تابستان ۱۴۰۰، دوره ۶، شماره ۲، پیاپی ۲۱
DOR :20.1001.1.2538161.1400.6.2.8.8

دریافت مقاله: ۱۳۹۹/۱۲/۲۵
پذیرش مقاله: ۱۴۰۰/۰۲/۱۳

**نوع مقاله: پژوهشی**

# تخصیص منابع پردازشی لبه در شبکه‌های اینترنت اشیاء

**علی سرورامینی**
دانشجوی دانشکدهٔ مهندسی برق و کامپیوتر، پردیس فنی، دانشگاه تهران، ایران
پست الکترونیکی: ali.sorouramini@ut.ac.ir

**وحید شاه‌منصوری***
استادیار دانشکدهٔ مهندسی برق و کامپیوتر، پردیس فنی، دانشگاه تهران، ایران
پست الکترونیکی: vmansouri@ut.ac.ir,

## چکیده

با حرکت به سمت عصر اینترنت اشیاء، تعداد دستگاه‌های متصل به اینترنت به صورت نمایی در حال افزایش است. تعداد زیاد دستگاه‌های متصل باعث ایجاد گلوگاه‌هایی در حوزه‌های مختلف مانند اتصال دستگاه‌ها، انتقال و پردازش داده‌ها می‌شود. پردازش لبه یک روش مناسب برای پردازش حجم زیاد داده‌های تولید شده توسط اشیاء متصل به اینترنت به شمار می‌رود. به لطف پیشرفت‌های ایجاد شده در فناوری‌های مجازی‌سازی، دروازه‌های شبکه و دستگاه‌های لبه شبکه می‌توانند ظرفیت پردازشی اضافی خود را در اختیار سرویس‌های اینترنت اشیاء قرار دهند. تعداد بسیار زیاد دستگاه‌های لبه و سرویس‌های شبکه اینترنت اشیاء، باعث پیچیده شدن مسئله تخصیص منابع پردازشی به سرویس‌های شبکه‌اینترنت اشیاء می‌شود. در این مقاله، وظیفه تخصیص منابع پردازشی مورد نیاز سرویس‌های اینترنت اشیاء را برای تعداد زیاد سرویس‌ها در نظر می‌گیریم. هر سرویس می‌تواند از چند منبع پردازشی استفاده کند و هر منبع پردازشی هم می‌تواند به چند سرویس اختصاص پیدا کند. مسئله به صورت یک مسئله بهینه‌سازی مدل می‌شود که هدف آن بیشینه کردن مجموع سود سرویس‌ها است. مسئله بهینه‌سازی نهایی یک مسئله برنامه‌ریزی غیرخطی عدد صحیح مخلوط است که در حالت کلی به سختی حل می‌شود. الگوریتم زیربهینه ارائه شده برای حل این مسئله جوابی قابل قبول برای آن به‌دست می‌دهد که به صورت توزیع شده قابل اجرا است.

**واژه‌های کلیدی:** اینترنت اشیاء، رایانش لبه، تخصیص منابع

## مقدمه

گسترش اینترنت اشیاء و موفقیت سرویس‌های ابری باعث ایجاد الگوی جدیدی در پردازش داده‌ها به نام رایانش لبه[1] شده است. طبق برآوردهای انجام شده در [۱] تعداد دستگاه‌های متصل شده به شبکه ۵۰ میلیارد عدد خواهد بود. بعضی از کاربردهای اینترنت اشیاء نیاز به زمان پاسخ کوتاه دارند، بعضی ممکن است دارای داده‌های

1- Edge Computing

* نویسندهٔ مسئول

محرمانه و شـــخصی باشـــند و بعضی از ایـــن کاربردها می‌توانند بار سنگینی برای شبکه داشته باشند و پردازش ابری ممکن است روش مناسبی برای این کاربردها نباشد.

در اینجا منظور از لبه، همه منابع پردازشی و شبکه‌ای اســـت که بین منبع داده‌ها (جایی که داده‌ها در آن جا تولید می‌شـــوند) و مراکز دادهٔ ابری قـــرار دارند. برای مثال یک دروازهٔ شـــبکه در یـــک خانه هوشـــمند، می‌تواند یک منبع پردازشـــی لبه بین اشـــیاء و مرکز دادهٔ ابری باشد یا یک مرکز دادهٔ کوچـــک، می‌تواند یک منبع پردازشـــی لبه بین دســـتگاه‌های ســـیار و ابر در نظر گرفته شـــود. منطق در رایانش لبه این است که پردازش داده‌ها باید در همسایگی منبع داده‌ها انجام شود. با این منطق رایانش لبه و رایانش مه دو مفهوم یکســـان را خواهند داشـــت با این تفاوت که رایانش لبه تمرکزش ســـمت اشیاء اســـت ولی رایانش مه تمرکزش سمت زیرساخت اســـت. فشار از سمت رایانش ابـــری، تغییر از مصرف کننده داده بـــه تولید کننده داده و کشش از سمت اینترنت اشـــیاء از عوامل جذابیت رایانش لبه است [۲].

در الگوی رایانش لبه، اشـــیاء نه تنها مصرف کننده داده بلکه تولید کننده داده نیز هســـتند. در واقع در لبه، اشیاء نه تنها می‌توانند ســـرویس و محتوا از ابر درخواســـت نمایند بلکه می‌توانند وظایف پردازشـــی را هم انجام دهند. در [۳] برنامه‌ای برای تشـــخیص چهره ساخته شـــده است که با انتقال پردازش از ابر به لبه شـــبکه، زمان پاســـخ برنامه از ۹۰۰ به ۱۶۰ میلی ثانیه کاهش پیدا کرده اســـت. نویسندگان در [٤] به بررســـی یک بُن‌ســـازهٔ رایانش مه که از Docker اســـتفاده می‌کند پرداخته‌اند. بررسی آن‌ها نشان می‌دهد که این بُن‌ســـازه کارایی بهتری نسبت به روش‌های فعلی دارد. در [٥] نویسندگان از واحدهای پردازشی ابری کوچک[2] برای پردازش وظایف دســـتیار شناختی پوشیدنی استفاده کرده اند و نتایج نشان می‌دهد که زمان پاسخ بین ۸۰ تا ۲۰۰ میلی ثانیه کاهش پیدا کرده اســـت. در [٦] نویســـندگان، Docker را به عنـــوان فناوری که امکان اســـتفاده از پردازش لبه را فراهم می‌کند بررســـی کرده‌اند. در [۷] و [۸] از فناوری‌های مجازی‌سازی سبک برای طراحی دروازه‌های[3] اینترنت اشیاء استفاده شده است. در [۷] از یک نرم‌افزار مجازی‌سازی برای استقرار انبوه سرویس‌ها در لبه شبکه استفاده کرده‌اند. نکته قابل توجه در تحلیل آن‌ها، امکان اثر متقابل بین حسگرهای اینترنت اشیاء و دروازه‌ها است. تحلیل آن‌ها نشان می‌دهد که تخصیص پویای سرویس‌ها با استفاده از کانتینرها مزایایی از بعد کارایی در دروازه‌ها دارد.

