یک ابراکتشافی مبتنی بر عامل برای پیمانه‌بندی سامانه‌های نرم‌افزاری

محجوبه تاج‌گردان [1]، دانشجوی دکتری، حبیب ایزدخواه[2*]، دانشیار، شهریار لطفی [3]، دانشیار

[1] دانشکده ریاضی، آمار و علوم کامپیوتر- دانشگاه تبریز- تبریز- ایران- m.tajgardan@tabrizu.ac.ir

[2] دانشکده ریاضی، آمار و علوم کامپیوتر- دانشگاه تبریز- تبریز- ایران- izadkhah@tabrizu.ac.ir

[3] دانشکده ریاضی، آمار و علوم کامپیوتر- دانشگاه تبریز- تبریز- ایران- shahriar_lotfi@tabrizu.ac.ir

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 مسئله چندجمله‌ای غیرقطعی‌سخت است، معمولا 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 به‌طور خودکار انتخاب و عامل‌های آن به‌صورت 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 می‌شوند. نتایج آزمایش‌ها نشان می‌دهند که ابراکتشافی پیشنهادی در بیشتر موارد پیمانه‌بندی‌هایی با کیفیت بالاتر را در زمان کمتری نسبت به الگوریتم‌های مقایسه‌شده تولید می‌کند. میانگین بهبود عددی الگوریتم پیشنهادی از نظر کیفیت پیمانه‌بندی و زمان اجرا روی ده پوشه از موزیلا فایرفاکس به ترتیب ۷۷,۶۰۷ و ۵۹,۴۴۸ درصد می‌باشد.

واژه‌های کلیدی: پیمانه‌بندی نرم‌افزار، بازیابی معماری، ابراکتشافی، سامانه‌های چندعاملی، یادگیری تقویتی

* حبیب ایزدخواه، **izadkhah@tabrizu.ac.ir**

An Agent-based Hyper-heuristic for Software Systems Modularization

Mahjoubeh Tajgardan [1], Ph.D. Candidate, Habib Izadkhah [2*], Associate Professor, Shahriar Lotfi [3], Associate Professor

1 Faculty of Mathematics, Statistics, and Computer Science, University of Tabriz, Tabriz, Iran, m.tajgardan@tabrizu.ac.ir

2 Faculty of Mathematics, Statistics, and Computer Science, University of Tabriz, Tabriz, Iran, izadkhah@tabrizu.ac.ir

3 Faculty of Mathematics, Statistics, and Computer Science, University of Tabriz, Tabriz, Iran, shahriar_lotfi@tabrizu.ac.ir

Abstract: **Modularization algorithms are used to recover the software architecture. These algorithms divide the source code of the software system into smaller and more understandable modules. Since software modularization is an NP-hard problem, search-based methods are usually used to solve it. In recent years, the use of hyper-heuristics with intelligent search approaches has increased to achieve a higher level of generality. In this paper, a general agent-based hyper-heuristic, using the concept of multi-agent systems, is presented for software modularization. In the proposed algorithm, agents with diversification and intensification search perspectives are used, and agents with the same perspective are placed in a coalition. In each step of the search, the most appropriate coalition is automatically selected using reinforcement learning, and its agents are executed in parallel. Also, in the design of some agents, to maintain diversity, the concept of chaos theory is used. To demonstrate the applicability of the proposed algorithm, eleven real-world software systems with small and medium sizes, along with ten folders of Mozilla Firefox with different functionalities and sizes are selected. The results of the experiments show that the proposed hyper-heuristic, in most cases, produces higher-quality modularizations in less time than the compared algorithms. The average numerical improvement of the proposed algorithm in terms of modularization quality and execution time on ten folders of Mozilla Firefox is 77.607 and 59.448%, respectively.**

Keywords: ***Software modularization; Architecture recovery; Hyper-heuristic; Multi-agent systems; Reinforcement learning.***

*** Habib Izadkhah, izadkhah@tabrizu.ac.ir**

۱. مقدمه

فهم برنامه نقش مهمی را در فرایند توسعه و نگه‌داری نرم‌افزار بازی می‌کند. یک گام مهم برای فهم برنامه در فرایند نگه‌داری، بازیابی معماری نرم‌افزار است. توسعه‌دهندگان نرم‌افزار با کمک روش‌های پیمانه‌بندی می‌توانند ساختار نرم‌افزاری را استخراج کنند که طراحان آن در دسترس نیستند [۱]. در بازیابی معماری نرم‌افزار با استفاده از روش‌های پیمانه‌بندی، کد منبع یک سامانه نرم‌افزاری به بخش‌های معنادار و قابل فهم افراز می‌شود [۲]. در فرایند پیمانه‌بندی نرم‌افزار، موجودیت‌های کد منبع مانند کلاس‌ها، توابع و غیره در مجموعه‌هایی به نام پیمانه‌ها به‌نحوی 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 [۳]. انسجام بالا و اتصال پایین ساختاری را برای توسعه‌دهنده نرم‌افزار فراهم می‌کنند که نگه‌داری آن آسان‌تر است [۴].

از آنجایی که هیچ تعریف جامع و تثبیت‌شده‌ای از ویژگی‌های پیمانه‌بندی بهینه وجود ندارد، امروزه برای محاسبه کیفیت پیمانه‌بندی‌های تولیدشده توسط الگوریتم‌ها از دو نوع ارزیابی داخلی و خارجی استفاده می‌شود. در ارزیابی داخلی، چندین معیار برای ارزیابی میزان جداسازی صحیح پیمانه‌ها استفاده و به‌صورت 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 [۱]. یک الگوریتم پیمانه‌بندی که پیمانه‌بندی به‌دست آمده از آن نزدیک به پیمانه‌بندی فراهم‌شده 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 سلسله‌مراتبی یا مبتنی بر جستجو هستند. روش‌های سلسله‌مراتبی ادغامی [۵] حریصانه هستند و در هر مرحله از جستجو شبیه‌ترین موجودیت‌ها را با هم ادغام می‌کنند. این الگوریتم‌ها زمان جستجوی معقولی دارند. تصمیم‌گیری اختیاری [۶] و محلی بودن معیارهای تشابه موجود برای این روش‌ها [۱]، از جمله معایب آنها هستند. با توجه به پیچیدگی مسئله پیمانه‌بندی، معمولا 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 مسئله جستجو در نظر گرفته می‌شود و برای هدایت فرایند پیمانه‌بندی، توابع تک‌هدفه یا چندهدفه استفاده می‌شوند. BasicMQ [۶] و TurboMQ [۶] دو تابع هدف خوب‌شناخته‌شده و پرکاربرد هستند. TurboMQ بر خلاف BasicMQ برای گراف‌های وزن‌دار نیز قابل استفاده است و همچنین پیچیدگی زمانی محاسبه آن کمتر است [۶]. در TurboMQ، ابتدا فاکتور پیمانه توسط فرمول (۱) برای همه پیمانه‌ها محاسبه می‌شود. در این فرمول، μ_i اتصالات داخلی پیمانه i و $\varepsilon_{i,j}$ اتصالات خارجی بین پیمانه i و پیمانه j است. مقدار TurboMQ توسط فرمول (۲) محاسبه می‌شود که در آن، k تعداد پیمانه‌ها است.

$$MF_i=\frac{2*\mu_i}{2*\mu_i+\varepsilon_{ij}} \qquad (۱)$$

$$TurboMQ=\sum_{i=1}^{k} MF_i \qquad (۲)$$

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

۱. با توجه به چندجمله‌ای غیرقطعی‌سخت بودن مسئله پیمانه‌بندی نرم‌افزار، معمولا الگوریتم‌های اکتشافی و فرااکتشافی برای حل آن استفاده می‌شوند. با این حال، این الگوریتم‌ها اغلب عمومی نیستند؛ یعنی در دامنه‌های مسئله متفاوت و یا حتی نمونه‌های متفاوت از دامنه مسئله یکسان کارایی متفاوتی دارند [۷].

