سال دوازدهم، شماره‌های ۴۵ و ۴۶، پاییز و زمستان۱۳۹۹

صص: ۹۵-۱۱۴

دو فصلنامه علمی

فناوری اطلاعات و ارتباطات ایران

بهبود کارائی و دقت یافتن یال‌های پرتکرار در خلاصه سازی gMatrix از جریان گراف

مسعود کاظمی* سیدحسین خواسته حمیدرضا رخصتی*****

*دانشجوی کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی

**استادیار، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی

***کارشناسی ارشد هوش مصنوعی و رباتیکز، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی

تاریخ دریافت: ۱۳۹۹/۰۳/۰۵ تاریخ پذیرش: ۱۳۹۹/۰۹/۲۶

نوع مقاله: پژوهشی

چکیده

در سیستم‌های کاربردی، گراف‌ها با دامنه وسیعی از راس‌ها وجود دارند و یال‌ها به سرعت زیادی در قالب جریان گراف تولید می‌شوند. یکی از مسائل موجود در جریان‌های گراف سنگین که به صورت لحظه‌ای وارد می‌شوند پیدا کردن زیرگراف‌های پرتکرار است. خلاصه‌های جریان مبتنی بر طرح، مانند count-min، اطلاعات گره‌های پرتکرار را با دقت قابل قبولی نگهداری می‌کنند ولی ساختار گراف اصلی را از دست می‌دهند. از بین این روش‌ها، gMatrix ساختاری می‌باشد که مشخصات گراف اصلی را نیز حفظ می‌کند. این روش از توابع درهم‌ساز مختلف، برای ذخیره‌ی خلاصه‌ی جریان گراف استفاده کرده و به کمک این توابع و معکوس آنها، زیرگراف‌های پرتکرار را به‌دست می‌آورد. به دلیل داشتن حجم کمتر از جریان اصلی، gMatrix معمولا به پرس و جوها با دقت بالایی پاسخ نمی‌دهد. همچنین این روش از مشکل مرتبه‌ی زمانیِ بالا در پاسخ به پرس و جوها هم رنج می‌برد. در این مقاله روش جدیدی ارائه شده است که به ازای هزینه‌ی کمِ حافظه‌ی مصرفی، زمان پاسخگویی به پرس و جو زیرگراف پرتکرار را به صورت چشم‌گیری کاهش می‌دهد. همچنین الگوریتم ارایه شده با افزایش استقلال بین توابع در هم سازی با استفاده از روش شباهت برداری کُساین، احتمال برخورد عناصر در هم سازی شده را کاهش می‌دهد. نتایج آزمایشات تجربی که به زبان C++ پیاده‌سازی شده است و بر روی داده‌های شبکه اجتماعی فرندستر اجرا شده است، نشان می‌دهد که روش پیشنهادی برای یافتن زیرگراف‌های پرتکرار پیچیدگی زمانی و دقت یافتن این زیر گراف‌ها را بهبود می‌بخشد.

واژگان کلیدی: جریان گراف، خلاصه‌سازی مبتنی بر طرح، gMatrix، توابع درهم‌ساز، شباهت برداری کُساین

نویسنده مسئول: سید حسین خواسته khasteh@kntu.ac.ir

۱. مقدمه

در علوم کامپیوتر، جریان داده[۱] به یک دنباله از داده‌ها که با گذشت زمان در دسترس خواهند بود گفته می‌شود. می‌توان یک جریان را آیتم‌های موجود روی یک تسمه نقاله در نظر گرفت که به صورت تک تک پردازش می‌شوند و نه به صورت جمعی. پردازش جریان‌ها به صورت کاملا متفاوتی از پردازش داده‌ها به صورت دسته‌ای صورت می‌گیرد. از آنجایی که ممکن است جریان‌ها نامحدود باشند، توابع معمولی نمی‌توانند روی جریان‌ها به صورت کلی استفاده شوند. به صورت رسمی جریان‌ها را داده[۲] نمی‌دانند چرا که داده محدود می‌باشد، حال آنکه جریان‌ها بالقوه نامحدود هستند و شبه‌داده[۳] نامیده می‌شوند. علاوه بر آن معمولا جریان‌ها حجیم هستند. به صورتی که نگهداری و ذخیره تمام اجزای آن امکان‌پذیر نمی‌باشد. حتی در صورت امکان ذخیره این حجم داده، به دلیل پویایی این داده‌ها و تغییر آن در هر ورود داده جدید در گذر زمان، عملا اجرای دوباره و دوباره یک عملیات بر روی کل داده‌هایی که تاکنون دریافت شده است امکان پذیر نمی‌باشد و بسیار کار پرهزینه‌ای می‌باشد. یکی از انواع جریان‌ها، جریان گراف[۴] می‌باشد. جریان گراف در قالب جریانی از رخدادهای صورت گرفته روی یک گراف تعریف می‌شود. در واقع جریان گراف به جای مجموعه‌ای ثابتی از رئوس و یال‌ها و مقادیر آنها، رخدادهایی در گذر زمان می‌دهد که اطلاعاتی در مورد راس‌ها و یال‌ها را ارائه می‌دهد. در نتیجه جریان گراف می‌تواند شامل یال‌های یک گراف که به صورت تدریجی وارد می‌شوند باشد. دامنه گسترده‌ای از داده‌ها مانند شبکه‌های حمل و نقل، شبکه‌های آی‌پی[۵] و شبکه‌های اجتماعی به این تقسیم بندی تعلق دارند.

بر روی گراف‌ها پرس و جو[۶]های کاربردی مختلفی تعریف شده است که در حوزه‌های مختلف اطلاعاتی از ساختار کلی گراف و همچنین اطلاعات جزیی مربوط به گره یا یال مشخصی را می‌دهند. حال آنکه در جریان گراف به خاطر وجود محدودیت‌هایی که در حفظ و نگهداری جریان گراف و همچنین محدودیت‌هایی که در پردازش تمام ساختار یک گراف سنگین (که به صورت جریان وارد می‌شود) داریم، عملا بسیاری از پرس و جوهای موجود به شکل کنونی خود قابل استفاده نیستند. برای پردازش سریع جریان‌های گراف روش‌های زیادی در انواع مختلف از جمله پنجره لغزان[۷] و طرح گراف[۸] معرفی شده‌اند. روش‌های مبتنی بر طرح یک خلاصه از جریان گراف را نگهداری می‌کنند و به محض ورود یال‌های جدید در گذر زمان آن خلاصه را بروزرسانی می‌کنند. از جمله خلاصه‌های جریان مبتنی بر طرح می‌توان به count-min اشاره کرد که می‌تواند اطلاعات تکرار گره‌ها با تکرار بالا را با دقت قابل قبولی نگهداری کنند. [۱]اما این خلاصه مبتنی بر طرح ساختار گراف اصلی را از دست می‌دهد. مگر اینکه ما اطلاعات راس شروع و پایان هر یال را ذخیره کنیم که این کار بسیار هزینه بر و نشدنی می‌باشد. به عنوان مثال اکثر روش‌های موجود می‌توانند راس‌ها و یال‌های تکرار بالا را تشخیص دهند، اما آنها قادر به جواب پرس و جوهای پیچیده‌تر مانند پیدا کردن مسیرهایی که از یال‌های با تکرار بالا تشکیل شده است نخواهند بود. در طرف دیگر، خلاصه مبتنی برطرح gMatrix که یک طرح سه بعدی می‌باشد، جریان‌های گراف سنگین را در حالی که اطلاعاتی از رفتارهای ساختاری گراف مذکور نگهداری می‌کند، به صورت بلادرنگ خلاصه می‌کند. در نتیجه gMatrix قادر به پاسخگویی پرس و جوهای ساختاری مهم از قبیل دسترسی پذیری از طریق یال‌های پرتکرار نیز خواهد بود.[۲]

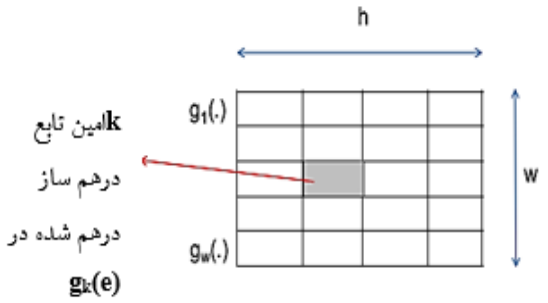
۲. کارهای پیشین

۲_۱ روش طرح count-min

در الگوریتم count-min با استفاده از یک روش درهم سازی، فرکانس حدودی تعداد زیادی از آیتم‌های مجزا در یک جریان داده محاسبه می‌شوند. در این الگوریتم به تعداد $w =$

[۱] Data Stream
[۲] Data
[۳] Codata
[۴] Graph Stream
[۵] IP
[۶] Query
[۷] Sliding Window
[۸] Graph Sketch

$\ln(\frac{۱}{delta})$ عدد تابع درهم سازی که دو به دو مستقل می‌باشند استفاده می‌شود و هرکدام به یک عدد در بازه $h =$ $[۰, \frac{e}{\epsilon}]$ به صورت یکنواخت نگاشت می‌شود. که e پایه لگاریتم طبیعی است. همچنین داده ساختار مورد نظر از یک آرایه دو بعدی با ابعاد $h * w$ تشکیل شده است. همان‌طور که در شکل ۱) مشخص است، هر تابع درهم سازی با یکی از w آرایه یک بعدی h خانه‌ای مطابقت دارد.



