علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۶، شماره ۲، ۱۳۹۷
صفحات ۱- ۹
مقاله عادی

ترکیب سرویس‌های وب مبتنی بر درخت K-d

طاهره داودی سیما عمادی

دانشکده فنی مهندسی، دانشگاه آزاد اسلامی واحد یزد، یزد، ایران

چکیده

در ترکیب سرویس با انتخاب سرویس‌های واقعی، نیازمندی‌های وظیفه‌مندی و غیر وظیفه‌مندی لحاظ می‌گردند و هنگامی که بسیاری از سرویس‌ها با وظیفه‌مندی معادل در دسترس هستند، ویژگی‌های کیفیت سرویس (همچون تاخیر، قیمت، دسترس‌پذیری) بسیار مورد توجه قرار می‌گیرند. همانطور که تعداد سرویس‌های توزیع شده به‌خصوص در ابر، به سرعت در حال افزایش هستند تاثیر کیفیت سرویس در شبکه نیز در حال افزایش است. با این وجود رویکردهای فعلی تمایزی ما بین کیفیت سرویس، سرویس‌های خودشان و کیفیت سرویس شبکه قائل نمی‌شوند و تاخیر محاسبه شده با تاخیر واقعی متفاوت است، در نتیجه کیفیت سرویس کمتر از حد مطلوب می‌شود. روش‌های گوناگونی برای حل این مشکل مطرح شده که از جمله آنها یک رویکرد آگاه از پارامترهای کیفی شبکه است. در این رویکرد مشکلاتی وجود دارد که از جمله آنها عدم‌سازگاری سرویس‌ها در ترکیب و زمانبر بودن اجرای الگوریتم است. جهت رفع مشکلات ذکر شده، الگوریتم پیشنهادی در این تحقیق، با استفاده از درخت k-d و روش نزدیکترین همسایگی، ترکیب سرویس مناسب‌تری را پیدا می‌کند. درخت k-d با در نظر گرفتن پارامتر کیفی تاخیر و مختصات مکانی کاربر و استفاده از لیست‌های خطی ایجاد شده از سرویس‌ها، جستجو را برای یافتن بهترین همسایگی جهت ترکیب و همچنین رفع ناسازگاری میان سرویس‌ها در حین ترکیب، انجام می‌دهد. نتایج حاصل از ارزیابی و تحلیل روش پیشنهادی، بیانگر برقراری سازگاری میان سرویس‌ها و یافتن ترکیب پویا همراه با کم‌ترین زمان اجرایی است.

کلمات کلیدی: ترکیب سرویس، کیفیت سرویس، درخت k-d، سرویس‌های ناسازگار.

۱- مقدمه

امروزه ترکیب سرویس‌های وب یک روش امیدبخش در ادغام برنامه‌های کسب و کار در داخل و سرتاسر مرزهای سازمانی است که در آن سرویس‌های وب ریز دانه مستقل می‌توانند برای تبدیل به یک سرویس دانه درشت، با یکدیگر ادغام شوند. نیاز به ترکیب سرویس زمانی بیشتر احساس می‌شود که درخواست یک مشتری نمی‌تواند به وسیله هر سرویس فردی برآورده شود، پس سرویس‌های وب موجود می‌توانند درون یک سرویس وب مرکب، ترکیب شوند. یک بحث مهم و چالش برانگیز در ترکیب سرویس وب، چگونگی ملاقات ویژگی‌های کیفی سرویس[1] موردنیاز مشتری است. وقتی تعداد زیادی از سرویس‌های وب در دسترس وجود دارند، پیدا کردن یک مسیر اجرایی از سرویس‌های وب مرکب که درخواست داده شده را راضی کند آسان نیست و فضای جستجو برای چنین مساله‌ای بصورت نمایی در حال افزایش است. برای پاسخگویی به این مشکل، اکثر روش‌های ترکیب موجود در محاسبات سرویس‌گرا از برنامه‌نویسی خطی[2] در پیدا کردن راه‌حل بهینه استفاده می‌کنند که اگر چه برای ترکیبات در مقیاس کوچک راه‌حل مفیدی است اما برای مشکلات ترکیب با مقیاس بزرگ مانند ترکیبات در محاسبات ابر، کارایی بسیار پایینی دارد.

Klein و همکاران [۱] یک روش خودتطبیق به نام sanGA برای ترکیب سرویس شبکه آگاه پیشنهاد داده‌اند که کیفیت هر یک از سرویس‌ها و کیفیت سرویس شبکه را بطور مستقل بکار می‌گیرد. در این رویکرد ابتدا یک مدل شبکه به‌منظور تخمین تاخیر شبکه میان سرویس‌های اختیاری و کاربران نهایی ایجاد می‌شود، سپس الگوریتم ژنتیک خودتطبیق برای پیدا کردن یک ترکیب با تاخیر کم بکار می‌رود. ارائه‌دهندگان این رویکرد، قابلیت مقیاس‌پذیری را جهت یافتن سرویس‌هایی که به مکان‌های شبکه اصلی یا مسیرهای شبکه نزدیک هستند، به مدل شبکه ژنتیکی خود اضافه کرده‌اند، که روش قبلی آنها [۱۲] را بهبود بخشیده

است. این رویکرد می‌تواند برای سرویس‌های توزیع شده در شبکه و مقیاس وسیع (مانند ابر) استفاده شود.

Ishikawa و همکاران [۲] یک روش اکتشافی بر پایه الگوریتم کوله‌پشتی[۳] ارائه داده‌اند که استفاده از اختلاف معیار اولیه[۴] محاسبه شده با برنامه‌نویسی خطی را موثر می‌سازد و روی یک فضای جستجوی کاهش یافته عمل می‌کند. ارزیابی‌ها نشان می‌دهد که این روش راه‌حلی تقریباً بهینه را در تنها کسری از زمان مورد نیاز الگوریتم کوله‌پشتی استاندارد، ارائه می‌کند و همچنین پیچیدگی زمانی کمتری از الگوریتم کوله‌پشتی استاندارد دارد. بنابراین، این روش هنگام استفاده از الگوریتم‌های اکتشافی، اختلاف معیار اولیه خوب و کاهش فضای جستجو را بسیار مهم می‌داند.

Ye و همکاران [۳] روشی بر مبنای الگوریتم ژنتیک برای ترکیب سرویس‌ها در محاسبات ابری ارائه نمودند که در این روش مشکل زمانبندی در موقعیتی که سرویس‌ها می‌توانند روی ماشین مجازی پیاده شوند، حل شده است. با این حال واضح نیست که رویکرد بتواند محاسبات کیفیت سرویس وابسته به ورودی و زمان انتقال شبکه را بررسی کند، زیرا نویسندگان روشی برای اعمال کیفیت سرویس ارائه نکرده‌اند.

Wang و همکاران [۴] یک مدل ترکیب سرویس ارائه نمودند که کیفیت سرویس را هم برای سرویس‌ها و محیط شبکه ابری در نظر می‌گیرد. در این رویکرد، ترکیب سرویس‌های وب براساس الگوریتم ژنتیک انجام می‌شود و راهکاری برای فراهم‌کنندگان سرویس که می‌خواهند نقض تفاهم‌نامه سطح سرویس را حداقل کنند، ارائه شده است. در این روش الگوریتم ژنتیک بر پایه Skyline [۵] ارائه می‌شود و معیارهای کیفیت سرویس چندگانه را مد نظر قرار می‌دهد. این روش در مقایسه با برخی کارهای مرتبط از جمله [۶]، [۷] که تنها بر روی کیفیت سرویس‌های خاص تمرکز دارند، مقیاس‌پذیرتر است و می‌تواند به یک نتیجه بهینه از نظر هزینه و زمان محاسباتی نزدیک باشد. یکی از معایب این روش این است که خطاهای استثناء ناسازگاری را حین اجرای سرویس مرکب در نظر نمی‌گیرد.

