ساختاری جدید برای سازمان‌دهی و ذخیره‌سازی داده‌ها در گراف‌ها

بهروز کوهستانی[۱]، استادیار

۱- دانشکده مهندسی فناوری‌های نوین - دانشگاه تبریز - تبریز - ایران – b.koohestani@tabizu.ac.ir

چکیده: مسائل بهینه‌سازی که با ساختارهای مبتنی بر گراف سر و کار دارند بخش بزرگی از مسائل بهینه‌سازی در فیلدهای مختلف را به خود اختصاص می‌دهند. امروزه برای مواجهه با چنین مسائلی، الگوریتم‌های جستجو از بهترین گزینه‌ها محسوب می‌شوند. بدین منظور، عملیاتی که اغلب مورد نیاز هستند عبارتند از تعویض پی در پی برچسب گره‌های یک گراف با یکدیگر با استفاده از یک استراتژی مناسب و سپس ارزیابی اثر هر تعویض روی کمیت تحت بررسی. مشکل عمده‌ای که برای انجام عملیات مذکور وجود دارد زمان اجرای بسیار زیاد خصوصاً برای گراف‌های بزرگ است. این طبیعتاً می‌تواند دشواری‌های بسیاری را در به‌کارگیری الگوریتم‌های جستجو برای حل مسائل دنیای واقعی که مدل گراف تئوریکی آن‌ها عموماً بسیار پیچیده بوده و اندازه بزرگی دارند به وجود آورد. با هدف حل مشکل مذکور، در این تحقیق ساختاری جدید برای سازمان‌دهی و ذخیره‌سازی داده‌ها در گراف‌ها ارائه می‌شود. نتایج آزمایش‌های عددی نشان می‌دهد که ساختار پیشنهادی بسیار مؤثر است.

واژه‌های کلیدی: تئوری گراف، بهینه‌سازی، ساختمان داده‌ها، الگوریتم‌های جستجو.

A New Structure for Organizing and Storing Data in Graphs

B. Koohestani[1], Assistant Professor

1- School of Engineering-Emerging Technologies, University of Tabriz, Tabriz, Iran, Email: b.koohestani@tabizu.ac.ir

Abstract: Optimization problems related to graph-based structures comprise a large proportion of optimization problems appearing in different fields. At present, search algorithms are among the best choices for dealing with such problems. For this purpose, operations which are often needed include successive swapping the vertex labels of a given graph using an appropriate strategy and evaluating the effect of each swap on the quantity under investigation. A major problem for performing the above-mentioned operations is an immense amount of runtime required, especially for large graphs. Obviously, this can present serious problems in the use of search algorithms for addressing real-world problems which usually have complex graph theoretical models and large sizes. In this research, a new structure for organizing and storing data in graphs is proposed with the aim of resolving the problem described above. The results of numerical experiments reveal that the proposed structure is very effective.

Keywords: Graph theory, optimization, data structures, search algorithms.

تاریخ ارسال مقاله: ۱۳۹۵/۱۲/۲۴
تاریخ اصلاح مقاله: ۱۳۹۶/۰۷/۰۵ و ۱۳۹۶/۰۷/۱۷
تاریخ پذیرش مقاله: ۱۳۹۶/۰۸/۲۶
نام نویسنده مسئول: بهروز کوهستانی
نشانی نویسنده مسئول: ایران - تبریز - بلوار ۲۹ بهمن - دانشگاه تبریز - دانشکده مهندسی فناوری‌های نوین.

۱- مقدمه

تئوری گراف[۱] شاخه‌ای از ریاضیات است که به مطالعه گراف‌ها می‌پردازد. یک گراف در واقع ساختاری است متشکل از مجموعه‌ای از نقاط و خطوطی که زوج‌های معینی از این نقاط را به هم وصل می‌کند [۱]. بسیاری از وضعیت‌های دنیای واقعی را می‌توان به‌راحتی با چنین ساختارهایی توصیف کرد. لازم به ذکر است که در طول سی سال گذشته، تئوری گراف به‌طور گسترده‌ای به‌عنوان یک ابزار ریاضی قدرتمند برای مدل‌سازی و حل مسائل مختلف در زمینه‌های گوناگون از قبیل علوم پایه، علوم کامپیوتر، علوم پزشکی، فنی و مهندسی، کشاورزی و حتی علوم انسانی و اجتماعی مورد استفاده قرار گرفته است [۲].

ذخیره‌سازی و سازمان‌دهی داده در ساختارهای مبتنی بر گراف از طریق ساختمان داده‌های[۲] متعددی که بدین منظور طراحی شده‌اند صورت می‌پذیرد. هرکدام از ساختمان داده‌های موجود مزایا و معایب خاص خودشان را دارند و عموماً برای انجام عمل یا عملیات خاصی روی گراف‌ها مناسب‌تر هستند. مهم‌ترین و پرکاربردترین ساختمان داده‌هایی که برای گراف ارائه شده‌اند؛ عبارتند از ماتریس مجاورت[۳]، لیست مجاورت[۴]، لیست یال[۵] و ماتریس وقوع[۶] [۳]. نسخه‌های دیگری از این چهار ساختمان داده با تغییراتی نه چندان قابل توجه نسبت به نسخه‌های اصلی نیز برای اهداف مختلف طراحی شده‌اند که در مراجع [۳-۷] به‌تفصیل به آن‌ها پرداخته شده است.

همچنین تاکنون انواع مختلف سیستم‌های تخصصی ذخیره‌سازی و سازمان‌دهی داده در گراف‌ها برای کاربردهای خاص نیز طراحی و پیاده‌سازی شده‌اند. به‌عنوان مثال: در مرجع [۸] ساختمان داده‌هایی برای گراف‌های بزرگ که در آن‌ها اطلاعات گره‌ها و یال‌ها با زمان تغییر می‌کنند، ارائه می‌شود. در [۹] یک ساختمان داده برای گراف با هدف کاربرد آن در نهان‌نگاری یا استگانوگرافی معرفی می‌شود. در [۱۰] یک ساختمان داده برای توصیف گراف‌ها به‌منظور مدل‌سازی مدارهای الکترونیکی خیلی پیچیده در یک مجموعه تراشه پیشنهاد می‌شود. در [۱۱] ساختمان داده‌ای برای ذخیره‌سازی گراف‌های بدون مقیاس با هدف استفاده از آن در شبکه‌های مختلف ارائه می‌شود. در [۱۲] یک ساختمان داده که به‌صورت پویا اطلاعات مرتبط با گره‌های متصل در یک گراف را نگهداری می‌کند، معرفی می‌شود.

