فصل دهم

Mining Social-Network Graphs

فرشاد اصغرزاده – امیر شکری
استاد رحمانی منش – درس داده کاوی – ترم ۹۸ - ۹۹ - دانشگاه سمنان

به نام خدا

در این بخش قصد داریم خلاصه و موضوعاتی که از این فصل از کتاب که تحت عنوان Mining Social Network Graphs می باشد را داشته باشیم.

در این کار گروهی ابتدا تمامی تصاویر کتاب جدا شده است و در نهایت فصل دهم به طور کامل توسط گروه ما ترجمه شده است. حالا در این فایل خلاصه ای به صورت گزارش ارائه شده است.

گروه ما :

- امیر شکری9811920009
- فرشاد اصغرزاده همپا 9811920004

تمامی کارهای مربوط به این کار گروهی که به عنوان تکلیف در داده کاوی در نظر گرفته شده در گیت هاب انجام شده است زیرا در دوران فعلی که بیماری کرونا باعث ایجاد طرح فاصله گذاری اجتماعی شده است کارهای گروهی باید از راه دور انجام شود.

لینک گیت هاب:

https://github.com/semnan-university-ai/Mining-Social-Network-Graphs

ایمیل اعضای گروه :

Amirsh.nll@gmail.com
Farshad_asgharzade@hotmail.com

در آخر از دکتر رحمانی منش بابت ارائه ی این تکلیف تشکر میکنیم زیرا با توجه به این فعالیت گروه ما این موضوع را به طور کامل بررسی کرد و تجربه ی انجام یک کار دانشگاهی به صورت تیمی را پیدا کرده است.
استفاده از گیت هاب خلاقیت فکری دونفره ی ما بوده است که امیدواریم مورد قبول شما واقع شود.
ترجمه ی این بخش توسط تیم ما انجام شده است و استفاده از آن در گیت هاب و موارد ... از لحاظ ما مانعی ندارد و به صورت open source در اختیاز عزیزان خواهد بود.

ارادتمند،
تابستان 1399
امیر شکری، فرشاد اصغر زاده همپا

۱۰.۱ شبکه های اجتماعی به صورت گراف

بحث خود را در مورد شبکه های اجتماعی با معرفی یک مدل گرافی آغاز می کنیم.

هر نمودار گرافی مناسب به نمایش یک اجتماع در شبکه های اجتماعی نیست.

بنابراین، ما درباره ی ایده ی اصل محلیت که جز ویژگی های اصلی شبکه های اجتماعی است با کمک نودها و یال ها در گراف ها صحبت می کنیم. با کمک گره ها و یال ها تمایل خوشه بندی در شبکه ها بررسی می کنیم.

در این بخش همچنین برخی از انواع مختلف شبکه های اجتماعی که در عمل مورد استفاده قرار می گیرند را بررسی می کنیم.

10.1.1 شبکه اجتماعی چیست؟

وقتی به یک مفهوم شبکه اجتماعی فکر می کنیم ، به فیس بوک ، توییتر ، Google+ یا وب سایت دیگری فکر می کنیم که "شبکه اجتماعی" نامیده می شود و در واقع این شبکه ها نماینده ای از مفهوم شبکه های اجتماعی هستند.

ویژگی های اساسی یک شبکه اجتماعی عبارتند از:

1. مجموعه ای از اشخاص در شبکه های اجتماعی در شبکه وجود دارند که به طور معمول این موجودات مردم هستند اما می توانند چیزهای دیگری نیز باشند؛ در بخش 10.1.3 به مثالهای بیشتری در این مورد می پردازیم.
2. حداقل یک رابطه بین موجودیت های موجود (کاربران) در شبکه های اجتماعی وجود دارد. به رابطه ی بین موجودیت ها در فیس بوک Relationship می گویند. ارتباط ها یا وجود داشته یا ندارد پس دو نفر یا دوست هستند یا نیستند؛

   گرچه در گراف های شبکه های اجتماعی رابطه ها دارای یک درجه هستند.

