یک الگوریتم مبتنی بر افرازبندی گراف برای خوشه‌بندی سامانه‌های نرم‌افزاری با ابعاد بزرگ

بابک پوراصغر*[۱]، حبیب ایزدخواه[۲]، شهریار لطفی[۳] و خیام صالحی[۴]
[۱، ۲ و۳]گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه تبریز، تبریز، ایران
[۴]گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه شهرکرد، شهرکرد، ایران

چکیده

از روش‌های خوشه‌بندی برای بازیابی ساختار نرم‌افزار جهت فهم درست آن و همچنین بازسازی نرم‌افزار استفاده می‌شود. در ادبیات موضوع، بیشتر الگوریتم‌های ارائه‌شده برای خوشه‌بندی سامانه‌های نرم‌افزاری به دو دستهٔ الگوریتم‌های مبتنی بر جستجو و الگوریتم‌های سلسله‌مراتبی طبقه‌بندی می‌شوند و الگوریتمی از ردهٔ مبتنی بر افراز برای خوشه‌بندی یک سامانهٔ نرم‌افزاری ارائه نشده است. این روش‌ها سعی دارند که گراف وابستگی موجودیت به‌دست‌آمده از کد منبع سامانهٔ نرم‌افزاری را به چند مجموعهٔ رأسی افراز کنند. در سامانه‌های نرم‌افزاری، موجودیت می‌تواند رده، تابع و یا یک فایل باشد. با توجه به چندجمله‌ای غیر قطعی، سخت‌بودن مسألهٔ خوشه‌بندی، در سال‌های اخیر از روش‌های تکاملی و مبتنی بر جستجو مانند الگوریتم ژنتیک برای این حل این مسأله، زیاد استفاده شده است. هر چند این الگوریتم‌ها در برخی موارد می‌توانند ساختار مناسبی از نرم‌افزار را به‌دست آورند، اما برای نرم‌افزارهای با ابعاد بزرگ، با توجه به زمان اجرا و حافظهٔ مصرفی زیاد، قابل اجرا نیستند؛ همچنین، این روش‌ها از اطلاعات و دانش گرافی موجود در گراف وابستگی موجودیت استفاده‌ی چندانی نمی‌کنند. در این مقاله یک الگوریتم مبتنی بر افراز ارائه شده است که بتوان از آن در خوشه‌بندی نرم‌افزار نیز استفاده کرد. همچنین، یک نوع فاصلهٔ جدید برای قیاس تشابه و عدم تشابه ارائه شده است. انتظار می‌رود روش پیشنهادی بتواند در قیاس با سایر روش‌های موجود، خوشه‌بندی‌هایی با کیفیت بالاتر و نزدیک به خوشه‌بندی فرد خبره، تولید کند. برای بررسی صحت اجرای الگوریتم، آن را بر روی نرم‌افزار موزیلا فایرفاکس اجرا کرده و نتایج را با الگوریتم‌های مطرح این حوزه، مقایسه کرده‌ایم.

واژگان کلیدی: مهندسی نرم‌افزار، مهندسی معکوس، خوشه‌بندی نرم‌افزار، الگوریتم K-means

A partition-based algorithm for clustering large-scale software systems

Babak Pourasghar[1], Habib Izadkhah[*2], Shahriar Lotfi[3] & Khayyam Salehi[4]
[1,2,3]Department of Computer Science, University of Tabriz, Tabriz, Iran
[4]Department of Computer Science, Shahrekord University, Shahrekord, Iran

Abstract

Clustering techniques are used to extract the structure of software for understanding, maintaining, and refactoring. In the literature, most of the proposed approaches for software clustering are divided into hierarchical algorithms and search-based techniques. In the former, clustering is a process of merging (splitting) similar (non-similar) clusters. These techniques suffered from the drawbacks such as finiteness criterion and arbitrary decisions occurred in the process. Because of the NP-hardness of clustering software systems, evolutionary and search-based algorithms are more commonly used algorithm than hierarchical ones. In evolutionary algorithms, the clustering of software systems is considered as a problem of searching over some possible clustering candidates. Although these algorithms are often able to achieve an appropriate structure of the software, they are not applicable in clustering large-scale software. Furthermore, these algorithms are unable to consider the knowledge in the artifact dependency graph, which extracted from the source code of the software. In software

* Corresponding author
*نویسندهٔ عهده‌دار مکاتبات

●تاریخ ارسال مقاله: ۱۳۹۸/۰۴/۱۹ ●تاریخ پذیرش: ۱۳۹۹/۰۵/۲۸ ●تاریخ انتشار: ۱۴۰۰/۱۲/۲۹ ●نوع مطالعه: پژوهشی

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

systems, an artifact can be everything like a class, a function, or a file. In this paper, a new partition-based clustering algorithm is presented. This algorithm attempts to partition the artifact dependency graph considering the knowledge therein. Moreover, a new distance criterion is presented to measure the similarity and dissimilarity of the artifacts. The proposed algorithm starts with the artifact dependency graph and creates the similarity matrices of the artifacts. So, it attempts to refine the partition candidate until a fixed point is reached. We expect that the proposed method compared with other methods could lead to achieve the clustering with high quality and similar to the expert's clustering based on MoJo-FM measure. To demonstrate the applicability and validity of the proposed algorithm, a large-scale case study, Mozilla Firefox, is employed. The results demonstrate that the proposed algorithm outperforms the commonly used evolutionary methods in the literature.

Keywords: Software Engineering, Reverse Engineering, Software Clustering, K-means algorithm.

۱- مقدمه

مهندسان نرم‌افزار در فرایند نگهداری نرم‌افزار وقت زیادی را صرف فهم کد منبع آن نرم‌افزار می‌کنند. عدم فهم درست کد نرم‌افزار نوشته شده، باعث می‌شود که تغییر ساختار آن نرم‌افزار جهت پاسخ‌گویی به نیازمندی‌های جدید و همچنین استفادهٔ مجدد از مؤلفه‌های موجود آن، با محدودیت مواجه شود. این تغییرات می‌تواند ناشی از تغییر نیازمندی‌ها، افزوده‌شدن نیازمندی‌های جدید و یا تغییر سیاست‌های اعمال شده باشد. فهم، یک مسألهٔ ذهنی است و نمی‌توان به‌راحتی، معیارهای دقیق ریاضیاتی برای آن تعیین کرد. هر چه کد منبع یک نرم‌افزار، قابل فهم‌تر باشد، توسعه‌دهندگان بعدی آن نرم‌افزار، درک درست‌تری از آن برنامه خواهند داشت و امکان استفادهٔ مجدد، اصلاح و توسعهٔ بهتر فراهم خواهد شد. فهم بهتر یک نرم‌افزار، با بررسی معماری آن نرم‌افزار که یک طرح از اجزای موجود در سامانه و ارتباط بین اجزا است، امکان‌پذیر است. خوشه‌بندی[1] نرم‌افزار (یا پیمانه‌بندی[2] نرم‌افزار) یک فعالیت کلیدی در مهندسی معکوس برای استخراج معماری نرم‌افزار در جهت فهم و نگهداری نرم‌افزار است[3]. خوشه‌بندی نرم‌افزار روشی است که در طی آن، ساختارهای منطقی کد منبع نظیر رده‌ها، توابع و غیره به مجموعه‌هایی به نام خوشه، افراز می‌شوند که با توجه به تعداد ارتباطات اجزا، اتصال[4] کمینه و انسجام[5] بیشینه شود. اتصال، ارتباطات خارجی بین مؤلفه‌ها و انسجام، ارتباطات داخلی مؤلفه‌ها را بیان می‌کند. به عبارت دیگر در یک سامانه نرم‌افزاری، کلاس‌ها ساختار منطقی سامانه را تشکیل می‌دهند و با گروه‌بندی رده‌های منطقاً مرتبط به هم در یک واحد یا خوشه، ساختار فیزیکی سامانه مشخص می‌شود [1، 2]. مهم‌ترین معیار کیفیت یک سامانه نرم‌افزاری این است که موجودیت[6]هایی که در یک خوشه قرار می‌گیرند، بیشینه ارتباط/تشابه را با هم داشته باشند و موجودیت‌های موجود در خوشه‌های مجزا، کمینه ارتباط/تشابه را داشته باشند.