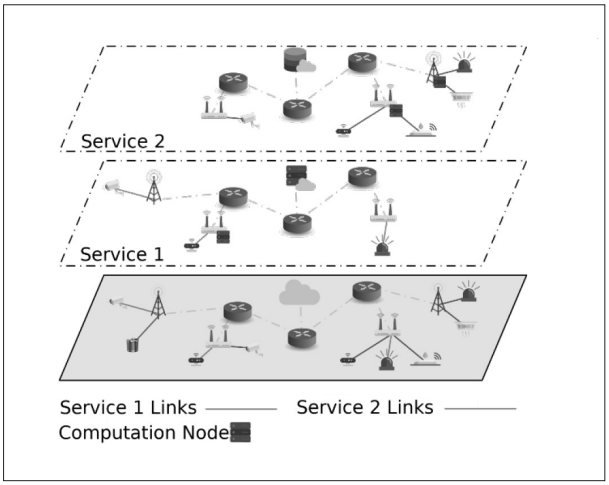
[۹] به بررســـی کارآیـــی فناوری‌هـــای کانتینری روی کامپیوترهای تک‌بـــوردی مختلف می‌پـــردازد. در همه این مقالات نتیجه‌گیری بر این است که با استفاده از کانتینرها در دستگاه‌های مورد نظر، سربار پردازشی قابل چشم پوشی است. تخصیص منابع پردازشی برای سرویس‌ها در اینترنت اشـــیاء به دلیل تعداد زیاد ســـرویس‌ها و منابع پردازشی، یک مســـئله پیچیده است. نویســـندگان در [۱۰] چهارچوبی پیاده ســـازی کرده‌اند که در اختصاص منابع پردازشی در یک شـــبکه ابری و مـــه، تأخیر انتقال و مصـــرف انرژی را بهینه می‌کند. در [۱۱] نویســـندگان مســـئله توزیع پردازش ســـرویس‌ها را به صورت مسئله بهینه‌ســـازی برای کمینه کردن هزینه فرمول‌بندی کرده‌اند. [۱۲] به بررسی تخصیص منابع پردازشـــی در یک شبکه رایانش مه می‌پردازد. در این مقاله شبکه با سه لایه مدل می‌شود که شامل لایهٔ گره‌های مـــه، لایهٔ اپراتورهای مراکز داده و لایه کاربران اســـت و از بازی استکلبرگ[4] و نظریهٔ تطبیق[5] برای پیدا کردن نقطه تعادل استفاده شده اســـت. در [۱۳] از رایانش مه برای کاهش بار پردازشی مراکز داده استفاده شده است. مسئله به صورت یک بهینه‌ســـازی فرمول‌بندی شده اســـت که تلاش می‌کند تأخیر، مصرف انرژی و هزینه را کمینه کند. نویسندگان در [١٤] مسئلهٔ تخصیص منابع رادیویی و پردازشی را بررسی کرده‌اند. در این مقاله تابع هدف شـــامل بیشـــینه کردن نرخ کاربران و کمینه کردن هزینه اســـت و از نظریهٔ تطبیق برای پیدا کردن جواب استفاده شده است.

2- Cloudlet

3- Wearable Cognitive Assistance

4- Gateway

5- Stackelberg Game




شکل ۱: مدل سیستم

در این مقاله، مسئله تخصیص منابع برای سرویس‌های اینترنت اشـیاء در نظر گرفته می شود. فرض بر این است که با کمک فناوری Docker هر منبع پردازشی لبه می‌تواند به چند سرویس اینترنت اشیاء سرویس دهد. همچنین هر سـرویس می‌تواند از چند منبع پردازشـی استفاده کند. مسئله تخصیصی منابع به‌صورت یک مسئله بهینه‌سازی مدل می‌شود. سپس یک روش زیربهینه برای حل مسئله با پیچیدگی پردازش پایین ارائه می شود.

سـاختار مقاله در ادامه به‌صورت زیر است. در فصل دوم به بررسی مدل سیستم می‌پردازیم و مسئله بهینه‌سازی تخصیص منابع را در قالب یک مسئله بهینه‌سازی معرفی می‌کنیم. فصل سوم روش زیربهینه پیش نهادی را معرفی می‌کند. فصل چهارم نتایج شبیه‌سـازی را نشان می‌دهد و فصل پنجم نتیجه گیری مقاله را شامل می شود.

## ۲- مدل سیستم

در یک منطقـه جغرافیایی تعداد $N$ سـرویس اینترنت اشـیاء وجود دارد و تعدادی فراهم کننده زیرساخت[6] که منابعی پردازشـی را در اختیار سرویس‌ها قرار می‌دهند. هر سـرویس از تعدادی حسـگر و عملگر تشکیل شده که نیاز بـه پردازش اطلاعات دارند. مجموعه سـرویس‌ها را با $S$ و مجموع منابع پردازشـی را بـا $C$ نمایش می‌دهیم.

منابع پردازشی قابل استفاده برای سرویس $s$ با مجموعه $C_s$ نشـان داده می شـود که زیرمجموعه‌ای از $C$ اسـت. فراهم‌کننده‌های زیرساخت[7] علاقه دارند که توان پردازشی خـود را با دیگران به اشـتراک بگذارند. مدل سیسـتم در شکل ۱ نشان داده شده است.

$\phi_i$ نشان‌دهنده توان پردازشی فراهم‌کننده زیرساخت $i$ اسـت و $v_i$ درصدی از توان پردازشـی که فراهم‌کننده زیرساخت پردازشـی $i$ مایل به اشـتراک‌گذاری است را نشان می‌دهد. در نتیجه، فراهم‌کننده زیرساخت پردازشی $i$ام، مایـل به اشـتراک‌گذاری تـوان پردازشـی برابر با $\varphi_i = \phi_i v_i$ خواهد بـود. در اینجا، گره‌های لبه[8]، گره‌هایی هسـتند که زیرسـاخت پردازشـی را در لبه شبکه فراهم می‌کنند. در مقابل، فراهم‌کننده‌های زیرساخت ابری، منابع پردازشـی را در فاصلـه‌ای دورتر از سـرویس‌ها فراهم می‌کننـد. بنابرایـن، گره‌های لبه، توان پردازشـی کم‌تر با تأخیر کم‌تر را در مقایسـه با فراهم‌کننده‌های زیرسـاخت ابری مهیا می‌کنند. بده‌بسـتان، بین میزان توان پردازشی و تأخیر در پردازش، باعث پیچیده‌شـدن انتخاب گره‌های پردازشی برای سرویس‌ها می‌شود.