۲. در سال‌های اخیر، پژوهشگران از فرااکتشافی‌های ترکیبی برای حل مسئله‌های بهینه‌سازی ترکیباتی مانند مسئله پیمانه‌بندی نرم‌افزار استفاده کرده‌اند. از جمله مزایای این ترکیب، به‌دست آوردن بهترین کیفیت راه‌حل در زمان کوتاه‌تر و افزایش توانایی مقابله با مسئله‌های پیچیده‌تر است [۸]. 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 مسئله به‌صورت دستی انتخاب می‌شوند [۹] که زمان‌بر و پرهزینه است. از طرف دیگر، طراحی برخی از این روش‌های ترکیبی قبل از فرایند بهینه‌سازی واقعی انجام می‌شود [۱۰] (طراحی آفلاین) و کارایی این روش‌ها روی نمونه‌های مسئله مختلف، به عمومیت نمونه‌های آموزشی وابستگی دارد. زمانی‌که نحوه اجرای فرااکتشافی‌های ترکیب‌شده به‌صورت ترتیبی باشد، با مسئله پیدا کردن بهترین دنباله فراخوانی الگوریتم‌ها روبه‌رو هستند که خود یک مسئله بهینه‌سازی است [۱۱]. در برخی از روش‌های ترکیبی که چندین عامل به‌طور موازی فضای جستجو را کاوش می‌کنند، عامل‌ها و همچنن عملگرهای تعریف‌شده برای آنها یکسان هستند [۱۱]، این در حالی است که استفاده از عامل‌های مختلف با عملگرهای متفاوت، به دلیل استفاده آنها از دیدگاه‌های جستجوی مختلف، می‌تواند به فرایند جستجو کمک کند.

۳. 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 مسئله، 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 [۸]. 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 مسئله 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 آزمایش‌وخطا و پاداش تاخیری پیروی نمی‌کنند. بنابراین، طراحی ابراکتشافی‌هایی که از معیارهای یادگیری تقویتی پیروی می‌کنند، ضروری است.
- مجموعه وضعیت‌های تعریف‌شده در برخی از ابراکتشافی‌ها ممکن است همه موقعیت‌هایی را که در طی جستجو اتفاق می‌افتند، شامل نشود. از طرف دیگر، در تعریف وضعیت‌ها از پارامترهای مختلفی

استفاده می‌شود که تنظیم درست آنها بر عهده کاربر است. تنظیم این پارامترها با استفاده از آزمایش‌وخطا زمان‌بر و پرهزینه است. بنابراین، تعریف مجموعه وضعیت‌ها به‌نحوی که شامل همه موقعیت‌ها و همچنین مستقل از مسئله 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 معرفی‌شده در بخش ۲ (کارهای مرتبط) تنها مراجع [۱۲] و [۱۳] به ترتیب از مفهوم سامانه‌های چندعاملی و ابراکتشافی برای پیمانه‌بندی نرم‌افزار استفاده کرده‌اند.

ما نیز در تحقیق قبلی خود در [۱۴] و نسخه توسعه‌یافته آن در [۱۵] 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 به‌ترتیب 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 به‌طور موازی به جستجو می‌پردازند. در روش پیشنهادی در مقاله حاضرممکن 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 به‌طور خودکار برای اجرا انتخاب می‌شود. لازم به ذکر است که تفاوت اصلی الگوریتم پیشنهادی در این مقاله با دو مقاله قبلی ما [۱۴], [۱۵] 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. به‌طور کلی می‌توان گفت اشتراک اصلی این مقاله با دو مقاله قبلی ما [۱۴], [۱۵] در این است که 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 [۱۴] و [۱۵]، 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 بزرگ‌مقیاس موزیلافایرفاکس با اندازه‌ها و عملکردهای مختلف نشان می‌دهد که الگوریتم پیشنهادی در بیشتر موارد پیمانه‌بندی‌هایی با کیفیت بالاتر از پیمانه‌بندی‌های به‌دست آمده توسط الگوریتم‌های مقایسه‌شده تولید می‌کند، در حالیکه 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 به‌طور 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 به‌دست آورد. در نهایت، سهم‌های علمی این مقاله در زیر خلاصه شده‌اند:

۱. استفاده از ابراکتشافی‌ها جهت بهبود عمومیت الگوریتم پیشنهادی.

۲. جستجوی هوشمندانه فضای راه‌حل‌ها با استفاده از مفهوم سامانه‌های چندعاملی.

۳. طراحی اکتشافی‌های سطح پایین به‌صورت اجتماع.

۴. استفاده از یادگیری تقویتی برای انتخاب خودکار اجتماع مناسب در هر وضعیت از جستجو.

۵. اجرای موازی عامل‌ها در اجتماع.

۶. 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.

۷. بهبود کیفیت راه‌حل‌های پیمانه‌بندی از نظر مقدار معیار TurboMQ.

۸. بهبود زمان اجرای روش پیشنهادی نسبت به الگوریتم‌های مبتنی بر جستجوی مقایسه‌شده

بخش‌های بعدی مقاله به شرح زیر سازمان‌دهی شده است. بخش ۲ برخی از کارهای مرتبط را توصیف می‌کند. بخش ۳ مفاهیم اولیه مورد نیاز را تعریف می‌کند. بخش ۴ مسئله پیمانه‌بندی سامانه‌های نرم‌افزاری را شرح می‌دهد. بخش ۵ الگوریتم پیشنهادی را توصیف می‌کند. بخش ۶ آزمایشات را توصیف می‌کند. بخش ۷ در مورد نتیجه پژوهش انجام‌شده بحث می‌کند.

۲. کارهای مرتبط

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

۱،۲. روش‌های سلسله‌مراتبی

الگوریتم‌های پیوند کامل (CL) [۶]، پیوند تکی (SL) [۶]، پیوند متوسط (AL) [۶] و پیوند متوسط وزنی (WAL) [۶] از جمله روش‌های سلسله‌مراتبی کلاسیک و الگوریتم‌های ترکیبی (CA) [۱۶]، ترکیبی وزنی (WCA) [۱۷] و پیمانه‌بندی تعاونی (CCT) [۱۸] از دیگر روش‌های سلسله‌مراتبی موجود هستند. جدول (۱) این روش‌ها را به همراه ویژگی‌های آنها شامل معیار شباهت استفاده‌شده و نوع آن خلاصه می‌کند.

