بهینه سازی پرس و جو با تطبیق الگوی توزیعی

چکیده: الگوریتم های حریصانه برای عملیات تطبیق الگوی گراف اغلب زمانی که دده های گراف را بتوان در حافظه بر روی تک ماشین نگه داری کرد کافی می باشد. با این حال، چون مجموعه داده های گراف به طور فزاینده ای توسعه می یابند و نیاز مند فضای ذخیره ای اضافی و پارتیشن بندی در دسته ای از ماشین ها می باشند، فنون بهینه سازی پرس و جوی پیشرفته تر برای اجتناب از انفجار در تاخیر پرس و جو اهمیت حیاتی دارد.در این مقاله، ما اقدام به معرفی روش های بهینه سازی پرس و جو برای تطبیق الگوی گراف توزیع یافته می کنیم. این فنون شامل، 1- الگوریتم بهینه سازی مبتنی بر برنامه نویسی دینامیک سبک سیستم R، که هر دو برنامه های خطی و نقطه ای را در نظر می گیرد، 2- الگوریتم مبتنی بر تشخیص چرخه که اهرمی برای چرخه ها جهت کاهش اندازه مجموعه های نهایی می باشد و 3- روش استفاده مجدد از محاسبه یا رایانش است که انتقال اطلاعات و اجرای پرس و جوی زائد را در شبکه حذف می کند. نتایج آزمایشی نشان می دهد که این الگوریتم ها می توانند موجب بهبود زیادی در عملکرد پرس و جو شوند.

1-مقدمه

مدل داده های گراف، یک شیوه بسیار محبوب برای زمینه های مختلف محسوب می شود. از دلایل این محبوبیت می توان به موارد زیر اشاره کرد:1- برای خلاصه سازی داده های پراکنده و یا نیمه ساختاری به مدل های داده های راس-لبه-راس نسبت به مدل های داده های سنتی ساده تر است 2- برخی از مجموعه داده های محبوب(نظیر شبکه های تویتر، فیسبوک و لینکدین) در مورد استفاده از پارادایم گراف منطق و استدلال طبیعی دارند و 3- عملیات گراف نظیر محاسبات با کوتاه ترین مسیر، تطبیق الگوی زیر گراف و پیج رنک به آسانی نسبت به مدل داده های گراف بیان می شوند.

بسیاری از مجموعه داده های گراف آن قدر بزرگ هستند که مدیریت آن ها بر روی یک ماشین سخت بوده و از این روی دسته ای از ماشین ها برای پردازش، ذخیره، مدیریت و تحلیل داده های گراف به کار برده می شوند. برای

مثال، از سال 2012، گراف کاربر فیسبوک دارای 900 میلیون راس(درجه متوسط راس، 130 است) می باشد(1). در جامعه وب معنایی، جنبش داده های اپن لینکینگ، 6 میلیون تریپل را (یک تریپل، برابر باست با یک لبه در یک گراف) از 300 مجموعه داده به هم پیوسته جمع اوری کرده است. از آن جا که بسیاری از الگوریتم های گراف در اصل با این فرض طراحی شده اند که گراف کامل را می توان در مموری بر روی یک ماشین ذخیره سازی کرد، معماری های توزیع یافته فوق نیاز به بازبینی این الکوریتم ها در زمینه توزیعی می باشند زیرا ملاحظاتی نظیر نهفتگی شبکه و بازدهی می تواند مانع از اجرای متعارف این الگوریتم ها شود.

تطبیق الگوی زیر گراف یک عملیات بسیار مهم می باشد که باید برای ذخایر گراف توزیعی باز بینی شود. عملیات تطبیق زیر گراف در عملیات داده کاوی شبکه اجتماعی(شمارش مثلث ها برای اندازه گیری اثر اجتماعی مراسم ها)، پرس و جو هایSPARQL در گراف داده های لینک شده و الگوریتم های یادگیری ماشین که موجب راه اندازی موتور ها برای سرگرمی و ابزار های تجارت الکترونیکی به شدت استفاده می شود.

مقاله حاضر به طور صریحی از روش های برنامه نویسی دینامیک سبک سیستم r استفاده می کند تا تطبیق الگوی زیر گراف توزیعی را بهینه سازی کند. با این حال، پی برده شد که این الگوریتم های متعارف باید به سه طریق اصلاح شوند تا عملکرد خوبی برای داده های گراف داشته باشند.

- اگرچه سایرین خاطر نشان کرده اند که حتی در زمینه های نسبی متعارف، مسئله سیستم R از درختان منجر به راهبرد بهینه سازی نیمه بهینه می شود، در نظر گرفتن برنامه های نقطه ای برای پرس و جو های تطبیق الگوی گراف توزیعی از اهمیت زیادی برای کاهش ترافیک شبکه و اندازه خروجی های میانی برخوردار است. بحی اکتشافی که برای آن ها طرح های نقطه ای در نظر گرفته می شوند باید اهرم خصوصیات گراف باشد.
- چرخه ها به طور مکرر در الگوهای پرس و جو در مدل گراف نسبت به داده های نشان داده شده در مدل های دیگر ظاهر می شوند. آن ها را می توان طوری اهرم بندی کرد که موجب بهبود عملکرد پرس و جو شده و به طور صریح طی تولید برنامه در نظر گرفته شوند.

- به طور کلی، تطبیق الگوی توزیعی زیر گراف با یافتن مولفه هایی از زیر گراف و اتصال این مولفه ها به هم انجام می شود. با این حال وقتی که الگوهای گراس خالص مورد جست و جو باشند، مجموعه های میانی حاصله، اندازه شان بزرگ تر می شود.

در این مقاله، ما دو قالب بهینه سازی پرس و جو را برای تطبیق الگوی زیر گراف ارایه می دهیم. با توجه به ذخیره داده ها در مدل داده های گراف، و پرس و جویی که همه نمونه های الگوی گراف را درون ذخیره گاه داده ها درخواست می کند،الگوریتم هایی را فراهم می کنیم که یک سری طرح های اجرای پرس و جو را تولید کرده، هر یک از طرح ها را براورد کرده و طرحی را با کم ترین هزینه برای اجرا انتخاب می کنند. هیچ گونه فرضی در مورد چگونگی تقسیم داده های گراف در خوشه وجود ندارد به جز این که همه لبه های بر گرفته از ورتکس در یک ماشین فیزیکی ذخیره شده و یک تابع قطعی (تابع نقطه ای) را می توان به لبه ها برای تعیین مکان آن اعمال کرد. به علاوه، ما هیچ گونه فرضی را در مورد زیر گراف ارایه نداده و الگوریتم ما به دو زیر گراف غیر توزیعی و توزیعی نسبت داده می شود(که هر لبه و راس در مجموعه داده های گراف دارای خصوصیات مربوط به آن بوده و زیر گراف می تواند شامل گزاره هایی در خصوص این صفات برای کاهش حوزه پژوهش باشد. اهمیت مطالعه ما در زیر اشاره شده است.

