مجله محاسبات نرم، جلد 13، شماره 2 (پاییز و زمستان 1403)، صفحه 2_21
https://doi.org/10.22052/scj.2024.248472.1109

مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir

دانشگاه کاشان
University of Kashan

یک ابراکتشافی مبتنی بر عامل برای پیمانه‌بندی سامانه‌های نرم‌افزاری ✧

مجوبه تاج‌گردان[1]، دانشجوی دکتری، حبیب ایزدخواه[*1]، دانشیار، شهریار لطفی[1]، دانشیار

[1] گروه علوم کامپیوتر، دانشکده ریاضی، آمار و علوم کامپیوتر، دانشگاه تبریز، تبریز، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت 29 مهر ماه 1401
پذیرش 20 مهر ماه 1402

کلمات کلیدی:
پیمانه‌بندی نرم‌افزار
بازیابی معماری
ابراکتشافی
سامانه‌های چندعاملی
یادگیری تقویتی

چکیده

الگوریتم‌های پیمانه‌بندی برای بازیابی معماری نرم‌افزار استفاده می‌شوند. این الگوریتم‌ها کد منبع سامانه نرم‌افزاری را به پیمانه‌های کوچک‌تر و قابل فهم‌تر تقسیم می‌کنند. از آنجایی که پیمانه‌بندی نرم‌افزار یک مساله چندجمله‌ای غیرقطعی‌سخت است، به طور معمول از روش‌های مبتنی بر جستجو برای حل آن استفاده می‌شود. در سال‌های اخیر، استفاده از ابراکتشافی‌ها با رویکردهای جستجوی هوشمند، برای دستیابی به سطح بالاتری از عمومیت رو به افزایش است. در این مقاله، یک ابراکتشافی عمومی مبتنی بر عامل، با استفاده از مفهوم سامانه‌های چندعاملی، برای پیمانه‌بندی نرم‌افزار ارائه می‌شود. در الگوریتم پیشنهادی، از عامل‌هایی با دیدگاه‌های جستجوی تقویتی و تنوعی استفاده می‌شود و عامل‌های دارای دیدگاه یکسان در یک اجتماع قرار می‌گیرند. در هر گام از جستجو، مناسب‌ترین اجتماع با استفاده از یادگیری تقویتی به طور خودکار انتخاب و عامل‌های آن به صورت موازی اجرا می‌شوند. همچنین، در طراحی برخی از عامل‌ها، برای حفظ تنوع، از مفهوم نظریه آشوب استفاده می‌شود. برای نشان دادن قابلیت اجرای الگوریتم پیشنهادی یازده سامانه نرم‌افزاری دنیای واقعی با اندازه کوچک و متوسط و ده پوشه از موزیلا فایرفاکس با دامنه‌ها و قابلیت‌های متفاوت انتخاب شده‌اند. نتایج آزمایش‌ها نشان می‌دهند که ابراکتشافی پیشنهادی در بیشتر موارد پیمانه‌بندی‌هایی با کیفیت بالاتر را در زمان کمتری نسبت به الگوریتم‌های مقایسه شده تولید می‌کند. میانگین بهبود عددی الگوریتم پیشنهادی از نظر کیفیت پیمانه‌بندی و زمان اجرا روی ده پوشه از موزیلا فایرفاکس به ترتیب 77/607 و 59/448 درصد می‌باشد.

1. مقدمه

فهم برنامه نقش مهمی را در فرآیند توسعه و نگهداری نرم‌افزار بازی می‌کند. یک گام مهم برای فهم برنامه در فرآیند نگهداری، بازیابی معماری نرم‌افزار است. توسعه‌دهندگان نرم‌افزار با کمک روش‌های پیمانه‌بندی می‌توانند ساختار نرم‌افزاری را استخراج کنند که طراحان آن در دسترس نیستند [1]. در بازیابی معماری نرم‌افزار با استفاده از روش‌های پیمانه‌بندی، کد منبع یک سامانه نرم‌افزاری به بخش‌های معنادار و قابل‌فهم افراز می‌شود [2]. در فرآیند پیمانه‌بندی نرم‌افزار، موجودیت‌های کد منبع مانند کلاس‌ها، توابع و غیره در مجموعه‌هایی به نام پیمانه‌ها به نحوی

✧ نوع مقاله: پژوهشی

* نویسنده مسئول
پست(های) الکترونیک: m.tajgardan@tabrizu.ac.ir (تاج‌گردان)
izadkhah@tabrizu.ac.ir (ایزدخواه)
shahriar_lotfi@tabrizu.ac.ir (لطفی)

نحوه ارجاع به مقاله: مجوبه تاج‌گردان، حبیب ایزدخواه، شهریار لطفی، «یک ابراکتشافی مبتنی بر عامل برای پیمانه‌بندی سامانه‌های نرم‌افزاری»، مجله محاسبات نرم، جلد 13، شماره 2، صفحات 2_21، پاییز و زمستان 1403.

گروه‌بندی می‌شوند که موجودیت‌های دارای پیمانه یکسان نسبت به موجودیت‌هایی که در پیمانه‌های دیگر قرار دارند، به هم شبیه‌تر هستند. در مهندسی نرم‌افزار، ارتباط در یک پیمانه و میزان وابستگی متقابل بین پیمانه‌های نرم‌افزاری به ترتیب انسجام و اتصال نامیده می‌شوند [3]. انسجام بالا و اتصال پایین ساختاری را برای توسعه‌دهنده نرم‌افزار فراهم می‌کنند که نگهداری آن آسان‌تر است [4].

از آنجایی که هیچ تعریف جامع و تثبیت‌شده‌ای از ویژگی‌های پیمانه‌بندی بهینه وجود ندارد، امروزه برای محاسبه کیفیت پیمانه‌بندی‌های تولید شده توسط الگوریتم‌ها از دو نوع ارزیابی داخلی و خارجی استفاده می‌شود. در ارزیابی داخلی، چندین معیار برای ارزیابی میزان جداسازی صحیح پیمانه‌ها استفاده و به صورت مستقیم اقدام به امتیازدهی پیمانه‌بندی‌ها می‌کنند. در ارزیابی خارجی، پیمانه‌بندی نهایی با پیمانه‌بندی فرد خبره مقایسه می‌شود. یک فرد خبره در یک سامانه نرم‌افزاری کسی است که آن سامانه را تجزیه کرده است [1]. یک الگوریتم پیمانه‌بندی که پیمانه‌بندی به دست آمده از آن نزدیک به پیمانه‌بندی فراهم شده توسط فرد خبره باشد، قابل اعتماد است.

تاکنون، الگوریتم‌های پیمانه‌بندی زیادی برای بازیابی معماری و درک سامانه‌های نرم‌افزاری ارائه شده است. بیشتر روش‌های پیمانه‌بندی نرم‌افزار سلسله مراتبی یا مبتنی بر جستجو هستند. روش‌های سلسله مراتبی ادغامی [5]، حریصانه هستند و در هر مرحله از جستجو شبیه‌ترین موجودیت‌ها را با هم ادغام می‌کنند. این الگوریتم‌ها زمان جستجوی معقولی دارند. تصمیم‌گیری اختیاری [6] و محلی بودن معیارهای تشابه موجود برای این روش‌ها [1]، از جمله معایب آنها هستند. با توجه به پیچیدگی مساله پیمانه‌بندی، به طور معمول روش‌های مبتنی بر جستجو برای حل آن استفاده می‌شوند.

با وجود دستیابی به راه‌حل نزدیک به بهینه، روش‌های مبتنی بر جستجو مانند الگوریتم‌های تکاملی زمانی که بر روی سامانه‌های نرم‌افزاری با مقیاس بزرگ اعمال می‌شوند با محدودیت‌های زمان اجرا و فضای جستجو روبه‌رو هستند و همچنین به دلیل ماهیت تصادفی بسیار کند عمل می‌کنند. در این رویکردها، پیمانه‌بندی نرم‌افزار به عنوان یک مساله جستجو در نظر گرفته می‌شود و برای هدایت فرآیند پیمانه‌بندی، توابع تک‌هدفه یا چندهدفه استفاده می‌شوند. تابع‌های BasicMQ [6] و TurboMQ [6] دو تابع هدف خوب‌شناخته‌شده و پرکاربرد هستند. تابع TurboMQ بر خلاف BasicMQ برای گراف‌های وزن‌دار نیز قابل استفاده است و همچنین پیچیدگی زمانی محاسبه آن کمتر است [6]. در TurboMQ، ابتدا فاکتور پیمانه توسط رابطه (1) برای همه پیمانه‌ها محاسبه می‌شود. در این رابطه، μ_i اتصالات داخلی پیمانه i و ε_i، j اتصالات خارجی بین پیمانه i و پیمانه j است. مقدار TurboMQ توسط رابطه (2) محاسبه می‌شود که در آن، k تعداد پیمانه‌ها است.

$$MF_i = \frac{2\mu_i}{2\mu_i + \varepsilon_{ij}} \quad (1)$$

$$TurboMQ = \sum_{i=1}^{k} MF_i \quad (2)$$

با توجه به الگوریتم‌های مختلفی که برای پیمانه‌بندی نرم‌افزار وجود دارند، لازم به ذکر است که:

1. با توجه به چندجمله‌ای غیرقطعی سخت بودن مساله پیمانه‌بندی نرم‌افزار، به طور معمول الگوریتم‌های اکتشافی و فرااکتشافی برای حل آن استفاده می‌شوند. با این حال، این الگوریتم‌ها اغلب عمومی نیستند؛ یعنی در دامنه‌های مساله متفاوت و یا حتی نمونه‌های متفاوت از دامنه مساله یکسان، کارایی متفاوتی دارند [7].
2. در سال‌های اخیر، پژوهشگران از فرااکتشافی‌های ترکیبی برای حل مساله‌های بهینه‌سازی ترکیباتی مانند مساله پیمانه‌بندی نرم‌افزار استفاده کرده‌اند. از جمله مزایای این ترکیب، به دست آوردن بهترین کیفیت راه‌حل در زمان کوتاه‌تر و افزایش توانایی مقابله با مساله‌های پیچیده‌تر است [8]. استفاده از سامانه‌های چندعاملی در پیاده‌سازی روش‌های ترکیبی، به دلیل امکان تعامل بین فرااکتشافی‌ها با هدف جستجوی هوشمندانه فضای حالت و همچنین امکان کاوش همزمان مناطق مختلف فضای جستجو با هدف تنوع بیشتر و رسیدن به راه‌حل‌های بهتر، برجسته

است. با این حال، روش‌های ترکیبی دارای محدودیت‌هایی نیز هستند. در برخی از آنها که چندین فرااکتشافی با هم ترکیب می‌شوند، زیرمجموعه‌ای از همه الگوریتم‌های فرااکتشافی خاص مساله به صورت دستی انتخاب می‌شوند [9]، که زمان‌بر و پرهزینه است. از طرف دیگر، طراحی برخی از این روش‌های ترکیبی قبل از فرآیند بهینه‌سازی واقعی انجام می‌شود [10] (طراحی غیربرخط) و کارایی این روش‌ها روی نمونه‌های مساله مختلف، به عمومیت نمونه‌های آموزشی وابستگی دارد. زمانی‌که نحوه اجرای فرااکتشافی‌های ترکیب شده به صورت ترتیبی باشد، با مساله پیدا کردن بهترین دنباله فراخوانی الگوریتم‌ها روبه‌رو هستند که خود یک مساله بهینه‌سازی است [11]. در برخی از روش‌های ترکیبی که چندین عامل به طور موازی فضای جستجو را کاوش می‌کنند، عامل‌ها و همچنن عملگرهای تعریف شده برای آنها یکسان هستند [11]، این در حالی است که استفاده از عامل‌های مختلف با عملگرهای متفاوت، به دلیل استفاده آنها از دیدگاه‌های جستجوی مختلف، می‌تواند به فرآیند جستجو کمک کند.