   این درجه ها می تواند مقدار گسسته داشته باشد. به عنوان مثال در گوگل پلاس این درجه ها با عنوان های دوستان، خانواده، آشنایان و ... شناخته می شود.

   این درجه می تواند یک عدد صحیح باشد یا یک عدد کسری باشد که از میزان صحبت بین دونفر بدست می آید.
3. محلیت در این شبکه ها به صورت غیرتصادفی است. این شرط برای نرمال سازی سخت ترین شرط است. اما ارتباطش به گرایش خوشه ها و روابط آنها می پردازد. یعنی اگر موجودیت A به هر دو موجودیت B و C مربوط باشد احتمال رخداد آن از میانگین B و C بدست می آید.

10.1.2 شبکه های اجتماعی به صورت گراف

شبکه های اجتماعی به طور معمول به عنوان گراف هایی مدل سازی می شوند که بعضا از آنها به عنوان یک گراف اجتماعی یاد می کنیم.

در این نمودار گراف گره ها موجودیت ها هستند و یال ها اتصال و ارتباط بین دو موجودیت (کاربر) هستند. اگر در این گراف درجه ای موجود باشد روی یال ها برچسب زده می شود.

غالبا گراف های موجود در شبکه های اجتماعی بدون جهت هستند؛ مثل گراف دوستان در شبکه اجتماعی فیس بوک.

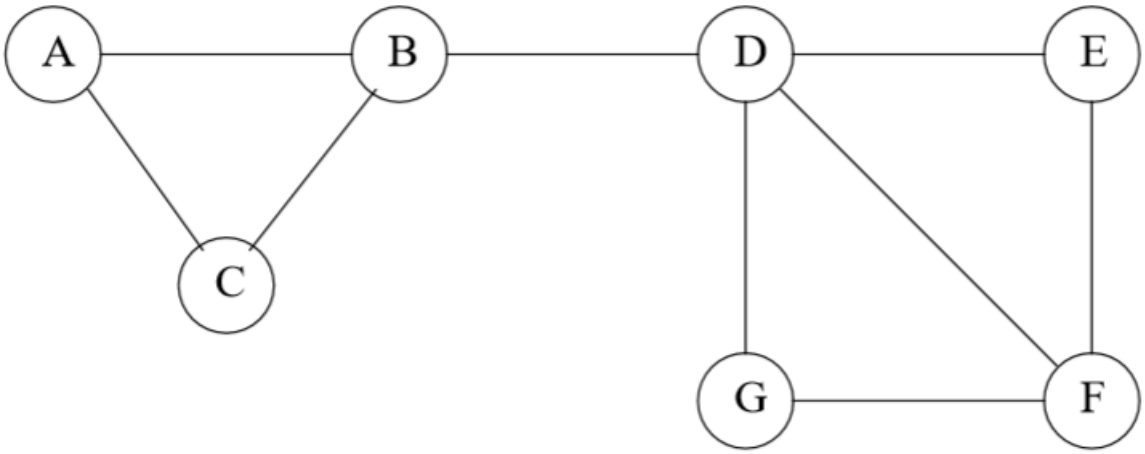
اما می توان گراف های جهت دار هم داشت؛ مانند نمودار فالورهای توییتر یا گوگل پلاس.

مثال 10.1 : شکل 10.1 نمونه ای از یک گراف کوچک شبکه اجتماعی است؛ گره های این گراف از A تا G نامگذاری شده است. رابطه ی موجود در این گراف به عنوان دوست شناخته می شود که یال ها را تشکیل می دهد.

به عنوان مثال موجودیت B با A و C و D دوست می باشد.

آیا این گراف واقعا نشان دهنده ی یک شبکه ی اجتماعی و نمایش روابط آنها است؟

ابتدا توجه داشته باشید که گراف زیر دارای 9 یال می باشد.



شکل 10.1 : نمونه ای از شبکه اجتماعی کوچک

$\binom{7}{2} = 21$. در واقع 21 جفت گره می تواند در این شبکه دارای لبه باشد یا حداکثر یال های موجود در این شبکه 21 باشد.