جدول (۱): برخی از الگوریتم‌های پیمانه‌بندی سلسله‌مراتبی موجود

Method	Similarity metric	Type of metric
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

۲،۲. 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 (۲) این الگوریتم‌ها و ویژگی‌های آنها را خلاصه می‌کند.

میچل [۱۹] 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 [۳] از طرف دیگر، تعداد راه‌حل‌های تکراری تولیدشده توسط این الگوریتم بسیار زیاد است [۳].

پارسا و بوشهریان [۲۰] 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 به‌طور قابل توجهی کاهش می‌دهد.

پرادیت هارمن و یائو [۲۱] دو الگوریتم ژنتیک چندهدفه به نام‌های ECA و MCA پیشنهاد کردند که هر یک دارای پنج تابع هدف هستند.

تاج‌گردان و همکارانش [۲۲] 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 [۲۳] 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 [۱۲]، سه عملگر تکاملی با اهداف رقابت، همکاری و خودآموزی برای عامل‌ها (راه‌حل‌ها) ارائه کردند.

کوماری و سرینیواس [۱۳] یک ابراکتشافی مبتنی بر یادگیری تقویتی برای انتخاب عملگرهای ژنتیکی انتخاب، ترکیب و جهش مناسب در هر نسل از الگوریتم ژنتیک ارائه کردند.

جدول (۲): برخی از الگوریتم‌های پیمانه‌بندی مبتنی بر جستجوی موجود

Method	Type	SO / MO	LS / GS	S / non-S features	Learning-based	Main disadvantage
Bunch [19]	Genetic algorithm	SO	GS	S	No	Time and space limitations
NAHC [24]	Hill-climbing algorithm	SO	LS	S	No	Trapping in local optima
SAHC [24]	Hill-climbing algorithm	SO	LS	S	No	Trapping in local optima
DAGC [20]	Genetic algorithm	SO	GS	S	No	Time and space limitations
ECA [21]	Two archive Genetic algorithm	MO	GS	S	No	Time and space limitations
MCA [21]	Two archive Genetic algorithm	MO	GS	S	No	Time and space limitations
EDA [22]	Estimation of Distribution Algorithm	SO	GS	S	Yes	Time and space limitations
GA-SMCP [23]	Genetic algorithm	SO	GS	S	No	Time and space limitations
EoD [25]	Estimation of Distribution Algorithm	MO	GS	S, non-S	Yes	Time and space limitations
SHC [26]	Hill-climbing algorithm	SO	LS	non-S	No	Trapping in local optima
Modified SCSO	Sand Cat Swarm Optimization					Time and space

[27]	Algorithm	SO	GS	S	No	limitations
MHypEA [13]	Hyper-heuristic-based Genetic Algorithm	MO	GS	S	Yes	Time and space limitations

جلالی و همکارانش [۲۵] یک تابع برازندگی چندهدفه جدید به نام MOF ارائه کردند که ویژگی‌های ساختاری و غیرساختاری را با هم در نظر می‌گیرد.

کارگر و همکارانش [۲۶] یک الگوریتم پیشرو با نام SHIC ارائه کردند که مستقل از زبان برنامه‌نویسی است و از یک گراف وابستگی معنایی برای به‌دست آوردن معماری نرم‌افزار استفاده می‌کند.

آراسته و همکارانش [۲۷] یک الگوریتم بهینه‌سازی ازدحام گربه شنی گسسته را برای خوشه‌بندی موثر کد منبع نرم‌افزار ارائه کردند.

الگوریتم های خوشه‌بندی سریع (FCA) [۲۸]، ACDC [۲۹] و k-means [۶] نیز متعلق به دسته روش‌های پیمانه‌بندی غیرسلسله‌مراتبی هستند.

۳. تعریف مفاهیم اولیه

در این بخش، مفاهیم اولیه مورد نیاز تعریف می‌شوند.

۱،۳. یادگیری-Q

یادگیری-Q یکی از روش‌های یادگیری تقویتی است که 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 کنید $S = [s_1, s_2, s_3, \ldots, s_n]$ و $A = [a_1, a_2, a_3, \ldots, a_m]$ به ترتیب یک مجموعه از وضعیت‌های ممکن و یک مجموعه از عمل‌های قابل انتخاب را نشان می‌دهند. مقادیر-Q با استفاده از فرمول (۳) محاسبه می‌شود که در آن، r_{t+1} سیگنال تقویتی فوری است، $\alpha \in [0,1]$ نرخ یادگیری است، $\gamma \in [0,1]$ فاکتور تخفیف است و $Q(s_t, a_t)$ مقدار-Q در زمان t است.

$$Q^{new}(s_t,a_t) = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma * \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \quad (۳)$$

۲،۳. نقشه آشوب منطقی

در ریاضیات، نقشه‌های آشوبی برای تولید دنباله‌های آشوبی استفاده می‌شوند [۳۰]. نقشه منطقی [۳۰] یکی از پرکاربردترین نقشه‌های آشوبی است. نقشه منطقی یک نقشه چندجمله‌ای مرتبه دوم است و فرمول (۴) آن را توصیف می‌کند. در این فرمول، r یک پارامتر کنترل بین ۰ و ۴ است که رفتار متغیر X را تعیین می‌کند. متغیر X می‌تواند همگرا ($0 \le r \le 3$)، دوره‌ای ($3 < r \le 3.56$) یا بی‌نظم ($3.56 < r \le 4$) باشد. واضح است که تحت شرایطی که initial $X_0 \in [0,1]$ و $X_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$، $X \in [0,1]$.

$$X_{n+1} = rX_n(1-X_n) \quad (۴)$$

از آنجایی که اعداد آشوبی تولیدشده توسط نقشه منطقی در بازه ۰-۱ 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. فرمول (۵) برای نگاشت عدد آشوبی x به عددی صحیح در باز ۱ به L استفاده می‌شود [۳۰].

$$f(x) = \mathrm{ROUND}\left(\frac{L-1}{X_{high}-X_{low}}(x-X_{low})+1\right) \quad (۵)$$

۳،۳. 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 مسئله داشته باشند. برای مثال، در اجتماع محلی، عامل‌ها می‌توانند الگوریتم‌های جستجوی محلی متفاوت، الگوریتم‌های جستجوی محلی یکسان یا الگوریتم‌های جستجوی محلی یکسان با پارامترها و مولفه‌های جستجوی متفاوت را پیاده‌سازی کنند.