- ما یک قالب بهینه سازی مبتنی بر برنامه نویسی دینامیک را پیشنهاد می کنیم که تنها طرح های نقطه ای را بدون مواجه با انفجار فضای برنامه پرس و چو در نظر می گیرد.
- ما یک چارچوب بهینه سازی مبتنی بر تشخیص چرخه را بر اساس این مشاهده ارایه می دهیم که چرخه ها به طور معنی داری، اندازه مجموعه های متوسط نهایی را کاهش می دهند.
- ما یک روش استفاده مجدد از محاسبات و رایانش را ارایه می دهیم که اجرای زیر پرس و جوی مشابه و انتقال شبکه زائد مجموعه متوسط را در پرس و جو حذف می کند.

نتایج آزمایشی نشان می دهد که روش های پیشنهادی ما، عملکرد تطبیق الگوی زیر گراف را در یک ذخیره گراف توزیعی به ترتیب بزرگی در الگوریتم های حریصانه بهبود بخشیده و منجر به افزایش عملکرد می شود.

2-مقدمات

الف:تطبیق الگوی زیر گراف

مسئله تطبیق الگوی زیر گراف، یافتن همه زیر گراف هایی است که دارای الگوهای خاصی در یک گراف می باشند..اگرچه تغییرات زیادی در این مسئله وجود دارد(14)، از جیث ایزومورفیسم زیر گراف تغریف می شود که تنها ساختار گراف را مد نظر قرار می دهد.عملا بسیاری از گراف ها با اطلاعات سمانتیک یا معنایی تفسیر می شوند و این اطلاعات را می توان به صورت بخشی از الکوی تطبیق یافته در نظر گرفت.سمانتیک ها یا معناها اغلب به صورت نوع، برچسب و یا خصویات رئوس یا لبه ها نمایش داده می شوند. انواع و نام ها را می توان به آسانی به صفات تبدیل کرد. از این روی ما اشاره به افزایش مفاهیم به عنوان گراف های نسبتی خواهیم داشت. در این مقاله، ما هر دو عملیات ایز مورفیسم زیر گراف و تطبیق عمومی را در گراف های نسبت داده شده در نظر می گیریم. تعریف رسمی نشان می دهد که این خود دارای هر دو نوع تطبیق ارایه شده در زیر است.

تعریف 1: لبه هدفمند یا جهت دار به صورت (A,B) یا $A \rightarrow B$ نمایش داده می شود.A موسوم به راس مبدا و راس B موسوم به راس مقصد است.

تعریف 2- یک گراف داده یک گراف جهت دارد $G = (V, E, A_V, A_E).$ می باشد.V مجموعه ای از رئوس بوده و E، مجموعه ای از لبه های جهت دار است. $\forall e \in E,\ e = (v_1, v_2)$ و $A_V(v), v_1, v_2 \in V.\ \forall v \in V$ یک چند تایی $(A_1 = a_1, A_2 = a_2, ..., A_n = a_n)$ می باشد که Ai، خصوصیت v و a_i ثابت بوده، $1 \leq i \leq n.$ می باشد. A_E یک تابع مشابه تعریف شده بر

روی E می باشد. یک زیر گراف $G' = (V', E')$ زیر گرافG است تنها اگر و اگر $V' \subseteq V$ و $E' \subseteq V' \times V' \subseteq E$ باشد.

تعریف 3: گراف الگو/ گراف پرس و جو /پرس و جو یک گراف جهت دار $Q = (V_Q, E_Q, f_{V_Q}, f_{E_Q})$ می باشد. V_Q مجموعه ای از رئوس و E_Q مجموعه ای از لبه های جهت دار است. $\forall e \in E_Q$ ، $e = (v_1, v_2)$ و $v_1, v_2 \in V_Q$. $\forall v \in V_Q$, $f_{V_Q}(v)$ فرمولی است که متشکل از گزاره هایی است که توسط اپراتور های منطقی $\vee$ (انفصال)، $\wedge$ (اتصال) و $\neg$ (رد) متصل می شود. یک گزاره متشکل از V، ثابت ها و توابع به صورت استدلال بوده و به عملگر مقایسه اعمال می شود نظیر $<, \leq, =, \neq, >$ و $\geq$. f_{E_Q} تابع مشابه تغریف شده بر روی E_Q. است.

تعریف 4-تطبیق گراف الگوی $Q = (V_Q, E_Q \quad f_{V_Q}, f_{E_Q})$ یک گراف داده ای $G = (V, E, A_V, A_E)$ می باشد که زیر گراف $G' = (V', E')$ از G می باشد طوری که

1-دو تابع دو سویه وجود دارد که یکی از V_Q به V' بوده و دیگری از E_Q به E' است. به عبارت دیگر، متناظر های یک به یک از V_Q به V' و از E_Q به E' می باشد.

2-f_{V_Q}، A_V را بر روی V' و A_E ، f_{E_Q} را بر روی E' براورده می کند.

به طور شهودی، A_V و A_E ، ص به ترتیب نشان دهنده صفات یا نسبت های رئوس و لبه می باشد در حالی که f_{V_Q} و f_{E_Q} نشان دهنده شرایط تطبیق بر روی خصوصیات می باشند. تطبیق الگوی زیر گراف باید همه تطابق ها را پیدا کند.

ب: پارتیشن بندی داده ها و الحاق

در سیستم های دیتابیس موازی و توزیعی، پارتیشن بندی داده ها اثر معنی داری بر روی اجرای پرس و جو دارد. هز هزینه الحاق به شدت بستکی به چگونگی پارتیشن بندی در دسته ای از ماشین ها دارد که پارتیشن های مختلف داده ها را ذخیره می کند. برای مجموعه داده های نشان داده شده در مدل داده های گراف، رئوس با اعمال یک تابع قطعی به رئوس پارتیشن بندی می شود که نتیجه این تابع نشان می دهد که رئوس باید ذخیره شود.برای لبه ها، یکی از دو رئوس که لبه به ان متصل می شود، راس پارتیشن بندی را شامل می شود و همین عمل پارتیشن بندی به راس برای تعیین محل لبه اعمال می شود. برای مثال، لبه های $A\rightarrow B$ و $A\rightarrow C$ با یک راس مبدا، در یک پارتیشن قرار می گیرند. از سوی دیگرف لبه های $A\rightarrow B$و $C\rightarrow B$ می توانند در یک پارتیشن قرار گیرند.

به طور کلی، وظیفه تطبیق الگوی زیر گراف به صورت تکه ای صورت می گیرد یعنی چست و جوی قطعاتی از زیر گراف ها به طور مستقل و اتصال این قطعات بر روی رئوس مشترک. این قطعات به صورت یک تک لبه بوده و به طور مکرر با هر اتصال بزرگ تر می شوند. همه تطبیق ها برای هر قطعه بعدی انجام می شوند که به آن مجموعه نتیجه متوسط می گویند. هر نتیجه متوسط برای قطعه زیر گراف دارای یک راس در قطعه می باشد که موسوم به راس پارتیشن بندی بوده و همین عمل قطعی برای راس های پارتیشن استفاده شده و لبه ها در مجموعه داده های گراف خام برای پارتیشن بندی مجموعه نتیجه های متوسط بر اساس این راس استفاده می شود. اتصال مجموعه های نهایی با روش های اتصال استاندارد در سیستم های دیتابیس موازی و توزیعی صورت می گیرد.: اتصالات هم محل، اتصالات جهت دار، اتصالات توزیعی و اتصالات انتشاری.

