دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گروه فناوری اطلاعات

پروژه پایانی

درس تجارت الکترونیکی ۲

استاد درس

دکتر آزاده محمدی

دانشجویان

سیدمحمد هاشمی

سجاد یزدان‌پرست

دی ۱۴۰۰

فهرست مطالب

۱- مقدمه

محتوای کامل پروژه شامل کدهای پیاده‌سازی، مجموعه‌داده، pdf مقاله‌ها، آدرس مقاله و مستند بر روی مخزن گیت‌هاب پروژه قرار دارد. مقاله‌های انتخاب شده‌ی ما هر دو مربوط به روش انتشار برچسب[1]در تشخیص اجتماع[2]هستند و علت انتخاب آن‌ها شفافیت قابل قبول در بیان مطالب و به‌خصوص الگوریتم مورد نظر و همچنین در مورد مقاله‌ی پیاده‌سازی شده، صراحت در توضیح نحوه‌ی ارزیابی و جزئیات مجموعه‌داده‌های مورد استفاده بوده است.

۲- مقاله اول (Community Detection by Information Flow Simulation)

۲-۱- چکیده

آنچه نویسنده‌ی مقاله دنبال کرده است، به طور خلاصه بیان فقدان روش‌های مقیاس‌پذیری برای تشخیص اجتماعات است که از جریان اطلاعات استفاده کرده باشند و همچنین ارائه‌ی یک روش سریع و مقیاس‌پذیر برای شبکه‌های اجتماعی در این زمینه را فارغ از وزن‌دار یا جهت‌دار بودن شامل می‌شود.

همچنین ضمن اشاره به پیچیدگی زمانی و فضایی مناسب رویکرد پیشنهادی ($O(|E|)$) ، به برتری این رویکرد نسبت به Markov Clustering Algorithm از نظر دقت و مقیاس‌پذیری (کاربر در شبکه‌های بزرگ) اشاره شده است.

۲-۲- مقدمه

ابتدا اشاره‌ای بر تحقیقات ابتدایی بر روی شبکه‌های اجتماعی با موضوع دوستی بین افراد و قضیه‌ی small-world problem شده است که در مورد ارتباط نزدیک افراد جهان صحبت می‌کند (دسترسی افراد با حداکثر ۶ نفر دوست واسطه). عنوان می‌کند که تحلیل شبکه‌های اجتماعی درخرده‌فروشی، مدیریت بلاهای طبیعی، تبلیغات و بسیاری موارد دیگر کاربرد دارد.

معمولاً دید عمومی به شبکه، مانع مشاهده ساختارهای مهمی تحت عنوان اجتماع‌ها می‌شود که با دید محلی به شبکه‌ی اجتماعی، قابل مشاهده و نقش مهم آن‌ها ملموس و واضح است. اجتماع‌ها یک زیرگراف یا زیرمجموعه‌ای از رئوس داخل شبکه هستند که ارتباطات قوی‌تری با یکدیگر نسبت به سایر رئوس دارند.

کاربردهای شناسایی اجتماع‌ها در تجارت الکترونیکی: از کاربردهای آن می‌توان به بازاریابی هدف‌گذاری شده و ویروسی اشاره کرد که بر اجتماع‌ها متمرکز باشد (هر اجتماع می‌تواند هدف یک کمپین تبلیغاتی ما قرار بگیرد و همچنین از نظر تبلیغات ویروسی، سرعت نشر تبلیغ ما به‌صورت دهان به دهان در بین افراد هر اجتماع بسیار زیاد است). مورد دیگری از استفاده‌ها، بحث ارائه‌ی خدمات و اطلاعات یا آگاه‌سازی افراد (به‌خصوص در مورد اخبار و اطلاعیه‌های کسب‌وکار ما یا خدمات و موارد مشابه) است که باز هم با انتشار آن در هر اجتماع، سرعت نشر و دسترسی بهتری خواهیم داشت. همچنین اگر بخواهیم تاثیرگذاری حداکثری بر مخاطبان خود در شبکه‌ی اجتماعی داشته باشیم، بهترین کار، شناسایی اجتماع‌ها و تمرکز بر آن‌هاست. زیرا ارتباطات عمیق و قوی بین افراد در این بخش‌ها بیشتر از سایرین بوده و افراد می‌توانند تأثیر عمیق‌تری بر یکدیگر بگذارند.

[1] Label propagation

[2] Community Detection

این تأثیر عمیق می‌تواند به متقاعدسازی افراد برای استفاده از محصول ما یا موارد تجاری دیگر منجر شود.

تشخیص اجتماع‌ها به سبب تعداد زیاد افراد در شبکه فرآیند سختی به دنبال دارد(کاما؟)، به طور مثال فیسبوک[1] حدود ۱,۹ میلیارد کاربر دارد و این یعنی مجموعه‌داده‌های بسیار بزرگی که نیاز به یک الگوریتم بهینه دارند. اغلب رویکردهای موجود به سبب پیچیدگی زمانی یا حافظه‌ای مقیاس‌پذیر نیستند اما رویکرد ارائه شده در این مقاله نه تنها از این نظر مناسب است بلکه با استفاده از داده‌ساختار بهتر، سبب کارایی بیشتر شده و برای شبکه‌های پویا نیز قابل بسط و توسعه است. این رویکرد ابتدا افراد مؤثر در شبکه را به‌عنوان منشأ اطلاعات یافته و با شبیه‌سازی جریان احتمالی اطلاعات (اطلاعات با احتمال مشخصی از هر یال و مسیر جریان می‌یابد) از این افراد به داخل شبکه، اجتماع‌ها را با برچسب‌گذاری تشخیص می‌دهد. برای پیچیدگی زمانی رویکرد نیز یک حد بالا بر اساس ویژگی‌های مورد استفاده‌ی شبکه‌اجتماعی که در ادامه ذکر خواهند شد، ارائه شده است.

۲-۳- مفاهیم مقدماتی

اصلی‌ترین مفهوم، گراف شبکه‌ی اجتماعی هست که در این مقاله به‌صورت مجموعه‌ای از رئوس و یال‌های جهت‌دار وزن‌دار در نظر گرفته شده که وزن یال‌ها مرتبط با احتمال انتقال اطلاعات از طریق آن‌ها است.

تعریفی از مجاورین رأس ($N(v)$)، درجه‌ی خروجی غیروزن‌دار ($\delta(v)$)، درجه‌ی ورودی غیروزن‌دار ($\delta_i(v)$)، درجه‌ی خروجی وزن‌دار ($\delta^*(v)$) برای هر رأس و همچنین تعریفی برای قطر گراف یعنی طولانی‌ترین کوتاه‌ترین مسیر در آن ارائه شده است.

علاوه بر تعاریف فوق که در مقاله استفاده شده، خصوصیاتی از شبکه‌های اجتماعی نیز که در مقاله کاربرد داشته است، ذکر شده اند از جمله:

- communities یا همان جوامع که قرار است شناسایی شوند
- مفهوم اثر دنیای کوچک که به کوتاهی مسیر ارتباط هر دو فردی در جهان به واسطه‌ی ارتباطات اشاره دارد (به بحث مؤلفه‌های قویاً همبند و وجود مؤلفه‌ی غول‌آسا در اکثر شبکه‌های اجتماعی واقعی نیز اشاره شده که در محاسبه‌ی پیچیدگی زمانی رویکرد ارائه شده مورد استفاده قرار می‌گیرد)
- گشت تصادفی (random walk) که همان پیمایش گراف بر حسب احتمال است. به عبارتی هر یال با یک احتمالی پیمایش می‌شود.

۲-۴- مدل

در این بخش به توضیح نحوه‌ی ذخیره‌سازی و بازیابی گراف اشاره شده است.