3. ابراکتشافی‌ها روش‌های عمومی هستند که برای غلبه بر مشکلات فرااکتشافی‌ها در یک سطح بالاتری از انتزاع عمل می‌کنند. ابراکتشافی‌ها به جای فضای جستجوی راه‌حل‌ها روی فضای جستجوی اکتشافی‌ها عمل می‌کنند. با توجه به منابع محدود و در دسترس بودن طیف گسترده‌ای از اکتشافی‌های خاص مساله، ابراکتشافی‌ها در هر مرحله از جستجو با توجه به شرایط موجود مناسب‌ترین اکتشافی را انتخاب و اعمال می‌کنند. استفاده از ابراکتشافی‌ها با چالش‌هایی روبه‌رو است که عبارتند از:

- برخی از این رویکردها ممکن است خودمختاری تصمیم‌گیری سامانه‌های چندعاملی را محدود کنند [8]. استفاده از یادگیری در ابراکتشافی‌ها به عامل‌ها اجازه می‌دهد تا بدون محدود شدن خودمختاری، ظرفیت عمل خود را بهبود داده و در زمان حل یک مساله بهینه‌سازی تصمیم بهتری بگیرند. با این حال، تعداد زیادی از ابراکتشافی‌هایی که از یادگیری تقویتی استفاده می‌کنند، تنها دارای یک طرح پاداش/جریمه هستند و از دو ویژگی مهم یادگیری تقویتی شامل آزمایش و خطا و پاداش تاخیری پیروی نمی‌کنند. بنابراین، طراحی ابراکتشافی‌هایی که از معیارهای یادگیری تقویتی پیروی می‌کنند، ضروری است.
- مجموعه وضعیت‌های تعریف شده در برخی از ابراکتشافی‌ها ممکن است همه موقعیت‌هایی را که در طی جستجو اتفاق می‌افتند، شامل نشود. از طرف دیگر، در تعریف وضعیت‌ها از متغیرهای مختلفی استفاده می‌شود که تنظیم درست آنها بر عهده کاربر است. تنظیم این متغیرها با استفاده از آزمایش و خطا زمان‌بر و پرهزینه است. بنابراین، تعریف مجموعه وضعیت‌ها به نحوی که شامل همه موقعیت‌ها و همچنین مستقل از مساله باشد، مناسب به نظر می‌رسد.

در نتیجه، برای مقابله با عملکرد کند روش‌های مبتنی بر جستجو در گراف‌های بزرگ، به بهبود روش‌های موجود و توسعه روش‌های جدید، برای جستجوی هوشمندانه فضای جستجو نیاز است. استفاده از ابراکتشافی‌ها در بستر سامانه‌های چندعاملی استراتژی جستجو را هوشمندانه‌تر و آموزنده‌تر می‌کند. از بین منابع معرفی شده در بخش 2 (کارهای مرتبط) تنها مراجع [12] و [13] به ترتیب از مفهوم سامانه‌های چندعاملی و ابراکتشافی برای پیمانه‌بندی نرم‌افزار استفاده کرده‌اند.

ما نیز در تحقیق قبلی خود در مرجع [14] و نسخه توسعه یافته آن در مرجع [15] از یک ابراکتشافی مبتنی بر یادگیری تقویتی برای بهبود کارایی الگوریتم جستجوی محلی تکراری، استفاده کردیم. این مقاله‌ها از یادگیری تقویتی برای انتخاب هوشمندانه جفت مولفه‌های آشفته‌کننده و جستجوی محلی در هر تکرار از الگوریتم جستجوی محلی تکراری استفاده می‌کنند. پس از انتخاب این جفت مولفه، ابتدا مولفه آشفته‌کننده و سپس مولفه جستجوی محلی به ترتیب اجرا می‌شوند. اما در روش پیشنهادی در مقاله حاضر، ما از یادگیری تقویتی برای ایجاد تعادل بین

تقویت (بهره‌برداری) و تنوع (اکتشاف) استفاده می‌کنیم. در هر مرحله از جستجو، یکی از اجتماعات تقویتی یا تنوعی بسته به وضعیت جستجو انتخاب می‌شود و عامل‌های آن به طور موازی به جستجو می‌پردازند. در روش پیشنهادی در مقاله حاضر ممکن است در چند تکرار متوالی اجتماع تقویتی یا اجتماع تنوعی فعال شود و ترتیب خاصی برای اجرای اجتماع‌ها از قبل در نظر گرفته نشده است و بسته به وضعیت جستجو، یک اجتماع مناسب توسط یادگیری تقویتی به طور خودکار برای اجرا انتخاب می‌شود. لازم به ذکر است که تفاوت اصلی الگوریتم پیشنهادی در این مقاله با دو مقاله قبلی ما [14]، [15]، در رویکرد آنها است و روش‌های جستجوی محلی و تنوعی می‌توانند از روش‌های موجود در ادبیات انتخاب شوند و ضرورتی بر متمایز بودن از سایر مقالات برای آنها وجود ندارد. به طور کلی می‌توان گفت اشتراک اصلی این مقاله با دو مقاله قبلی ما [14]، [15]، در این است که در همه آنها از یادگیری Q به عنوان روش یادگیری تقویتی استفاده شده است. با این حال، این یادگیری در مقاله فعلی و دو مقاله قبلی کابرد خاص خود را دارد. این یادگیری در مقاله حاضر به عنوان روش انتخاب اکتشافی سطح پایین برای ابراکتشافی ارائه شده عمل می‌کند اما در مقاله‌های قبلی [14]، [15]، یک ابراکتشافی مبتنی بر یادگیری تقویتی برای اصلاح رفتار یک فرااکتشافی جستجوی محلی تکراری استفاده می‌شود.

در نهایت، این مقاله یک ابراکتشافی مبتنی بر عامل برای پیمانه‌بندی نرم‌افزار ارائه می‌دهد که از یادگیری تقویتی برای انتخاب خودکار عامل‌های تقویتی و عامل‌های تنوعی در هر وضعیت جستجو استفاده می‌کند. همچنین، ما از یک طبقه‌بندی وضعیت مبتنی بر برازندگی استفاده می‌کنیم که نیاز به اطلاعات وابسته به دامنه کمتری دارد و همه موقعیت‌هایی را که ممکن است در طول جستجو اتفاق بیفتد را شامل می‌شود.

نتایج آزمایشات روی یازده سامانه نرم‌افزاری دنیای واقعی با اندازه کوچک و متوسط و ده پوشه از نرم‌افزار بزرگ مقیاس موزیلافایرفاکس با اندازه‌ها و عملکردهای مختلف نشان می‌دهد که الگوریتم پیشنهادی در بیشتر موارد پیمانه‌بندی‌هایی با کیفیت بالاتر از پیمانه‌بندی‌های به دست آمده توسط الگوریتم‌های مقایسه شده تولید می‌کند، در حالی که به زمان اجرای کمتری نیاز دارد.

انگیزه این مقاله این است که با جستجوی هوشمندانه فضای جستجو با استفاده از یک ابراکتشافی مبتنی بر یادگیری تقویتی، کیفیت پیمانه‌بندی و زمان اجرا را به طور همزمان بهبود دهد. ما معتقدیم که با در نظر گرفتن چالش‌های ابراکتشافی‌ها و استفاده از مفهوم سامانه‌های چندعاملی در روش پیشنهادی، می‌توان پیمانه‌بندی‌هایی با کیفیت بالاتر را در زمان کمتر به دست آورد.

در نهایت، سهم‌های علمی این مقاله در زیر خلاصه شده‌اند:

1. استفاده از ابراکتشافی‌ها جهت بهبود عمومیت الگوریتم پیشنهادی.
2. جستجوی هوشمندانه فضای راه‌حل‌ها با استفاده از مفهوم سامانه‌های چندعاملی.
3. طراحی اکتشافی‌های سطح پایین به صورت اجتماع.
4. استفاده از یادگیری تقویتی برای انتخاب خودکار اجتماع مناسب در هر وضعیت از جستجو.
5. اجرای موازی عامل‌ها در اجتماع.
6. استفاده از یک طبقه‌بندی وضعیت مبتنی بر برازندگی که نیاز به اطلاعات وابسته به دامنه کمتری دارد و همه موقعیت‌هایی را که ممکن است در طی جستجو اتفاق بیفتند، شامل می‌شود.
7. بهبود کیفیت راه‌حل‌های پیمانه‌بندی از نظر مقدار معیار TurboMQ.
8. بهبود زمان اجرای روش پیشنهادی نسبت به الگوریتم‌های مبتنی بر جستجوی مقایسه شده.

بخش‌های بعدی مقاله به شرح زیر سازمان‌دهی شده است. بخش 2، برخی از کارهای مرتبط را توصیف می‌کند. بخش 3، مفاهیم اولیه مورد نیاز را تعریف می‌کند. بخش 4، مساله پیمانه‌بندی سامانه‌های نرم‌افزاری را شرح می‌دهد. بخش 5، الگوریتم پیشنهادی را توصیف می‌کند. بخش 6، آزمایش‌ها را توصیف می‌کند. بخش 7، در مورد نتیجه پژوهش انجام شده بحث می‌کند.

2. کارهای مرتبط

در ادبیات، الگوریتم‌های مختلفی برای حل مساله پیمانه‌بندی نرم‌افزار وجود دارند. به طور معمول، می‌توان این الگوریتم‌ها را به دو گروه مجزا سلسله مراتبی و غیرسلسله‌مراتبی طبقه‌بندی کرد. در ادامه، ما برخی از این روش‌ها را معرفی می‌کنیم.

1.2. روش‌های سلسله مراتبی

الگوریتم‌های پیوند کامل (CL) [6]، پیوند تکی (SL) [6]، پیوند متوسط (AL) [6] و پیوند متوسط وزنی (WAL) [6]، از جمله روش‌های سلسله مراتبی سنتی و الگوریتم‌های ترکیبی (CA) [16]، ترکیبی وزنی (WCA) [17] و پیمانه‌بندی تعاونی (CCT) [18]، از دیگر روش‌های سلسله مراتبی موجود هستند. در جدول (1)، این روش‌ها به همراه ویژگی‌های آنها شامل معیار شباهت استفاده شده و نوع آن خلاصه شده است.

جدول (1): برخی از الگوریتم‌های پیمانه‌بندی سلسله مراتبی موجود

روش	معیار شباهت	نوع معیار
SL	Jaccard	Local
CL	Jaccard	Local
AL	Jaccard	Local
WAL	Jaccard	Local
CA	Jaccard	Local
WCA	Ellenberg	Local
CCT	Jaccard-NM and Unbiased Ellenberg-NM	Local

2.2. روش‌های غیرسلسله‌مراتبی

در این بخش، برخی از الگوریتم‌های پیمانه‌بندی غیرسلسله‌مراتبی، به‌ویژه روش‌های مبتنی بر جستجو، را معرفی می‌کنیم که دارای ویژگی‌های مختلف مانند جستجوی سراسری (GS)، جستجوی محلی (LS)، تک‌هدفه (SO)، چندهدفه (MO)، مبتنی بر ویژگی‌های ساختاری (S)، مبتنی بر ویژگی‌های غیرساختاری (non-S)، مبتنی بر یادگیری و بدون یادگیری هستند. جدول (2) این الگوریتم‌ها و ویژگی‌های آنها را خلاصه می‌کند.

