E-ISSN: | P-ISSN:

Management: Modeling, Analysis and Application

www.mma.reapress.com

Manag. Model. Anal. Appl. Vol. 2, No. 3 (2025) 178–188.

Paper Type: Original Article

Providing an Efficient Kernel Search-Based Algorithm for Controller Placement in SDN Networks

Ali Abdi Seyedkolaei*

Department of Computer Engineering, Faculty of Engineering and Technology, University of Mazandaran, Babolsar, Iran; a.abdi@umz.ac.ir.

Citation:

Abdi Seyedkolaei, A. (2025). Providing an efficient kernel search-based algorithm for controller placement in SDN networks. *Management: Modeling, analysis and applications, 2*(3), 178-188.

Received: 10/01/2025 Reviewed: 27/03/2025 Revised: 17/04/2025 Accepted: 23/05/2025

Abstract

Purpose: This paper addresses the controller placement problem in Software-Defined Networks (SDN), which is a complex NP-hard optimization problem. The main objective is to find the optimal number and locations of controllers to minimize network implementation costs.

Methodology: To solve this problem, the Kernel Search metaheuristic algorithm was employed. The algorithm's performance was tested on a set of network instances of varying sizes and compared with results from the standard CPLEX solver.

Findings: The proposed algorithm demonstrated decisive superiority over CPLEX in terms of execution time across all cases. Furthermore, in many medium and large instances, the algorithm was able to find solutions with lower costs.

Originality/Value: The most significant innovation of this research is the successful adaptation and application of the Kernel Search algorithm to solve the controller placement problem in SDN. This method provides a practical and efficient alternative to conventional approaches.

Keywords: Software-defined networking, Optimal controller placement, Kernel search.

Corresponding Author: a.abdi@umz.ac.ir https://doi.org/10.22105/mmaa.v2i3.99

مدیریت: مدلسازی، تحلیل و کاربرد

دوره ۲، شماره (۳)، (۱۴۰۴)، ۱۸۸-۱۷۸

www.mma.reapress.com

نوع مقاله: پژوهشی

ارایه یک الگوریتم کارآمد مبتنی بر جست‌وجوی Kernel برای مکان‌یابی کنترلر در شبکه‌های SDN

علی عبدی سید کلایی•

گروه مهندسی کامپیوتر، دانشکده مهندسی و فناوری، دانشگاه مازندران، بابلسر، ایران.

چکیده

هدف: این مقاله به مساله مکان‌یابی کنترلرها در شبکه‌های نرم‌افزار-محور[1] می‌پردازد که یک مساله بهینه‌سازی پیچیده و *NP-hard* است. هدف اصلی، یافتن تعداد و مکان بهینه کنترلرها برای حداقل‌سازی هزینه‌های پیاده‌سازی شبکه است.

روش‌شناسی پژوهش: برای حل این مساله، از الگوریتم فراهیاب جست‌وجوی کرنل استفاده شده است. عملکرد این الگوریتم روی مجموعه‌ای از نمونه‌های شبکه با اندازه‌های مختلف آزمایش و با نتایج حاصل از حل‌کننده استاندارد *CPLEX* مقایسه شده است.

یافته‌ها: الگوریتم پیشنهادی در تمامی موارد از نظر زمان اجرا برتری قاطعانه‌ای نسبت به *CPLEX* نشان داده است. علاوه‌بر این، در بسیاری از نمونه‌های متوسط و بزرگ، این الگوریتم توانسته است راه‌حل‌های با هزینه کم‌تری نیز پیدا کند.

اصالت/ارزش افزوده علمی: مهم‌ترین نوآوری این تحقیق، ارایه و سازگاری موفق الگوریتم جست‌وجوی کرنل برای حل مساله مکان‌یابی کنترلر در *SDN* است. این روش، یک جایگزین عملی و کارآمد برای روش‌های متعارف ارایه می‌دهد.

کلیدواژه‌ها: شبکه نرم‌افزار محور، مکان‌یابی بهینه کنترلر، جست‌وجوی کرنل.

۱- مقدمه

معماری تجهیزات ارتباطی شبکه از جمله سوئیچ‌ها و مسیریاب‌ها از دو بخش کنترل و داده تشکیل شده است [1]. بخش کنترل، اطلاعات لازم برای مسیریابی در شبکه را فراهم می‌کند. بخش داده نیز وظیفه انتقال بسته‌ها از درگاه ورودی به درگاه خروجی بر اساس اطلاعات درج‌شده در جدول مسیریابی خود را بر عهده دارد. در سال‌های اخیر محققان توجه خود را به جداسازی بخش کنترل از بخش داده معطوف کرده‌اند. در این جداسازی، بخش کنترل در سرور یا برنامه‌ای که کنترلر نام دارد قرار می‌گیرد. بخش داده نیز در سوئیچ یا مسیریاب باقی می‌ماند. جداسازی بخش کنترل از داده مزایای زیادی همچون قابلیت برنامه‌ریزی بیشتر سطح کنترل، امکان مجازی‌سازی شبکه، کاهش هزینه‌های عملیاتی، استقلال بیشتر شرکت‌های تولیدکننده تجهیزات شبکه و ... را به ارمغان آورده است. ایده‌ی فوق ظهور *SDN* را به‌دنبال داشته است [2]. پیدایش *SDN* توجه تعداد زیادی از دانشگاه‌ها و صنایع را به پیاده‌سازی آن در زیرساخت‌های ارتباطی خود معطوف ساخته است [3]. روی این شبکه‌ها، روش‌های پیکربندی، اغلب ساده‌تر و دقیق‌تر هستند و امکان بهره‌گیری بیشتر از زیرساخت‌های فیزیکی فراهم آمده است [4]. گروهی از اپراتورهای شبکه، ارایه‌دهندگان خدمات

[1] Software Defined Networks (SDN)

شبکه و تولیدکنندگان تجهیزات شبکه، سازمانی به نام بنیاد شبکه‌سازی باز[1] [5] ایجاد کردند تا به ترویج شبکه‌های *SDN* و ارایه یک پروتکل ارتباطی استاندارد برای این شبکه بپردازند.

در شبکه‌های *SDN*، کنترلر اغلب مسئول انتشار هر جریان در شبکه است که این عمل را با تخصیص جریان‌های ورودی به سوئیچ‌ها انجام می‌دهد [6]. این مهم، نقشی محوری به کنترلر بخشیده است چرا که به واسطه‌ی آن می‌توان دانش کاملی از شبکه را در بهینه‌سازی مدیریت جریان و پشتیبانی از نیاز کاربر به خدمت گرفت [7].

از جمله مسایل مهم تعریف‌شده روی شبکه‌های *SDN*، مساله‌ی مکان‌یابی کنترلر است. در این مساله، هدف تعیین مکان کنترلرها و تعداد مورد نیاز آن‌ها در شبکه است. به‌طوری که بتوان حداقل هزینه را برای مجموع هزینه‌های نصب کنترلر در مکان‌ها، هزینه‌های اتصال سوئیچ‌ها به کنترلرها و نیز هزینه‌های اتصال کنترلرها به یکدیگر صرف کرد [8]. از نظر پیچیدگی محاسباتی، این مساله به کلاس مسایل *NP*-تعلق دارد [9]. در این مقاله، به‌منظور حل مساله از الگوریتمی موسوم به جست‌وجوی کرنل [10] استفاده شده است.