در این مقاله به جای ماتریس مجاورت از یک ساختمان‌داده پویا مشابه به لیست مجاورت استفاده شده است.

ورودی‌ها در هر خط به فرم $u\ v\ w_{uv}$ است که یک یال از u به v را با وزن wuv را نشان می‌دهد (هر وزن یا بزرگتر و مساوی ۱ است، یا اگر مجموعه‌داده به این صورت نیست باید به این فرم در آید). یک جدول هش در نظر می‌گیریم که کلید آن uهای به فرم ($\delta(u) > 0$) است و مقدار یک دوتایی است. هر کدام از اعضای این دوتایی یک لیست kتایی است که ($k = |N(u)|$) اولین لیست، شامل مجاورین u یا همان N(u) است و دیگری وزن یال‌های نظیر به نظیر یا W(u).

[1] Facebook

این ساختمان‌داده و پیاده‌سازی دو ویژگی دارد؛ یک، پیچیدگی مناسب و دیگری ماهیت پویا.

در مورد پیچیدگی زمانی، بازیابی لیست مجاورین و وزن‌ها برای هر u دارای پیچیدگی O(1) است. افزودن یک رأس به لیست نیز همینطور. دسترسی به وزن یال مشخص uv نیز برابر O(k) طول می‌کشد. همچنین دسترسی‌های الگوریتم ما به این یال‌ها معمولاً به ترتیب رخداد آن‌ها اتفاق می‌افتد که تعداد پیمایش کامل لیست را کاهش می‌دهد.

از نظر حافظه، ساختار مورد استفاده‌ی ما مجموعه‌ای از رئوس به‌عنوان کلید دارد ($|\{v \in V, \delta(v) > 1\}| \leq |E|$) که تعداد آن ها نهایتاً برابر تعداد یال‌هاست. ذخیره‌ی مقدارهای جدول هش نیز شامل دو لیست مجاورین و وزن‌ها است که در مجموع 2|E| فضا اشغال می‌کند. در نتیجه پیچیدگی حافظه‌ی ما $O(2|E| + 2|\{v \in V, \delta(v) > 1\}|)$ است یا O(|E|).

از نظر پویایی، اضافه کردن رأس به ماتریس، پیچیدگی O(n) دارد که n تعداد رئوس قبلی است. در حالی که برای رأسی مانند v پیچیدگی اضافه کردن یال‌های آن به اندازه‌ی مجموع درجه‌ی خروجی و ورودی غیروزن‌دار آن است. که به نسبت |E| بسیار کم است. این به ما اجازه می‌دهد در آینده رویکرد را برای شبکه‌های پویا نیز بسط دهیم.

درمورد گشت احتمالی نیز معیار باید مشخص شود. در روش کلاسیک خوشه‌بندی گراف از احتمال پیمایش به شکل زیر استفاده می‌شود:

$$P(u,v) = \frac{w_{uv}}{\delta^*(u)}$$

این فرمول برابر وزن یال بر روی درجه‌ی خروجی وزن‌دار مبدأ آن است. اگر گراف بدون جهت باشد تقریباً می‌تواند مناسب باشد اما در کل دو کمبود و ایراد دارد:

اولین ایراد این است که در روش کلاسیک، گشت احتمالی صرفاً بر روی یکی از یال‌ها در هر لحظه امکان پیشرفت دارد در حالی که در فضای واقعی، اطلاعات به‌صورت موازی، همزمان و مستقل می‌تواند انتقال یابد. مثال آن نیز پست‌هایی است که افراد در فضای مجازی می‌گذارند. بنابراین جمع احتمالات یال‌های خروجی یک رأس دیگر الزامی ندارد برابر ۱ شود. پس برای بهتر شدن مدل، فرمول زیر یشنهاد شده است:

$$P(u,v) = \left(\frac{w_{uv}}{\delta^*(u)}\right)^{\beta} \; ; \; \beta \in (0,1)$$

با این کار مقدار احتمال برای هر یال را بالاتر می‌بریم و به واقعیت نزدیک می‌کنیم.

برای β مقدار ¼ در نظر گرفته شده که برای مجموعه‌داده‌های مقاله، جریان پویای اطلاعاتی خوبی رقم می‌زند.

فرمول زیر حالت نهایی احتمال یال هاست.

$$P(u,v) = \frac{\sqrt[4]{w_{uv}}}{\sqrt[4]{\delta^*(u)}}$$

دومین ایراد مربوط به متعادل و متوازن نبودن وزن یال‌ها در قسمت‌های مختلف گراف است که روش کلاسیک اصلا به آن توجه ندارد.

به‌عنوان مثال، ۳ مورد از این عدم توازن ذکر مورد بررسی قرار گرفته است:

۱- رأسی با یال‌های با وزن بالا: در این مورد در روش کلاسیک فقط یکی از یال‌ها در هر زمان احتمال پیمایش داشت

در حالی که وزن آن‌ها نشان می‌دهد همگی مستعد انتشار اطلاعات در یک زمان هستند. و همچنین طبق فرمول احتمال انتشار برای همه‌ی آن‌ها به دلیل نزدیک بودن وزن‌هایشان توزیع شده و بسیار پایین است.

۲- رأسی با یال‌های بسیار ولی وزن کم: در این حالت با افزایش تعداد یال‌ها احتمال به‌شدت در روش کلاسیک پایین می‌آید. درحالی که طبق جداول ارائه شده از محاسبه‌ی احتمال با فرمول ارائه شده این یال‌ها همگی احتمال مناسبی دارند. در این مورد باید به مثال پست گذاشتن در فضای مجازی اشاره کرد. وقتی پستی گذاشته می‌شود افراد حتی کم ارتباط هم احتمال زیادی دارد آن پست را ببینند. بنابراین زمان خلق اطلاعات در شبکه مهم است و باید در نظر گرفته شود.

۳- رأسی با تعداد یال زیاد و تعداد اندکی یال سنگین وزن: بر اساس محاسبات روش کلاسیک به یال‌های سنگین وزن هم احتمالی کمتر از ۰,۵ می‌دهد. در حالی که در فرمول ارائه شده کاملاً برعکس است و احتمال مناسبی برای آن‌ها در نظر گرفته می‌شود.

۲-۵- رویکرد

تشخیص اجتماع در یک گراف، یافتن یک مجموعه از k زیرگراف است که هر یک ساختاری به‌خوبی متصل دارند. راه حل این مسئله به‌صورت بهینه با استفاده از معیارهای اتصال یک مسئله‌ی NP-hard است، یعنی در پیچیدگی چند جمله‌ای قابل حل نیست. بنابراین رویکرد در این مقاله نیز یک رویکرد تقریبی است.

ابتدا افراد مؤثر در شبکه را می‌یابیم و از آن‌ها به عنوان مبدأ انتشار اطلاعات استفاده می‌کنیم. یعنی جریان یافتن اطلاعات یا همان برچسب‌ها را به احتساب احتمال هر یال شبیه‌سازی می‌کنیم تا زمانی که کل گراف برچسب‌گذاری شود یا بعد از تعداد مشخصی (λ) تکرار که حالت پایداری پیدا کند یعنی دیگر برچسبی منتشر نشود.

بر خلاف همه‌ی رویکردها، رویکرد این مقاله پیدا کردن برچسب و اجتماع مشخص برای هر رأس را ضمانت نمی‌کند. در واقع این انتخاب آگاهانه‌ای است که با توجه به شرایط شبکه‌های واقعی در نظر گرفته شده است. در شبکه های واقعی افرادی هستند که فعالیت منظمی ندارند و اطلاعات تولید یا منتشر نمی‌کنند. با این حال با پذیرش نتیجه‌ی غیردقیق می‌توان نتایجی به دست آورد و در ارزیابی، رئوس غیرفعال را اجتماع تک‌عضوی در نظر می‌گیریم. با این حال در مجموع این مقاله، به مسئله به‌صورت بهینه‌سازی بر اساس یک سری معیار اتصال نگاه نکرده است، زیرا هر کدام از این معیارها به تنهایی محدودیت‌هایی را باعث می‌شوند. در عوض از انتشار برچسب استفاده کرده است و برچسب‌ها را با معیارهای مختلف مورد سنجش قرار داده است.