فرض کنید X و Y و Z نودهای شکل 10.1 هستند که بین x و y و همچنین بین x و z دارای یال می باشد.

احتمال وجود یال بین y و Z چقدر است؟

اگر این نمودار بزرگ باشد این احتمالا به صورت کسری می باشد. یعنی در واقع 9 تقسیم بر 21 که برابر 0.429 می شود.

اما ، از آنجا که نمودار کوچک است ، بین احتمال واقعی و نسبت تعداد لبه ها به تعداد جفت گره ها تفاوت قابل ملاحظه ای وجود دارد.

از آنجایی که می دانیم یال های (X، Y) و (X، Z) وجود دارد ، تنها 7 یال باقی مانده است. این 7 یال می توانند بین هر 19 گره باقی مانده از گره ها قرار بگیرند.

بنابراین احتمال یک لبه بین z و y برابر 7 تقسیم بر 19 که برابر 0.368 می شود است.

حال باید این احتمال را محاسبه کنیم که لبه (Z ،Y) در شکل 10.1 وجود داشته باشد ، با توجه به اینکه لبه ها (X، Y) و (X، Z) وجود دارند. آنچه در واقع باید حساب کنیم ، جفت گره هایی است که می توانند Y و Z باشند ، بدون اینکه نگرانی در مورد کدام گره Y

باشد و کدام Z باشد. اگر A X باشد ، باید Y و Z به ترتیب B و C باشند. از آنجا که لبه (C ،B) وجود دارد ، A یک مثال مثبت (که در آن لبه وجود دارد) است و هیچ نمونه منفی (جایی که لبه وجود ندارد) در آن نیست.

در مواردی که X به جای C، E یا G است نتایج یکسان است.

در هر حالت ، X فقط دو همسایه دارد و لبه بین همسایگان وجود دارد. بنابراین ، ما تاکنون چهار نمونه مثبت و صفر مثال منفی را دیده ایم.

حال ، در نظر بگیرید که X = F. F دارای سه همسایه ، D، E و G است. لبه هایی بین دو سه جفت همسایه وجود دارد ، اما هیچ لبه ای بین G و E وجود ندارد. بنابراین ، ما دو نمونه مثبت دیگر را می بینیم و اولین نمونه منفی خود را می بینیم.

اگر X = B ، دوباره سه همسایه وجود دارد ، اما فقط یک جفت همسایه ، A و C ، یک لبه دارند. بنابراین ، ما دو مثال منفی دیگر ، و یک مثال مثبت ، برای کل هفت مثبت و سه منفی داریم.سرانجام ، وقتی X = D ، چهار همسایه وجود دارد. از شش جفت همسایه ، فقط دو نفر بین آنها لبه دارند.

بنابراین ، تعداد کل نمونه های مثبت 9 و تعداد کل نمونه های منفی 7 است. در شکل 10.1 می بینیم که کسر ما برابر 9 تقسیم بر 16 است که در واقع برابر 0.563 می شود. این کسر خیلی بیشتر از مقدار قابل انتظار ما که 0.368 است می باشد.

در نتیجه شکل 10.1 واقعا اصل locality در شبکه های اجتماعی را نشان می دهد.

10.1.3 انواع شبکه های اجتماعی

نمونه های زیادی از شبکه های اجتماعی وجود دارد که ماهیت دوستان ندارد. در اینجا ، اجازه دهید تعدادی از نمونه های دیگر شبکه های اجتماعی که با اصل locality روابط را نشان می دهند ذکر کنیم.

شبکه های تلفن

در اینجا گره ها شماره تلفن ها را نشان می دهند ، که در واقع افراد هستند. اگر در طی مدت زمان مشخصی مانند ماه گذشته یا از ابتدا تا به حال بین این تلفن ها تماس برقرار شده باشد بین دو گره وجود دارد. یال ها را می توان با تعداد تماس های انجام شده بین این تلفن ها در طول دوره ی مشخص وزن داد. جوامع در یک شبکه تلفنی از گروههایی تشکیل می شوند که مرتباً ارتباط برقرار می کنند: برای مثال گروه هایی از دوستان ، اعضای یک باشگاه یا افرادی که در همان شرکت کار می کنند.