اتصال هم محل یک اتصال محلی می باشد. این اتصال زمانی استفاده می شود که راس اتصال، راس پارتیشن بندی برای همه مجموعه های منتج باشد. راس پارتیشن بندی مجموعه منتج متوسط اتصال، یک راس اتصالی است.

در صورتی که تنها یکی از مجموهه های منتج اتصالی با راس اتصالی تقسیم شود، یک اتصال جهت دار را می توان استفاده کرد. در این صورت، مجموعه متوسط تقسیم شده توسط راس اتصال، مجددا با راس اتصالی پارتیشن بندی می شود. بعد از پارتیشن بندی مجدد، یک اتصال هم محل انجام می شود.

در صورتی که هیچ یک از مجموعه های میانی اتصال نیابند، اتصال توسط راس اتصالی صورت گرفته و اتصال توزیعی یا هاش استفاده می شود. اتصال هاش موجب پارتیشن بندی همه ووردی ها با راس اتصالی شده و سپس یک اتصال هم محل صورت می گیرد. از این روی، راس پارتیشن بندی مجموعه منتج نهایی تولید شده با این اتصال، یک راس اتصالی است. اتصال انتشاری اقدام به تکرار مجموعه میانی کوچک نر برای هر محل کرده و سپس بخش های بزرگ تر را تقسیم می کند. یک اتصال بین هر پارتیشن با نتیجه میانی بزرگ تر و نتیجه میانی کوچک تر صورت می گیرد.

3-بهینه سازی مبتنی بر برنامه نویسی دینامیک

بهینه سازی پرس و جو در سیستم های دیتابیس سنتی را می توان به عنوان یک مسئله جست و جو در نظر گرفت که مشکل از سه بخش یعنی تولید فضای جست و جو، براورد هزینه و جست و جو است.بهینه ساز پرس وجو اقدام به تعریف یک فضای برنامه های پرس و جو کرده . فضایی را با یک الکوریتم جست و جو و براورد های هزینه ای طرح ها کشف می کند. مسئله تطبیق الگوی زیر گراف نیز به همین طریق بوده و الگوی زیر گراف تطبیقی یک پرس و جویی است که در داده های گراف خام پردازش می شود. همان طور که در بخش قبلی گفته شد، این پرس و جو ها را می توان از طریق تطبیق قطعات زیر گراف و اتصال این قطعات انجام داد. از این روی مسئله بهنینه سازی برای تطبیق الگوی زیر گراف را می توان از حیث بهینه سازی رتبه بندی اتصال انجام داد.

به طور کلی، با توجه به n راس، $((2n-2)!)/((\bar{n}-1)!)$ متفاوت بوده ولی اتصال معادل منطقی صورت می گیرد. در صورتی که هر اتصال دارای چندین گزینه اجرایی باشد، فضای حالت بیشتر منفجر می شود. در نتیجه، بسیاری از نمونه های تحقیقاتی و اجرای سیستم های دیتابیس موجب محدود شدن فضای جست و جو در زیر مجموعه محدود و ایجاد خطر بر روی طرح و برنامه بهینه می شود.

در این بخش، ما برنامه نویسی دینامیک سبک سیستم r را که به پرس و جوی تطبیق الگوی زیر گراف بر روی داده های گراف توزیع یافته بررسی می کنیم.

الف: الگوریتم جست و جو

شکل 1(که کد تماس را در شکل های 2-3-5- و 7 تعریف می کند) یک شبه کد را برای الگوریتم FindPlan می دهد که با توجه به پرس و جوی ورودی، ایجاد یک فضای حالت برای پردازش پرس و جو کرده و از الکوریتم جست و جوی برنامه نویسی دینامیک برای یافتن کم هزینه ترین روش در پرس و چو استفاد ه می کند. توجه کنید گه برنامه های اجرای یمختلف منجر بخ یک نتجه و پرس و جوی خروجی می شود که به روش های مختلف پارتیشن بندی می شود. به طور کلی در صورتی که پرس و جو دارای n راس باشد، n شیوه مختلف برای تقسیم خروجی وجود خواهد داشت. راس پارتیشن بندی خروجی در صورتی مهم است که ورودی یک قطعه پرس و جو باشد و خروجی با قطعه پرس و جوی متقاوت اتصال شود. از این روی به جای اولید یک روش کم هزینه برای کل پرس و جو، این خود برنامه کم هزینه را برای هر راس پارتیشن بندی خروجی منحصر به فرد می کند.خروجی FindPlan آر ایه ای از سه تایی های حاوی 1- رئوس پارتیشن بندی خروجی، 2- هزینه برنامه کم هزینه و 3- کم هزینه ترین برنامه می باشد.

به طور خلاصه،FindPlan از همه تجزیه های گراف پرس و جو به دو قطعه یا بیشتر تبعیت کرده و هزینه های اتصال هر جفت قطعه را محاسبه می کند. چون همین قطعه توسط الگوریتم های مختلف خوانده می شود، محاسبه و رایانش مکرر با تهیه نقشه از هر قطعه پرس و جو به یک سه تایی خالی اجتناب می شود که برای اولین بار FindPlan را اشغال کرده و به نام همان قطعه نام گذاری می شود. از این روی، نخستین لاین شبه کد FindPlan در ابتدا بررسی می کند که آیا ارایه تریپل مربوط به این قطعه خالی است یا نه؟ اگر خالی نبود، آن گاه، FindPlan برای این قطعه درخواست شده و نباید مجددا اجرا شود.

برای هر پرس و چویی که به طور مکرر به قطعات کوچک تر تجزیه می شود، در نهایت قطعات به اندازه یک لبه کوچک تر می شوند. در صورتی که قطعه دارای یک لبه باشد، می توان به آسانی هزینه تطبیق لبه را محاسبه کرد. با این حال، می توان دومین پایگاه داده ها را افزود. در صورتی که مبدا همه لبه ها در بخش پرس و چو از یک راس باشد، اتصال n شیوه می توان یک روش بهینه برای پردازش این قطعه باشد و تجزیه بیشتر لازم است.

در صورتی که گزاره های اگر و آیا در شبه کد متناظر با این موارد نامطلوب باشند، آنگاه پرس و جو به قطعات با LinearDecompositionو BushyDecomposition تقسیم می شود. سپس ما FindPlan را برای هر یک از این قطعات درخواست می کنیم. وقتی برنامه های پرس و جو برا برای این قطعات محاسبه شد، برنامه های با کم ترین هزینه برای اتصال به ترتیب با GenerateLinearPlansو GenerateBushyPlans محاسبه می شوند. بعد از تولید این برنامه ها، موارد نامطلوب با تابعEliminateNonMinCosts کنار گذاشته می شوند. برای مثال، فرض کنید که ما دارای سه تریپل می باشیم یعنی t_1=(A, 100, p_1), t_2=(A, 200, p_2) و t_3=(B, 400, p_3) است. اگرچه t2 دارای هزینه کم تر از t3 است،حذف می شود زیرا هزینه آن بیش از t1 بوده و از این رو توسط هر اتصال در آینده استفاده نمی شود. از این روی خروجی FindPlan به صورت [(A, 100, p_1), (B, 400, p_3)] است.