روال کلی خوشه‌بندی نرم‌افزار از مرحله کد منبع تا مرحله نمایش، برگرفته از [2] در شکل (۱) نشان داده شده است. خوشه‌بندی یک سامانه نرم‌افزاری از سه مرحله تشکیل شده است. در مرحله نخست کد منبع پیمایش شده و گراف وابستگی از آن استخراج می‌شود. در مرحله دوم گراف وابستگی استخراج‌شده به‌وسیلهٔ الگوریتم‌هایی، خوشه‌بندی می‌شود. در مرحله آخرخوشه‌بندی حاصل با استفاده از ابزارهای نمایش گرافیکی نشان داده می‌شوند.

خوشه‌بندی در دو زمینه متفاوت در مهندسی نرم‌افزار به کار رفته است که عبارتند از بازیابی ساختار نرم‌افزار[7] و بازسازی نرم‌افزار[8].

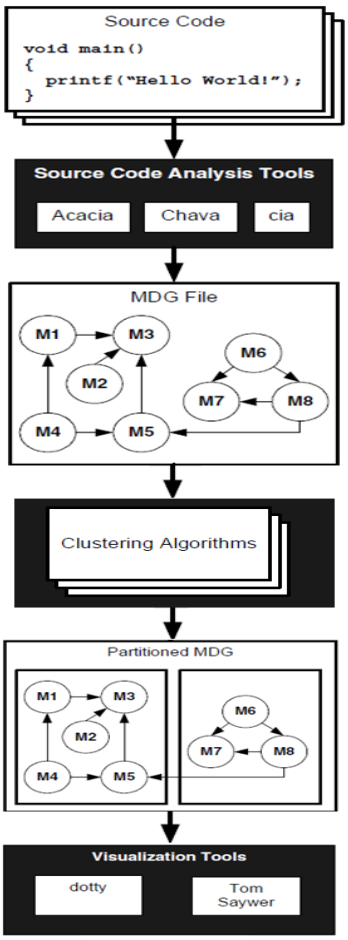
هدف از بازیابی معماری نرم‌افزار، استفاده از روش‌های خوشه‌بندی برای تجزیه یک سامانه نرم‌افزاری از روی کد منبع به زیرسامانه‌های معنی‌دار و قابل فهم است. درواقع، این کار به فهم نرم‌افزار در فرایند مهندسی معکوس نرم‌افزار کمک می‌کند. بازسازی نرم‌افزار نیز از خوشه‌بندی استفاده می‌کند، اما با اهدافی متفاوت. در روش‌های بازسازی نرم‌افزار یا روش‌های خوشه‌بندی مجدد یک سامانه به‌عنوان ورودی گرفته می‌شود و به‌عنوان خروجی یک سامانه به‌طورکامل جدید با خوشه‌بندی جدید ارائه می‌شود [3-5].

در منابع از نظر فرآیند خوشه‌بندی، دو نوع روش برای خوشه‌بندی سامانه‌های نرم‌افزاری پیشنهاد شده است که عبارتند از روش‌های خوشه‌بندی سلسله‌مراتبی مانند single linkage [1], complete linkage [1], average linkage [1], combined algorithm [6],

[1] clustering
[2] modularization
[3] لازم به ذکر است که عنوان کلی این کار، خوشه‌بندی است. ولی در ادبیات موضوع، برای خوشه‌بندی نرم‌افزار، اصطلاح پیمانه‌بندی به کار می‌رود.
[4] cohesion
[5] coupling
[6] artifact
[7] software architecture recovery
[8] refactoring

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

weighted combined algorithm [6], Information theory [7] و روش‌های مبتنی بر جستجو مانند EDA [8]، E-CDGM [9]، Bunch [2]، DAGC [10]، SHC [11]، ECA [12] و MCA [12].



(شکل-۱): طرح کلی خوشه‌بندی نرم‌افزار
(Figure-1): Outline of software Clustering

در روش‌های سلسله‌مراتبی، ابتدا تمام موجودیت‌ها به‌عنوان خوشه‌های مجزا در نظر گرفته می‌شوند و در طی فرآیندی تکراری در هر مرحله خوشه‌هایی که شباهت بیشتری با هم دارند ترکیب می‌شوند.

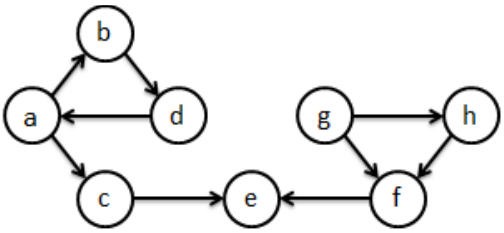
روش‌های خوشه‌بندی سلسله‌مراتبی از سه مشکل عمده رنج می‌برند که عبارتند از: حریصانه‌بودن این روش‌ها، که با توجه فضای بسیار بزرگ، روش‌های حریصانه نمی‌توانند به‌خوبی فضای مسأله را مورد جستجو قرار دهند که باعث می‌شود کیفیت خوشه‌بندی به‌دست‌آمده پایین باشد. مشکل دوم این روش‌ها این است که معیارهای مناسبی برای این که در چه مرحله‌ای باید عمل خوشه‌بندی قطع شود و ادامه نیابد وجود ندارد و مشکل سوم این است که گاهاً در گامی از فرآیند خوشه‌بندی، انتخاب اختیاری دو موجودیت از بین چندین موجودیت که شباهت یکسانی دارند، رخ می‌دهد، که این روش‌ها نیز از این مشکل تصمیم‌گیری اختیاری رنج می‌برند. با توجه به این مشکلات، امروزه از روش‌های سلسله‌مراتبی در خوشه‌بندی سامانه‌های نرم‌افزاری، کمتر استفاده می‌شود.

در روش‌های خوشه‌بندی مبتنی بر جستجو، با مسأله خوشه‌بندی به‌عنوان یک مسأله جستجو برخورد می‌شود. بیش‌تر روش‌هایی که از جستجو برای پیداکردن خوشه‌بندی مناسب استفاده می‌کنند، هدف آنها پیداکردن یک خوشه‌بندی است که انسجام را بیشینه و اتصال را کمینه کند. مشکل اصلی این روش‌ها این است که در گراف‌های بزرگ، بسیار کند عمل می‌کنند.