به‌طور کلی مقاله شامل دو الگوریتم مهم است که رویکرد را تشکیل می‌دهند. اولین الگوریتم، برای یافتن افراد تاثیرگذار در شبکه است که به آن‌ها آلفا گفته می‌شود. این افراد می‌توانند منبع خوبی برای انتشار اطلاعات باشند زیرا معمولاً دارای تعداد زیادی دوست یا دنبال‌کننده هستند و همچنین ارتباطات قوی با این افراد دارند. به بیان دیگر تعداد یال‌های خروجی آن‌ها و همجنین مجموع وزن این یال‌ها زیاد است. وجود چنین افرادی در شبکه اثبات شده است.

الگوریتم یافتن آلفاها، گراف G و پارامتر k را دریافت می‌کند و مجموعه رئوس آلفا را خروجی می‌دهد.

این کار را با استفاده از دو لیست NumRank و DegRank انجام می‌دهد که شامل رئوس G هستند. اولی را بر حسب تعداد یال خروجی و دومی را بر حسب مجموع وزن یال‌های خروجی مرتب می‌کند. سپس از بین k درصد برتر این دو لیست،

مشترک‌ها را به عنوان آلفا تشخیص داده و مجموعه آلفاها را باز می‌گرداند.

نکته در مورد پارامتر k: یکی از مشکلاتی که خود نویسنده نیز به آن اشاره کرده است یافتن مقدار بهینه برای k است که در این مقاله به آن پرداخته نشده و حل آن دشوار است. برای مجموعه‌داده‌های مقاله نیز صرفاً حدس‌های آزموده‌شده‌ای بر اساس هر مجموعه، انجام شده و به کار رفته است.

۲-۵-۱- پیاده‌سازی

الگوریتم یافتن آلفاها در فایل alpha_detection.py در مخزن گیت‌هاب[1] پروژه که آدرس آن در ابتدای مستند آورده شده، پیاده‌سازی شده است.

پیاده‌سازی این الگوریتم منطبق با شبه کد و به‌صورت بهینه پیاده‌سازی شده است. به این صورت که از توابعی با پیچیدگی زمانی بهتر برای افزودن به لیست استفاده شده یا در اشتراک‌گیری انتهای الگوریتم مطابق با مرتبه‌ی زمانی بیان شده در بخش‌های بعدی مقاله، با مرتبه‌ی O(|V|log|V|) انجام شده است.

۲-۶- الگوریتم

رویکرد اصلی ارائه شده به این صورت است که پس از به دست آوردن افراد مؤثر، این افراد را منبع اطلاعات در نظر می‌گیریم و نشر اطلاعات (برچسب‌ها) را شبیه‌سازی می‌کنیم. یعنی اگر α_i اولین آلفایی باشد که u از آن اطلاعات می‌گیرد. u به اجتماع iام تعلق دارد. پس تعداد اجتماع‌ها در نهایت برابر تعداد آلفاها است. اگر آلفایی به یک رأس مثل x دسترسی نداشته باشد، x به هیچ اجتماعی تعلق ندارد. (ما در ارزیابی خود این رئوس را یک اجتماع مستقل در نظر گرفتیم)

ابتدا به هر آلفا یک برچسب داده می‌شود، سپس به‌صورت موازی از هر آلفا به مجاورینش بر حسب احتمال P(u, v) که در بخش‌های قبل توضیح دادیم، برچسب انتشار می‌دهیم. در گام بعدی، هر رأس دیده‌شده در تکرار قبلی برچسب دریافتی‌اش را به مجاورین به شکل مشابه انتشار می‌دهد. رأسی دیده‌شده که برچسب دریافت کرده باشد. برای هر تکرار، یال‌هایی که قبلاً جریان داده از آن‌ها عبور نکرده است، مسیرهای فعال باقی خواهند ماند. یعنی در ادامه شاید اطلاعات در تکرارهای بعدی از آن‌ها عبور کند. به شکل مشابه رئوسی که مجاور دیده‌نشده دارند، فعال باقی می‌مانند. این روند تا برچسب‌گذاری شدن همه‌ی رئوس یا رسیدن گراف به یک حالت پایدار (۳ تکرار پیاپی بدون برچسب‌گذاری جدید) ادامه می‌یابد.

در هر تکرار، رئوس فعال را به زیرمجموعه‌هایی تقسیم می‌کنیم و با استفاده از چند نخی به‌طور موازی فرایند انتشار انجام می شود.

نکته: انتخاب تعداد این زیرمجموعه‌ها بستگی به سخت‌افزار دارد و در صورت زیاد در نظر گرفتن آن‌ها باعث افزایش زمان انتظار نخ‌ها برای استفاده از منابع خواهد شد.

به طور جزئی‌تر الگوریتم انتشار برچسب، گراف G، آلفاها و آستانه تکرار (λ) را دریافت می‌کند و یک جدول هش که شامل رئوس و برچسب‌های آن‌هاست خروجی می‌دهد.

ابتدا مبدأ انتشار را با آلفاها مقداردهی می‌کند و رئوس دیده‌شده را تهی در نظر می‌گیرد. به هر رأس نیز ابتدا مقدار 0 را

[1] Github Repository

به‌عنوان مقدار پیش فرض برچسب می‌دهد. یک شمارنده با نام add در نظر می‌گیرد.

در جدول هش، برچسب آلفاها را برابر خودشان قرار می‌دهد. یک حلقه با شرط رسیدن شمارنده‌ی add به مقدار λ در نظر می گیرد. هر دور به add یک واحد اضافه می‌شود و به‌طور موازی رئوس داخل لیست مبداها، وارد پروسه‌ی انتشار می‌شوند. اگر رأس مبدا u مجاوری نداشته باشد، از لیست مبدأ حذف شده و به لیست دیده‌شده‌ها اضافه می‌شود. اگر داشته باشد به ازای هر مجاور v آن، اگر تا به حال دیده‌نشده باشد یا به عبارتی برچسب ۰ داشته باشد، در صورت بیشتر بودن احتمال یال uv از یک مقدار رندوم بین ۰ و ۱، v به لیست مبداها اضافه می‌شود، برچسب u را به‌عنوان برچسب خود می‌پذیرد (این قسمت در شبه کد اشتباه ذکر شده بود) و مقدار add برابر 0 قرار می گیرد. اگر u دیگر رأس برچسب‌نخورده یا همان دیده‌نشده نداشته باشد، از لیست مبدأها حذف می‌شود و به لیست دیده‌شده‌ها اضافه می‌شود.

۲-۶-۱- پیاده‌سازی

پیاده‌سازی این الگوریتم در قالب دو تابع در فایل label_propagation در مخرن گیت‌هاب پروژه انجام شده است.