میچل [19]، یک الگوریتم به نام Bunch برای پیمانه‌بندی نرم‌افزار ارائه کرد که از الگوریتم ژنتیک و دو نسخه از تپه‌نورد به نام‌های NAHC و SAHC استفاده می‌کند. فضای جستجوی بزرگ الگوریتم Bunch، سرعت این الگوریتم را برای یافتن پیمانه‌بندی مناسب کاهش می‌دهد [3]. از طرف دیگر، تعداد راه‌حل‌های تکراری تولید شده توسط این الگوریتم بسیار زیاد است [3].

پارسا و بوشهریان [20]، یک الگوریتم ژنتیک به نام DAGC ارائه کردند که از یک کدگذاری مبتنی بر جایگشت استفاده می‌کند. DAGC روش کدگذاری پیچیده‌ای دارد ولی فضای جستجو را در مقایسه با Bunch به طور قابل توجهی کاهش می‌دهد.

پرادیت هارمن و یائو [21]، دو الگوریتم ژنتیک چندهدفه به نام‌های ECA و MCA پیشنهاد کردند که هر یک دارای پنج تابع هدف هستند.

تاج‌گردان و همکارانش [22]، یک الگوریتم تخمین توزیع به نام EDA برای پیمانه‌بندی نرم‌افزار ارائه کردند که به جای عملگرهای ژنتیکی از یک مدل احتمالی برای تولید راه‌حل‌های جدید استفاده می‌کند.

هوانگ و لیو [23]، یک تابع هدف جدید به نام MS و همچنین سه الگوریتم پیمانه‌بندی به نام‌های الگوریتم تپه‌نورد، الگوریتم ژنتیک و الگوریتم تکاملی چندعاملی ارائه کردند. آنها در الگوریتم تکاملی چندعاملی [12]، سه عملگر تکاملی با اهداف رقابت، همکاری و خودآموزی برای عامل‌ها (راه‌حل‌ها) ارائه کردند.

کوماری و سرینیواس [13]، یک ابراکتشافی مبتنی بر یادگیری تقویتی برای انتخاب عملگرهای ژنتیکی انتخاب، ترکیب و جهش مناسب در هر نسل از الگوریتم ژنتیک ارائه کردند.

جلالی و همکارانش [25]، یک تابع برازندگی چندهدفه جدید به نام MOF ارائه کردند که ویژگی‌های ساختاری و غیرساختاری را با هم در نظر می‌گیرد.

جدول (2): برخی از الگوریتم‌های پیمانه‌بندی مبتنی بر جستجوی موجود

روش	نوع	Learning-based	S / non-S features	LS / GS	SO / MO	محدودیت اصلی
Bunch [19]	Genetic algorithm	No	S	GS	SO	محدودیت فضا و زمان
NAHC [24]	Hill-climbing algorithm	No	S	LS	SO	گیرکردن در بهینه محلی
SAHC [24]	Hill-climbing algorithm	No	S	LS	SO	گیرکردن در بهینه محلی
DAGC [20]	Genetic algorithm	No	S	GS	SO	محدودیت فضا و زمان
ECA [21]	Two archive Genetic algorithm	No	S	GS	MO	محدودیت فضا و زمان
MCA [21]	Two archive Genetic algorithm	No	S	GS	MO	محدودیت فضا و زمان
EDA [22]	Estimation of Distribution Algorithm	Yes	S	GS	SO	محدودیت فضا و زمان
GA-SMCP [23]	Genetic algorithm	No	S	GS	SO	محدودیت فضا و زمان
EoD [25]	Estimation of Distribution Algorithm	Yes	S, non-S	GS	MO	محدودیت فضا و زمان
SHC [26]	Hill-climbing algorithm	No	non-S	LS	SO	گیرکردن در بهینه محلی
Modified SCSO [27]	Sand Cat Swarm Optimization Algorithm	No	S	GS	SO	محدودیت فضا و زمان
MHypEA [13]	Hyper-heuristic-based Genetic Algorithm	Yes	S	GS	MO	محدودیت فضا و زمان

کارگر و همکارانش [26]، یک الگوریتم تپه‌نورد بـه نـام SHC ارائه کردند که مستقل از زبان برنامه‌نویسی است و از یک گراف وابستگی معنایی برای به دست آوردن معماری نرم‌افزار استفاده می‌کند.

آراسته و همکارانش [27] یک الگوریتم بهینه‌سازی ازدحام گربه شنی گسسته را برای خوشه‌بندی موثر کد منبع نرم‌افـزار ارائـه کردند.

الگوریتم های خوشه‌بندی سریع (FCA) [28]، ACDC [29] و k-means [6]، نیـز متعلـق بـه دسـته روش‌هـای پیمانه‌بنـدی غیرسلسله‌مراتبی هستند.

3. تعریف مفاهیم اولیه

در این بخش، مفاهیم اولیه مورد نیاز تعریف می‌شوند.

1.3. یادگیری Q-

یادگیری Q- یکی از روش‌های یادگیری تقـویتی اسـت کـه از ویژگی‌های این یادگیری، یعنی آزمون و خطا و پاداش تاخیری، پیروی می‌کند. در یادگیری Q-، هر جفت وضعیت-عمل دارای یک مقدار Q- است که مقدار پاداش انباشته کل را نشان می‌دهد و توسط یک تابع Q- محاسبه می‌شود. مقادیر Q- همه جفت‌های وضعیت-عمل در یک جدول Q- ذخیره می‌شوند. فرض کنیـد $S = \{s_1, s_2, \dots, s_n\}$ و $A = \{a_1, a_2, \dots, a_m\}$ به ترتیب یـک مجموعه از وضعیت‌های ممکن و یک مجموعـه از عمل‌هـای قابل انتخاب را نشان می‌دهند. مقادیر Q- با استفاده از رابطه (3) محاسبه می‌شود که در این رابطه، r_{t+1} سیگنال تقویتی فـوری، $\alpha \in [0,1]$ نـرخ یـادگیری، $\gamma \in [0,1]$ فـاکتور تخفیـف و $Q(s_t, a_t)$ مقدار Q- در زمان t است.

$$Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (3)$$

2.3. نقشه آشوب منطقی

در ریاضیات، نقشه‌های آشوبی برای تولید دنباله‌های آشوبی استفاده می‌شوند [30]. نقشه منطقی [30]، یکی از پرکاربردترین نقشه‌های آشوبی است. نقشه منطقی یک نقشه چندجمله‌ای مرتبه دوم است و رابطه (4) آن را توصیف می‌کند.

$$X_{n+1} = rX_n(1 - X_n) \quad (4)$$

در این رابطه، r یک پارامتر کنترل بین 0 و 4 است که رفتار متغیر X را تعیین می‌کند. متغیر X می‌تواند همگرا ($0 \leq r \leq 3$)، دوره‌ای ($3 < r \leq 3.56$) یا بی‌نظم ($3.56 < r \leq 4$) باشد. واضح است که تحت شرایطی که $initial\ X_0 \in [0,1]$ و $X_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$، آنگاه $X \in [0,1]$ است.

از آنجایی که اعداد آشوبی تولید شده توسط نقشه منطقی در بازه 0-1 هستند، از توابعی برای نگاشت اعداد آشوبی به اعدادی در بازه مورد نظر استفاده می‌شود. فرض کنید که اعداد آشوبی در بازه X_{low} به X_{high} هستند و عدد آشوبی تولید شده x است. رابطه (5) برای نگاشت عدد آشوبی x به عددی صحیح در بازه 1 به L استفاده می‌شود [30].

$$f(x) = ROUND\left(\frac{L-1}{X_{high} - X_{low}}(x - X_{low}) + 1\right) \quad (5)$$

3.3. تعریف اجتماع

الگوریتم پیشنهادی ما یک ابراکتشافی مبتنی بر اجتماع است؛ یعنی در هر وضعیت از جستجو به جای یک اکتشافی سطح پایین (عامل)، یک اجتماع از عامل‌ها انتخاب می‌شود. ما اجتماع را گروهی از عامل‌ها تعریف می‌کنیم که متعلق به دسته یکسانی از اکتشافی‌ها (دسته‌های محلی و آشفتگی) هستند و هر کدام از آنها می‌توانند دیدگاه خاص خود را برای حل مساله داشته باشند. برای مثال، در اجتماع محلی، عامل‌ها می‌توانند الگوریتم‌های جستجوی محلی متفاوت، الگوریتم‌های جستجوی محلی یکسان یا الگوریتم‌های جستجوی محلی یکسان با متغیرها و مولفه‌های جستجوی متفاوت را پیاده‌سازی کنند.

4. طرح مساله

پیمانه‌بندی نرم‌افزار فرآیند گروه‌بندی موجودیت‌های نرم‌افزار به پیمانه‌ها است به طوری که موجودیت‌هایی که وابستگی بالایی دارند در پیمانه یکسان گروه‌بندی می‌شوند. پیمانه‌بندی به درک سامانه‌های نرم‌افزاری کمک می‌کند و باعث آسان‌تر شدن فرآیند نگهداری آنها می‌شود. عموما، گراف‌ها برای نمایش سامانه‌های نرم‌افزاری پیچیده استفاده می‌شوند [6]. گراف وابستگی موجودیت یک نوع گراف خوب‌شناخته‌شده است که دیدگاهی انتزاعی از ساختار نرم‌افزار را نشان می‌دهد. در این گراف، گره‌ها و یال‌ها به ترتیب موجودیت‌های سامانه نرم‌افزاری و روابط بین آنها را نشان می‌دهند. ما از این گراف به عنوان ورودی در الگوریتم پیمانه‌بندی پیشنهادی استفاده می‌کنیم. فرض کنید یک گراف وابستگی موجودیت به صورت $ADG = (V, E)$ برچسب‌گذاری شده است، که در آن $V = \{v_1, v_2, \dots, v_n\}$ مجموعه‌ای از n موجودیت است و مجموعه روابط بین آنها برابر با $E \subseteq V \times V = \{(v_i, v_j) | v_i, v_j \in V \wedge i \neq j\}$ باشند. فرآیند پیمانه‌بندی منجر به گروه‌بندی همه موجودیت‌ها به k پیمانه غیرهمپوشان $\{m_1, m_2, \dots, m_k\}$ می‌شود، که در آن $M_1 \cup M_2 \cup \dots \cup M_k = V$، $M_i \neq \emptyset$، $M_i \cap M_j = \emptyset$، $i \neq j$ و $i, j = 1, 2, \dots, k$ است به‌طوری که مقدار تابع هدف (رابطه (2))، یعنی TurboMQ، بیشینه شود. در مهندسی نرم‌افزار، ارتباط‌های بالا در یک پیمانه و میزان وابستگی متقابل پایین بین پیمانه‌ها به عنوان ویژگی‌های سامانه‌های نرم‌افزاری خوب‌طراحی‌شده در نظر گرفته می‌شوند [3].

5. الگوریتم پیشنهادی

در این بخش، یک ابراکتشافی مبتنی بر عامل به نام AbHyh-SSM برای پیمانه‌بندی سامانه‌های نرم‌افزاری ارائه می‌شود. ابراکتشافی انتخاب پیشنهادی از دو سطح بالا و پایین تشکیل

می‌شود. در سطح بالا، لازم است یک اکتشافی سطح بالا که شامل دو مولفه روش انتخاب اجتماع و روش پذیرش حرکت است، مشخص شود. AbHyh-SSM از یادگیری Q- (بخش 1.3) به عنوان روش انتخاب اجتماع استفاده می‌کند و قادر است به طور هوشمندانه اجتماع‌های مناسب را در طول مراحل متفاوت فرآیند پیمانه‌بندی انتخاب کند. الگوریتم پیشنهادی از روش همه حرکات به عنوان روش پذیرش حرکت استفاده می‌کند؛ یعنی همیشه راه‌حل تولید شده جدید را می‌پذیرد.