ادامه این مقاله به شرح ذیل سازماندهی شده است: بخش ۲، پیشینه‌ی موجود از روش‌های حل مساله را مرور می‌کند. بخش ۳ به تشریح یک مدل برنامه‌ریزی صحیح استاندارد از مساله پرداخته است. بخش ۴، به معرفی الگوریتم پیشنهادی مبنی بر روش جست‌وجوی کرنل و بحث در ساختار آن اختصاص یافته است. بخش ۵ به تحلیل محاسباتی الگوریتم مذکور و بررسی عملکرد آن در مقایسه با یک حل‌کننده‌ی استاندارد مدل‌های برنامه‌ریزی صحیح، پرداخته است. در نهایت نتیجه‌گیری حاصل از محاسبات به‌دست آمده در بخش ۶ بیان شده است.

۲- تحقیقات مرتبط

در این بخش، پژوهش‌های انجام‌شده روی مساله مکان‌یابی کنترلر مرور می‌شود.

هلر و همکاران [8] دو معیار میانگین زمان تاخیر و تاخیر در بدترین حالت را برای انتخاب مکان مناسب کنترلر به‌کار گرفته‌اند. در این روش، ابتدا کوتاه‌ترین مسیر برای هر دو سوئیچ مشخص و سپس میانگین طول این مسیرها به‌عنوان میانگین زمان تاخیر و طول بزرگترین آن‌ها به‌عنوان تاخیر در بدترین حالت در نظر گرفته می‌شود.

شیائو و همکاران [11] ابتدا به‌کمک الگوریتمی موسوم به خوشه‌بندی طیفی، شبکه را به چندبخش تقسیم می‌کنند و سپس بهترین مکان برای نصب کنترلر در هر بخش مشخص می‌نمایند.

برخلاف دو پژوهش قبل، تحقیق انجام‌شده در [12] به قابلیت اطمینان در شبکه‌های *SDN* پرداخته است. در این تحقیق قابلیت اطمینان بیشتر در دریافت اطلاعات از سوئیچ‌ها در یک مکان، اولویت انتخاب آن برای نصب کنترلر را افزایش خواهد داد. در صورتی که سایر پارامترهای تاثیرگزار در مکان‌یابی کنترلر همچون، هزینه، تاخیر و تعادل بار در نظر گرفته نشده است.

تمرکز بر روی خواص گراف شبکه، ژانگ و همکاران [13] را برآن داشت تا به‌کمک یک الگوریتم تجزیه گراف مکان‌هایی برای نصب کنترلر پیشنهاد دهند که احتمال از دست دادن اتصال سوئیچ-کنترلر کمینه گردد.

اوبادیا و همکاران [14] مساله مکان‌یابی کنترلر را با اعمال توپولوژی درختی برای متصل‌کردن کنترلرها به یکدیگر در نظر گرفته‌اند. القای چنین توپولوژی به کنترلرها، سربار عملیاتی آن‌ها را کاهش خواهد داد.

یائو و همکاران [9] مساله مکان‌یابی کنترلر را در حضور کنترل‌هایی با ظرفیت محدود مورد مطالعه قرار داده‌اند.

---

[1] Open Networking Foundation (ONF)

در پژوهش ژانگ و دیگران [15] توپولوژی‌های متفاوتی را به‌منظور اتصال کنترلرها در نظر گرفته‌اند.

اخیرا، صالحی و همکاران در [16] به‌منظور حداقل‌کردن هزینه شبکه، مساله مکان‌یابی کنترلر را به‌صورت مجموع چند تابع هدف بیان کرده‌اند. آن‌ها استدلال کردند که مکان‌یابی کنترلر می‌بایست محدودیت‌های خاصی را به‌ویژه برای سطح کنترل برآورده کند. ایده آن‌ها به‌طور همزمان تعداد بهینه، مکان و نوع کنترلر و همچنین ارتباطات بین عناصر شبکه را تعیین کرده و برای حداقل‌کردن هزینه شبکه، محدودیت‌های مختلف در نظر گرفته شده است.

مدل برنامه‌ریزی ریاضی مطرح‌شده در [16] و حل دقیق آن که مبتنی بر استفاده از حل‌کننده‌های استاندارد مسایل برنامه‌ریزی ریاضی است، معیار مقایسه‌های انجام‌شده در این تحقیق خواهد بود.

۳- بیان دقیق مساله به‌کمک مدل‌بندی ریاضی

در این بخش مدل ریاضی مساله مکان‌یابی کنترلر، برگرفته از [16] به‌منظور بیان دقیق مساله شرح داده می‌شود. این مدل شامل مجموعه‌ی سوئیچ‌ها، مجموعه‌ی کنترلرها، مجموعه‌ی لینک‌ها و مجموعه‌ی مکان‌های ممکن برای نصب کنترلر است که به‌ترتیب با نمادهای S، C، L و P نمایش داده می‌شود.

در این مدل، شبکه به شکل یک گراف فرض شده است، هر گره از گراف مبین مکان قرارگرفتن یک سوئیچ یا کنترلر است. در حالت خاص، در یک گره سوئیچ و کنترلر می‌توانند کنار هم قرار گیرند. هر سوئیچ تنها به یکی از کنترلرها متصل است. با در نظر گرفتن محدودیت تعداد درگاه کنترلر، چندین سوئیچ می‌تواند توسط یک کنترلر مدیریت شود. همچنین در این گراف هر کنترلر می‌بایست با سایر کنترلرها در ارتباط مستقیم قرار گیرد. از این رو، توپولوژی القاشده به کنترلرها در این مساله از نوع مش کامل است.