در تابع label_propagator، ساختار گراف، آلفاها، مقدار آستانه، لیستی از اعداد تصادفی برای استفاده در انتشار احتمالی به همراه سه پارامتر برای تقسیم‌بندی لیست مبدأها:

- آستانه‌ی سایز لیست مبدأها: اگر از یک مقدار مشخصی کمتر بود تعداد زیرمجموعه‌ها برابر حاصل تقسیم تعداد مبدأها بر پارامتر اندازه‌ی پیش‌فرض زیرمجموعه، است. و در غیر این صورت به اندازه پارامتر تعداد پیش‌فرض زیرمجموعه‌ها، زیرمجموعه داریم.
- تعداد پیش‌فرض زیر مجموعه‌ها: این تعداد درصورت بیشتر بودن تعداد مبدأها از آستانه‌ی سایز، به این صورت که ۱۰۰ تا ۱۰۰ تا نمونه‌برداری کند مبدأها را زیرمجموعه می‌کند. به طور مثال مبدأ ۱ با ۱۰۱ با ۲۰۱ با هم در یک زیرمجموعه می‌افتند.
- اندازه‌ی پیش‌فرض زیرمجموعه‌ها: در صورتی که تعداد مبدأها کمتر از آستانه باشد. هر زیر مجموعه، این تعداد مبدأ خواهد داشت.

ضمن مقداردهی‌های گفته‌شده بر طبق شبه کد الگوریتم، یک شمارنده‌ی rand_it برای استفاده از لیست اعداد تصادفی مقداردهی اولیه می‌شود. یک سمافور نیز برای کنترل نخ‌ها و جلوگیری از مشکل استفاده‌ی موازی از منابع یکسان نیز مقداردهی می‌شود.

حال بر طبق الگوریتم، حلقه شروع می‌شود. بر اساس آنچه در توضیح پارامترها گفته شد زیرمجموعه‌ها تشکیل می‌شوند و تابع propagator با ورودی هر زیرمجموعه و سمافور به هر نخ تخصیص می‌یابد. سپس نخ‌ها شروع به اجرا می‌کنند و پس از خاتمه‌ی انتشار موازی در این گام حلقه به گام بعدی می‌رود. تا در نهایت جدول هش به‌عنوان نتیجه از این الگوریتم بازگردانی شود.

تابع propagator نیز با دو ورودی زیرمجموعه‌ای از مبدأها و سمافور و بدون خروجی به شکل زیر کار می‌کند:

هر مبدأ u مشابه شبه کد در صورت مجاور نداشتن از لیست مبدأها حذف و به لیست دیده‌شده‌ها اضافه می‌شود. در صورت مجاور داشتن تعداد مجاورین برچسب‌نخورده را با یک شمارنده نگه می‌داریم که در صورت 0 ماندن، u را از مبدأها حذف و به

دیده‌شده‌ها اضافه کنیم. به ازای هر مجاور u مثل رأس v، شمارنده‌ی برچسب‌نخورده‌ها را یک واحد اضافه کرده، احتمال انتشار را بر اساس فرمول محاسبه می‌کنیم و اگر از عدد تصادفی rand_itام بیشتر باشد اقدام به انتشار می‌کنیم. یعنی ضمن قفل کردن سمافور، v را برچسب زده و به لیست مبدأها اضافه می‌کنیم تا در دورهای بعدی انتشار دهد. add را صفر می‌کنیم و شمارنده را یک واحد کم می‌کنیم. سپس سمافور را آزاد می‌کنیم. برای هر مجاور rand_it یک واحد افزوده می‌شود.

نکته: شرط 0 بودن برچسب v داخل محدوده‌ی قفل سمافور باید تکرار شود زیرا ممکن است یک مبدأ به‌صورت موازی v را اضافه کرده باشد و نخ فعلی صرفاً پشت قفل، منتظر افزودنش مانده باشد.

یکی از مهمترین مسائل، بحث پیچیدگی الگوریتم ارائه شده است.

برای مرتبه‌ی زمانی این رویکرد، یک حد بالا با ارائه ۵ لم و اثبات آن‌ها ارائه شده است.

۱. لم اول: امید ریاضی تعداد امتحان لازم برای رسیدن به حداقل یک موفقیت در امتحان موازی n رویداد مستقل برنولی برابر است با:

$$E = \frac{1}{\min p_i \, , i \ \in [1, \mathrm{n}], i \in \mathrm{Z}}$$

طبق اثبات مقاله برای هر p به فرمول زیر می‌رسیم:

$$E = \frac{1}{p}$$

سپس به ازای pهای موازی به این نتیجه می‌رسیم که در بدترین حالت کمترین p حادث می‌شود و لم اثبات می‌شود.

۲. لم دوم: در گراف همبند با قطر d با شروع از هر رأس دلخواه می‌توان با d پیمایش همسایه پشت سر هم به رأس دلخواه دیگری در گراف رسید.

اثبات این لم نیز بسیار ساده است. طبق تعریف قطر طول بلندترین کوتاه‌ترین مسیر بین دو رأس در گراف است. بنابراین فاصله‌ی هر دو رإس دلخواه نهایتا d است.

۳. لم سوم: در هر گراف n راسی G با قطر d و کمترین مقدار ممکن برای حداکثر درجه‌ی رئوس:

بیشترین درجه‌ی بهینه برابر : $\left\lceil n^{\frac{1}{\lfloor d/2 \rfloor}} \right\rceil + 1$

G حداقل $n - \left((k-1)^{\left\lfloor \frac{d}{2} \right\rfloor} - 1\right) \times K$ راس با درجه حداقل k دارد که $k = \left\lceil n^{\frac{1}{\lfloor d/2 \rfloor}} \right\rceil + 1$

۴. برای هر شبکه‌ی اجتماعی داریم: $|V|\log|V| = O(|E|)$

اثبات این لم با استفاده از ضریب خوشه‌بندی انجام می‌شود. ضریب خوشه‌بندی برابر است با تعداد مثلث های حول رئوس گراف به تعداد مسیرهای به طول ۲ یا همان سه‌تایی‌های به هم متصل غیرمثلث است. مدل Barabasi-Albert یک مدل است که گراف‌های تصادفی با توزیع درجه‌ی power-law تولید می‌کند. در شبکه‌های واقعی معمولاً ضریب خوشه‌بندی

از مقدار به‌دست‌آمده برای گراف‌های مدل BA بسیار بیشتر است. بنابراین از مقدار ضریب خوشه‌بندی آن‌ها به‌عنوان یک حد پایین استفاده می‌کنیم و برابر است با:

$$CC(G) = \frac{(\ln |V|)^2}{|V|}$$

به طور تقریبی، میانگین ضریب خوشه‌بندی محلی با ضریب خوشه‌بندی سراسری در شبکه برابر است. بنابراین $CC(u) \approx CC(G)$ و به عبارتی:

$$\frac{2|\{(u,v) \in E | v \in N(u)\}|}{|N(u)| * (|N(u)| - 1)} = \frac{(\ln |N(u)|)^2}{|N(u)|}$$

و در نهایت:

$$|\{(u,v) \in E | v \in N(u)\}| = \frac{|N(u)|(\ln |N(u)|)^2}{2}$$

با توجه به این که شبکه ما باید حدوداً به این اندازه یال داشته باشد تا طبق power-law توزیع درجه داشته باشد. تعداد یال‌های شبکه حداقل برابر فرمول فوق است یعنی:

$$|E| \geq c|V|(\ln |V|)^2$$

پس به عبارتی: $|V|\log|V| = O(|E|)$

۵. زمان مورد انتظار اجرای الگوریتم در بدترین حالت برابر $O(|E|)$ است.

پیچیدگی زمانی الگوریتم ارائه شده برای یافتن آلفاها، مرتب‌سازی با مرتبه‌ی $|V|\log|V|$ است و همچنین اشتراک‌گیری که همانطور که در بخش پیاده‌سازی این الگوریتم گفته شد با مرتبه‌ی $|V|\log|V|$ پیاده شده یعنی با مرتب‌سازی و جست‌وجوی خطی. پس در مجموع $O(|E|)$ می‌شود.