سطح پایین ابراکتشافی پیشنهادی شامل یک نمایش از مساله، توابع ارزیابی و مجموعه‌ای از اجتماع‌ها است. در این مقاله، کدگذاری مبتنی بر مقدار استفاده‌شده در Bunch [19]، برای نمایش یک راه‌حل از مساله استفاده می‌شود، اما با اشاره به مرجع [1]، ما حداکثر تعداد پیمانه‌ها را در سامانه‌های نرم‌افزاری بزرگ (پوشه‌های موزیلا فایرفاکس) به $min(100, n/3)$ محدود می‌کنیم. همچنین، TurboMQ پیشنهادشده در Bunch [19] برای اندازه‌گیری کیفیت پیمانه‌بندی‌های به دست آمده در طول جستجو استفاده می‌شود. AbHyh-SSM یک ابراکتشافی مبتنی بر اجتماع است؛ یعنی در هر مرحله از جستجو به جای یک اکتشافی سطح پایین (عامل) یک اجتماع از عامل‌ها انتخاب می‌شوند که به صورت هدفمند برای جستجوی فضای حالت مساله پیمانه‌بندی طراحی شده‌اند. الگوریتم (1) شبه کد سطح بالای AbHyh-SSM پیشنهادی را نشان می‌دهد.

همان‌طور که در الگوریتم (1) نشان داده شده است، پس از انتخاب بهترین اجتماع؛ یعنی یک اجتماع با بالاترین مقدار Q- (خط 6)، عامل‌های اجتماع انتخاب‌شده به صورت موازی و با شروع از راه‌حل فعلی الگوریتم اجرا می‌شوند و بهترین راه‌حل تولیدشده توسط عامل‌ها به عنوان راه‌حل جدید در نظر گرفته می‌شود (خط 7). سپس، با استفاده از یک معیار پذیرش حرکت در مورد پذیرش یا عدم پذیرش راه‌حل جدید تصمیم گرفته می‌شود (خط 8). پس از آن، وضعیت بعدی بر اساس برازندگی نرمال‌شده آخرین راه‌حل پذیرفته‌شده مشخص می‌شود (خط 9). مقدار Q- اجتماع انتخاب‌شده بر اساس کارایی اجرای آن بروزرسانی می‌شود (خط 10). سپس، وضعیت بروزرسانی می‌شود (خط 11) و بهترین راه‌حل سراسری ثبت می‌شود (خط 12). یک اجتماع برای تکرار بعدی الگوریتم انتخاب می‌شود (خط 5). در ادامه، جزئیات الگوریتم پیشنهادی توصیف خواهد شد.

الگوریتم (1): شبه کد سطح بالای الگوریتم پیشنهادی

```
1: Initialization( )
2: GlobalBest = sol_initial
3: Sol_current = Sol_initial
4: Determine_initial_state( )
5: while (!stopping_criteria) do
6:     selected_action = select_coalition( )
7:     Sol_new = Produce_new_solution(selected_action,
       Sol_current)
8:     Move_acceptance()
9:     Determine_next_state( )
10:    Update_reward of action( )
11:    s_t = s_t+1
12:    Update_GlobalBest( )
13:end while
```

1.5. مجموعه عمل‌ها

از آنجایی که ما در الگوریتم پیشنهادی خود از یادگیری تقویتی استفاده می‌کنیم، تعریف مجموعه عمل‌ها و مجموعه وضعیت‌ها لازم است. همان‌طور که در بالا توضیح داده شد، در روش پیشنهادی هر عمل، یک اجتماع را نشان می‌دهد. AbHyh-SSM از دو اجتماع محلی و آشفتگی به ترتیب با هدف تقویت و تنوع استفاده می‌کند. از آنجایی که هر کدام از عامل‌های موجود در یک اجتماع می‌توانند دیدگاه خاص خود را در مورد مساله داشته باشند، آنها مجاز هستند که الگوریتم‌های متفاوتی را از دسته اکتشافی اجتماع خود پیاده‌سازی کنند. این امر موجب می‌شود مساله پیمانه‌بندی از دیدگاه‌های مختلف بررسی و جستجوی بهتری انجام شود. از طرفی، اجرای موازی عامل‌ها در هر اجتماع باعث افزایش قدرت محاسباتی می‌شود.

2.5. مجموعه وضعیت‌ها

در یادگیری تقویتی، وضعیت، شرایط محیطی برای تصمیم‌گیری در مورد یک عمل را نشان می‌دهد [31]. ما از یک طبقه‌بندی

وضعیت مبتنی بر برازندگی، اشاره‌شده در مرجع [31]، استفاده می‌کنیم که نیاز به اطلاعات وابسته به دامنه کمتری دارد. از آنجایی که برازندگی مساله‌های مختلف دامنه‌های متفاوتی دارد، توسط میانگین حرکت برازندگی (f_{new}) محاسبه‌شده از تکرار اول تا تکرار فعلی نرمال می‌شود. برازندگی نرمال‌شده با استفاده از رابطه (6) به دست می‌آید. در این مقاله، وضعیت‌ها به سه طبقهز $norm(f) \in [0,0.67), [0.67,1.33), [1.33,\infty)$ تقسیم می‌شوند.

$$norm(f) = \frac{f}{Avg(f_{new})} \quad (6)$$

3.5. سیگنال تقویتی فوری

در تابع Q- (رابطه (3))، r یک سیگنال یادگیری تقویتی است که پاداش یا جریمه را نشان می‌دهد. این متغیر از یادگیری Q- توسط کاربر تعریف می‌شود. با اشاره به مرجع [31]، اگر بهترین راه‌حل سراسری بتواند در پایان اجرای عامل‌های اجتماع فعلی بهبود یابد، جفت وضعیت-عمل یک پاداش $r = 1$ دریافت می‌کند؛ در غیر این صورت، یک جریمه $r = -1$ اعمال می‌شود.

4.5. متغیرهای تابع Q-

فاکتور تخفیف و نرخ یادگیری دو متغیر در تابع Q- (رابطه (3)) هستند. فاکتور تخفیف، γ، تاثیر پاداش آینده را تعیین می‌کند. در AbHyh-SSM، با اشاره به مرجع [31]، مقدار این متغیر 0/8 در نظر گرفته می‌شود. نرخ یادگیری، α، نسبت پذیرش اطلاعات جدید آموخته‌شده یا نگه‌داشتن اطلاعات موجود را نشان می‌دهد. در این مطالعه، با اشاره به مرجع [31]، از رابطه (7) برای کنترل پویای این متغیر استفاده می‌شود که در آن، $iteration_{current}$ تکرار فعلی الگوریتم و $iteration_{max}$ حداکثر تعداد تکرار مجاز را برای اجرای الگوریتم پیشنهادی نشان می‌دهد.

$$\alpha_t = 1 - \left(0.9 \times \frac{iteration_{current}}{iteration_{max}}\right) \quad (7)$$

5.5. متغیر کنترل نقشه آشوب منطقی

در الگوریتم پیشنهادی، برای بهبود سرعت همگرایی و کارایی، در طراحی برخی عامل‌ها از اعداد آشوبی به جای اعداد تصادفی استفاده می‌شود. با استفاده از اعداد آشوبی تنوع حفظ و از افتادن در بهینه محلی جلوگیری می‌شود. از آنجایی که ما به رفتار آشوبی متغیر X نیاز داریم، مقدار r را برابر 4 در نظر می‌گیریم.

6.5. عامل‌های اجتماع آشفتگی

اجتماع آشفتگی شامل سه عامل است. این عامل‌ها با هدف حفظ تنوع، راه‌حل دریافتی را به روش‌های مختلف مختل می‌کنند تا به مناطق ناشناخته فضای جستجو دست یابند. عامل‌های طراحی‌شده برای اجتماع آشفتگی در ادامه شرح داده خواهند شد.

1.6.5. عامل 1 اجتماع آشفتگی

الگوریتم (2) رفتار اولین عامل از اجتماع آشفتگی را توصیف می‌کند. در این الگوریتم، ابتدا، یک عدد تصادفی صحیح t بین 1 تا $\lfloor n/2 \rfloor$ تولید می‌شود که تعداد تکرار الگوریتم را نشان می‌دهد. n تعداد ژن‌های کروموزوم است. سپس، در هر تکرار، دو ژن با استفاده از اعداد آشوبی انتخاب و شماره پیمانه آنها با یکدیگر تعویض می‌شود. برای انتخاب هر ژن، ابتدا، یک عدد آشوبی با استفاده از رابطه (4) تولید می‌شود و سپس عدد آشوبی تولیدشده با استفاده از رابطه (5) به یک عدد صحیح در بازه 1 به n نگاشت می‌شود. لازم به ذکر است که این عامل اجازه جابجایی‌های تکراری و همچنین جابجایی شماره پیمانه ژن‌هایی با پیمانه یکسان را نمی‌دهد. استفاده از اعداد آشوبی توسط این عامل باعث می‌شود که اعداد تکراری کمتری برای شماره ژن‌ها تولید شود. در نتیجه، محاسبات کمتر و راه‌حل‌های آشفته‌تری تولید می‌شود.

2.6.5. عامل 2 اجتماع آشفتگی

عامل دوم از اجتماع آشفتگی الگوریتم (3) را پیاده‌سازی می‌کند. این عامل از دو مفهوم vertex cohesion و vertex coupling،

اشاره‌شده در مرجع [32]، برای شناسایی هدفمند گره‌هایی استفاده می‌کند که باید شماره پیمانه آنها تغییر کند. vertex cohesion نسبت تعداد راس‌های متصل‌شده به راس u در داخل پیمانه شامل این راس به تعداد کل اتصال‌های درونی در این پیمانه است. vertex coupling نسبت تعداد راس‌های اتصال خارجی به راس u به تعداد کل راس‌های اتصال خارجی ممکن به این پیمانه است.

الگوریتم (2): شبه کد عامل 1 اجتماع آشفتگی

```
Ensure: Perturbated solution s′
 1: s′ ← received_solution
 2: t ← generate an integer number between 1 to ⌊n/2⌋
        randomly (n: the length of chromosome)
 3: i ← 1
 4: while (i<=t) do
 5:     x ← generate an integer number between 1 to n
            using chaos theory
 6:     y ← generate an integer number between 1 to n
            using chaos theory
 7:     s′ ← swap (s′ (x), s′ (y))
 8:     i ← i+1
 9: end while
```

الگوریتم (3): شبه کد عامل 2 اجتماع آشفتگی

```
Ensure: Perturbated solution s′
 1: s ← received_solution
 2: s′ ← s
 3: k ← number of modules in s
 4: TurboMQ ← fitness of s (fs)
 5: i ← 1
 6: while (i<=n   n: the number of nodes) do
 7:     compute vertex cohesion for node i of the s
 8:     compute vertex coupling for node i of the s
 9:     if (vertex cohesion-vertex coupling<TurboMQ/k)
10:         s′(i) ← choose from the s, the module number of
                one of the dissimilar-module neighbors of
                node i, randomly
11:    end if
12:    i ← i+1
13: end while
```

فرض کنید s راه‌حل دریافتی است، k تعداد پیمانه‌های s است و TurboMQ برازندگی s است. این عامل، برای همه گره‌ها در s رابطه (8) را بررسی می‌کند.

$$\text{vertex cohesion-vertex coupling} < \text{TurboMQ/k} \qquad (8)$$

اگر گره i این رابطه را ارضا کند، شماره پیمانه آن به طور تصادفی به شماره پیمانه یکی از همسایه‌های غیرهم‌پیمانه خود در s تغییر می‌کند و در صورت عدم وجود چنین همسایه‌ای شماره پیمانه آن به طور تصادفی از میان شماره پیمانه‌های موجود در s انتخاب می‌شود. لازم به ذکر است که اگر هیچ کدام از گره‌ها این رابطه را ارضا نکنند، یک گره به طور تصادفی انتخاب می‌شود.