Honiden و همکارانش [۸] یک رویکرد جهت ارزیابی کیفیت سرویس وابسته به زمان اجرایی معرفی کرده‌اند. این رویکرد یک الگوریتم ترکیب بر مبنای جستجوی اکتشافی ارائه می‌دهد که زمان شروع اجرای هر سرویس را محاسبه می‌کند. بنابراین بخوبی می‌تواند برای محاسبه کیفیت سرویس وابسته به زمان اجرایی استفاده شود.

Rosenberg و همکارانش [۹] یک چارچوب فرااکتشافی برای مشکل ترکیب آگاه از کیفیت سرویس ارائه می‌کنند که در تشخیص و پشتیبانی از سه مورد زیر رویکرد جدیدی است:

۱. روشی انعطاف‌پذیر در تعیین نیازمندی‌های کیفیت سرویس با استفاده از محدودیت‌های سلسله مراتبی.
۲. یک مدل کیفیت سرویس توسعه‌پذیر و تعریف شده توسط کاربر.
۳. بهبود جهش و تولید همسایه‌ی اکتشافی.

در این روش سه نسخه از فرااکتشافی‌های شناخته شده با نام‌های GA[۵]، SA[۶] و TS[۷] را پیاده‌سازی می‌کنند که با توجه به بهینه‌سازی زمان، خروجی بهتری نسبت به روش‌های موجود بخصوص برای سرویس‌های مقیاس بزرگ دارد. این روش یک چارچوب مفید برای ترکیب زمان اجرا و ترکیب مجدد در محیط‌های سازمانی است، زیرا برای ترکیب‌های کوچک و متوسط سربار کمی متحمل می‌شود. بعلاوه در حوزه‌هایی که سیستم‌های مقیاس بزرگ غالب هستند، همچون علوم محاسباتی یا محاسبات علمی این روش بخوبی قابل اجرا است.

Chen و همکارانش [۱۰] ترکیب سرویس وب آگاه از کیفیت سرویس را مبتنی بر راه‌حل‌های بهینه پارتو انجام داده‌اند. در این روش شش دسته از ویژگی‌های کیفیت سرویس و توابع ادغام آنها بررسی شده است. همچنین از پارتو و یک الگوریتم توزیع شده، جهت کاهش فضای جستجو استفاده شده است. استفاده از مزایای هرس اجزاء بصورت چند-سطحی و موازی‌سازی، موجب شده است که این الگوریتم بتواند بهبود قابل توجهی در کارایی داشته باشد در حالی که بهینگی ترکیب سرویس را نیز تضمین می‌کند. انتظار می‌رود این کار یک چارچوب نظری و راه‌حل‌های عملی برای ترکیب سرویس در سیستم‌های وب سرویس مقیاس بزرگ فراهم کند.

Banerjee و همکاران [۱۱] یک مکانیزم تقریبی و مقیاس‌پذیر برای ترکیب سرویس وب ارائه داده‌اند. این رویکرد یک مکانیزم ترکیب سرویس آگاه از کیفیت ارائه می‌دهد که تعادلی میان زمان اجرا و تضمین کیفیت سرویس برقرار می‌کند. این روش سه گام اساسی دارد:

۱- ایجاد یک گراف متصل بین سرویس‌ها در گام پیش پردازش.
۲- تولید زیرگراف برای ارزیابی پرس‌وجوها.
۳- یافتن یک راه‌حل جهت انتخاب مسیر آگاه از کیفیت سرویس.

نتایج حاصل از ارزیابی این رویکرد نشان می‌دهند که راه‌حل پیشنهادی بسیار سریع‌تر از راه‌حل‌های بهینه موجود عمل می‌کند.

Honiden و همکاران [۱۲] یک روش آگاه از شبکه برای ترکیب سرویس در ابر پیشنهاد داده‌اند که شامل مراحل ایجاد مدل شبکه، محاسبه کیفیت سرویس آگاه از شبکه و الگوریتم انتخاب آگاه از شبکه براساس الگوریتم ژنتیک است. در مدل شبکه رویکرد پیشنهادی از سیستم مختصات شبکه جهت تخمین تاخیر میان دو نقطه استفاده می‌شود و از یک طرح درهم‌سازی حساس به محل، جهت یافتن سرویس‌های نزدیک به مکان شبکه‌ی اصلی و مسیرهای شبکه‌ای استفاده می‌شود. نتایج نشان می‌دهند که رویکرد پیشنهادی به یک راه‌حل نزدیک به بهینه با تاخیر کم دست پیدا می‌کند. از معایب این روش این است که کیفیت سرویس‌های متعددی را در نظر نمی‌گیرد و به دلیل انجام جستجوی کامل جهت ترکیب، زمان محاسباتی بالایی نیاز دارد.

در مقابل این روش‌های موجود، الگوریتم ژنتیک و روش‌های اکتشافی برای تکرار در یافتن راه‌حل‌های نزدیک به بهینه در فضای جستجوی بزرگ در بسیاری از تحقیق‌های گذشته استفاده شده‌اند.

با وجود بسیاری از روش‌های ارائه شده در این زمینه، نتایج نشان می‌دهند هنوز مشکلاتی برای حل مساله ترکیب سرویس‌ها براساس ویژگی‌های کیفی وجود دارد. در همین راستا، این پژوهش به چگونگی استفاده از درخت k-d[۸] برای حل مشکل ناسازگاری داده‌های ورودی در ترکیب سرویس، پیدا کردن ترکیب مناسب با کمترین تاخیر[۹] و مقیاس‌پذیر بودن روش پرداخته است، که در ادامه، به تفصیل بیان شده است.

ساختار این مقاله به شرح زیر ادامه می‌یابد؛ در بخش دوم، رویکرد پیشنهادی به تفصیل شرح داده می‌شود، در بخش سوم ارزیابی روش و الگوریتم گزارش خواهد شد، در بخش آخر نیز نتیجه‌گیری و کارهای آتی عنوان می‌شود.

۲- روش پیشنهادی

هر سرویس انتزاعی به تعدادی سرویس واقعی با وظیفه‌مندی‌های معادل متصل است، بنابراین سرویس‌های واقعی، قابل جایگزین با سرویس‌های انتزاعی هستند. انتخاب میان سرویس‌های واقعی بوسیله خصوصیات غیر وظیفه‌مندی‌شان که به عنوان ویژگی‌های کیفیت سرویس همچون قیمت، زمان پاسخ، دسترس‌پذیری و غیره شناخته می‌شوند، انجام می‌شود. از طرفی کاربر ممکن است محدودیت‌های خاصی برای برخی از ویژگی‌های کیفی سرویس درخواستی خود داشته باشد؛ یا فراهم‌کننده سرویس، مقدار تخمین زده شده برای برخی ویژگی‌های کیفیت سرویس را بعنوان بخشی از قرارداد با کاربران بالقوه در تفاهم‌نامه سطح سرویس

قرار دهد. یافتن ترکیب مناسب با ویژگی‌های کیفی درخواستی کاربر، ترکیب سرویس آگاه از کیفیت نامیده می‌شود که یک مشکل ترکیب سرویس است. روش‌های گوناگونی برای حل این مشکل مطرح شده که از جمله آنها [۱] است که یک رویکرد آگاه از پارامترهای کیفی شبکه است.

این رویکرد کیفیت سرویس در سرویس‌ها و شبکه را بطور مستقل و با استفاده از الگوریتم ژنتیک خودتطبیق، جهت یافتن یک ترکیب با تاخیر کم، استفاده کرده است. در این رویکرد مشکلاتی وجود دارد که از جمله آنها ناسازگاری بین سرویس‌ها در ترکیب و زمانبر بودن اجرای الگوریتم است. این پژوهش برای حل مشکل ذکر شده با استفاده از درخت k-d الگوریتمی را جهت بهبود محدودیت‌های این روش ارائه می‌دهد که به تفصیل شرح داده شده است.