شبکه های ایمیلی

در شبکه های ایمیلی گره ها آدرس ایمیل افراد را نشان می دهند. یال ها بیانگر وجود حداقل یک ایمیل بین دو آدرس ایمیل می باشد. از طرف دیگر ممکن است لبه ها در این نوع شبکه ها به صورت یک طرفه یا دوطرفه باشند. از نمایش هرزنامه به عنوان دوست در این

شبکه ها خودداری می شود یا با رویکردی آدرس های ایمیل هرزنامه را با لبه های ضعیف و لبه های دیگر را با لبه های قوی تر نشان می دهیم.

اجتماعاتی که در شبکه های ایمیلی مشاهده می شود از همان گروه بندی هایی هستند که در ارتباط با شبکه های تلفنی از آنها یاد کردیم. یک نوع دیگر برای مرتب سازی شبکه های ایمیلی از افرادی که از طریق تلفن های همراه متن های خود را می نویسند است.

شبکه های همکاری

گره ها افرادی را نشان می دهند که مقالات تحقیقاتی را منتشر کرده اند. بین دو فرد که یک یا چند مقاله را به طور مشترک منتشر کرده اند ، لبه وجود دارد. به صورت اختیاری ، می توان لبه ها را با تعداد انتشارات مشترک برچسب گذاری کرد. گروه های این نوع شبکه نویسنده هایی هستند که روی یک موضوع خاص کار می کنند. نمای جایگزین از همان داده ها به عنوان گرافی است که در آن گره ها مقالات هستند. اگر حداقل یک نویسنده مشترک داشته باشند ، دو مقاله به یک لبه متصل می شوند. اکنون ، گروه هایی را تشکیل می دهیم که مجموعه ای از مقالات در همین موضوع را شامل هستند.

چندین نوع داده دیگر وجود دارد که دو شبکه را به روشی مشابه ایجاد می کند.

به عنوان مثال ، می توانیم به افرادی که مقالات ویکی پدیا را برای بار اول منتشر می کنند و افرادی که مقاله هایی را ویرایش می کنند تقسیم کنیم. اگر ویرایش یک مقاله به صورت مشترک باشد ، دو گره ویرایشگر به یکدیگر متصل هستند. گروه هایی از ویراستاران که به صورت مشترک کار کرده اند را در یک دسته قرار می دهیم. بطور مضاعف ، می توانیم شبکه ای از مقالات بسازیم و در صورت ویرایش آنها توسط همان شخص ، مقاله ها را وصل کنیم.

در اینجا ، ما مقالاتی را درمورد موضوعات مشابه یا مرتبط با هم جمع می کنیم.

در واقع ، داده های مربوط به همکاری ، همانطور که در فصل 9 مورد بحث قرار گرفت ، اغلب می توانند به عنوان تشکیل یک جفت شبکه ، یکی برای مشتریان و دیگری برای محصولات مشاهده شوند.

مشتریانی که کالاهای مشابهی را خریداری می کنند ، به عنوان مثال ، کتاب های علمی تأیید می کنند ، جوامع تشکیل می دهند و بصورت دوگانه ، کالاهایی که توسط همان مشتریان خریداری می شوند ، جوامع را تشکیل می دهند؛

به عنوان مثال ، تمام کتاب های علمی تولیدی.

مثال های دیگر از گراف های شبکه های اجتماعی

بسیاری از پدیده های دیگر گراف هایی ا ایجاد می کنند که چیزی شبیه به گراف های شبکه های اجتماعی است ، به خصوص نمایش محلیت ها.

مثالهای این بخش عبارتند از: شبکه های اطلاعاتی (اسناد ، گراف های در بستر وب ، ثبت اختراعات) ، شبکه های زیرساختی (جاده ها ، هواپیماها ، لوله های آب ، نیروگاهها) ، شبکه های بیولوژیکی (ژن ها ، پروتئین ها ، شبکه های غذایی حیوانات که یکدیگر را می خورند) و همچنین انواع دیگر ، مانند شبکه های خرید محصول (به عنوان مثال ، Groupon).

