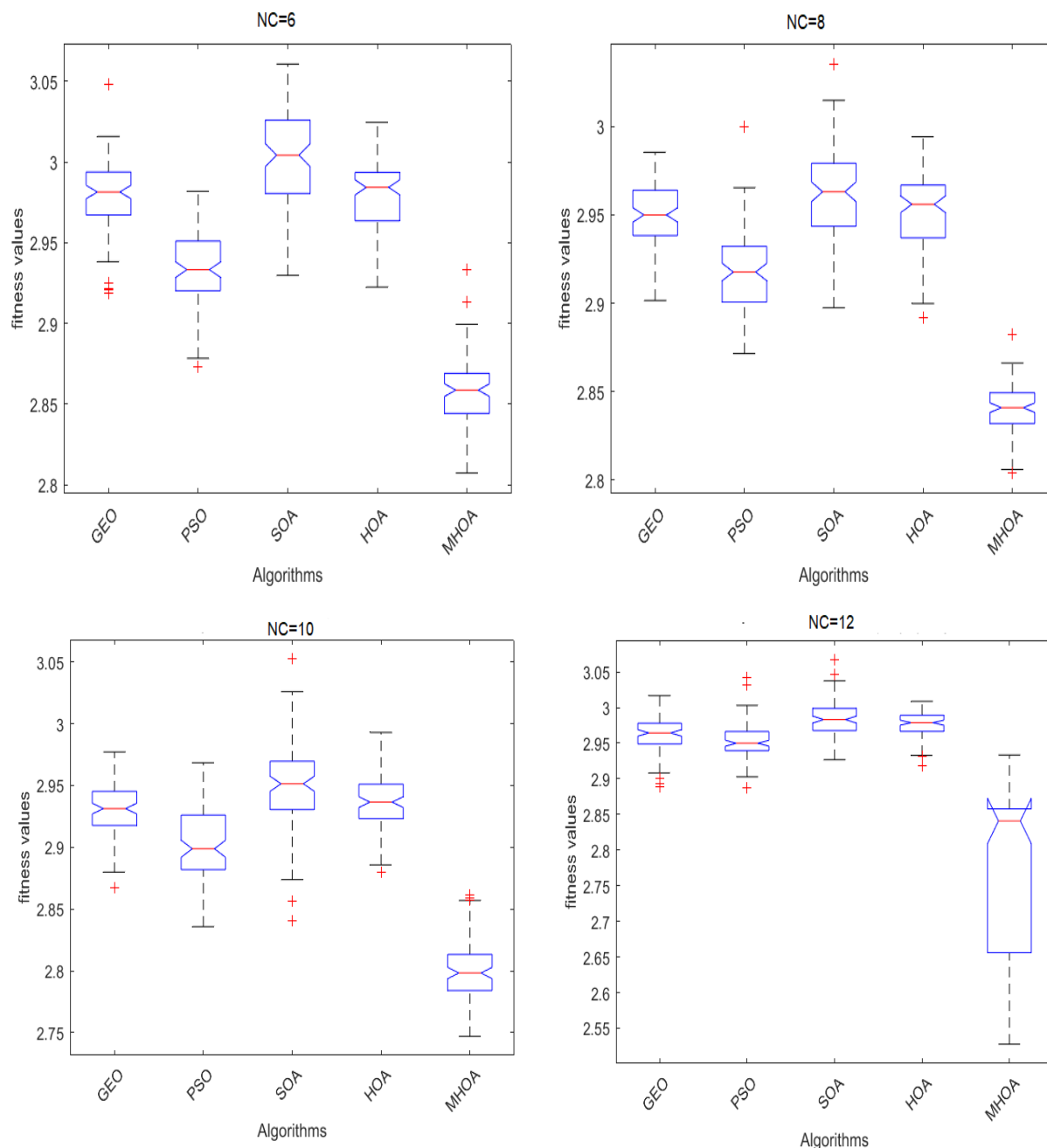
¹ Golden Eagle Optimization

² Partial swarm Optimization

³ Seagull optimization algorithm

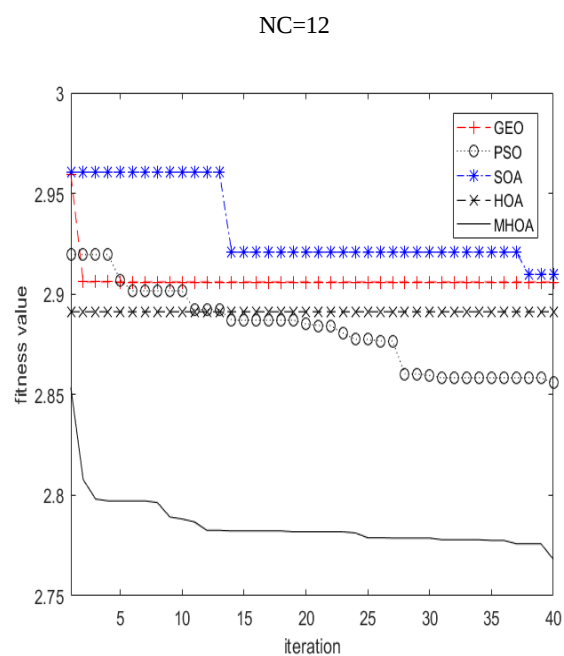
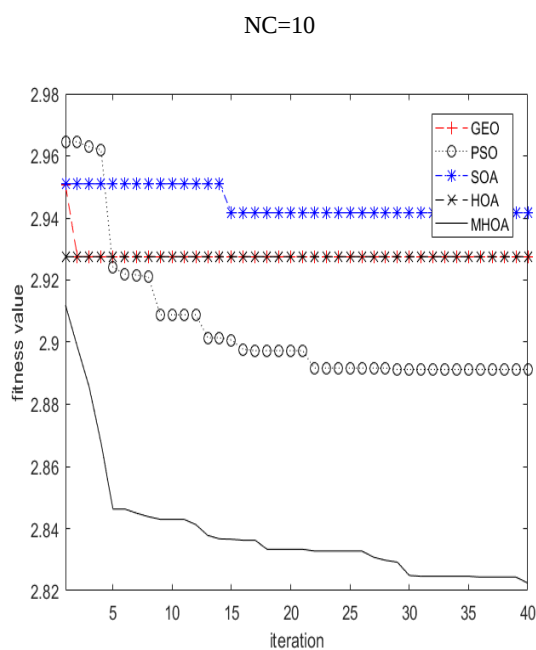
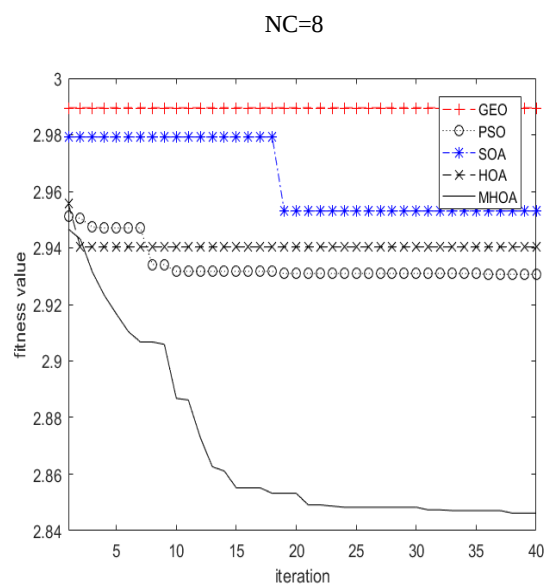