روش‌پیشنهادی، شامل ۲ فاز پیش‌پردازش و پردازش است که در ادامه شرح داده می‌شود.

۲-۱- فاز پیش‌پردازش

فاز پیش‌پردازش، کلیه‌ی عملیاتی است که قبل از ترکیب سرویس باید انجام شود تا فرایند ترکیب براساس درخواست و پارامترهای کیفی مورد نظر کاربر با حفظ محدودیت‌ها انجام شود. در این مرحله با توجه به درخواست کاربر، سرویس‌ها از مخزن پایگاه داده UDDI[۱۰] که حاوی اطلاعات سرویس‌ها از جمله آدرس سرویس، پارامترهای کیفی و مختصات مکانی سرویس هستند، فراخوانی می‌شوند. سرویس‌های فراخوانی شده از UDDI، کلیه سرویس‌هایی هستند که دارای پارامترهای ورودی یا پارامترهای خروجی مشابه درخواست کاربر هستند. در ادامه کاربر با توجه به وظیفه‌مندی سرویس مورد انتظار خود، پارامترهای ورودی و خروجی مورد نظر خود را درخواست می‌کند و مختصات مکانی خود را که بصورت مختصات دوبعدی x, y است را وارد می‌کند.

۲-۲- فاز پردازش

در فاز پیش‌پردازش، انجام فیلترینگ سرویس‌ها براساس درخواست کاربر موجب کاهش تعداد سرویس‌های کاندید، در ترکیب سرویس‌ها می‌شود. فاز پردازش مدل پیشنهادی، با استفاده از درخت‌های k-d و لیست‌های خطی، عملیات ترکیب سرویس را بصورت پویا همراه با رفع مشکل ناسازگاری سرویس‌ها طبق مراحلی که در ادامه ذکر شده است، انجام می‌دهد.

جدول ۱- نمایش درخت‌های k-d موجود در روش پیشنهادی

$k\text{-}d^1$	براساس List1	درخت $k\text{-}d^1$ براساس List1 جهت مرتب‌سازی بر مبنای مختصات مکانی کاربر ایجاد شده است.
$k\text{-}d^2$	براساس List2	درخت $k\text{-}d^2$ براساس List2 جهت مرتب‌سازی بر مبنای مختصات مکانی کاربر ایجاد شده است.
$k\text{-}d^3$	فیلترینگ سرویس‌ها	درخت $k\text{-}d^3$ براساس کل سرویس‌های بارگذاری شده از مخزن پایگاه داده جهت انجام جستجوی نزدیکترین همسایه ایجاد شده است.

۲-۲-۱- پالایش سرویس‌ها با استفاده از درخت k-d

پس از فراخوانی سرویس‌ها براساس درخواست کاربر، طبق جدول ۱ یک درخت k-d با نام k-d3 جهت نگهداری تمام سرویس‌های فیلتر شده و انجام عمل جستجوی نزدیکترین همسایه، ساخته می‌شود. در ادامه، ابتدا ساختار درخت جستجوی دودویی استاندارد بیان، سپس جهت وضوح بیشتر، مراحل تشکیل درخت جستجوی دودویی سه‌بعدی همراه با مثال شرح داده خواهد شد.

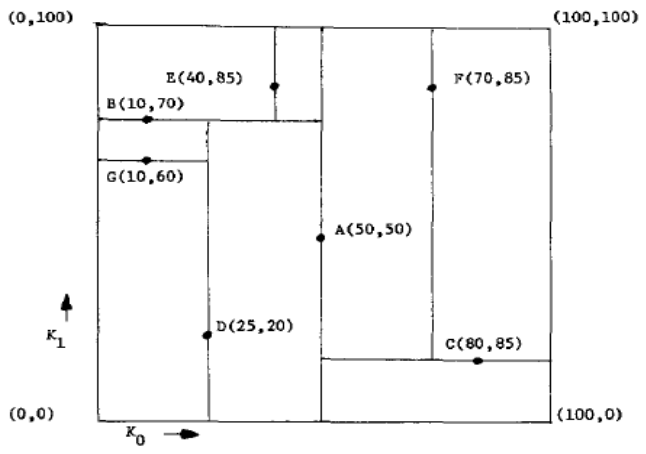
- **ساختار درخت جستجوی دودویی چندبعدی استاندارد**

روش‌های بسیاری برای ساخت یک سیستم بازیابی اطلاعات که توانایی بکار بردن پرس‌وجوهای انجمنی را داشته باشد، وجود دارند، یکی از این روش‌ها نوع جدیدی از ساختار داده است که درخت جستجوی چندبعدی یا k-d tree (k ابعاد جستجو) نامیده می‌شود. ساختار درخت k-d بدین‌گونه است که اگر یک فایل به عنوان یک درخت k-d نمایش داده شود، پس هر رکورد در فایل به عنوان یک گره در درخت ذخیره می‌شود. علاوه بر کلیدهای k که رکورد را تشکیل می‌دهند، هر گره حاوی ۲ اشاره‌گر است که یا مقدار NULL یا آدرس گره‌ی دیگری در درخت k-d را دارد (هر اشاره‌گر می‌تواند به عنوان یک زیردرخت مشخص در نظر گرفته شود). هر گره‌ی غیربرگ می‌تواند یک مولد جداساز ابر صفحه، که فضا را به دو قسمت تقسیم می‌کند، باشد که آن گره یک عدد صحیح بین ۰ و K-1 را شامل می‌شود. کلیدهای k برای گره P بصورت $K_0(P), \ldots, K_{K-1}(P)$ نام‌گذاری می‌شوند، اشاره‌گرهای گره شامل اشاره‌گرهای چپ LOSON(P) و اشاره‌گرهای راست HISON(P) خواهند بود و یک جداساز ابر صفحه با نام DISC(P) برای آن گره تعریف می‌شود. مفهوم این تعاریف اعمال شده توسط درخت k-d این است که برای گره ریشه P در درخت k-d، اگر j بعنوان جداساز ابر صفحه باشد، پس برای هر گره Q در اشاره‌گر چپ درخت، رابطه $k_j(Q)<k_j(P)$ و همچنین، برای هر گره R در اشاره‌گر راست درخت، رابطه $k_j(R)>k_j(P)$ باید برقرار باشد. در اینجا هنگامی که دو کلید k_j با هم برابر باشند، برای بازگرداندن مقدار LOSON یا HISON، یک تابع تصمیم بصورت SUCCESSOR(P,Q) تعریف می‌شود که روش انتخاب برای این تابع می‌تواند بصورت اختیاری تعریف شود [۱۳].

همه گره‌ها در هر سطح معین از درخت، جداساز هم‌اندازه با سطح‌شان دارند. طبق تعریف، ریشه گره‌ها جداساز صفر دارد و دو فرزندش جداساز یک دارند، به همین ترتیب سطح Kام جداساز K-1 و سطح K+1ام جداساز صفر دارد، و به همین شیوه چرخه تکرار می‌شود. در واقع جداساز بعدی بصورت یک تابع طبق رابطه ۱، تعریف شده است [۱۳].

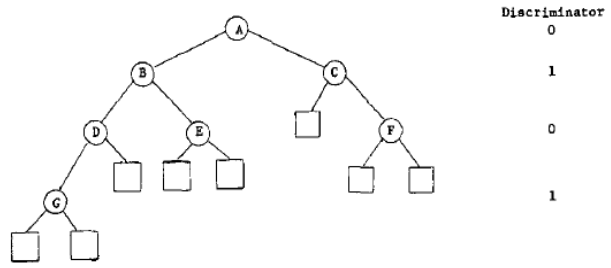
$$NEXTDISC\ (i) = (i+1) \bmod k$$
$$NEXTDISC\ (disc\ (p)) = disc\ (LOSON\ (p)) \qquad (۱)$$

همچنین رابطه ۱ برای HISON(P) نیز تعریف شده است (اگر آن گره‌ها NULL نباشند). در ادامه شکل ۱ یک مثال از رکوردهای ذخیره شده به عنوان گره‌های درخت دوبعدی را نشان می‌دهد که مستطیل‌ها نشان‌دهنده‌ی رنج زیردرخت هستند.