خوشه‌بندی نرم‌افزار از گراف‌هایی که نمایش انتزاعی نرم‌افزار را نشان می‌دهند، به‌عنوان ورودی استفاده می‌کند. این گراف‌ها می‌توانند از روی مستندهای نرم‌افزار مانند نمودارهای طراحی UML کد منبع و غیره به‌دست آمده باشند و یا با کمک مهندسی معکوس حاصل شده باشند.

استفاده از دو نوع گراف در خوشه‌بندی نرم‌افزار مرسوم است، گراف وابستگی فراخوانی CDG و گراف وابستگی ویژگیِ موجودیت AFDG. در گراف CDG هر راس یک موجودیت از سامانه است و هر یال، فراخوانی‌های بین موجودیت‌ها و یا رابطهٔ ارث‌بری بین آنها را نشان می‌دهد. در یک نرم‌افزار، یک موجودیت می‌تواند از هر نوعی مانند مولفه‌ی نرم‌افزاری، زیرسامانه، رده، متد، فایل و توابع در نظر گرفته شود. در گراف AFDG نیز دو نوع رأس به نام‌های موجودیت و ویژگی وجود دارد. ویژگی می‌تواند خود یک موجودیت، جدول پایگاه داده، کتابخانه و یا حتی یک متغیر سراسری باشد. یال‌های بین موجودیت‌ها و ویژگی‌ها نشان‌دهندهٔ فراخوانی‌ها و یا استفاده از ویژگی‌ها است. گفتنی است که در ادبیات موضوع، این دو گراف را از نوع ADG دانسته و در برخی موارد به جای عبارت ADG از عبارت MDG نیز استفاده می‌شود.

ورودی بیشتر الگوریتم‌های خوشه‌بندی نرم‌افزار، یک گراف وابستگی موجودیت است. شکل (۲) نشان‌دهنده یک ADG برای یک سامانه نرم‌افزاری است.



(شکل-۲): یک گراف وابستگی موجودیت
(Figure-2): An artifact dependency graph

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

الگوریتم K-means به‌دلیل داشتن کارآیی مناسب، در بیش‌تر کاربردها استفاده شده است، کاربردهایی مثل داده‌کاوی، تشخیص الگو، آنالیز تصاویر، بیوانفورماتیک، یادگیری ماشین، پردازش صوت، پردازش تصویر، پردازش متن، خوشه‌بندی محتوای وب، آنالیز گزارش‌های هواشناسی و غیره.

اما از این روش برای خوشه‌بندی یک سامانه نرم‌افزاری جهت فهم بهتر آن استفاده نشده است. ما در این مقاله، یک الگوریتم الهام گرفته شده از K-means جهت خوشه‌بندی یک سامانه نرم‌افزاری ارائه می‌دهیم.

مسألهٔ بررسی شده در این مقاله، ارائهٔ یک خوشه‌بندی جدید برای سامانه‌های نرم‌افزاری است، به‌طوری‌که خوشه‌بندی به‌دست‌آمده، به خوشه‌بندی یک فرد خبره نزدیک باشد. ما معتقدیم که با استفاده از یک روش خوشه‌بندی مبتنی بر افزار با استفاده از مفاهیم طول کوتاه‌ترین مسیر دسترسی بین موجودیت‌ها می‌توان خوشه‌هایی با کیفیت بالاتر به‌دست آورد.

ادامهٔ مقاله به‌صورت زیر سازماندهی شده است. در بخش ۲، الگوریتم k-means استاندارد توضیح داده شده و در بخش ۳، الگوریتم پیشنهادی برای خوشه‌بندی یک سامانه نرم‌افزاری ارائه شده و در بخش ۴، مورد مطالعهٔ استفاده‌شده در این مقاله مورد بررسی قرار گرفته است. در بخش ۵، نتایج تجربی و در بخش ۶ خلاصهٔ مقاله آمده است.

۲- الگوریتم K-means استاندارد

روش K-means یکی از روش‌های خوشه‌بندی داده‌ها است. این الگوریتم یک روش مسطح[1] مبتنی بر افراز[2] محسوب می‌شود. در الگوریتم K-means ابتدا k عضو (که k تعداد خوشه‌ها است) به‌صورت تصادفی از میان n عضو به‌عنوان مراکز خوشه‌ها انتخاب می‌شوند؛ سپس n-k عضو باقی‌مانده هرکدام به نزدیک‌ترین خوشه تخصیص می‌یابند. بعد از تخصیص همه اعضا، مراکز خوشه دوباره محاسبه می‌شوند و داده‌ها دوباره با توجه به مراکز جدید به خوشه‌ها تخصیص می‌یابند و این کار تا زمانی که خوشه‌ها تغییر نیابند یا تابع خطا کم نشود ادامه می‌یابد.

به چهار دلیل نمی‌توان K-means استاندارد را روی خوشه‌بندی یک نرم‌افزار از روی ADG اعمال کرد که عبارتند از: (۱) عدم امکان محاسبه میانگین خوشه در هر مرحله به‌دلیل اینکه محاسبه میانگین گره‌ها در ساختار نرم‌افزار بی‌معنی است؛ (۲) عدم امکان اعمال معیارهایی نظیر فاصله اقلیدسی به‌دلیل این‌که فضای اقلیدسی در نرم‌افزار بی‌معنی است؛ (۳) به‌دلیل اینکه همواره فاصله داده‌ها از میانگین خوشه سنجیده می‌شود، بنابراین خوشه‌های تولیدشده در K-means همواره محدب هستند؛ درحالی‌که شکل خوشه‌های به‌دست‌آمده در نرم‌افزار هر شکلی می‌توانند داشته باشند، (۴) این روش به نوفه در داده‌ها حساس است. اگر یک داده، فاصله زیادی با بقیه داشته باشد، میانگین خوشه را به سمت خودش می‌کشد و درنتیجه امکان دارد خوشه‌بندی مناسبی ایجاد نشود. اغلب برای حل این مسأله، از روش‌های گوناگون حذف داده‌های پرت استفاده می‌کنند.

هدف روش K-means خوشه‌بندی موجودیت‌ها بر اساس ویژگی‌ها است. در این روش موجودیت‌ها از ویژگی‌ها جدا هستند؛ اما در ADG بحث ویژگی وجود ندارد و ارتباط بین موجودیت‌ها به‌صورت رابطه وابستگی است و چیزی به نام ویژگی وجود ندارد. برای مثال جدول (۱) نشان‌دهندهٔ چهار موجودیت و سه ویژگی برای خوشه‌بندی به‌وسیلهٔ K-means است و جدول (۲) نشان‌دهنده جدول داده برای گراف ADG شکل (۲) است. همان‌طور که مشاهده می‌کنید، عناوین سطرها با عناوین ستون‌ها یکسان و همجنس هستند.