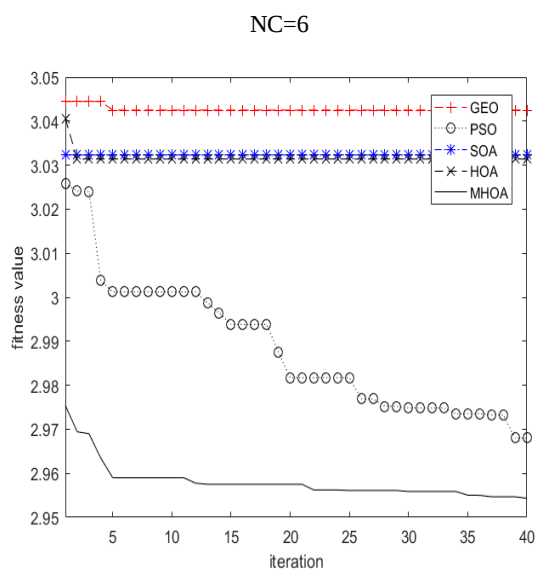


شکل ۶: نرخ همگرایی MHOA در شبکه‌ی Aarnet با تعداد کنترل‌کننده‌های متفاوت NC=6,8,10,12



شکل ۷: تست ANOVA برای MHOA در شبکه ی Aarnet با تعداد کنترل کننده های متفاوت NC=6,8,10,12