برنامه های خطی

دو گام برای ایجاد یک برنامه خطی برای پرس و جو وجود دارد. در ابتدا پرس و جو به دو قطعه تجزیه می شود که حداقل یکی از آت ها غیر قابل تجزیه می باشد و سپس یک روش اتصال عملی برای اتصال قطعات استفاده می شود. این دو مرحله با LinearDecomposition وGenerateLinearPlans صورت می گیرند. همان طور که در بالا گفته شد، یک قطعه با همه لبه های با یک راس مشترک به صورت غیر قابل تجزیه در نظر گرفته می شود.

مثال 1: با توجه به این که پرس و جوی، $A\rightarrow B,\ A\rightarrow C,\ B\rightarrow D,\ B\rightarrow A,\ C\rightarrow E,$ است، لبه ها به سه قطعه بر اساس راس های مبدا طبقه بندی می شوند: Q_1: $A\rightarrow B$,$A\rightarrow C$. Q_2: $B\rightarrow D$,$B\rightarrow A$. Q_3: $C\rightarrow E$ سه تجزیه به صورت زیر هستند

$$1{:}\{Q_1\},\{Q_2,Q_3\}\ \ 2{:}\{Q_2\},\{Q_1,Q_3\}\ \ 3{:}\{Q_3\},\{Q_1,Q_2\}$$

در تجزیه 1، الف: به دو لبه در Q1 اتصال می یابد ب: دو لبه در Q2 متصل می شوند 3- حاصل 2 باQ3 ترکیب شده و در نهایت حاصل 1 و 3 ترکیب می شوند. این یک طرح خطی نیست. با این حال اتصالات در مراحل 1 و 2 از

طریق اتصال هم مکان صورت می گیرد که ا رزان تر از اتصالات توزیعی بوده و به صورت نهاد های اتمی در نظر گرفته شده و برنامه با توجه به اتصالات غیر محلی، خطی است.

شکل 1:الگوریتم جست و جوی برنامه نویسی دینامیک که تولید کم هزینه ترین برنامه برای هر راس پارتیشن بندی خروجی از پرس و جوی ورودی می کند. توابع زیر در شکل های 2-3-5-7 نشان داده شده است.

بعد از این که FindPlan،LinearDecomposition را برای تولید تجزیه خطی درخواست می کند، برنامه های پپرس و جو برای هر جفت قطعه با LinearDecomposition تولید می شوند. برنامه مونتاژ مجدد هر جفت قطعه با عملیات GenerateLinearPlans صورت می گیرد. این عملیات هر ترکیب زوجی را یکی از هر قطعه تولید کرده و هزینه این اتصالات را برای هر الکوریتم توزیعی محاسبه می کند.

یافتن برنامه های خطی دارای پیچیدگی زمانی $\mathcal{O}(|V|^2 \cdot |E|!)$ است.

ج:طرح های نقطه ای

همان طور که در بالا گفته شد، ف ضای جست و جوی کامل برنامه های پرس و چو به شدت بزرگ بوده و هزینه الگوبرداری از همه برنامه ها ممکن است زیاد باشد. با این حال محدود کردن جست و جوی برنامه های خطی می تواند منجر به اختلال در پهنای باند برای اتصالات توزیعی شود. از این روی در این بخش، ما یک بحی اکتشافی را ارایه می کنیم که از طریق آن می توان اقدام به کشف برنامه های مناسب با در نظر گرفتن خصوصیات پرس و جوی تطبیق الگو به صورت کم هزینه کرد.1-تبدیل گراف پرس و چو به نوع خاصی از گراف های غیر چرخه ای 2- تجزیه DAG به چندین قطعه در رئوس با بیش از یک لبه ورودی3-ایجاد طرح های پرس و جو با اتصال قطعات با اتصالات نقطه ای و توزیعی.

شکل 2- شبه کد برای تولید برنامه های خطی

1-تبدیل

دو هدف برای این منظور وجود دارد: در ابتدا بسیاری از گراف های پرس و جو دارای سیکل هایی هستند که منجر به پیچیده شدن قطعه بندی پرس و جو ش ده و چرخه ها را با تبدیل پرس و جو به DAG خارج کرده و منجر به

قطعه بندی مستقیم می شود.دوما چون طرح نقطه ای دارای فضای زیادی است، بازنویسی پرس و جو به عنوان DAG موجب محد ود شدن فضای جست و جو شده طوری که تعداد کمی از نامزد ها کشف می شود.

نتیجه تبدیل، مجموعه ای از انواع خاص DAG می باشد که موسوم به DAG لایه بندی شده است. گراف یک DAG لایه بندی شده است در صورتی که دو شرط زیر برقرار باشد1- یک DAG باشد و 2- هر راس v در گراف دارای یک ID l لایه باشد و لبه ها از v به راس های با $\mathrm{ID}\ l+1$ انتقال یابد.

شکل 3، توابع اصلی مورد استفاده در فرایند تبدیل یعنی GenerateLayeredDAGs, GenerateLayeredDAG, GenerateComponent, وMerge را نشان می دهد.

GenerateLayeredDAGs یک DAG لایه بندی شده را با این راس نشان می دهد.کد برای تولید یک dag لایه بندی شده با راس مشخص در GenerateLayeredDAG یافت می شود. این کار با یافتن همه رئوس و لبه های قابل دسترس از v بدون هر گذار دو طرفه به لبه صورت می گیرد. این مجموعه از رئوس و لبه ها موسوم به مولفه یا جز می باشند. در صورای که مولفه تولید شده مساوی با کل گراف پرس و جو نباشد، راس افزایشی به صورت نقطه شروع برای دومین مولفه در نظر گرفته می شود. این فرایند تا زمانی تکرار می شود که همه رئوس و لبه های یک گر اف در یک مولفه توجیه شود. در این نقطه، همه مولفه های تولید شده با استفاده از عملیات ادغام، ترکیب شده و نتیجه یک DAG تک لایه است.

عملیات GenerateComponent ایجاد یک DAG لایه ای می کند. راس ورودی به خودی خود در لایه DAGقرار دارد. همه رئوس قابل دسترس از اطریق یک لبه دارای نخستین لایه بوده و همه رئوس قابل دسترس از یک لبه دارای دومین لایه است. این فرایند تا زمانی ادامه می یابد که لایه l+1 دارای رئوسی باشد که از طریق لبه از بردار در l امین لایه حاصل می شود مادامی که لبه متصل به آن ها قبلا در لایه قبلی حاصل نشده باشد. همراه با این رئوس، هر لایه دارای مجموعه ای از لبه ها می باشد که در لایه بعدی به راس می رسد.

اگرچه، هر لبه تنها یک بار در کل لایهDAG قرار دارد ، راس ها بیش از یک بار وجود خواهند اشت. برای مثال، در صورتی که راس های a-B از طریق یک لبه متصل شوند، DAG لایه بندی شده از a تولید می شود. برای تفکیک

چند نمونه از رئوس در یک مولفه، ما از ID لایه به عنوان علامت راس ID جهت شناسایی راس استفاده می کنیم. از این رویف DAG لایه بندی شده به صورت زیر نشان داده می شود. $\mathrm{A}_0 \rightarrow \bar{\mathrm{B}}_1 \rightarrow \mathrm{A}_2$. همه نمونه های رئوس در یک لایه به یک نسخه از راس افزوده می شوند.

