به نام خدا

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

درس پردازش داده‌های حجیم

استاد حقیرچهرقانی

پروژه پایانی

سیدحسین خاتمی‌بیدگلی

علیرضا مازوچی

بررسی اجمالی مقاله

در این مقاله الگوریتم‌هایی برای پیش‌بینی تعداد مثلث‌ها و چهارتایی‌ها در مجموعه‌داده‌های جریانی گرافی ارائه شده است. مشابه با سایر الگوریتم‌های کلان‌داده در این مقاله نیز هدف پیدا کردن دقیق تعداد مثلث‌ها و چهارتایی‌ها نبوده است و تلاش برای یافتن تقریب با دقت مناسب در زمان مناسب و با پیچیدگی حافظه مناسب بوده است. به ادعای نویسندگان مقاله شمارش دورهای یک گراف و به طور خاص دورها به طول ۳ یا همان مثلث در کاربردهای مختلف اهمیت دارد و توجه ویژه‌ای بر روی آن شده است.

در این مقاله به صورت تک‌گذر (one-pass) جریان داده خوانده می‌شود. این مسئله می‌تواند باعث افزایش سرعت و حافظه شود چراکه هر عنصر در جریان تنها یک بار پردازش می‌شود و تنها تعداد کمی از آن‌ها (و نه همه آن‌ها) ذخیره می‌شوند. وقتی قرار باشد تنها تعداد کمی از یال‌ها ذخیره شود باید نمونه‌برداری انجام داد. برای نمونه‌برداری منطقا باید یال‌هایی انتخاب شوند که بیشترین تاثیر را تعداد مثلث‌های تخمینی دارند. از طرفی امکان برگشت به گذشته به دلیل تک‌گذر بودن میسر نیست؛ پس به ناچار الگوریتم باید در لحظه و بر اساس اطلاعات معدود ذخیره‌شده تصمیم بگیرد که چه یالی در ادامه مهم‌تر است و چه یالی کم‌تر مهم.

نگهداری یالی که در تعداد بیشتری از مثلث‌ها حضور دارد می‌تواند منجر به تخمین مناسب‌تری شود. چراکه می‌توان تعداد بیشتری از مثلث‌ها را ردیابی کرد. در مقاله یال‌های با تعداد مثلث بالا اصطلاحا یال سنگین نامیده شده است. اگر بتوان تشخیص داد که یک یال سنگین است یا خیر می‌توان تصمیم‌گیری بهتری در انتخاب یال‌های موثر داشت. اما سوال مهمی که پیش می‌آید این است که از کجا بتوان متوجه شد که یک یال سنگین است؟ اگر بخواهیم به طور دقیق سنگینی یک یال را تعیین کنیم باید کل جریان را مورد پردازش قرار دهیم که در این حالت امکان دست‌یابی به الگوریتمی تک‌گذر غیرممکن خواهد شد.

در مقاله پیشنهاد شده است که از یک عنصر پیش‌بینی کننده میزان سنگینی یال که اصطلاحا اوراکل (Oracle) نامیده می‌شود استفاده کنیم. بدین ترتیب در پردازش

جریان می‌توان یک کوئری را به اوراکل کوئری زد و اگر از نظر اوراکل آن یال به اندازه کافی سنگین بود ذخیره شود و در ادامه جریان اطلاعات مربوط به آن نگهداری شود.

گام بعدی آن است بدانیم چگونه می‌توانی اوراکلی داشت که پیش‌بینی کند. ایده مقاله آن بوده است که تنها بخش کوچکی از جریان داده را به طور کامل بررسی کنیم و با کمک آن اوراکل آموزش پیدا کند و در ادامه از آن استفاده شود. این ایده تنها زمانی کاربرد دارد که جریان داده تغییر شدیدی نداشته باشد. اگر این فرض برقرار باشد اوراکل به اندازه کافی دقیق خواهد بود تا در ادامه جریان از آن استفاده شود. نویسندگان مقاله مدعی هستند که این فرض برای مجموعه‌داده‌های واقعی برقرار است.