(جدول-۱): جدول داده مورد استفاده در K-means استاندارد
(Table-1): Data table for standard K-means

	F1	F2	F3
A1			
A2			
A3			
A4			

(جدول-۲): جدول داده برای ADG شکل (۲)
(Table-2): Data table for ADG in figure (2)

	a	b	c	d	e	f	g	h
a	0	1	1	1	0	0	0	0
b		0	0	1	0	0	0	0
c			0	0	1	0	0	0
d				0	0	0	0	0
e					0	1	0	0
f						0	1	1
g							0	1
h								0

در جدول (۱) اگر سطرهای مرتبط با موجودیت‌های A1 و A2 درکل برابر با عدد یک یا درکل برابر با عدد صفر باشد در این‌صورت فاصله اقلیدسی بین این دو موجودیت برابر صفر خواهد شد که نشان‌دهنده

[1] Flat
[2] Partition based

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

بیشینه تشابه بین دو موجودیت است؛ درحالی‌که این قضیه در نرم‌افزار درکل متفاوت است. اگر سطرهای مربوط به دو موجودیت درکل صفر باشد نشان‌دهنده عدم تشابه کامل بین دو موجودیت است؛ درحالی‌که فاصله اقلیدسی، آنها را به‌عنوان بیشینه تشابه معرفی می‌کند و آنها را به یک خوشه یکسان وارد می‌کند.

سؤال مهم دیگری که باید در این مرحله پاسخ داده شود، این است که با توجه به این‌که حوزهٔ داده‌کاوی و به‌خصوص خوشه‌بندی، حوزه‌ای پرطرفدار و بسیار مترقی است و الگوریتم‌های پرشماری برای خوشه‌بندی معرفی شده است، چرا مسألهٔ خوشه‌بندی نرم‌افزار در قیاس با سایر مسائل خوشه‌بندی، کم کار شده است و آن همه الگوریتم کارآمد، چرا در این حوزه استفاده نمی‌شود؟

در جواب این مسأله باید گفت که دلیل اصلی ناکارآمدبودن بیشتر روش‌های خوشه‌بندی در این زمینه، فضای غیراقلیدسی داده‌ها است. داده‌هایی که خوشه‌بندی روی آنها باید انجام شود از جنس گراف هستند. گراف‌ها در فضای غیر اقلیدسی قرار دارند؛ درحالی‌که بیشتر روش‌های خوشه‌بندی برای فضای اقلیدسی هستند. در فضای اقلیدسی هر داده، مختصاتی ثابت و مشخص دارد و هرگونه تغییر در مختصات آن داده، ماهیت آن را عوض می‌کند؛ درحالی‌که رأس‌های گراف فاقد مختصات هستند و تغییر محل رأس در گراف رسم‌شده، ماهیت گراف را تغییر نمی‌دهد. با این حال، بازهم الگوریتم‌های پرشماری برای خوشه‌بندی گراف مطرح شده‌اند. در ادامه، برخی تفاوت‌های گراف‌های نرم‌افزاری با سایر گراف‌ها بر شمرده می‌شود تا دلیل عدم استفاده از الگوریتم‌های رایج خوشه‌بندی گراف در این مسأله بیان شود:

۱. وجود یک سری رأس خاص با عنوان مؤلفه‌های کاربردی[1] در گراف‌های نرم‌افزاری؛
۲. وجود یک سری رأس خاص با عنوان کتابخانه در گراف‌های نرم‌افزاری؛
۳. عدم وجود نوفه در گراف‌های نرم‌افزاری؛
۴. وجود رأس‌هایی با عنوان مبدأ جریان و خروجی جریان در گراف‌های نرم‌افزاری؛
۵. وجود خاصیت جریانی در گراف‌های نرم‌افزاری؛
۶. خلوت‌بودن بیشتر گراف‌های نرم‌افزاری؛
۷. وجود اطلاعات غیر ساختاری در رأس‌ها از قبیل کد منبع و توضیحات؛
۸. وجود اطلاعات غیر ساختاری در رأس‌ها نظیر زبان کد نوشته‌شده و یا تیم نویسندهٔ آن قطعه کد که به‌صورت رأس نشان داده شده است؛
۹. رفتارهای عجیب و غیر منتظرهٔ فرد خبرهٔ نرم‌افزار که با ماهیت گراف‌های دیگر و سیاست‌های افراد خبرهٔ آنها تفاوت دارد.

[1] Utility

۳- روش پیشنهادی

در این بخش، روش پیشنهادی خود را برای خوشه‌بندی گراف نرم‌افزار با استفاده از روش K-means تغییر یافته، توضیح می‌دهیم. برای اعمال K-means روی نرم‌افزار لازم است، برخی از تعاریف آن، دوباره تعریف شوند.

۳-۱- تعاریف

تعریف فاصله: فاصله دو موجودیت (رأس) را طول کوتاه‌ترین مسیر بین دو موجودیت تعریف می‌کنیم. فاصله تمام جفت رأس‌ها را می‌توان با الگوریتم فلوید-وارشال در یک ماتریس، به‌دست آورد [13]. نتیجه حاصل را، ماتریس فاصله می‌نامیم. علت انتخاب این نوع فاصله این است که دو موجودیت که یک وابستگی (برای مثال فراخوانی) بین آن‌ها وجود دارد، نسبت به دو موجودیت مجزا، دارای فاصله کمتری هستند. سلسله‌مراتب فراخوانی‌ها نیز قابل توجیه است. دو موجودیت که با یک واسطه همدیگر را فراخوانی می‌کنند، فاصله کمتری نسبت به دو موجودیت با بیشتر از یک واسطه دارند.

در مهندسی نرم‌افزار و به‌خصوص در مبحث خوشه‌بندی نرم‌افزار از مفهوم تشابه، بیشتر از مفهوم فاصله استفاده می‌شود. ما نیز درعمل، مفهوم فاصله را تغییر داده و از مفهوم تشابه استفاده خواهیم کرد. برای پژوهش‌گران این حوزه، تشابه و یا عدم تشابه دو مؤلفهٔ نرم‌افزاری با معنی‌تر از تفاوت و فاصلهٔ بین آن دو است.

در این مقاله از مفهوم شباهت بین جفت موجودیت‌ها برای خوشه‌بندی استفاده می‌شود. با توجه به اینکه شباهت، یک رابطه دوطرفه است و وجود یک یال بین دو موجودیت را دلیل شباهت به حساب می‌آوریم، لذا یال‌ها بدون جهت در نظر گرفته می‌شوند.

تعریف قطر گراف ADG: بیشترین فاصله موجود در بین جفت رأس‌ها. این مقدار، بیشینه مقدار موجود در ماتریس فاصله است. برای گراف شکل (۲)، مقدار ۵ به‌دست می‌آید.