شکل ۱. طرح count-min برای جریان داده[۱]

در یک جریان داده یک بعدی که عناصر آن دامنه‌های وسیعی دارند وقتی المنت جدیدی وارد می‌شود تمام w تابع درهم سازی روی آن اجرا می‌شود تا به یک عدد در بازه $[۰, h - ۱]$ نگاشت شوند. سپس مقادیر این w خانه یکی اضافه می‌شوند. به جهت محاسبه تعداد یک آیتم، مجموعه w خانه مطابق با آن که هر یک را w تابع درهم سازی می‌دهند بازیابی شده و مقدار کمینه میان این خانه‌ها پیدا می‌شوند. اگر فرض کنیم Ct تعداد واقعی این آیتم باشد، مقدار تخمین زده شده در کمترین حالت با Ct برابر است. از آنجایی که شمارش غیر منفی مد نظر است و به دلیل امکان وجود تداخل در خانه‌های درهم سازی ممکن است تخمین زیاد انجام پذیرد. همچنین در یک جریان داده با L داده ورودی، تخمین برآورده شده برای خانه Ct در بیشترین مقدار با احتمال $(۱ - \delta)$ برابر با $(Ct + \epsilon * L)$ خواهد بود. به همین شکل در یک رخداد که آیتم‌ها فرکانس‌های مربوط به خود را دارند، باید تعداد هر آیتم را با فرکانس مربوطه آن افزایش داد. در این حالت L را مجموع فرکانس‌های تمام آیتم‌ها تا این لحظه در نظر گرفته می‌شود.

۲_۲ روش gMatrix

در این مقاله فرض شده است که داده‌های گراف $G = (V, E)$ به صورت جریان داده‌ای یالی دریافت می‌شود. هر راس $i \in V$ از مجموعه $N = \{۱, ۲, \dots, n\}$ برداشته می‌شود و هر یال $(i, j) \in E$ یک یال جهت دار است. جریان داده ورودی گراف مورد نظر شامل عضوهایی به صورت $(i_۱, j_۱, f_{i_۱ j_۱}), (i_۲, j_۲, f_{i_۲ j_۲}), \dots, (i_t, j_t, f_{i_t j_t}), \dots$ می‌باشد. اینجا $(i_t, j_t, f_{i_t j_t})$ نشان دهنده دریافت یال t-ام جریان داده با تکرار $f_{i_t j_t}$ می‌باشد. در بسیاری از کاربردها، مقدار $f_{i_t j_t}$ به صورت طبیعی برابر یک است، ولی ما یک تکرار عمومی غیر از یک را برای کلیت بخشیدن به نتایج در نظر می‌گیریم. برای مثال در کاربردهای مخابراتی، عدد تکرار $f_{i_t j_t}$ می‌تواند نشان دهنده تعداد ثانیه‌هایی باشد که شخص i_t با شخص j_t با شروع در زمان t در حال مکالمه بوده است. یک یال می‌تواند چندین بار در یک جریان داده ظاهر شود. زمانی که یک پرس و جو را مطرح می‌کنیم، مثلا پیدا کردن تعداد تکرار یال (i, j)، به دنبال جمع تکرارهای یال (i, j) تا کنون هستیم. درحالی که روش‌های طرح محور می‌توانند با کمی تغییر برای پرس و جوهای زمانی نیز استفاده شوند، آنها را در این قسمت بررسی نمی‌کنیم. همچنین توجه کنید که نتایج ما با فرض بر اینکه یال (i, j) همیشه به صورت مرتب شده است به سادگی می‌توانند به گراف‌های غیر جهت دار تعمیم داده شوند.

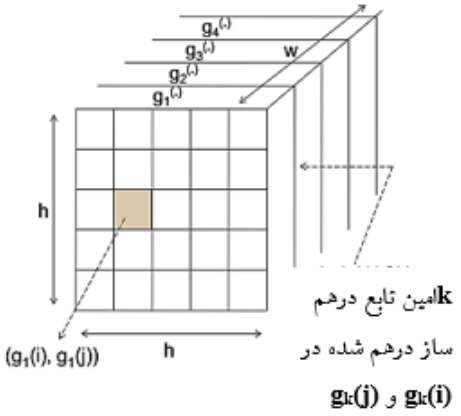
پرس و جوهای تعریف شده در روش gMatrix، تکرار یال (محاسبه تکرار یک یال (i, j))، یال‌های پرتکرار (پیدا کردن همه یال‌هایی که تکرار آنها بیشتر از یک حد آستانه داده شده F می‌باشد)، مجموع تکرار راس (محاسبه مجموع تکرار تمام یال‌های ورودی و خروجی برای یک راس i)، راس‌های پر تکرار (پیدا کردن همه راس‌های که مجموع تکرار آنها (روی همه یال‌های ورودی و خروجی از آنها) از یک حد آستانه داده

شده F بیشتر می‌باشد.)، مجموع تکرار زیرگراف (محاسبه مجموع تکرار همه‌ی یال‌های یک زیرگراف که روی زیرمجموعه‌ای از راس‌های داده شده S می‌باشد) و قابل دستیابی بودن(یافتن اینکه آیا یک راس مبدا[۹] به راس دیگری به عنوان مقصد[۱۰] از طریق یال‌های با تکرار بیشتر از آستانه داده شده F قابل دسترسی می‌باشد و یا خیر).

در دو پرس و جوی اول فقط تکرار یال‌ها مطرح می‌باشد که با طرح count-min هم قابل پاسخ گویی می‌باشند. به همین صورت پرس و جوی سوم و چهارم فقط به تکرار راس‌ها نیازمند می‌باشد که مجددا با یک طرح بر روی راس‌ها قابل پاسخ گویی می‌باشد. اما دو پرس و جوی آخر به دنبال یافتن رفتار ساختاری جریان گراف مورد نظر بر مبنای تکرار یال‌های آن می‌باشند. که این پرس و جوها با استفاده از طرح‌های کنونی موجود قابل پاسخ گویی نیستند و باید اطلاعات شروع و پایان همه یال‌ها نیز ذخیره شود که بسیار سنگین می‌باشد. ساختار gMatrix طراحی شده یک ماترسی سه بعدی می‌باشد. به جای ارایه یک راه حل کاملا متفاوت، gMatrix بر پایه اصول ساختار count-min طراحی شده است.

۲_۳ ساختار gMatrix

gMatrix یک داده ساختار ۳ بعدی از گراف داده می‌باشد که دو بعد آن با راس‌های شروع و پایان مطابقت دارد. هر تابع درهم سازی یک نگاشت از مجموعه رئوس N به یک عدد صحیح در بازه $[۰, h-۱]$ تعریف می‌کند. در نتیجه k امین تابع درهم سازی با $g_k(.)$ نمایش داده می‌شود و یال (i,j) را به $(g_k(i), g_k(j))$ نگاشت می‌کند. در شکل ۲) طرح این روش برای جریان گراف قابل مشاهده است، مانند الگوریتم count-min در اینجا نیز از w تابع درهم سازی که دو به دو مستقل می‌باشند استفاده شده است. در نتیجه k از ۱ تا w می‌باشد. جدول درهم سازی در اینجا یک آرایه سه بعدی با $h * h * w$ خانه می‌باشد. هر خانه با مختصات (p,q,r) آدرس دهی می‌شود که در آن p و q اندیس‌های رئوس نگاشت شده می‌باشند و r اندیس تابع درهم سازی استفاده شده در نگاشت می‌باشد. مقدار خانه با مختصات (p,q,r) یک عدد صحیح می‌باشد که با $V(p,q,r)$ نمایش داده می‌شود و در آن فرکانس یال درهم سازی شده (p,q) برای تابع درهم سازی r ام ذخیره می‌شود.



شکل ۲. طرح gMatrix برای جریان‌های گراف[۲]

۲_۳_۱ به‌روزرسانی خلاصه‌ی gMatrix

برای بروز رسانی gMatrix، کافی است در شروع تمام خانه‌ها را با ۰ مقدار دهی اولیه کنیم و برای هر یال ورودی (i,j) با فرکانس $f(i,j)$ یال درهم سازی شده متناظر $(g_k(i), g_k(j))$ را برای تمام مقادیر k از ۱ تا w محاسبه میکنیم و سپس فرکانس هر یک از این w خانه را با مقدار $f(i,j)$ افزایش دهیم. میزان فضای مورد استفاده در gMatrix از مرتبه $O(h * h * w)$ می‌باشد و همچنین پیچیدگی زمانی برای بروزرسانی یک یال ورودی از جریان برابر با $O(۲ * w)$ می‌باشد [۳, ۴].

۲_۳_۲ تابع درهم ساز و عکس درهم سازی

به جهت کارا بودن gMatrix، باید توابع درهم ساز دو به دو مستقل باشند. به علاوه، به جهت، پردازش پرس و جوهای

[۹] Source

[۱۰] Destination

یال‌های پرتکرار و همچنین پرس و جوهای مبتنی بر ساختار که در قسمت‌های بالا ذکر شده اند، به محاسبه عکس نگاشت تابع درهم ساز نیاز است. اگر فرض کنیم تابع g یک تابع در هم ساز می‌باشد، ما معکوس این تابع را با g^{-1} به صورت (۱) تعریف می‌کنیم.

$$g^{-1}(p) = \{i: g(i) = p\} \qquad (۱)$$

طول عکس نگاشت درهم سازی فقط $[P/h]$ باشد و از طریق تعمیم الگوریتم اقلیدس به صورت کارآیی با پیچیدگی زمانی $O([\frac{P}{h}] * log(P))$ محاسبه می‌شود.[۵]

۲_۴ محاسبه پرس و جوی تکرار یال

برای محاسبه تکرار یک یال (i,j) با استفاده از gMatrix، مقادیر $V(g_k(i), g_k(j), k)$ برای w مقدار مختلف k محاسبه می‌شوند. کمینه این مقادیر به عنوان تخمین تکرار $Q(i,j)$ از یال (i,j) در نظر گرفته می‌شود. این تخمین را با نماد $Q'(i,j)$ نشان می‌دهیم. پیچیدگی زمانی مورد نیاز برای تخمین تکرار یک یال $O(2w)$ است که w تعداد توابع درهم ساز می‌باشد [۶, ۷].

۲_۵ محاسبه پرس و جوی یال‌های پرتکرار

یال‌های پرتکرار تمام یال‌های با تکرار بیشتر از یکی آستانه مشخص شده توسط کاربر می‌باشند. از آنجایی که gMatrix یک طرح احتمالی است، نمی‌توان تمام یال‌های پرتکرار را به صورت قطعی پیدا کرد. در نتیجه، یک روش ارائه شده است که هیچ منفی کاذب[۱۱]ای را بازیابی نمی‌کند، اما یک یال ممکن است نمونه مثبت کاذب[۱۲] باشد. متعاقبا احتمال مثبت کاذب بودن یک یال نیز محاسبه می‌شود. این روش دو گام دارد:

۱. در گام اول، gMatrix اسکن می‌شود و تمام یال‌های درهم سازی شده که تکرار آنها از حد آستانه F بیشتر است برای توابع درهم ساز مختلف پیدا می‌شوند.
۲. در گام دوم، برای هر یال (p,q) پر تکرار تحت تابع درهم ساز g_k، مجموعه تمام یال‌های گراف اصلی پر تکرار ممکن با استفاده از تکنیک عکس نگاشت درهم سازی پیدا شده که با () نمایش داده می‌شود.