استفاده از یک اوراکل و متکی بودن به تنها یک گذر روی جریان ایده کلیدی مقاله است که باعث متمایزشدن کار تحقیقاتی آن‌ها از کارهای پیشین بوده است. در سایر کارهای تحقیقاتی همواره به بیشتر از یک گذر نیاز بوده است و به صورت صریح عنصری به نام اوراکل وجود نداشته است. البته شایان ذکر است که در کارهای پیشین عملا و به صورت ضمنی عملکرد آن در الگوریتم‌ها تاحدی وجود داشته است. مثلا در برخی از کارها گذر اول برای پیدا کردن یک دید اولیه از تعداد یال‌های با اندازه مثلث بالا بوده است که چنین چیزی مشابه کاری است که اوراکل در این مقاله می‌کند؛ با این تفاوت که در این مقاله از بخش کوچکی از جریان داده استفاده می‌شود و یک گذر کامل شکل نمی‌گیرد و اوراکل هم دقت کمتری دارد.

حسن دیگر مقاله ارائه الگوریتم‌های متنوع برای فرض‌های مختلف بوده است. به طور کلی پنج الگوریتم با تحلیل‌های کامل و اثبات‌های تئوری ارائه شده است. برای جریان داده دو مدل لیست مجاورت (Adjacency List) و ترتیب اختیاری (Arbitrary Order) در نظر گرفته شده است. برای اوراکل دو حالت باینری و مقداری ارائه شده است و نهایتا هم برای شمارش مثلث‌ها و هم شمارش چهارتایی‌ها روش پیشنهاد شده است. که در ادامه بیشتر بررسی می‌شود.

منظور از لیست مجاورت آن است که در جریان داده به توان برای یک راس به تمام رئوس مجاور آن دسترسی داشت. این درحالی که در مدل ترتیب اختیاری یال‌ها می‌توانند با هر ترتیبی ظاهر شوند و دلیلی برای آنکه یال‌های یک راس پشت‌سرهم بیایند وجود ندارد. قاعدتا در مدل دوم محدودیت‌ها بیشتر است.

همچنین دو نوع اوراکل در مقاله تعریف شده است. یکی از آن‌ها به شکل باینری تعیین می‌کند که یک یال سنگین است یا خیر و دیگری تعداد مثلث آن یال را پیش‌بینی می‌کند. اولین نوع آن اوراکل یال سنگین (Heavy Edge Oracle) نامیده می‌شود و دومی اوراکل مقداری (Value Oracle). نحوه پیش‌بینی هم بر اساس نتایج بخش کوچک ابتدا جریان داده است. یعنی اگر یک یال در این بخش تعدادی مثلث داشته باشد همان تعداد به عنوان پیش‌بینی در نظر گرفته می‌شود.

بر اساس نتایجی که در مقاله عنوان شده است این مقاله توانسته به دقت‌ها و سرعت‌های بهتری نسبت به روش‌های پیشین دست پیدا کند که با توجه به تک‌گذر بودن آن نکته خیلی مثبتی است.

در کنار نقاط قوه مقاله باید نکات منفی‌ای هم اشاره کرد؛ این مقاله بر روی جنبه‌های تئوری و اثبات‌های آن به اندازه کافی متمرکز شده است ولی جنبه‌های پیاده‌سازی مورد غفلت واقع شده است. خیلی از پارامترهای مهم تعیین نشده‌اند. بخشی از سودوکدها مبهم است و پیاده‌سازی آن را سخت می‌کند. مشخصات سخت افزاری مورد نیاز و نبود کد مقاله هم مشکلات را بیشتر کرده است.