همان طور که در بالا گفته شد،GenerateLayeredDAG تا زمانی ایجاد مولفه می کند که همه رئوس و لبه ها در حداقل یک مولفه قرار گیرند. دو مولفه با یافتن یک راسی ترکیب می شوند که بین آن ها مشترک باشدو اتصال این دو مولفه در این لبه صورت گیرد. برای مثال، فرض کنید دو مولفه c1-C2 دارای رئوس B1-B3 باشند، چون این دو متناظر با یک راس B در گراف پرس و جو می باشند، دو مولفه را می توان در B1-B3 ترکیب کرد. ترکیب لایه ها به صورت زیر رخ می دهد: ترکیب لایه 0 از c1 و لایه 2 از c2، ترکیب لایه 1 از C1 و لایه 3 از C2. در صورتی که لایه فاقد لایه متناظر باشد، به یک لایه جدید با مولفه جدید تبدیل می شود و موقعیت نسبی خود را به لایه های ترکیبی دیگر حفظ می کند.

مثال 2- این مثال نشان می دهد که اگرGenerateLayeredDAG بر روی پرس و جوی ورودی نمونه از شکل چهار با راس شروع d درخواست شود چه اتفاقی رخ می دهد. تنها یک راس از D قابل دسترس است و از این روی نخستین مولفه نشان داده شده در شکل 4 الف، متشکل از $D \rightarrow E$ است. فرض کنید که A به صورت یک راس آغازین بعدی در نظر گرفته می شود. این موجب یک مولفه ای می شود که در شکل 4 ب نشان داده می شود. توجه کنید که لبه $D \rightarrow E$ در شکل 4 الف استفاده شده و از این روی نمی توان آن را به شکل 4 ب افزود. C تنها راس باقی مانده با لبه بر کرفته از آن بوده و به صورت یک راس آغازین برای مولفه بعدی انتخاب می شود و مولفه نشان داده شده در شکل 4 ج ایجاد می شود. در این نقطه، همه لبه ها در یکی از مولفه های فوق توجیه می شود و گام بعدی می تواند تلفیق باشد. مولفه های شکل 4 الف و ب بر روی راس E ترکیب می شود. مولفه های شکل 4 ج و شکل 4 د بر روی راس B ترکیب شده و DAG لایه بندی شده نهایی در شکل 4 ی تولید می شود.

اگرچه GenerateLayeredDAG تولید DAG لایه بندی شده برای هر راس در گراف می کند، امکان دارد برخی از لایه های DAG مشابه باشند. برای مثال، یک پرس و چوی مسیر ساده $A \rightarrow B \rightarrow C$ با طول 2 منجر به DAG دقیق برای همه سه راس آغازین شود.

نتیجه نهایی مهم مرحله تبدیل این است که همه DAGs های لایه بندی شده به طور منطقی معادل با گراف پرس و جوی ورودی هستند. در صورتی که DAGs برای مجموعه داده های خام درخواست شود، نتیجه نهایی مطابق با DAGs منطبق با پرس و جوی اصلی می شوند. از این روی، مرحله تبدیل معادل با مرحله بازنویسی پرس و جوی بهینه ساز های دیتابیس می باشد.

2- تجزیه نقطه ای: وقتی DAGs لایه بندی شده توسط مرحله تبدیل تولید شد، آن ها را می توان به اجزایی طبقه بندی کرد که به طور مستقل طراحی و اجرا شوند. رئوس با چندین لبه، نقاط تجزیه طبیعی باشند، یک قطعه برای شاخهDAG متناظر با هر لبه ورودی و یک قطعه اضافی در نظر گرفته می شود.

شکل 5، شبه کد را برای تجزیه نقطه ای فوق نشان می دهد. این روش از لایه انتهایی DAGs برگرفته شده و سعی می کند تا رئوس نهایی را با بیش از یک لبه ورودی پیدا کند. در صورتی که یک راس v وجود داشته باشد، عملیات Decomposable انجام می شود تا پی برده شود که آیا تجزیه DAG به قطعات مجزا حول V امکان دارد یا نه و اگر چنین است، نتیجه نهایی این قطعات به صورت تجزیه مشخص می شود. عملیات Decomposable موسوم به ReachableDAG می باشد که ایجاد یک قطعه مشخص با همه زادگان می کند/ برای هر لبه ورودی، هدف عملیات ReversiblyReachableVertices از V آغاز شونده با همان لبه قابل دسترس بوده و ایجاد قطعه ای می کند که حاوی همه زیر زادگان راس هایی است که بدون عبور از V قابل دسترسی هستند. بسته به میزان هم پوشانی نیاها، این قطعات با هم هم پوشانی داشته و به صورت انفصال زوجی در نظر کرفته می شوند. در صورت هم پوشانی، عملیات Decomposable به صورت کاذب درنظر گرفته می شود. در غیر صورت، به صورت حقیقی شده و همراه با مجموعه ای از قطعات انفصالی ایجاد می شود.

3- تولید طرح های نقطه ای

در شکل 1 به یاد آورید که بعد از این که تجزیه نقطه ای انجام شد، FindPlan ایجاد طرح هایی برای هر قطعه در تجزیه شده و GenerateBushyPlans را برای برنامه ریزی چگونگی ترکیب قطعات تجزیه شده درخواست می کند. کد GenerateBushyPlans در شکل 7 مشاهده می شود.

عملیات فوق از طریق ترکیبی از کرفتن یک تریپل از هر قطعه ورودی تکرار شده و ایجاد یک برنامه برای اتصال می کنند.در صورتی یک اتصال هدفمند را می توان استفاده کرد که راس پارتیشن بندی حدقل یکی از سه گانه ها برابر با راس اتصال باشد. در غیر این صورت، اتصال هاش باید استفاده هاش شود.

مثال 4- فرض کنید که تجزیه نقطه ای متشکل از سه نقطه باشد یعنی $q_1,\ q_2$ و q_3 که دارای سه طرح زیر است.

q_1 :تریپل t1 = $(A,\ 100,\ p_1)$ و $t_2=(B,\ 200,\ p_2)$.

q_2 :تریپل t3 = $(A,\ 300,\ p_3)$ و t4 = $(B,\ 400,\ p_4)$

q_3 :تریپل t5 = $(A,\ 500,\ p_5)$ و t6= $(B,\ 600,\ p_6)$

$2^3 = 8$ شیوه مختلف برای انتخاب یک تریپل از هر قطعه وجود دارد. یک شیوه t4 –t5-t1-4 می باشد .و سپس اتصال هدفمند را می توان استفاده کرد و طرح به صورت زیر دنبال می شود:p4-p1 را اجرا کرده و p4 را بر روی a پارتیشن بندی کرده و p2 را بر روی a اتصال کنید. هزینه طرح 100 +

400 + 500+ هزینه اتصال+ هزینه شبکه پارتیشن بندی p4 می باشد.