10.1.4 گراف ها دارای چندین نوع گره

پدیده های اجتماعی دیگری وجود دارند که موجودیت های مختلفی را درگیر می کنند. تحت عنوان "شبکه های همکاری" ، انواع مختلفی از گراف ها که از دو نوع گره واقعاً تشکیل شده اند را دیده ایم.

شبکه های نویسندگی می توانند گره های نویسنده و گره های کاغذ را تشکیل دهند. در بحث بالا ، ما با از بین بردن گره های یکی از این دو نوع ، دو شبکه اجتماعی ایجاد کردیم اما لازم نیست این کار را انجام دهیم. ما می توانیم به طور کلی به ساختار فکر کنیم.

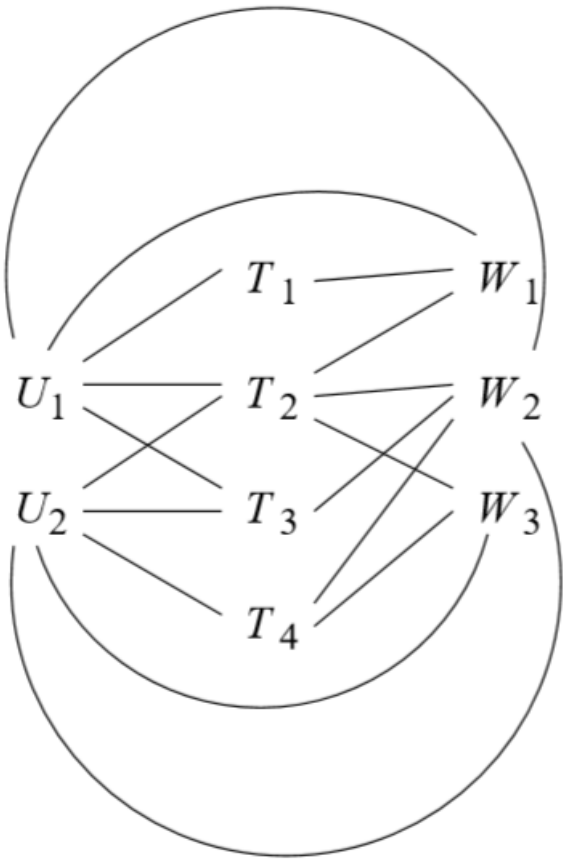
به عنوان مثال پیچیده تر ، کاربران در سایتی مانند del.icio.us برچسب ها را در صفحات وب قرار می دهند. بنابراین سه نوع مختلف موجودیت وجود دارد: کاربران ، برچسب ها و صفحات. ممکن است فکر کنیم اگر تمایل به استفاده از همان برچسب ها به طور مکرر یا اگر تمایل به برچسب زدن به همان صفحات دارند ، کاربران به نوعی به یکدیگر متصل شده اند. به طور مشابه ، برچسب ها اگر در همان صفحات ظاهر شوند می توانند مرتبط باشند؛ یا اگر برچسب ها توسط کاربران زیادی استفاده شود ممکن است صفحات مشابه به نظر برسند.

یک روش طبیعی برای نمایش چنین اطلاعاتی است که به عنوان گراف k-partite شناخته می شود؛ که k همیشه بزرگتر از 1 است. در بهش 8.3 گرافی دارای k=2 را مشاهده کرده ایم. به طور کلی یک نمودار k-partite از مجموعه گره هایی با k جداکننده تشکیل شده است که بین گره های یک گروه هیچ یالی وجود ندارد.

مثال 10.2 : شکل 10.2 نمونه ای از یک گراف k-partite است که مقدار k = 3 است. سه نوع گره وجود دارد: گره کاربران را در این گراف با {U1, U2} و گره برچسب ها را با {T1, T2, T3, T4} و گره صفخات وب را با {W1, W2, W3} نمایش می دهیم. توجه کنید تمام یال ها بین دو مجموعه مختلف متصل هستند.