برای هر سوئیچ $s \in S$، تعداد بسته‌های ارسال‌شده به کنترلر با پارامتر σ^s نشان داده می‌شود. هر کنترلر $c \in C$، دارای پارامترهای α^c، μ^c، k^c و φ^c است که به‌ترتیب بیانگر تعداد پورت، تعداد بسته، هزینه و تعداد کنترلرهای از نوع c می‌باشد. لینک $l \in L$، شامل دو پارامتر ω^l و Φ^l است که پهنای باند و هزینه‌ی نصب آن را بیان می‌کند. سایر پارامترهای نامنفی در مدل ریاضی مساله، مقادیر اندازه بسته، تاخیر شبکه و زمان پردازش بسته در کنترلر است که به‌ترتیب با نمادهای β، γ و δ نشان داده می‌شود. اکنون با توجه به تعریف مجموعه‌ها و پارامترها، متغیرهای تصمیم‌گیری مساله به شرح زیر است:

$$x_{cp} = \begin{cases} 1, & \text{اگر کنترلر } c \text{ در مکان } p \text{ نصب شده است،} \\ 0, & \text{در غیر این صورت،} \end{cases}$$

$$\mathrm{v}_{sp}^{l} = \begin{cases} 1, & \text{اگر لینک } l \text{ بین سوئیچ } s \text{ و کنترلر در مکان } p \text{ برقرار باشد،} \\ 0, & \text{در غیر این صورت،} \end{cases}$$

$$z_{pq}^{l} = \begin{cases} 1, & \text{اگر لینک } l \text{ بین مکان } p \text{ و مکان } q \text{ برقرار باشد،} \\ 0, & \text{در غیر این صورت،} \end{cases}$$

هدف در این مساله حداقل‌کردن هزینه پیاده‌سازی شبکه است. این هزینه‌ها شامل هزینه‌های نصب کنترلر در مکان‌ها، هزینه‌های اتصال سوئیچ‌ها به کنترلرها و نیز هزینه‌های اتصال کنترلرها به یکدیگر است که به‌ترتیب با نمادهای $C_c(x)$، $C_l(v)$ و $C_t(z)$ نشان داده می‌شوند. **معادله (۱)** تا **معادله (۳)** نیز نحوه محاسبه هر یک از این هزینه‌ها را بیان می‌کند. تابع *dist(a,b)* در **معادله (۲)** و **معادله (۳)** فاصله اقلیدسی بین مکان a و مکان b را به‌دست می‌دهد.

$$C_c(x) = \sum_{c \in C} k^c \sum_{p \in P} x_{cp}, \quad (۱)$$

$$C_l(v) = \sum_{l \in L} \Phi^l \sum\nolimits_{s \in S} \sum_{p \in P} dist(s,p) v_{sp}^l, \quad (۲)$$

$$C_t(z) = \sum_{l \in L} \Phi^l \sum_{\substack{q \in p \\ q<p}} \sum_{p \in P} dist(p,q) z_{pq}^l. \quad (۳)$$

در ادامه، تابع هدف مساله به‌همراه قیدهای آن بیان می‌شوند.

$$Minimize \left(C_c(x) + C_l(v) + C_t(z)\right), \quad (۴)$$

$$\sum_{c \in C} x_{cp} \leq 1, \quad For\ all\ p \in P, \quad (۵)$$

$$\sum_{q \in P} \sum_{l \in L} \left(z_{pq}^l + z_{qp}^l\right) + \sum_{s \in S} \sum_{l \in L} v_{sp}^l \leq \sum_{c \in C} \alpha^c x_{cp}, \quad For\ all\ p \in P, \quad (۶)$$

$$\sum_{l \in L} \sum_{p \in P} v_{sp}^l = 1, \quad For\ all\ s \in S, \quad (۷)$$

$$\sum_{l \in L} \sum_{s \in S} \sigma^s v_{sp}^l \leq \sum_{c \in C} \mu^c x_{cp}, \quad For\ all\ p \in P, \quad (۸)$$

$$\sum_{p \in P} x_{cp} \leq \varphi^c, \quad For\ all\ c \in C, \quad (۹)$$

$$g(\sigma^s, \beta) \leq \sum_{l \in L} f(w^l) v_{sp}^l, \quad For\ all\ s \in S, \quad For\ all\ p \in P, \quad (۱۰)$$

$$2Tran(v) + 2Prop(x,v) + Proc(x) \leq \gamma, \quad (۱۱)$$

$$\sum_{c \in C} x_{cq} + \sum_{c \in C} x_{cp} \leq \sum_{l \in L} z_{pq}^l + 1, \quad (q < p, For\ all\ q \in P, For\ all\ p \in P), \quad (۱۲)$$

$$x_{cp} \in \{0,1\}, \quad For\ all\ c \in C, \quad For\ all\ p \in P, \quad (۱۳)$$

$$v_{sp}^l \in \{0,1\} \quad For\ all\ l \in L, \quad For\ all\ s \in S, \quad For\ all\ p \in P, \quad (۱۴)$$

$$z_{pq}^l \in \{0,1\} \quad For\ all\ l \in L, \quad For\ all\ p \in P, \quad For\ all\ q \in P. \quad (۱۵)$$

قیدهای (۵) و **(۶)** محدودیت تعداد استفاده از کنترلر c در مکان p و محدودیت تعداد درگاه کنترلر c را بیان می‌کنند. **قید (۷)** بیانگر محدودیت تعداد استفاده از لینک l بین سوئیچ s و کنترلر c است. **قیدهای (۸)** و **(۹)** نشان‌دهنده محدودیت تعداد پردازش بسته توسط کنترلر c و محدودیت تعداد موجودی کنترلر c می‌باشد. **قید (۱۰)** بیانگر محدودیت پهنای باند لینک l است. تابع $g(a,b)$ در **قید (۱۰)** که دارای دو آرگومان ورودی a و b است، تعداد a بسته به طول b بایت را برحسب بایت در هر ثانیه به‌دست می‌آورد. تابع $f(a)$ نیز که مقدار a در آن برحسب مگابایت یا گیگابایت است را برحسب بایت محاسبه می‌کند. **قیدهای (۱۱)** و **(۱۲)** نشان‌دهنده محدودیت تاخیر کلی شبکه است؛ یعنی مجموع تاخیر انتقال توسط سوئیچ s، تاخیر انتشار بین کنترلر c و سوئیچ s و تاخیر پردازش کنترلر c که به‌ترتیب با نمادهای $Tran(v), Prop(x,v), Proc(x)$ نشان داده شده است می‌بایست از تاخیر کلی شبکه کمتر یا مساوی باشد. **قید (۱۲)** محدودیت توپولوژی شبکه را نشان می‌دهد؛ به این معنی که توپولوژی بین گره‌های شبکه می‌بایست از نوع مش‌کامل باشد. در نهایت **قید (۱۳)** تا **قید (۱۵)** نشان‌دهنده مقادیر باینری است که متغیرهای تصمیم‌گیری

می‌توانند داشته باشند. هر تخصیص باینری به متغیرهای مساله که **قید (۵)** تا **قید (۱۲)** را برآورد یک جواب شدنی مساله، به اختصار جواب مساله، نامیده می‌شود.

۴- الگوریتم جست‌وجوی کرنل

جست‌وجوی کرنل یک الگوریتم ابتکاری برای حل مسایل برنامه‌ریزی خطی عدد صحیح مختلط[1] با استفاده از متغییرهای باینری است. در این بخش توصیفی از جست‌وجوی کرنل بیان می‌شود که علاوه‌بر به‌کارگیری در این مقاله برای هر مساله برنامه‌ریزی خطی عدد صحیح دودویی[2] نیز قابل استفاده می‌باشد. در مسایل *BILP* از متغیرهای باینری (به‌عنوان مثال، x، v و z) استفاده می‌شود. یک متغیر باینری امیدوارکننده است اگر مقدار یک در راه حل بهینه از مساله اصلی بگیرد. مجموعه تمام متغییرهای امیدوارکننده به‌عنوان کرنل در نظر گرفته می‌شود. کرنل به دو نوع تقسیم می‌شود: کرنل جزیی و کرنل عمومی. کرنل جزیی شامل مجموعه متغیرهای امیدوارکننده متعلق به یک مجموعه متغییر داده شده (برای مثال x) است و کرنل عمومی شامل اجتماع تمام کرنل‌های جزیی مجموعه متغیرها است. هر دو کرنل جزیی و عمومی در طول الگوریتم جست‌وجوی کرنل متفاوت می‌باشند.