(۲)

$$E_k = \{(i,j): i \in g_k^{-1}(p), j \in g_k^{-1}(q), V(g_k(p), g_k(q), k) \geq F\}$$

در نهایت، تصادم[۱۳] مجموعه یال‌های مختلف E_k برای همه $k \in (1, w)$ یال‌های پرتکرار را به ما می‌دهد. توجه داشته باشید که گام دوم پرهزینه ترین گام محاسباتی این روش می‌باشد. به همین منظور، تکنیک‌های بهینه سازی برای پرس و جو یال‌های پرتکرار نیز ارائه می‌شوند [۸, ۹, ۱۰].

۳. روش پیشنهادی

در این مقاله مشکلات روش gMatrix از دو جهت بهبود یافته است. در رویکرد اول سعی شده است تا زمان اجرای یافتن یال‌های پرتکرار کاهش یابد و در رویکرد دوم به استفاده از توابع درهم‌سازی مستقل سعی شده است تا نرخ مثبت کاذب در یافتن این یال‌ها بهبود یابد.

۳_۱ بهینه‌سازی زمان اجرا به کمک داده ساختارها

الگوریتم موجود در روش gMatrix برای محاسبه‌ی اشتراک مجموعه‌ها از روش مناسبی استفاده نمی‌کند. برای سریع‌تر کردن مرحله یافتن اشتراک دو مجموعه از داده ساختار درخت قرمز-سیاه استفاده می‌کنیم که با توجه به شناخته بودن این درخت از توضیح نحوه کارکرد آن خودداری می‌کنیم. در این قسمت مراحل استفاده از داده‌ساختار قرمز-سیاه و نحوه بهینه‌سازی پرس و جوی یال‌های پرتکرار به صورت زیر

[۱۱] False Negative
[۱۲] False Positive
[۱۳] Collision

توضیح داده می‌شود. توجه داشته باشید که درخت قرمز-سیاه فقط مقادیر یکتا را نگهداری می‌کند.

۱. *دو درخت قرمز-سیاه $IntersectedQ۱$ و $IntersectedQ۲$ را تعریف کنید.*

۲. *برای تابع درهم ساز اول، برای هر خانه (p,q) از gMatrix که مقدار آن از آستانه F بیشتر است مراحل الف و ب را انجام دهید:*

أ. *عکس نگاشت p را بدست آورید و همه مقادیر را در $IntersectedQ۱$ قرار دهید.*

ب. *عکس نگاشت q را بدست آورید و همه مقادیر را در $IntersectedQ۲$ قرار دهید.*

۳. *برای تابع درهم ساز ۲ تا w مراحل الف تا ث را انجام دهید:*

أ. *دو درخت قرمز-سیاه $Q۱$ و $Q۲$ را تعریف کنید.*

ب. *برای هر خانه (p,q) از gMatrix که مقدار آن از آستانه F بیشتر است مراحل ۱ و ۲ را انجام دهید:*

۱. *عکس نگاشت p را بدست آور و همه مقادیر را در $Q۱$ قرار دهید.*

۲. *عکس نگاشت q را بدست آور و همه مقادیر را در $Q۲$ قرار دهید.*

ت. *عناصری از $IntersectedQ۱$ که در $Q۱$ نیستند را حذف کنید.*

ث. *عناصری از $IntersectedQ۲$ که در $Q۲$ نیستند را حذف کنید.*

۴. *همه یال‌های ممکن از مجموعه گره‌های $IntersectedQ۱$ به مجموعه گره‌های $IntersectedQ۲$ را می‌سازیم.*

۵. *حذف کردن منفی کاذب‌ها از یال‌های تولید شده و ارائه جواب نهایی*

۳_۲ تحلیل پیچیدگی زمانی

پیچیدگی پیمایش کردن خلاصه gMatrix برابر $O(h^۲w)$ می‌باشد. فرض می‌کنیم که k-مین تابع درهم ساز از gMatrix که $۱ \le k \le w$ در مجموع تعداد c_k خانه با تعداد تکرار بیشتر یا مساوی آستانه F دارد [۱۱]. آنگاه بر اساس طراحی ما برای مدولار بودن تابع‌های درهم ساز، هرکدام از این خانه‌ها به $[P/h]$ راس‌های مبدا و $[P/h]$ راس‌های مقصد مربوط می‌شوند که نیاز به زمان $O(۲c_k[P/h]\ logP)$ برای یافتن تمامی این مجموعه راس‌ها داریم. نهایتا، همانگونه که بحث شد اشتراک را روی مجموعه راس‌ها می‌گیریم. بنابراین در مجموع پیچیدگی زمانی پرس و جوی یال‌های پرتکرار برابر است با رابطه (۳).

$$O\left(h^۲w + ۲\left[\frac{P}{h}\right]\log P\ \prod_{k=۱}^{w} c_k\right) \qquad (۳)$$

با توجه به مراحلی که در بخش قبل توضیح داده شد، برای هر تابع درهم‌ساز شماره k، تعداد $h^۲$ خانه بررسی می‌شود که اگر فرض کنیم به تعداد C_k خانه مقدار بیشتر از حد آستانه F داشته باشند و از آنجایی که برای هر گره تعداد $\left[\frac{P}{h}\right]$ مقدار عکس نگاشت بدست می‌اید، در نتیجه در بیشترین حالت (همه مقادیر عکس نگاشت‌ها یکتا باشند) به تعداد $C_k\left[\frac{P}{h}\right]$ مقدار در هر درخت قرار می‌گیرد که مرتبه زمانی قرار دادن هر مقدار در درخت $O(\log(C_k\left[\frac{P}{h}\right]))$ خواهد بود. پس برای هر تابع درهم‌ساز مرتبه زمانی کلی $O\left(h^۲+۲C_k\left[\frac{P}{h}\right]\log\left(C_k\left[\frac{P}{h}\right]\right)\right)$ می‌باشد.

همچنین برای توابع درهم‌ساز شماره ۲ تا w، علاوه بر این، یک بار روی درخت عکس نگاشت‌های تابع درهم‌ساز اول حرکت می‌کنیم و تمام مقادیر را در درخت تابع درهم‌ساز کنونی جستجو می‌کنیم که در مجموع مرتبه زمانی $O(۲C.\left\lceil\frac{P}{h}\right\rceil log\,(C_k\left\lceil\frac{P}{h}\right\rceil))$ خواهد بود[۱۲]. در نتیجه به صورت کلی مرتبه زمانی الگوریتم پیشنهادی در رابطه (۴) آمده است. همانطور که در رابطه‌های (۳) و (۴) مشخص است پیچدگی زمانی الگوریتم از ضرب تعداد ضریب C_k به جمع آنها تقلیل یافته است.

(۴)

$$O(wh^{۲} + \sum_{k=۱}^{w} {}^{۲}C_k\left\lceil\frac{P}{h}\right\rceil log\,(C_k\left\lceil\frac{P}{h}\right\rceil) + \sum_{k=۲}^{w} {}^{۲}C.\left\lceil\frac{P}{h}\right\rceil log\,(C_k\left\lceil\frac{P}{h}\right\rceil))\,)$$

۳_۳ بهبود دقت پرس و جو به کمک توابع درهم‌ساز مستقل

وجود وابستگی بین توابع درهم‌ساز باعث می‌شود که در پاسخ به پرس و جوهای مختلف، و در مرحله‌ی ابهام‌زدایی برای یافتن پاسخ پرس و جو، تصادم بسیار زیادی به وجود آید. این مشکل باعث می‌شود که دقت پاسخگویی به پرس و جوها در روش gMatirx کاهش یابد. برای مرتفع کردن این مشکل باید از توابع درهم‌سازی استفاده کرد که وابستگی کمتری به یکدیگر داشته باشند تا بتوانند روی جدول‌های درهم‌سازی مختلف تصادم کمتری ایجاد کنند. برای این منظور از روش بردارهای شباهت یا معیارهای فاصله بین توابع درهم‌ساز استفاده می‌شود. هرچقدر که میزان فاصله‌ی برداری بین توابع درهم‌ساز بیشتر باشد، باعث می‌شود که تصادم کمتری در ابهام‌زدایی توابع درهم‌ساز ایجاد شود [۱۳, ۱۴]. در این پژوهش از روش‌های بردار فاصله‌ی کُساین، اطلاعات متقابل و فاصله‌ی اقلیدسی استفاده شده است. در بخش بعدی به روش کلی الگوریتم ارائه شده برای یافتن توابع درهم‌ساز مستقل پرداخته می‌شود و نحوه‌ی تبدیل مسئله به یک مسئله در فضای برداری ارائه می‌شود.

۳_۳_۱ شباهت توابع درهم ساز با معیار فاصله

تابع درهم سازی مانند $h_i\,(x)$ را به صورت زیر تعریف می‌کنیم:

(۵)

$$x_h^i = h_i(x)$$

در رابطه (۵) x را ورودی تابع درهم سازی و $h_i\,(x)$ را تابع در هم سازی i-ام و x_h^i را خروجی تابع در هم سازی i-ام می‌نامیم. برای ارزیابی دو تابع درهم سازی $h_i\,(x)$ و $h_j\,(x)$ در اولین گام، برداری به طول دلخواه $K \geq ۱۰۰۰$ از اعداد طبیعی در فاصله‌ی $[۱, P-۱]$ در نظر می‌گیریم و خروجی تابع‌های درهم‌ساز را برای این بردار محاسبه کرده و بردارهای x_{h_i} و x_{h_j} می‌نامیم. با یافتن فاصله برای دو بردار بدست آمده، میزان شباهت این دو تابع درهم سازی به صورت زیر محاسبه می‌شود:

(۶)

$$Sim(h_i, h_j) = d_?(x_{h_i}, x_{h_j})$$

در رابطه شماره‌ی (۶) تابع محاسبه‌ی فاصله یعنی $d_?(x_{h_i}, x_{h_j})$ یکی از سه روش فاصله‌ی اقلیدسی، فاصله‌ی کُساین و یا اطلاعات متقابل است. رابطه‌های ریاضی هر یک از این معیارهای فاصله در بخش قبلی توضیح داده شد. در ادامه به بررسی نحوه‌ی استفاده از هر یک از معیارهای فاصله‌ی برای محاسبه‌ی شباهت دو تابع درهم‌ساز یعنی مقدار عددی تابع $Sim(h_i, h_j)$ پرداخته می‌شود.