3.6.5. عامل 3 اجتماع آشفتگی

الگوریتم (4) رفتار عامل سوم از اجتماع آشفتگی را توصیف می‌کند. این عامل مشابه با عامل آشفتگی دوم رفتار می‌کند با این تفاوت که در هر مرحله، به منظور بررسی درستی رابطه (8) برای هر گره i، متغیرهای این رابطه با توجه به جدیدترین راه‌حل ایجادشده، نه بر اساس اولین راه‌حل دریافتی s محاسبه می‌شوند. همچنین اگر گره‌ای این رابطه را ارضا کند، تغییر شماره پیمانه آن نیز بر اساس جدیدترین راه‌حل تولیدشده انجام می‌شود.

الگوریتم (4): شبه کد عامل 3 اجتماع آشفتگی

```
Ensure: Perturbated solution s′
 1: s′ ← received_solution
 2: i← 1
 3: while (i<=n   n: the number of nodes) do
 4:     k ← number of modules in s′
 5:     TurboMQ ← fitness of s′ (fs′)
 6:     compute vertex cohesion for node i of the s′
 7:     compute vertex coupling for node i of the s′
 8:     if (vertex cohesion-vertex coupling<TurboMQ/k)
 9:         s′(i) ← choose from the s′, the module number of
                one of the dissimilar-module neighbors
                of node i, randomly
10:    end if
11:    i ← i+1
12: end while
```

7.5. عامل‌های اجتماع محلی

اجتماع محلی شامل سه عامل است که روش‌های جستجوی محلی را با هدف تقویت جستجو اجرا می‌کنند. از آنجایی که ما الگوریتم پیشنهادی را روی ده پوشه از موزیلا فایرفاکس و یازده

سامانه نرم‌افزاری با اندازه کوچک آزمون می‌کنیم، عامل‌های اجتماع محلی برای هر کدام از این دو دسته به صورت متفاوت طراحی شده‌اند. دلیل این امر این است که پوشه‌های موزیلا فایرفاکس از مقیاس بالاتری برخوردار هستند و عامل‌های اجتماع محلی باید به‌گونه‌ای طراحی شوند که زمان اجرای طولانی نداشته باشند؛ یعنی از سرعت بالایی برخوردار باشند. از طرف دیگر، سرعت بالای عامل‌های محلی در سامانه‌های نرم‌افزاری با اندازه کوچک، منجر به همگرایی زودرس و گیرافتادن آنها در بهینه‌های محلی می‌شود. عامل‌های اجتماع محلی 1، برای ده پوشه از موزیلا فایرفاکس و اجتماع محلی 2، برای یازده سامانه نرم‌افزاری با اندازه کوچک و متوسط، در ادامه توصیف می‌شوند.

1.7.5. عامل 1 اجتماع محلی 1

اولین عامل از اجتماع محلی 1، الگوریتم (5) را پیاده‌سازی می‌کند. این الگوریتم یک روش جستجوی محلی را توصیف می‌کند. فرض کنید k تعداد پیمانه‌های راه‌حل دریافتی و TurboMQ برازندگی آن است. در این الگوریتم، برای حرکت از یک پیمانه‌بندی به پیمانه‌بندی همسایه آن، یک گره به نحوی انتخاب می‌شود که در رابطه (8) صدق می‌کند و سپس شماره پیمانه این گره به شماره پیمانه یکی از همسایه‌های غیرهم‌پیمانه آن در راه‌حل فعلی تغییر می‌کند. در صورت عدم وجود چنین همسایه‌ای، شماره پیمانه گره انتخاب‌شده به طور تصادفی از میان شماره پیمانه‌های موجود در راه‌حل فعلی انتخاب می‌شود. اگر گره‌ای با این ویژگی وجود نداشته باشد یا راه‌حل همسایه نتواند بهترین راه‌حل یافت‌شده را بهبود دهد، اجرای الگوریتم قبل از رسیدن به حداکثر زمان مجاز خاتمه می‌یابد.

2.7.5. عامل‌های 2 و 3 اجتماع محلی 1

عامل‌های دوم و سوم از اجتماع محلی 1 رفتار مشابهی دارند و الگوریتم (6) را اجرا می‌کنند. در این الگوریتم، ابتدا برای هر گره تعداد یال‌ها از آن به همه پیمانه‌ها محاسبه می‌شود. سپس، یک گره که تعداد یال‌های آن به پیمانه خود کمتر از پیمانه‌های دیگر است، انتخاب می‌شود. اگر گره‌ای با این ویژگی وجود نداشته باشد، اجرای الگوریتم قبل از رسیدن به حداکثر زمان مجاز خاتمه می‌یابد. از طرف دیگر، اگر چندین گره با این ویژگی وجود داشته باشد، یک گره به طور تصادفی انتخاب می‌شود. در گام بعدی، گره انتخاب‌شده به پیمانه‌ای که بیشترین تعداد یال را با آن دارد، انتقال می‌یابد. اگر چندین پیمانه با این ویژگی وجود دارد، این گره به بهترین پیمانه انتقال می‌یابد. این عملیات تا زمانی‌که شرایط خاتمه ارضا نشوند، تکرار می‌شوند. بخش‌های تصادفی این الگوریتم ممکن است منجر به عملکرد متفاوت این دو عامل شود.

الگوریتم (5): شبه کد عامل 1 اجتماع محلی 1

```
Ensure: Improved solution s_best
 1: s ← received_solution
 2: s_best ← s
 3: k ← number of modules in s
 4: TurboMQ ← fitness of s (fs)
 5: while (stopping condition not reached) do
 6:     i ← 1
 7:     while (i<=n   n: the number of nodes) do
 8:         compute vertex cohesion for node i of the s
 9:         compute vertex coupling for node i of the s
10:         if (vertex cohesion-vertex coupling<TurboMQ/k)
11:             s(i) ← choose from the s, the module number of
                       one of the dissimilar-module neighbors of
                       node i, randomly
12:             TurboMQ ← fitness of s (fs)
13:             break
14:         else
15:             i ← i+1
16:         end if
17:     end while
18:         if (TurboMQ>f(s_best))
19:             s_best ← s
20:             k ← number of modules in s
21:         else
22:             break
23:         end if
24:end while
```

3.7.5. عامل‌های 1 و 2 اجتماع محلی 2

عامل‌های اول و دوم از اجتماع محلی 2 روش جستجوی محلی تکراری را پیاده‌سازی می‌کنند. الگوریتم (7)، شبه کد این روش را نشان می‌دهد. تفاوت این دو عامل در روش جستجوی محلی استفاده‌شده توسط آنها است. عامل اول از روش NAHC [24]

و عامل دوم از روش SAHC [24] در بخش جستجوی محلی استفاده می‌کند. هر دو عامل برای قسمت آشفتگی راه‌حل از الگوریتم (2) (عامل 1 اجتماع آشفتگی) و برای قسمت معیار پذیرش از روش فقط بهبود استفاده می‌کنند. در روش فقط بهبود، بهینه محلی جدید فقط زمانی پذیرفته می‌شود که بهترین راه‌حل یافت‌شده توسط الگوریتم را بهبود دهد. در این عامل‌ها، با هدف تنوع بیشتر، برای تعیین تعداد تکرار الگوریتم (2) در هر بار اجرا، ابتدا یک عدد آشوبی با استفاده از رابطه (4) تولید می‌شود و سپس عدد آشوبی تولیدشده با استفاده از رابطه (5) به یک عدد صحیح در بازه 1 به $\lfloor n/2 \rfloor$ نگاشت می‌شود که n تعداد ژن‌های کروموزوم است.

الگوریتم (6): شبه کد عامل‌های 2 و 3 اجتماع محلی 1

Ensure: Improved solution
1: while (stopping condition not reached) do
2: calculate the number of links from each node to all modules
3: select a node that has fewer links to its module than other modules
/* if there are several nodes, select a node randomly */
4: transfer the selected node to the module that has the highest number of links to it
/* if there are several modules, select the best module in terms of modularization quality (MQ) */
5: end while

الگوریتم (7): شبه کد عامل‌های 1 و 2 اجتماع محلی 2

Ensure: Improved solution s″
1: $s_0 \leftarrow$ received_solution
2: s ← local_search(s_0)
3: while stopping condition not reached do
4: s′ ← solution_perturbation(s, history)
5: s″ ← local_search(s′)
6: s ← acceptance_criterion (s, s″, history)
7: end while

4.7.5. عامل 3 اجتماع محلی 2

عامل سوم از اجتماع محلی 2 روش جستجوی همسایگی متغیر را پیاده‌سازی می‌کند. الگوریتم (8) شبه کد این روش را نشان می‌دهد. این عامل از روش NAHC [24] به عنوان مولفه جستجوی محلی استفاده می‌کند. از آنجایی که الگوریتم جستجوی همسایگی متغیر از ایده تغییر همسایگی استفاده می‌کند، دو استراتژی همسایگی برای این عامل به صورت زیر تعریف می‌شوند:

- استراتژی اول همان همسایگی رایج در پیمانه‌بندی نرم‌افزار است که در آن دو راه‌حل پیمانه‌بندی همسایه در نظر گرفته می‌شوند اگر آنها بتوانند به سادگی با انتقال یک گره از یک پیمانه به دیگری به یکدیگر تبدیل شوند [6]. در نتیجه، برای یک پیمانه‌بندی با k پیمانه و n گره، حداکثر تعداد همسایه‌ها $n \times k$ است.
- در استراتژی دوم، ابتدا یک گره i از راه‌حل s که رابطه (8)، اشاره شده در مرجع [32]، را برآورده می‌کند برای تولید راه‌حل همسایه انتخاب می‌شود. سپس، شماره پیمانه گره انتخاب‌شده به طور تصادفی به شماره پیمانه یکی از همسایه‌هایش در s تغییر می‌کند. اگر گره i و همه همسایگانش شماره پیمانه یکسانی در s داشته باشند، شماره پیمانه گره i به طور تصادفی به یکی از شماره‌های پیمانه در s یا یک شماره پیمانه جدید تغییر داده می‌شود. اگر هیچ گره‌ای از s رابطه بالا را ارضا نکند، یک گره به طور تصادفی انتخاب می‌شود و شماره پیمانه آن به طور تصادفی به یکی از شماره‌های پیمانه در s یا یک شماره پیمانه جدید تغییر می‌کند.

الگوریتم (8): شبه کد عامل 3 اجتماع محلی 2

Ensure: Improved solution s_{best}
1: s ← received_solution, choose {N_k} , k = 1, 2
2: s_{best} ← s
3: repeat
4: k = 1
5: repeat
6: s′ ← random_solution(N_k))
7: s″ ← local_search_NAHC(s′)
8: if (f(s″) > f(s′)) then
9: s ← s″
10: else
11: k ← k + 1
12: end if
13: if (f(s″) > f(s_{best})) then
14: s_{best} ← s″
15: end if
16: until k = 2
17: until stopping condition is not reached

8.5. شرط خاتمه الگوریتم

از آنجایی که الگوریتم پیشنهادی یک روش مبتنی بر تکامل است، ما شرط خاتمه را همگرا شدن آن به یک راه‌حل تعریف می‌کنیم. این الگوریتم زمانی متوقف می‌شود که بهترین راه‌حل سراسری در 10 تکرار متوالی بهبود داده نشود (یعنی الگوریتم همگرا شده باشد). حداکثر تعداد تکرار مجاز برای الگوریتم پیشنهادی نیز 1000 در نظر گرفته شده است.