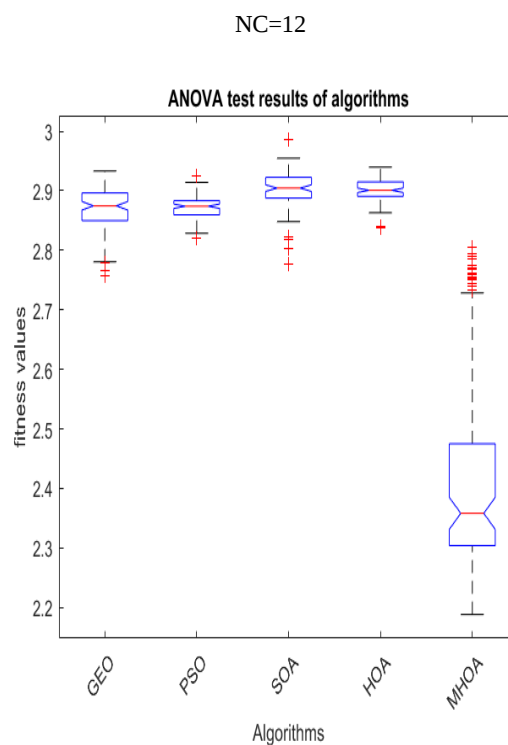
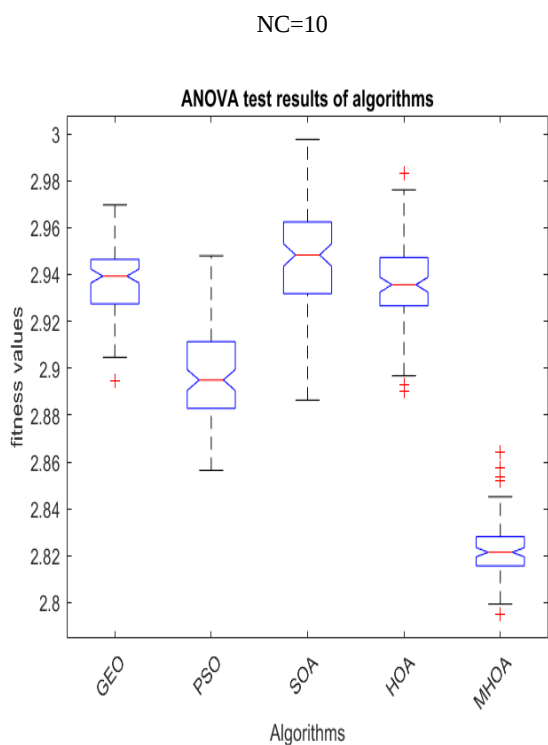
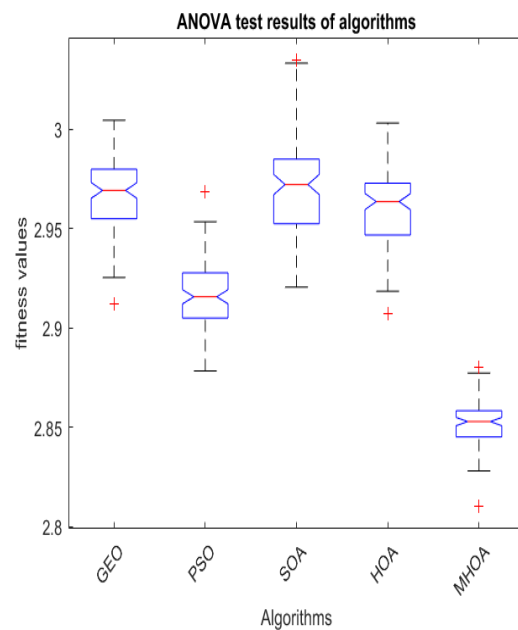
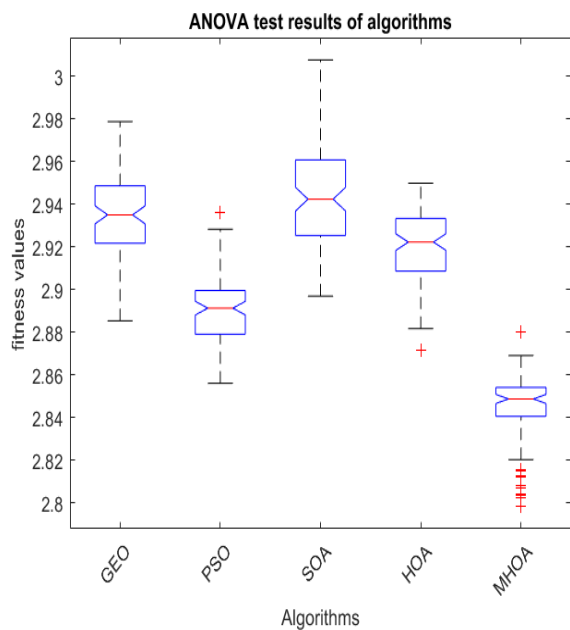
تصاویر ۸ و ۹ عملکرد MHOA را از نظر نرخ همگرایی و نتایج آزمون ANOVA بر روی شبکه Bic نشان می دهند. یافته ها نشان می دهند که MHOA به طور مداوم از الگوریتم های دیگر پیشی گرفته است. آزمایش ها با تعداد کنترل کننده های متفاوت ۶، ۸، ۱۰ و ۱۲، انجام شده اند. نمودارهای Anova نشان می دهد که MHOA با کمترین انحراف معیار نسبت به الگوریتم های دیگر نتایج پایداری تولید می کند.