در الگوریتم جست‌وجوی کرنل، ابتدا *linear relaxation* متناظر با مساله اصلی حل می‌شود. سپس کرنل‌های جزیی اولیه برای هر مجموعه متغیر تشکیل می‌شود. اجتماع کرنل‌های جزیی اولیه تمام مجموعه متغیرها کرنل عمومی اولیه را به‌دست می آورد. متغیرهای هر مجموعه که در کرنل جزیی اولیه وجود نداشته باشد به دسته‌هایی به نام باکت‌های جزیی دسته‌بندی می‌شوند.

قسمت اصلی الگوریتم، حل دنباله‌ای از مسایل محدود شده است. مساله *BILP* در ابتدا به دنباله‌ای از کرنل عمومی اولیه محدود می‌شود. سپس هر زیردنباله از مساله *BILP* به کرنل عمومی فعلی محدود می‌شود که توسط اجتماع کرنل‌های جزیی فعلی برای تمام مجموعه متغییرها و باکت جزیی برای هر متغییر تشکیل می‌شود. کرنل جزیی فعلی متغیرها بر اساس کرنل جزیی قبلی که تعدادی از متغیرهای آن به‌دلیل امیدوارکننده نبودن حذف و برخی متغیرها به‌دلیل امیدوارکننده بودن اضافه می‌شود به‌دست می آید.

هر راه حل امکان‌پذیر مساله *BILP* محدود شده، یک راه حل اکتشافی از مساله اصلی است و یک حد بالایی را روی راه حل بهینه خود فراهم می‌کند که به‌عنوان یک مقدار *off cut* برای تمام مسایل *BILP* محدودشده استفاده می‌شود. جست‌وجوی کرنل زمانی متوقف می‌شود که یکی از شرایط توقف برقرار گردد. ساختار عمومی الگوریتم جست‌وجوی کرنل برای مسایل *BILP* به شرح ذیل است:

الگوریتم ۱- ساختار عمومی الگوریتم جست‌وجوی کرنل برای مسایل BILP.
Algorithm 1- General structure of the kernel search algorithm for BILP problems.

۱. حل linear relaxation مساله
۲. برای هر مجموعه از متغییرها، کرنل جزیی اولیه و باکت‌های جزیی تعیین شود
۳. کرنل عمومی اولیه ایجاد شود
۴. مساله BILP روی کرنل عمومی اولیه حل شود
۵. مرحله ۱-۵ تا مرحله ۳-۵ تا زمان برقراری شرط توقف تکرار گردد
۱-۵. کرنلی عمومی فعلی ساخته شود
۲-۵. مساله BILP روی کرنل عمومی فعلی به همراه یک باکت جزیی برای هر مجموعه از متغیرها حل شود
۳-۵. کرنل جزیی هر مجموعه متغیرها به‌روزرسانی گردد

کرنل جزیی اولیه بر اساس حل بهینه *linear relaxation* انتخاب می‌شود. باکت‌ها با استفاده از ضریب هزینه‌های کاهش‌یافته، از کوچک‌ترین به بزرگ‌ترین ساخته می‌شوند. کرنل جزیی فعلی به‌صورت زیر به‌روزرسانی می‌گردد:

[1] Mixed Integer Programming (MILP)
[2] Binary Integer Linear Programming (BILP)

یک متغیر متعلق به باکت جزیی که در راه حل بهینه مساله *BILP* مقدار مثبت می‌گیرد به متغیر امیداورکننده تبدیل می‌شود. در مقابل، یک متغیر امیدوارکننده متعلق به کرنل جزیی فعلی که در راه حل بهینه مساله *BILP* مقدار مثبت نمی‌گیرد به‌عنوان متغییر امیدوارکننده در نظر گرفته نمی‌شود.

با توجه به مدل ریاضی بیان‌شده، مساله *BILP* به‌صورت *BILP(X,V,Z)* نشان داده می‌شود به‌طوری که X شامل مجموعه متغیرهای نشان‌دهنده نصب کنترلر در مکان p، V شامل مجموعه متغیرهای نشان‌دهنده اتصال بین سوئیچ s و کنترلر در مکان p و Z شامل مجموعه متغیرهای نشان‌دهنده اتصال کنترلر در مکان p با کنترلر در مکان q است. به بیان دیگر به‌صورت ریاضی:

$$X = \{X_{cp}\}, \quad For\ all\ p \in P, \quad For\ all\ c \in C.$$
$$V = \{V_{sp}\}, \quad For\ all\ p \in P, \quad For\ all\ s \in S.$$
$$Z = \{Z_{pq}\}, \quad For\ all\ p \in P, \quad For\ all\ q \in P.$$

همچنین *linear relaxation*، *BILP(X,V,Z)* به‌صورت *LP(X,V,Z)* نمایش داده می‌شود. در الگوریتم جست‌وجوی کرنل، کرنل جزیی (اولیه و فعلی) برای مجموعه متغیرهای V و Z باید با کرنل جزیی (اولیه و فعلی) برای مجموعه متغیرهای X سازگار باشد. همین استدلال نیز برای باکت‌های جزیی به‌کار گرفته می‌شود.

کرنل جزیی برای متغیرهای X، V و Z به‌ترتیب به‌صورت *K(X)*، *K(V)* و *K(Z)* نمایش داده می‌شود. هر X_{cp} در *K(X)* تنها به زیرمجموعه‌ای از تمام سوئیچ‌ها خدمات می‌دهد. *K(V)* شامل هیچ متغیر V_{sp} نمی‌شود اگر X_{cp} در *K(X)* نباشد. همچنین، *K(Z)* شامل هیچ متغیر Z_{pq} نمی‌شود اگر X_{cp} در *K(X)* نباشد. کرنل عمومی با نماد K نمایش داده می‌شود که از اجتماع کرنل‌های جزیی *K(X)*، *K(V)* و *K(Z)* به‌دست می آید.