ممکن است حس کنید این نمودار اطلاعاتی در مورد سه نوع موجودیت را نشان می دهد. به عنوان مثال لبه ی (U1, T2) به این معنی است که کاربر U1 برچسب T2 را حداقل در یک صفحه ی وب قرار داده است.

توجه کنید که گراف جزئیاتی را که می تواند مهم باشد را به ما نمی گوید. به عنوان مثال برای نمایش اینکه چه کسی چنین برچسبی را در این صفحه قرار داده است نیاز به نمایش پیچیده تری مانند روابط موجود در بانک های اطلاعاتی سه ستونه نیاز داریم.



شکل 10.2 : نمودار k-partite با مقدار k=3 برای نمایش کاربران، برچسب ها و صفحات وب

10.1.5 تمرینات مربوط به بخش 10.1

تمرین 10.1. : لبه های گراف G را به عنوان گره های G’ در نظر می گیریم.

1. اگر (X، Y) یالی از گراف G باشد ، XY ، نمایانگر مجموعه ای هماهنگ از X و Y گره ای از گراف G’ است. توجه داشته باشید که XY و Y X یک گره G’ را نشان می دهند ، و نه دو گره مختلف.
2. اگر (X، Y) و (X، Z) لبه های گراف G باشند ، در گراف G’ یک لبه بین XY و XZ وجود دارد. اگر گره های G’ که این گره ها نمایان هستند دارای یک گره (از G) مشترک هستند ، گره های G’ بین آنها یال دارد.

a) اگر ساخت گراف را به صورت دوتایی را در شبکه ای از دوستان بکار گیریم ، تعبیر لبه های نمودار نتیجه چیست؟

b) ساخت دوتایی را در گراف شکل 10.1 اعمال کنید.

c) درجه گره XY در گراف G’ چگونه با درجه X و Y در گراف G ارتباط دارد؟

d) تعداد لبه های گراف G’ مربوط به درجه گره های گراف G توسط یک فرمول خاص است. آن فرمول را کشف کنید.

e) آنچه ما آنرا به عنوان دوگان نامیدیم در واقع یک دوتایی واقعی نیست؛ زیرا استفاده از این روش ساختن گراف در گراف G' لزوماً یک ایزومورف گراف G می دهد. یک نمونه از گراف G را بدهید که در آن دوتایی از G' از نظر گراف G ایزومورف است و مثال دیگری هم بدهید که در آن گراف G' از نظر G ایزومورف نیست.

۱۰.۲ خوشه بندی گراف شبکه های اجتماعی

جنبه مهم شبکه های اجتماعی این است که آنها حاوی جوامع موجوداتی هستند که توسط بسیاری از لبه ها به هم وصل می شوند. به عنوان مثال ، اینها با گروهی از دوستان در مدرسه یا گروههایی از محققان علاقمند به همان موضوع مطابقت دارد. در این بخش ، خوشه بندی نمودار را به عنوان راهی برای شناسایی جوامع در نظر می گیریم. به نظر می رسد که تکنیک هایی که در فصل 7 آموخته ایم ، معمولاً برای مشکل خوشه بندی نمودارهای شبکه های اجتماعی نامناسب است.

10.2.1 معیار فاصله در گراف شبکه های اجتماعی

اگر بخواهیم از تکنیک های خوشه بندی استاندارد در یک گراف شبکه های اجتماعی استفاده کنیم ، اولین گام ما تعیین یک روش اندازه گیری فاصله است. هنگامی که یال های نمودار دارای برچسب هستند ، این برچسب ها بسته به آنچه که آنها نشان می دهند ، می توانند به عنوان اندازه گیری فاصله قابل استفاده باشند. اما هنگامی که لبه ها بدون برچسب هستند ، مانند نمودار «دوستان» ، برای تعیین فاصله ی مناسب ، کار زیادی نمی توان انجام داد.