$\mu_{i,s}$ نشـان‌دهنده نرخ سرویس منبع پردازشی $i$ برای سرویس $s$ است. $r_s$ نرخ انتخابی ارسال نمونه‌های سرویس $s$ است و $R_s$ نرخ مطلوب ارسال نمونه‌ها برای سرویس $s$ را نشـان می‌دهد. جدول ۱ به صورت خلاصه پارامترهای استفاده شده در این مقاله را معرفی می‌کند.

در ایـن جا فرض می‌کنیم که ارزش منبع پردازشـی $i$ برای سرویس $s$ از دو بخش تشکیل شده است. بخش اول، تابع نرخ ارسال نمونه‌ها توسط حسگرهای سرویس $s$ به منبع پردازشـی $i$ است. این تابع را $f_r$ نام‌گذاری می‌کنیم و فرض می‌کنیم که $f_r$ تابعی از نرخ ارسـال نمونه‌ها توسط حسگرهای سـرویس $s$ و نرخ مطلوب آن سرویس است. در واقع $f_r$ بیان کنندهٔ تاثیر $r_s$ در ارزش به‌دسـت‌آمده از پردازش اطلاعات برای سـرویس‌ها اسـت. بخش دوم هم

6- Matching Theory

7-Infrastructure Provider
8- Ege Nodes

جدول ۱: نمادهای استفاده شده

| نشانه | توضیح |
|---|---|
| $N$ | تعداد سرویس‌های شبکه |
| $S$ | مجموعه سرویس‌های شبکه |
| $C$ | مجموعه منابع پردازشی شبکه |
| $C_s$ | مجموعه منابع پردازشی که سرویس $s$ می‌تواند از آن‌ها استفاده کند. |
| $\alpha_s$ | ضریب اهمیت سرویس $s$ |
| $\mu_{i,s}$ | نرخ سرویس منبع پردازشی $i$ برای سرویس $s$ |
| $r_s$ | نرخ انتخابی ارسال نمونه‌های سرویس $s$ |
| $r_{i,s}$ | نرخی که سرویس $s$ برای پردازش به منبع پردازششی $i$ می‌فرستد |
| $R_s$ | نرخ مطلوب ارسال نمونه‌ها برای سرویس $s$ |
| $\omega_s$ | ضریب وزن دهی برای مشخص کردن نسبت اهمیت تأخیر نسبت به اختلاف نرخ انتخابی و نرخ مطلوب برای سرویس‌ها |
| $\delta_{i,s}$ | متغیر دودویی که مشخص می‌کند سرویس $s$ از منبع پردازشی $i$ استفاده می‌کند یا نه |
| $d_{i,s}$ | تأخیر منبع پردازشی $i$ برای سرویس $s$ |
| $d_{i,s}^{CPU}$ | تأخیر پردازشی منبع $i$ برای سرویس $s$ |
| $d_{i,s}^{net}$ | تأخیر شبکه سرویس $s$ زمانی که از منبع پردازشی $i$ استفاده می‌کند |
| $d_s^{max}$ | بیشترین تأخیر سرویس $s$ بین منابع پردازشی که از آن‌ها استفاده می‌کند |
| $u_{i,s}$ | بخشی از ظرفیت پردازشی منبع پردازشی $i$ که توسط سرویس $s$ استفاده می‌شود |
| $Q_s$ | حداکثر تعداد منابع پردازشی که سرویس $s$ مجاز به استفاده از آن‌ها است |
| $Q_i$ | حداکثر تعداد سرویس‌هایی که منبع پردازشی $i$ می‌تواند به آن‌ها اختصاص پیدا کند |
| $f_r, f_d$ | توابع مطلوبیت تأخیر و سرعت |

تابعی از تأخیر ایجاد شده (مجموع تأخیر شبکه و تأخیر پردازش) است که آن را با $f_d$ نمایش می‌دهیم. $d_s^{max}$ نشان‌دهنده تأخیر سرویس $s$ است که در ادامه معرفی می‌کنیم. اگر فرض کنیم $R_s$ نرخ ارسال مطلوب نمونه‌ها برای سرویس $s$ و $r_s$ نرخ ارسال نمونه‌های سرویس $s$ باشد، ارزشی که سرویس $s$ با استفاده از منبع پردازشی $i$ به‌دست می‌آورد را به صورت زیر می‌توان نوشت

$$U_s = \omega_s f_r(r_s, R_s) + (1-\omega_s) f_d(d_s^{max}). \qquad (۱)$$

نکته قابل توجه در این رابطه این است که $\omega \in [0,1]$ یک ضریب وزن دهنده است که تعیین کننده نسبت تاثیر $f_r$ به $f_d$ است. اگر $\omega = 1$ باشد ضریب $f_d$ صفر خواهد بود، در نتیجه تأخیر اثری در ارزش به‌دست‌آمده برای سرویس نخواهد داشت. اما $\omega = 0$ بیان‌گر این است که نرخ اختصاص یافته برای سرویس اهمیتی ندارد و فقط تأخیر برای سرویس مهم است. واضح است که اگر $\omega = 0$ باشد، نرخی که ارزشی به‌دست آمده برای سرویس را بیشینه می‌کند $r_s = 0$ است. در نتیجه $\omega = 0$ نتیجه قابل قبولی نخواهد داشت. $\alpha_s$ برای اثر دادن ارزش ذاتی سرویس‌ها است. پس باید برای سرویس با ارزش بالاتر، $\alpha_s$ مقدار بیشتری داشته باشد. در ادامه برای نشان دادن اختصاص منبع پردازشی $i$ به سرویس $s$ از $\delta_{i,s}$ استفاده می‌کنیم. $\delta_{i,s}$ برابر یک است اگر منبع پردازشی $i$ به سرویس $s$ اختصاص پیدا کرده باشد. در غیر این‌صورت $\delta_{i,s}$ برابر صفر است.