مسائل بهینه‌سازی[۷] که با ساختارهای مبتنی بر گراف سر و کار دارند (به‌عنوان یک مثال بارز، مسائل طرح‌بندی گراف) بخش بزرگی از مسائل بهینه‌سازی در فیلدهای مختلف را به خود اختصاص می‌دهند [۱۳]. امروزه برای مواجهه با چنین مسائلی، الگوریتم‌های جستجو از بهترین گزینه‌ها محسوب می‌شوند. بدین منظور، عملیاتی که اغلب مورد نیاز هستند عبارتند از تعویض پی در پی برچسب گره‌های یک گراف با یکدیگر با استفاده از یک استراتژی مناسب و سپس ارزیابی اثر هر تعویض روی کمیت تحت بررسی [۱۴].

مشکل عمده‌ای که برای انجام عملیات مذکور وجود دارد، زمان اجرای بسیار زیاد خصوصاً برای گراف‌های بزرگ است که این خود ناشی از افزایش بسیار قابل توجه در تعداد عملیات اضافی که برای انجام عملیات اصلی ضروری هستند، با افزایش یافتن اندازه گراف می‌باشد. این طبیعتاً می‌تواند دشواری‌های بسیاری را در به‌کارگیری الگوریتم‌های جستجو[۸] برای حل مسائل دنیای واقعی که مدل گراف تئوریکی آن‌ها عموماً بسیار پیچیده بوده و اندازه بزرگی دارند به وجود آورده و امکان رسیدن به جواب در یک بازه زمانی معقول را حتی با استفاده از کامپیوترهای پرقدرت غیرممکن سازد [۱۴-۱۸]. در چنین مواردی کاهش فضای جستجو می‌تواند راهکاری مناسب باشد که در این ارتباط دو رویکرد مختلف در [۱۹، ۲۰] ارائه شده‌اند.

یکی از عواملی که می‌تواند در حل مشکلات مذکور نقش کلیدی ایفا کند ساختمان داده مورد استفاده برای گراف است. این در حالی است که ساختمان داده‌های موجود که برای توصیف گراف‌ها مورد استفاده قرار می‌گیرند از کارایی مناسبی در این خصوص برخوردار نیستند [۲۱-۲۳]. با توجه به مطالب فوق، وجود ساختاری برای انجام عملیات مذکور بر روی گراف‌ها به‌طوری که در آن تعداد عملیات مورد نیاز وابسته به اندازه گراف نباشد یا اینکه تعداد این عملیات مورد نیاز با افزایش اندازه گراف به‌کندی رشد کند، ممکن است بتواند کارایی الگوریتم‌های جستجوی اعمال‌شده به مسائل بهینه‌سازی مرتبط با ساختارهای گراف را به‌شدت افزایش بدهد.

در این تحقیق، ساختاری جدید که برای استفاده در چنین شرایطی بسیار مناسب است؛ ارائه شده و عملکرد آن در برابر ساختمان داده معروف و کارآمد لیست مجاورت مورد ارزیابی قرار می‌گیرد. نتایج آزمایش‌های عددی انجام‌شده نشان می‌دهد که ساختار پیشنهادی به‌طور قابل ملاحظه‌ای در افزایش کارایی الگوریتم‌های جستجو برای مواجهه با مسائل مورد بحث مؤثر است. ساختار این مقاله به‌شرح زیر است:

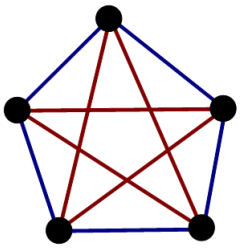
در بخش ۲، گراف و ساختمان داده‌های مهم و پرکاربرد گراف معرفی می‌شوند. در بخش ۳، مسئله مورد بحث در این تحقیق بیان می‌شود. در بخش ۴، روش پیشنهادی ارائه می‌شود. در بخش ۵، آزمایش‌های عددی انجام‌یافته مورد بررسی قرار می‌گیرند. در نهایت در بخش ۶، نتیجه‌گیری ارائه می‌شود.

۲- آشنایی با گراف و ساختمان داده‌های مرتبط

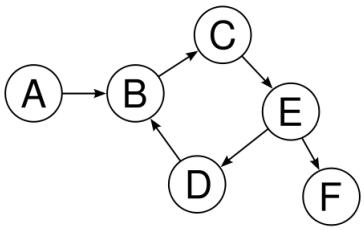
تئوری گراف شاخه‌ای از ریاضیات است که به مطالعه گراف‌ها می‌پردازد. اویلر در سال ۱۷۳۶ با حل مسئله پل‌های کونیگسبرگ نظریه گراف‌ها را بنیان گذاشت؛ اما جیمز جوزف سیلوستر نخستین کسی بود که در سال ۱۸۷۸ از واژه گراف برای نامیدن این مدل‌های ریاضی استفاده کرد [۲۴].

یک گراف مجموعه‌ای از رأس‌ها است که توسط مجموعه‌ای از زوج‌های مرتب که همان یال‌ها هستند به هم متصل شده‌اند. با استفاده

از نمادهای ریاضی یک گراف را به‌صورت $G = (V, E)$ نشان می‌دهند که در آن V مجموعه رأس‌ها و E مجموعه یال‌ها می‌باشند. یال‌ها می‌توانند به‌صورت ساده یا جهت‌دار باشند که هرکدام در جای خود کاربردهای بسیاری دارد. به‌عنوان مثال، اگر صرفاً اتصال دو شهر روی نقشه از طریق یک آزادراه مد نظر باشد، کافی است آن دو شهر را با دو نقطه و آزادراه مزبور را با یالی ساده نمایش داد؛ اما اگر بین دو شهر جاده‌ای یک‌طرفه وجود داشته باشد آنگاه باید با قرار دادن یالی جهت‌دار مسیر حرکت را در آن جاده مشخص کرد. همچنین برای اینکه فاصله بین دو شهر را توسط چنین گرافی نشان دهیم، می‌توانیم از گراف وزن‌دار استفاده نموده و مسافت بین شهرها را با یک عدد بر روی هر یال نشان دهیم [۲۴]. نمونه‌هایی از گراف‌های ساده و جهت‌دار در شکل‌های ۱ و ۲ نشان داده شده‌اند.



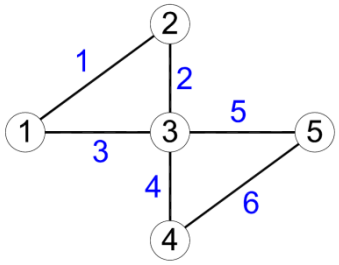
شکل ۱: یک گراف ساده بدون جهت با ۵ رأس و ۱۰ یال.



شکل ۲: یک گراف جهت‌دار با ۶ رأس و ۶ یال.