الگوریتم جست‌وجوی کرنل شامل دو فاز است. در فاز اول که به‌نام فاز مقداردهی اولیه است، *LP(X,V,Z)* مساله حل می‌گردد. (X^{LP}, V^{LP}, Z^{LP}) مقدار بهینه بعد از حل مساله را نشان می‌دهد. اگر مقدار (X^{LP}, V^{LP}, Z^{LP}) صحیح باشد، آن‌گاه مقدار به‌دست آمده یک مقدار بهینه برای مساله اصلی است و جست‌وجوی کرنل متوقف می‌شود. در غیر این صورت، حل بهینه *LP(X,V,Z)* یک باند پایین روی مقدار بهینه است که می‌تواند برای شناسایی متغیرهای امیدوارکننده استفاده می‌شود. در ادامه، الگوریتم جست‌وجوی کرنل تمام مکان‌ها در مجموعه P را مرتب می‌کند. برای این منظور، ابتدا هزینه تقلیل‌یافته متغیر X_{cp} که با نماد $c'(X_{cp})$ نشان داده می‌شود، در حل بهینه *LP(X,V,Z)* به‌دست آورده؛ سپس مکان‌ها به‌صورت نزولی بر اساس تعداد سوئیچ‌های متصل مرتب می‌شوند. مکان‌هایی که در حل بهینه *LP(X,V,Z)* انتخاب نشده‌اند، یعنی $X_{cp}=0$، بر اساس مقدار $c'(X_{cp})$ به‌صورت صعودی مرتب می‌شوند. این مرتب‌سازی با هدف ایجاد لیست L است که در آن مکان‌هایی که دارای بیشترین احتمال برای نصب کنترلر هستند در ابتدای لیست و سایر مکان‌ها در انتهای لیست قرار گیرند.

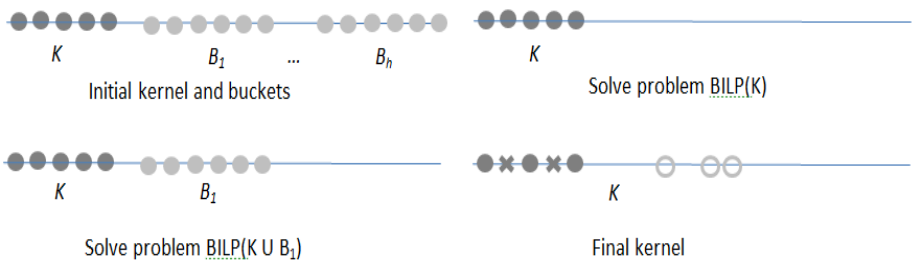
پس از مرتب‌سازی مکان‌ها، سوئیچ‌ها باید به این مکان‌ها متصل گردند. به همین منظور، مقدار هزینه تقلیل‌یافته متغیر V_{sp} که با نماد $c'(V_{sp})$ نمایش داده می‌شود در حل بهینه *LP(X,V,Z)* به‌دست آورده؛ سپس، زیرمجموعه‌ای از سوئیچ‌ها به مکان‌های انتخاب‌شده متصل می‌گردند با این شرط که مقدار $c'(V_{sp})$ برای هر جفت (S,P) نباید از مقدار حد آستانه (γ) بیشتر باشد. یعنی، $\hat{C}(V_{sp}) \le \gamma$. مقدار پارامتر γ، برابر میانه هزینه‌های تقلیل‌یافته متغیرهای V_{sp} مرتبط با مکان‌های عضو کرنل جزیی اولیه $K(X)$ است. یعنی، $\gamma = median\{\hat{C}(V_{sp}) | s \in S, p \in K(X)\}$.

در ادامه، کرنل عمومی به نام K ایجاد می‌شود. مجموعه $K(X)$ شامل m متغیر X_{cp} از ابتدای لیست L است که مقدار m به‌عنوان پارامتر به الگوریتم داده می‌شود. مجموعه $K(V)$ نیز برای هر مکان متعلق به $K(X)$، شامل زیرمجموعه‌ای از سوئیچ‌های متصل به آن مکان است. $|P| - m$ مکان باقی‌مانده به NB مجموعه دسته‌بندی می‌شود که هر مجموعه به‌صورت $B(X)_h$ نشان داده می‌شود. h مقدار بین ۱ تا NB را می‌گیرد. $B(X)_{h=1,\dots,NB}$ بیان‌کننده باکت‌های جزیی است. هر مجموعه $B(X)_h$ به جز آخرین مجموعه یعنی $B(X)_{h=NB}$، دارای تعداد *lbuck* عضو است که مقدار *lbuck* برابر پارامتر m در نظر گرفته می‌شود. آخرین مجموعه نیز تعداد *(|P|-m)-((NB-1)*m)* عضو دارد. بنابراین، با توجه به مطالب بیان‌شده تعداد باکت را نیز می‌توان با $NB = \left\lceil \frac{|P|-m}{lbuck} \right\rceil$ محاسبه کرد. به همین ترتیب، تعداد NB باکت جزیی برای متغیر V ایجاد می‌شود که به‌صورت $B(V)_{h=1,\dots,NB}$ نمایش داده می‌شود. به‌منظور تعیین شرط توقف برای اجرای الگوریتم جست‌وجوی کرنل، حداکثر تعداد اجرای الگوریتم را می‌توان NB در نظر گرفت. از این رو پارامتر NB به‌عنوان تعداد دفعات اجرای الگوریتم جست‌وجوی کرنل تعیین می‌شود.

در الگوریتم جست‌وجوی کرنل، از متغیر z^H به‌عنوان باند بالا جواب بهینه مساله *BILP* استفاده می‌شود. قبل از حل مساله *BILP*، بررسی می‌شود که هر سوئیچ حداقل به یک مکان در مجموعه *K(X)* متصل گردد. اگر سوئیچی متصل نباشد، مجموعه *K(V)* اصلاح می‌گردد، به این صورت که، سوئیچ مربوطه به تمام مکان‌ها در *K(X)* متصل گردیده و مجموعه *K(V)* به‌روزرسانی می‌گردد.

در فاز دوم که به فاز حل معروف است، مساله *BILP* به تعداد *NB* حل می‌شود. در هر تکرار *h*، که $h=1,...,NB$ است، کرنل عمومی جاری با اضافه‌کردن باکت‌های جزیی فعلی $B(X)_h$ و $B(V)_h$ به‌روزرسانی می‌گردد. سپس مساله *BILP* بر اساس کرنل به‌روزرسانی شده و نیز تعریف دو محدودیت جدید به‌منظور کاهش زمان محاسباتی حل می‌گردد. یکی از این محدودیت‌ها تغییر باند بالا راه حل بهینه و دیگری انتخاب حداقل یک مکان از باکت‌های جزیی فعلی برای حل مساله است. در حقیقت، هدف بهبود باند بالای z^H فعلی و به‌کارگیری حداقل یک مکان جدید از باکت جزیی فعلی $B(X)_h$ در حل مساله *BILP* است. اگر مساله *BILP* امکان‌پذیر باشد، آن‌گاه راه حل بهینه، بهترین جواب به‌دست آمده را بهبود می‌دهد. در این حالت، حداقل یک مکان از باکت جزیی فعلی در راه حل بهینه انتخاب می‌شود. یعنی، متغیرهای امیدوارکننده جدید شناسایی می‌شوند و مجموعه کرنل جزیی فعلی *K(X)* با این متغییرها به‌رورزرسانی می‌شود. مجموعه شامل این مکان‌ها به‌صورت $B(X)^+_h$ نشان داده می‌شود. به‌طور عکس، اگر یک مکان در کرنل جزیی فعلی در راه حل بهینه مساله *BILP* انتخاب نشود و به تعداد دفعات مقدار *t*، متغییر امیدوارکننده نشود از کرنل جزئیش حذف می‌گردد. مقدار پارامتر *t* در این الگوریتم برابر ۲ است. مجموعه شامل این مکان‌ها به‌صورت $B(X)^-_h$ نشان داده می‌شود. بنابراین، در پایان تکرار *h*، کرنل جزیی جاری *K(X)* در شروع تکرار *h+1* شامل مکان‌های عضو مجموعه $B(X)^+_h$ و منهای مکان‌های عضو مجموعه $B(X)^-_h$ است. همین روال برای کرنل جزیی فعلی *K(V)* انجام می‌شود. اگر یک مکان جدید به مجموعه *K(X)* اضافه گردد، سوئیچ‌های متصل به مکان جدید نیز به کرنل جزیی فعلی *K(V)* اضافه می‌گردند. به‌طور عکس، زمانی که یک مکان از *K(X)* حذف می‌گردد، سوئیچ‌های متصل به آن مکان از کرنل جزیی فعلی *K(V)* حذف می‌گردند. زمانی که آخرین باکت مورد ارزیابی قرار گیرد، الگوریتم پایان می‌یابد.