۴. طرح مسئله

پیمانه‌بندی نرم‌افزار فرایند گروه‌بندی موجودیت‌های نرم‌افزار به پیمانه‌ها است به‌طوری که موجودیت‌هایی که وابستگی بالایی دارند در پیمانه یکسان گروه‌بندی می‌شوند. پیمانه‌بندی به درک سامانه‌های نرم‌افزاری کمک می‌کند و باعث آسان‌تر شدن فرایند نگه‌داری آن‌ها می‌شود. عموما، گراف‌ها برای نمایش سامانه‌های نرم‌افزاری پیچیده استفاده می‌شوند [۶]. 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 $V=\{v_1, v_2, ..., v_n\}$ مجموعه‌ای از n موجودیت و $E \subseteq V \times V = \{(v_i, v_j) | v_i, v_j \in V \text{ and } i \neq j\}$ مجموعه‌ای از روابط بین موجودیت‌ها باشند. فرآیند پیمانه‌بندی منجر به گروه‌بندی همه موجودیت‌ها به k پیمانه غیرهمپوشان $\{m_1, m_2, ..., m_k\}$ می‌شود، که در آن $M_1 \cup M_2 \cup ... \cup M_k = V$ ، $M_i \neq \emptyset$ ، $M_i \cap M_j = \emptyset$ ، $i, j = 1, 2, ..., k$ و $i \neq j$ به‌طوری که مقدار تابع هدف (فرمول (۲))، 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 [۳].

۵. 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 ۱,۲) به عنوان روش انتخاب اجتماع استفاده می‌کند و قادر است به‌طور هوشمندانه اجتماع‌های مناسب را در طول مراحل متفاوت فرایند پیمانه‌بندی انتخاب کند. الگوریتم پیشنهادی از روش همه حرکات به عنوان روش پذیرش حرکت استفاده می‌کند؛ یعنی همیشه راه‌حل تولیدشده جدید را می‌پذیرد.

سطح پایین ابراکتشافی پیشنهادی شامل یک نمایش از مسئله، توابع ارزیابی و مجموعه‌ای از اجتماع‌ها است. در این مقاله، کدگذاری مبتنی بر مقدار استفاده‌شده در Bunch [۱۹] برای نمایش یک راه‌حل از مسئله استفاده می‌شود، اما با اشاره به [۱]، ما حداکثر تعداد پیمانه‌ها را در سامانه‌های نرم‌افزاری بزرگ (پوشه‌های موزیلا فایرفاکس) به $\min(100, n/3)$ محدود می‌کنیم. همچنین، TurboMQ پیشنهادشده در Bunch [۱۹] برای اندازه‌گیری کیفیت پیمانه‌بندی‌های به‌دست 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 به‌صورت هدفمند برای جستجوی فضای حالت مسئله پیمانه‌بندی طراحی شده‌اند. الگوریتم ۱ شبه کد سطح بالای AbHyh-SSM پیشنهادی را نشان می‌دهد.

همان‌طورکه در الگوریتم ۱ نشـان داده شـده اسـت، پـس از انتخاب بهترین اجتماع؛ یعنی یک اجتماع با بـالاترین مقـدار-Q (خط ۶، الگوریتم ۱)، عامل‌های اجتماع انتخاب‌شده بـه‌صـورت موازی و با شروع از راه‌حل فعلـی الگـوریتم اجـرا مـی‌شـوند و بهترین راه‌حل تولیدشده توسط عامل‌ها به عنوان راه‌حـل جدیـد در نظر گرفته می‌شود (خط ۷، الگوریتم ۱). سپس، با اسـتفاده از یک معیار پذیرش حرکت در مـورد پـذیرش یـا عـدم پـذیرش راه‌حل جدید تصمیم گرفته می‌شود (خط ۸، الگوریتم ۱). پس از آن، وضعیت بعدی بر اساس برازندگی نرمال‌شده آخرین راه‌حل پذیرفته‌شده مشخص می‌شود (خـط ۹، الگـوریتم ۱). مقـدار-Q اجتماع انتخاب‌شده بـر اسـاس کـارایی اجـرای آن بروزرسـانی می‌شود (خـط ۱۰، الگـوریتم ۱). سـپس، وضـعیت بروزرسـانی می‌شود (خط ۱۱، الگوریتم ۱) و بهترین راه‌حل سراسـری ثبـت می‌شود (خط ۱۲، الگوریتم ۱). یک اجتماع برای تکـرار بعـدی الگـوریتم انتخـاب مـی‌شـود (خـط ۵، الگـوریتم ۱). در ادامـه، جزئیات الگوریتم پیشنهادی توصیف می‌شوند.

الگوریتم (۱): 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
```

۱،۵. مجموعه عمل‌ها

از آنجایی که ما در الگوریتم پیشنهادی خود از یادگیری تقـویتی استفاده می‌کنیم، تعریف مجموعه عمل‌ها و مجموعه وضعیت‌هـا لازم اسـت. همـان‌طورکـه در بـالا توضـیح داده شـد، در روش پیشنهادی یک عمل یک اجتمـاع را نشـان مـی‌دهـد. AbHyh-SSM از دو اجتماع محلی و آشفتگی به ترتیب با هدف تقویت و تنوع استفاده می‌کند. از آنجـایی کـه هـر کـدام از عامـل‌هـای موجود در یک اجتماع می‌توانند دیدگاه خاص خود را در مـورد مسئله داشته باشند، آنها مجاز هستند که الگوریتم‌هـای متفـاوتی را از دسته اکتشافی اجتماع خود پیاده‌سازی کنند. این امر موجب می‌شود مسئله پیمانـه‌بنـدی از دیـدگاه‌هـای مختلـف بررسـی و جستجوی بهتری انجام شود. از طرفی، اجرای موازی عامل‌ها در هر اجتماع باعث افزایش قدرت محاسباتی می‌شود.

۲،۵. مجموعه وضعیت‌ها

در یادگیری تقویتی، وضعیت، شرایط محیطی برای تصمیم‌گیری در مورد یک عمل را نشان می‌دهد [۳۱]. ما از یـک طبقـه‌بنـدی وضعیت مبتنی بر برازندگی، اشاره‌شده در [۳۱]، استفاده می‌کنیم که نیاز به اطلاعات وابسته به دامنه کمتـری دارد. از آنجـایی کـه برازندگی مسئله‌های مختلـف دامنـه‌هـای متفـاوتی دارد، توسـط میانگین حرکت برازندگی (f_{new}) محاسبه‌شـده از تکـرار اول تـا تکرار فعلی نرمال می‌شود. برازندگی نرمـال‌شـده بـا اسـتفاده از فرمول (۶) به‌دست می‌آید. در این مقاله، وضعیت‌ها به سه طبقـه تقسیم می‌شوند: $\text{norm}(f)\in[0,0.67),[0.67,1.33),[1.33,\infty)$.

$$\text{norm}(f)=\frac{f}{\text{Avg}(f_{new})} \qquad (۶)$$

۳،۵. سیگنال تقویتی فوری

در تابع-Q (فرمول (۳))، r یک سیگنال یادگیری تقـویتی اسـت که پاداش یا جریمه را نشان می‌دهد. این پارامتر از یـادگیری-Q توسط کاربر تعریف می‌شود. با اشاره به [۳۱]، اگر بهترین راه‌حل سراسری بتواند در پایان اجرای عامل‌های اجتمـاع فعلـی بهبـود یابد، جفت وضعیت-عمل یک پاداش $r=1$ دریافت مـی‌کنـد؛ در غیر این صورت، یک جریمه $r=-1$ اعمال می‌شود.

۴،۵. پارامترهای تابع-Q

فاکتور تخفیف و نرخ یادگیری دو پـارامتر در تـابع-Q (فرمـول (۳)) هستند. فـاکتور تخفیـف، γ، تـاثیر پـاداش آینـده را تعیـین

می‌کند. در AbHyh-SSM ، با اشاره به [۳۱]، مقدار این پارامتر ۰٫۸ 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 [۳۱]، از فرمول (۷) برای کنترل پویای این پارامتر استفاده می‌شود که در آن، $iteration_{current}$ تکرار فعلی الگوریتم و $iteration_{max}$ حداکثر تعداد تکرار مجاز را برای اجرای الگوریتم پیشنهادی نشان می‌دهد.

$$\alpha_t=1-\left(0.9*\frac{iteration_{current}}{iteration_{max}}\right) \quad (۷)$$

۵٫۵. پارامتر 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 پارامتر r را برابر ۴ در نظر می‌گیریم.

۶٫۵. 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 به شرح زیر هستند.

۱٫۶٫۵. عامل ۱ اجتماع آشفتگی

الگوریتم ۲ رفتار اولین عامل از اجتماع آشفتگی را توصیف می‌کند. در این الگوریتم، ابتدا، یک عدد تصادفی صحیح t بین ۱ 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 فرمول (۴) تولید می‌شود و سپس عدد آشوبی تولیدشده با استفاده از فرمول (۵) به یک عدد صحیح در بازه ۱ به n نگاشت می‌شود. لازم به ذکر است که این عامل اجازه به 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.

الگوریتم (۲): شبه کد عامل ۱ 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
```