بسیاری از وضعیت‌های دنیای واقعی را می‌توان به‌راحتی با چنین ساختارهایی توصیف کرد. لازم به ذکر است که در طول سی سال گذشته، تئوری گراف به‌طور گسترده‌ای به‌عنوان یک ابزار ریاضی قدرتمند برای مدل‌سازی و حل مسائل مختلف در زمینه‌های گوناگون از قبیل علوم پایه، علوم کامپیوتر، علوم پزشکی، فنی و مهندسی، کشاورزی و حتی علوم انسانی و اجتماعی مورد استفاده قرار گرفته است. ساختمان داده‌های متعددی برای توصیف گراف‌ها ارائه شده‌اند که مهم‌ترین آن‌ها عبارتند از ماتریس مجاورت، لیست مجاورت، لیست یال و ماتریس وقوع که در ادامه به آن‌ها خواهیم پرداخت.

ماتریس مجاورت [۳] یک ماتریس مربعی است که تعداد سطر و ستون آن با تعداد رأس‌های گراف برابر است. این ماتریس متقارن بوده، فقط از اعداد یک یا صفر تشکیل شده است و درایه‌های واقع بر قطر اصلی آن فقط شامل صفر می‌باشد. اندیس هر سطر یا ستون از ماتریس مجاورت به یکی از رئوس گراف اشاره می‌کند. اگر یک رأس از گراف که با اندیس سطر مشخص می‌شود، به یک رأس یا رئوس دیگر گراف متصل باشد، درایه‌های مرتبط با آن رأس یا رئوس که با استفاده از اندیس ستون یا ستون‌های مربوطه مشخص می‌شوند، شامل عدد یک و در غیر این‌صورت شامل عدد صفر خواهند بود. در واقع تعداد یک‌های هر سطر نشان‌دهنده درجه رأس مربوطه می‌باشد. ماتریس مجاورت هرچند بسیار پرکاربرد و مفید است ولی به‌علت اینکه در ساختار آن درایه‌های صفر زیادی وجود دارد (بخصوص در ماتریس‌های اسپارس) باعث هدر رفتن فضای ذخیره‌سازی (حافظه) می‌گردد. درایه‌های صفر زیاد همچنین موجب افزایش تعداد عملیات مورد نیاز مرتبط با گراف (به‌عنوان مثال، اضافه کردن به رأس، حذف یک رأس، جستجو و ...) می‌گردد. طبیعتاً، موارد مذکور با بزرگ‌تر شدن اندازه گراف هر چه بیشتر باعث کاهش کارایی این ساختمان داده می‌شود. شکل ۳ یک گراف ساده بدون جهت و شکل ۴ ماتریس مجاورت آن گراف را نشان می‌دهد.



شکل ۳: یک گراف ساده بدون جهت با ۵ رأس و ۶ یال.

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
3	1	1	0	1	1
4	0	0	1	0	1
5	0	0	1	1	0

شکل ۴: ماتریس مجاورت گراف ارائه‌شده در شکل ۳.

ماتریس وقوع [۳] یک ماتریس مستطیلی است که در حقیقت قرار گرفتن رئوس بر روی یال‌ها را به‌طور مستقیم مشخص می‌کند. در ماتریس وقوع اندیس هر سطر به یکی از رئوس گراف اشاره می‌کند در حالی که اندیس هر ستون به یکی از یال‌های گراف اشاره می‌کند. درایه‌های این ماتریس نیز همانند ماتریس مجاورت از صفر یا یک تشکیل شده است. اگر یک رأس روی یک یال قرار گرفته باشد، درایه متناظر شامل عدد یک و در غیر این‌صورت شامل عدد صفر خواهد بود. در ماتریس وقوع همواره مجموع اعداد هر ستون ۲ می‌باشد زیرا هر ستون معرف یک یال بوده و هر یال تنها دو سر دارد. این ساختمان داده نیز همانند ماتریس مجاورت با مشکلات درایه‌های خالی زیاد دست به گریبان است و لذا در آن اتلاف حافظه و اتلاف زمان در انجام عملیات مختلف به‌وضوح دیده می‌شود با این تفاوت که در مقایسه با

ماتریس مجاورت، اتلاف حافظه کمتر است ولی اگر گراف متراکم باشد، عموماً اتلاف زمان در اکثر عملیات مورد نیاز بیشتر است. شکل ۵ ماتریس وقوع گراف ترسیم‌شده در شکل ۳ را نشان می‌دهد.

	1	2	3	4	5	6
1	1	0	1	0	0	0
2	1	1	0	0	0	0
3	0	1	1	1	1	0
4	0	0	0	1	0	1
5	0	0	0	0	1	1

شکل ۵: ماتریس وقوع گراف ارائه‌شده در شکل ۳.

لیست یال [۳] که یکی از ساده‌ترین ساختمان داده‌های موجود برای توصیف یک گراف است، در واقع یک ماتریس است که فقط دو ستون دارد و تعداد سطرهای آن نیز برابر با تعداد یال‌های گراف است. در هر سطر شماره دو رأس که یالی بین آن‌ها وجود دارد در دو ستون مرتبط با آن سطر درج می‌گردد. لازم به ذکر است که لیست یال ممکن است به‌صورت مرتب یا نامرتب باشد. با استفاده از این ساختار می‌توان صرفه‌جویی قابل ملاحظه‌ای در حافظه انجام داد ولی در مقایسه با ماتریس مجاورت، عملیاتی چون مشخص کردن اینکه دو رأس مجاور هستند یا نه و همچنین حذف یک یال مشخص بسیار دشوارتر است. پیاده‌سازی این ساختمان داده با لیست‌های پیوندی نیز امکان‌پذیر است. شکل ۶ لیست یال گراف ترسیم‌شده در شکل ۳ را نشان می‌دهد.

	1	2
1	1	2
2	2	3
3	3	1
4	4	3
5	3	5
6	4	5

شکل ۶: لیست یال گراف ارائه‌شده در شکل ۳.