یافتن طرح های نقطه ای دارای پیچیدگی زمانی زیر است: $\mathcal{O}(|V|^2$

در این بخش، ما یک چارچوب بهینه سازی دیگر را بر اساس تشخیص چرخه ارایه می دهیم. چارچوب بهینه سازی برنامه نویسی دینامیک ارایه شده در بخش قبلی، لبه های ناشی از راس یکسان را به دلیل ارجحیت گروه بندی در یک مکان طبقه بندی می کند. دلیل این است که تطبیق چرخه به عنوان یک گزاره انتخاب عمل کرده و زیر

مجموعه ای از مجموعه داده های خام را محدود می کند. از این روی تطبیق الگو ها منجر به مجموعه ننایج میانی شده و عملکرد بهتری را در اختیار می گذارد. با توجه به پرس و جوی $A\to B,\ A\to C,\ B\to A,$ ، یک الکوریتم مبتنی بر سیکل منطبق با $A\to B,\ B\to A$ می شود این در حالی است که الگوریتم های ارایه شده در بخش قبلی منطبق با $A\to B,$ می باشد. با این حال، اگر راس در مجموعه داده های خام دارای 20 لبه خروجی باشد، 190 انطباق با $A\to B$ وجود خواهد داشت. با این حال اگر تنها 2 مورد از 10 لبه از این راس دارای یک لبه در جهت دیگر باشند، اتصال $A\to B,\ B\to A$ موجب کاهش مجموعه نهایی می شود. توجه کنید که هر دو چرخه جهت دار و بدون جهت قادر به کاهش اندازه مجموعه نهایی می شود.

مثال 5- الگوی $A\to B,\ B\to C,\ C\to A$ یک چرخه جهت دار است. الگوی $A\to B,\ A\to C,\ B\to C$ یک چرخه بدون جهت است.

شکل 8 یک شبه کد را برای چارچوب بهینه سازی مبتنی بر جهت چرخه نشان می دهد. ابتدا کراف را به یک گراف بی جهت تبدیل کرده و همه چرخه ها را پیدا می کند. بعد از این که چرخه ها شناسایی شدند، دو رویکرد برای ترکیب آن ها استفاده می شود. رویکرد اول، نوع حریصانه است. این رویکرد یک سیکل را از نقطه شروع انتخاب کرده و چرخه های دارای هم پوشانی را به طور حریصانه می افزاید. روش حریص در الگوریتمها اصولایکی از تکنیکهای مهم در طراحی الگوریتمها بشمار می اید و در حل طیف گسترده ای از مسائل از ان استفاده می شود، بیشتر این مسائل (نه همه شان n (تا ورودی دارند و ما نیاز داریم به مجموعه جوابی برسی مکه شرایط مرزی و محدود کننده را در صورت مسأله ارضا کند . هر مجموعه ای که این محدودیتها و شرایط مرزی را برآورده سازد در واقع یک راه حل)) ممکن))خوانده می شود . ما به یافتن مجموعه ای نیاز داریم که راه حل))ممکن)) را برای تابع مورد نظر بیابد. راه حل((ممکنی))که این نقش را برای ما بتواند ایفا کند، یک راه حل « بهینه» نامیده می شود. روش حریص میگوید که شما می توانید الگوریتمی را ارائه دهید که بصورت گام به گام میباشد که در هر زمان یک ورودی دارند در هر گام این امر که ایا یک ورودی مخصوص، راه حل بهینه است یا نه ، تصمیم گیری و مشخص می شود . این کار با این فرض صورت می گیرد که ورودیها با ترتیبی مشخص و معین که به وسیله یک روال انتخاب گرفته می شوند، وارد

گامهای الگوریتم می شوند ، حال اگر در یک گام یک ورودی آمد و در گام بعدی مشخص شد که ان ورودی به مجموعه جواب ››ممکن‹‹ نمی انجامد ، از مجموعه کل خروجی (راه حل ممکن) کنار گذاشته می شود. این روال انتخابگر خود مبتنی بر برخی سنجشهای بهینه سازی است . این سنجش ممکن است همان تابع هدف باشد یا نباشد.

در واقع، چندین سنجش و اندازه گیری و یا همان راه کار برای تابع مورد نظر ما ، معقول هستند و اصولا غیر از این نمی تواند باشد،پس بسیاری از اینها › راه حلهای بهینه‹ در الگوریتمهایی که برای راه حلهای فرعی بهینه هستند، به کار می ایند. ما می توانیم روش حریص را بصورت مجرد و انتزاعی ولی بسیار دقیقتر از بالا توضیح دهیم.

مثال 6- ما از $Q3, Q4$ و $Q5$ در شکل 11 برای تشریح مفهوم چرخه های بدون هم پوشانی استفاده می کنیم.در Q3، چرخه های ABCوDE هم پوشانی ندارد زیرا آن ها هیچ گونه لبه مشترکی را ندارند.در Q4، چرخه ABوCBD هم پوشانی ندارند. آن ها دارای راس b مشترک می باشند ولی لبه مشترک ندارند. در Q5، چرخه های BDوBDE هم پوشانی دارد زیرا آن ها لبه مشترک BD دارند.

مثال 7- سه چرخه در Q4 در شکل 11 وجود دارد یعنی AB, BCD BCE. چرخه های BCDوBCE با لبه مشترک BC شناخته می شود. برای رویکرد حریصانه، اگر اولی منطبق با BCD باشد، دومی لبه $B \rightarrow E$ و $C \rightarrow E$ می افزاید زیرا BCE و BCE هم پوشانی دارند. در نهایت، تطبیق دو مولفه به صورت جداگانه صورت گرفته و نتیجه نهایی ترکیب می شود.

بهینه سازی تشخیص چرخه نقطه ای و حریصانه دارای به ترتیب پیچیدگی $\mathcal{O}(|V|^2 \cdot \check{C}^2)$ و $\mathcal{O}(|V|^2 \cdot C!)$ می باشد که C تعداد چرخه های موجود در پرس و جو است.

V: براورد هزینه

هزینه اتصال بستکی به اندازه و اماره های ورودی دارد. با توجه به این که $q_1 \bowtie q_2$, است، هزینه را با جمع چهار بخش براورد می کنیم 1-هزینه q_1 2-هزینه شبکه جا به جایی نتایج q_1و q_2 و 4- هزینه انجام اتصال بعد

از انتقال شبکه.1 و 2 به طور مکرر محاسبه شده و براورد هزینه استاندارد در DBMS صورت می گیرد. شکل 9 هزینه های شبکه را در چهار اتصال مورد استفاده در سیستم نشان می دهد.

استفاده مجدد از محاسبات

ما مفهوم استفاده مجدد از محاسبات را با چندین ممونه شهودی بر روی تطبیق الگوی ساختاری معرفی می کنیم. با توجه به یک پرس و جوی ساده، $A\rightarrow B,\ C\rightarrow B$ یکی از طرح های احتمالی برای اجرای اولیه $A\rightarrow B$ و $C\rightarrow B$ بوده و سپس یک اتصال هاش توزیعی بر روی B صورت می گیرد. لازم به ذکر است که تطبیق $A\rightarrow B$ و $C\rightarrow B$ دارای تطابق دقیق مشابه است. هر دو همه لبه ها را در کراف و انتخاب آن موثر هستند طوری که لبه مبدا از لبه مقصد متفاوت است و آن را می توان به صورت $V_1 \rightarrow V_2$ نشان داد. از این روی به جای تولید یک نتیجه نهایی به صورت دو بار، می توان آن را یک بار ایجاد کرد و طرح پرس و جوی حاصله در Q1 در شکل 10 نشان داده شده است.