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

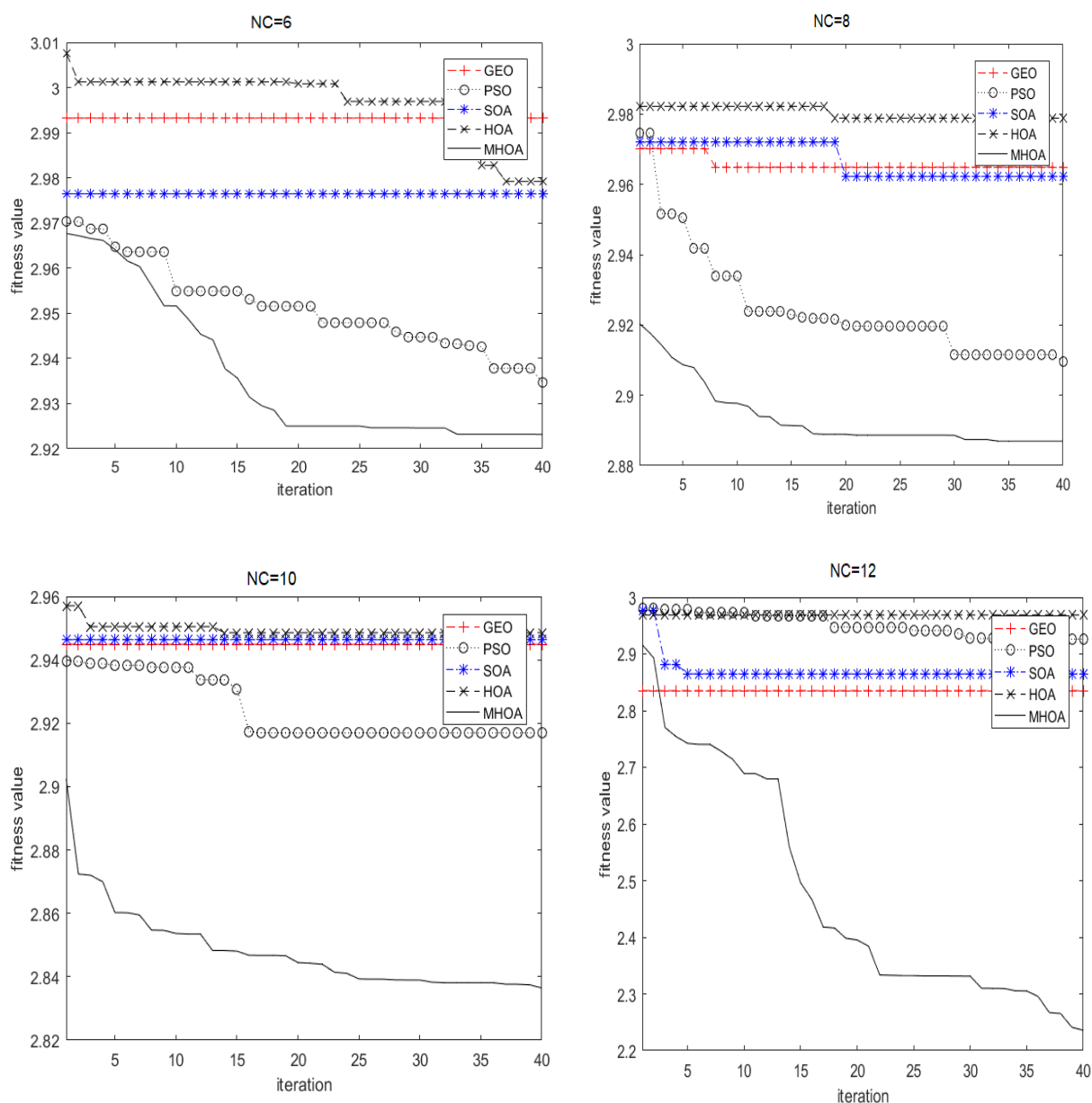
NC=6

NC=8

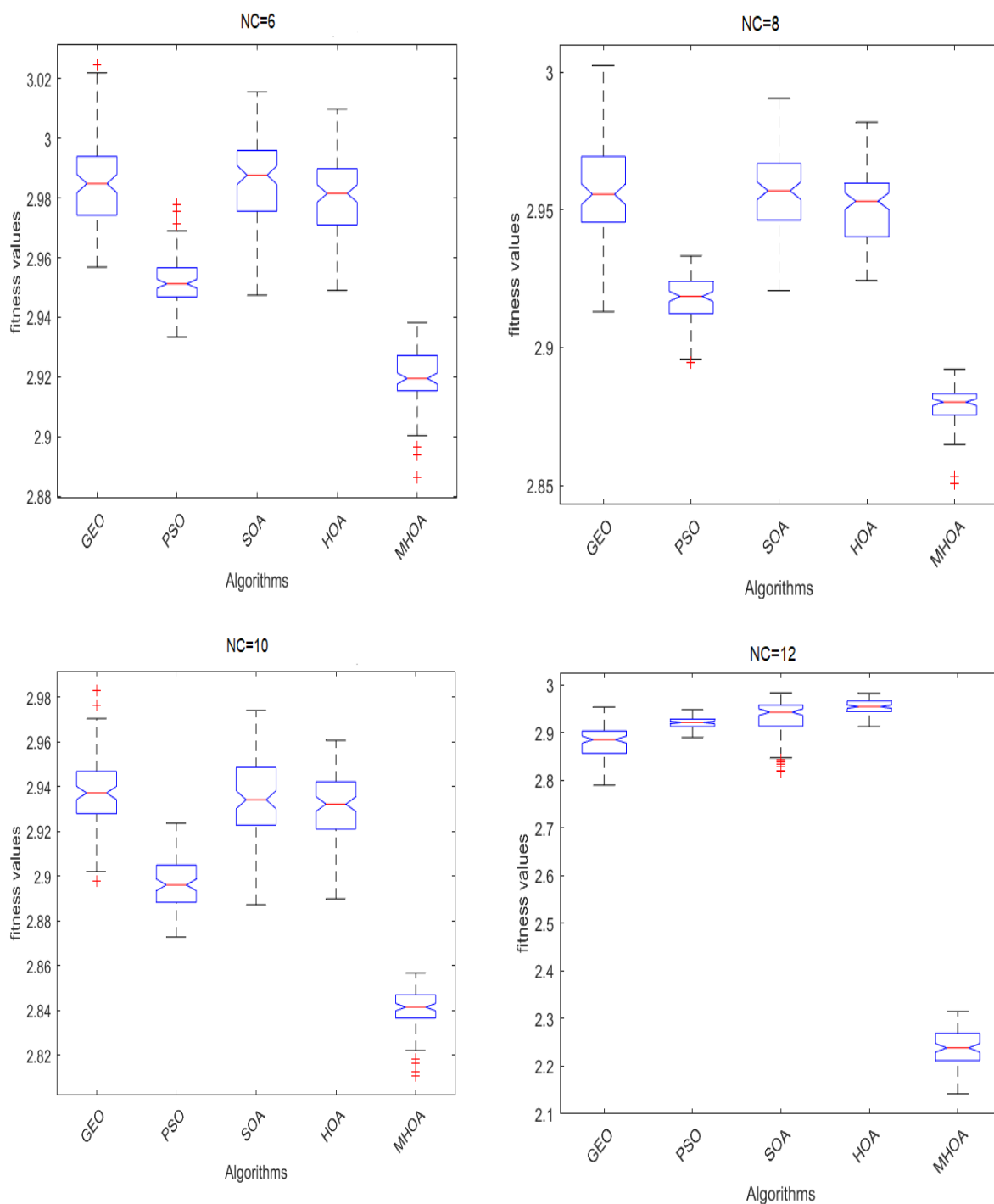


شکل ۹: تست ANOVA برای MHOA در شبکه ی Bics با تعداد کنترل کننده های متفاوت NC=6,8,10,12

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

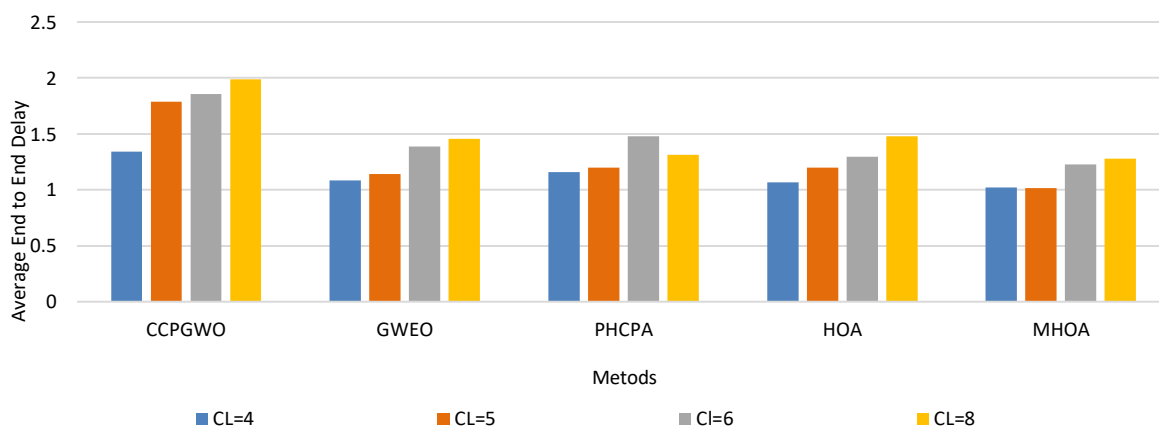


شکل ۱۱: تست ANOVA برای MHOA در شبکه ی Colt با تعداد کنترل کننده های متفاوت NC=6,8,10,12