خلاصه الگوریتم‌ها

در این مقاله الگوریتم ۱ برای مدل لیست مجاورت و اوراکل یال سنگین طراحی شده است. بر اساس پیش‌بینی اوراکل یال‌های جریان به سه دسته یال‌های سبک، متوسط و سنگین تقسیم می‌شوند. سپس هر کدام از این‌ها بسته دسته‌شان احتمال انتخاب دارند. یعنی یال‌های متوسط احتمال انتخاب بیشتری دارند تا یال‌های سبک. برای هر یک از سه دسته تعداد مثلث‌های مربوط به یال‌های آن شمرده می‌شود و بر احتمال آن تقسیم می‌شود. مثلا اگر نیمی از یال‌های سبک را انتخاب کرده باشیم و ۱۰۰۰ مثلث شکل گرفته باشد، تعداد کل مثلث‌های مربوط به یال‌های سبک ۲۰۰۰ پیش‌بینی می‌شود. برای شمارش مثلث‌ها هم از یک حالت خاص استفاده شده است. در مقاله برای یال xy دو نوع نماد مربوط به شمارش مثلث معرفی شده است: N_{xy} و R_{xy}. N_{xy} شامل تمام مثلث‌هایی می‌شود که یال xy عضو آن باشد ولی R_{xy} شامل

مثلث‌هایی می‌شود که هم xy عضو آن باشد و هم عضو سوم در جریان داده در حالت لیست مجاورت بین دو عضو دیگر آمده باشد. محاسبه N_{xy} به غیر از در زمان آموزش اوراکل ممکن نیست ولی R_{xy} را می‌توان محاسبه کرد. چراکه اگر xy دیده شود و ذخیره شود می‌توان تعداد مثلث‌ها را تا زمان دیدن yx شمارش کرد.

الگوریتم ۲ و ۳ هر دو مربوط به مدل لیست مجاورت و برای اوراکل مقداری طراحی شده‌اند. اما جزئیات هر دو الگوریتم کاملا متفاوت و خلاقانه است. در الگوریتم ۲ از متغیر تصادفی نمایی استفاده شده است و بر پایه احتمالات تعداد مثلث‌ها تخمین زده می‌شود. برای آنکه تخمین پایدارتر باشد برخلاف الگوریتم ۱ تعداد زیادی پیش‌بینی انجام می‌گیرد و میانه آن به عنوان پیش‌بینی نهایی معرفی می‌شود. همچنین در این الگوریتم سعی شده است که به مرور یال‌های نگهداری شده بروز شود. چیزی که در الگوریتم ۱ دیده نمی‌شد. علت این تفاوت هم به اوراکل بر می‌گردد. در الگوریتم ۱ اوراکل باینری است و نمی‌تواند بین یال‌های سنگین تفاوتی بگذارد و به طور مشابه توانایی تمایز بین یال‌های سبک را هم ندارد.

در الگوریتم ۳ یال‌ها به تعدادی بازه نابرابر نگاشت می‌شوند که تعدادشان i تاست. اولین بازه متعلق به یال‌های دارای تعداد مثلث خیلی کم و بازه‌های بعد به ترتیب جایگاه یال‌های سنگین‌تر است. احتمال انتخاب یال متناسب با همین بازه‌بندی تغییر می‌کند. یعنی یال‌های بازه اول احتمال انتخاب کمتری دارند تا یال‌های بازه‌ی بعدی و الی آخر. همچنین برای هر بازه j مجموعه مجزا درنظر گرفته شده است که کمک می‌کند تا میانه‌ی چندین تخمین، تخمین نهایی باشد تا بدین شکل واریانس پیش‌بینی کم شود.

الگوریتم ۴ برخلاف سه الگوریتم دیگر مربوط به مدل ترتیب اختیاری است. در این الگوریتم مانند الگوریتم ۱ از اوراکل یال سنگین استفاده می‌شود. همچنان مطابق با منطق الگوریتم ۱ یال‌ها به دو دسته سبک و سنگین تقسیم می‌شوند که احتمال انتخاب متفاوتی دارند. به بیان دقیق‌تر یال‌های سنگین تماما انتخاب می‌شوند ولی یال‌های سبک نمونه‌برداری می‌شوند. نهایتا وقتی یک یال جدید می‌رسد باید بررسی کرد که با یال‌های پیشین می‌تواند مثلث تشکیل دهد یا خیر.