۳_۳_۲ شباهت با فاصله‌ی اقلیدسی

فاصله‌ی اقلیدسی به عنوان ساده‌ترین روش محاسبه‌ی فاصله یا شباهت در مسائل فضاهای برداری در نظر گرفته می‌شود.

در این مسئله به سادگی می‌توان میزان شباهت دو تابع درهم‌ساز را به کمک فاصله‌ی دو بردار درهم‌سازی شده به کمک این دو تابع یعنی x_{h_i} و x_{h_j} به صورت رابطه (۷) در نظر گرفت:

(۷)

$$Sim(h_i, h_j) = \mathrm{d_E}\left(x_{h_i}, x_{h_j}\right) = \sqrt{\sum_{k=1}^{K}\left(x_{h_i}^k - x_{h_j}^k\right)^2}$$

که در این رابطه، K همان طول بردار در نظر گرفته شده است و $x_{h_i}^k$ درایه‌ی k-ام از نمایش برداری خروجی درهم‌سازی شده‌ی درایه‌های بردار ابتدایی به کمک تابع درهم‌سازی i-ام یعنی h_i است [۱۵].

۳_۳_۳ شباهت با اطلاعات متقابل

با توجه به تعریف اطلاعات متقابل در بخش قبلی، می‌توان شباهت دو بردار x_{h_i} و x_{h_j} را از طریق اطلاعات متقابل به دست آورد.

(۸)

$$Sim(h_i, h_j) = \mathrm{d_I}\left(x_{h_i}, x_{h_j}\right) = \sum_{y \in x_{h_j}} \sum_{x \in x_{h_i}} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

که در () احتمال رخداد یک $x \in x_{h_i}$ به صورت زیر تعریف می‌شود:

(۹)

$$p\left(x \in x_{h_i}\right) = \frac{n(x)}{K}$$

که $n(x)$ تعداد رخدادهای مقدار x در x_{h_i} است. این احتمال به همین ترتیب برای هر $y \in y_{h_j}$، تعریف می‌شود:

(۱۰)

$$p\left(y \in x_{h_j}\right) = \frac{n(y)}{K}$$

که در آن $n(y)$ تعداد رخدادهای مقدار y در x_{h_j} است.

احتمال توام $p(x,y)$ نیز به این ترتیب تعریف می‌شود که در این رابطه $n(x,y)$ تعداد رخدادهایی به صورت (x,y) است که x عضوی از بردار x_{h_i} و y عضوی از بردار x_{h_j} باشد [۱۶].

(۱۱)

$$p(x,y) = \frac{n(x,y)}{K * K}$$

۴_۳_۳ شباهت با فاصله‌ی کُساین

با یافتن فاصله‌ی کُساین برای دو بردار بدست آمده، میزان شباهت این دو تابع در هم سازی به صورت زیر محاسبه می‌شود:

(۱۲)

$$Sim(h_i, h_j) = \mathrm{d_C}\left(x_{h_i}, x_{h_j}\right) = Cos\,(\theta) = \frac{x_{h_i} \cdot x_{h_j}}{\left\|x_{h_i}\right\| * \left\|x_{h_j}\right\|}$$

که در رابطه‌ی بالا · نماد ضرب داخلی بین دو بردار x_{h_i} و x_{h_j} است که در بخش فاصله‌ی کُساین تعریف شد. همچنین θ هم زاویه‌ی بین این دو بردار است. در نهایت معیار شباهت دو بردار x_{h_i} و x_{h_j} به کمک فاصله‌ی کساین یعنی به صورت زیر تعریف می‌شود [۱۷, ۱۸]:

(۱۳)

$$\mathrm{d_C}\left(x_{h_i}, x_{h_j}\right) = \cos(\theta) = \frac{\sum_{k=1}^{\mathrm{K}} x_{h_i}^k * x_{h_j}^k}{\sqrt{\sum_{k=1}^{K} x_{h_i}^k}\sqrt{\sum_{k=1}^{K} x_{h_j}^k}}$$

۴_۳ روش پیشنهادی برای تعیین توابع درهم‌ساز مستقل

برای صحبت با اطمینان بیشتر در مورد کاهش برخوردهای اتفاقی، می‌توان تعدادی مجموعه از توابع درهم‌ساز تعریف کرد که هر مجموعه شامل w تابع درهم‌ساز مجزا است. این مجموعه را M می‌نامیم که شامل m مجموعه تابع درهم‌ساز است که هر مجموعه شامل w تابع درهم‌ساز است. این مجموعه به عنوان ورودی الگوریتم به صورت تصادفی از اعداد اول موجود در رابطه‌های توابع درهم‌ساز تولید می‌شود.

(۱۴)

$$M = \{\{h_{11}, \dots, h_{1w}\}, \dots, \{h_{i1}, \dots, h_{iw}\}, \dots, \{h_{m1}, \dots, h_{mw}\}\}$$

سپس با استفاده از معیارهای فاصله‌ی تعریف شده، برای هر یک از مجموعه‌های موجود در M فاصله‌ی بین دوبه دوی توابع درهم‌ساز موجود در آن را به کمک روش رابطه (۱۹-۳) محاسبه کرده و میانگین این فاصله‌ها را به عنوان فاصله‌ی نهایی برای آن مجموعه یا میزان عدم وابستگی توابع درهم‌ساز آن مجموعه در نظر می‌گیریم. پس اگر فرض کنیم که مجموعه‌ی توابع درهم‌سازی کاندید i-ام در مجموعه‌ی اصلی M به صورت رابطه (۱۵) باشد:

(۱۵)

$$M_i = \{h_{i1}, \dots, h_{iw}\}$$

معیار عدم وابستگی بین توابع درهم‌ساز موجود در مجموعه‌ی M_i برابر با میانگین شباهت دوبه‌دوی توابع درهم‌ساز است و به صورت رابطه (۱۶) تعریف می‌شود:

(۱۶)

$$Dependence(M_i) = \left\{\frac{\sum_{j=1}^{w}\sum_{k=j+1}^{w} sim(h_{ij}, h_{ik})}{\frac{w*(w-1)}{2}}\right\}$$

که در (۱۶) تابع $sim(h_{ij}, h_{ik})$ همان تابع محاسبه‌ی فاصله‌ی بین دو تابع درهم‌ساز است که در (۷)، () و () تعریف شد. همچنین w تعداد توابع درهم‌ساز مجزای موجود در روش gMatrix است. در انتها با استفاده از مجموعه‌ی توابعی که کمترین وابستگی را نسبت به یکدیگر دارند می‌توانیم بهترین مجموعه‌ی توابع درهم‌ساز را برای gMatrix به صورت رابطه (۱۷) انتخاب کنیم:

(۱۷)

$$BestHashSet(M) = \arg\min_i\{Dependence(M_i)\}$$

با این روش می‌توانیم با اطمینان بیشتری در مورد کاهش برخوردهای اتفاقی در توابع درهم‌سازی در هنگام ساخت gMatrix پیش رویم. در این راستا، الگوریتم ارائه شده برای یافتن توابع درهم‌ساز مستقل از یکدیگر دارای بخش‌های اصلی زیر است:

۱. *مجموعه‌ی M شامل m مجموعه‌ی wتایی از توابع درهم سازی مجزا بساز*
۲. *برای هر مجموعه از توابع درهم‌سازی ساخته شده مثل M_i که $i = 1 \dots m$ انجام بده:*
   أ. میزان عدم وابستگی توابع درهم‌ساز عضو را از (۱۶۶) محاسبه کن.
۳. *بهترین مجموعه از M را به کمک (۱۷) به عنوان M_b انتخاب کن.*
۴. *از مجموعه‌ی توابع درهم‌ساز M_b در ساخت نهایی $gMatirx$ استفاده کن.*

۴. ارزیابی روش پیشنهادی

کلیه‌ی آزمایش‌های انجام شده در این بخش با کمک یک کامپیوتر شخصی با حافظه‌ی اصلی ۸ گیگابایت، پردازنده‌ی Intel Core i۵ ۲,۷ GHz، روی سیستم عامل مک، انجام شده‌اند. در این پژوهش الگوریتم‌ها توسط زبان برنامه‌نویسی C++ پیاده‌سازی شده‌اند که توضیح کلی روند پیاده‌سازی را در این قسمت خواهیم داشت. توجه داشته باشید که برای هر مرحله برنامه‌ای جداگانه نوشته شده است که شامل برنامه‌ای برای الحاق فرکانس به مجموعه داده فرندستر، برنامه‌ای برای ساخت خلاصه گراف gMatrix از روی جریان داده، برنامه‌ای برای ساخت پایگاه داده‌ی کلی شامل همه اطلاعات مجموعه داده، سه برنامه برای تست و مقایسه سه پرس و جوی مختلف روی داده واقعی و خلاصه gMatrix، می‌باشند.