برچسب‌گذاری تنها در بزرگترین مؤلفه‌ی همبندی در عمل انجام می‌شود و بدترین زمان اجرا برای اجرای متوالی تا پوشش کامل گراف و برچسب‌گذاری کامل است. با این حساب اگر در هر مرحله مثل i، iامین محله مورد پیمایش پشت سر هم قرار گیرد، تعداد یال‌های بین 1-iامین و iامین محله در بدترین حالت برای هر یال uw برابر امید ریاضی محاسبه شده در لم اول ضرب در تعداد یال‌های آن هاست که در بدترین حالت برابر کل تعداد یال‌هاست. بنابراین زمان اجرا با توجه به مرتبه‌ی ثابت امید ریاضی و حداکثر طول مسیر بین رأس‌ها که قطر است، برابر ضریبی از $|E|$ است. بنابراین پیچیدگی زمانی رویکرد برابر $O(|E|)$ است.

۲-۷- آزمایشات

در بخش آزمایشات لازم است علاوه بر معرفی داده‌های مورد استفاده در مقاله‌ی اصلی، معیارهای ارزیابی جهت سنجش

کیفیت عملکرد الگوریتم را نیز بررسی کنیم.

۲-۷-۱- دادگان

خروجی الگوریتم برای ۱۴ مجموعه‌داده ارزیابی شده است. اولین مجموعه‌داده، شبکه‌ی اعضای یک باشگاه کاراته دانشگاهی است که ۳۴ نود و ۷۸ یال دارد. در اصل این مجموعه‌داده، مربوط به یک گراف بدون جهت و بدون وزن است که ما آن را با قرار دادن دو یال به جای هر یال و اعطای وزن ۱ به هر یال، به یک گراف جهت‌دار و وزن‌دار تبدیل کردیم. شبکه‌ی تعاملات فیسبوک، ۴ داده از پروژه‌ی تحلیل شبکه‌های اجتماعی استنفورد، شبکه‌ی مؤلفان آثار علمی، گراف شبکه‌ی اجتماعی یوتیوب[۱] شبکه‌ای از صفحات ویکیپدیا[۲] شبکه‌ای از ایمیل‌های یک مرکز علمی در اروپا و ۸ مجموعه‌داده‌ی دیگر که هر ۸تای آن‌ها توسط ToreOpsahl منتشر شده‌اند، دیگر داده‌هایی هستند که در این مقاله استفاده شده‌اند.

۲-۷-۲- ارزیابی

به جهت نشان دادن کارایی الگوریتم، برای هر مجموعه‌داده‌ای که امکان مقایسه وجود داشته باشد، مقاله‌ی اصلی عملکرد خودش را با الگوریتم MCL مقایسه کرده است. زیرا MCL که بر اساس الگوریتم خوشه‌بندی کارکوف کار می‌کند، در بسیاری از زمینه‌ها مورد استفاده قرار می‌گیرد. علاوه بر این، شبیه‌ترین روش به الگوریتم طراحی‌شده در این مقاله نیز هست. این مقایسه هم برای داده‌های دارای برچسب درست انجام شده و هم برای داده‌هایی که برچسب نداشتند. علاوه بر مقایسه در نتیجه و دقت، مقایسه در زمان لازم برای انجام الگوریتم نیز صورت پذیرفته است که در بخش نتایج، به آن خواهیم پرداخت.

برای مجموعه‌داده‌هایی که برچسب واقعی آن‌ها در دسترس بوده است (مانند شبکه ایمیل‌ها و صفحات ویکیپدیا)، نویسنده از دو معیار FPR و FNR استفاده کرده‌است. به این صورت که به ازای هر کلاس (برچسب) این دو معیار رو محاسبه کرده و برای کل الگوریتم، از آن‌ها میانگین گرفته است. این دو معیار طبق روابط زیر تعریف می‌شوند.

$$FPR = \frac{FP}{FP + TN}$$

$$FNR = \frac{FN}{FN + TP}$$

اما برای آن دسته از داده‌هایی که جواب درست مسئله برایشان نامشخص بود از معیاری به اسم رسانایی (conductance) استفاده شده که طبق رابطه‌ی زیر تعریف می‌شود.

$$\phi = \frac{c_s}{c_s + 2m_s}$$

در این رابطه c_s تعداد یال‌های مرزی یک اجتماع[۳] و m_s تعداد یال‌های درون یک اجتماع است. به گفته‌ی مقاله هر مقدار رسانایی کمتر باشد، الگوریتم بهتر عمل کرده‌است.

---

[۱] Youtube

[۲] Wikipedia

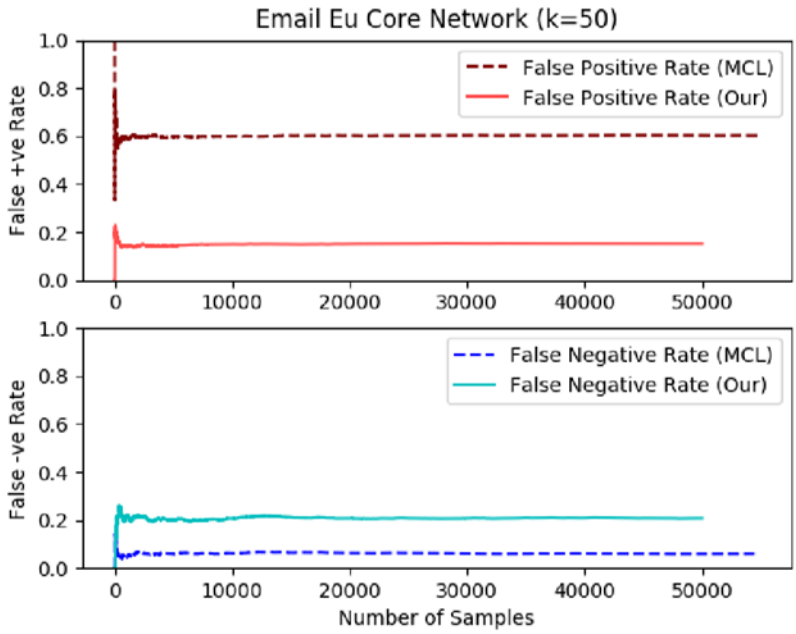
[۳] مقاله اصلی در اینجا از واژه «خوشه» استفاده کرده‌است.

۲-۸- نتایج

مقاله‌ی اصلی نتایج را در سه بخش ارائه می‌دهد. در بخش اول به ارائه و تحلیل نتایج اجرای الگوریتم روی داده‌هایی است که جواب درست مسئله را نیز دارند. بخش دوم به داده‌هایی اختصاص دارد که نویسنده، جواب درست مسئله را به ازای آن‌ها نداشته است. در بخش سوم نیز زمان اجرای الگوریتم طراحی‌شده در این مقاله و MCL را مقایسه کرده است.