انتخاب مجموعه‌داده و تنظیمات

برای این پروژه سه مجموعه‌داده Oregon، CAIDA2006 و CAIDA2007 انتخاب شده است. مطابق با مقاله از اولین گراف هر سه مجموعه‌داده به عنوان گراف آموزشی اوراکل استفاده شده است. به ترتیب از گراف ۴-ام، ۳۰-ام و ۲۵-ام سه مجموعه‌داده مذکور به عنوان گراف ارزیابی بهره گرفتیم.

در مقاله به طور کلی پارامترهای الگوریتم غالبا بیان نشده است و ما با استفاده از آزمون و خطا سعی در پیداکردن پارامترهایی داشتیم که همزمان دقت قابل قبول و پیچیدگی حافظه و زمانی قابل قبول داشته باشد. در جدول زیر لیست کلی پارامترهای استفاده شده در پروژه آورده شده است.

حوزه پارامتر	پارامتر	مقدار
اوراکل مقداری	α	۱
	β	۱۰
	K	۱
الگوریتم ۱	ϵ	۰/۰۵
	α	۰/۱
	β	۰/۵
	γ	۰/۰۱
الگوریتم ۲	ϵ	۰/۰۵
	c	۰/۲
	ضریب H	۰/۵
الگوریتم ۳	ϵ	۰/۰۵
	c	۰/۵
	ضریب i	۳
	ضریب j	۳
الگوریتم ۴	ϵ	۰/۰۵
	C	۰/۷

در جدول پارامترها تعدادی پارامتر با پیشوند ضریب ذکر شده است. این‌ها مربوط به متغیرهایی هستند که در سودوکد به صورت y=$O(x)$ نشان داده شده بودند. در این موارد منطقا حق داریم که y را برابر با ضریب ثابت و کوچکی از x قرار دهیم. پارامترهای مذکور مربوط به همین موارد هستند.

نتایج

برای هر الگوریتم سعی کردیم بسته به زمان اجرا چندین مرتبه اجرا گرفته شود و میانگین و انحراف معیار دفعات مختلف به عنوان جواب نهایی برگردانده شود. مطابق با مقاله مقدار میانگین بیان شده است و بعد از آن با نماد ± مقدار انحراف معیار تعیین شده است.

برای معیار خطا هم مطابق با مقاله از خطای نسبی که رابطه زیر آمده است استفاده کردیم. در این معیار T تعداد مثلث واقعی و $\tilde{T}$ تعداد مثلث پیش‌بینی‌شده است.

$$\left|1-\frac{\tilde{T}}{T}\right|$$

ابتدا نتایج مدل لیست مجاورت آورده می‌شود و سپس به بررسی نتایج مدل ترتیب اختیاری خواهیم پرداخت.

برای الگوریتم ۱ به تعداد ۲۵ بار اجرا گرفته شده است و نتایج آن مطابق جدول زیر است:

مجموعه‌داده	فضا (#یال)	زمان اجرا (s)	خطای نسبی (٪)
Oregon	۵×۱۰^{۳}	۴	۵.۳۲ ± ۴.۴۹
CAIDA2006	۵.۷×۱۰^{۳}	۷	۴.۵۶ ± ۳.۳۱
CAIDA2007	۵.۷×۱۰^{۳}	۸	۳.۰۸ ± ۲.۳۹

برای الگوریتم ۲ به تعداد ۳ بار اجرا گرفته شده است و نتایج آن در جدول زیر ارائه شده است:

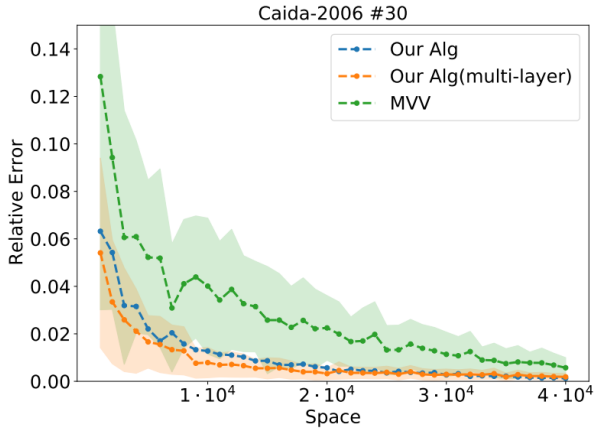
مجموعه‌داده	فضا (#یال)	زمان اجرا (s)	خطای نسبی (%)
Oregon	۴.۸×۱۰^{۳}	۲۲۹	۱۶.۰۶ ± ۷.۴۸
CAIDA2006	۱۰.۶×۱۰^{۳}	۸۲۹	۱۴.۷۴ ± ۸.۲۲
CAIDA2007	۹×۱۰^{۳}	۸۸۰	۴.۵۱ ± ۳.۵۴

برای الگوریتم ۳ به تعداد ۵ بار اجرا گرفته شده است و نتایج آن در قالب جدول زیر آورده شده است:

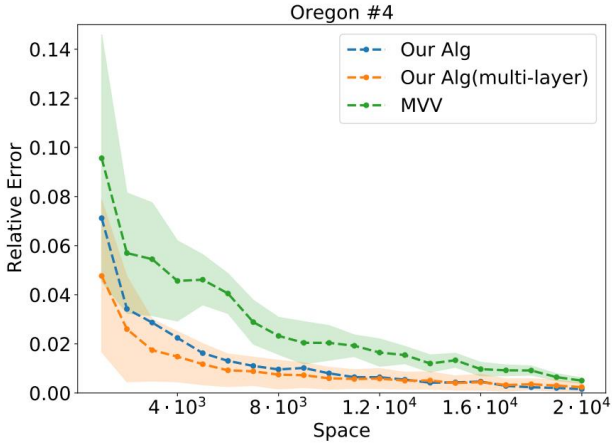
مجموعه‌داده	فضا (#یال)	زمان اجرا (s)	خطای نسبی (%)
Oregon	۱۶×۱۰^{۳}	۲۷	۳.۵۸ ± ۲.۷۲
CAIDA2006	۲۱.۳×۱۰^{۳}	۷۷	۴.۵۳ ± ۰.۸۲
CAIDA2007	۲۱×۱۰^{۳}	۷۸	۴.۴۹ ± ۲.۷۰

در میان پیاده‌سازی‌هایی که ما انجام داده‌ایم الگوریتم ۱ در مجموع هر سه شاخص فضا، زمان اجرا و خطای نسبی از دو الگوریتم ۲ و ۳ بهتر بوده است. هر از یک از دو الگوریتم دیگر از منظری دارای ایراد است. الگوریتم ۲ پیچیدگی فضایی مناسبی دارد ولی زمان اجرای آن و خطای نسبی آن مناسب نیست؛ از طرف دیگر الگوریتم ۳ خطای نسبی خیلی خوبی دارد ولی پیچیدگی فضایی آن خیلی خوب نیست.