شکل ۱ یک مرحله از اجرای الگوریتم جست‌وجوی کرنل را نشان می‌دهد. در فاز اول، کرنل اولیه (دایره‌های خاکستری تیره) و باکت‌ها (دایره‌های خاکستری روشن) تعریف می‌شوند. در پایان فاز اول، *BILP(K)* حل می‌شود. اولین مساله حل‌شده در فاز دوم $BILP(K\ U\ B_1)$ است. اگر یک راه حل بهبودیافته یافت شود، کرنل به‌روزرسانی می‌شود، یعنی، متغیرهای امیدوارکننده (دایره‌های باز) اضافه می‌شوند و متغیرهای که برای مدت زمان طولانی امیدوارکننده نیستند از کرنل حذف می‌گردند (ضرب‌در). فاز دوم با حل $BILP(K\ U\ B_2)$ ادامه می‌یابد که کرنل *K* به‌روزرسانی شده است.



شکل ۱- یک مرحله از اجرای الگوریتم جست‌وجوی کرنل.

Figure 1- A step in the implementation of the kernel search algorithm.

در دو حالت الگوریتم جست‌وجوی کرنل با شکست روبه‌رو می‌شود. در حالت اول، زمانی که هیچ راه حل امکان‌پذیری برای $LP(X,V,Z)$ یافت نشود. این تضمین می‌دهد که هیچ راه حل امکان‌پذیری برای مساله اصلی وجود ندارد. حالت دوم زمانی رخ می‌دهد که هیچ راه حل امکان‌پذیری برای مساله *BILP* وجود نداشته باشد. این به این معنی است که نه راه حل امکان‌پذیری برای مساله اصلی وجود دارد و نه جست‌وجوی کرنل قادر به یافتن هیچ یک راه حل امکان‌پذیر است. حالت بعدی که ممکن است رخ دهد زمانی است که ظرفیت مکان‌ها برای اتصال سوئیچ‌ها به آن‌ها به اندازه کافی نیست. پیاده‌سازی الگوریتم جست‌وجوی کرنل وابسته به تعیین دقیق پارامترها است.

۵- نتایج شبیه‌سازی

در این بخش، آزمایشات و محاسبات انجام‌شده به‌منظور ارزیابی روش‌های حل مساله مکان‌یابی کنترلر گردآوری شده است. در این راستا، کمیت‌های زمان اجرا و کیفیت جواب برای هر یک از روش‌های حل مساله با استفاده از *CPLEX* و الگوریتم جست‌وجوی کرنل بر روی مجموعه‌ای

از نمونه‌های استاندارد مساله، محاسبه و در مقایسه با یکدیگر گزارش شده است. *CPLEX* [17]-[19] در حال حاضر یکی از سریع‌ترین حل‌کننده‌های غیرتجاری برای حل مسایل برنامه‌ریزی خطی و آمیخته‌ی عدد صحیح (*MIP*) است. این حل‌کننده‌ها امکان کنترل کامل بر روی فرآیند حل مساله و دستیابی به اطلاعات دقیق را فراهم می‌کند. *CPLEX* به‌صورت کتابخانه‌های قابل فرخوانی از زبان *C* پیاده‌سازی شده است. نحوه‌ی ساخت نمونه‌های مورد آزمایش به شرح زیر است:

برای هر نمونه ابتدا توپولوژی شبکه از روی یک گرید ۲۰×۲۰ و به‌طور تصادفی استخراج می‌شود. به این معنی که هر گره از گرید مذکور با احتمال *pr* یک نود از گراف شبکه خواهد بود. مقدار *pr* برای نمونه‌ها ۰/۲۵ در نظر گرفته شده است. پس از انتخاب گره‌های مذکور، گراف کامل القا شده به آن توپولوژی شبکه از نمونه‌ی مذکور خواهد بود. روی این گراف وزن هر یال فاصله‌ی اقلیدسی بین دو سر آن تعریف می‌شود. در مرحله بعد گره‌های حاوی سوئیچ روی گراف مذکور مشخص می‌شوند. به این ترتیب که برای یک نمونه با *i* سوئیچ، *i* نود از گراف مذکور به تصادف انتخاب شده و روی هر یک از آن‌ها یک سوئیچ نصب می‌گردد.

پارامتر *i*، مقدار سوئیچ‌های نصب‌شده در شبکه، در ساخت نمونه‌های فوق، عددی از مجموعه‌ی $\{5k \mid k=0,1,\ldots,20\}$ انتخاب شده است.

محاسبات این بخش بر روی یک پردازشگر از نوع *Intel core i5* در حالت تک‌پردازنده و تحت سیستم عامل اوبونتو با حافظه‌ی اصلی ۴ گیگابایت انجام شده است. الگوریتم پیشنهادی جست‌وجوی کرنل توسط زبان *C* پیاده‌سازی شده است. برای *CPLEX*، محدودیت زمانی برابر ۳۶۰۰ ثانیه در نظر گرفته شده است. پارامترهای تعریف‌شده برای حل مساله در *CPLEX* و الگوریتم پیشنهادی جست‌وجوی کرنل در **جدول ۱** بیان شده است.

جدول ۱- پارامترهای مساله.
Table 1- Problem parameters.