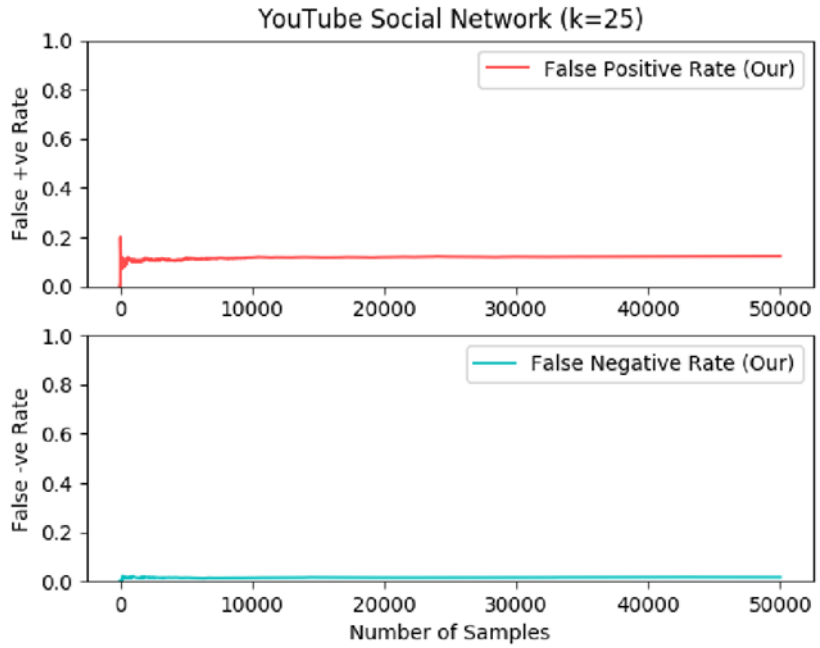
۲-۸-۱- نتایج همراه با جواب درست مسئله

از بین ۴ مجموعه‌داده‌ای که مقاله به ارائه و تحلیل نتایج آن‌ها پرداخته است، ما صرفاً نتایج اجرای الگوریتم بر روی داده‌ی شبکه‌ی ایمیل‌های مرکز علمی اروپا و شبکه‌ی اجتماعی یوتیوب را آورده‌ایم. چون تعداد یال‌های بعضی گراف‌ها بسیار زیاد است، ارزیابی همه آن‌ها ممکن نیست؛ بنابراین به شکل تصادفی از مجموعه‌ی یال‌ها نمونه‌برداری شده‌است. با افزایش اندازه نمونه‌ها، تا حدی پیش می‌رویم که اندازه FPR و FNR به مقدار واقعی همگرا شوند.



در شکل بالا که عملکرد دو الگوریتم را برای مجموعه‌داده‌ی ایمیل، مقایسه شده‌است، همانطور که می‌بینیم در معیار FPR الگوریتم طرح شده در این مقاله، بسیار بهتر از MCL عمل کرده‌است، اما در FNR ضعیف‌تر ظاهر شده است. اگر ماتریس درهم‌ریختگی را در نظر بگیریم، متوجه می‌شویم برای این مجموعه داده، الگوریتم روی کلاس T متعصب (bias) شده‌است. بنابراین نسبت به کلاس N عملکرد جالب توجهی نداشته است.

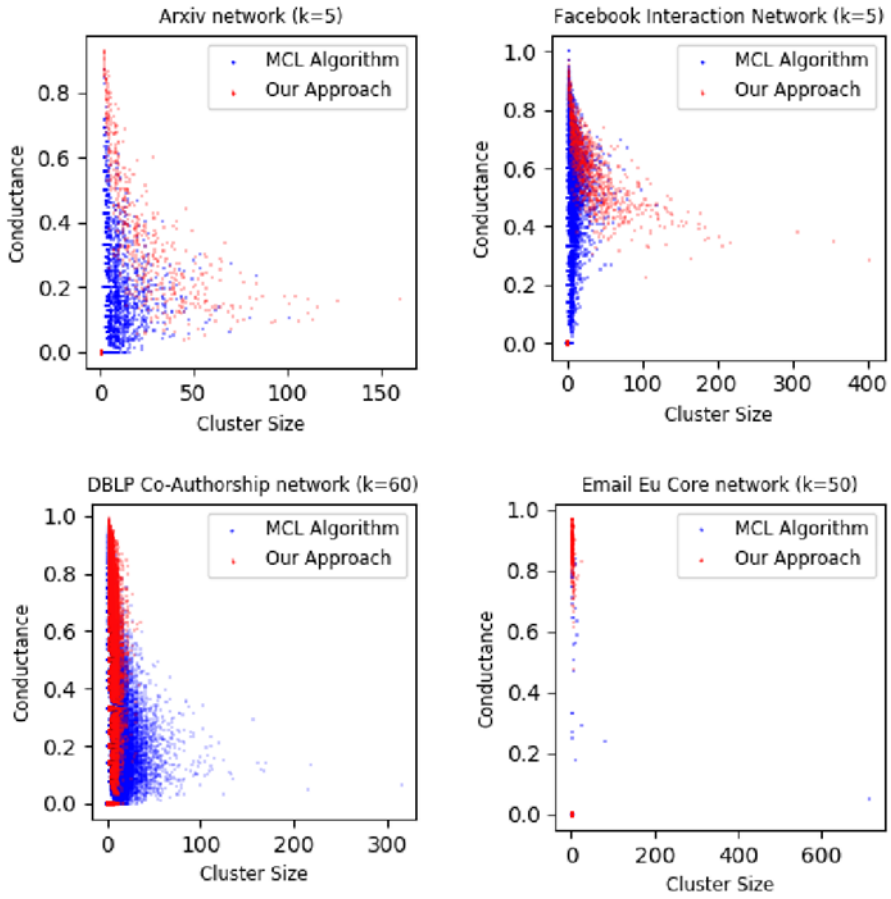
اما نتایج الگوریتم روی مجموعه‌داده‌ی یوتیوب، خارق‌العاده است. زیرا پردازش مجموعه داده‌ای به بزرگی یوتیوب، به شدت سخت و چالش‌برانگیز است و کمتر الگوریتمی، عملکردش را در مواجهه با این مجموعه داده محک می‌زند. همانطور که در شکل زیر پیداست، الگوریتم این مقاله، FPR و FNR را همزمان کاهش داده‌است. علاوه بر این، شکل زیر نشان از مقیاس‌پذیری بالای الگوریتم دارد، به شکلی که الگوریتم MCL اصلا نتوانسته حجم زیاد اطلاعات در مجموعه‌داده‌ی یوتیوب را پردازش کند.



۲-۸-۲- نتایج بدون داشتن جواب درست مسئله

همانطور که پیشتر گفته شد، برای ارزیابی و تحلیل داده‌هایی که جواب درست را برای آن‌ها در اختیار نداریم، از معیار رسانایی استفاده می‌کنیم. درست است که در مجموعه‌داده‌هایی که جواب درست را در اختیار داریم، عمدتاً عملکرد الگوریتم این مقاله بهتر از MCL عمل کرده است، اما در داده‌هایی بدون جواب درست، به‌صورت کلی می‌توان گفت، MCL عمکلرد بهتری را به نمایش گذاشته است. البته به گفته نویسنده مقاله، این دلیلی برای ضعف الگوریتم ما نیست، بلکه معیار رسانایی به تنهایی برای ارزیابی چنین داده‌های کافی نیست.

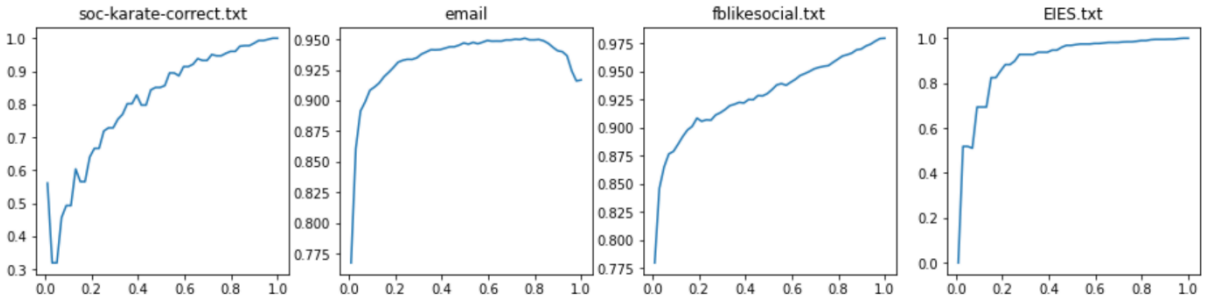
تصویر زیر، مقادیر مختلف این معیار را به ازای kها و مجموعه‌داده‌های متفاوت نشان می‌دهد. با توجه به این تصویر و نتایج استفاده از رسانایی برای هر ۱۴ مجموعه داده، می‌توانیم استدلال کنیم ، برای مقادیر بزرگ k، عملکرد دو الگوریتم نزدیک به یکدیگر هستند. اما برای مقادیر کوچک k، الگوریتم این مقاله می‌تواند با رسانایی کمتر (عملکرد بهتر) تعداد زیادتری از اجتماعات بزرگ را شناسایی کند.