لیست مجاورت [۳] ساختاری است که در آن هر رأس گراف لیستی از رئوس متصل به آن رأس را در اختیار دارد. برای پیاده‌سازی لیست مجاورت می‌توان از لیست‌های پیوندی، آرایه‌ای از بردارها و آرایه‌ای دوبعدی استفاده کرد. استفاده از لیست‌های پیوندی می‌تواند انجام اکثر عملیات مورد نیاز روی گراف را تسهیل نماید ولی از طرف دیگر می‌تواند موجب افزایش مصرف حافظه یا افزایش زمان اجرا بشود. استفاده از آرایه‌ای از بردارها مشکل مصرف حافظه زیاد را از بین می‌برد و پیاده‌سازی را راحت‌تر می‌کند ولی به‌شدت موجب کندی اجرای برنامه می‌شود. در صورتی که تعداد کل یال‌های گراف معلوم باشد یا بتوان تخمین نسبتاً دقیقی از آن را در اختیار داشت، استفاده از آرایه‌ای دوبعدی گزینه بسیار مناسبی است چرا که مشکل مصرف حافظه زیاد و کندی اجرای برنامه تا حد زیادی مرتفع می‌گردد. در مقایسه با سایر ساختمان داده‌ها، پیدا کردن رئوس مجاور یک رأس معین در لیست مجاورت بسیار ساده است. در این ساختار همچنین، مشخص کردن اینکه دو رأس مجاور هستند یا نه نیازمند جستجو در لیست یال‌های مرتبط با آن رئوس می‌باشد. شکل ۷ لیست مجاورت گراف ترسیم‌شده در شکل ۳ را با استفاده از آرایه‌ای دوبعدی نشان می‌دهد.

	1	2	3	4
1	2	3	0	0
2	1	3	0	0
3	1	2	4	5
4	3	5	0	0
5	3	4	0	0

شکل ۷: لیست مجاورت گراف ارائه‌شده در شکل ۳.

ملاک‌های ارزیابی عملکرد ساختمان داده‌های مرتبط با یک گراف متعدد هستند. در این بخش به چند مورد از این ملاک‌ها به‌اختصار اشاره شد ولی به‌طور کلی می‌توان گفت که قضاوت در مورد کارایی یک ساختمان داده تا حد بسیار زیادی بستگی به جایی دارد که قرار است از آن استفاده بشود. به‌عبارت دیگر، یک ساختار ممکن است برای یک کاربرد خاص عملکرد خوبی از خود نشان بدهد در حالی که برای کاربرد دیگری نتواند عملکرد مناسبی داشته باشد. با وجود این، با در نظر گرفتن همه جوانب می‌توان گفت که عموماً در بیشتر موارد کارایی لیست مجاورت نسبت به سه ساختار دیگر بیشتر است [۳].

۳- بیان مسئله

در بخش گذشته مهم‌ترین و پرکاربردترین ساختمان داده‌های موجود برای توصیف یک گراف معرفی شدند و همان‌طور که ذکر شد هرکدام از این ساختمان داده‌ها مزایا و معایب خاص خودشان را دارند و عموماً برای انجام عمل یا عملیات خاصی روی گراف‌ها مناسب‌تر هستند. به‌طور کلی می‌توان گفت که چهار ساختار مذکور کم و بیش می‌توانند در اکثر موارد مفید بوده و نیازهای ما را برطرف سازند اما بدون شک موارد استثنایی هم وجود دارند.

یکی از مهم‌ترین موارد استثنا که می‌توان به آن اشاره کرد مواجهه با مسائل بهینه‌سازی مرتبط با گراف‌ها از طریق الگوریتم‌های جستجو

است. در این حالت معمولاً با استفاده از یک استراتژی متناسب با مسئله مورد نظر ابتدا برچسب یک گره از گراف با برچسب گره دیگر از همان گراف (یا برچسب چند گره از گراف با برچسب چند گره دیگر از همان گراف) تعویض شده و سپس اثر این تعویض روی کمیت تحت بررسی مورد ارزیابی قرار می‌گیرد و این فرآیند عموماً به تعداد بسیار زیاد تکرار می‌شود.

حال فرض کنید که از ساختمان داده لیست یال استفاده شود. در این‌صورت برای تعویض برچسب دو گره فرضی در گراف، کل لیست یال‌ها باید مورد جستجو قرار بگیرد تا گره‌های مجاور با دو گره انتخاب‌شده برای تعویض پیدا شوند و این عملیات باید بارها و بارها تکرار شوند که این طبیعتاً بسیار هزینه‌بر خواهد بود.

اگر از ساختمان داده لیست مجاورت استفاده شود، برای تعویض برچسب دو گره در گراف، همه لیست‌هایی که شامل هرکدام از گره‌های مجاور با دو گره انتخاب‌شده برای تعویض هستند باید پویش بشوند و در نهایت دو لیست مرتبط با گره‌های انتخاب‌شده نیز باید با هم تعویض بشوند و طبیعتاً مثل حالت قبل این عملیات نیز باید بارها و بارها تکرار شوند. در این حالت وضعیت به مراتب بهتر از حالت اول است ولی باز هم هزینه انجام عملیات بسیار بالا خواهد بود بخصوص برای گراف‌های متراکم.

در صورتی که ساختمان داده ماتریس مجاورت بکار گرفته شود، تعویض برچسب دو گره در گراف شبیه به لیست مجاورت است با این تفاوت که به‌علت وجود درایه‌های صفر زیاد در ساختار آن، فضای ذخیره‌سازی شدیداً به هدر می‌رود و همچنین موجب افزایش تعداد عملیات مورد نیاز برای تعویض برچسب‌ها در مقایسه با لیست مجاورت می‌شود.

همان‌طور که ملاحظه می‌شود، ساختمان داده‌های مذکور که در حقیقت پرکاربردترین و کارآمدترین ساختمان داده‌های موجود به حساب می‌آیند برای انجام عملیات مورد بحث از کارایی مناسبی برخوردار نیستند. دلیل اصلی آن هم عملیات اضافی زیادی است که برای انجام عملیات اصلی باید انجام بگیرد که با افزایش اندازه گراف این عملیات اضافی مورد نیاز نیز با شیب تندی افزایش پیدا می‌کند. هدف اصلی از این تحقیق، یافتن راه‌حلی برای مهار مشکل مذکور از طریق ارائه ساختاری جدید برای ذخیره‌سازی و سازمان‌دهی داده‌ها در گراف‌ها می‌باشد.

۴- ساختار پیشنهادی

برای رفع مشکلاتی که در بخش قبل به آن‌ها اشاره شد، وجود یک ساختمان داده جدید برای گراف ضروری است که با به‌کارگیری آن تا حد امکان عملیات غیرضروری حذف شود و همچنین تعداد عملیات مورد نیاز وابسته به اندازه گراف نباشد یا اینکه تعداد این عملیات مورد نیاز با افزایش اندازه گراف خیلی به‌کندی رشد کند. در ادامه، ساختار پیشنهادی به‌طور کامل معرفی می‌شود.