۴_۱ مجموعه‌های داده‌ی مورد آزمایش

در این پژوهش از مجموعه داده شبکه اجتماعی فرندستر[۱۴] که از سایت[۱۵] دانشگاه استنفورد موجود می‌باشد، به دو شکل متفاوت استفاده شده است و در ادامه به جزییات این دو شکل متفاوت داده خواهیم پرداخت. فرندستر یک شبکه بازی آنلاین است. قبل از راه اندازی مجدد به عنوان یک وب سایت بازی، فرندستر یک سایت شبکه اجتماعی بود که در آن کاربران می‌توانستند یال دوستی با هم تشکیل دهند. همچنین شبکه اجتماعی فرندستر به کاربران اجازه می‌داد گروه تشکیل دهند که اعضای دیگر نیز می‌توانستند به آن ملحق شوند. ما چنین گروه‌های تعریف شده توسط کاربر را به عنوان جوامع حقیقی می‌شناسیم. برای شبکه اجتماعی، ما زیرگراف القا شده از گره‌هایی که به حداقل یک اجتماع متعلق هستند یا به گره‌های دیگر که آنها متعلق به حداقل یک جامعه هستند را برمی‌داریم. این داده‌ها توسط پروژه بایگانی وب[۱۶] ارائه شده است، جایی که نمودار کامل موجود است. اطلاعات تکمیلی این مجموعه داده در جدول۱) آورده شده است [۲۱, ۲۰, ۱۹].

جدول۱. اطلاعات و خصوصیات مجموعه داده فرندستر

تعداد گره‌ها	۶۵۶۰۸۳۶۶
تعداد یال‌ها	۱۸۰۶۰۶۷۱۳۵
تعداد گره‌ها در بزرگترین مولفه ضعیفا همبند[۱۷]	۶۵۶۰۸۳۶۶
تعداد یال‌ها در بزرگترین مولفه ضعیفا همبند	۱۸۰۶۰۶۷۱۳۵
تعداد گره‌ها در بزرگترین مولفه قویا همبند[۱۸]	۶۵۶۰۸۳۶۶
تعداد یال‌ها در بزرگترین مولفه قویا همبند	۱۸۰۶۰۶۷۱۳۵
ضریب خوشه‌بندی[۱۹] میانگین	۰,۱۶۲۳
تعداد مثلث‌ها	۴۱۷۳۷۲۴۱۴۲
کسر مثلث‌های بسته	۰,۰۰۵۸۵۹
قطر (طولانی‌ترین مسیر کوتاه[۲۰])	۳۲
قطر موثر ۹۰-درصدی[۲۱]	۵,۸

باید توجه داشت که این مجموعه داده بدون جهت می‌باشد و همچنین فاقد فرکانس برای هر یال می‌باشد. درواقع این مجموعه داده فقط شامل گره شروع و گره پایان هر یال می‌باشد. در نتیجه برای نسبت دادن فرکانس به این مجموعه داده از توزیع زیفان استفاده شده که استفاده از این توزیع در

[۱۴] Friendster
[۱۵] Snap.standford.edu
[۱۶] Web Archive Project
[۱۷] Weakly connected component (WCC)
[۱۸] Strongly connected component (SCC)
[۱۹] Clustering
[۲۰] Longest shortest path
[۲۱] ۹۰-percentile

ادبیات جریان داده [۲۲, ۱] به یک عمل استاندارد تبدیل شده است. همچنین در تولید فرکانس برای یال‌های مجموعه داده از دو ضریب انحراف[۲۲] مختلف $s = ۱.۰$ و $s = ۱.۴$ استفاده شده است.

شایان ذکر است که دلیل استفاده از مجموعه داده فرندستر با علم به عدم وجود فرکانس در این مجموعه این بوده است که با الحاق مصنوعی فرکانس به این مجموعه داده ما می‌توانیم نتایج خود را برای ضریب انحراف‌های مختلف تست و بررسی کنیم. از طرف دیگر، همانطور که در گزارش شده است، بیشتر مجموعه داده‌های واقعی ضریب انحراف مشترکی در بازه استفاده شده توسط ما را دارند. [۱۶, ۱۷]در جدول) ابعاد مجموعه داده استفاده شده نمایش داده شده است. که حجم جریان[۲۳] نمایانگر حجم جریان گراف با تکرار یال‌ها می‌باشد.

جدول ۲. ابعاد و خصوصیات مجموعه داده فرندستر

حجم جریان	بیشترین فرکانس یال	فرکانس تجمعی جریان	یال‌ها	گره‌ها
۸۰ گیگابایت	۴.۴۳ $*۱۰^{۸}(s = ۱.۰)$ ۱*۳,۲۲۰ $^{۹}(s = ۱.۴)$	$L = ۱۰^{۱۰}$	۱,۸ میلیارد	۶۶ میلیون

۴_۲ افزودن فرکانس به مجموعه داده

در ابتدای امر به جهت آماده‌سازی مجموعه داده فرندستر باید برای هر یال یک فرکانس با استفاده از توزیع زیفان تولید کنیم [۲۳]. به همین منظور برنامه‌ای نوشته شده است تا گره شروع و گره پایان هر یال را از روی فایل مجموعه داده فرندستر بخواند، سپس با استفاده از رابطه کلی توزیع زیفان که در رابطه (۱۸) آورده شده است فرکانس مربوطه را محاسبه کند و به همراه گره شروع و گره پایان یال در یک فایل مجزا به عنوان خروجی چاپ نماید. اگر N را تعداد عناصر در نظر بگیریم و همچنین k را مرتبه هر عنصر و s مقدار ضریب انحراف باشد، رابطه کلی توزیع زیفان به صورت زیر می‌باشد:

(۱۸)

$$f(k;s,N) = \frac{۱/k^{s}}{\sum_{n=۱}^{N}(۱/n^{s})}$$

ما برای هر عنصر مقدار محاسبه شده از توزیع زیفان را در $۱۰^{۱۰}$ ضرب می‌کنیم و مقدار سقف نتیجه را به عنوان فرکانس آن عنصر در نظر می‌گیریم. بدین صورت از آنجایی که جمع همه مقادیر تولید شده توسط توزیع زیفان برابر با مقدار ۱ خواهند بود، جمع تمام فرکانس‌های عناصر تولید شده توسط ما نیز $۱۰^{۱۰} \approx$ خواهد بود. توجه داشته باشید که مرتبه عناصر به جهت سادگی، طبق ترتیب ورودی آنها در نظر گرفته شده است. همچنین با توجه به مراحل بالا، دو مجموعه داده مختلف از روی مجموعه داده فرندستر به ازای ضرایب انحراف $s = ۱.۰$ و $s = ۱.۴$ ساخته شد.

۴_۳ انتخاب بهترین بردار شباهت

قبل از انجام عملیات خلاصه‌سازی جریان به کمک روش gMatrix باید یک مجموعه تابع درهم‌ساز مناسب انتخاب گردد [۲۴]. در این مرحله با انتخاب یک بردار شباهت از روش‌های مطرح‌شده‌ی فاصله‌ی اقلیدسی، شباهت کُساین و اطلاعات متقابل یک روش انتخاب می‌شود و بعد از انجام الگوریتم پیشنهادی مجموعه‌ی توابع درهم‌ساز مناسب مشخص می‌گردد. سپس به کمک این مجموعه مراحل بعدی انجام می‌شود که همان ساخت خلاصه‌ی گراف و پاسخ به پرس و جوها و ارزیابی دقت پاسخ‌گویی است. برای انتخاب از

[۲۲] Skew

[۲۳] Stream Size

بین سه معیار فاصله‌ی معرفی شده، بعد از انجام چندین اجرا و مقایسه‌ی نتایج، بهترین روش به عنوان معیار معرفی می‌گردد. برای ساخت بردار شباهت در همه‌ی آزمایش‌ها متغیر $K = ۱۰۰۰$ در نظر گرفته شده است.

۴_۴ ارزیابی عملکرد روش پیشنهادی

برای ارزیابی یک سیستم خلاصه سازی جریان گراف در مراجع تعدادی پرس و جو تعریف می‌شود. برای ارزیابی عملکرد روش gMatrix، تعداد ۶ پرس و جوی مختلف تعریف شده است. [۲] از این ۶ پرس و جو، دو پرس و جوی یافتن فرکانس یک یال و یافتن یال‌های پرتکرار به عنوان پایه‌ای برای سایر پرس و جوها استفاده شده است. بنابراین علاوه بر پرس و جوی اصلی مورد تاکید در این پژوهش(یافتن زیرگراف‌های پر تکرار) برای ارزیابی عملکرد روش پیشنهادی این دو پرس و جو هم در نظر گرفته شده است.

برای ارزیابی دو آزمایش مستقل در نظر گرفته شده است. همان‌طور که در بخش ساخت داده گفته شد، در این دو آزمایش متغیر ضریب انحراف توزیع زیفان متفاوت است. همچنین سایر متغیرهای ثابت در این دو آزمایش در

جدول۳. مقدار متغیرهای مشترک در آزمایش‌ها

	ضریب انحراف توزیع زیفان	طول بردار سنجش شباهت توابع درهم‌ساز	تعداد مجموعه توابع درهم‌ساز	تعداد توابع درهم ساز	بُرد تابع درهم ساز
نماد	s	K	m	w	$\boldsymbol{h}$
آزمایش ۱	۱,۰	H	۱۰۰۰	۱۰	**۱۰۰۰**
آزمایش ۲	۱,۴	H	۱۰۰۰	۱۰	**۱۰۰۰**

جدول) آمده است. به علت اینکه طول جدول درهم‌سازی در gMatrix برابر با h=۱۰۰۰ است، طول بردار سنجش شباهت توابع درهم‌ساز هم برابر با این مقدار قرار می‌گیرد. زیرا خروجی توابع درهم‌ساز در بازه‌ی $[۰, h]$ قرار دارد بنابراین انتخاب $K = h$ می‌تواند باعث پوشش بیشتر فضای مسئله گردد.

۴_۴_۱_۱ پرس و جوی یافتن فرکانس یک یال

برای این پرس و جو یک برنامه نوشته شده است که ابتدا خلاصه gMatrix، عدد اول P و توابع درهم ساز را که روی فایل ذخیره شده‌اند می‌خواند و بازسازی می‌کند. سپس ۱۰۰۰۰ یال پرتکرار گراف اصلی را پیدا می‌کند و تعداد ۱۰۰ یال از آنها را به صورت تصادفی انتخاب و فرکانس تخمینی آنها را محاسبه می‌کند. در ادامه با پیدا کردن فرکانس واقعی همین یال‌ها از پایگاه داده ساخته شده، واریانس خطای gMatrix را در برآورد کردن فرکانس این یال‌ها را طبق (۱۹) محاسبه می‌کند. در این رابطه تعداد نمونه‌ها یا n برابر

تعداد ۱۰۰ یال، μ میانگین فرکانس‌ها و x_i فرکانس بدست آمده توسط هر روش است. مقدار خطای این واریانس برای دو آزمایش مطرح‌شده و به کمک هر یک از معیارهای فاصله‌ی مطرح شده، در

جدول) آمده است.