۲٫۶٫۵. عامل ۲ اجتماع آشفتگی

عامل دوم از اجتماع آشفتگی الگوریتم ۳ را پیاده‌سازی می‌کند. این عامل از دو مفهوم vertex cohesion و vertex coupling، اشاره‌شده در [۳۲]، برای شناسایی هدفمند گره‌هایی استفاده می‌کند که باید شماره پیمانه آنها تغییر کند. vertex cohesion نسبت تعداد رئوس متصل‌شده به راس u در داخل پیمانه شامل این راس به تعداد کل اتصال‌های درونی در این پیمانه است. vertex coupling نسبت تعداد رئوس اتصال خارجی به راس u به تعداد کل رئوس اتصال خارجی ممکن به این پیمانه است.

فرض کنید s راه‌حل دریافتی است، k تعداد پیمانه‌های s است و TurboMQ برازندگی s است. این عامل، برای همه گره‌ها در s رابطه vertex cohesion-vertex coupling<TurboMQ/k را بررسی می‌کند. اگر گره i این رابطه را ارضا کند، شماره پیمانه آن به‌طور تصادفی به شماره

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

الگوریتم (۳): شبه کد عامل ۲ 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 (f_s)
 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
```

۳,۶,۵. عامل ۳ اجتماع آشفتگی

الگوریتم ۴ 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، برای بررسی رابطه vertex cohesion-vertex coupling<TurboMQ/k برای هر گره i، پارامترهای این رابطه با توجه به جدیدترین راه‌حل ایجادشده محاسبه می‌شوند نه بر اساس اولین راه‌حل دریافتی s. همچنین اگر گره‌ای این رابطه را ارضا کند، تغییر شماره پیمانه آن نیز بر اساس جدیدترین راه‌حل تولیدشده انجام می‌شود.

الگوریتم (۴): شبه کد عامل ۳ 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′ (f_s′)
 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
```

۷,۵. 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 تست می‌کنیم، عامل‌های اجتماع محلی برای هر کدام از این دو دسته به‌صورت 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 گیر افتادن آنها در بهینه‌های محلی می‌شود. عامل‌های اجتماع محلی ۱، برای ده پوشه از موزیلا فایرفاکس و اجتماع محلی ۲، برای یازده سامانه نرم‌افزاری با اندازه کوچک و متوسط، در ادامه توصیف می‌شوند.

۱,۷,۵. عامل ۱ اجتماع محلی ۱

اولین عامل از اجتماع محلی ۱ الگوریتم ۵ 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 انتخاب می‌شود که در رابطه vertex cohesion-vertex coupling<TurboMQ/k 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 به‌طور 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 الگـوریتم قبـل از رسیدن به حداکثر زمان مجاز خاتمه می‌یابد.

الگوریتم (۵): شبه کد عامل ۱ اجتماع محلی ۱

```
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 (f_s)
 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 (f_s)
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
```

۲،۷،۵. عامل‌های ۲ و ۳ اجتماع محلی ۱

عامل‌های دوم و سوم از اجتماع محلی ۱ رفتار مشابهی دارنـد و الگوریتم ۶ را اجرا می‌کنند. در این الگوریتم، ابتدا برای هر گـره تعداد یال‌ها از آن به همه پیمانه‌ها محاسبه می‌شود. سـپس، یـک گره که تعداد یال‌های آن به پیمانه خود کمتر از پیمانه‌های دیگر است، انتخاب می‌شود. اگر گره‌ای با این ویژگی وجـود نداشـته باشد، اجرای الگوریتم قبل از رسـیدن بـه حـداکثر زمـان مجـاز خاتمه می‌یابد. از طرف دیگر، اگر چندین گـره بـا ایـن ویژگـی وجود داشته باشد، یک گره به‌طور تصادفی انتخاب می‌شـود. 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 می‌یابـد. ایـن عملیـات تـا زمانی‌که شرایط خاتمه ارضا نشوند، تکرار می‌شوند. بخـش‌هـای تصادفی این الگوریتم ممکن است منجر به عملکرد متفاوت این دو عامل شود.

الگوریتم (۶): شبه کد عامل‌های ۲ و ۳ اجتماع محلی ۱

```
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
```

۳،۷،۵. عامل‌های ۱ و ۲ اجتماع محلی ۲

عامل‌های اول و دوم از اجتماع محلی ۲ روش جستجوی محلی تکراری را پیاده‌سازی می‌کنند. الگوریتم ۷ شبه کد ایـن روش را نشان می‌دهد. تفاوت این دو عامـل در روش جسـتجوی محلـی استفاده‌شده توسط آنها است. عامـل اول از روش NAHC [۲۴] و عامل دوم از روش SAHC [۲۴] در بخش جستجوی محلـی استفاده می‌کند. هر دو عامل بـرای قسـمت آشـفتگی راه‌حـل از الگوریتم ۲ (عامل ۱ اجتمـاع آشـفتگی) و بـرای قسـمت معیـار پذیرش از روش فقط بهبـود اسـتفاده مـی‌کننـد. در روش فقـط بهبود، بهینه محلی جدید فقط زمانی پذیرفته می‌شود که بهتـرین راه‌حل یافت‌شده توسط الگوریتم را بهبود دهد. در این عامل‌هـا، با هدف تنوع بیشتر، برای تعیین تعداد تکرار الگـوریتم ۲ در هـر بار اجرا، ابتدا یک عدد آشوبی بـا اسـتفاده از فرمـول (۴) تولیـد می‌شود و سپس عدد آشوبی تولیدشده با اسـتفاده از فرمـول (۵) به یک عدد صحیح در بازه ۱ به $\lfloor n/2 \rfloor$ نگاشت مـی‌شـود کـه n تعداد ژن‌های کروموزوم است.