در گام اول یک ماتریس که آن را با **G** نشان می‌دهیم ایجاد می‌کنیم که یک ماتریس مستطیلی به ابعاد $n \times (2+d)$ است و در آن n تعداد کل رئوس گراف و d حداکثر درجه رأس‌های موجود در گراف مورد نظر می‌باشد. هدف از ایجاد این ماتریس ذخیره کردن آدرس هر درایه در لیست یال مرتبط با گراف به صورتی فشرده است. هر سطر از ماتریس **G** شامل دو مجموعه از اعداد است که هر مجموعه با یکی از ستون‌های لیست یال گراف مرتبط است. به عبارت دقیق‌تر ، سطر i از ماتریس **G** با رأس i مطابقت می‌کند و به ترتیب شامل یک عدد (f_c)، یک مجموعه اعداد (S_1)، یک عدد (s_c) و یک مجموعه اعداد (S_2) می‌باشد. متغیرها و مجموعه‌های مذکور به‌صورت زیر تعریف می‌شوند:

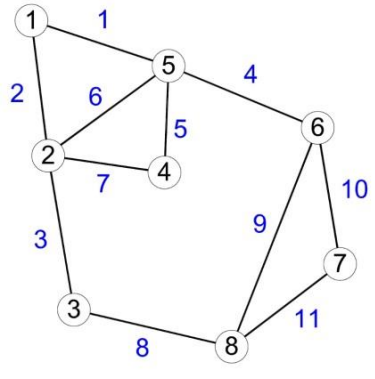
f_c برابر است با تعداد کل دفعات مشاهده رأس i در ستون اول لیست یال گراف.

S_1 مجموعه‌ای است شامل اندیس سطرهایی از ستون اول لیست یال گراف که درایه‌ای برابر با i دارند.

s_c برابر است با تعداد کل دفعات مشاهده رأس i در ستون دوم لیست یال گراف.

S_2 مجموعه‌ای است شامل اندیس سطرهایی از ستون دوم لیست یال گراف که درایه‌ای برابر با i دارند.

به عبارت بهتر، در سطر i از ماتریس G، ابتدا عدد f_c در ستون اول درج می‌شود، سپس اندیس‌های موجود در مجموعه S_1 به ترتیب در ستون‌های دوم، سوم و غیره درج می‌شوند. بلافاصله پس از درج آخرین اندیسی که در S_1 وجود دارد عدد s_c درج می‌شود و در نهایت پس از آن اندیس‌های موجود در مجموعه S_2 به ترتیب درج می‌شوند. برای درک بهتر مطلب ، در اینجا یک مثال بسیار ساده ارائه می‌شود. شکل ۹، یک گراف بدون جهت با ۸ رأس و ۱۱ یال را به همراه لیست یال مرتبط با این گراف نشان می‌دهد. شکل ۱۰ نیز ماتریس **G** مرتبط با گراف ترسیم‌شده در شکل ۹ را نشان می‌دهد.



۱	۵
۱	۲
۲	۳
۵	۶
۴	۵
۲	۵
۲	۴
۸	۳
۶	۸
۶	۷
۷	۸

شکل ۹: یک گراف بدون جهت با ۸ رأس و ۱۱ یال به همراه لیست یال مرتبط با این گراف.

$$\mathbf{G}=\begin{bmatrix} ۲ & ۱ & ۲ & ۰ & ۰ & ۰ \\ ۳ & ۳ & ۶ & ۷ & ۱ & ۲ \\ ۰ & ۲ & ۳ & ۸ & ۰ & ۰ \\ ۱ & ۵ & ۱ & ۷ & ۰ & ۰ \\ ۱ & ۴ & ۳ & ۱ & ۵ & ۶ \\ ۲ & ۹ & ۱۰ & ۱ & ۴ & ۰ \\ ۱ & ۱۱ & ۱ & ۱۰ & ۰ & ۰ \\ ۱ & ۸ & ۲ & ۹ & ۱۱ & ۰ \end{bmatrix}$$

شکل ۱۰: ماتریس G مرتبط با گراف ترسیم‌شده در شکل ۹.

در این مثال، n تعداد کل رئوس گراف برابر است با ۸ و d حداکثر درجه رأس‌های موجود در گراف برابر است با ۴. در نتیجه ماتریس G به‌صورت $(۴ + ۲) \times ۸$ یا به عبارت بهتر ۶ × ۸ (۸ سطر و ۶ ستون) است. حال فرض کنید که سطر دوم ماتریس **G** که در واقع به رأسی از گراف با برچسب شماره ۲ اشاره می‌کند، مد نظر ما باشد. در این‌صورت:

$$f_c = 3$$
$$S_1 = \{3, 6, 7\}$$
$$S_2 = \{2\} \quad ، \quad s_c = 1$$

جدول ۱ این مقادیر را برای تمام سطرهای ماتریس **G** و جدول ۲ اندیس ستون‌های مرتبط را نشان می‌دهند. لازم به ذکر است که اگر مقادیر f_c و s_c برابر با صفر باشند در این‌صورت مجموعه‌های S_1 و S_2 فاقد عضو یا به عبارت بهتر تهی خواهند بود.

جدول ۱: مقادیر f_c، S_1، s_c و S_2 برای تمام سطرهای ماتریس G.

شماره رأس	f_c	S_1	s_c	S_2
۱	۲	{۱ , ۲}	۰	{Ø}
۲	۳	{۳ , ۶ , ۷}	۱	{۲}
۳	۰	{Ø}	۲	{۳ , ۸}
۴	۱	{۵}	۱	{۷}
۵	۱	{۴}	۳	{۱ , ۵ , ۶}
۶	۲	{۹ , ۱۰}	۱	{۴}
۷	۱	{۱۱}	۱	{۱۰}
۸	۱	{۸}	۲	{۹ , ۱۱}

نکته‌ای که باید به آن توجه شود این است که ماتریس G به‌سادگی و فقط با یک‌بار پویش کردن لیست یال گراف ایجاد می‌شود. همچنین، تعویض برچسب‌های دو رأس می‌تواند به‌سادگی و فقط با تعویض سطرهای متناظر آن دو رأس در ماتریس G بدون تحت تأثیر قرار دادن درایه‌های آن ، انجام بگیرد. این ویژگی اصلی‌ترین مزیت سیستم تخصصی پیشنهادی در این تحقیق نسبت به ساختمان داده‌های موجود است. حال فرض کنید، رأس‌های x و y برای تعویض برچسب‌هایشان انتخاب شده‌اند. در این‌صورت رابطه بین f_c و s_c و اندیس‌های اعضای مجموعه S_1 و S_2 در ماتریس G به‌صورتی که در جدول ۳ نشان داده شده است؛ خواهد بود.

جدول ۲: اندیس ستون‌های مرتبط با f_c، s_c، S_1 و S_2 در ماتریس G.