دیگر پرس و جوی مثال $B\rightarrow A,\ C\rightarrow B,\ D\rightarrow B$ است. مشابه با فوق، می توان مجموعه ای از همه لبه های $V_1\rightarrow V_2$ بدست اورد . در این صورت، ما از سه رونوشت نتیجه میانی استفاده می کنیم. یک رونوشت قرار داده شده و دو نسخه توسط V2 پارتیشن بندی می شوند. در این نقطه، سه مجموعه از نتایج میانی $V_1\rightarrow V_2,\ V_2\rightarrow V_1$ و $V_2\rightarrow V_1$ بر روی راس پارتیشن بمدی هر مجموعه میانی متصل می شود. طرح پرس و جو به صورت Q2 در شکل 10 نشان داده شده است.

مادامی که قطعات چندگانه تولید نتایج نهایی مستقل از مجموعه داده ها کنند،اجرای قطعه برگشت پذیر است. استفاده مجدد از محاسبات در تطبیق الگوی ساختاری صورت گرفته و به ندرت در تطبیق الگوی معنایی دیده می شود زیرا تطبیق خصوصات قبل از تولید نتایج میانی موجب کاهش قابلیت استفاده مجدد می شود.

ارزیابی آزمایشی

در این بخش، ما عملکرد الکوریتم های بهینه سازی را بر روی چهار مجموعه گراف با اندازه و ماهیت متفاوت اندازه گیری می کنیم. کد و پرس و جو های مورد استفاده در این ارزیابی در http://db.cs.yale.edu/icde2014 قابل دسترس است.

الف: شرایط آزمایشی

1-شرایط سیستم و محیط آزمایشی

آزمایشات بر روی یک دسته از ماشین ها انجام شدند. هر ماشین دارای یک پردازنده 2.40 GHz Intel Core 2 Duo بر روی 64-bit Red Hat Enterprise Linux 5 با 4GB RAM و دو هارد دیسک 250GB SATA-I می باشد. بر اساس hdparm، هارد دیسک 74MB/sec را برای خواندن های بافر تحویل می دهد. همه ماشین ها بر روی یک ریک قرار داشته و از طریق شبکه 1Gbps به سوییچ Cisco Catalyst 3750E-48TD متصل می شود. آزمایشات بر روی 10 ماشین انجام شدند.

سیستم ما در جاوا و فیتون اجرا شد. این سیستم از PostgreSQL 9.2.1 بر روی هر ماشین به عنوان واحد پردازنده استفاده می کند.

2-مجموعه داده های گراف: معیار عملکرد تطبیق الگوها بر روی چهار مجموعه داده گراف در نظر گرفته شد. کراف خرید محصولات آمازون، گراف تماشای ویدئوی یوتیوب، کراف گوگل وب، و گراف کاربران تویتر.

در صورت امکان، زیر گراف هایی که ما در مجموعه داده های فوق به دنبال هستیم از مقاله های پژوهشی توسط گروه های دیگری استخراج می شوند که از یک مجموعه داده برای بررسی تطبیق الگوی زیر گراف بهره می برند. به خصوص مجموعه داده های آماوزن در 24 و 25 و مجموعه داده های یوتیوب در 24 و داده های گوگل در 25 بررسی شدند. در زیر توصیف مفصلی از مجموعه داده ها ارایه می شود.

گراف محصول آمازون یک گراف جهت دار خریدار محصولات آمازون با 548,552 راس و 1,788,725 لبه می باشد. این گراف از بخش مشتریانی که این ایتم را خریداری کرده اند در وب سایت آمازون در تابستان 2006

موجود است. لبه محصول A-B به معنی خرید محصول B بعد از a توسط مشتری است. محصولات با چند مقوله به صورت صفر در نظر گرفته شدند.

گراف ویدئو یوتیوب: یک گراف جهت دار از یوتیوب در 2007 با 155,513 راس و 3,110,120 لبه است. لبه ویدئوی a به B به این معنی است که بینندگانی که a را دیده اند، B را نیز دیده اند. ویدئو ها به صورت نوار در نظر گرفته شدند که در آزمایشات ما به صورت صفات می باشند.

گراف وب: گراف وب گرافی است که در آن گره ها نشان دهنده صفحات وب بوده و لبه های جهت دار نشان دهنده هایپرلینک بین آن هاست. دارای 875,713 راس و 5,105,039 لبه بوده و در 2005 توسط شرکت گوگل به صورت بخشی از برنامه نویسی گوگل ارایه شد. راس ها نقشی در این مجموعه داده ندارند.

گراف تویتر: گراف اطلاعات کاربران تویتر است که در آن یک لبه از A-B به این معنی است که A ، B را بر روی تویتر دنبال می کند. این خود دارای 1,652,252 راس و 1,468,365,182 لبه بوده و در 2009 جمع اوری شد. ما تاریخ تصادفی را به هر لبه برای نشان دادن تاریخی که هر لبه ایجاد کرده بود نسبت داده و آن را به صورت یک صفت لبه ای در نظر گرفتیم.

3- پرس و جوی الگو: معیار ما متشکل از تطبیق الگوی ساختار و معنایی بوده و پرس و جوی ساختاری برای الگوی ها راس ها و لبه ها در نظر گرفته شده و پرس و جو های معنایی شاخل خصوصیاتی بر روی هر یک از گراف ها هستند.

شش پرس و جوی ساختاری در معیار وجود داشته و متشکل از دو پرس و جو در تحقیقات مربوطه و چهار پرس و جوی پیچیده تر می باشند که بهینه سازی آن ها سخت است. این شش پرس و جو در شکل 11 نشان داده شده اند. برای پرس و جو های معنایی، ما خصوصیات را به راس ها در پرس و جو های ساختاری می افزاییم. برای پوشش دادن موارد استفاده، ما تعدادی از راس ها را با خصوصیات متغیر در نظر گرفتیم. برخی از پرس و جو ها دارای خصوصیاتی بر روی هر راس در زیر گراف هستند.

علاوه بر زیر گراف های فوق ما از زیر گراف اضافی ابرای مجموعه داده های تویتر استفاده کردیم. مطالعات قبلی نشان دادند که یافتن مثلثات یگ گام مهم در اندازه گیری ضریب خوشه ای گره ها در شبکه اجتماعی می باشد. پرس و جو مثلث هایی را می یابد که قبل از سال 2009 تثبیت شده اند. از این روی، نتایج این پرس و جو جدا از پرس و جو های فوق تحلیل می شود.