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

با درنظرگرفتن این تعریف‌ها، الگوریتم تغییریافته K-means به‌صورت زیر پیشنهاد می‌شود:

۲-۳- الگوریتم

۱. ماتریس مجاورت از گراف اولیه به‌دست آید؛
۲. طول کوتاه‌ترین مسیر بین تمام جفت‌رأس‌ها با استفاده از الگوریتم فلوید-وارشال به‌دست آید؛
۳. از خروجی مرحله دو، ماتریس فاصله که نشان‌دهنده فاصله بین تمام جفت‌رأس‌ها است، ساخته می‌شود؛
۴. قطر گراف را به‌دست آورده و تمام مقادیر ماتریس فاصله را بر آن تقسیم کرده و یک منهای آن مقادیر را متناظراً در ماتریسی به اسم ماتریس تشابه ذخیره می‌کنیم. تشابه هر رأس با خودش یک می‌شود. تشابه دو رأس که سر و ته قطر هستند نیز صفر به‌دست می‌آید. این کار، هم با معنی، هم نسبی و هم نرمال است. نسبی‌بودن به این معنی که مقادیر، متناسب با گراف است و در گراف‌های متفاوت، مقادیر متفاوت به‌دست می‌آید.
۵. از روی تعداد k اولیه، تعدادی گره به‌عنوان مراکز خوشه‌های اولیه انتخاب شوند (تصادفی یا قطعی).
۶. هر موجودیت با توجه به میزان تشابه (از روی ماتریس تشابه) به مراکز خوشه، به خوشه با مرکز شبیه‌تر تخصیص یابد.
۷. مراکز خوشه به‌روزرسانی شوند. بر خلاف K-means معمولی، مرکز جدید خوشه را چنین تعریف می‌کنیم: رأسی از آن خوشه باشد که بیشترین مجموع تشابه با هم خوشه‌ای‌هایش را داشته باشد. اگر دو یا چند رأس چنین خاصیتی را داشته باشند، رأسی انتخاب شود که کمترین تشابه را به مجموع رأس‌های غیر هم‌خوشه داشته باشد. اگر باز هم چند رأس این خاصیت را داشته باشند، به‌دلخواه از بین آنها انتخاب شود.
۸. اگر با اعمال جابه‌جایی‌ها، شباهت‌ها بیشتر نشوند در این‌صورت شرط اتمام برقرار است و خوشه‌بندی پایان می‌یابد. در غیر این صورت برو به مرحله شش.

الگوریتم (۱) نیز به شکل ساده‌تری شبه‌کد الگوریتم بالا را نشان می‌دهد.

(الگوریتم-۱): شبه‌کد مربوط به الگوریتم GMA

(Algorithm-1): Pseudocode for the GMA algorithm

Algorithm 1

Input: Get the similarity matrix and *K*
Output: A clustered ADG

1. **Select *K* initial nodes (artifacts) randomly as the centers of the first clusters.**
2. **Each artifact, according to the similarity value (from the similarity matrix) assigns to the most similar cluster center (we have a partitioning on vertices as the current clustering).**
3. **Update cluster centers as follow:**
   For a cluster, the new center is the vertex that is the most similar to its other co-clusters, and for this purpose, the sum of the similarity of that vertex with its co-clusters has to be maximized in that clusters. If two or more vertices share such a property, a vertex is chosen that has the least sum of similarity to non-co-cluster vertices. If several more vertices still have the same value, one of them will be randomly selected.
4. **If there are no changes in clusters through three iterations, the clustering process ends and report the resulting clustering. Otherwise, go to step two.**

حال با یک مثال به تشریح الگوریتم پیشنهادی می‌پردازیم. ابتدا ماتریس فاصله را از روی ماتریس مجاورت (جدول ۲) ایجاد می‌کنیم. این ماتریس را D می‌نامیم.

با توجه به اینکه یال‌ها را بدون جهت در نظر می‌گیریم، ماتریس فاصله، ماتریسی متقارن خواهد بود و نیازی به ثبت مقادیر زیر قطر ماتریس (یا بالای قطر ماتریس) نیست. هر درایه مثل D_{ij} را بر مقدار قطر گراف تقسیم می‌کنیم و ۱ منهای آن مقدار را تشابه دو رأس تعریف می‌کنیم. ماتریسی هم‌اندازه با ماتریس فاصله به‌دست می‌آید. این ماتریس را ماتریس تشابه نامیده و آن را با S نشان می‌دهیم. تشابه هر رأس با خودش ۱ می‌شود. تشابه دو رأس که سر و ته قطر هستند نیز صفر به‌دست می‌آید.

(جدول-۳): ماتریس فاصله برای گراف شکل (۲)

(Table-3): Distance matrix for graph in figure (2)

	a	b	c	d	e	f	g	h
a	0	1	1	1	2	3	4	4
b		0	2	1	3	4	5	5
c			0	2	1	2	3	3
d				0	3	4	5	5
e					0	1	2	2
f						0	1	1
g							0	1
h								0

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]

[DOI: 10.52547/jsdp.18.4.37]

(جدول-۴): ماتریس تشابه برای جدول (۳)
(Table-4): Similarity matrix for Table (3)

	a	b	c	d	e	f	g	h
a	1	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{1}{5}$	$\frac{1}{5}$
b		1	$\frac{3}{5}$	$\frac{4}{5}$	$\frac{2}{5}$	$\frac{1}{5}$	0	0
c			1	$\frac{3}{5}$	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{2}{5}$
d				1	$\frac{2}{5}$	$\frac{1}{5}$	0	0
e					1	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{3}{5}$
f						1	$\frac{4}{5}$	$\frac{4}{5}$
g							1	$\frac{4}{5}$
h								1

به مقادیر قطر ماتریس در محاسبات نیازی نخواهیم داشت، ولی برای تأکید بر تشابه کامل هر رأس با خودش، در ماتریس لحاظ کردیم.

فرض می‌کنیم $k = 2$ و a و c مراکز خوشه‌های اولیه هستند. تشابه تک‌تک رأس‌ها را با این دو مرکز سنجیده می‌شود و مقدار تشابه هر کدام با هر مرکز، بیشتر بود به آن اختصاص می‌یابد. برای مثال f با a مقدار تشابهش $\frac{2}{5}$ است؛ ولی با c مقدار تشابهش $\frac{3}{5}$ شده است، پس به خوشه c تخصیص داده می‌شود.

دور نخست:

در دور اول با مراکز اولیه a وc افراز زیر به‌دست می‌آید:

Cluster 1: a – b, d
Cluster 2: c – e, f, g, h

مرکز خوشه را با خط تیره از بقیه مؤلفه‌های آن خوشه جدا کرده‌ایم.

دور دوم:

در هر دور باید مراکز جدید انتخاب کنیم. برای خوشه ۱، برای هر سه رأس، جمع مقادیر تشابه هر رأس با هم خوشه‌ای‌هایش مقدار $\frac{8}{5}$ به‌دست می‌آید، پس باید مقدار تشابه آن‌ها با غیر هم‌خوشه‌هایشان بررسی شود و هر کدام کمتر بود انتخاب شود.

$$a = \frac{4}{5} + \frac{3}{5} + \frac{2}{5} + \frac{1}{5} + \frac{1}{5} = \frac{11}{5}$$
$$b = \frac{3}{5} + \frac{2}{5} + \frac{1}{5} + \frac{0}{5} + \frac{0}{5} = \frac{6}{5} \quad \text{min}$$
$$d = \frac{3}{5} + \frac{2}{5} + \frac{1}{5} + \frac{0}{5} + \frac{0}{5} = \frac{6}{5} \quad \text{min}$$

با توجه به مقدار کمینه یکسان b و d، بنا به الگوریتم، یکی از آن دو به تصادف انتخاب می‌شوند. فرض می‌کنیم b انتخاب می‌شود (این دو مؤلفه در سامانه، فرق خاصی با هم ندارند). برای خوشه دوم مقادیر تشابه با هم‌خوشه‌ای‌ها برای تک‌تک رأس‌ها چنین به‌دست می‌آید:

$$c = \frac{4}{5} + \frac{3}{5} + \frac{2}{5} + \frac{2}{5}$$
$$e = \frac{4}{5} + \frac{4}{5} + \frac{3}{5} + \frac{3}{5}$$
$$f = \frac{4}{5} + \frac{4}{5} + \frac{4}{5} + \frac{3}{5} \quad \text{max}$$
$$g = \frac{4}{5} + \frac{4}{5} + \frac{3}{5} + \frac{2}{5}$$
$$h = \frac{4}{5} + \frac{4}{5} + \frac{3}{5} + \frac{2}{5}$$

پس f مرکز خوشه جدید خواهد بود.