$r_s$ مجموع نرخی است که توسط منابع پردازشی برای سرویس $s$ تخصیص می یابد. اگر $r_{i,s}$ را نرخی که توسط

منبع پردازشی $i$ برای سرویس $s$ پردازش می‌شود در نظر بگیریم، رابطه زیر برای $r_s$ برقرار است

$$r_s = \sum_{i \in C_s} r_{i,s}. \quad (۲)$$

فرض بر این است که هر سرویس می‌تواند از چند منبع پردازشی استفاده کند، بیشترین تأخیر منابع پردازشی مورد استفاده هر سرویس را به عنوان تأخیر آن سرویس در نظر می‌گیریم. اگر فرض کنیم $d_{i,s}$ تأخیر سرویس $s$ در منبع پردازشی $i$ و $d_s^{max}$ بیشترین تأخیر سرویس $s$ باشد، $d_s^{max}$ باید از همه $d_{i,s}$ها بزرگتر باشد. در نتیجه قید زیر باید برای همه منابع پردازشی انتخاب شده توسط سرویس $s$ برقرار باشد

$$d_s^{max} = \max_i \{d_{i,s}\} \geq d_{i,s}; \forall s \in S, i \in C_s, \delta_{i,s} = 1. \quad (۳)$$

هر کدام از تأخیرهای $d_{i,s}$ها از دو بخش تشکیل شده‌اند. قسمت اول آن مربوط به تأخیر شبکه است که برابر است با زمانی که طول می‌کشد تا نمونه‌ها به منبع پردازشی برسند به علاوه زمانی که طول می‌کشد نتیجه به مقصد برسد. برای سرویس $s$ تأخیر شبکه زمانی که از منبع $i$ استفاده می شود با $d_{i,s}^{net}$ نشان داده می شود. قسمت دوم، تأخیر پردازش نمونه‌ها در منابع پردازشی است. این تأخیر برابر زمانی است که طول می‌کشد تا نمونه‌ها پس از رسیدن به منابع پردازشی، پردازششان پایان یابد. برای محاسبه تأخیر پردازشی از نظریهٔ صف استفاده می‌کنیم. به دلیل اینکه فرض بر این است که منابع پردازشی بین چند سرویس ممکن است تقسیم شوند، از مدل $M/M/1$ استفاده می‌کنیم چرا که در این حالت پردازنده منابع پردازشی بین سرویس‌های آن منبع به اشتراک گذاشته می‌شود. در مدل $M/M/1$ میانگین تأخیر برابر است با $1/(\mu - \lambda)$ [۱۵].

ظرفیت پردازشی منبع پردازشی $i$ را با $\varphi_i = \phi_i \nu_i$ نشان می‌دهیم. متغیر $u_{i,s}$ تعیین می‌کند که چه درصد از ظرفیت پردازشی منبع پردازشی $i$ به سرویس $s$ اختصاص پیدا می‌کند. واضح است که سرویس‌ها نمی‌توانند بیش‌تر از ظرفیت منابع پردازشی از آن‌ها استفاده کنند. در نتیجه رابطه زیر برای مقادیر ظرفیت‌های پردازشی تخصیص یافته به سرویس‌ها برای همه منابع پردازشی باید برقرار باشد

$$\sum_{s \in S_i} u_{i,s} \leq 1, \forall i \in C. \quad (٤)$$

این رابطه بیان می‌کند که مجموع ظرفیت پردازشی اختصاص داده شده به سرویس‌ها برای هر منبع پردازشی نباید از ظرفیت کلی آن منبع پردازشی بیشتر باشد. نرخ سرویس برای سرویس $s$ در منبع پردازشی $i$ از رابطه زیر به‌دست خواهد آمد

$$\mu_{i,s} = \frac{\varphi_i}{F_s} u_{i,s} = \zeta_{i,s} u_{i,s}, \forall s \in S, i \in C_s. \quad (٥)$$

در واقع $\mu_{i,s}$، نرخ اجرا شدن $F_s$ دستورالعمل پردازنده توسط قسمتی از کل ظرفیت پردازنده منبع پردازشی $i$ که به وسیلهٔ $u_{i,s}$ مشخص می‌شود است. در این رابطه $\zeta_{i,s}$ یک عدد ثابت است که مقدار آن برابر است با $\frac{\varphi_i}{F_s}$. با این تفاسیر رابطهٔ زیر را برای تأخیر پردازشی سرویس $s$ در منبع پردازشی $i$ وقتی از آن منبع پردازشی استفاده می‌کند می‌توان نوشت

$$d_{i,s}^{CPU} = \frac{1}{\zeta_{i,s} u_{i,s} - r_{i,s}}. \quad (٦)$$

درنتیجه رابطه زیر برای تأخیر سرویس $s$ در منبع پردازشی $i$ برقرار است

$$d_{i,s} = d_{i,s}^{net} + \frac{1}{\zeta_{i,s} u_{i,s} - r_{i,s}}. \quad (٧)$$

این رابطه به عنوان قید در یک بهینه‌سازی محدب قابل استفاده نیست. به همین دلیل سعی می‌کنیم تغییراتی در آن ایجاد کنیم تا به یک قید محدب تبدیل شود. با توجه به وجود $d_{i,s}^{max}$ در تابع هدف بهینه‌سازی، شرایط تابع $f_d$ و اینکه رابطهٔ (۳) جزء قیدهای مسئله است، می‌توانیم علامت $=$ را با علامت $\leq$ جایگزین کنیم.

$$\frac{1}{\zeta_{i,s} u_{i,s} - r_{i,s}} \leq d_{i,s} - d_{i,s}^{net}. \quad (٨)$$

چون لگاریتم یک تابع صعودی است، می‌توان از دو طرف نامساوی لگاریتم گرفت و جهت علامت نامساوی تغییری نمی‌کند. با لگاریتم گرفتن از طرفین می‌توان قید مناسب برای استفاده در بهینه‌سازی محدب را به‌دست آورد

$$-\log(\zeta_{i,s} u_{i,s} - r_{i,s}) - \log(d_{i,s} - d_{i,s}^{net}) \leq 0. \quad (٩)$$

دلیل محدب بودن رابطهٔ (۹) این است که $\zeta_{i,s}u_{i,s} - r_{i,s}$ و $d_{i,s} - d_{i,s}^{net}$ هر دو تابع محدب هستند. همچنین تابع log– هم یک تابع محدب است. با توجه به قضیه ترکیب توابع محدب [۱۶]، (۹) یک قید محدب است.

تابع مطلوبیت سرویس $s$، $U_s$، ترکیبی وزن دار از تاثیر تأخیر و نرخ پردازش اختصاص داده شده به سرویس است که در فرمول رابطهٔ(۱) ارائه شده است. رابطهٔ (۱۰) تابع هدف بهینه‌سازی را نشان می‌دهد که مجموع وزن دار $U_s$ها می‌باشد. در این رابطه $\alpha_s$ یک مقدار ثابت است که برای سرویس‌های مختلف می‌تواند متفاوت باشد و اهمیت یک سرویس را در مقایسه بقیه سرویس‌ها نشان دهد.

$$\sum_{s=1}^{N} \alpha_s\left(\omega_s f_r(r_s,R_s) + (1-\omega_s) f_d(d_s^{max})\right). \qquad (۱۰)$$

با توجه به آنچه که تا این جا گفته شد، می‌توان مسئله بهینه‌سازی را به صورت زیر نوشت

$$\max_{r_{i,s},u_{i,s},\delta_{i,s}} \alpha_s\left(\omega_s f_r(r_s,R_s) + (1-\omega_s) f_d(d_s^{max})\right)$$