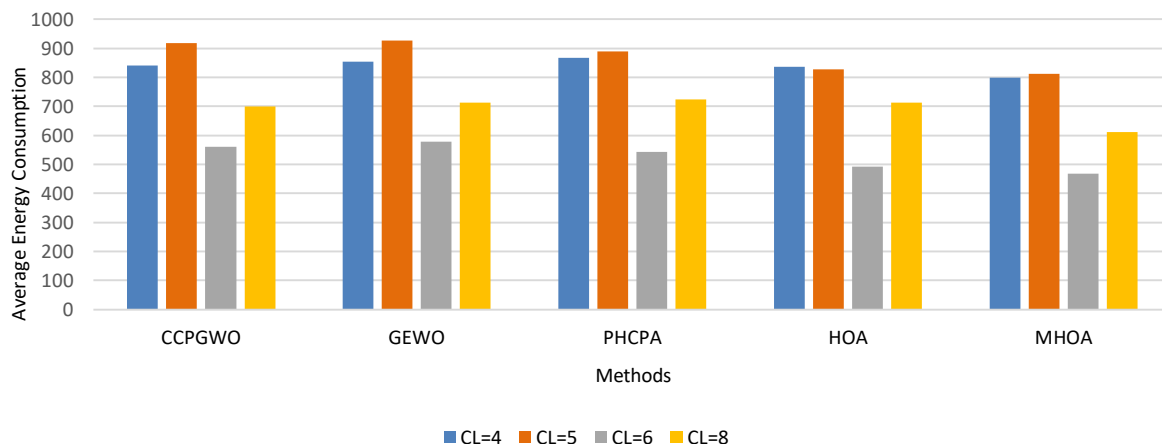
۵-۲- ارزیابی MHOA در پیدا کردن مکان بهینه کنترل کننده در داخل هر خوشه

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، ارزیابی‌ها 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 [23]CCPGWO، [35]GEWO، [14]PHCPA و HOA نشان داده شود. با پیدا کردن تعداد بهینه کنترل‌کننده‌ها و انتساب سوئیچ‌ها 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.