(۱۹)

$$Variance\ of\ Error = \frac{۱}{۱۰۰}\sum_{i=۱}^{۱۰۰}(x_i - \mu)^۲$$

جدول ۴. واریانس خطای یافتن فرکانس یک یال

روش	آزمایش ۱	آزمایش ۲
فاصله‌ی اقلیدسی	۱۲۹۵	**۸۶۰**
اطلاعات مشترک	۱۲۸۶	**۸۶۳**
شباهت کُساین	۱۲۸۷	**۸۶۰**
gMatrix	۱۲۹۰	**۸۵۵**

لازم به ذکر است که برای یافتن فرکانس تخمینی هر یال، همه یال‌های درهم ساز مربوطه در هر w تابع درهم ساز را بدست می‌آوریم و بین مقادیر ذخیره شده در این خانه‌ها کوچکترین مقدار را انتخاب می‌کنیم که نزدیک‌ترین مقدار به مقدار واقعی یال در بین آنها می‌باشد. همچنین از آنجایی که در همه روش‌ها یک الگوریتم یکسان استفاده شده است و صرفا خلاصه ساخته شده متفاوت می‌باشد، در نتیجه از لحاظ مرتبه زمانی همه یکسان هستند و نیازی به مقایسه نمی‌باشد [۲۵].

۴_۴_۱_۲ مقایسه و ارزیابی نتایج

همانطور که در
جدول) نمایش داده شده است، واریانس خطا در هر آزمایش، فارغ از نحوه انتخاب توابع درهم‌ساز، مقادیر یکسانی دارد. با توجه به اینکه در نحوه ذخیره‌سازی فرکانس یال‌ها در gMatrix و همچنین پرس و جوی آن تغییری ایجاد نشده است، این انتظار را داشتیم که جواب این پرس و جو برای همه روش‌های انتخاب تابع درهم ساز روی یک مجموعه داده یکسان، جوابی مشابه و با دقت یکسانی به دست بدهد.
از نظر ریاضی علت این است که gMatrix برای یافتن فرکانس یک یال، مقدار درون w جدول درهم‌ساز را بررسی می‌کند و از بین آنها کمترین مقدار را به عنوان فرکانس یال درخواست شده در نظر می‌گیرد. با این تعریف انتخاب توابع درهم‌ساز مستقل یا وابسته تغییر در نتایج این پرس و جو نخواهد داشت. در نهایت از بین مقادیر درون w جدول درهم‌ساز در نهایت مقدار یکی از جدول‌ها به عنوان فرکانس یال انتخاب می‌شود. که این مقدار به مقدار خانه‌های جدول‌های دیگر ارتباطی ندارد. به همین علت در این پرس و جو با هر مجموعه‌ی توابع درهم‌ساز ورودی دقت نهایی یافتن فرکانس یک یال تغییر چندانی نخواهد کرد. همچنین از نظر زمان پاسخ‌گویی به این پرس و جو هم به علت عدم تغییر در الگوریتم یافتن خانه‌ی موجود در تابع درهم‌ساز، تغییری بین روش gMatrix و روش پیشنهادی این پژوهش وجود ندارد و زمان پاسخ‌گویی به این پرس و جو در هر دو روش دقیقا به یک اندازه است.

۴_۴_۲_۱ پرس و جوی یافتن یال‌های پر تکرار

برای این پرس و جو نیز یک برنامه نوشته شده است که با توجه به فرکانس تجمعی همه یال‌های جریان گراف که $L = ۱۰^{۱۰}$ می‌باشد، پرس و جو را اجرا می‌کند. برای ارزیابی دقت این پرس و جو از کسر یک ده هزارم از فرکانس تجمعی یعنی $۱۰^{۶}$ می‌باشد، به عنوان F در نظر گرفته می‌شود. سپس با این مقدار F تمام یال‌های پر تکرار یال‌ها را با استفاده از خلاصه gMatrix محاسبه می‌شود و در نهایت با بررسی فرکانس هر کدام از پایگاه داده‌ی کلی، مقدار نرخ مثبت کاذب در جواب این پرس و جو را به عنوان پارامتر اصلی ارزیابی دقت در نظر گرفته می‌شود. همچنین زمان پاسخ‌گویی به

پرس و جوها به همراه حافظه‌ی مصرفی روش پیشنهادی در مقایسه با روش gMatrix به عنوان پارامترهای دیگری برای ارزیابی کارآیی محاسبه می‌شود. اما با توجه به اثبات مرتبه‌ی زمانی برای روش ارائه شده برای ارزیابی کارآیی از نظر مرتبه‌ی زمانی، باید از معیارهای مختلف برای آستانه‌ی F استفاده کرد زیرا مرتبه‌ی زمانی به این معیار وابستگی دارد. برای این پرس و جو دو جدول (۵) و (۶) در ادامه آمده است که به ترتیب نمایانگر دقت الگوریتم (نرخ مثبت کاذب) و دیگری نمایانگر زمان اجرای هر کدام از روش‌ها می‌باشد که به تفکیک هر روش و آزمایش برای مجموعه داده به نمایش گذاشته شده‌اند.

۴_۴_۲_۲ نرخ مثبت کاذب

آزمایش‌هایی برای یافتن یال‌های پرتکرار که تکرار بالاتر از یک آستانه مشخصی داشتند انجام داده ایم. مقدار این آستانه تکرار را با درصدی از تکرار جریان بیان می‌کنیم، زیرا خطای تخمین تکرار یک یال نسبت به جمع تکرار جریان حساس می‌باشد. gMatrix تضمین می‌کند که مجموعه یال‌های گزارش شده مجموعه‌ای شامل یال‌های واقعی می‌باشد. نرخ مثبت کاذب را برای پرس و جوهای یال‌های پرتکرار به این صورت بیان می‌شود:

نرخ مثبت کاذب = تعداد یال‌هایی که به اشتباه به عنوان یال‌های پرتکرار گزارش شده‌اند تقسیم بر تعداد درست یال‌های پرتکرار، طبق ():

(۲۰)

$$False\_Positive\ Rate = \frac{\#Number\ of\ edges\ Incorrectly\ reported\ as\ Heavy\ Hitters}{\#Number\ of\ reported\ edges\ as\ Heavy\ Hitters}$$

گرچه، این نرخ مثبت کاذب کاهش داده شده به قیمت حافظه و زمان اجرای بالاتر به دست آمده است. این پرس و جو را یکبار با استفاده از الگوریتم پیشین یافتن یال‌های پر تکرار و روی gMatrix ساخته شده بدون بهبود توابع درهم‌ساز اجرا و نتایج را یادداشت کردیم. بار دیگر از الگوریتم پیشنهادی بهبود یافته برای یافتن یال‌های پر تکرار و روی gMatrix ساخته شده به همراه بهبود یافته توابع درهم‌ساز استفاده کردیم و مجددا نتایج را یادداشت کردیم. مقایسه این دو آزمایش با روش gMatirx از نظر مقدار نرخ مثبت کاذب (دقت) در

جدول ۲) آورده شده است.

جدول ۲. نرخ مثبت کاذب در یافتن یال‌های پرتکرار برای هر آزمایش با آستانه‌ی $F = 10^6$

آزمایش/ روش	آزمایش ۱: $S = 1.0$	آزمایش ۲: $S = 1.4$
فاصله اقلیدسی	۰,۰۱۳	۰,۰۰۹
اطلاعات مشترک	۰,۰۱۱	۰,۰۰۳
شباهت کساین	**۰,۰۰۷**	**۰,۰۰۱**
gMatrix	۰,۰۱۲	۰,۰۰۴

۴_۴_۲_۳ مقایسه و تحلیل نرخ مثبت کاذب

در

جدول ۲) که نمایانگر نرخ مثبت کاذب می‌باشد، دیده می‌شود که در هر دو آزمایش (دو مجموعه داده متفاوت) میزان نرخ مثبت کاذب در روش شباهت کساین از روش gMatrix کمتر شده است. بدین ترتیب در آزمایش اول به میزان ۴۱ درصد و در آزمایش دوم به میزان ۷۵ درصد افزایش یافته است. همچنین روش فاصله‌ی اقلیدسی نمی‌تواند بهبود خاصی در نتایج ایجاد کند و حتی نتایج آن از خود gMatrix هم

بدتر است. در مورد روش اطلاعات مشترک هم نتایج به دست آمده به رغم داشتن برتری جزئی نسبت به روش gMatirx به این روش بسیار نزدیک هستند. علت این است که روش اطلاعات مشترک برای این مسئله در یافتن احتمال رخداد یک واقعه ضعیف عمل می‌کند. زیرا اطلاعات کافی برای یافتن میزان احتمال یک رخداد مثل $p(x = x_i)$ را در دست نداریم. در نهایت از بین این سه بردار فاصله بهترین نتایج را روش فاصله‌ی کساین ارائه می‌دهد. دلیل برتری فاصله‌ی کُساین نسبت به سایر روش‌ها این است که این فاصله سعی می‌کند بردارها را از نظر زاویه‌ی بین آنها شباهت سنجی کند و به مقادیر هر یک از ابعاد یک بردار وابسته نیست [۲۶].

در این پژوهش، نشان دادیم که استفاده از فاصله‌ی کُساین بر روی توابع درهم سازی برای یافتن میزان شباهت و سازگاری آن‌ها از کارایی بالایی برخوردار می‌باشد، زیرا:

- در توابع درهم سازی مورد استفاده در این مطالعه، بزرگی مقدار عدد خروجی تابع درهم سازی در هنگامی که این اعداد بزرگتر از حد بالایی مانند F باشند، از اهمیت یکسانی برخوردار می‌باشند. به طور مثال، x_1 و x_2 دو خروجی از تابع درهم سازی ۱ و ۲ می‌باشند. با فرض اینکه $x_1 > F$ و $x_2 > F$ و $x_1/x_2 = 10^3$ در گام‌های بعدی الگوریتم ارائه شده بر پایه ی توابع درهم سازی، اهمیت x_1 و x_2 یکسان در نظر گرفته خواهد شد (و نه 10^3برابر). حال می‌توان به اصلی ترین ویژگی فاصله ی کُساین اشاره نمود. اهمیت زاویه‌ی ساخته شده در بردارها و نه اندازه ی آن‌ها. از این ویژگی می‌توان به عنوان اثباتی برای کارایی فاصله ی کُساین در مساله‌ی ارائه شده یاد کرد.
- پیچیدگی زمانی مناسب فاصله ی کُساین با توجه به محدودیت‌های زمانی تعریف شده در مساله
- ارایه ی نتایج قابل قبول در مقایسه با دیگر روش‌های یافتن فاصله در هنگام کار با بردارهای تنک.