*Subject to*

$$r_s = \sum_{i \in S_i} r_{i,s}, \forall s \in S \qquad (۱۱ا)$$

$$r_{i,s} \le \eta \zeta_{i,s} u_{i,s}, \forall s \in S, i \in C_s \qquad (۱۱ب)$$

$$u_{i,s} \le \delta_{i,s}, \forall s \in S, i \in C_s \qquad (۱۱ج)$$

$$d_{i,s} \le d_s^{max}, \forall s \in S, i \in C_s, if\ \delta_{i,s} = 1 \qquad (۱۱د)$$

$$-\log\left(\left(\zeta_{i,s}u_{i,s} - r_{i,s}\right)\left(d_{i,s} - d_{i,s}^{net}\right)\right) \le 0, \forall s \in S, i \in C_s, if\ \delta_{i,s} = 1 \qquad (۱۱ه)$$

$$\sum_{s \in S_i} u_{i,s} \le 1, \forall i \in C \qquad (۱۱و)$$

$$\sum_{s \in S_i} \delta_{i,s} \le Q_i, \forall i \in C \qquad (۱۱ز)$$

$$\sum_{i \in C_s} \delta_{i,s} \le Q_s, \forall s \in S \qquad (۱۱ح)$$

$$r_{i,s} \ge 0, \forall s \in S, i \in C_s \qquad (۱۱ط)$$

$$0 \le u_{i,s} \le 1, \forall s \in S, i \in C_s \qquad (۱۱ی)$$

$$\delta_{i,s} \in \{0.1\}, \forall s \in S, i \in C_s \qquad (۱۱ک)$$

در این مسئله قید (۱۱ج) باعث می‌شود که زمانی که سرویس $s$ از منبع پردازشی $i$ استفاده نمی‌کند $u_{i,s} = 0$ باشد و در صورت استفاده $u_{i,s} \le 1$ باشد. باید توجه کرد که قیدهای (۱۱ط) و (۱۱ب) باعث می‌شوند که $u_{i,s}$ها مثبت باشند. قید (۱۱ و) برای این در بهینه‌سازی حضور دارد که میزان استفاده از منابع پردازشی نمی‌تواند بیشتر از ظرفیت آن‌ها باشد. قید (۱۱ز) نشان‌دهندهٔ حداکثر تعداد سرویس‌هایی است که یک منبع پردازشی می‌تواند به آن‌ها اختصاص پیدا کند. قید (۱۱ح) برای محدود کردن حداکثر تعداد منابع پردازشی که یک سرویس می‌تواند استفاده کند است.

این مسئله بهینه‌سازی هم یک مسئله برنامه‌ریزی غیرخطی عدد صحیح مخلوط است که پیدا کردن جواب بهینه آن ساده نیست. به همین دلیل در بخش بعدی یک الگوریتم برای پیدا کردن جواب زیر بهینه آن معرفی می‌کنیم.

## ۳- الگوریتم زیربهینه

از آن‌جایی که پیدا کردن مقدار بهینه مسئله (۱۱) نیاز به زمان طولانی دارد، برای پیدا کردن جواب آن یک الگوریتم مبتنی بر تکرار معرفی می‌کنیم که در زمان کوتاهی به جواب می‌رسد. در هر تکرار الگوریتم، برای هر سرویس، یک تغییر در منابع پردازشی اختصاص یافته به آن سرویس ایجاد می‌کنیم. این تغییر می‌تواند حذف کردن، اضافه کردن یا جابه‌جایی منابع پردازشی آن سرویس باشد به شرطی که پس از تغییر، قیدهای حداکثر تعداد سرویس‌های منابع پردازشی و حداکثر تعداد منابع پردازشی سرویس برقرار باشند. این کار را در هر تکرار برای همهٔ سرویس‌ها انجام می‌دهیم و تغییر ایجاد شده در تابع هدف بهینه‌سازی با مرحله قبل را درنظر می‌گیریم. پس از انجام این کار، تغییری که بیشترین افزایش را در تابع هدف بهینه‌سازی دارد به تخصیص منابع اعمال می‌کنیم. این الگوریتم تا زمانی که مقدار افزایش تابع هدف بهینه‌سازی در هر مرحله بیشتر از مقدار مشخص $\epsilon$ باشد ادامه می‌دهیم.

الگوریتم ۱ به صورت خلاصه مراحل این الگوریتم را نشان می‌دهد. خط ۳ باعث می‌شود که الگوریتم تا زمانی که تغییرات مرحله قبل بیش از $\epsilon$ باشد، ادامه پیدا کند. سپس در خط ٤ مقدار تغییرات در مرحله جدید برابر صفر قرار داده می‌شود. در خط ٥ همه سرویس‌ها مورد بررسی قرار می‌گیرند. برای بررسی هر سرویس در خط ٦ ،همه منابع پردازشی ممکن $c_1$ و $c_2$ بررسی می‌شوند. $c_1$ منبع پردازشی است که از مراحل قبل به سرویس مورد بررسی اختصاص پیدا کرده و حذف شدن آن را بررسی می‌کنیم و $c_2$ منبع پردازشی است که اختصاص پیدا کردن

**الگوریتم ۱: الگوریتم تخصیص منابع پردازشی مبتنی بر مزایده**

```
1: ν ← −∞
2: d ← ∞
3: while d > ϵ do
4:     d ← 0
5:     for s ∈ S do
6:         for (c1, c2) ∈ (I_s ∪ {_}) × ((C_s \ I_s) ∪ {_}) do
7:             if s ∈ S_c2 then
8:                 I_s ← (I_s \ {c1}) ∪ {c2}
9:                 I_c1 ← I_c1 \ {s}
10:                I_c2 ← I_c2 ∪ {s}
11:                if |I_s| ≤ Q_s| and |I_c2| ≤ Q_c2| then
12:                    ν' = Optimal Value
13:                    if ν' > ν then
14:                        d ← d + ν' − ν
15:                        ν ← ν'
16:                        s^new ← s
17:                        c_1^new ← c1
18:                        c_2^new ← c2
19:                    end if
20:                end if
21:                I_s ← (I_s \ {c2}) ∪ {c1}
22:                I_c2 ← I_c2 \ {s}
23:                I_c1 ← I_c1 ∪ {s}
24:            end if
25:        end for
26:    end for
27:    I_s^new ← (I_s^new \ {c_1^new}) ∪ {c_2^new}
28:    I_c1^new ← I_c1^new \ {s^new}
29:    I_c2^new ← I_c2^new ∪ {s^new}
30: end while
```