شکل ۱- درخت دوبعدی [۱۳]

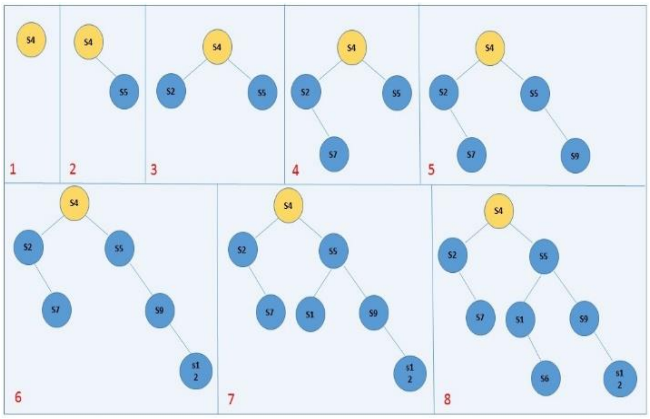
شکل ۲ نیز گراف مسطح درخت دوبعدی ۱ را نشان می‌دهد که LOSON بوسیله شاخه‌های چپ و HISON بوسیله شاخه‌های راست نشان داده شده‌اند و مربع‌ها نیز نشان‌دهنده‌ی فرزندان تهی هستند [۱۳].



شکل ۲- گراف مسطح درخت دوبعدی [۱۳]

- **تشکیل درخت جستجوی دودویی سه‌بعدی**

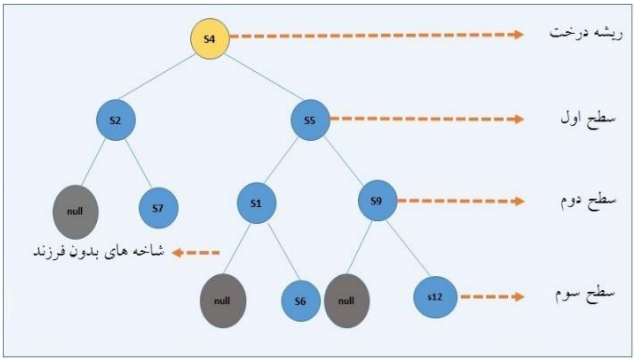
در روند تشکیل درخت جستجوی دودویی، هر سرویس فراخوانی شده از مخزن پایگاه داده UDDI بعنوان یک گره در درخت جستجوی دودویی سه‌بعدی ذخیره می‌شود. به منظور عملیات فیلترینگ سرویس‌ها از مخزن پایگاه داده براساس پارامترهای درخواستی کاربر، ابتدا یک گره به عنوان ریشه انتخاب می‌شود و سطح صفر درخت را تشکیل می‌دهد، در ادامه بقیه گره‌ها با مقایسه پارامترهای مختصات و زمان تاخیر در سطح اول، دوم و سوم قرار می‌گیرند. طبق رابطه ۲ مختصه x یک گره با مختصه x گره‌ی ریشه مقایسه می‌شود و در جای مناسب درج خواهد شد. به همین ترتیب برای سطح دوم درخت مختصه y و برای سطح سوم نیز پارامتر کیفی تاخیر هر گره با گره ریشه مقایسه می‌شوند.



شکل ۳- مراحل تشکیل درخت k-d3

$$\begin{cases} \text{if } Node(x) >= Root(x) & Node added to Right \\ otherwise & Node added to Left \end{cases} \quad (۲)$$

همانطور که در شکل ۳ مثالی از مراحل تشکیل درخت k-d3 نشان داده شده است گره S4 به عنوان ریشه انتخاب می‌شود و اولین سرویس S5 است که طبق رابطه بالا بصورت S5(6)>=S4(6) مقایسه می‌گردد. در اینجا طبق تابع تصمیم تعریف شده، الگوریتم تساوی را بعنوان بزرگ‌تر بودن در نظر می‌گیرد که در این مثال سرویس S5 در سمت راست درخت قرار می‌گیرد. به همین ترتیب سرویس بعدی S2(5)<S4(6) است در نتیجه S2 در سمت چپ درخت قرار می‌گیرد. مقایسه تمام سرویس‌ها به همین شکل انجام می‌شود، شاخه‌هایی از درخت نیز که فرزند ندارد NULL در نظر گرفته می‌شوند و درخت نهایی مطابق شکل ۴ تشکیل می‌شود.



شکل ۴- درخت k-d3

۲-۲-۲- رفع ناسازگاری میان سرویس‌ها

ترکیب ایستای سرویس‌ها در زمان طراحی و معماری سیستم نرم‌افزاری ایجاد می‌گردد. بدین‌صورت که سرویس‌های موردنیاز انتخاب، به‌هم مرتبط و در نهایت کامپایل شده و استقرار می‌یابند. این حالت درصورتی مناسب است که مولفه‌های سرویس به ندرت تغییر یابند یا به‌طور کلی تغییر نکنند. در حالت ترکیب ایستا، عواملی از جمله فراهم کردن سرویس جدید یا جایگزینی سرویس قبلی، همچنین در نظر نگرفتن پارامترهای ورودی و خروجی درخواست کاربر موجب ناسازگاری سرویس‌ها می‌شود. بنابراین ترکیب ایستا ممکن است بسیار محدودکننده باشد.

جدول ۲- نمایش لیست‌های موجود در روش پیشنهادی

List1	ذخیره ورودی	سرویس‌هایی که ورودی آنها با ورودی درخواست شده کاربر برابر باشد.
List2	ذخیره خروجی	سرویس‌هایی که خروجی آنها با خروجی درخواست شده کاربر برابر باشد.
List3	ذخیره ترکیب بهینه	لیست ترکیب نهایی سرویس هاست که شامل زیر مجموعه‌ای از ورودی و خروجی درخواست شده کاربر همراه با کمترین تاخیر باشد.

الگوریتم ۱- list۱ & list۲

```
Serach in all service
      Select service that have same input with user request
            Add all of them to list1
Search in all service
      Select service that have same result with user request
            Add all of them to list2
```

شکل ۵- الگوریتم ۱

در این مرحله از پژوهش جهت رفع ناسازگاری سرویس‌ها، پس از تشکیل درخت k-d3، سرویس‌ها براساس پارامترهای ورودی و خروجی درخواست کاربر به دو لیست خطی با نام‌های List1 و List2 تقسیم می‌شوند. همانطور که جدول ۲ و الگوریتم ارائه شده در شکل ۵ نشان می‌دهد، List1 شامل زیرمجموعه‌ای از سرویس‌هایی است که ورودی آنها با ورودی درخواست شده از سوی کاربر برابر باشد، List2 نیز از زیرمجموعه‌ای از سرویس‌هایی که خروجی آنها با خروجی درخواست شده کاربر برابر باشند، تشکیل می‌شود. بعنوان مثال فرض کنید، کاربری

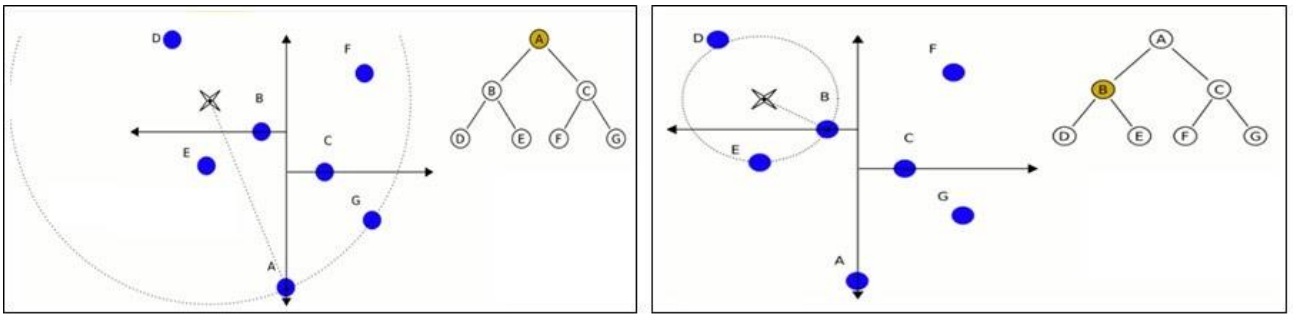
با مختصات مکانی (4, 5) سرویس S را با ورودی‌های A, B و خروجی‌های C, D درخواست کرده است. طبق تعاریف بیان شده در List1=(s) Input A, B و در List2=(s) Output C, D قرار می‌گیرند. در ادامه از این دو لیست جهت جستجوی نزدیک‌ترین پاسخ به درخواست کاربر استفاده می‌شود.