شماره رأس	f_c	S_1	s_c	S_2
۱	۱	۲ , ۳	۴	-
۲	۱	۲ , ۳ , ۴	۵	۶
۳	۱	-	۲	۳ , ۴
۴	۱	۲	۳	۴
۵	۱	۲	۳	۴ , ۵ , ۶
۶	۱	۲ , ۳	۴	۵
۷	۱	۲	۳	۴
۸	۱	۲	۳	۴ , ۵

جدول ۳: رابطه بین f_c و s_c و اندیس‌های اعضای مجموعه S_1 و S_2 در ماتریس G.

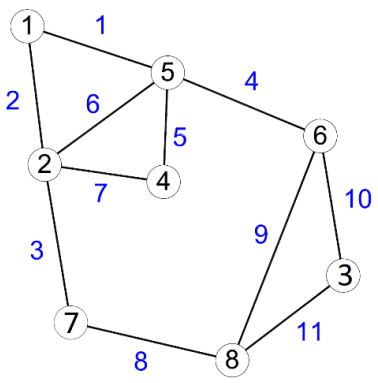
y	x
$f_c = \mathbf{G}(y, 1)$ اندیس‌های اعضای مجموعه $\boldsymbol{S_1}$ در ماتریس **G** عبارتند از : $2, \dots, 2 + f_c - 1$	$f_c = \mathbf{G}(x, 1)$ اندیس‌های اعضای مجموعه $\boldsymbol{S_1}$ در ماتریس **G** عبارتند از : $2, \dots, 2 + f_c - 1$
$s_c = \mathbf{G}(y, f_c + 2)$ اندیس‌های اعضای مجموعه $\boldsymbol{S_2}$ در ماتریس **G** عبارتند از : $f_c + 3, \dots, f_c + s_c + 2$	$s_c = \mathbf{G}(x, f_c + 2)$ اندیس‌های اعضای مجموعه $\boldsymbol{S_2}$ در ماتریس **G** عبارتند از : $f_c + 3, \dots, f_c + s_c + 2$

این نکته نیز باید مورد توجه قرار گیرد که اگر $f_c = 0$ باشد، مقدار صفر در ستون اول ماتریس **G** و در سطر مربوطه درج می‌شود و بلافاصله بعد از آن مقدار s_c درج می‌شود. همچنین، اگر $s_c = 0$ باشد، مقدار صفر در سطر و ستون مربوطه ماتریس **G** درج می‌شود و بقیه ستون‌ها در آن سطر با صفر پر می‌شوند.

در اینجا یک مثال ساده از تعویض برچسب دو رأس فرضی از گراف ترسیم‌شده در شکل ۹ با استفاده از این ساختمان داده جدید ارائه می‌شود. فرض کنید که هدف ما تعویض گره‌هایی با برچسب‌های ۳ و ۷ باشد. در این‌صورت $x = 3$ و $y = 7$ خواهد بود. جدول ۴ و شکل ۱۱ این فرایند را به‌طور کامل نشان می‌دهند.

جدول ۴ : محاسبه مقادیر f_c و s_c برای $x=3$ و $y=7$

$y=7$	$x=3$
$f_c=\mathbf{G}(7,1)=1$ اندیس‌های اعضای مجموعه $\boldsymbol{S}_1$ در ماتریس $\mathbf{G}$ عبارتند از : 2	$f_c=\mathbf{G}(3,1)=0$ مجموعه $\boldsymbol{S}_1$ فاقد عضو است بنابراین در ماتریس $\mathbf{G}$ فقط مقدار f_c درج می شود.
$s_c=\mathbf{G}(7,1+2)=1$ اندیس‌های اعضای مجموعه $\boldsymbol{S}_2$ در ماتریس $\mathbf{G}$ عبارتند از : 4	$s_c=\mathbf{G}(3,0+2)=2$ اندیس‌های اعضای مجموعه $\boldsymbol{S}_2$ در ماتریس $\mathbf{G}$ عبارتند از : 3, 4



۱	۵
۱	۲
۲	۷
۵	۶
۴	۵
۲	۵
۲	۴
۸	۷
۶	۸
۶	۳
۳	۸

$\mathbf{G}=$

۲	۱	۲	۰	۰	۰
۳	۳	۶	۷	۱	۲
۱	۱۱	۱	۱۰	۰	۰
۱	۵	۱	۷	۰	۰
۱	۴	۳	۱	۵	۶
۲	۹	۱۰	۱	۴	۰
۰	۲	۳	۸	۰	۰
۱	۸	۲	۹	۱۱	۰

شکل ۱۱ : گراف ترسیم‌شده در شکل ۹ بعد از تعویض برچسب رئوس و لیست یال مرتبط با آن به همراه ماتریس G بعد از تعویض برچسب رئوس.

با توجه به شکل ۱۱ به‌سادگی مشخص می‌شود که در ستون اول لیست یال گراف ترسیم‌شده در شکل ۹ هیچ درایه‌ای که برابر ۳ باشد وجود ندارد در حالی که در ستون دوم فقط در سطرهای ۳ و ۸ لیست یال گراف (اندیس‌های ستونی ۳ و ۴ در ماتریس **G**) دو درایه با مقدار ۳ وجود دارند.

همچنین در ستون اول و دوم لیست یال این گراف هرکدام یک درایه با مقدار ۷ به ترتیب در سطرهای ۱۱ و ۱۰ وجود دارد. این بدین معنی است که برای تعویض دو رأس با برچسب‌های ۳ و ۷ فقط چهار درایه در لیست یال گراف باید جایشان با هم عوض شود.

همچنین در ماتریس **G** فقط باید جای سطرهای ۳ و ۷ با هم عوض بشوند. بخش‌های مختلف شکل ۱۱ این تغییرات را به‌وضوح منعکس می‌کنند.

همان‌طور که واضح است، انجام این عملیات بسیار ساده است و به زمان بسیار کمی نیاز دارد که علت آن این است که در واقع با استفاده از ساختار پیشنهادی، برای تعویض برچسب دو رأس، دیگر نیازی به خواندن همه درایه‌های لیست یال گراف نیست و دقیقاً مشخص است که چه بخش‌هایی از آن باید دستخوش تغییر بشوند. بعد از به‌روزرسانی لیست یال گراف و ماتریس **G** همه‌چیز برای انجام تعویضی دیگر به‌آسانی مهیا است.

۵- آزمایش‌های عددی

در بخش پیشین، ساختار پیشنهادی به‌طور کامل مورد بررسی قرار گرفت. در این بخش با استفاده از آزمایش‌های عددی کارایی آن را مورد ارزیابی قرار می‌دهیم.