آن به سـرویس را بررسـی می‌کنیم که هرکدام می‌تواند تهی باشـد. در واقع در هر بار بررسـی، منبع پردازشـی $c_1$ را از مجموعه منابع پردازشـی اختصاص پیدا کرده به سـرویس حذف کرده و $c_2$ را بـه آن اضافه می‌کنیم. اگر منبع پردازشـی $c_2$ تهی باشد یعنی فقط منبع پردازشی $c_1$ از منابع اختصاص پیدا کرده به سرویس حذف می‌شود و اگر $c_1$ تهی باشد یعنی فقط منبع پردازشی $c_2$ به سرویس اختصـاص پیدا می‌کند. برای ایـن کار در خطوط ۸ تا ۱۰ این تغییر را به صـورت موقت در اختصاص منابع اعمال می‌کنیـم و در خطوط ۲۱ تـا ۲۳ اختصـاص منابع را به حالت قبل باز می‌گردانیم. پس از بررسـی معتبر بودن این تغییـرات در خط ۱۱، در خـط۱۲ مقدار بهینه تابع هدف با منابع اختصاص یافته جدید محاسـبه می‌شود و نتیجه در $v'$ قرار می‌گیرد. در خط ۱۳، $v'$ با بیشـترین مقداری که تا الان به‌دست آمده، $v$، مقایسه می‌شود و در صورتی که $v'$ از آن بیشـتر بود، مقادیر $v$، $d$ و بهترین تخصیص منابع به روز رسـانی می شوند (خطوط ۱۴ تا ۱۸). در پایان هر مرحلـه تخصیص منابعی که بیشـتری افزایش را در تابع هدف مسئله داشته، اعمال می‌شود (خطوط ۲۷ تا ۲۹).

در ادامـه به بررسـی پیچیدگـی الگوریتـم زیربهینه می‌پردازیم. در ابتدای شـروع الگوریتم، نرخ انتخابی همهٔ سرویس‌ها صفر اسـت و هیچ منبع پردازشی اختصاص پیدا نکرده اسـت. در نتیجه مقدار تابع هدف بهینه‌سـازی $-\sum_{s=1}^{N}\alpha\omega_s R$ اسـت. علاوه براین تابع هدف بهینه‌سـازی همواره یک عدد منفی اسـت. با در نظـر گرفتن این‌که در هر تکـرار، مقدار تابع هـدف حداقل به انـدازهٔ $\epsilon$ افزایش پیدا می‌کنـد، تعداد تکرارهای الگوریتـم نمی‌تواند بیش از $\sum_{s=1}^{N}\alpha\omega_s R/\epsilon$ باشد.

اگر فـرض کنیم در مرحله‌ای دلخـواه از این الگوریتم سـرویس $s$ به $m$ منبع پردازشـی اختصـاص پیدا کرده باشد و $m'$ تعداد منابع پردازشی اختصاص پیدا نکرده به سرویس $s$ باشـد، باید $(m+1)\times(m'+1)-1$ بار مسئله بهینه‌سازی برای سـرویس $s$ حل شود. بنابراین، می‌توان نتیجه گرفت که تعداد دفعاتی که مسئله بهینه‌سازی در هر بار تکرار الگوریتم حل می‌شـود، کم‌تر از $N(M+1)^2$ است. در نتیجه تعداد دفعاتی که مسئله بهینه‌سازی حل می‌شود کم‌تر از $N(M+1)^2\sum_{s=1}^{N}\alpha_s\omega_s R_s/\epsilon$ اسـت. از این رابطه واضح اسـت که با افزایش $\epsilon$ انتظار می‌رود الگوریتم زودتر پایان یابد. باید توجه داشت که $(m+1)\times(m'+1)-1$ حد بالای تعداد دفعات حل شدن مسئله بهینه‌سازی برای هر سرویس را نشان می‌دهد و محدود بودن انتخاب‌های سرویس‌ها و منابع پردازشـی(مجموعه‌های $S$ و $C$ برای هر سرویس و منبع پردازشی) باعث می‌شود که تعداد واقعی، کم‌تر از این تعداد باشد.

## ۴- نتایج شبیه‌سازی

یک شبکه با تعداد $N$ سرویس و $M$ منبع پردازشی که در داخل یک دایره به شعاع $R=100m$ به صورت تصادفی پراکنده شده‌اند را در نظر می‌گیریم. برای محاسبه تأخیر،

تأخیر ناشی از مسیریاب‌های شبکه را در نظر می‌گیریم و فرض می‌کنیم این تأخیر بین ۱۵ تا ۲۰ میلی ثانیه است. فرض می‌کنیم ظرفیت پردازشی منابع پردازشی لبه شبکه به صورت تصادفی در محدودهٔ $[1200,1400]$ توزیع شده‌اند. برای ظرفیت پردازشی منابع پردازشی ابری هم مقدار ثابت $10000$ را در نظر می‌گیریم. برای $\alpha_s$ فرض می‌کنیم که به صورت تصادفی در بازه $[1,10]$ توزیع شده است. برای رسیدن به نتایج معنی دار، $\beta = (1-\omega_s)/\omega_s$ را به صورت یکنواخت در بازه $[30,70]$ در نظر می‌گیریم.

برای شبیه‌سازی توابع زیر را فرض می‌کنیم. قابل ذکر است الگوریتم با تغییر توابع نیز همچنان کارکرد درستی دارد.

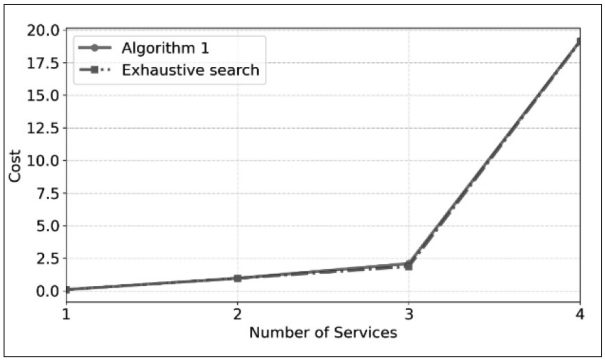
$$f_r(r_s, R_s) = -|R_s - r_s| \quad (۱۲)$$

$$f_d\left(d_{i,s}^{CPU} + d_{i,s}^{net}\right) = -d_{i,s}^{CPU} - d_{i,s}^{net} \quad (۱۳)$$

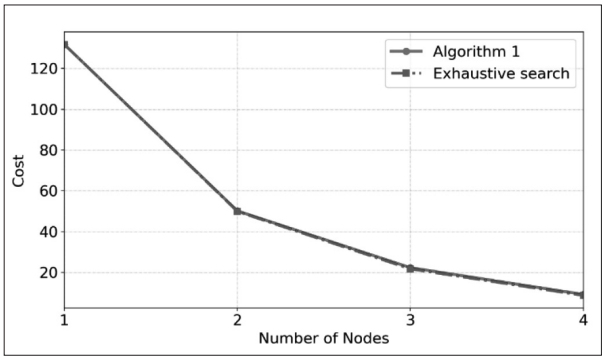
برای بررسی کارآمدی الگوریتم مقدار تابع هدف را با روش جستجوی فراگیر برای ٤ سرویس و ٤ گره پردازشی مقایسه می‌کنیم. در شکل ۲ تعداد سرویسی‌ها را از ۱ تا ٤ افزایش می‌دهیم و در شکل ۳ تعداد گره‌های پردازشی را از ۱ تا ٤ افزایش می‌دهیم. همانطور که از این اشکال مشخص است نتیجه حاصل از الگوریتم ۱ بسیار به مقدار بهینه نزدیک است.