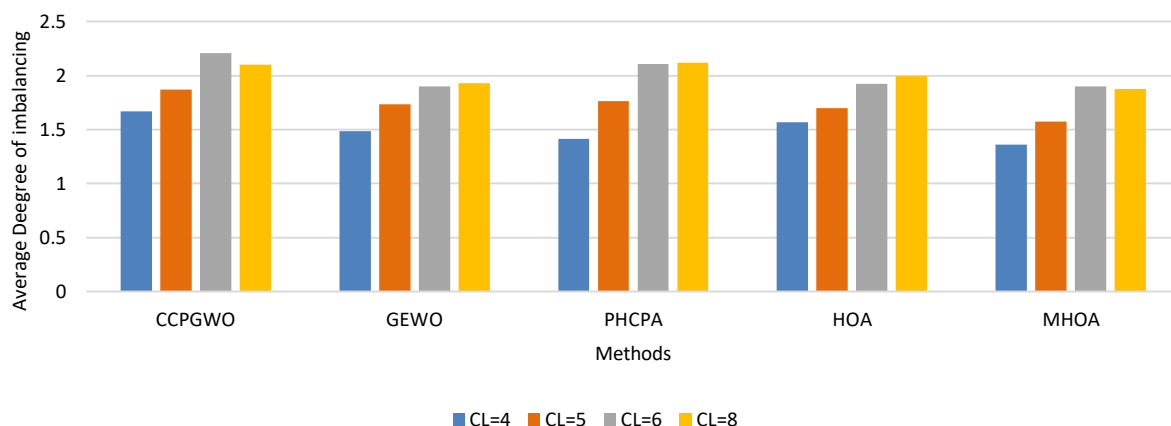
شکل ۱۲ تحلیل مقایسه‌ای از الگوریتم MHOA نسبت به الگوریتم‌های CCPGW، GEWO، PHCPA و HOA را نشان می‌دهد. یافته‌ها برتری عملکرد الگوریتم MHOA در دستیابی به توازن بار، کاهش مصرف انرژی و کمینه کردن تاخیر انتها به انتها نسبت به سایر الگوریتم‌ها را بیان می‌کنند.



(الف)



(ب)



(ج)

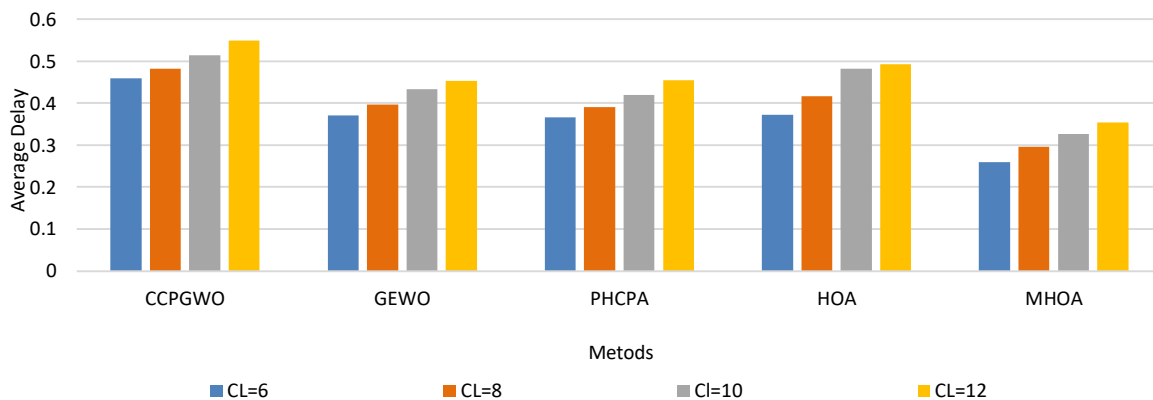
شکل ۱۲: نمودار میله ای نتایج به دست آمده بر روی شبکه Aarnet. الف) میانگین تاخیر انتها به انتها (ب) میانگین مصرف انرژی ج) میانگین عدم تعادل بار

به همین ترتیب، نتایج آماری الگوریتم ها در جدول ۴ خلاصه شده است، که درصد بهبودهای MHOA نسبت به سایر الگوریتم ها در شبکه Aarnet را نشان می دهد. 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 می نماید. به طور کلی MHOA، ۱۲٪/۳٪ در مصرف انرژی نسبت به سایر الگوریتم ها بهتر عمل می کند. علاوه بر این، در بهبود تاخیر انتها به انتها در شبکه ی Aarnet، MHOA به طور میانگین، ۵۳٫۷۹۹۶۳٪ نسبت به CCPGWO، ۱۱٫۵۶۰۲۴٪ نسبت به GEWO، ۱۳٫۷۶۰٪ نسبت به PHCPA، و ۱۰٫۹۸٪ نسبت به HOA بهتر عمل می کند. نتایج نشان می دهند که MHOA تاخیر انتها به انتها را ۲۲٫۵۲٪ به طور میانگین بهبود می دهد. MHOA با موفقیت عدم توازن بار را