۲-۲-۳- مرتب‌سازی و جستجوی نزدیک‌ترین همسایه لیست‌ها با استفاده از درخت k-d

لیست‌های ایجاد شده در مرحله‌ی قبل باید طبق درخواست کاربر و براساس مختصات مکانی آن، مرتب شوند. برای انجام مرتب‌سازی از دو لیست ایجاد شده، دو درخت k-d ایجاد می‌شود. همانطور که جدول ۱ نشان می‌دهد، درخت k-d1 شامل سرویس‌های موجود در List1، و درخت k-d2 شامل سرویس‌های موجود در List2 هستند. پس از ایجاد دو درخت k-d با استفاده از تکنیک جستجوی نزدیک‌ترین همسایگی‌ها، لیست‌ها به گونه‌ای مرتب می‌شوند که نزدیک‌ترین سرویس به مختصات مکانی کاربر در اولین عنصر و دورترین سرویس به مختصات مکانی کاربر در آخرین عنصر لیست قرار گیرد. این عمل موجب کاهش سرعت یافتن بهترین سرویس جهت ترکیب خواهد شد. در ادامه روش کار جستجوی نزدیک‌ترین همسایه[11] در قالب یک مثال بیان شده است.

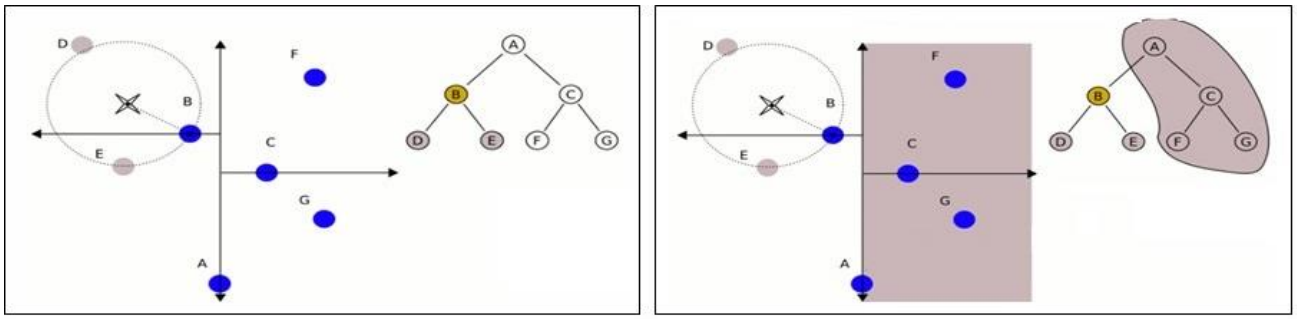
- **جستجوی نزدیک‌ترین همسایه**

جستجوی نزدیک‌ترین همسایه که همچنین با نام جستجوی مجاورت نیز شناخته می‌شود، یک مساله بهینه‌سازی جهت یافتن نزدیک‌ترین نقاط در فضاهای متریک است. در این جستجو، مجموعه S شامل تعدادی نقطه در یک فضای متریک مانند M و یک نقطه پرس‌وجوی q∈M داده شده است، هدف یافتن نزدیک‌ترین نقطه در S به q است. در بسیاری از موارد، فضای M بصورت یک فضای اقلیدسی d بعدی بین نقاط با معیار فاصله اقلیدسی، فاصله منهتن یا دیگر فاصله‌های متریک سنجیده می‌شود. شکل ۶ روش کار جستجوی نزدیک‌ترین همسایه را نشان می‌دهد. همانطور که در شکل (۶-a) مشخص است، گره‌های درخت در یک فضای



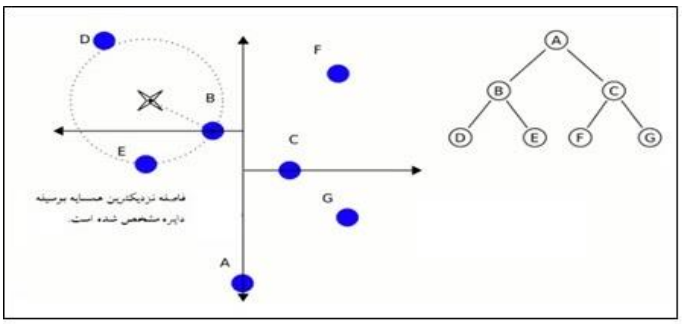
جستجوی اول - عمق از گره A به سمت شاخه چپ درخت (B)

تقسیم گره‌های درخت در یک فضای مختصات دوبعدی (a)



بررسی همه فرزندان A و بازگشت به شاخه‌ی B (d)

بررسی مسافت دو گره D و E (c)



مشخص شدن فاصله نزدیک‌ترین همسایه بوسیله "دایره" (e)

شکل ۶- جستجوی نزدیک‌ترین همسایه در درخت دودویی دوبعدی [۱۴]

مختصات دوبعدی تقسیم شده‌اند و محدوده در نظر گرفته شده برای جستجوی نزدیک‌ترین همسایه تمام نقاط را شامل می‌شود، و نمی‌تواند از هیچ ناحیه‌ای چشم‌پوشی کند. جستجو از گره A آغاز و با جستجوی اول-عمق ادامه می‌یابد (یک پشته گره‌های والد را نگهداری می‌کند) طبق شکل (۶-b)، گره‌های موجود در سمت چپ مجموعه‌ای از بهترین مسافت‌های تخمین‌زده شده برای A محسوب می‌شوند. طبق شکل (۶-c) جستجو در گره B ادامه می‌یابد، در این مرحله فاصله B محاسبه و با بهترین تخمین مقایسه می‌شود، چون فاصله‌ی محاسبه شده به نسبت بهترین تخمین کوچک‌تر است پس بهترین تخمین به‌روزرسانی می‌شود و جستجو در فرزندان چپ سپس راست ادامه می‌یابد. پس از بررسی مسافت دو گره D و E و مقایسه با بهترین تخمین بدست آمده، این نتیجه بدست می‌آید که B بهترین تخمین برای این زیرشاخه است، پس دو گره D و E دور انداخته می‌شوند و جستجو به گره‌های والد طبق شکل (۶-d) باز می‌گردد. در ادامه‌ی جستجو و با بررسی همه‌ی فرزندان A، این نتیجه حاصل شده است که B بهترین تخمین برای کل درخت است که فاصله آن در شکل (۶-e) بوسیله "دایره" مشخص شده است [۱۴].