۴_۴_۳_۱ زمان پاسخ به پرس و جو

در

جدول) برای هر یک از روش‌ها میانگین زمان پاسخ به پرس و جوهای یافتن یال‌های پرتکرار نمایش داده می‌شود. در جدول قابل مشاهده است زمان پاسخ‌گویی به پرس و جو ارتباطی به معیار فاصله‌ی شباهت استفاده شده در الگوریتم ندارد اما به علت بهینه‌سازی استفاده شده به کمک داده ساختار درخت قرمز-سیاه این زمان در روش پیشنهادی پژوهش نسبت به روش gMatrix بسیار کمتر است. این نتایج با توجه به بحث مرتبه‌ی زمانی مطرح شده قابل پیش بینی هستند و محاسبات را تایید می‌کند.

جدول ۶. زمان پاسخ به پرس و جوی یال‌های پرتکرار برای هر آزمایش بر حسب ثانیه

مقدار آستانه	$F = 10^6$	$F = 10^7$	$F = 10^8$			
آزمایش/روش	$S = ۱.۰$	$S = ۱.۴$	$S = ۱.۰$	$S = ۱.۴$	$S = ۱.۰$	$S = 1.4$
بهینه‌سازی به کمک داده ساختار درخت قرمز-سیاه	۷۹۴	۳۹۱	۳۱	۳۰	۱	۴
gMatrix	۱۰۵۶۲	۷۰۷۱	۶۱۲	۵۴۹	۹	۲۸

۴_۴_۳_۲ مقایسه و تحلیل زمان پاسخ به پرس و جو

روش پیشنهادی همان‌طور که در جداول نتایج مشاهده می‌گردد روش پیشنهادی به کمک داده ساختار درخت قرمز-سیاه زمان پاسخ به پرس و جوها نسبت به روش gMatrix بهبود یافته است. به علت مشابه نبودن شرایط اجرای آزمایش با سیستم اجرای آزمایشات در این پژوهش، همه‌ی آزمایشات مقایسه‌ی مرتبه‌ی زمانی و زمان پاسخ‌گویی به پرس و جوها، برای روش gMatrix هم روی سیستم مشابه این پژوهش انجام شده است. [۲]بنابراین زمان اجرای گزارش شده برای روش gMatrix، برای اجرا آن روی محیط شبیه‌سازی مشابه با روش پیشنهادی و با شرایط جریان گراف کاملا مشابه است.

با توجه به اثبات مرتبه‌ی زمانی روش پیشنهادی در بخش ۰ انتظار می‌رود که این بهبود تقریبا یک حالت لگاریتمی باشد. زیرا مرتبه‌ی زمانی اصلی روش gMatrix برابر است با ضرب ضرایب c_k اما در روش پیشنهادی ما این مرتبه برابر با جمع ضرایب c_k است. بهبود نمایی در زمان پاسخ به پرس و جو یافتن یال‌های پرتکرار از مقایسه‌ی نتایج گرفته شده در

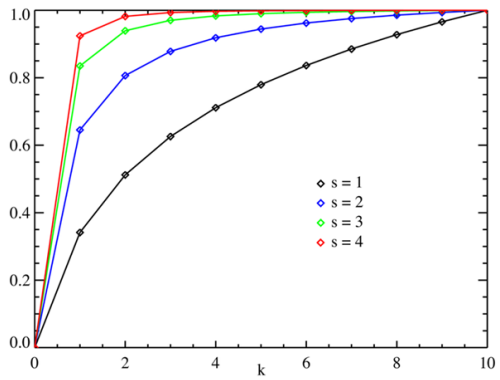
جدول) قابل مشاهده است.

جدول۷. تعداد یال‌های پرتکرار با ضریب انحراف برای مقدار آستانه‌ی مختلف

مقدار آستانه	$F = 10^6$		$F = 10^7$		$F = 10^8$	
ضریب انحراف توزیع زیفان	S = ۱.۰	S = ۱.۴	S = ۱.۰	S = ۱.۴	$S = ۱.۰$	S = **1.4**
تعداد یال‌های پرتکرار	۸۶۳۱۶	۴۳۷۵	۴۵	۶۱	۴	۱۱

در جدول) مربوط به نتایج زمان اجرای پرس و جو ها، زمان پاسخ به پرس و جو برای مقادیر تعیین شده برای S در توزیع زیفان، با بالا رفتن ضریب انحراف، تعداد یال‌ها با فرکانس بالا کمتر می‌شود اما فرکانس خود این یال‌ها بسیار بیشتر خواهد بود. اما با صریب انحراف کمتر، تعداد یال‌های پرتکرار بیشتری خواهیم داشت. تعداد این یال‌ها برای مقدار آستانه‌ی F و برای مقدار ضریب انحراف توزیع زیفان در جدول) آمده است.

توجیه اینکه زمان پاسخ به پرس و جوی یافتن یال‌های پرتکرار در هر دو روش برای مقدار آستانه‌ی فرکانس $F = ۱۰^۶$ برای مقدار ضریب انحراف $S = ۱.۰$ از مقدار ضریب انحراف $S = ۱.۴$ بیشتر است همین امر است. زیرا همان‌طور که شکل) نشان می‌دهد، وقتی ضریب انحراف بیشتر باشد یال‌های پرتکرار کم ولی وزن‌های خیلی بالا دارند. اما وقتی ضریب انحراف کمتر باشد تعداد یال‌های پرتکرار بیشتر است اما یال‌ها وزن‌های بسیار بالایی ندارند.



شکل ۳. احتمال فرکانس یال‌ها برای ضریب انحراف‌های مختلف در توزیع زیفان

به عبارت دیگر وقتی ضریب انحراف کمتر باشد توزیع زیفان میزان وزن یال کلی L را در بین تعداد زیادتری از یال‌ها توزیع می‌کند. به همین علت وقتی فرکانس برابر با $F = ۱۰^۶$ است، برای ضریب انحراف $S = ۱.۴$ تعداد یال‌های پرتکرار کمتری داریم. اما وقتی ضریب انحراف $S = ۱.۰$ باشد تعداد یال‌های پرتکرار برای فرکانس $F = ۱۰^۶$ بیشتر است. در نهایت هنگامی میزان فرکانس به فرکانس تجمعی جریان گراف در

توزیع زیفان یعنی $L = ۱۰^{۱۰}$ نزدیک‌تر می‌شود، یعنی $F = ۱۰^{۷}$ و یا $F = ۱۰^{۸}$، تعداد یال‌های پرتکرار برای ضریب انحراف کمتر یعنی $S = ۱.۰$ کمتر می‌شود. اما برای ضریب انحراف بیشتر یعنی $S = ۱.۴$ به علت بالابودن وزن یال‌های ایجاد شده، تعداد یال‌های پرتکرار در این حالت بیشتر خواهد بود.

۵. نتیجه گیری و راهکارهای آتی

هدف از این پژوهش بهبود کارائی یافتن یال‌های پرتکرار در مسئله‌ی جریان گراف بود. برای حل این مسئله یک روش مبتنی بر توابع در هم ساز به نام gMatrix مطرح شده است که دارای مشکلاتی از جمله پیچیدگی زمانی بالا و دقت پایین در پاسخ به پرس و جوهای مطرح در مسئله‌ی جریان گراف است. با بررسی دقیق مراحل و الگوریتم‌های ارائه شده در روش gMatrix، دلایل این دو مشکل و راه حل‌های آنها به شرح زیر می‌باشد:

۱- روش gMatrix برای یافتن یال‌های پرتکرار دارای یک مرحله‌ی ابهام زدایی است که در این مرحله معکوس مقدار توابع درهم ساز برای یک نمایه از هر تابع در نظر گرفته می‌شود و در انتها این معکوس‌ها با یکدیگر اشتراک داده می‌شوند تا یال‌های نهایی پرتکرار بدست آیند. برای بخش اشتراک گیری بین معکوس نمایه‌های توابع در هم ساز، خود روش gMatrix راه‌حلی با مرتبه‌ی زمانی خطی ارائه کرده است. و این مرتبه‌ی زمانی به علت زیاد بودن تعداد نمایه‌های معکوس شده، می‌تواند باعث کندی فرایند پاسخ به پرس و جوها شود.

۲- راه حل ارائه شده برای حل این مشکل استفاده از داده‌ساختارها برای کاهش مرتبه‌ی زمانی مرحله‌ی ابهام زدایی در روش gMatrix برای پاسخ به پرس و جوهای زیر گراف‌های پر تکرار و یال‌های پرتکرار، بود. در این پژوهش از داده ساختار درخت قرمز-سیاه برای یافتن اشتراک بین نمایه‌های معکوس شده با مرتبه‌ی زمانی لگاریتمی استفاده شد.

۳- نتایج آزمایش‌های تجربی روی داده‌ها مسئله‌ی جریان گراف بهبود زمان پاسخ به پرس و جوهای مطرح شده را از حالت خطی به حالت لگاریتمی نشان می‌دهد. اما در ازای این کاهش مرتبه‌ی زمانی، به علت استفاد از داده ساختار ها، مرتبه‌ی حافظه‌ی مورد استفاده در روش پشنهادی دو برابر روش gMatrix شده است.