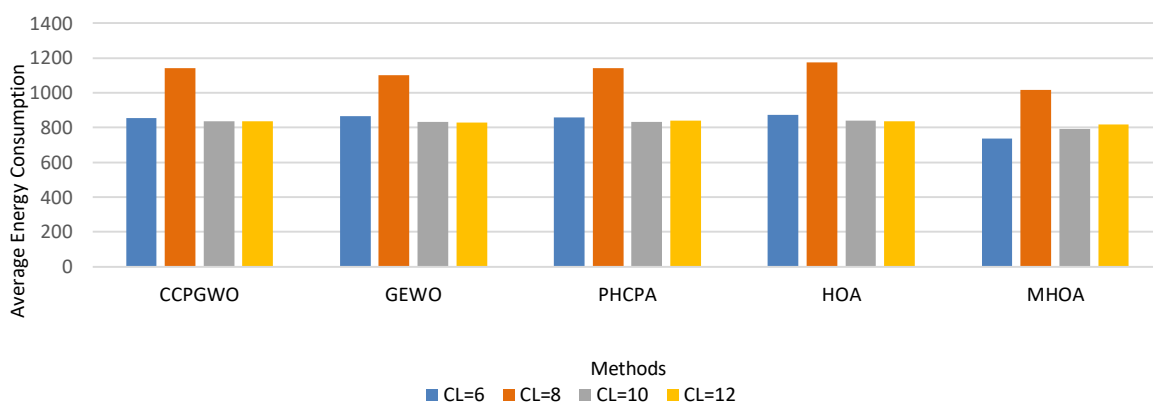
مدیریت می‌کند و به‌طور میانگین ۹,۲۲٪ بهتر از سایر الگوریتم‌های مورد بررسی عمل می‌کند. به‌طور خاص، این الگوریتم در مدیریت عدم توازن بار با بهترین عملکرد ۱۶,۰۴٪ نسبت به CCPGWO، ۶,۸۶٪ نسبت به GEWO، ۸,۴۷٪ نسبت به PHCPA و ۳۶,۳۳٪ نسبت به HOA بهتر عمل می‌کند.

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

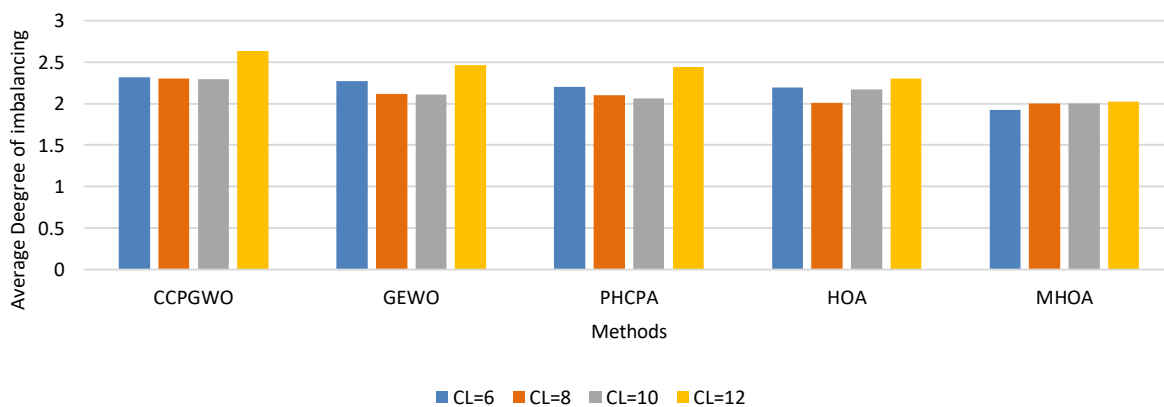
جدول ۴: مقایسه آماری الگوریتم‌ها در Aarnet با ۳۰ اجرای متفاوت



(الف)



(ب)



(ج)

شکل ۱۳: نمودار میله ای نتایج به دست آمده 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 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، ۹۷،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 الگوریتم‌های CCPGWO، GEWO، PHCPA و HOA در شبکه Colt نشان می‌دهد. ارزیابی با تعداد متفاوتی از کنترل‌کننده‌ها (۶، ۸، ۱۲ و ۱۶) انجام می‌شود. 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 می‌دهد که الگوریتم MHOA به‌طور مداوم عدم تعادل بار را به حداقل می‌رساند، و در عین حال تاخیر انتها به انتها و مصرف انرژی در شبکه را کاهش می‌دهد.

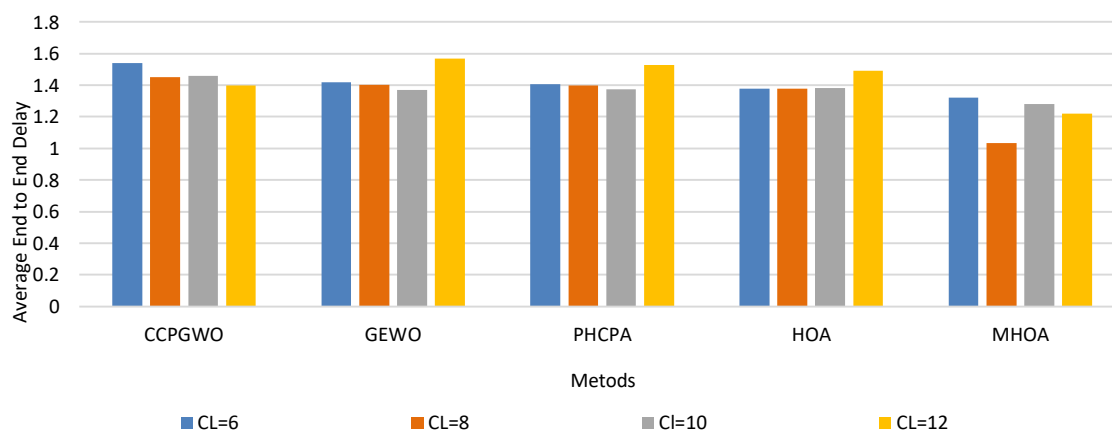
بر طبق جدول ۵، نتایج نشان می‌دهند که MHOA از نظر تاخیر انتها به انتها CCPGWO، GEWO، PHCPA و HOA به ترتیب ۱۹/۶۶٪، ۱۷/۴۹٪، ۱۵/۶۰٪ و ۱۴/۳۷٪ با میانگین کاهش ۱۶/۷۸٪ عملکرد بهتری بر روی شبکه‌ی Colt دارد. علاوه بر این، MHOA در مقایسه با چهار الگوریتم دیگر به‌طور متوسط ۴/۸۳٪ بهبود در مصرف انرژی نشان می‌دهد.