9.5. مرتبه فضایی الگوریتم پیشنهادی

برای محاسبه مرتبه فضایی الگوریتم پیشنهادی، ابتدا، ما فرض‌های زیر را در نظر می‌گیریم:

n: تعداد گره‌ها در گراف وابستگی موجودیت

k: تعداد پیمانه‌های یک راه‌حل پیمانه‌بندی

برای محاسبه مرتبه فضایی روش پیشنهادی (الگوریتم (1))، لازم است مرتبه فضایی برای همه دستورات محاسبه شود و حداکثر مرتبه فضایی از بین آنها به عنوان مرتبه فضایی الگوریتم پیشنهادی انتخاب شود. در ادامه، مرتبه فضایی خطوط مختلف الگوریتم (1) آورده شده است:

خط 1: تولید یک راه‌حل اولیه (کروموزوم) تصادفی به طول n ($O(n)$) و مقداردهی اولیه جدول Q- با سه سطر و دو ستون ($O(1)$)، در نتیجه مرتبه فضایی $O(n)$ است.

خط 2: راه‌حل اولیه به عنوان بهترین راه‌حل سراسری در نظر گرفته می‌شود، در نتیجه مرتبه فضایی $O(n)$ است.

خط 3: راه‌حل اولیه به عنوان راه‌حل فعلی در نظر گرفته می‌شود، در نتیجه مرتبه فضایی $O(n)$ است.

خط 4: وضعیت اولیه با استفاده از رابطه (6) محاسبه می‌شود، در نتیجه مرتبه فضایی $O(1)$ است.

خط 5: شرط خاتمه بررسی می‌شود، در نتیجه مرتبه فضایی $O(1)$ است.

خط 6: با توجه به وضعیت جستجو، بهترین اجتماع با توجه به جدول Q- انتخاب می‌شود، در نتیجه مرتبه فضایی $O(1)$ است.

خط 7: عامل‌های اجتماع آشفتگی یا اجتماع محلی به طور موازی اجرا می‌شوند. مرتبه فضایی عامل 1 اجتماع آشفتگی $O(1)$، عامل 2 اجتماع آشفتگی $O(n)$، عامل 3 اجتماع آشفتگی $O(1)$، عامل 1 اجتماع محلی 1 $O(1)$، عامل‌های 2 و 3 اجتماع محلی 1 $O(1)$، عامل 1 اجتماع محلی 2 $O(n)$، عامل 2 اجتماع محلی 2 $O(n \times k \times n)$ و عامل 3 اجتماع محلی 2 $O(n)$ است. در نتیجه مرتبه فضایی این خط $O(n^2k)$ است.

خط 8: در مورد پذیرش راه‌حل جدید به عنوان راه‌حل فعلی تصمیم می‌گیرد، در نتیجه مرتبه فضایی $O(1)$ است.

خط 9: وضعیت بعدی جستجو با استفاده از رابطه (6) محاسبه می‌شود، در نتیجه مرتبه فضایی $O(1)$ است.

خط 10: درایه وضعیت-عمل در جدول Q- با استفاده از رابطه (3) بروزرسانی می‌شود، در نتیجه مرتبه فضایی $O(1)$ است.

خط 11: وضعیت بعدی به عنوان وضعیت فعلی در نظر گرفته می‌شود، در نتیجه مرتبه زمانی $O(1)$ است.

خط 12: بهترین راه‌حل سراسری در صورت لزوم بروزرسانی می‌شود، در نتیجه مرتبه زمانی $O(1)$ است.

با توجه به موارد ذکر شده در بالا، مرتبه فضایی الگوریتم پیشنهادی $O(n^2k)$ است.

6. آزمایش‌ها

این بخش تنظیمات آزمایشی، نتایج آزمایش‌ها و پژوهش‌های آینده را ارائه می‌دهد. لازم به ذکر است که داده‌ها و کد شبیه‌سازی مقاله در پیوند زیر قابل دسترسی است:
https://github.com/mahjoubeh-t/Software-Modularization
آزمایش‌ها در نرم‌افزار متلب نسخه 2017 بر روی یک سیستم کامپیوتری با مشخصات رم 4 گیگابایت و پردازنده پنج هسته‌ای 2/4 گیگاهرتز اجرا شده است.

1.6. تنظیمات آزمایشی

در این بخش، به بیان تنظیمات آزمایشی لازم برای ارزیابی الگوریتم پیشنهادی و مقایسه آن با سایر الگوریتم‌ها پرداخته خواهد شد.

1.1.6. سامانه نرم‌افزاری

یازده سامانه نرم‌افزاری دنیای واقعی با اندازه کوچک و متوسط و ده پوشه از موزیلا فایرفاکس با قابلیت‌ها و اندازه‌های مختلف برای ارزیابی و مقایسه الگوریتم پیشنهادی انتخاب شده است که جداول (3) و (4) به‌ترتیب ویژگی‌های آنها را نشان می‌دهند.

2.1.6. تجزیه متخصص

یک الگوریتم معتبر است اگر نتیجه پیمانه‌بندی آن نزدیک به تجزیه فرد خبره باشد. در این مقاله، از تجزیه خبره سامانه mtunis برای ارزیابی دقت و قابلیت اطمینان الگوریتم پیشنهادی استفاده می‌شود.

3.1.6. ارزیابی نتایج

در این مقاله از تابع هدف TurboMQ (رابطه (2)) برای ارزیابی کیفیت پیمانه‌بندی‌ها استفاده می‌شود. برای ارزیابی قابلیت اطمینان الگوریتم پیشنهادی از معیارهای خارجی MoJo [6]، Edge MoJo [6] و Fm [6] (میانگین هارمونیک Precision و Recall) استفاده می‌شود. علاوه بر این، زمان اجرای الگوریتم روی موردهای مطالعه یکسان با الگوریتم‌های دیگر مقایسه می‌شود.

4.1.6. سوال‌های پژوهش

برای ارزیابی اثربخشی الگوریتم پیشنهادی، سوال‌های پژوهش زیر پاسخ داده می‌شوند.

1. آیا الگوریتم پیشنهادی پیمانه‌بندی نزدیک به تجزیه فرد خبره تولید می‌کند؟
2. آیا ابراکتشافی مبتنی بر عامل پیشنهادی از نظر معیار TurboMQ بهتر از الگوریتم‌های پیمانه‌بندی سلسله‌مراتبی و غیرسلسله‌مراتبی مقایسه‌شده عمل می‌کند؟
3. آیا الگوریتم پیشنهادی پایدار است؟
4. آیا زمان اجرای الگوریتم پیشنهادی کمتر از الگوریتم‌های مبتنی بر جستجوی مقایسه‌شده است؟

2.6. نتایج آزمایش‌ها

این بخش نتایج مطالعه تجربی را ارائه می‌دهد. هدف، مقایسه روش پیشنهادی با برخی از الگوریتم‌های سلسله‌مراتبی و غیرسلسله‌مراتبی است. الگوریتم‌های پیمانه‌بندی مبتنی بر جستجویی که ما با الگوریتم پیشنهادی از نظر TurboMQ مقایسه کردیم Bunch-GA [19]، DAGC [20]، Bunch-NAHC [24]، SHC [26]، GA-SMCP [23]، EOD [25] و SCSO [27] هستند. ویژگی‌های این الگوریتم‌ها به‌طور خلاصه در جدول (2) آورده شده است. علاوه بر این، ما از الگوریتم‌های پیمانه‌بندی سلسله‌مراتبی ادغامی مانند single linkage [6]، complete linkage [6]، average linkage [6] و WCA-UE [17] برای مقایسه استفاده می‌کنیم. ویژگی‌های این الگوریتم‌ها به‌طور خلاصه در جدول (1) آورده شده است. الگوریتم‌های FCA [28]، k-means [6] و ACDC [29] نیز برای مقایسه انتخاب شده‌اند.

جدول (3): توصیف سامانه‌های نرم‌افزاری استفاده شده

Name	Description	#Files	#Links
mtunis	An operating system for educational purpose written in the Turing language	20	57
compiler	A small compiler developed at the University of Toronto	13	32
nos	A file system	16	52
boxer	Graph drawing tool	18	29
spdb	A tool to analyze several proteins at the same time	21	33
ispell	Spelling and typographical error correction software	24	103
ciald	Program dependency analysis tool	26	64
Rcs	System used to manages multiple revisions of files	29	163
Star	A program understanding tool	36	89
Bison	Parser generator	37	179
Cia	Program dependency graph generator for C programs	38	87

جدول (4): ویژگی‌های پوشه‌های انتخاب‌شده از موزیلا فایرفاکس [1]

Folder Name	# Files	# Links	#Modules	Folder Functionality
ACCESSIBLE	179	293	8	Enabling as many people as possible to use Web sites, even when those people's abilities are limited in some way. Files for accessibility (i.e., MSAA (Microsoft Active Accessibility), ATK (Accessibility Toolkit, used by GTK+ 2) support files).
BROWSER	45	45	4	Contains the front end code (in XUL, Javascript, XBL, and C++) for the Firefox browser Contains the front end code for the DevTools (Scratchpad, Style Editor, etc.) Contains images and CSS files to skin the browser for each OS (Linux, Mac and Windows)
BUILD	21	4	2	Miscellaneous files used by the build process.
CONTENT	881	2948	13	The data structures that represent the structure of Web pages (HTML, SVG, XML documents, elements, text nodes, etc.) containing the implementation of many DOM interfaces and also implement some behaviors associated with those objects, such as link handling, form control behavior, and form submission. This directory also contains the code for XUL, XBL, XTF as well as the code implementing XSLT and event handling.
DB	97	494	4	Container for database-accessing modules.
DOM	163	324	5	IDL definitions of the interfaces defined by the DOM specifications The parts of the connection between JavaScript and the implementations of DOM objects Implementations of a few of the core "DOM Level 0" objects, such as window, window.navigator, window.location, etc.
EXTENSIONS	179	206	13	Contains several extensions to mozilla, which can be enabled at compile-time Implementation of the negotiate auth method for HTTP and other protocols. Has code for SSPI, GSSAPI, etc. Content- and locale-pack switching user interface Permissions backend for cookies, images, etc., as well as the user interface to these permissions and other cookie features. Support for the datetime protocol. Support for the finger protocol. A two-way bridge between the CLR/.NET/Mono/C#/etc. world and XPCOM Implementation of W3C's Platform for Privacy Preferences standard. Support for implementing XPCOM components in python. Support for accessing SQL databases from XUL applications Support for Webservices.
GFX	342	644	7	Contains interfaces that abstract the capabilities of platform specific graphics toolkits, along with implementations on various platforms These interfaces provide methods for things like drawing images, text, and basic shapes It also contains basic data structures such as points and rectangles used here and in other parts of Mozilla.
INTL	573	957	7	Internationalization and localization support, Code for "sniffing" the character encoding of Web pages Code for dealing with Complex Text Layout, related to shaping of south Asian languages Code related to determination of locale information from the operating environment Code that converts (both ways: encoders and decoders) between UTF-16 and many other character encodings Code related to implementation of various algorithms for Unicode text, such as case conversion.
IPC	391	59	4	Container for implementations of IPC (Inter-Process Communication).