نام پارامتر	مقدار
هزینه هر کنترلر (k^c)	2500 دلار
تعداد پورت هر کنترلر (α^c)	32
تعداد بسته قابل پردازش توسط کنترلر (μ^c)	4000
تعداد کنترلر موجود (φ^c)	15
هزینه لینک به متر (Φ^l)	8.25 دلار
پهنای باند لینک (ω^l)	100 مگابیت بر ثانیه
اندازه بسته (β)	150 بایت
حداکثر تاخیر شبکه (γ)	250 میلی ثانیه
متوسط زمان پردازش بسته توسط کنترلر (δ)	0.001 میلی ثانیه

نتایج به‌دست آمده از این آزمایشات در **جدول ۲** بر اساس نمونه‌ها گردآوری شده‌اند. در این جدول، برای هر نمونه از مسایل موارد زیر گزارش شده‌اند.

cost: بهترین هزینه روی این نمونه‌ها.

time: زمان صرف‌شده روی این نمونه‌ها.

imp %: درصد بهبودی الگوریتم پیشنهادی جست‌وجوی کرنل نسبت به *CPLEX* از لحاظ بهترین هزینه روی این نمونه‌ها که به شیوه‌ی زیر محاسبه می‌شوند.

$$100 * \frac{cost_{CPLEX}(\rho) - cost_{KS}(\rho)}{cost_{KS}(\rho)}. \qquad (۱۶)$$

در محاسبه این کمیت هر یک از توابع $cost_{KS}(.)$ و $cost_{CPLEX}(.)$ به‌ترتیب بهترین هزینه به‌دست آمده از اجرای الگوریتم پیشنهادی جست‌وجوی کرنل و حل‌کننده *CPLEX* را به‌دست می‌دهند.

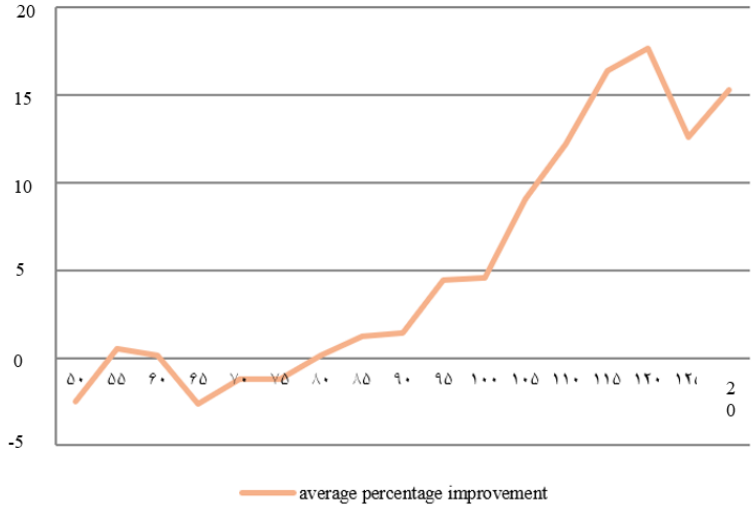
جدول ۲- نتایج حاصل از اجرای الگوریتم KS و CPLEX بر روی نمونه‌ها.
Table 2- Results of running the KS and CPLEX algorithms on the samples.

	CPLEX		KS		CPLEX vs KS
Instance	Time	Cost	Time	Cost	Imp
Switch	(Sec.)	$	(Sec.)	$	%
5	>3600	2101445	83.10	2156934	-2.57
10	>3600	2086498	85.01	2121060	-1.63
15	>3600	1997456	82.23	2032568	-1.73
20	>3600	2003364	76.15	2057015	-2.61
25	>3600	2037644	74.10	2121134	-3.94
30	>3600	2137956	72.70	2152656	-0.68
35	>3600	2123838	82.77	2124522	-0.03
40	>3600	2166462	70.15	2138024	1.33
45	>3600	2205382	63.67	2184970	0.93
50	>3600	2105740	71.12	2081748	1.15
55	>3600	2274063	80.94	2279755	-0.25
60	>3600	2362716	88.19	2295361	2.93
65	>3600	2283537	78.14	2282079	0.06
70	>3600	2283537	78.67	2289079	0.06
75	>3600	2290199	78.70	2339523	-2.11
80	>3600	2416234	88.98	2482917	-2.69
85	>3600	2470685	87.36	2556031	-3.34
90	>3600	2419417	90.90	2513834	-3.76
95	>3600	2565170	114.60	2568095	-0.11
100	>3600	2405980	104.64	2494901	-3.18

در **جدول ۲**، ستون اول نام نمونه‌های مورد آزمایش را بیان می‌کند. ستون دوم، مدت زمان اجرا *CPLEX* بر روی نمونه‌ها را نشان می‌دهد. ستون سوم نشان‌دهنده مقدار هزینه بهترین جواب به‌دست آمده برای هر نمونه توسط *CPLEX* با در نظر گرفتن حداکثر مدت زمان اجرا ۳۶۰۰ ثانیه است. ستون چهارم و پنجم نیز به‌ترتیب نشان‌دهنده مدت زمان اجرا و مقدار هزینه بهترین جواب به‌دست آمده توسط الگوریتم پیشنهادی جست‌وجوی کرنل است. در انتها، ستون ششم درصد بهبودی الگوریتم پیشنهادی جست‌وجوی کرنل در مقایسه با *CPLEX* از لحاظ مقدار هزینه بهترین راه حل به‌دست آمده را بیان می‌کند.

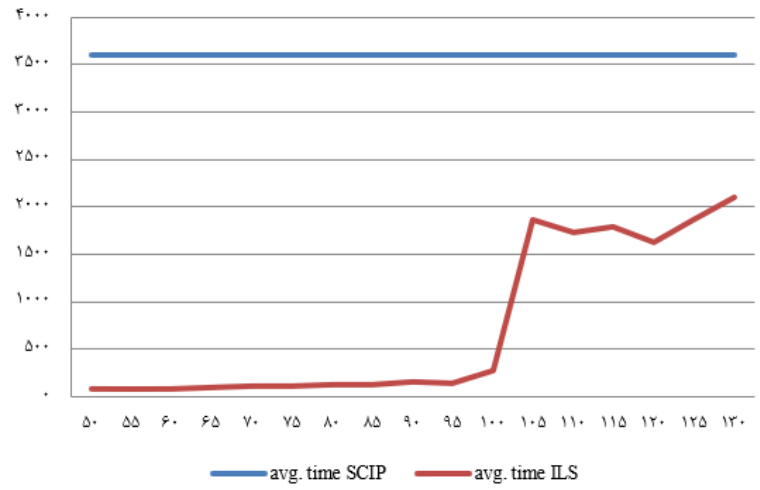
نتایج به‌دست آمده از **جدول ۲** به برتری الگوریتم پیشنهادی جست‌وجوی کرنل در زمان اجرا برای تمام نمونه‌ها و بهترین هزینه به‌دست آمده در برخی از این نمونه‌ها اشاره دارد.

شکل ۲ میانگین درصد بهبودی الگوریتم پیشنهادی جست‌وجوی کرنل در مقایسه با *CPLEX* برای نمونه‌های مختلف را نشان می‌دهد. محور افقی اندازه نمونه‌ها بر اساس تعداد سوئیچ نصب‌شده در شبکه و محور عمودی میانگین درصد بهبودی روی نمونه را نشان می‌دهد. با توجه به این نمودار، برتری الگوریتم جست‌وجوی کرنل در یافتن جواب‌هایی با هزینه‌ی کمتر با افزایش تعداد سوئیچ‌های موجود در نمونه و در مقایسه با *CPLEX* رفته رفته بیشتر می‌شود. این مهم، الگوریتم پیشنهادی را به‌عنوان ابزاری مهم برای حل مساله در مقیاس بزرگ مطرح می‌کند.