به دلیل اینکه ما دقیقا همان نسخه از داده‌هایی که نویسنده استفاده کرده‌است را در اختیار نداشتیم، نتوانستیم داده‌هایی را پیدا کنیم که برچسب درست را داشته باشند، بنابراین تنها ملاک ارزیابی ما استفاده از معیار رسانایی بود. کدی که برای محاسبه معیار رسانایی توسعه دادیم را در این پیوند از مخزن گیت‌هاب می‌توانید مشاهده کنید.

با مراجعه به نمودارهای بالا، یکی از مجموعه‌داده‌ها را انتخاب و به ازای همان kای که در مقاله ذکر شده بود، مقدار رسانایی را محاسبه کردیم. نتیجه‌ی مقایسه‌ی نتیجه‌ی ما با نتایج موجود در مقاله، اعداد مشابهی را نشان می‌دادند. این مشابهت دال بر پیاده‌سازی صحیح الگوریتم‌های مقاله است.

برای مشاهده‌ی تأثیر k در الگوریتم، از ۴ داده‌ی باشگاه کاراته، ایمیل‌ها، تعاملات فیسبوک و شبکه‌ی محققان اطلاعات الکترونیکی استفاده کردیم تا نمودار زیر را رسم کنیم.



تصویر بالا معیار رسانایی الگوریتم را به ازای ۵۰ مقدار متفاوت k که به فاصله خطی از ۰/۰۱ تا ۱ پراکنده شده‌اند، نشان

می‌دهد که برای مجموعه‌داده‌های مختلفی محاسبه شده‌است.

۳,۲. مقایسه زمان اجرا

همانطور که پیشتر دیدیم، الگوریتم MCL برای برخی مجموعه‌داده‌ها نظیر یوتیوب، اصلا مقیاس‌پذیر نیست. اما خوب است به صورت واضح‌تر، تفاوت این دو الگوریتم را در زمان اجرا نیز مشاهده کنیم. با استناد به جدول زیر و متن مقاله، می‌توانیم نتیجه بگیریم پیچیدگی زمانی الگوریتم نسبت به تعداد یال‌ها، به شکل خطی است.

Dataset	No. of Edges	MCL	Our Approach
EIES network	830	0.009s	0.081s
Fb-like Social	20296	0.924s	0.234s
Email	25571	0.810s	0.190s
Arxiv	95188	1.957s	1.102 s
Fb-like forum	142760	0.949s	0.492s
FB-Interaction	253831	20.643s	4.903s
DBLP	1.05 M	8m 33s	6m 28s
YouTube	5.98 M	Does not scale	1hr 7min
Wikipedia	28.5 M	Does not scale	3hr 4min

۲-۹- محدودیت‌ها و کارهای آینده

یکی از محدودیت‌هایی که نویسنده‌ی مقاله برای آن برشمرده است، سختی و چالش‌برانگیز بودن انتخاب k است. در متن مقاله عنوان شده‌است که بهتر بود انتخاب k را به عهده‌ی کاربر نگذاریم، بلکه با روش‌های هوشمند، مقادیر بهتری از k را انتخاب کنیم. روش‌های هوشمند می‌توانند بر اساس نوع شبکه تعیین شوند. مثلا در گرافی که ارتباط بین مؤلفان مختلف مقالات علمی را مدل کرده بود، هر نود با تعداد اندکی از نودهای دیگر در ارتباط بوده است. پس بهتر است در این مجموعه‌داده مقادیر بزرگتری برای k انتخاب کنیم که بتوانیم اجتماعات بیشتری را پیدا کنیم.

یکی دیگر از محدودیت‌هایی که مقاله هیچ صحبتی از آن نکرد، عدم توجه به ادغام اجتماعات است. به این معنی که در الگوریتم alph_detection هیچگاه کنترل نشد که چه نودهایی انتخاب می‌شوند. چه بسا نودهایی به عنوان مؤلفه‌های اصلی انتخاب شوند که مجاوران مشترک بسیار زیادی داشته باشند و در نهایت منجر به تشکیل یک rich club شوند.

یکی دیگر از محدودیت‌های مقاله استفاده از معیار رسانایی است. در متن مقاله اشاره شد که این معیار به تنهایی، معیار کاملی برای بررسی عملکرد الگوریتم‌ها نیست، اما هیچ جایگزینی برای آن معرفی نشد. معیاری که ما پیشنهاد می‌کنیم استفاده از اینرسی است که طبق فرمول پایین تعریف می‌شود.

$$inertia = \sum_{s \in S} \sum_{v_i \in s} \frac{1}{d(v_{alpha}, v_i)}$$

که در رابطه بالا، $d(v_{alpha}, v_i)$ هزینه کوتاه‌ترین مسیر از نود آلفا یک خوشه به نود v_i موجود در آن خوشه است. البته معیار اینرسی فقط برای شبکه‌هایی خاصی جوابگو خواهد بود.

۱۰-۲- نحوه‌ی اجرا

مانند هر پروژه پایتونیک دیگری نیاز است به‌عنوان اولین گام، یک محیط مجازی بسازیم و پس از فعال‌سازی آن، با صدور دستور زیر در خط فرمان، نیازمندی‌های پروژه را درون محیط مجازی نصب کنیم.

pip install -r requirements.txt

حال برای اجرای کد، در ابتدا باید پوشه‌ای به اسم *output* درون پوشه‌ی اصلی بسازیم تا گراف‌هایی که برچسب‌گذاری شده‌اند در این پوشه ذخیره شوند. سپس لازم است فایل *run.py* را به وسیله‌ی پایتون اجرا کنیم. حین اجرا، سؤالات متداولی از ما پرسیده می‌شود. در اولین سوال باید مشخص کنیم قصد داریم کدام یک از گراف‌های موجود در فایل *datasets* را برچسب‌گذاری کنیم. به‌عنوان مقدار دومی که به فایل در حال اجرا می‌دهیم، باید k را مشخص کنیم، در نهایت با سومین ورودی، اسم فایلی را به کد می‌دهیم که می‌خواهیم گراف برچسب‌گذاری‌شده را در آن ذخیره کنیم.

۳- مقاله دوم (A Novel Approach for Community Detection)

۳-۱- چکیده

نویسنده در این مقاله با استفاده از روش انتشار برچسب، سعی در ارائه‌ی یک الگوریتم برای تشخیص اجتماع پویایی[۱] که با هم هم‌پوشانی[۲] داشته باشند. در این روش ابتدا به هر نود در یک شبکه، برچسبی معادل با شناسه آن نود داده می‌شود و در ادامه می‌تواند برچسب‌های مختلفی را از همسایگانی که تأثیرگذاری بیشتری دارند نیز دریافت کند.