تشابه تک‌تک رأس‌ها با دو مرکز جدید را به‌دست می‌آوریم و با توجه به بیشترین تشابه، رأس‌ها را به دو دسته زیر افراز می‌کنیم:

رأس c تشابه یکسانی با هر دو مرکز دارد، به‌طور تصادفی به یکی از خوشه‌ها تخصیص می‌دهیم. اگر به خوشه دوم تخصیص دهیم، خوشه‌بندی دور نخست تکرار می‌شود و دور دوم دوباره باید تکرار شود (لزوماً به تعادل نمی‌رسیم، بستگی به شرایط مسأله و arbitrary decision ها دارد). فرض می‌کنیم به خوشه نخست تخصیص یابد (در صورت رخداد تکراری بالا نیز، این حالت محتمل است).

Cluster 1: b – a, c, d
Cluster 2: f – e, g, f

دور سوم:

برای خوشه نخست، مرکز خوشه جدید با این محاسبات بدست می‌آید:

$$b = \frac{4}{5} + \frac{3}{5} + \frac{2}{5} = \frac{9}{5}$$
$$a = \frac{4}{5} + \frac{4}{5} + \frac{4}{5} = \frac{12}{5} \quad \text{max}$$
$$c = \frac{4}{5} + \frac{3}{5} + \frac{4}{5} = \frac{11}{5}$$
$$d = \frac{4}{5} + \frac{4}{5} + \frac{3}{5} = \frac{11}{5}$$

پس a به‌عنوان مرکز خوشه خود انتخاب می‌شود. برای خوشه دوم نیز محاسبات مشابهی داریم:

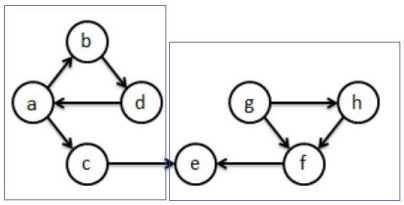
$$e = \frac{4}{5} + \frac{3}{5} + \frac{3}{5} = \frac{10}{5}$$
$$f = \frac{4}{5} + \frac{4}{5} + \frac{4}{5} = \frac{12}{5} \quad \text{max}$$
$$g = \frac{3}{5} + \frac{4}{5} + \frac{4}{5} = \frac{11}{5}$$
$$h = \frac{3}{5} + \frac{4}{5} + \frac{3}{5} = \frac{11}{5}$$

پس f نیز به‌عنوان مرکز خوشه دوم انتخاب می‌شود. خوشه‌بندی جدید برحسب تشابه با این دو خوشه به شرح زیر خواهد بود:

Cluster 1: a – b , c , d
Cluster 2: f – e , g , h

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

این حالت، پایدار هست و هر تکرار بعدی نیز همین نتیجه را دربر خواهد داشت. به‌طور شهودی نیز، نتیجه قابل قبول است. در شکل (۳) خوشه‌بندی نهایی نشان داده شده است.



(شکل-۲): خوشه‌بندی به‌دست‌آمده برای شکل (۲)
(Figure-3): Obtained clustering for figure (2)

۳-۳- تغییرات و نوآوری‌ها

۱. تعریف فاصله‌ای معقول در مهندسی نرم‌افزار
۲. استفاده از تشابه برای K-means به‌دلیل بامعنی‌بودن در مهندسی نرم‌افزار
۳. تعیین مرکز جدید خوشه‌ها به‌طوری‌که همیشه مرکز خوشه، یکی از رأس‌ها باشد و انتخاب، بامعنی باشد.

۴- مورد مطالعه

در این بخش، نرم‌افزار مورد استفاده در آزمایش‌ها و مقایسه‌ها توصیف می‌شود. ده پوشه با اندازه و عملکرد متفاوت از نرم‌افزار موزیلا فایرفاکس که یک مرورگر محبوب متن باز است، به‌عنوان نمونه مورد مطالعه، انتخاب شده است. این نرم‌افزار در پژوهش‌گران دیگران نیز به عنوان مورد مطالعه استفاده شده است [11 ، 16، 18، 20]. برای نشان‌دادن کارآیی روش پیشنهادی، الگوریتم را روی گراف وابستگی فراخوانی استخراج‌شده از آن ده پوشه اجرا کرده و نتایج به‌دست‌آمده را با نتایج حاصل از اجرای سایر الگوریتم‌ها روی این نرم‌افزار مقایسه خواهیم کرد. نسخه انتخاب‌شده در سال ۲۰۱۵ منتشر شده است. برای مقایسه، الگوریتم‌های Bunch, DAGC, ECA, MCA, NAHC و SHC انتخاب شده است. دلیل این انتخاب این است که این الگوریتم‌ها ویژگی‌های متفاوتی (جستجوی سراسری/محلی و تک/چندهدفه) نسبت به هم دارند. الگوریتم‌های ECA و MCA (الگوریتم ژنتیک، جستجوی سراسری، چندهدفه، درنظرگرفتن ویژگی‌های ساختاری و دارای کدینگ مبتنی بر مقدار)، الگوریتم Bunch (الگوریتم ژنتیک، جستجوی سراسری، تک‌هدفه، با درنظرگرفتن ویژگی‌های ساختاری و دارای کدینگ مبتنی بر مقدار)، الگوریتم DAGC (الگوریتم ژنتیک، جستجوی سراسری، تک‌هدفه، با درنظرگرفتن ویژگی‌های ساختاری و دارای کدینگ مبتنی بر جای‌گشت)، الگوریتم NAHC (الگوریتم تپه‌نوردی، جستجوی محلی، تک‌هدفه، با درنظرگرفتن ویژگی‌های ساختاری) و الگوریتم SHC (الگوریتم تپه‌نوردی، جستجوی محلی، تک‌هدفه، درنظرگرفتن ویژگی‌های معنایی) دارای ویژگی‌های متفاوتی هستند. جدول (۵) مشخصات ده پوشه انتخابی را که خوشه‌بندی خبرهٔ آنها نیز موجود است، نشان می‌دهد.