۲-۲-۴- ترکیب سرویس

برای انجام عمل ترکیب و حل مشکل ناسازگاری سرویس‌ها، پس از تشکیل دو درخت k-d1 و k-d2 و انجام مراحل بخش قبل، طبق جدول ۲، یک لیست جدید جهت ذخیره بهترین ترکیب به نام List3، ایجاد می‌شود. سپس عنصر اول List1 که پس از مرتب‌سازی، نشان‌دهنده‌ی نزدیک‌ترین همسایه به مختصات مکانی کاربر است، به List3 افزوده می‌شود. با توجه به مثال بیان شده S2(5, 6)=A, B.E, F اولین عنصر $List_1$ است که بعنوان نزدیک‌ترین سرویس به مختصات مکانی کاربر، در List3 درج می‌شود. در ادامه با توجه به ورودی درخواستی کاربر، پارامترهای خروجی عنصر اول از List1 در نظر گرفته می‌شود و با جستجو در درخت k-d3 نزدیک‌ترین همسایگی به مکان این سرویس بررسی می‌گردد. در اینجا دو حالت متفاوت طبق الگوریتم ارائه شده در شکل ۷ جهت پر شدن عنصر دوم از List3 و ایجاد ترکیب مناسب در نظر گرفته شده است:

الگوریتم ۲- state۱ & state۲

```
for i 0 to list count
 {
    for j 0 to count data
    {
        for k 0 to count data j
        {
            if (list 0,i returnval1 == Data j,k Input1 &&
                list 0,i returnval2 == Data j,k Input2 &&
                (list 1,i Input1 == Data j,k returnval1 ||
                List 1,i Input2 == Data j,k returnval2))//حالت اول
            {
                if (list 1,i Input1 == Data j,k returnval1 &&
                list 1,i Input2 == Data j,k returnval2)//حالت دوم
                {
                   List of WebService s
                   Add  list 0,i ,data j,k  ,list 1,i to s

                                WebService w
                                Fitess(s)
                                Add w to s
                                Add s to result
                                found = true
                                break;
                            }
                            else
                            {
                              Add data j,k to temp
                            }
                        }
                    }
                    if (found)
                        break
                }
                if (found)
                    break
                Clear temp
```

شکل ۷- الگوریتم ۲

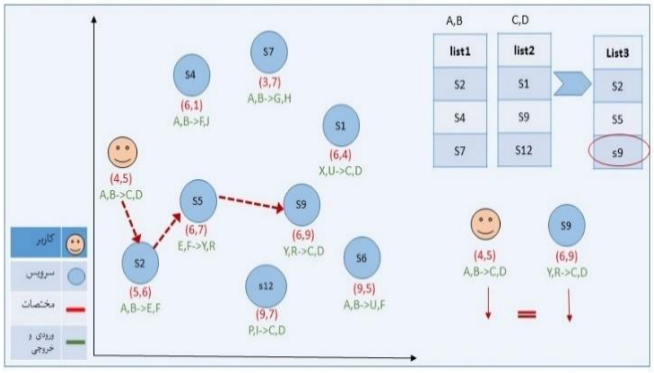
- **حالت اول**

در این حالت سرویسی انتخاب و به عنصر دوم List3 افزوده خواهد شد که پارامترهای ورودی آن با پارامترهای خروجی سرویس اول از List3 برابر باشد. در غیراینصورت این سرویس نادیده گرفته شده و برای جلوگیری از بررسی مجدد در جستجوهای بعدی از لیست حذف خواهد شد. جهت یافتن عنصر دوم لیست ترکیب، جستجو در درخت k-d3 ادامه می‌یابد و با یافتن هر سرویس نزدیک به مختصات مکانی کاربر، مجدداً شرط ذکر شده بررسی می‌شود. این عمل تا پایان درخت k-d3، برقراری شرط و یافتن سرویس موردنظر ادامه می‌یابد. در ادامه‌ی مثال ذکر شده، با توجه به برقراری شرایط موجود در حالت اول S5(6, 7)=E, F.Y, R با ورودی مشابه خروجی S2(5, 6)=A, B.E, F بعنوان عنصر دوم List3 درج می‌شود.

- **حالت دوم**

در این حالت علاوه بر برقراری حالت اول، خروجی سرویسی که انتخاب می‌شود باید ورودی یکی از سرویس‌های موجود در List2 باشد، که در اینصورت ترکیب مناسب ایجاد شده است. برای نشان دادن این حالت، می‌توان در مثال ذکر شده به عنصر دوم List3 اشاره کرد. اگر خروجی S5(6, 7)=E, F.Y, R با سرویسی که ورودی آن C, D (همان خروجی درخواست شده کاربر است)، برابر باشد پس شرط برقرار است، بنابراین ترکیب مناسب ایجاده شده و جستجو به اتمام می‌رسد. در این مثال شرط حالت دوم برقرار نیست و جهت ادامه‌ی عمل جستجوی سرویس مناسب، روال ذکر شده در حالت اول تا انتهای درخت k-d3 و جستجوی تمام سرویس‌های موجود، ادامه می‌یابد. در صورت اتمام جستجو و یافت نشدن ترکیب مناسب، اخرین عنصر افزوده شده به List3 در نظر گرفته می‌شود و مجدداً جستجو در درخت k-d3 آغاز خواهد شد.

در پایان جستجو، List3 مطابق شکل ۸ شامل سرویس‌های S2>S5>S9 است. در این روش نقشه سرویس از ابتدا مشخص نیست و طی گام‌های اجرایی الگوریتم ایجاد شده است. بدین شکل که در هرمرحله از اجرای الگوریتم و جستجو جهت انتخاب سرویس درخواست شده کاربر، یک سرویس به List3 افزوده می‌شود که این عمل موجب ایجاد نقشه سرویس بصورت پویا شده است.



شکل ۸- ایجاد ترکیب بهینه

۳- ارزیابی مدل پیشنهادی و تحلیل نتایج

در بخش تحلیل و ارزیابی مدل پیشنهادی، ابتدا رویکرد ارائه شده توسط زبان برنامه‌نویسی c# و یک مجموعه داده‌ی تصادفی که حاوی زیرمجموعه‌ای از سرویس‌های مشابه درخواست کاربر است، پیاده‌سازی و ارزیابی شد. نتایج حاصل از

ارزیابی در قالب زمان اجرا و حافظه‌ی مصرفی، با داده‌های یکسان بر روی الگوریتم ژنتیک و الگوریتم k-d مورد استفاده در [۱] بررسی و مقایسه گردید.

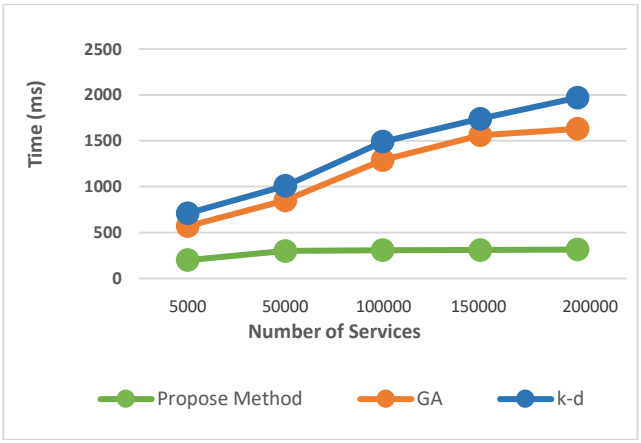
۳-۱- زمان اجرا

در بخش ارزیابی زمان اجرا، جهت ارزیابی صحت عملکرد و مقیاس‌پذیر بودن روش، الگوریتم در دو مقیاس پایین و بالا آزموده شده است. در تحلیل نتایج به دست آمده، مشاهده می‌شود که هر چقدر تعداد سرویس‌ها در الگوریتم مورد استفاده افزایش یابند، زمان ایجاد ترکیب مناسب به میزان بسیار کمی افزایش می‌یابد. با توجه به تعداد زیاد سرویس‌های افزوده شده در مقیاس بالا، زمان ایجاد ترکیب مناسب، قابل قبول است. انجام فیلترینگ سرویس‌ها جهت کاهش تعداد سرویس‌های کاندید در ترکیب، زمان جستجو برای یافتن سرویس مناسب را بسیار کاهش داده است که نسبت به روش‌های مقایسه شده موجب پیچیدگی کمتری می‌شود، همچنین به دلیل اینکه الگوریتم تنها مناسب‌ترین جواب را نمایش می‌دهد توانسته است زمان بسیار کمی را صرف کند.