برای ارزیابی این ساختار جدید، ابتدا الگوریتم آن در زبان برنامه‌نویسی C# پیاده‌سازی گردید. سپس مجموعه‌ای از گراف‌های استاندارد با تعداد رئوس متفاوت و تعداد یال‌های متفاوت برای انجام آزمایش‌ها انتخاب شدند. این نمونه‌ها از آدرس https://www.cise.ufl.edu/research/sparse/matrices/Chen/index.html قابل دسترسی هستند.

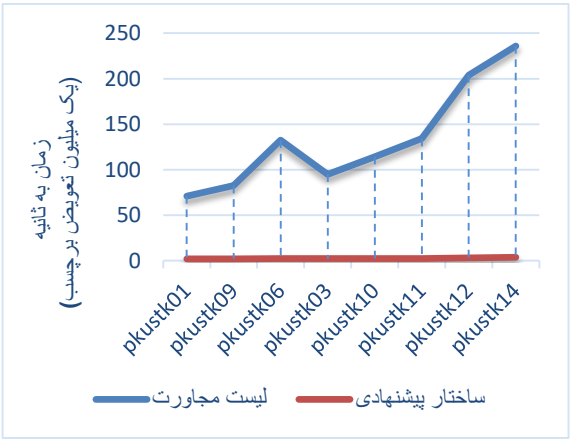
لازم به ذکر است که در تمامی آزمایش‌های انجام‌یافته در این تحقیق از کامپیوتری با پردازنده Intel Xeon(R) X5680 3.33 GHz استفاده شد. برای ارزیابی عملکرد ساختار پیشنهادی، آن را با لیست مجاورت (پیاده‌سازی شده در زبان برنامه‌نویسی C#) که یکی از بهترین ساختمان داده‌های موجود برای توصیف گراف‌ها می‌باشد، مقایسه کردیم. هر ساختار در برابر ۸ گراف استاندارد با انجام دادن ۱۰۰۰۰۰۰ تعویض برچسب رئوس (انتخاب‌شده به‌صورت تصادفی) تست شد. جدول ۵ نتایج به‌دست‌آمده از آزمایش‌های انجام‌یافته را نشان می‌دهد. شکل ۱۲ نیز نمودار زمان اجرای ساختار جدید و لیست مجاورت را برای گراف‌های pkustk نشان می‌دهد. این نمودار که بر اساس اطلاعات جدول ۵ ترسیم شده است به طرز بهتری اختلاف فاحش بین زمان اجرای دو ساختار مورد مقایسه را آشکار می‌سازد.

مقایسه زمان اجرای ساختار جدید و لیست مجاورت در برابر هرکدام از گراف‌های pkustk نشان می‌دهد که اختلاف بسیار چشمگیری بین زمان اجرای آن‌ها وجود دارد. همچنین، میانگین زمان اجرای دو ساختار برای ۸ گراف pkustk نشان می‌دهد که ساختار جدید حدوداً ۵۲ برابر سریع‌تر از لیست مجاورت روی دیتاست مورد استفاده عمل کرده است. این نتایج منعکس‌کننده این واقعیت نیز

هست که وابستگی ساختار پیشنهادی به اندازه گراف در تمامی نمونه‌های مورد آزمایش بسیار کمتر از لیست مجاورت بوده و میزان رشد زمان اجرا با افزایش اندازه گراف بسیار کمتر محسوس است. دلیل اصلی این اختلاف زیاد بین زمان اجرای دو ساختار این است که در ساختار جدید یک سری عملیات اضافی، غیرضروری و یا تکراری که ساختمان داده لیست مجاورت برای انجام عمل تعویض پی در پی برچسب‌ها به آن‌ها نیاز دارد، حذف شده‌اند. ساختار جدید همچنین نیازی به جستجو در لیست گره‌های مجاور یک گره و همچنین پویش کردن کل لیست یالها برای انجام عمل تعویض برچسب‌ها ندارد که طبیعتاً این امر به‌شدت باعث صرفه‌جویی در زمان و افزایش سرعت پردازش می‌شود.

جدول ۵ : مقایسه زمان اجرای ساختار پیشنهادی با لیست مجاورت.

pkustk	تعداد گره‌های گراف	زمان به ثانیه لیست مجاورت	زمان به ثانیه ساختار پیشنهادی
pkustk01	۲۲۰۴۴	۷۰/۹۸۳۲۷۹۹	۱/۸۷۲۹۷۵۵
pkustk09	۳۳۹۶۰	۸۲/۵۵۰۷۴۹۲	۲/۰۳۶۲۸۸۹
pkustk06	۴۳۱۶۴	۱۳۲/۴۰۱۹۷۸۶	۲/۴۵۱۷۵۷۵
pkustk03	۶۳۳۳۶	۹۴/۹۸۹۳۲۴۲	۲/۲۰۴۹۶۶۷
pkustk10	۸۰۶۷۶	۱۱۴/۳۱۲۹۱۰۱	۲/۳۸۱۸۶۵۶
pkustk11	۸۷۸۰۴	۱۳۴/۲۲۳۰۶۵۴	۲/۵۴۸۱۶۷۳
pkustk12	۹۴۶۵۳	۲۰۳/۹۸۲۸	۳/۲۰۶۷۰۲۱
pkustk14	۱۵۱۹۲۶	۲۳۶/۰۰۲۸۱۴۸	۳/۸۷۴۴۲۹۵
میانگین	-	۱۳۳/۶۸۰۸۶۵۳	۲/۵۷۲۱۴۴۱۳۸



شکل ۱۲: نمودار زمان اجرای ساختار پیشنهادی و لیست مجاورت.

۶- نتیجه‌گیری

همان‌طور که ذکر شد، ساختمان داده‌های متعددی برای توصیف گرافها ارائه شده‌اند که مهم‌ترین آن‌ها عبارتند از ماتریس مجاورت، لیست مجاورت، لیست یال و ماتریس وقوع. هرکدام از این ساختارها نقاط ضعف و قوت خاص خودشان را دارند و برای انجام عمل یا عملیات خاصی روی گرافها عملکرد بهتری از خود نشان می‌دهند.

مشکل عمده‌ای که در به‌کارگیری این ساختمان داده‌ها وجود دارد این است که با افزایش یافتن اندازه گراف، تعداد عملیات اضافی که برای انجام عملیات اصلی روی گراف ضروری هستند به‌شدت افزایش پیدا کرده و به‌تبع آن زمان اجرای کلی نیز با شیب خیلی تندی افزایش پیدا می‌کند. این مشکل ابعاد بسیار وسیع‌تری به خود می‌گیرد اگر بخواهیم از الگوریتم‌های جستجو برای حل مسائل بهینه‌سازی که مدل گراف تئوریکی آن‌ها عموماً پیچیده است و اندازه بزرگی دارند، استفاده نماییم که امروزه بسیار رایج است.