علاوه بر این، در مدیریت موثر عدم تعادل بار، MHOA در مقایسه با چهار الگوریتم دیگر به‌طور متوسط ۱۰،۱۰٪ بهبود قابل توجهی را نشان می‌دهد. 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.

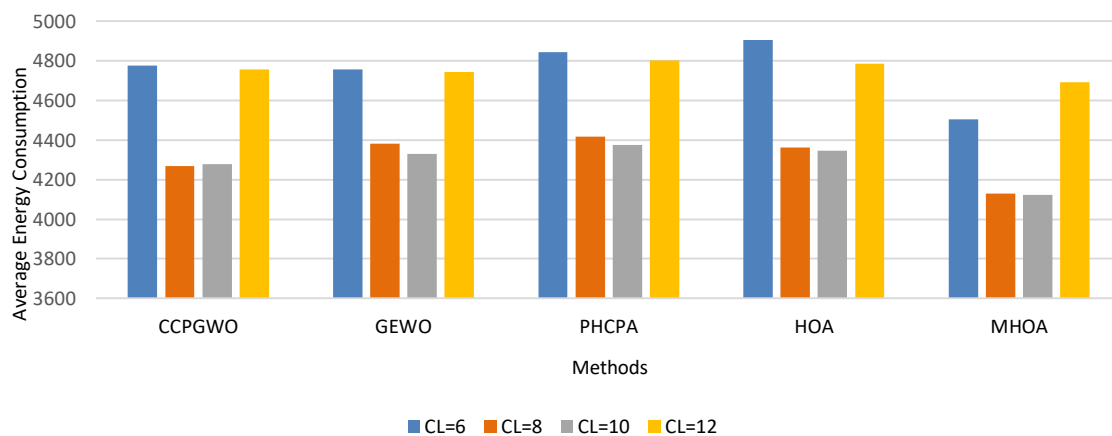
درصد بهبود	HOA	PHCPA	GEWO	CCPGWO	
میانگین تعادل بار					
۱۱/۱۵	۸/۵۹	۸/۸۳	۱۲/۳۳	۱۴/۸۳	CI=۶
۶/۵۹	۰/۴۳	۵/۱۶	۵/۸۱	۱۴/۹۵	CI=۸
۷/۸۸	۸/۵۶	۳/۰۴	۵/۱۸	۱۴/۷۳	CI=۱۰
۱۵/۸۹	۸/۲۹	۱۴/۹۲	۱۶/۱۰	۲۴/۲۴	CI=۱۲

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

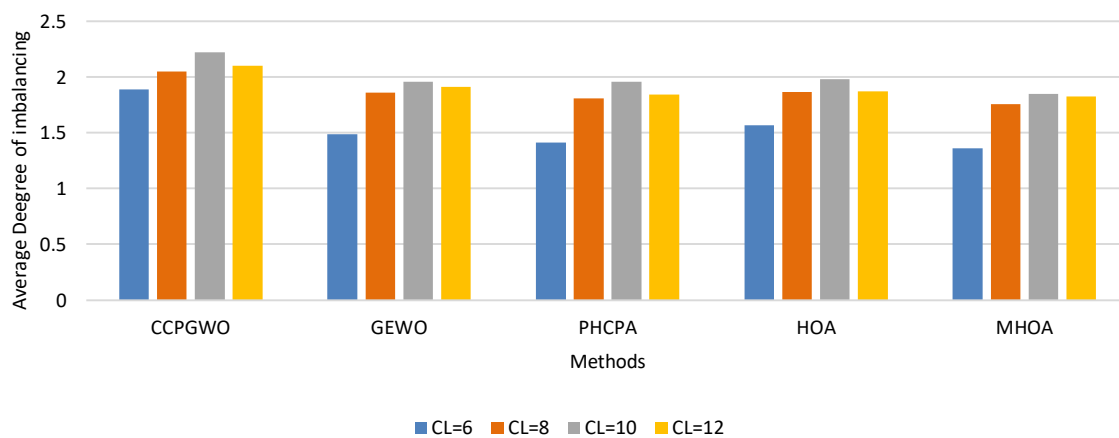
جدول ۵: مقایسه آماری الگوریتم ها در Bics با ۳۰ اجرای متفاوت



(الف)



(ب)



(ج)

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

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

جدول ۶: مقایسه آماری الگوریتم ها در Colt با ۳۰ اجرای متفاوت

نتایج نشان دهنده 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 می کند. 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 می گیرد. از نظر مصرف انرژی، MHOA به ترتیب ۸/۳۰، ۹/۸۱، ۹/۳۸ درصد و ۶/۲۲ درصد در مقایسه با CCPGWO، GEWO، PHCPA و HOA برتری دارد. به طور خلاصه، نتایج نشان می دهد که MHOA به طور 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 می دهد.

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 می‌دهد. با این حال، الگوریتم به منابع CPU و ظرفیت ذخیره‌سازی 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, 2۰۱۲.
- [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, 2011.
- [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.