۳-۲- مقدمه

در ابتدا هدف تحلیل شبکه‌های اجتماعی این طور بیان شده است که ما با تحلیل روابط بین افراد و گروه‌ها سعی در شناخت فراز و فرود آن‌ها و پی‌بردن به طبعات آن‌ها را داریم. در بخش مقدمه، نویسنده یک اجتماع در شبکه‌های اجتماعی را دسته‌ای از نودها معرفی می‌کند که خصوصیات مشابه زیادی دارند که به‌صورت متداول با هم در ارتباط هستند. به عبارت بهتر دسته‌ای از نودها درون یک اجتماع قرار می‌گیرند که یال‌های درونی زیاد و یال‌های بیرونی کمتری داشته باشند. نویسنده اجتماعات را به دو دسته‌ی کلی تقسیم کرده است. در دسته‌ی اول اجتماعاتی قرار دارند که از یکدیگر مجزا و دسته‌ی دوم متعلق به اجتماعاتی است که با یکدیگر هم‌پوشانی دارند.

الگوریتم‌های مختلفی برای تشخیص اجتماعات در یک شبکه اجتماعی وجود دارد که نویسنده به انتشار برچسب، الگوریتم ژنتیک، بخش‌بندی گراف و خوشه‌بندی اشاره کرده است؛ همچنین ذکر شده که روش انتشار برچسب، بیش از سایر روش‌ها مورد توجه واقع شده است.

---

[۱] Dynamic

[۲] Overlapping

۳-۳- کارهای مرتبط

در ۱۰ سال گذشته[1]الگوریتم‌های مختلفی برای تشخیص اجتماع طراحی شده‌اند که اکثر آن‌ها به دلیل پیچیدگی زیاد زمان و حافظه، کارایی مناسبی ندارند و در عمل برای استفاده در کاربردهای روزمره قابل اتکا نیستند. اولین مقاله‌ی مرتبطی که نویسنده به آن اشاره می‌کند، مقاله‌ای است که برای اولین بار، الگوریتم انتشار برچسب در آن مطرح شد. این الگوریتم تنها با استفاده از ساختار شبکه، اجتماعات هم‌پوشا و مجزا را در مرتبه خطی نسبت به تعداد یال، تشخیص می‌دهد. البته برای این الگوریتم دو عیب نیز ذکر شده است. ایراد اول آن است که در ابتدا به همه نودها به‌صورت تصادفی برچسب می‌زند؛ این کار باعث ایجاد کارکرد غیرقابل‌پیش‌بینی خواهد شد. ایراد دوم در انتخاب روش حریصانه[2]برای انتصاب برچسب و به‌روزرسانی نودهاست که به تشکیل اجتماعات بسیار بزرگ و نادیده‌گرفته‌شدن اجتماعات کوچک می‌انجامد. دومین مقاله‌ای که مورد بررسی قرار می‌گیرد، مقاله‌ای است که برای بار اول، تشخیص اجتماعات هم‌پوشا را مطرح کرد. به گفته‌ی نویسنده، الگوریتم COPRA، در هر بار به همه‌ی نودها اجازه می‌دهد به شکل همزمان، ضرایب خود را با توجه به میانگین ضرایب همسایه‌ها، به‌روز کنند. این الگوریتم با معرفی متغیر v تحت عنوان بیشینه‌ی اجتماعاتی که یک نود می‌تواند به آن‌ها تعلق داشته باشد، به برخی نودها اجازه می‌دهد عضو تعداد زیادی اجتماع باشند، در حالی که برخی نودها به تعداد اجتماع کمتری تعلق دارند؛ که البته استفاده از v معایبی هم دارد. یکی از ایرادات واردشده به این الگوریتم، آن است که انتخاب v مناسب، دشوار است. زیرا باید مقداری داشته باشد که هر دو نوع نود را بتواند پشتیبانی کند.

۴-۳- روش طراحی شده در این مقاله

روش معرفی‌شده در این مقاله، با استفاده از همسایه‌های مشترک[3]و شباهت همسایه‌های مشترک[4]که بر اساس شاخص جاکارد[5]است، اجتماعات را تشخیص می‌دهد. الگوریتم برای اینکه بتواند پویایی شبکه را نیز در نظر بگیرد، میزان تغییر درجه‌ی نودها را با توجه به زمان، در الگوریتم اثر داده است. علاوه بر این، الگوریتم از نوع تکراری[6]است و در هر تکرار، تغییر زمان نیز مؤثر خواهد بود.

در این مقاله تأکید بر استفاده از برچسب یکتا برای هر نود است؛ یعنی در ابتدا به هر نود، برچسبی معادل با شناسه آن نود می‌زند و حین اجرا، اگر نیاز بود، نودها به برچسب‌های دیگری نیز منتسب می‌شوند. بنابراین در ابتدا هر نود، برچسب مشخصی می‌گیرد. در ادامه به صورت تصادفی نودی را از شبکه انتخاب می‌کند که عضو هیچ اجتماعی نباشد و درجه‌ی آن بزرگتر یا مساوی ۳ باشد. سپس همسایه‌ای از این نود که نام آن را v_i گذاشته است، به شکلی انتخاب می‌شود که بیشترین درجه را داشته و آزاد باشد و اسم آن را v_j می‌گذارد. سپس الگوریتم، همسایه‌های مشترک نود i ام و j ام را پیدا می‌کند. $L_t(v_i, v_j)$ نزدیکی همسایه‌های مشترک نود i ام و j ام را به دست می‌دهد. در هر تکرار، $D_t(v_c)$ را به ازای همه‌ی v_c های عضو همسایه‌های مشترک نود i ام و j ام، محاسبه می‌کنیم. سپس با استفاده از این دو فرمول، معیار شباهت همسایه‌های مشترک را چنین به

[1] مقاله در سال ۲۰۱۹ منتشر شده است؛ بنابراین بهتر است در زمان حال بگویم :«در ۱۲ سال گذشته ...»

[2] Greedy
[3] Common Neighbors
[4] Common Neighbor Similarity
[5] Jaccard Index
[6] Itterative

دست می‌آوریم.

$$CNS_t(v_i, v_j) = L_t(v_i, v_j). \sum_{v_m \in CN_t(v_i, v_j)} D_t(v_c).$$

اگر دو نود در شبکه، مشترکاً دارای بیشترین CNS باشند، آن‌ها را به یک گروه منتقل می‌کنیم، اما اگر CNS نودی بیشتری از نود دیگری باشد، نود با CSN کمتر را درون گروه نود با CNS بیشتر قرار می‌دهیم. اگر نودی بیش از یک همسایه داشت که هر چند تای آن‌ها، مشترکاً بیشترین CNS موجود را داشتند، برچسب همه‌ی آن همسایه‌ها به نود مورد بررسی، منتسب می‌شود. در این حالت است که یک نود می‌تواند عضو بیش از یک اجتماع باشد.

در شرایطی ممکن است شبکه با اجتماعات کوچک ولی پر تعداد تقسیم شود. در این حالت الگوریتم با ادغام اجتماعات کوچک در اجتماعات بزرگتر، یک اجتماع واحد بزرگ می‌سازد. فرایند ادغام تا زمانی ادامه خواهد داشت که یک نود برچسبی را بگیرد که اکثر نودهای همسایه‌اش دارند.

۳-۵- نتیجه‌گیری و کارهای آینده

به دلیل کاربرد زیاد شبکه‌های اجتماعی، تحلیل آن‌ها به مسئله‌ی مهمی تبدیل شده است. در این مقاله، روشی معرفی شد که با در نظر گرفتن به‌روزشدن درجه‌ی نودها با گذشت زمان، اجتماعات مختلف از نوع هم‌پوشا و مجزا را تشخیص دهد. البته این الگوریتم فضای زیادی را اشغال می‌کند که به عنوان یکی از حوزه‌هایی که در آینده می‌توان در آن به تحقیق و نوآوری پرداخت، کاهش دادن هزینه‌ی فضایی الگوریتم است. علاوه بر بهبود پیچیدگی فضایی، می‌توان این الگوریتم را به شبکه‌های وزن‌دار و دوبخشی نیز تعمیم داد.