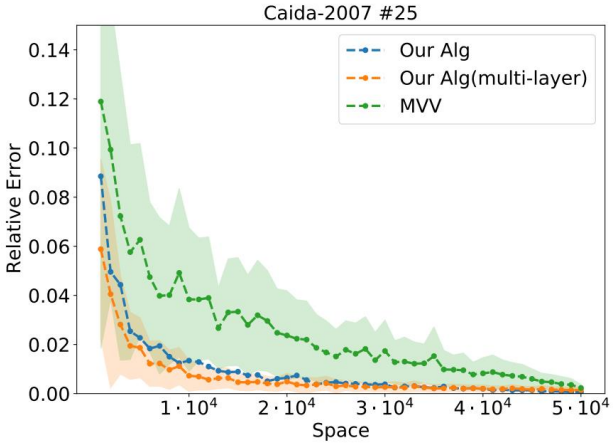
در مقاله برای مدل لیست مجاورت و سه مجموعه‌داده انتخاب‌شده نمودار زیر یافت شد. بر اساس این نمودار می‌توان بررسی کرد که آیا توانسته‌ایم به دقت مقاله برسیم یا خیر. برای انجام مقایسه تنها نتیجه الگوریتم ۱ به عنوان پیاده‌سازی برگزیده ما بررسی می‌شود.



(a) CAIDA 2006



(a) Oregon



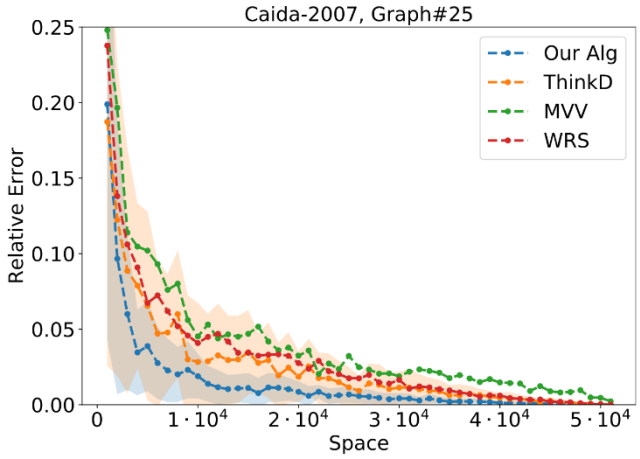
(b) CAIDA 2007

برای مجموعه‌داده Oregon ما با $۱۰^۳ \times ۵$ یال به خطای ۵/۳۲٪ رسیده‌ایم. مطابق با ادعای مقاله باید به خطای ۲٪ می‌رسیدیم که موفق نشدیم. در مجموعه‌داده CAIDA2006 با تعداد $۱۰^۳ \times ۵.۷$ یال به خطای ۴/۵۶٪ رسیده‌ایم درحالی که در مقاله به دقت ۳٪ رسیده بودند. نهایتا برای CAIDA2007 با تعداد $۱۰^۳ \times ۵.۷$ یال به خطای ۳/۰۸٪ دست پیدا کردیم که در مقاله حدودا عدد ۳/۵٪ دیده می‌شود که از عدد ما بدتر است. پس در مجموع می‌توان گفت که نتایج حاصل‌شده با نتایج مقاله فاصله دارد ولی تفاوت جدی نیست و با بهبود پیاده‌سازی و تنظیم بهتر پارامترها امکان رسیدن به اعداد مقاله وجود دارد. در عین حال باید توجه کرد که خطای ما با پارامتر ϵ که ۰/۰۵ تعیین شده است تقریبا در همه موارد سازگار است.

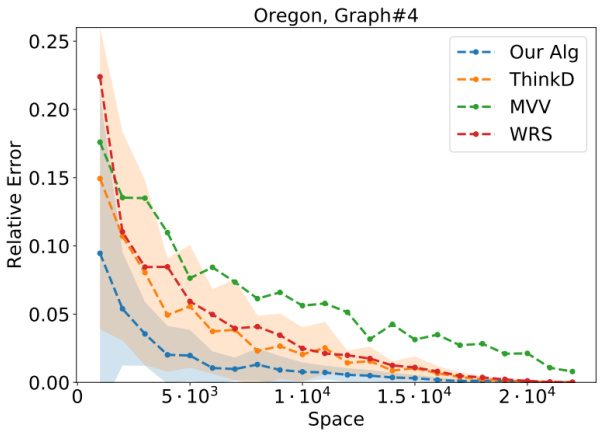
برای الگوریتم ۴ فقط یک بار اجرا گرفته شده است و نتیجه همان یک بار در جدول زیر آمده است:

مجموعه‌داده	فضا (#یال)	زمان اجرا (s)	خطای نسبی (%)
Oregon	۲.۴×۱۰^{۳}	۱۷۶	۷.۵۴
CAIDA2006	۷×۱۰^{۳}	۱۹۶۹	۵۸۴.۱۹
CAIDA2007	۶.۹×۱۰^{۳}	۲۲۳۴	۵۳۵.۸۲

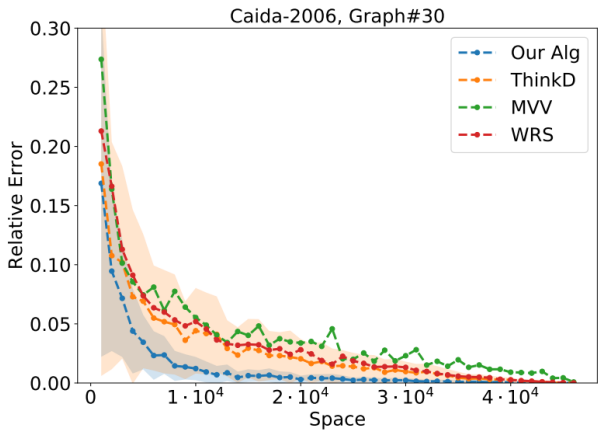
در مقاله برای مدل ترتیب اختیاری نمودارهای زیر پیدا شد:



(a) CAIDA 2007



(a) Oregon



(b) CAIDA 2006

برای مجموعه‌داده Oregon با ۲.۷×۱۰^{۳} یال به خطای ۷.۵۴٪ رسیده‌ایم که این عدد در مقاله ۴٫۵٪ است که فاصله معناداری با پیاده‌سازی مقاله دارد. برای مجموعه‌داده CAIDA 2006 و CAIDA 2007 متاسفانه جواب اصلا مناسب نبوده است.

چالش‌های پیاده‌سازی

در این پروژه با چالش‌های مختلف پیاده‌سازی مواجه بودیم که مهم‌ترین آن‌ها را در ادامه بیان می‌کنیم:

یال‌های ناقص در مجموعه‌داده: در مجموعه‌داده Oregon برخی از یال‌ها تنها از یک راس به دیگری وجود داشت و نه بالعکس. در این موارد تمام یال‌های یک طرفه را به صورت دو طرفه در نظر گرفتیم؛ به بیان دیگر اگر یال xy در مجموعه‌داده وجود داشت یال yx هم به آن اضافه شد. اگر یال‌ها را اضافه نمی‌کردیم احتمال شکست الگوریتم‌ها زیاد می‌شد و تعریف مثلث هم دارای چالش می‌شد.

تعیین پارامترها: همانگونه که پیش‌تر اشاره شد در مقاله تعداد کمی از پارامترها به صورت صریح مقدار دهی شده است و تعیین سایر پارامترها تنها از طریق آزمون و خطا و شهود ذهنی که از الگوریتم داشتیم ممکن بود. این امر باعث شد تا وقت زیادی صرف تعیین پارامترها گردد.

یافتن ساختمان‌داده مناسب: زمان اجرای خوب تنها با استفاده از ساختمان‌داده‌های بهینه برای جاهای مختلف ممکن است. در این پروژه هم پیدا کردن ساختمان‌داده مناسب نظیر دیکشنری، مجموعه، لیست و هیپ یک چالش پیاده‌سازی بود.

ابهام‌های پیاده‌سازی: با توجه به اینکه کد مقاله موجود نبود و سودوکدها دارای ابهام‌هایی بودند برای قسمت‌های از سودوکد تفسیرهای مختلف قابل برداشت بود که حل آن‌ها و انتخاب یکی از آن‌ها محل تردید بود.