1.2.6. سوال پژوهش شماره 1

برای پاسخ به سوال شماره 1، ما الگوریتم پیشنهادی را با برخی از روش‌های مبتنی بر جستجو مانند Bunch [19]، DAGC [20]، HC+Bunch [33]، NAHC [24]، SAHC [24] و EDA [22] روی سامانه نرم‌افزاری mtunis مقایسه کرده‌ایم. جدول (5) نتایج مقایسه را شرح می‌دهد. بهترین نتایج در این جدول به‌صورت پررنگ مشخص شده‌اند. نتایج نشان می‌دهند که الگوریتم پیشنهادی قادر به تولید پیمانه‌بندی نزدیک به تجزیه فرد خبره بوده است.

جدول (5): مقایسه الگوریتم پیشنهادی (AbHyh-SSM) با برخی از الگوریتم‌های مبتنی بر جستجو از نظر معیارهای خارجی

Algorhims	MoJo	Edge MoJo	F_m
Bunch [13]	**5.00**	**7.47**	**0.57**
DAGC [14]	7.00	10.33	0.48
HC+Bunch [31]	9.00	11.14	0.25
NAHC [27]	**5.00**	13.14	0.53
SAHC [27]	**5.00**	10.81	0.55
EDA [16]	**5.00**	**7.47**	**0.57**
AbHyh-SSM	**5.00**	**7.47**	**0.5**

2.2.6. سوال پژوهش شماره 2

برای پاسخ به سوال شماره 2، ما الگوریتم پیشنهادی را با برخی از الگوریتم‌های سلسله‌مراتبی و غیرسلسله‌مراتبی موجود مقایسه می‌کنیم. جدول (6) نتایج مقایسه الگوریتم پیشنهادی با برخی از برجسته‌ترین الگوریتم‌های مبتنی بر جستجو را روی ده پوشه از موزیلا فایرفاکس نشان می‌دهد. بهترین نتایج در این جدول به صورت پررنگ مشخص می‌شوند. ستون آخر این جدول، رتبه الگوریتم پیشنهادی را در بین الگوریتم‌های مقایسه شده، از نظر کیفیت راه‌حل نشان می‌دهد. همان‌طورکه مشاهده می‌کنید، الگوریتم پیشنهادی در هشت مورد از ده مورد دارای رتبه یک است؛ یعنی پیمانه‌بندی‌هایی با کیفیت بالاتر یا مساوی با بهترین پیمانه‌بندی یافت شده از بقیه الگوریتم‌ها تولید می‌کند. در دو مورد دیگر، یعنی پوشه‌های Accessible و Browser، الگوریتم پیشنهادی رتبه دو را در بین الگوریتم‌های مقایسه شده دارد. در نهایت، می‌توان گفت میانگین بهبود عددی الگوریتم پیشنهادی روی این ده پوشه نسبت به بهترین روش مقایسه شده 77/607 درصد است.

جدول (7) نتایج مقایسه الگوریتم پیشنهادی با روش‌های سلسله‌مراتبی، سه الگوریتم k-means، ACDC و FCA و سه الگوریتم مبتنی بر جستجوی Bunch-GA، DAGC و SCSO را روی ده سامانه نرم‌افزاری نشان می‌دهد. بهترین نتایج در این جدول به‌صورت پررنگ مشخص می‌شوند. واضح است که الگوریتم پیشنهادی در همه موارد پیمانه‌بندی‌هایی با کیفیت بالاتر از روش‌های سلسله‌مراتبی و الگوریتم‌های k-means، ACDC و FCA تولید می‌کند. از طرف دیگر، کیفیت پیمانه‌بندی‌های به دست آمده توسط AbHyh-SSM بهتر یا مساوی کیفیت پیمانه‌بندی‌های تولید شده توسط Bunch و DAGC هستند. در ستون مربوط به الگوریتم SCSO، از آنجایی که مقدار TurboMQ برای پنج سامانه نرم‌افزاری در مقاله [27] گزارش نشده است، با علامت «-» مشخص شده‌اند. در پنج مورد دیگر، الگوریتم پیشنهادی در چهار مورد بهتر از SCSO و در یک مورد مساوی با آن عمل می‌کند. در نهایت، می‌توان گفت میانگین بهبود عددی الگوریتم پیشنهادی روی این ده سامانه نسبت به بهترین روش مقایسه شده برای هر سامانه 0/787 درصد است.

3.2.6. سوال پژوهش شماره 3

برای پاسخ به سوال شماره 3، الگوریتم پیشنهادی 10 بار برای هر پوشه از موزیلا فایرفاکس اجرا می‌شود و پایداری نتایج توسط روش آماری t-test تحلیل می‌شود. انتظار می‌رود نتایج به دست آمده توسط الگوریتم برای 10 اجرای مستقل به اندازه کافی به یکدیگر نزدیک باشند.

برای اعمال t-test، نتایج در دو گروه هم‌اندازه به نام‌های G1 و G2 گروه‌بندی شده و سپس برخی آمارهای توصیفی و استنباطی از آنها استخراج می‌شود. جدول (8) نتایج این آزمون را نشان می‌دهد. سه ستون اول به ترتیب میانگین، انحراف معیار و خطای استاندارد بین میانگین دو گروه را به عنوان

آمارهای توصیفی نشان می‌دهند. دو ستون آخر جدول خروجی آمارهای استنباطی را نشان می‌دهند. آزمون لون یک آمار استنباطی برای ارزیابی برابری واریانس‌ها برای یک متغیر محاسبه شده برای دو گروه است. اگر p-value (ستون .Sig در جدول) بیشتر از سطح معنی‌داری (0/05 در آزمون‌های ما) باشد، نمی‌توان فرضیه صفر واریانس‌های برابر را رد کرد. ستون‌های آخر جدول (8) به نتایج آزمون t دو نمونه‌ای مستقل با اندازه‌های نمونه یکسان و واریانس‌های برابر (بر اساس نتایج آزمون لون) بر روی دو گروه از نتایج الگوریتم پیشنهادی که به طور تصادفی از هم جدا شده‌اند، اشاره دارد. همه مقادیر .Sig (مقادیر پررنگ) بزرگتر از 0/05 هستند که نشان می‌دهد نمی‌توانیم فرضیه صفر میانگین‌های برابر را رد کنیم. از این رو، نتایج آزمون‌های مختلف به محدوده قابل قبولی همگرا می‌شوند و در نتیجه الگوریتم پیشنهادی پایدار است.

جدول (6): مقایسه الگوریتم پیشنهادی با برخی از برجسته‌ترین الگوریتم‌های مبتنی بر جستجو از نظر TurboMQ

Folder name	Bunch-GA	DAGC	Bunch-NAHC	SHC	GA-SMCP	EoD	AbHyh-SSM	Rank
Accessible	6.260	0.930	4.800	10.010	14.000	**29.210**	17.109	2
Browser	3.720	0.920	5.850	**9.000**	6.500	**9.000**	8.330	2
Build	**3.000**	0.500	1.000	0.920	1.230	2.900	**3.000**	1
Content	6.760	0.190	5.410	16.970	12.010	10.110	**33.060**	1
Db	2.340	0.860	2.600	2.340	1.900	2.510	**5.085**	1
Dom	6.160	0.920	4.300	12.870	5.110	8.000	**15.923**	1
Extensions	11.800	0.910	6.660	8.900	10.990	13.000	**27.412**	1
Gfx	6.500	0.820	4.320	3.010	6.860	12.220	**17.854**	1
Intl	5.460	0.900	2.540	7.900	4.110	12.900	**62.825**	1
Ipc	5.640	0.840	3.920	4.500	5.640	10.900	**14.980**	1

جدول (7): مقایسه الگوریتم پیشنهادی با برخی از الگوریتم‌های سلسله‌مراتبی و مبتنی بر جستجو، K-means، ACDC و FCA از نظر TurboMQ

Software systems	WCA-UE	Average Linkage	Complete Linkage	Single Linkage	FCA	ACDC	k-means	Bunch	DAGC	SCSO	AbHyh-SSM
Compiler	0.836	0.527	0.527	0.933	1.220	1.000	0.850	**1.506**	**1.506**	-	**1.506**
Boxer	1.343	0.964	0.983	0.964	3.020	2.820	0.790	**3.101**	**3.101**	-	**3.101**
Ispell	1.489	1.739	1.639	0.995	1.970	1.750	1.200	2.177	1.997	2.189	**2.190**
Bison	0.994	0.994	0.994	0.994	2.250	1.000	1.000	2.606	1.763	2.455	**2.684**
Cia	0.997	0.997	0.997	0.997	2.049	1.860	1.780	2.706	1.833	2.500	**2.787**
Ciald	0.984	0.487	1.093	0.984	1.720	1.700	0.780	**2.851**	2.463	-	**2.851**
Nos	0.969	0.990	0.990	0.990	1.080	1.000	0.980	**1.636**	1.606	-	**1.636**
Rcs	0.977	0.990	1.018	1.018	1.810	1.000	1.260	2.175	1.894	2.171	**2.202**
Spdb	0.933	0.933	0.933	0.933	5.000	5.000	1.150	**5.741**	5.314	**5.741**	**5.741**
Star	1.388	0.989	0.805	0.989	3.048	2.090	0.810	3.809	2.831	-	**3.832**

جدول (8): نتایج آزمون T-TEST

Case Study	Descriptive Statistics						Inferential Statistics			
	Mean		Standard Deviation		Standard Error between Mean		levene's test		t-test	
	G1	G2	G1	G2	G1	G2	F	Sig.	T	Sig.
Accessible	19.895	20.150	0.697	0.569	0.311	0.254	1.524	0.252	-0.632	**0.545**
Browser	6.811	6.715	0.376	0.395	0.168	0.176	0.277	0.613	0.395	**0.703**
Build	2.600	2.800	0.548	0.447	0.245	0.200	1.524	0.252	-0.632	**0.545**
Content	30.743	30.634	0.633	0.610	0.283	0.273	0.040	0.847	0.275	**0.790**
Db	3.892	3.847	0.321	0.337	0.144	0.151	0.180	0.683	0.212	**0.837**
Dom	14.111	13.856	0.598	0.447	0.267	0.200	0.520	0.491	0.764	**0.467**
Extensions	26.688	26.511	0.929	0.868	0.415	0.388	0.099	0.761	0.310	**0.764**
Gfx	17.538	17.610	0.201	0.144	0.090	0.064	2.442	0.157	-0.657	**0.530**
Intl	65.062	64.772	2.597	2.414	1.161	1.080	0.013	0.911	0.183	**0.859**
Ipc	17.826	17.836	0.504	0.510	0.226	0.228	0.000	0.984	-0.031	**0.976**

4.2.6. سوال پژوهش شماره 4

برای پاسخ به سوال شماره 4، ما زمان اجرای روش پیشنهادی را با برخی از روش‌های مبتنی بر جستجوی سراسری مقایسه کردیم. برای الگوریتم‌های مبتنی بر ژنتیک مقایسه شده، ما از تنظیمات متغیر استفاده شده در مراجع [1] و [28] استفاده کردیم و برای روش EoD [25]، از همان متغیرهایی استفاده کردیم که نویسندگان این الگوریتم استفاده کرده‌اند. فرض کنید N تعداد موجودیت‌ها را نشان می‌دهد، تنظیمات متغیر الگوریتم‌های تکاملی مقایسه شده در جدول (9) ذکر شده‌اند.