(جدول-۵): مشخصات پوشه‌های انتخاب‌شده
(Table-5): Properties of selected folders

	Folder Name	File Number	Number of Calls	Number of Clusters
1	ACCESSIBLE	179	293	9
2	BROWSER	45	45	8
3	BUILD	21	4	2
4	CONTENT	881	2948	13
5	DB	97	494	4
6	DOM	163	324	5
7	EXTENSIONS	179	206	13
8	GFX	342	644	8
9	INTL	573	957	7
10	IPC	393	59	4

۵- نتایج تجربی

در این مقاله، برای ارزیابی درستی نتایج روش پیشنهادی، از معیار معروف MoJo-FM برای مقایسهٔ خوشه‌بندی به‌دست‌آمده از این الگوریتم با خوشه‌بندی‌های به‌دست‌آمده از سایر روش‌ها استفاده خواهد شد. این معیار، معیاری خارجی است، بدین معنی که در نحوهٔ محاسبهٔ آن، از خوشه‌بندی خبره استفاده می‌شود که برای موزیلا فایرفاکس این خوشه‌بندی موجود است.

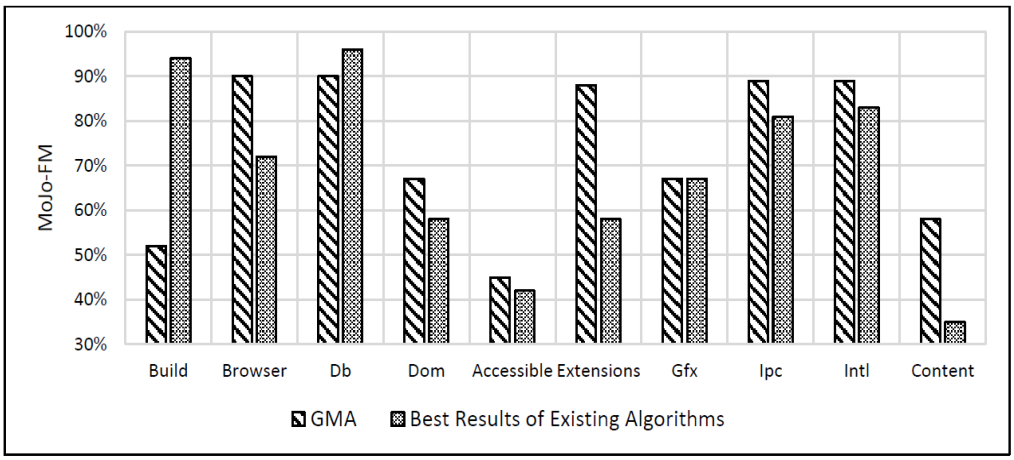
معیار MoJo-FM برای ارزیابی روش‌های خوشه‌بندی به کار می‌رود [14]. این معیار خارجی، رایج و بسیار پرکاربرد و در بسیاری از مقالات این زمینه استفاده شده است [15-20]. این معیار، خوشه‌بندی حاصل از یک الگوریتم خوشه‌بندی را با خوشه‌بندی فرد خبره مقایسه می‌کند. این معیار در رابطه (۱) نشان داده شده است.

$$\text{MoJoFM} = \left(1 - \frac{\text{mno}(A, B)}{\max\left(\text{mno}(\forall A, B)\right)}\right) * 100 \quad (۱)$$

که در آن mno(A, B) کمترین تعداد عمل‌گرهای جابه‌جایی یا ادغام برای تبدیل خوشه‌بندی A به B است.

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

(شکل-۴): مقایسه الگوریتم پیشنهادی با بهترین نتایج سایر روش‌ها
(Figure-4): Comparing the GMA with the best result of others



روال ارزیابی بدین ترتیب است که برای گراف نرم‌افزار موزیلا فایرفاکس که خوشه‌بندی خبرهٔ آن موجود است، این الگوریتم اجرا شده و خوشه‌بندی حاصل از اجرای آن، با معیار یادشده بررسی می‌شود. این معیار برای هر خوشه‌بندی یک عدد می‌دهد. مقدار حاصل را با مقدار حاصل از ارزیابی نتایج سایر روش‌ها مقایسه می‌کنیم. هر چه این مقدار بیشتر باشد، الگوریتم خوشه‌بندی مربوطه، کارکرد بهتری داشته است. در جدول (۶) مقدار MoJo-FM روش‌های یادشده خوشه‌بندی نرم‌افزار روی مجموعهٔ ده گراف نرم‌افزاری موزیلا فایرفاکس نشان داده شده است. در یک ستون نیز بهترین مقادیر مشاهده‌شده از اجرای الگوریتم پیشنهادی درج شده است. با توجه به اینکه الگوریتم پیشنهادی، الگوریتمی برای خوشه‌بندی و مبتنی بر گراف است، اسم آن را GMA[1] گذاشتیم. برای مقایسه بهتر، در شکل (۴) نتایج به‌دست‌آمده به‌وسیلهٔ الگوریتم پیشنهادی با بهترین نتایج به‌دست‌آمده به‌وسیلهٔ سایر روش‌ها نشان داده شده است.

۶- نتیجه‌گیری و کارهای آینده

ما در این مقاله با درنظرگرفتن خاصیت گراف‌های نرم‌افزاری، یک الگوریتم مبتنی بر افراز جدید برای خوشه‌بندی نرم‌افزار ارائه دادیم. با توجه به سادگی و سرعت آن، می‌توان از آن در خوشه‌بندی گراف‌های نرم‌افزاری بزرگ استفاده کرد. هم‌چنین از این الگوریتم می‌توان برای تقسیم‌کردن یک کد منبع به زیرکدها جهت توزیع روی سامانه‌های توزیع‌شده و یا موازی استفاده کرد. با پیاده‌سازی الگوریتم و اجرای آن روی گراف وابستگی نرم‌افزار موزیلا فایرفاکس و مقایسهٔ نتایج آن با سایر الگوریتم‌های مطرح می‌توان مشاهده کرد که این الگوریتم نتیجهٔ بهتری داشته است. کارهای آینده زیر روی این الگوریتم قابل تعریف است:

۱- درنظرگرفتن تعداد وابستگی (به‌عنوان مثال تعداد فراخوانی) در الگوریتم؛
۲- ارائه روشی برای تخمین k مناسب؛
۳- ارائه روشی به‌کمک تئوری گراف برای تخمین گره‌های اولیه مناسب؛
۴- برطرف‌کردن برخی حالات تصمیم‌گیری‌های اختیاری؛
۵- درنظرگرفتن جهت فراخوانی.

تقدیر و سپاس‌گزاری

نسخهٔ انگلیسی این مقاله در کنفرانس بین‌المللی محاسبات و سامانه‌های توزیع‌شده (DCHPC2018) پذیرفته و ارائه شده است. این مقاله، نسخهٔ گسترش‌یافته و ارتقاداده‌شدهٔ مقالهٔ ارائه‌شده در کنفرانس یادشده است، کلیه روش‌ها، الگوریتم‌ها، و نتایج، ارتقا داده شده‌اند.

۷- مراجع 7- References

[1] A. Isazadeh, H. Izadkhah, and I. Elgedawy, ''Source code modularization: theory and techniques'', Springer, 2017.

[2] B.S. Mitchell, S. and Mancoridis, ''On the automatic modularization of software systems using the bunch tool'', *IEEE Transactions on Software Engineering*, vol. 32(3), pp.193-208 , 2006.