الگوریتم (۷): شبه کد عامل‌های ۱ و ۲ اجتماع محلی ۲

```
Ensure: Improved solution s''
 1: s_0 ← 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
```

۴،۷،۵. عامل ۳ اجتماع محلی ۲

عامل سوم از اجتماع محلی ۲ روش جستجوی همسایگی متغیر را پیاده‌سازی می‌کند. الگوریتم ۸ شبه کد این روش را نشان می‌دهد. این عامل از روش NAHC [۲۴] 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 به‌صورت 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 [۶]. در نتیجه، برای یک پیمانه‌بندی با k پیمانه و n گره، حداکثر تعداد همسایه‌ها $n*k$ است.

- در استراتژی دوم، ابتدا یک گره i از راه‌حل s که رابطه vertex cohesion-vertex coupling<turboMQ/k، اشاره شده در [۳۲]، را برآورده می‌کند برای تولید راه‌حل همسایه انتخاب می‌شود. سپس، شماره پیمانه گره انتخاب‌شده به‌طور تصادفی به شماره‌پیمانه یکی از همسایه‌هایش در s تغییر می‌کند. اگر گره i و همه همسایگانش شماره پیمانه یکسانی در s داشته باشند، شماره پیمانه گره i به‌طور 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 به‌طور تصادفی انتخاب می‌شود و شماره پیمانه آن به‌طور تصادفی به یکی از شماره‌های پیمانه در s یا یک شماره پیمانه جدید تغییر می‌کند.

الگوریتم (۸): شبه کد عامل ۳ اجتماع محلی ۲

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

۸،۵ شرط خاتمه الگوریتم

از آنجاییکه 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 ۱۰ تکرار متوالی بهبود داده نشود (یعنی الگوریتم همگرا شده باشد). حداکثر تعداد تکرار مجاز برای الگوریتم پیشنهادی نیز ۱۰۰۰ در نظر گرفته شده است.

۹،۵. 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 ۱)، 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 ۱ آورده شده است:

خط ۱: تولید یک راه‌حل اولیه (کروموزوم) تصادفی به طول n (O(n)) و مقدار دهی اولیه جدول-Q با سه سطر و دو ستون (O(1))، در نتیجه مرتبه فضایی O(n) است.

خط ۲: راه‌حل اولیه به عنوان بهترین راه‌حل سراسری در نظر گرفته می‌شود، در نتیجه مرتبه فضایی O(n) است.

خط ۳: راه‌حل اولیه به عنوان راه‌حل فعلی در نظر گرفته می‌شود، در نتیجه مرتبه فضایی O(n) است.

خط ۴: وضعیت اولیه با استفاده از فرمول (۶) محاسبه می‌شود، در نتیجه مرتبه فضایی O(1) است.

خط ۵: شرط خاتمه بررسی می‌شود، در نتیجه مرتبه فضایی O(1) است.

خط ۶: با توجه به وضعیت جستجو، بهترین اجتماع با توجه به جدول-Q انتخاب می‌شود، در نتیجه مرتبه فضایی O(1) است.

خط ۷: عامل‌های اجتماع آشفتگی یا اجتماع محلی به طور موازی اجرا می‌شوند. مرتبه فضایی عامل ۱ اجتماع آشفتگی O(1)، عامل ۲ اجتماع آشفتگی O(n)، عامل ۳ اجتماع آشفتگی O(1)، عامل ۱ اجتماع محلی ۱ O(1)، عامل‌های ۲ و ۳ اجتماع محلی ۱ O(1)، عامل ۱ اجتماع محلی ۲ O(n)، عامل ۲ اجتماع محلی ۲ O(n*k*n) و عامل ۳ اجتماع محلی ۲ O(n) است. در نتیجه مرتبه فضایی این خط $O(n^2*k)$ است.

خط ۸: در مورد پذیرش راه‌حل جدید به عنوان راه‌حل فعلی تصمیم می‌گیرد، در نتیجه مرتبه فضایی O(1) است.

خط ۹: وضعیت بعدی جستجو با استفاده از فرمول (۶) محاسبه می‌شود، در نتیجه مرتبه فضایی O(1) است.

خط ۱۰: درایه وضعیت-عمل در جدول-Q با استفاده از فرمول (۳) بروزرسانی می‌شود، در نتیجه مرتبه فضایی O(1) است.

خط ۱۱: وضعیت بعدی به عنوان وضعیت فعلی در نظر گرفته می‌شود، در نتیجه مرتبه زمانی O(1) است.

خط ۱۲: 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 $O(n^2*k)$ است.

۶. آزمایش‌ها

این بخش تنظیمات آزمایشی، نتایج آزمایش‌ها و پژوهش‌های آینده را ارائه می‌دهد. لازم به ذکر است که دیتاست و کد شبیه‌سازی مقاله در لینک زیر قابل دسترسی است:

https://github.com/mahjoubeh-t/Software-Modularization

آزمایش‌ها در متلب نسخه ۲۰۱۷ بر روی یک کامپیوتر با مشخصات رم ۴ گیگابایت و پردازنده پنج هسته‌ای ۲,۴ گیگاهرتز اجرا شده است.

۱,۶. تنظیمات آزمایشی

در این بخش، تنظیمات آزمایشی الگوریتم پیشنهادی توصیف می‌شود.

۱,۱,۶. سامانه نرم‌افزاری

ما یازده سامانه نرم‌افزاری دنیای واقعی با اندازه کوچک و متوسط و ده پوشه از موزیلا فایرفاکس با قابلیت‌ها و اندازه‌های مختلف را برای ارزیابی و مقایسه الگوریتم پیشنهادی انتخاب کرده‌ایم که جدول (۳) و جدول (۴) به ترتیب ویژگی‌های آنها را نشان می‌دهند.

۲,۱,۶. 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.

جدول (۳): 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

۲,۶. 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 [۱۹]، DAGC [۲۰]، Bunch-NAHC [۲۴]، SHC [۲۶]، GA-SMCP [۲۳]، EOD [۲۵] و SCSO [۲۷] هستند. ویژگی‌های این الگوریتم‌ها به‌طور خلاصه در جدول (۲) آورده شده است. علاوه بر این، ما از الگوریتم‌های پیمانه‌بندی سلسله‌مراتبی ادغامی مانند single linkage [۶]، complete linkage [۶]، average linkage [۶] و WCA-UE [۱۷] را برای مقایسه استفاده می‌کنیم. ویژگی‌های این الگوریتم‌ها به‌طور خلاصه در جدول (۱) آورده شده است. الگوریتم‌های FCA [۲۸]، k-means [۶] و ACDC [۲۹] نیز برای مقایسه انتخاب شده‌اند.

۱,۲,۶. سوال پژوهش ۱

برای پاسخ به سوال پژوهش ۱، ما الگوریتم پیشنهادی را با برخی از روش‌های مبتنی بر جستجو مانند Bunch [۱۹]، DAGC [۲۰]، HC+Bunch [۳۳]، NAHC [۲۴]، SAHC [۲۴] و EDA [۲۲] روی سامانه نرم‌افزاری mtunis مقایسه می‌کنیم. جدول (۵) 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.