جدول (9): تنظیمات متغیر برای الگوریتم‌های تکاملی

Parameter	Value
Selection	Roulette Wheel Selection
Mutation operation	Randomly changed a gene
Crossover operation	One-point
Termination condition	There has been no improvement in the population for 50 iterations
Population size	10N
Mutation Rate	$0.004\log_2(N)$
Crossover Rate	0.8
Maximum number of generations	200N

جدول (10) نتایج مقایسه زمان اجرا را روی ده پوشه از موزیلا فایرفاکس نشان می‌دهد. لازم به ذکر است که زمان اجرای الگوریتم‌های مبتنی بر جستجوی مقایسه شده از مقاله [28] به دست آمده است که روی کامپیوتری با قدرت و حافظه بیشتر از کامپیوتر ما اجرا شده‌اند. بهترین نتایج در جدول (10) به صورت پررنگ مشخص شده‌اند. نتایج این جدول اثبات می‌کند که در بیشتر موارد (هشت مورد از ده مورد) زمان اجرای روش پیشنهادی به طور قابل توجهی کمتر از الگوریتم‌های مقایسه شده است. در نهایت، می‌توان گفت میانگین بهبود عددی زمان اجرای الگوریتم پیشنهادی روی این ده پوشه نسبت به بهترین روش مقایس شده 59/448 درصد است.

3.6. پژوهش‌های آینده

پژوهش آینده باید به توسعه موارد زیر اختصاص داده شود:

1. اعمال الگوریتم پیشنهادی روی مساله‌های بهینه‌سازی ترکیباتی دیگر مانند رنگ‌آمیزی گراف و غیره.
2. استفاده از برنامه‌های کاربردی دیگر برای ارزیابی روش پیشنهادی.
3. استفاده از دیگر معیارهای ارزیابی الگوریتم.
4. استفاده از اجتماع‌های بیشتر در ابراکتشافی پیشنهادی مانند اجتماع سراسری و غیره.
5. تعریف همکاری بین عامل‌های موازی در هر اجتماع.

جدول (10): مقایسه زمان اجرای الگوریتم پیشنهادی با برخی از روش‌های مبتنی بر جستجوی سراسری (ثانیه)

Folder name	Bunch-GA	DAGC	ECA	MCA	GA-SMCP	EoD	AbHyh-SSM
Accessible	4535	7471	4521	4437	5921	3990	**498.836**
Browser	708	609.5	733.5	796.5	901	541	**429.636**
Build	512	**383.805**	421.5	425	540	431	428.341
Content	950337.5	4794247.5	943040	943021	5224315	890021	**1176.531**
Db	494.5	1834.495	1363.5	1470.5	2301	**481**	686.402
Dom	3110	6139	3082.5	3153	6341	2208	**471.660**
Extensions	6264	7653	6461	6730	6421	6259	**1351.885**
Gfx	15563	28173	14781	14966	21540	13238	**713.791**
Intl	222888	1333765.5	223645	222884	1034921	22198	**2234.378**
Ipc	62770.5	424153.5	62642.5	62683.5	99101	61211	**464.337**

7. نتیجه‌گیری

پیمانه‌بندی یک روش کلیدی برای درک و نگه‌داری برنامه است. در این مقاله، ما یک الگوریتم ابراکتشافی مبتنی بر عامل به نام AbHyh-SSM برای پیمانه‌بندی نرم‌افزار پیشنهاد کردیم. ما نشان دادیم که با استفاده از الگوریتم ابراکتشافی و روش یادگیری در بستر سامانه‌های چندعاملی می‌توان پیمانه‌بندی‌های خوبی ایجاد کرد. همچنین، در این مطالعه، یک طبقه‌بندی وضعیت مبتنی بر برازندگی که نیاز به اطلاعات وابسته به دامنه کمتری دارد، استفاده شده است. در کار ما، عمل به صورت یک اجتماع که شامل چندین عامل موازی است، تعریف شده است. نتایج تجربی الگوریتم پیشنهادی و مقایسه آن با نتایج الگوریتم‌های پیمانه‌بندی شناخته شده نشان داد که روش AbHyh-SSM به طور همزمان میزان کیفیت پیمانه‌بندی‌ها و زمان اجرا را بهبود می‌دهد.

تعارض منافع: نویسندگان اعلام می‌کنند که هیچ تعارض منافعی ندارند.

مراجع

[1] B. Pourasghar, H. Izadkhah, A. Isazadeh, and S. Lotfi, "A graph-based clustering algorithm for software systems modularization," Inf. Softw. Technol., vol. 133, p. 106469, 2021, doi: 10.1016/j.infsof.2020.106469.

[2] H. Izadkhah and M. Tajgardan, "Information theoretic objective function for genetic software clustering," Multidisciplinary Digit. Pub. Instit. Proc., vol. 46, no. 1, p. 18, 2019, doi: 10.3390/ecea-5-06681.

[3] M. Tajgardan and H. Izadkhah, "Critical Review of the Bunch: A Well-Known Tool for the Recovery and Maintenance of Software System Structures," Critical Rev., vol. 6, no. 3, pp. 363-367, 2017, doi: 10.17148/IJARCCE.2017.6383.

[4] S. Gholamshahi and S.M.H. Hasheminejad, "A method for identifying software components based on Non-dominated Sorting Genetic Algorithm," Soft Comput. J., vol. 7, no. 2, pp. 47-64, 2019, dor: 20.1001.1.23223707.1397.7.2.4.5 [In Persian].

[5] M. Nabiloo and N. Daneshpour, "A clustering algorithm for categorical data with combining measures," Soft Comput. J., vol. 5, no. 1, pp. 14-25, 2017 [In Persain].

[6] A. Isazadeh, H. Izadkhah, and I. Elgedawy, Source Code Modularization Theory and Techniques, Springer International Publishing, 2017.

[7] S. Asta, Machine learning for improving heuristic optimisation, Doctoral dissertation, University of Nottingham, 2015.

[8] M.A.L. Silva, S.R. de Souza, M.J.F. Souza, and M.F. de Franca Filho, "Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of

frameworks and a comparative analysis," Appl. Soft Comput., vol. 71, pp. 433-459, 2018, doi: 10.1016/j.asoc.2018.06.050.

[9] R. Malek, "An agent-based hyper-heuristic approach to combinatorial optimization problems," in IEEE Int. Conf. Intell. Comput. Intell. Syst., 2010, pp. 428-434.

[10] A. Hassan and N. Pillay, "Hybrid metaheuristics: An automated approach," Expert Syst. Appl., vol. 130, pp. 132-144, 2019, doi: 10.1016/j.eswa.2019.04.027.

[11] D. Meignan, A. Koukam, and J.-C. Creput, "Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism," J. Heuristics, vol. 16, no. 6, pp. 859-879, 2010, doi: 10.1007/s10732-009-9121-7.

[12] J. Huang, J. Liu, and X. Yao, "A multi-agent evolutionary algorithm for software module clustering problems," Soft Comput., vol. 21, no. 12, pp. 3415-3428, 2017, doi: 10.1007/s00500-015-2018-5.

[13] A.C. Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," J. Syst. Softw., vol. 117, pp. 384-401, 2016, doi: 10.1016/j.jss.2016.04.007.

[14] M. Tajgardan, H. Izadkhah, and S. Lotfi, "A Reinforcement Learning-based Iterated Local Search for Software Modularization," in 8th Iranian Conf. Signal Process. Intell. Syst. (ICSPIS), 2022, pp. 1-6, doi: 10.1109/ICSPIS56952.2022.10043949.

[15] M. Tajgardan, H. Izadkhah, and S. Lotfi, "An Iterated Local Search Strengthened by a Q-learning-based Hyper-heuristic for Software Modularization," Soft Comput. J., 2023, 10.22052/SCJ.2023.252654.1135.

[16] M. Saeed, O. Maqbool, H.A. Babri, S.Z. Hassan, and S.M. Sarwar, "Software clustering techniques and the use of combined algorithm," in 7th European Conf. Soft. Maintenan. Reeng., 2003, pp. 301-306, doi: 10.1109/CSMR.2003.1192438.

[17] O. Maqbool and H. Babri, "Hierarchical clustering for software architecture recovery," IEEE Trans. Softw. Eng. vol. 33, no. 11, pp. 759-780, 2007, doi: 10.1109/TSE.2007.70732.

[18] R. Naseem, O. Maqbool, and S. Muhammad, "Cooperative clustering for software modularization," J. Syst. Softw., vol. 86, no. 8, pp. 2045-2062, 2013, doi: 10.1016/j.jss.2013.03.080.

[19] B.S. Mitchell, A heuristic search approach to solving the software clustering problem, Ph.D. Thesis, Drexel University, 2002.

[20] S. Parsa and O. Bushehrian, "A New Encoding Scheme and a Framework to Investigate Genetic Clustering Algorithms," J. Res. Pract. Inf. Technol., vol. 37, no. 1, pp. 127-143, 2005.

[21] M. Harman and X. Yao, "Software module clustering as a multi-objective search problem," IEEE Trans. Softw. Eng., vol. 37, no. 2, pp. 264-282, 2010, doi: 10.1109/TSE.2010.26.

[22] M. Tajgardan and H. Izadkhah, "Software Systems Clustering Using Estimation of Distribution Approach," J. Appl. Comput. Sci. Methods., vol. 8, no. 2, pp. 99-113, 2016, dx.doi: 10.1515/jacsm-2016-0007.

[23] J. Huang and J. Liu, "A similarity-based modularization quality measure for software module clustering problems," Inf. Sci., vol. 342, pp. 96-110, 2016, doi: 10.1016/j.ins.2016.01.030.

[24] B.S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool," IEEE Trans. Softw. Eng., vol. 32, no. 3, pp. 193-208, 2006, doi: 10.1109/TSE.2006.31.

[25] N. Sadat Jalali, H. Izadkhah, and S. Lotfi, "Multi-objective search-based software modularization: structural and non-structural features," Soft Comput., vol. 23, no. 21, pp. 11141-11165, 2019, doi: 10.1007/s00500-018-3666-z.

[26] M. Kargar, A. Isazadeh, and H. Izadkhah, "Semantic-based software clustering using hill climbing," in Int. Symp. Comput. Sci. Soft. Eng. Conf. (CSSE), 2017, pp. 55-60, doi: 10.1109/CSICSSE.2017.8320117.

[27] B. Arasteh, A. Seyyedabbasi, J. Rasheed, and A. Abu-Mahfouz, "Program Source-Code Re-Modularization Using a Discretized and Modified Sand Cat Swarm Optimization Algorithm," Symmetry, vol. 15, no. 2, p. 401, 2023, doi: 10.3390/sym15020401.

[28] N. Teymourian, H. Izadkhah, and A. Isazadeh, "A fast clustering algorithm for modularization of large-scale software systems," IEEE Trans. Softw. Eng., vol. 48, no. 4, pp. 1451-1462, 2020, doi: 10.1109/TSE.2020.3022212.

[29] V. Tzerpos and R.C. Holt, "Accd: an algorithm for comprehension-driven clustering," in Proc. 7th Work. Conf. Rev. Eng., 2000, pp. 258-267, doi: 10.1109/WCRE.2000.891477.

[30] B. Zarei, M.R. Meybodi, and B. Masoumi, "Chaotic memetic algorithm and its application for detecting community structure in complex networks," Chaos: Interdisciplinary J. Nonlinear Sci., vol. 30, no. 1, p. 13125, 2020, doi: 10.1063/1.5120094.

[31] S.S. Choong, L.P. Wong, and C.P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," Inf. Sci., vol. 436, pp. 89-107, 2018, doi: 10.1016/j.ins.2018.01.005.

[32] H. Izadkhah, I. Elgedawy, and A. Isazadeh, "E-CDGM: An Evolutionary Call-Dependency Graph Modularization Approach for Software Systems," Cybern. Inf. Technol., vol. 16, no. 3, 2016, dx.doi: 10.1515/cait-2016-0035.

[33] K.A. Mahdavi, Clustering genetic algorithm for software modularisation with a multiple hill climbing approach, Ph.D. Thesis, Brunel University, 2005.