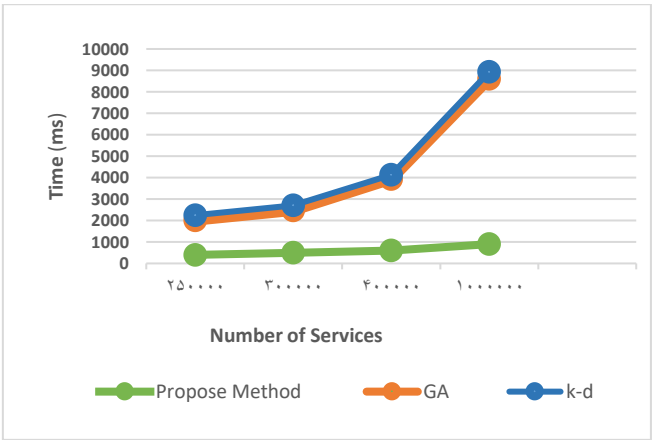
جهت صحت عملکرد الگوریتم ارائه شده، مقایسه‌ای میان الگوریتم‌های k-d و GA در [۱] با الگوریتم موجود در این پژوهش انجام شد. طبق نتایج نشان داده شده در جدول ۳ و نمودارهای ارائه شده در شکل ۹ در ترکیب پویای ایجاد شده، بطور میانگین با هر صد هزار افزایش تعداد سرویس‌ها، تنها یک صدم ثانیه به زمان یافتن ترکیب مناسب اضافه می‌شود. انجام عملیات فیلترینگ سرویس‌ها، مرتب‌سازی براساس نزدیک‌ترین همسایگی به درخواست کاربر و نمایش مناسب‌ترین جواب، موجب بهبود قابل توجه زمان اجرایی الگوریتم با پیچیدگی n شده است در صورتی که در مقایسه با ترکیب ایستا [۱] و بدون انجام عمل فیلترینگ، با افزایش تعداد سرویس‌ها این زمان سیر صعودی بیشتری دارد و تقریبا بصورت نمایی افزایش می‌یابد که پیچیدگی این الگوریتم‌ها $2n^2$ محاسبه شده است.

جدول ۳- مقادیر زمان (میلی ثانیه) با تعداد نمونه سرویس‌های یکسان

Count	Propose Method	GA	k-d
5000	0.02	0.57	0.61
50000	0.03	0.85	0.92
100000	0.03	1.29	1.39
150000	0.03	1.56	1.64
200000	0.03	1.63	1.87
250000	0.04	2.01	2.15
300000	0.05	2.47	2.61
400000	0.06	3.93	3.99
1000000	0.09	8.61	8.83

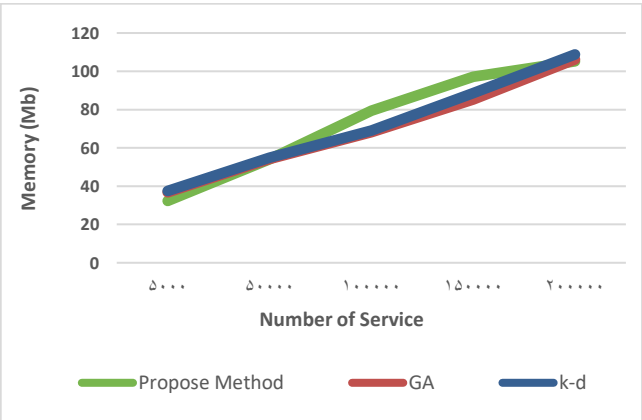


نتایج حاصل از ارزیابی زمان ترکیب سه الگوریتم در مقیاس پایین (a)

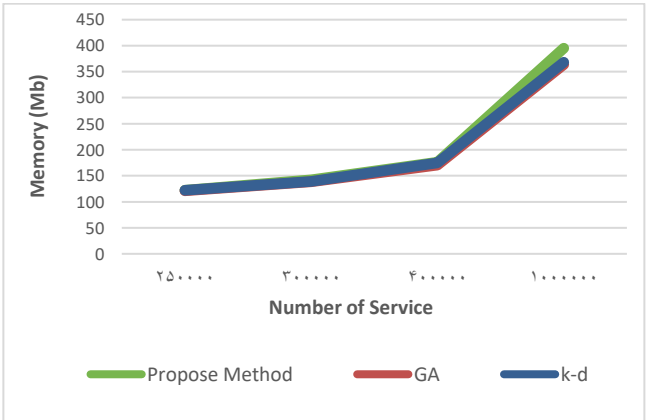


نتایج حاصل از ارزیابی زمان ترکیب سه الگوریتم در مقیاس بالا (b)

شکل ۹- نتایج حاصل از ارزیابی زمان ترکیب سه الگوریتم با تعداد نمونه سرویس‌های یکسان در مقیاس پایین (a) و مقیاس بالا (b)



نتایج حاصل از ارزیابی حافظه مصرفی سه الگوریتم در مقیاس پایین (a)



نتایج حاصل از ارزیابی حافظه مصرفی سه الگوریتم در مقیاس بالا (b)

شکل ۱۰- نمایش نتایج حاصل از مصرف حافظه سه الگوریتم در مقیاس بالا با تعداد نمونه سرویس‌های یکسان در مقیاس پایین (a) و مقیاس بالا (b)

۳-۲- مصرف حافظه

نتایج در جدول ۴ و نمودارهای ارائه شده در شکل ۱۰ نشان می‌دهند کـه در مـدل پیشنهادی با افزایش تعداد سرویس‌ها در دو مقیاس پایین و بـالا، میـزان حافظـه‌ی مصرفی در الگوریتم پیشنهادی نسبت به الگوریتم‌های مورد استفاده در [۱] افزایش می‌یابد. میزان افزایش مصرف حافظه‌ی رویکرد خیلی قابل توجه نیست و دلیـل آن پویایی روش می‌باشد.

جدول ۴- مقادیر حافظه مصرفی (مگابیت) با تعداد نمونه سرویس‌های یکسان

Count	Propose Method	GA	k-d
5000	32.3	36.8	37.5
50000	54.0	54.2	54.8
100000	79.4	68.2	69.1
150000	97.1	85.2	88.6
200000	105.2	106.2	108.8
250000	122.2	121.2	122.1
300000	142.3	138.4	139.2
400000	176.3	170.3	174.9
1000000	395.2	363.8	368.2

۴- نتیجه‌گیری و کارهای آتی

چالش کلیدی دریـافتن وب سـرویس مناسـب بـه منظـور ترکیـب سـرویس‌ها کـه انتظارات درخواست کننده سرویس را برآورده کند، بصورت یک مسـاله بهینه‌سـازی مطرح است. با بررسی روش‌های ارائه شده در زمینه ترکیـب سـرویس، ایـن نتیجـه حاصل می‌شود که عمده‌ترین مشـکلات ترکیـب سـرویس‌های وب نادیـده گـرفتن کیفیت سرویس در سرویس‌های متصل به شبکه، عدم مقیاس‌پذیری، عدم محاسـبه کیفیت سرویس وابسـته بـه زمـان ورودی و زمـان انتقـال شـبکه و تـاخیر ترکیـب سرویس‌ها هستند. تاکنون روش‌های بسیاری بـرای حـل مشـکل ترکیـب سـرویس پیشنهاد شده است که از جمله آنها می‌توان به روش‌هـای ابتکـاری اشـاره کـرد. بـا بررسی انواع الگوریتم‌های ابتکاری می‌توان نتیجه گرفت هنوز مشکلات زیادی بـرای حل مسأله ترکیب وب سرویس‌ها براساس ویژگی‌های کیفی وجود دارد و هـر کـدام از روش‌ها معمولاً به تنهـایی مشـکل داشـته و خیلـی سـریع در دام بهینـه محلـی می‌افتند. از دیگر روش‌های مورد استفاده، که در ایـن پـژوهش مـورد مطالعـه قـرار گرفت یک رویکرد آگاه از شبکه است که کیفیـت سـرویس، سـرویس‌ها و کیفیـت سرویس شبکه را بطور مستقل بکار می‌گیرد. با توجه به بررسی نتایج ایـن رویکـرد، انجام عمل ترکیب سرویس بصورت ایسـتا و در نظـر نگـرفتن پارامترهـای ورودی و خروجـی کـاربر، موجـب عدم‌سـازگاری میـان سـرویس‌ها در ترکیـب می‌شـود. همچنین استفاده از الگوریتم ژنتیـک موجـب شـده اسـت اجـرای الگـوریتم بسـیار زمان‌بر باشد.