فرض می‌کنیم تعداد سرویس‌ها $N = 15$ باشد و تعداد منابع پردازشی لبه شبکه $(M)$ از ۱۲ تا ۲۰ تغییر کند. دو فرانامهٔ با و بدون حضور منبع پردازشی ابری را در نظر می‌گیریم و میانگین هزینه سرویس‌ها $(-U/N)$ را رسم می‌کنیم. انتظار داریم که با افزایش تعداد منابع پردازشی، میانگین هزینه سرویس‌ها کاهش پیدا کند. چرا که قدرت پردازشی بیشتری در شبکه وجود دارد و می‌تواند باعث کاهش تأخیر یا اختلاف نرخ مطلوب و نرخ بهینه شود. جهت مقایسه، موارد را با حالتی که منابع ابری نیز در شبکه وجود دارد مقایسه می‌کنیم. منابع ابری دارای قدرت پردازشی بیشتری هستند ولی تأخیر بالاتری دارند.

نتیجه این شبیه‌سازی در شکل ٤ آورده شده‌است.



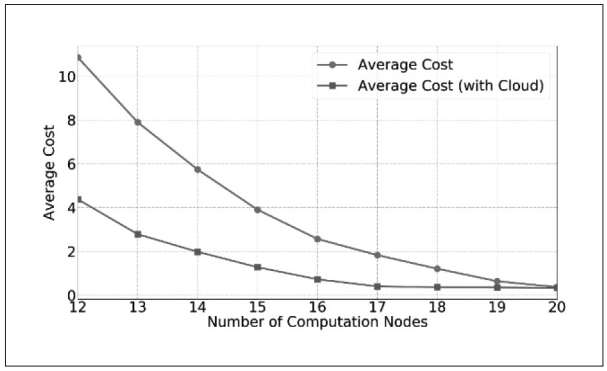

شکل ۲: مقایسه مقدار بهینه حاصل از روش جستجوی فراگیر و الگوریتم ۱ وقتی تعداد سرویس‌ها افزایش پیدا می‌کند.



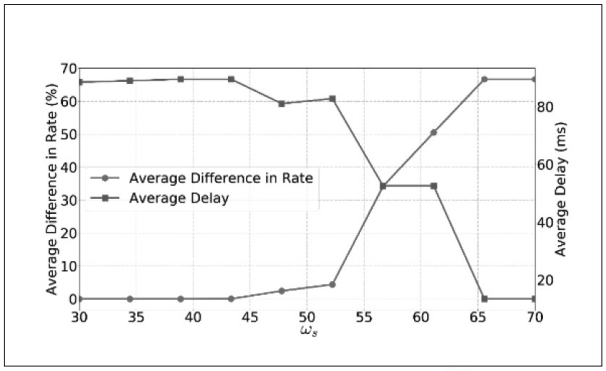

شکل ۳: مقایسه مقدار بهینه حاصل از روش جستجوی فراگیر و الگوریتم ۱ وقتی تعداد گره‌های پردازشی افزایش پیدا می‌کند.

همانطور که در شکل هم مشخص است با افزایش تعداد منابع پردازشی، میانگین هزینه سرویس‌ها کاهشی است. علاوه بر این، میانگین هزینهٔ سرویس‌های فرانامه‌ای که منبع پردازشی ابری در آن وجود داشت، همواره کم‌تر از فرانامهٔ بدون منبع رایانش ابری است. دلیلش آن است که هرچه تعداد منابع پردازشی بیشتر باشد، ظرفیت پردازشی کل بیشتر می‌شود و برای سرویس‌ها انتخاب بیش‌تری وجود دارد. همچنین، با افزایش تعداد منابع پردازشی لبه شبکه، اختلاف این دو فرانامه کاهش پیدا می‌کند تا در تعداد ۲۰ منبع پردازشی لبه شبکه، تقریبا صفر می‌شود. دلیلش آن است که با افزایش منابع پردازشی لبه شبکه همه پردازش‌ها در لبه شبکه انجام شده و تاثیر وجود منبع پردازشی ابری در میانگین هزینه‌ها به صفر می‌رسد.

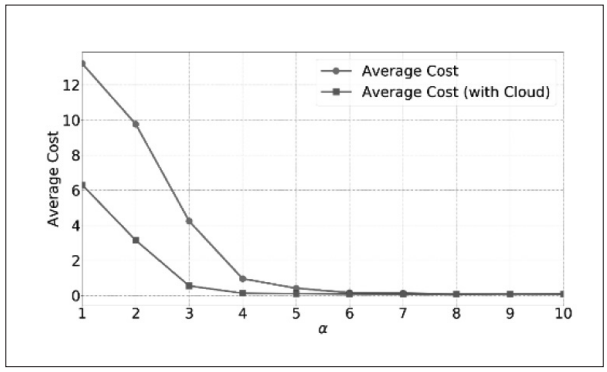
اثر پارامتر $\omega_s$ در این روش تخصیص منابع در شکل ۵ بررسی شده‌است. در این شکل اثر $\omega_s$ روی اختلاف نرخ بهینه و مطلوب و بیشترین تأخیر برای یک سرویس در بازه $\omega_s \in [30,70]$ رسم شده‌است. همان‌طور که بیان شد $\omega_s$



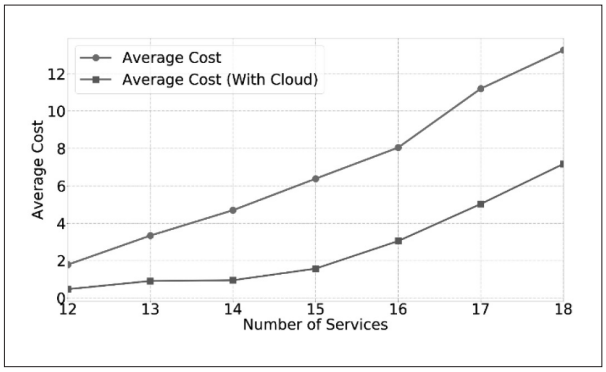

شکل ۴: میانگین هزینه سرویس‌ها در برابر تعداد منابع پردازشی در لبه شبکه




شکل ۵: تاثیر $\omega_s$ بر تأخیر و اختلاف نرخ بهینه با نرخ مطلوب برای یک سرویس




شکل ۶: تاثیر $\alpha_s$ بر میانگین هزینه سرویس‌ها




شکل ۷: تاثیر تعداد سرویس‌ها بر میانگین هزینه سرویس‌ها

نسبت اهمیت تأخیر به اختلاف نرخ بهینه و مطلوب است. در نتیجه انتظار داریم با افزایش $\omega_s$ اختلاف نرخ، افزایش پیدا کرده و تأخیر کاهش پیدا کند که شکل هم همین را نشان می‌دهد.