۳,۱,۶. ارزیابی نتایج

در این مقاله از تابع هدف TurboMQ (فرمول (۲)) برای ارزیابی کیفیت پیمانه‌بندی‌ها استفاده می‌شود. برای ارزیابی قابلیت اطمینان الگوریتم پیشنهادی از معیارهای خارجی MoJo [۶]، Edge MoJo [۶] و F_m [۶] (میانگین هارمونیک Precision و Recall) استفاده می‌شود. علاوه بر این، زمان اجرای الگوریتم روی موردهای مطالعه یکسان با الگوریتم‌های دیگر مقایسه می‌شود.

۴,۱,۶. سوال‌های پژوهش

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

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

جدول (۴): ویژگی‌های پوشه‌های انتخاب‌شده از موزیلا فایرفاکس [۱]

Folder Name	Number of Files	Number of Links	Number of 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).

جدول (۵): 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**

۲،۲،۶. سوال پژوهش ۲

برای پاسخ به سوال پژوهش ۲، ما الگوریتم پیشنهادی را با برخی از الگوریتم‌های سلسله‌مراتبی و غیرسلسله‌مراتبی موجود مقایسه می‌کنیم. جدول (۶) 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 به‌صورت پررنگ مشخص می‌شوند. ستون آخر این جدول، رتبه الگوریتم پیشنهادی را در بین الگوریتم‌های مقایسه‌شده، 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 یافت‌شده از بقیه الگوریتم‌ها تولید می‌کند. در دو مورد دیگر، یعنی پوشه‌های Accessible و Browser، الگوریتم پیشنهادی رتبه دو را در بین الگوریتم‌های مقایسه‌شده دارد. در نهایت، می‌توان گفت میانگین بهبود عددی الگوریتم پیشنهادی روی این ده پوشه نسبت به بهترین روش مقایسه‌شده ۷۷،۶۰۷ درصد است.

جدول (۷) 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 به‌دست آمده توسط AbHyh-SSM بهتر یا مساوی کیفیت پیمانه‌بندی‌های تولیدشده توسط Bunch و DAGC هستند. در ستون مربوط به الگوریتم SCSO، از آنجایی که مقدار TurboMQ برای پنج سامانه نرم‌افزاری در مقاله [۲۷] 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 مقایسه‌شده برای هر سامانه ۰،۷۸۷ درصد است.

۳،۲،۶. سوال پژوهش ۳

برای پاسخ به سوال پژوهش ۳، الگوریتم پیشنهادی ۱۰ بار برای هر پوشه از موزیلا فایرفاکس اجرا می‌شود و پایداری نتایج توسط روش آماری t-test تحلیل می‌شود. انتظار می‌رود نتایج به‌دست آمده توسط الگوریتم برای ۱۰ اجرای مستقل به اندازه کافی به یکدیگر نزدیک باشند.

برای اعمال t-test، نتایج در دو گروه هم‌اندازه به نام‌های G_1 و G_2 گروه‌بندی شده و سپس برخی آمارهای توصیفی و استنباطی از آنها استخراج می‌شود. جدول (۸) 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 محاسبه‌شده برای دو گروه است. اگر p-value (ستون .Sig در جدول) بیشتر از سطح معنی‌داری (۰،۰۵ در آزمون‌های ما) باشد، نمی‌توان فرضیه صفر

واریانس‌های برابر را رد کرد. ستون‌های آخر جدول (۸) 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 به‌طور تصادفی از هم جدا شده‌اند، اشاره دارد. همه مقادیر .Sig (مقادیر پررنگ) بزرگتر از ۰,۰۵ 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 (۶): 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 (۷): 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 (۸): 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**

جدول (۱۰): مقایسه زمان اجرای الگوریتم پیشنهادی با برخی از روش‌های مبتنی بر جستجوی سراسری (ثانیه)

Folder name zaman	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**

۴،۲،۶. سوال پژوهش ۴

برای پاسخ به سوال پژوهش ۴، ما زمان اجرای الگوریتم پیشنهادی را با برخی از روش‌های مبتنی بر جستجوی سراسری مقایسه کردیم. برای الگوریتم‌های مبتنی بر ژنتیک مقایسه‌شده، ما از تنظیمات پارامتر استفاده‌شده در [۱] و [۲۸] استفاده کردیم و برای روش EoD [۲۵] از همان پارامترهایی استفاده کردیم که نویسندگان این الگوریتم استفاده کرده‌اند. فرض کنید n تعداد موجودیت‌ها را نشان می‌دهد، تنظیمات پارامتر الگوریتم‌های تکاملی مقایسه‌شده در جدول (۹) نشان داده شده است. جدول (۱۰) نتایج مقایسه زمان اجرا را روی ده پوشه از موزیلا فایرفاکس نشان می‌دهد. لازم به ذکر است که زمان اجرای الگوریتم‌های مبتنی بر جستجوی مقایسه‌شده از مقاله [۲۸] به‌دست آمده است که روی کامپیوتری با قدرت و حافظه بیشتر از کامپیوتر ما اجرا شده‌اند. بهترین نتایج در جدول (۱۰) به صورت پررنگ مشخص شده‌اند. نتایج این جدول اثبات می‌کند که در بیشتر موارد (هشت مورد از ده مورد) زمان اجرای روش پیشنهادی به‌طور قابل توجهی کمتر از الگوریتم‌های مقایسه‌شده است. در نهایت، می‌توان گفت میانگین بهبود عددی زمان اجرای الگوریتم پیشنهادی روی این ده پوشه نسبت به بهترین روش مقایسه‌شده ۵۹،۴۴۸ درصد است.

جدول (۹): تنظیمات پارامتر 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

۳،۶. پژوهش‌های آینده

پژوهش آینده باید به توسعه موارد زیر اختصاص داده شود:

۱- اعمال الگوریتم پیشنهادی روی مسئله‌های بهینه‌سازی ترکیباتی دیگر مانند رنگ‌آمیزی گراف و غیره.

۲- استفاده از برنامه‌های کاربردی دیگر برای ارزیابی روش پیشنهادی.

۳- استفاده از دیگر معیارهای ارزیابی الگوریتم.

۴- استفاده از اجتماع‌های بیشتر در ابراکتشافی پیشنهادی مانند اجتماع سراسری و غیره.

۵- تعریف همکاری بین عامل‌های موازی در هر اجتماع.

۷. نتیجه

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 به‌صورت یک اجتماع که شامل چندین عامل موازی است، تعریف شده است. نتایج تجربی الگوریتم پیشنهادی و مقایسه آنها با نتایج الگوریتم‌های پیمانه‌بندی شناخته‌شده نشان داد که AbHyh-SSM به‌طور همزمان میزان کیفیت پیمانه‌بندی‌ها و زمان اجرا را بهبود می‌دهد.