این پژوهش در جهت رفع برخی از محدودیت‌های ذکر شده، راه‌حلی پیشـنهاد می دهد که با در نظر گرفتن پارامتر کیفیت اخیر و موقعیت کاربر در یک شـبکه‌ی آگاه، موجب یافتن ترکیب مناسب با کمترین تاخیر می‌شود. نسبت بـه بررسـی‌های انجام شده، الگوریتم پیشنهادی با انجام عمـل فیلترینـگ سـرویس‌ها و اسـتفاده از تکنیک جستجوی نزدیک‌ترین همسایه، نزدیک‌ترین سرویس بـه مختصـات مکـانی کاربر را می‌یابد که موجب می‌شود زمان اجرایی بسـیار کـاهش یابـد. ایـن رویکـرد نسبت به روش‌هـای مقایسـه شـده دارای پیچیـدگی کمتـری اسـت. همچنـین بـا استفاده از لیست‌های خطی و در نظر گرفتن پارامترهای درخواستی کـاربر، موجـب پویایی روش و رفع عدم‌سازگاری سرویس‌ها شـده اسـت. جهـت بهبـود روش ارائـه شده در آینده، می‌توان روی مصرف حافظه‌ی کمتـر و در نظـر گـرفتن پارامترهـای کیفی بیشتری تمرکز کرد. همچنین پیاده‌سازی رویکرد در مقیاس وسـیع از جملـه ابر نیز می‌تواند مد نظر قرار گیرد.

مراجع

[1] A. Klein, F. Ishikawa, and SH. Honiden, "SanGA: A Self-Adaptive Network-Aware Approach to Service Composition," IEEE Transactions on Services Computing, vol. 7, no. 3, 2014.

[2] A. Klein, F. Ishikawa, and S. Honiden, "Efficient Heuristic Approach with Improved Time Complexity for Qos-Aware Service Composition," In Web Services (ICWS), 2011 IEEE International Conference on, pp. 436-443, 2011.

[3] Z. Ye, X. Zhou, and A. Bouguettaya. "Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing," In International Conference on Database Systems for Advanced Applications, pp. 321-334, 2011.

[4] D. Wang, Y. Yang, and ZH. Mi, "A Genetic-Based Approach to Web Service Composition in Geo-Distributed Cloud Environment," InProceedings of theComputer & Electrical Engineering, vol. 43, no. 3, pp. 1-330, 2014.

[5] M. Moradi, and S. Emadi, "Reducing the Calculations of Quality-Aware Web Services Composition Based on Parallel Skyline Service," International Journal of Advanced Computer Science and Applications, vol. 7, no. 7, 2016.

[6] M. Alrifai, and T. Risse, "Combining Global Optimization with Local Selection for Efficient Qos-Aware Service Composition," In Proceedings of the 18th international conference on World wide web, ACM; pp. 881-90, 2009.

[7] Z. Zheng, T. C. Zhou. M. R. Lyu, and I. King, "Component Ranking for Fault-Tolerant Cloud Applications," IEEE Transactions on Services Computing, vol. 5, no. 4, pp. 540-50, 2012.

[8] B. Klöpper, F. Ishikawa, and S. Honiden "Service Composition with Pareto-Optimality of Time-Dependent QoS Attributes," in Service-Oriented Computing, vol. 6470, Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, pp. 635-640, 2010.

[9] F. Rosenberg, M. B. Müller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar, "Metaheuristic Optimization of Large-Scale QoSAware Service Compositions," In Services Computing (SCC), 2010 IEEE International Conference on, pp. 97-104, 2010.

[10] Y. Chen, J. Huang, C. Lin, and J. Hu, "A Partial Selection Methodology for Efficient qos-Aware Service Composition," IEEE Transactions on Services Computing, vol. 8, no. 3, pp. 384-397, 2015.

[11] S. Chattopadhyay, A. Banerjee, and N. Banerjee, "A Scalable and Approximate Mechanism for Web Service

Composition," In 2015 IEEE International Conference on Web Services (ICWS), pp. 9-16, 2015.

[12] A. Klein, F. Ishikawa, and S. Honiden, "Towards Network-Aware Service Composition in the Cloud," Proceedings of the 21st international conference on World Wide Web. ACM, pp. 959-968, 2012.

[13] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," Communications of the ACM, vol. 18, no. 9, pp. 45-47, 1975.

[14] Wikipedia, https://en.wikipedia.org/wiki /File:KDTree-animation.gif.

طاهره داوودی دریافت مدرک کاشناسی ارشد کامپیوتر - نرم‌افزار از دانشگاه آزاد اسلامی واحد یزد در سال ۱۳۹۵. زمینه‌های تحقیقاتی: انتخاب و ترکیب سرویس‌ها در محیط‌های مبتنی بر سرویس، داده‌کاوی و الگوریتم‌های خوشه‌بندی و دسته‌بندی.

آدرس پست‌الکترونیکی ایشان عبارت است از:

ms.davodi@iauyazd.ac.ir

سیما عمادی دریافت مدرک دکتری کامپیوتر -نرم‌افزار از دانشگاه آزاد اسلامی واحد علوم و تحقیقات با اخذ رتبه اول در سال ۱۳۸۷. مربی دانشگاه آزاد اسلامی واحد میبد از سال ۱۳۷۶ الی ۱۳۸۷ و استادیار دانشگاه آزاد اسلامی واحد یزد از ۱۳۹۰ تا کنون. مدیر گروه مقاطع کارشناسی‌ارشد و دکتری دانشگاه آزاد اسلامی واحد یزد. زمینه‌های تحقیقاتی: ارزیابی کارایی و قابلیت اطمینان معماری نرم‌افزار، الگوهای طراحی نرم‌افزار، انتخاب و ترکیب سرویس‌ها در محیط‌های مبتنی بر سرویس، تحلیل داده‌های حجیم ، داده‌کاوی و تست نرم‌افزار.

آدرس پست‌الکترونیکی ایشان عبارت است از:

emadi@iauyazd.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۱۳۹۶/۰۴/۱۸

تاریخ اصلاح: ۱۳۹۶/۰۴/۲۶

تاریخ قبول شدن: ۱۳۹۶/۰۶/۲۸

نویسنده مرتبط: دکتر سیما عمادی، دانشکده فنی مهندسی، دانشگاه آزاد اسلامی واحد یزد، یزد، ایران.

---

[1]Quality of Service (QOS)
[2]Integer Programing (IP)
[3]Hill-Climbing (HC)
[4]Initial Bias (IB)
[5]Genetic Algorithm (GA)
[6]Simulated Annealing (SA)
[7]Tabu Search (TS)
[8]Multidimensional Binary Search Tree (k-d tree)
[9]Latency
[10]Multidimensional Binary Search Tree (k-d tree)
[11]Nearest Neighbor Search (NN)