۴- مشکل دوم روش gMatrix دقت پایین در پاسخ به پرس و جوی یافتن زیرگراف‌های پر تکرار و یال‌های پرتکرار است. با بررسی دقیق‌تر روش gMatrix دریافت می‌شود که در حین یافتن معکوس نمایه‌های توابع درهم‌سازی هرچه توابع درهم‌ساز مستقل تر از یکدیگر باشند باعث می‌شود که تفاوت الگوی استفاده از تمام خانه‌های جدول درهم‌ساز بیشتر شود و برعکس هرچه توابع وابسته تر باشند احتمال اینکه تابع‌های درهم‌ساز از یک الگوی مشابه برای نگاشت راس‌ها به خانه‌های خود استفاده کنند بیشتر است و این مسئله باعث کاهش دقت روش gMatrix شده است. زیرا این روش برای بهبود روش طرح count-min به جایw جدول درهم سازی یک بعدی، w جدول درهم‌سازی دو بعدی را به طور همزمان استفاده می‌کند.

۵- برای حل مشکل توابع درهم ساز وابسته در روش gMatrix از روشی استفاده شد تا بتوان شباهت دو تابع درهم‌ساز را به‌دست آورد. برای این منظور با نگاشت فضای مسئله به یک فضای برداری، برای محاسبه‌ی عدم شباهت دو تابع درهم ساز، از معیارهای فاصله‌ی اقلیدسی، اطلاعات مشترک و شباهت کساین استفاده شد. با استفاده از این معیارهای فاصله می‌توان با در نظرگرفتن خروجی هر یک از توابع درهم ساز در یک فضای برداری میزان وابستگی الگوی آنها را بدست آورد و توابع درهم سازی با وابستگی کمتر را به عنوان ورودی روش gMatrix استفاده کرد.

۶- نتایج آزمایش‌های تجربی روی داده‌ی مسئله نشان می‌دهند که استفاده از معیارهای فاصله و شباهت مطرح شده می‌تواند باعث افزایش دقت یا کاهش نرخ مثبت کاذب در پاسخ به پرس و جوهای زیر گراف‌های پر تکرار و یال‌های پرتکرار شود. همچنین نتایج نشان می‌دهد که از بین سه روش مطرح شده برای سنجش وابستگی تابع‌های درهم‌ساز روش شباهت کساین بهترین نتایج را در بر دارد. زیرا ماهیت آن طوری است که به جای توجه به خود مولفه‌ی‌ها بردار مثل روش اطلاعات مشترک و یا توجه به اندازه‌ی بردارها مثل روش فاصله‌ی اقلیدسی، به زاویه‌ی بین دو بردار توجه دارد.

۷- از منظر میزان بهبود نتایج بدست آمده، روش پیشنهادی به کمک داده ساختار درخت قرمز-سیاه زمان پاسخ به پرس و جوها نسبت به روش gMatrix بسیار بهبود یافته است، بیش از ۱۰ برابر، البته باید این نکته ذکر شود که پیاده سازی gMatrix توسط خود ما انجام شده است. در مورد نرخ مثبت کاذب نیز، میزان نرخ مثبت کاذب در روش شباهت کساین(بهترین جواب در بین روشهای پیشنهادی) از روش gMatrix کمتر شده است. بدین ترتیب در آزمایش اول به میزان ۴۱ درصد و در آزمایش دوم به میزان ۷۵ درصد بهبود بدست آمده است

۵_۱ کارهای آتی

برای کارهای آتی و مرتفع کردن مشکلات این پژوهش راه‌حل‌هایی مدنظر است از جمله:

- استفاده از داده‌ساختارهای دیگر که از نظر مرتبه‌ی زمانی مشابه با درخت قرمز-سیاه عمل کنند اما از نظر مرتبه‌ی حافظه‌ی مورد نیاز نسبت به درخت قرمز-سیاه حافظه‌ی کمتری را نیاز داشته باشند. مثلا روش‌های پوشش بیتی[۲۴]، یا روش‌های مجموعه‌ها مجزا[۲۵]
- استفاده از معیارهای فاصله یا شباهت بردار دیگری مثل جاکارد[۲۶] ، همینگ[۲۷] و ...
- استفاده از توابع درهم‌ساز جدول بندی ترکیبی[۲۸][۲۷]، زیرا در این جدول‌ها گستردگی پخش نقاط در تابع درهم ساز بیشتر است. اما برای یافتن معکوس چنین توابعی باید چاره‌ای اندیشیده شود.

مراجع

[۱] G. Cormode and S. Muthukrishnan, "An Improved Data-Stream Summary: The Count-min Sketch and its Applications", *J. of Algorithms*, ۵۵(۱), ۲۰۰۵.

[۲] Khan, Arijit, and Charu Aggarwal. "Query-friendly compression of graph streams", *Proceedings of the ۲۰۱۶ IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, ۲۰۱۶.

[۳] M.R.Henzinger, P.Raghavan, and S.Rajagopalan, "Computing on data streams", *External memory algorithms*, pages ۱۰۷–۱۱۸, ۱۹۹۹.

[۴] J.Feigenbaum, S.Kannan, A.McGregor, S.Suri, and J.Zhang, "On graph problems in a semi-streaming model", *Theoretical Computer Science*, ۳۴۸(۲-۳):۲۰۷–۲۱۶, ۲۰۰۵.

[۵] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, M. Strauss, "Fast, small-space algorithms for approximate histogram maintenance", in: *Proceedings of the ۳۴th*

[۲۴] Bit Mask
[۲۵] Disjoint Set
[۲۶] Jaccard Similarity
[۲۷] Hamming Distance
[۲۸] Mixed Tabulation Hash Functions

ACM Symposium on Theory of Computing, pp. ۳۸۹–۳۹۸, ۲۰۰۲.

[۶] B.Bollobaٔs. "Extremal Graph Theory", *AcademicPress*, New York, ۱۹۷۸.

[۷] S.Baswana, "Streaming algorithm for graph spanners single pass and constant processing time per edge", *Inf. Process. Lett*, ۱۰۶(۳):۱۱۰–۱۱۴, ۲۰۰۸.

[۸] M. Elkin. "Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners", *ACM Transactions on Algorithms*, ۷(۲):۲۰, ۲۰۱۱.

[۹] M. Elkin and J. Zhang, "Efficient algorithms for constructing $(۱+\epsilon,\beta)$-spanners in the distributed and streaming models", *Distributed Computing*, ۱۸(۵):۳۷۵–۳۸۵, ۲۰۰۶.

[۱۰] R.Tarjan, "Data Structures and Network Algorithms", *SIAM*, Philadelphia, ۱۹۸۳.

[۱۱] A.A.Benczuٔrand D.R.Karger, "Approximating s-t minimum cuts in $O(n^۲)$ time", In *ACM Symposium on Theory of Computing*, pages ۴۷–۵۵, ۱۹۹۶.

[۱۲] D.A.Spielmanand S.-H.Teng, "Spectral sparsificationof graphs", *SIAM J. Comput.*, ۴۰(۴):۹۸۱–۱۰۲۵, ۲۰۱۱.

[۱۳] J.D.Batson, D.A.Spielman, and N.Srivastava, "Twice-ramanujan sparsifiers", *SIAM J. Comput.*, ۴۱(۶):۱۷۰۴–۱۷۲۱, ۲۰۱۲.

[۱۴] K.J.Ahnand S.Guha, "Graph sparsificationin the semi-streaming model", In *International Colloquium on Automata, Languages and Programming*, pages ۳۲۸–۳۳۸, ۲۰۰۹.

[۱۵] J.A.Kelnerand A.Levin. "Spectral sparsification in the semi-streaming setting", *Theory Comput. Syst.*, ۵۳(۲):۲۴۳–۲۶۲, ۲۰۱۳.

[۱۶] N. Manerikar and T. Palpanas, "Frequent Items in Streaming Data: An Experimental Evaluation of the State-of-the-art", *Data Knowl. Eng.*, ۶۸(۴):۴۱۵–۴۳۰, ۲۰۰۹.

[۱۷] P. Roy, A. Khan, and G. Alonso, "Augmented Sketch: Faster and More Accurate Stream Processing", In *SIGMOD*, ۲۰۱۶.

[۱۸] L.S.Buriol, G.Frahling, S.Leonardi, A. Marchetti-Spaccamela, and C. Sohler, "Counting triangles in data streams", In *ACM Symposium on Principles of Database Systems*, pages ۲۵۳–۲۶۲, ۲۰۰۶.

[۱۹] A.Pavan, K.Tangwongsan, S.Tirthapura, and K.-L.Wu. "Counting and sampling triangles from a graph stream", In *International Conference on Very Large Data Bases*, ۲۰۱۳.

[۲۰] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs", In *ACM Symposium on Principles of Database Systems*, pages ۵–۱۴, ۲۰۱۲.

[۲۱] H.Jowhari, M.Saglam, and G.Tardos, "Tight bounds for lp samplers, finding duplicates in streams, and related problems", In *ACM Symposium on Principles of Database Systems*, pages ۴۹–۵۸, ۲۰۱۱.

[۲۲] R.Paghand C.E.Tsourakakis, "Colorful triangle counting and a map-reduce implementation", *Inf. Process. Lett.*, ۱۱۲(۷):۲۷۷–۲۸۱, ۲۰۱۲.

[۲۳] L.Becchetti, P.Boldi, C.Castillo, and A.Gionis, "Efficient algorithms for large-scale local triangle counting", *TKDD*, ۴(۳), ۲۰۱۰.

[۲۴] H.Jowhari and M.Ghodsi, "New streaming algorithms for counting triangles in graphs", In *COCOON*, pages ۷۱۰–۷۱۶, ۲۰۰۵.

[۲۵] D.M.Kane, K.Mehlhorn, T.Sauerwald, and H.Sun, "Counting arbitrary subgraphs in

data streams", In *International Colloquium on Automata, Languages and Programming*, pages ۵۹۸–۶۰۹, ۲۰۱۲.

[۲۶] M.Manjunath, K.Mehlhorn, K.Panagiotou, and H.Sun, "Approximate counting of cycles in streams" In *European Symposium on Algorithms*, pages ۶۷۷–۶۸۸, ۲۰۱۱.

[۲۷] Dahlgaard, Søren, Mathias Knudsen, and Mikkel Thorup, "Practical Hash Functions for Similarity Estimation and Dimensionality Reduction", *Advances in Neural Information Processing Systems*, ۲۰۱۷.