مراجع

[1] Pourasghar, B., Izadkhah, H., Isazadeh, A., Lotfi, S., "*A graph-based clustering algorithm for software systems modularization*", Information and Software Technology 133: 106469 (2021). https://doi.org/10.1016/j.infsof.2020.106469

[2] Izadkhah, H., Tajgardan, M., "*Information theoretic objective function for genetic software clustering*", Multidisciplinary Digital Publishing Institute Proceedings 46(1): 18 (2019). https://doi.org/10.3390/ecea-5-06681

[3] Tajgardan, M., Izadkhah, H., "*Critical Review of the Bunch: A Well-Known Tool for the Recovery and Maintenance of Software System Structures*", Critical Review 6(3): 363–367 (2017). DOI10.17148/IJARCCE.2017.6383

[۴] غلامشاهی، شبنم، هاشمی‌نژاد، سید محمدحسین، «**روشی برای تشخیص مؤلفه‌های نرم‌افزاری مبتنی بر الگوریتم ژنتیک مرتب‌سازی نامغلوب**»، مجله محاسبات نرم، جلد ۷، شماره ۲، ص ۴۷-۶۴، دانشگاه کاشان، پاییز و زمستان ۱۳۹۷.

[۵] نبی‌لو، مریم، دانشپور، نگین، «**ارائه یک الگوریتم خوشه‌بندی برای داده‌های دسته‌ای با ترکیب معیارها**»، مجله محاسبات نرم، جلد ۵، شماره ۱، ص ۱۴-۲۵، دانشگاه کاشان، بهار و تابستان ۱۳۹۵.

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

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

[8] Silva, M. A. L., de Souza, S. R., Souza, M. J. F., de Franca Filho, M. F., "*Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis*", Applied Soft Computing 71: 433–459 (2018). https://doi.org/10.1016/j.asoc.2018.06.050

[9] Malek, R., "*An agent-based hyper-heuristic approach to combinatorial optimization problems*", IEEE International Conference on Intelligent Computing and Intelligent Systems 3: 428–434 (2010).

[10] Hassan, A., Pillay, N., "*Hybrid metaheuristics: An automated approach*", Expert Systems with Applications 130: 132–144 (2019). https://doi.org/10.1016/j.eswa.2019.04.027

[11] Meignan, D., Koukam, A., Créput, J.-C., "*Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism*", Journal of Heuristics 16(6): 859–879 (2010). https://doi.org/10.1007/s10732-009-9121-7

[12] Huang, J., Liu, J., Yao, X., "*A multi-agent evolutionary algorithm for software module clustering problems*", Soft Computing 21(12): 3415–3428 (2017). https://doi.org/10.1007/s00500-015-2018-5

[13] Kumari, A. C., Srinivas, K., "*Hyper-heuristic approach for multi-objective software module clustering*", Journal of Systems and Software 117: 384–401 (2016). https://doi.org/10.1016/j.jss.2016.04.007

[14] Tajgardan, M., Izadkhah, H., Lotfi, S., "*A Reinforcement Learning-based Iterated Local Search for Software Modularization*", 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS) 1–6 (2022). 10.1109/ICSPIS56952.2022.10043949

Engineering 32(3): 193-208 (2006). https://doi.org/10.1109/TSE.2006.31

[25] Sadat Jalali, N., Izadkhah, H., Lotfi, S., "*Multi-objective search-based software modularization: structural and non-structural features*", Soft Computing 23(21): 11141–11165 (2019). https://doi.org/10.1007/s00500-018-3666-z

[26] Kargar, M., Isazadeh, A., Izadkhah, H., "*Semantic-based software clustering using hill climbing*", International Symposium on Computer Science and Software Engineering Conference (CSSE) 55–60 (2017). https://doi.org/10.1109/CSICSSE.2017.8320117

[27] Arasteh, B., Seyyedabbasi, A., Rasheed, J., M. Abu-Mahfouz, A., "*Program Source-Code Re-Modularization Using a Discretized and Modified Sand Cat Swarm Optimization Algorithm*", Symmetry 15(2): 401 (2023). https://doi.org/10.3390/sym15020401

[28] Teymourian, N., Izadkhah, H., Isazadeh, A., "*A fast clustering algorithm for modularization of large-scale software systems*", IEEE Transactions on Software Engineering 48(4): 1451-1462 (2020). https://doi.org/10.1109/TSE.2020.3022212

[29] Tzerpos, V., Holt, R. C., "*Accd: an algorithm for comprehension-driven clustering*", Proceedings Seventh Working Conference on Reverse Engineering 258-267 (2000). https://doi.org/10.1109/WCRE.2000.891477

[30] Zarei, B., Meybodi, M. R., Masoumi, B., "*Chaotic memetic algorithm and its application for detecting community structure in complex networks*", Chaos: An Interdisciplinary Journal of Nonlinear Science 30(1): 13125 (2020). https://doi.org/10.1063/1.5120094

[31] Choong, S. S., Wong, L. P., Lim, C. P., "*Automatic design of hyper-heuristic based on reinforcement learning*", Information Sciences 436: 89–107 (2018). https://doi.org/10.1016/j.ins.2018.01.005

[32] Izadkhah, H., Elgedawy, I., Isazadeh, A., "*E-CDGM: An Evolutionary Call-Dependency Graph Modularization Approach for Software Systems*", Cybernetics and Information Technologies 16(3): (2016). http://dx.doi.org/10.1515/cait-2016-0035

[33] Mahdavi, K., *A clustering genetic algorithm for software modularisation with a multiple hill*

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

[16] Saeed, M., Maqbool, O., Babri, H. A., Hassan, S. Z., Sarwar, S. M., "*Software clustering techniques and the use of combined algorithm*", Seventh European Conference on Software Maintenance and Reengineering 301-306 (2003). 10.1109/CSMR.2003.1192438

[17] Maqbool, O., Babri, H., "*Hierarchical clustering for software architecture recovery*", IEEE Transactions on Software Engineering 33(11): 759-780 (2007). https://doi.org/10.1109/TSE.2007.70732

[18] Naseem, R., Maqbool, O., Muhammad, S., "*Cooperative clustering for software modularization*", Journal of Systems and Software 86(8): 2045–2062 (2013). https://doi.org/10.1016/j.jss.2013.03.080

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

[20] Parsa, S., Bushehrian, O., "*A New Encoding Scheme and a Framework to Investigate Genetic Clustering Algorithms*", Journal of Research and Practice in Information Technology 37(1): 127–143 (2005).

[21] Harman, M., Yao, X., "*Software module clustering as a multi-objective search problem*", IEEE Transactions on Software Engineering 37(2): 264–282 (2010). https://doi.org/10.1109/TSE.2010.26

[22] Tajgardan, M., Izadkhah, H., "*Software Systems Clustering Using Estimation of Distribution Approach*", Journal of Applied Computer Science Methods 8(2): 99–113 (2016). http://dx.doi.org/10.1515/jacsm-2016-0007

[23] Huang, J., Liu, J., "*A similarity-based modularization quality measure for software module clustering problems*", Information Sciences 342: 96–110 (2016). https://doi.org/10.1016/j.ins.2016.01.030

[24] Mitchell B. S., Mancoridis, S., "*On the automatic modularization of software systems using the bunch tool*", IEEE Transactions on Software

climbing approach, Ph.D. Thesis, Brunel University, 2005.