در شکل ٦ تاثیر $\alpha_s$ بر روی هزینه یک سرویس بررسی شده‌است. به دلیل تاثیر مستقیم $\alpha_s$ در $U_s$، مقدار $-U_s/\alpha_s$ رسم شده‌است. افزایش $\alpha_s$ باعث افزایش تاثیر $U_s$ در تابع هدف بهینه‌سازی می‌شود. پس انتظار داریم، افزایش $\alpha_s$ باعث کاهش $-U_s/\alpha_s$ بشود. این کاهش معادل کاهش تأخیر و اختلاف نرخ بهینه و مطلوب سرویس $s$ است.

در شبیه‌سازی بعدی فرض می‌کنیم که تعداد منابع پردازشی ثابت و برابر ۱۵ باشد و تعداد سرویس‌ها در بازهٔ [12, 18] تغییر کند. به دلیل این که تعداد منابع پردازشی ثابت است، انتظار داریم افزایش تعداد سرویس‌ها باعث افزایش هزینه میانگین سرویس‌هایی که از ابتدا در سیستم وجود داشتند شود. در شکل ۷ میانگین هزینهٔ ۱۲ سرویس اول را برای تعداد سرویس‌های شبیه‌سازی شده رسم کرده‌ایم. در این شکل مطابق انتظار با افزایش تعداد سرویس‌ها، میانگین هزینه سرویس‌هایی که از ابتدا در سیستم قرار داشتند افزایش پیدا کرده‌است. علاوه بر این مانند بقیه شبیه‌سازی‌ها، در فرانامه‌ای که دارای منبع پردازشی ابری است، میانگین هزینه ۱۲ سرویس اول کم‌تر از سناریوی بدون منبع پردازشی ابری است. هم‌چنین با افزایش تعداد سرویس‌ها، اختلاف این دو فرانامه افزایش پیدا می‌کند، چراکه منبع پردازشی ابری ظرفیت پردازشی بالایی دارد و سرویس‌ها با استفاده از آن هزینهٔ خود را کاهش می‌دهند.

## ۵-جمع‌بندی و نتیجه‌گیری

در این مقاله مسئله تخصیص منابع پردازشی به صورت چند به چند برای سرویس‌های اینترنت اشیاء بررسی شد. هر سرویس می‌تواند از چند منبع پردازشی استفاده کند و منابع پردازشی هم می‌توانند به چند سرویس اختصاص پیدا کنند. ابتدا این مسئله به صورت یک مسئله برنامه‌ریزی

غیرخطی عدد صحیح مخلوط فرمول بندی شـد. با توجه به پیچیدگی حل مسـئله ارائه شـده، یک روش زیر بهینه برای حل آن ارائه شـد. با مقایسه این روش با مقدار بهینه مشخص شـد که این روش که دارای پیچیدگی محاسباتی بسـیار کم‌تری نسبت به روش جسـتجوی فراگیر است، برای تعداد سـرویس‌ها و گره‌های پردازشی کم به مقدار بهینه بسیار نزدیک اسـت. نتایج شبیه‌سازی با این روش نشـان می‌دهد که در این روش بـا افزایش تعداد گره‌های لبهٔ شبکه، هزینه سرویس‌ها کاهش پیدا می‌کند. همچنین با اسـتفاده از این روش، در حضور منابع پردازشـی ابری، با افزایش تعداد منابع پردازشـی لبه شبکه، سرویس‌های بیشـتری پردازششان را به لبه شـبکه انتقال می‌دهند. تا جایی که سـرویس‌ها به طور کامل از منابع پردازشی لبه شـبکه استفاده می‌کنند و از منابع پردازشی ابری استفاده نمی‌شـود. برای کارهای آینده می‌تـوان روی روش‌هایی کار کرد که برای حل مسـئله در ابعاد بسیار بالا با سرعت بالا به نتیجه می‌رسـند یا به صورت توزیع شده جواب را به‌دست می‌آورند.

## مراجع


1. Cisco, “Cisco Edge-to-Enterprise IoT Analytic for Electric Utilities,” 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/big-data/solution-overview-c22-740248.pdf}.

2. W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, “Edge computing: Vision and challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, 2016.

3. S. Yi, Z. Hao, Z. Qin and Q. Li, “Fog computing: Platform and applications,” in Proc. of IEEE HotWeb, Washington, DC, 2015.

4. T. Pfandzelter and D. Bermbach, “tinyFaaS: A lightweight faas platform for edge environments,” in 2020 IEEE International Conference on Fog Computing (ICFC), 2020.

5. K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai and M. Satyanarayanan, “Towards wearable cognitive assistance,” in Proc. of international conference on Mobile systems, applications, and services (MobiSys), Bretton Woods, NH, 2014.

6. B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke and O. H. Hoe, “Evaluation of docker as edge computing platform,” in Proc. of ICOS, Melaka, Malaysia, 2015.

7. S. K. Viswanath, C. Yuen, W. Tushar, W.-T. Li, C.-K. Wen, K. Hu, C. Chen and X. Liu, “System design of the internet of things for residential smart grid,” IEEE Wireless Communications, vol. 23, no. 5, pp. 90-98, 2016.

8. R. Morabito and N. Beijar, “Enabling data processing at the network edge through lightweight virtualization technologies,” in Proc. of SECON Workshops, London, United Kingdom, 2016.

9. R. Morabito, “Virtualization on internet of things edge devices with container technologies: a performance evaluation,” IEEE Access, vol. 5, pp. 8835-8850, 2017.

10. R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, “Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption,” IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, 2016.

11. M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario and A. Morell, “IoT-cloud service optimization in next generation smart environments,” IEEE JSAC, vol. 34, no. 12, pp. 4077-4090, 2016.

12. H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu and Z. Han, “Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1204-1215, 2017.

13. L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, “Multiobjective optimization for computation offloading in fog computing,” IEEE Internet of Things Journal, vol. 5, no. 1, pp. 283-295, 2017.

14. Y. Gu, Z. Chang, M. Pan, L. Song and Z. Han, “Joint radio and computational resource allocation in IoT fog computing,” IEEE TVT, vol. 67, no. 8, pp. 7475-7484, 2018.

15. L. Kleinrock, “Queuing Systems,” Wiley-Interscience, 1975.

16. S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge university press, 2004.