[3] I. Candela, G. Bavota, B. Russo, and R. Oliveto, ''Using cohesion and coupling for software remodularization: Is it enough?'', *ACM Transactions on Software Engineering*

[1] Graph-based Modularization Algorithm

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]
[DOI: 10.52547/jsdp.18.4.37]

[15] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidović, and R. Kroeger, ‘’ Measuring the impact of code dependencies on software architecture recovery techniques’’, *IEEE Transactions on Software Engineering*, vol. 44(2), pp.159-181, 2017.

[16] N.S. Jalali, H. Izadkhah, and S. Lotfi, ‘’Multi-objective search-based software modularization: structural and non-structural features’’, Soft Computing, vol23(21), pp.11141-11165. 2019

[17] R. Naseem, O. Maqbool, and S. Muhammad, ‘’Cooperative clustering for software modularization’’, *Journal of Systems and Software*, vol. 86(8), pp.2045-2062, 2013.

[18] S. Mohammadi, and H. Izadkhah, ‘’A new algorithm for software clustering considering the knowledge of dependency between artifacts in the source code’’, *Information and Software Technology*, vol.105, pp.252-256. 2019.

[19] H. Sözer, ‘’Evaluating the Effectiveness of Multi-level Greedy Modularity Clustering for Software Architecture Recovery’’, *In European Conference on Software Architecture*, pp. 71-87, 2019.

[20] M. Kargar, A. Isazadeh, and H. Izadkhah, ‘’ Multi-programming language software systems modularization’’, *Computers & Electrical Engineering*, vol.80, pp.106-500, 2019.

بابک پوراصغر دانشجوی دکترای علوم کامپیوتر دانشگاه تبریز است. او مدرک کارشناسی خود را در رشتهٔ علوم کامپیوتر از دانشگاه تهران و مدرک کارشناسی ارشد خود را در همین رشته از دانشگاه صنعتی شریف دریافت کرده است. از زمینه‌های پژوهشی مورد علاقه او می‌توان به نظریه گراف، داده‌کاوی، مهندسی نرم‌افزار، خوشه‌بندی نرم‌افزار و نظریه محاسبه اشاره کرد.

نشانی رایانامه ایشان عبارت است از:

b.pourasghar@tabrizu.ac.ir

حبیب ایزدخواه استادیار گروه علوم کامپیوتر دانشگاه تبریز است. علاقه مندی‌های پژوهشی او شامل مهندسی نرم‌افزار، معماری نرم‌افزار، مهندسی معکوس و نرم‌افزارهی پویا است. هم اکنون او در استفاده از مهندسی معکوس و خوشه‌بندی

and Methodology (TOSEM), vol. 25(3), pp.1-28, 2016.

[4] G. Bavota, F. Carnevale, A. De Lucia, M. Di Penta, and R. Oliveto, ‘’Putting the developer in-the-loop: an interactive GA for software re-modularization’’, *In International Symposium on Search Based Software Engineering*, 2012, pp. 75-89.

[5] W. Mkaouer, M. Kessentini, A. Shaout, P. Koligheu, S. Bechikh, K. Deb, andA. Ouni, ‘’Many-objective software remodularization using NSGA-III’’, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol.24(3), pp.1-45, 2017.

[6] O. Maqbool, and H. Babri, ‘’Hierarchical clustering for software architecture recovery’’, *IEEE Transactions on Software Engineering*, vol. 33(11), pp.759-780, 2007.

[7] P. Andritsos, and V. Tzerpos, ‘’Information-theoretic software clustering’’, *IEEE Transactions on Software Engineering*, vol. 31(2), pp.150-165, 2007.

[8] M. Tajgardan, H. Izadkhah, and S. Lotfi, ‘’Software systems clustering using estimation of distribution approach,’’ *Journal of Applied Computer Science Methods*, vol.8(2), pp.99-113, 2016.

[9] H. Izadkhah, I. Elgedawy, A. Isazadeh, ‘’E-cdgm: an evolutionary call-dependency graph modularization approach for software systems’’, *Cybernetics and Information Technologies*, vol.16(3), pp.70-90, 2016.

[10] S. Parsa, and O. Bushehrian, ‘’ The design and implementation of a framework for automatic modularization of software systems’’, *The Journal of Supercomputing*, vol.32(1), pp.71-94, 2005.

[11] M. Kargar, A. Isazadeh, and H. Izadkhah, ‘’Semantic-based software clustering using hill climbing’’, *In 2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)* , pp. 55-60, IEEE, 2017, October.

[12] K. Praditwong, M. Harman, M. and X. Yao, ‘’Software module clustering as a multi-objective search problem’’, *IEEE Transactions on Software Engineering*, vol. 37(2), pp.264-282, 2010.

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, ‘’Introduction to algorithms’’, MIT press, 2009.

[14] Z. Wen, and V. Tzerpos, ’’ An effectiveness measure for software clustering algorithms’’, *In Proceedings. 12th IEEE International Workshop on Program Comprehension*, 2004. pp. 194-203, IEEE.

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]

[DOI: 10.52547/jsdp.18.4.37]

برای استخراج فهم نرم‌افزارهای بزرگ و چندزبانه کار می‌کند. او در پروژه‌های پژوهشی مختلف شرکت کرده است. او یکی از نویسندگان کتاب Source Code Modularization: Theory and Techniques است که به‌وسیلهٔ انتشارات Springer چاپ شده است. دکتر ایزدخواه هم اکنون یکی از سردبیران مجله Iran Journal of Computer Science است که به‌وسیلهٔ انتشارات Springer چاپ می‌شود.

نشانی رایانامه ایشان عبارت است از:

izadkhah@tabrizu.ac.ir

شهریار لطفی دانشیار گروه علوم کامپیوتر دانشگاه تبریز است. نامبرده، تحصیلات خود را در مقطع کارشناسی و کارشناسی ارشد در رشتهٔ مهندسی کامپیوتر-نرم‌افزار در دانشگاه اصفهان گذرانده و دکترای خود را در همین رشته از دانشگاه علم و صنعت ایران دریافت کرده است. زمینه‌های پژوهشی مورد علاقه ایشان شامل کامپایلرها، سوپر کامپایلرها، الگوریتم‌های موازی، محاسبات تکاملی و الگوریتم‌ها است.

نشانی رایانامه ایشان عبارت است از:

shahriar_lotfi@tabrizu.ac.ir

خیام صالحی استادیار گروه علوم کامپیوتر در دانشگاه شهرکرد است. نامبرده، تحصیلات خود را در رشته علوم کامپیوتر در مقطع کارشناسی و کارشناسی ارشد در دانشگاه‌های یزد و صنعتی شریف گذرانده و دکترای خود را در همین رشته از دانشگاه تبریز اخذ کرده است. از زمینه‌های پژوهشی مورد علاقه ایشان می‌توان به روش‌های صوری، یادگیری ماشین و جریان اطلاعات امن در نرم‌افزار اشاره کرد.

نشانی رایانامه ایشان عبارت است از:

kh.salehi@sku.ac.ir

[Downloaded from jsdp.rcisp.ac.ir on 2026-01-02]

[DOI: 10.52547/jsdp.18.4.37]