در این تحقیق، یک ساختار جدید برای ذخیره‌سازی و سازمان‌دهی داده‌ها در گرافها ارائه شده است. هدف اصلی از این تحقیق، طراحی و پیاده‌سازی ساختاری بود که بتواند مشکلات مذکور را مهار کرده و این امکان را برای ما فراهم آورد تا بدون نگرانی از زمان اجرای بسیار زیاد، از الگوریتم‌های جستجو برای حل مسائل بزرگ و پیچیده بهینه‌سازی که با ساختارهای مبتنی بر گراف سر و کار دارند و در فیلدهای مختلف مطرح هستند، استفاده نماییم.

نتایج به‌دست‌آمده به‌وضوح نشان می‌دهند این هدف تا حدود زیادی محقق شده است چرا که ساختار پیشنهادی عملکرد بسیار مطلوبی از خود نشان داد و توانست ساختمان داده لیست مجاورت که یکی از بهترین ساختمان داده‌های موجود برای گراف است را با اختلاف بسیار چشمگیری پشت سر بگذارد. البته هیچ شکی در این نیست که ساختار پیشنهادی همچون هر الگوریتم دیگری ایدئال نبوده و قطعاً امکان بهبود کیفیت آن وجود دارد که در کارهای آینده به این امر پرداخته خواهد شد.

منابع

[1] F. Harary, Graph Theory. Reading, Mass: Addison-Wesley, 1969.

[2] J. L. Gross, J. Yellen, and P. Zhang, Handbook of Graph Theory, Second Edition, ser. Discrete Mathematics and Its Applications. CRC Press, Taylor & Francis Group, 2014.

[3] J. P. Spinrad, Efficient Graph Representations: The Fields Institute for Research in Mathematical Sciences, ser. Fields Institute monographs. American Mathematical Soc, 2003.

[4] R. P. Grimaldi, Discrete and Combinatorial Mathematics: An Applied Introduction, ser. Addison-Wesley World Student Series. Addison-Wesley, 1994.

[5] D. K. Blandford, G. E. Blelloch, and I. A. Kash, "An experimental analysis of a compact graph representation," in Proceeding of the Sixth Workshop on Algorithm Engineering and Experiments (ALENEX), New Orleans, LA, USA, pp. 49-61, 2004.

[6] D. Dominguez-Sal, V. Munts-Mulero, N. Martnez-Bazan, and J. Larriba-Pey, "Graph representation." in Graph Data Management: Techniques and Applications. IGI Global, pp. 1-28, 2012.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, Third Edition, 3rd ed. The MIT Press, 2009.
[8] D. Caro, M. A. Rodrguez, and N. R. Brisaboa, "Data structures for temporal graphs based on compact sequence representations," Information Systems, vol. 51, pp. 1 – 26, 2015.
[9] V. Kumar and S. K. Muttoo, "A data structure for graph to facilitate hiding of information in a graph's segments a graph theoretic approach to steganography," International Journal of Communication Networks and Distributed Systems, vol. 3, no. 3, pp. 268–282, 2009.
[10] J. Kim, H. seong Park, and Y. H. Kim, "Memory efficient data structure for graph representation of dspf netlist," in 2014 International SoC Design Conference (ISOCC), pp. 292-293, Nov 2014.
[11] R. Veras, T. M. Low, and F. Franchetti, "A scale-free structure for power-law graphs," in 2016 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-7, Sept 2016.
[12] B. M. Kapron, V. King, and B. Mountjoy, "Dynamic graph connectivity in polylogarithmic worst case time," in Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms, ser. SODA '13. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, pp. 1131-1142, 2013.
[13] J. D´ıaz, J. Petit, and M. Serna, "A survey of graph layout problems," Computing Surveys, vol. 34, pp. 313–356, 2002.
[14] F. Chicano, B. Hu, and P. Garc´ıa-S´anchez, Evolutionary Computation in Combinatorial Optimization. Springer, 2016.
[15] R. Marti, M. Laguna, F. Glover, and V. Campos, "Reducing the bandwidth of a sparse matrix with tabu search," European Journal of Operational Research, vol. 135, no. 2, pp. 450–459, 2001.
[16] B. Koohestani and R. Poli, "A hyper-heuristic approach to evolving algorithms for bandwidth reduction based on genetic programming," in Research and Development in Intelligent Systems XXVIII, M. Bramer, M. Petridis, and L. Nolle, Eds. Springer London, pp. 93-106, 2011.
[17] B. Koohestani and R. Poli, "Addressing the envelope reduction of sparse matrices using a genetic programming system," Computational Optimization and Applications, vol. 60, no. 3, pp. 789–814, 2015.
[18] A. Lim, B. Rodrigues, and F. Xiao, "Integrated genetic algorithm with hill climbing for bandwidth minimization problem," in Proceedings of the Genetic and Evolutionary Computation Conference, ser. LNCS, vol.2724, E. Cant´u-Paz and et al., Eds. Springer, Heidelberg, pp. 1594-1595, 2003.
[۱۹] مریم مرادی، رزا یوسفیان، وحید رافع. «ارائه راهکاری جهت مقابله با مشکل انفجار فضای حالت در سیستم‌های تبدیل گراف با استفاده از الگوریتم‌های پرندگان و جستجوی گرانشی». مجله مهندسی برق دانشگاه تبریز، صفحه ۱۶۳-۱۷۷، جلد ۴۵، شماره ۴، ۱۳۹۴.
[۲۰] سمیه توکلی، افسانه فاطمی. «تشکیل تیم دوهدفه در شبکه‌های اجتماعی». مجله مهندسی برق دانشگاه تبریز، صفحه ۴۲۳-۴۳۳، جلد ۴۷، شماره ۲، ۱۳۹۶.
[21] A. Lim, J. Lin, and F. Xiao, "Particle swarm optimization and hill climbing for the bandwidth minimization problem," Applied Intelligence., vol. 26, no. 3, pp. 175–182, 2007.
[22] E. Pi˜nana, I. Plana, V. Campos, and R. Mart´ı, "GRASP and path relinking for the matrix bandwidth minimization," European Journal of Operational Research, vol. 153, no. 1, pp. 200–210, 2004.
[23] B. Koohestani and R. Poli, "Evolving an improved algorithm for envelope reduction using a hyper-heuristic approach," IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 543–558, Aug 2014.
[24] S. Pissanetskey, Sparse Matrix Technology. London: Academic Press, 1984.

زیرنویس‌ها

[1] Graph Theory
[2] Data Structures
[3] Adjacency Matrix
[4] Adjacency List
[5] Edge List
[6] Incidence Matrix
[7] Optimization
[8] Search Algorithms