الگوریتم ها: ما 5ا الگوریتم بهینه سازی پرس و جو را بررسی کردیم. الگوریتم حریص به عنوان معیار بوده که در هر تکرار، یک لبه با کم ترین هزینه را به لبه ای با زیر گراف تطبیق یافته اضافه می کند تا زمانی که همه لبه ها تطبیق شوند. دیکر چهار رویکرد بر اساس الکوریتم های ارایه شده در این مقاله هستند. DPخطی یک برنامه نویسی دینامیک بوده و تنها طرح های پرس و جوی خطی را در قالب برنامه نویسی دینامیک در نظر می گیرد در حالی که DP نقطه ای هر دو طرح های خطی و نقطه ای رل در نظر کرفته و کم هزینه ترین را انتخاب می کند. الگوریتم چرخه ای حریصانه، روش بهینه سازی تشخیص چرخه را به رویکرد حریصانه ارتباط داده در حالی که نوع چرخه ای- نقطه ای همه رویکرد ها را در نظر می گیرد.

ب:عملکرد تطبیق ساختاری

شکل 12 زمان اجرایی هر روش را بر روی تطبیق الگوی ساختاری برای محصولا ت امازون ، یوتیوب، و مجموعه داده های گراف وب نشان می دهد. به طور کلی الکوریتم های بهینه سازی تشخیص چرخه بهترین بوده و الگوریتم حریص توسط الگوریتم های دیگر عملکرد بهتری حاشیه بزرگ برای بیشتر پرس و جو ها دارد.

طرح های حریصانه قادر به اتمام طی یک ساعت در پرس و جو های 2 و 5 برای مجموعه داده های یوتیوب و پرس جو های 2, 3, 4, 5 و 5 برای گراف وب می باشند. عملکرد آن در مجموعه داده های آمازون به دلیل اندازه نسبتا کم زیاد بد نیست.

برای تحلیل بیشتر این نتایج، ما به بررسی پرس و جو های 2 و 4 می پردازیم. پرس و جوی 2 در مقایسه با پرس و جو های دیگر نسبتا ساده است ولی طرح تولید شده توسط الکوریتم حریص قادر به اتمام دو مجموعه داده نمی باشد. دلیل این است که بعد از اتصال $A\rightarrow C$ و $A\rightarrow D$, با $B\rightarrow C$ تطبیق پیدا کرده و به مجموعه

نهایی موجود متصل می شود. یک طرح بهتر شامل اتصال $A\rightarrow C$ و $A\rightarrow D,$ و سپس اتصال $B\rightarrow C$ و $B\rightarrow D$ است. DP خطی و تقطه ای علاوه بر الگوریتم های تشخیص چرخه این طرح برتر را تولید کردند.

پرس و جوی 4 جالب تر از پرس و جوی 2 می باشد زیرا همه الگوریتم های فوق تولید طرح های مختلف می کنند. این طرح ها در شکل 14 خلاصه شده اند. انواع حریصانه، خطی، نقطه ای به ترتیب از شش، چهار و سه اتصال استفاده می کنند. در هر مرحله پس از اتمام گام 2 و اضافه شدن یک عنصر جدید به مجموعه جواب، باید بررسی کنیم که آیا به یک جواب مطلوب رسیده‌ایم یا نه؟ اگر نرسیده باشیم به گام اول رفته و چرخه را در مراحل بعدی ادامه می‌دهیم. به زبان ساده، در روش حریصانه طی هر مرحله یک عنصر به روش حریصانه به مجموعه جواب اضافه شده، شرط محدودیت‌ها بررسی شده و در صورت نبود مشکل، عنصر و عناصر بعدی به همین ترتیب به مجموعه جواب اضافه می‌شوند. در طی این گام‌ها اگر به یک شرط نهایی خاص برسیم، یا امکان انتخاب عنصر دیگری برای اضافه کردن به مجموعه جواب وجود نداشته باشد، الگوریتم پایان یافته و مجموعه جواب به دست آمده به عنوان جواب بهینه ارائه می‌شود. توجه داشته باشید که ممکن است بر اساس نوع مساله، ترتیب اضافه شدن عناصر به مجموعه جواب اهمیت داشته باشد

با این حال با بررسی تعداد اتصالات می توان تفاوت ها را در عملکرد طرح ها توجیه کرد. طرح ایجاد شده توسط الگوریتم های چرخه ای دارای شش اتصال می باشد. برای الگوریتم حریصانه نیز همین تعداد اتصال است. با این حال عملکرد طرح آن بسیار بهتر از همه انواع دیگر است. دلیل این است که چندین سیکل را در کراف پرس و جو کشف کرده و در ابتدا این چرخه ها را تطبیق می دهد. محدودیت های ساختاری که چرخه ها را نشان می دهند به صورت یک پرس و جو یا گزاره پرس و جو می باشند که اندازه مجموعه نهایی را محدود می کند و موجب کاهش هزینه های اتصال و ترافیک می شوند. از این روی، اگرچه اتصالا بیشتر برای تطبیق چرخه ها انجام می شوند، طرح تولید شده منجر به هزینه کم و عملکرد بهتر می شود.

برای برجسته تر کردن تفاوت بین الکوریتم های بهینه سازی مبتنی بر تشخیص چرخه و مبتنی بر برنامه نویسی دینامیک، ما زمان و اندازه مجموعه های نهایی را برای طرح های خطی و نقطه ای بر روی مجموعه داده های

آمازون در شکل 13 برای سه پرس و جو در نظر می گیریم که برای آن ها این الکوریتم ها طرح ها و برنامه های مختلفی دارند. اگرچه بدیهی است که برنامه های چرخه موجب کاهش مجموعه نتایج نهایی با فاکتور های 3-X5 می شوند، عملکرد در بهبود زیاد هم خوانی ندارد. این ناشی از محدودیت تغییر دیتابیس های تحلیل تجاری بهینه سازی شده است که بر اساس PostgreSQL است که الگوریتم های بهینه سازی ضعیفPostgreSQL را اصلاح می کند. بهبود عملکرد برای برنامه های چرخه تحت شرایط بهینه سازی صورت می گیرد.

ج: عملکرد تطبیق معنایی

شکل 15 عملکرد هر الگوریتم را بر روی تطبیق الگوی معنایی برای داده های آمازون و یوتیوب نشان می دهد.

تفاوت کلیدی بین این مجموعه از ازمایشات و ازمایشات تطبیق ساختاری این است که عملکرد نسبی الگوریتم های مبتنی بر برنامه نویسی دینامیک و الکوریتم های مبتنی بر تشخیص سیکل معکوس می شود./ دلیل این است که مزیت کلیدی الکوریتم های تشخیص چرخه این است که چرخه ها به صورت نوعی گزاره انتخاب می باشند که موثر ترین آن ها کاهش تطبیق ساختاری است. با این حال، تطبیق معنایی امکان گزاره های انتخاب بسیار مهم است. این گزاره های انتخاب برای کاهش جست و جو به جای تشخیص چرخه مناسب هستند. جدول زیر اندازه مجموعه نتایج میانی را برای پرس و جو های 3 و 4 بر روی مجموعه داده های آمازون نشان می دهد. با کاهش مزیت اصلی این الگوریتم ها، معایب دیگر روشن تر می شوند و الکوریتم های مبتنی بر برنامه نویسی دینامیک برای بهینه سازی کاهش تعداد اتصالات در نظر گرفته شده و آن ها عملکرد بهتری از طرح های چرخه دارند.

عملکرد مثلث یابی

این جدول زمان اجرایی هر روش را بر روی مثلث یابی در گراف اجتماعی تویتر نشان می دهد.