شکل ۲- میانگین درصد بهبود به‌دست آمده از اجرای الگوریتم جست‌وجوی کرنل در مقایسه با CPLEX.

Figure 2- Average percentage improvement achieved by implementing the kernel search algorithm compared to CPLEX.



شکل ۳- میانگین زمان اجرای الگوریتم جست‌وجوی کرنل روی نمونه‌های هم اندازه در مقایسه با CPLEX.

Figure 3- Average execution time of the kernel search algorithm on samples of equal size compared to CPLEX.

شکل ۳، زمان اجرای الگوریتم جست‌وجوی کرنل را در مقایسه با *CPLEX* در نظر قرار داده است. محور افقی در این شکل، اندازه نمونه‌ها بر اساس تعداد سوئیچ نصب‌شده و محور عمودی میانگین زمان اجرا روی نمونه‌ها را نشان می‌دهد. نتایج به‌دست آمده در این شکل، به برتری زمانی روش پیشنهادی اشاره دارد. به بیان دیگر، الگوریتم پیشنهادی قادر است با به‌کارگیری مناسب منبع زمان، جواب‌های بهتری را در زمان کوتاه‌تر بیابد. این مهم، روش پیشنهادی را به‌عنوان روش کارا در حل مساله‌ی مکان‌یابی کنترلر مطرح خواهد کرد.

۶- نتیجه‌گیری

در این مقاله مساله مکان‌یابی کنترلر در شبکه‌های *SDN* مورد مطالعه قرار گرفته است. برای حل این مساله الگوریتمی مبتنی بر روش جست‌وجوی کرنل پیشنهاد شده است. الگوریتم پیشنهادشده از مفاهیمی همچون کرنل، *Bucket* و *reduced cost* برای رسیدن به هدف استفاده می‌کند تا به توپولوژی کارامدی از شبکه در یک زمان قابل قبول دست یابد. به‌منظور بررسی عملکرد الگوریتم پیشنهادی، آزمایشاتی بر روی چندین نمونه از شبکه انجام شده است. نتایج به‌دست آمده از الگوریتم پیشنهادی با نتایج *CPLEX* که بر روی همان نمونه‌ها اجرا شده است مورد مقایسه قرار گرفته است. نتایج مقایسه، از برتری الگوریتم پیشنهادی در زمان اجرا برای تمام نمونه‌ها و در بهترین هزینه به‌دست آمده در برخی از نمونه‌ها حکایت دارد.

منابع

[1] Blial, O., Ben Mamoun, M., & Benaini, R. (2016). An overview on SDN architectures with multiple controllers. *Journal of computer networks and communications*, *2016*(1), 9396525. https://doi.org/10.1155/2016/9396525

[2] Selvi, H., Güner, S., Gür, G., & Alagöz, F. (2015). The controller placement problem in software defined mobile networks (SDMN). In *Software defined mobile networks (SDMN) beyond lte network architecture* (pp. 129–147). Wiley online library. https://doi.org/10.1002/9781118900253.ch8

[3] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE communications surveys & tutorials*, *16*(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

[4] Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). A survey on software-defined networking. *IEEE communications surveys & tutorials*, *17*(1), 27–51. https://doi.org/10.1109/COMST.2014.2330903

[5] ONF. (2024). *Open networking foundation.* https://opennetworking.org/

[6] Jarraya, Y., Madi, T., & Debbabi, M. (2014). A survey and a layered taxonomy of software-defined networking. *IEEE communications surveys & tutorials*, *16*(4), 1955–1980. https://doi.org/10.1109/COMST.2014.2320094

[7] Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., … ., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE communications magazine*, *51*(7), 36–43. https://doi.org/10.1109/MCOM.2013.6553676

[8] Heller, B., Sherwood, R., & McKeown, N. (2012). The controller placement problem. *Association for computing machinary special interest group on data communication computer communication review.*, *42*(4), 473–478. https://doi.org/10.1145/2377677.2377767

[9] Yao, G., Bi, J., Li, Y., & Guo, L. (2014). On the capacitated controller placement problem in software defined networks. *IEEE communications letters*, *18*(8), 1339–1342. https://doi.org/10.1109/LCOMM.2014.2332341

[10] Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics* (pp. 129–168). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-91086-4_5

[11] Xiao, P., Qu, W., Qi, H., Li, Z., & Xu, Y. (2014). The SDN controller placement problem for wan. *2014 IEEE/CIC international conference on communications in china (ICCC)* (pp. 220–224). IEEE. https://doi.org/10.1109/ICCChina.2014.7008275

[12] Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2012). On the placement of controllers in software-defined networks. *The journal of China universities of posts and telecommunications*, *19*, 92–171. https://doi.org/10.1016/S1005-8885(11)60438-X

[13] Zhang, Y., Beheshti, N., & Tatipamula, M. (2011). On resilience of split-architecture networks. *2011 IEEE global telecommunications conference - globecom 2011* (pp. 1–6). IEEE. https://doi.org/10.1109/GLOCOM.2011.6134496

[14] Obadia, M., Bouet, M., Rougier, J. L., & Iannone, L. (2015). A greedy approach for minimizing SDN control overhead. *Proceedings of the 2015 1st IEEE conference on network softwarization (netsoft)* (pp. 1–5). IEEE. https://doi.org/10.1109/NETSOFT.2015.7116135

[15] Zhang, T., Bianco, A., & Giaccone, P. (2016). The role of inter-controller traffic in sdn controllers placement. *2016 IEEE conference on network function virtualization and software defined networks (NFV-SDN)* (pp. 87–92). IEEE. https://doi.org/10.1109/NFV-SDN.2016.7919481

[16] Sallahi, A., & St-Hilaire, M. (2015). Optimal model for the controller placement problem in software defined networks. *IEEE communications letters*, *19*(1), 30–33. https://doi.org/10.1109/LCOMM.2014.2371014

[17] SCIP. (2015). *Solving Constraint Integer Programs*. https://www.scipopt.org/

[18] Mueller, J., Wierz, A., & Magedanz, T. (2013). Scalable on-demand network management module for software defined telecommunication networks. *2013 IEEE SDN for future networks and services (SDN4FNS)* (pp. 1–6). IEEE. https://doi.org/10.1109/SDN4FNS.2013.6702550

[19] Herbaut, N., Negru, D., Magoni, D., & Frangoudis, P. A. (2016). Deploying a content delivery service function chain on an SDN-NFV operator infrastructure. *2016 international conference on telecommunications and multimedia (TEMU)* (pp. 1–7). IEEE. https://doi.org/10.1109/TEMU.2016.7551917