اولین فرض ما این است که در نظر بگیریم گره ها نزدیک هستند و اگر یالی بین آنها باید یا نباشد دارای یک فاصله ی معین است. بنابراین ، می توان گفت که فاصله d(x,y) در صورت وجود لبه (x ،y) 1 است و در صورت عدم وجود چنین لبه ای 0 است. ما می توانیم از دو مقدار دیگری مثل 1 و ∞ استفاده کنیم تا زمانی که فاصله لبه ها به هم نزدیک تر باشد.

هیچ یک از معیارهای بررسی فاصله با دو ارزش 0 و 1 یا 0 و ∞ یک معیار واقعی و درست برای فاصله نیست. دلیل این امر این است که در هنگام اتصال مثلثی بین گره ها این مقدار فاصله بین دو گره نقض می شود. یعنی اگر لبه های (A ،B) و (B ،C) وجود داشته باشد ، اما هیچ لبه ای (A ،C) وجود ندارد ، در این صورت فاصله از A تا C از مجموع مسافت های A تا B از C بیشتر می شود. می توانیم برای این مشکل فاصله ی تا یک لبه را به طور مستقیم 1 در نظر بگیریم و برای missing edge ها از فاصله ی 1.5 استفاده کنیم که اینکار مشکل بالا را حل می کند اما مشکل توابع محاسبه ی فاصله به نابرابری مثلثی محدود نمی شود که در بخش بعدی این موضوع را خواهیم دید.

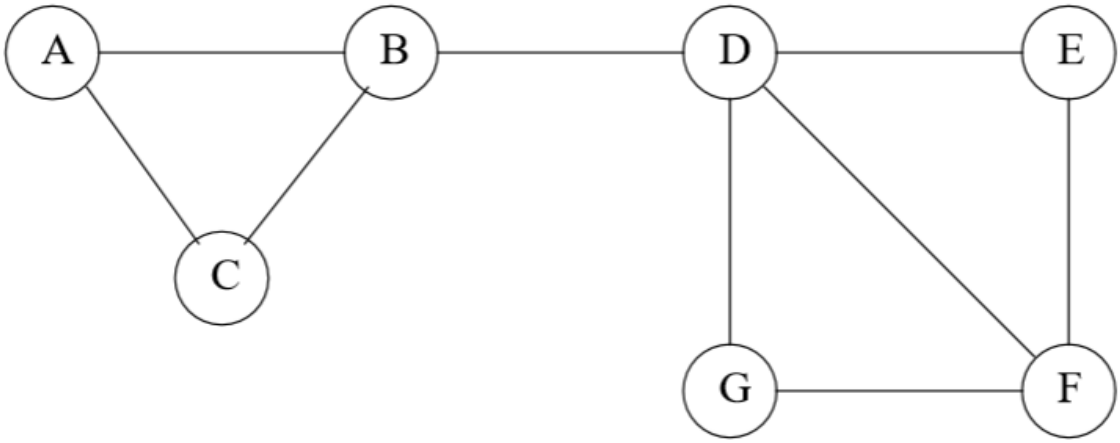
10.2.2 اعمال متدهای خوشه بندی استاندارد

از بخش 7.1.2 به یاد بیاورید که دو رویکرد کلی برای خوشه بندی وجود دارد: سلسله مراتبی (agglomerative) و تعیین امتیاز. در اینجا قصد این را داریم که چگونگی کار هر یک از این موارد را روی گراف های شبکه های اجتماعی بررسی کنیم.

ابتدا روش های سلسله مراتبی را که در بخش 7.2 است را در نظر بگیرید. به طور خاص ، فرض کنید که ما به عنوان فاصله متقاطع از حداقل فاصله بین گره های دو خوشه استفاده می کنیم.

خوشه بندی سلسله مراتبی از یک گراف شبکه های اجتماعی با ترکیب دو گره که به یک لبه متصل هستند ، آغاز می شود. به طور موفقیت آمیز ، لبه هایی که بین دو گره از یک خوشه یکسان نیستند به طور تصادفی انتخاب می شوند تا خوشه هایی را که دو گره آنها به آنها تعلق دارد ، ترکیب کنند. گزینه ها تصادفی هستند ، زیرا تمام مسافت هایی که توسط یک لبه نشان داده می شوند یکسان هستند.

مثال 10.3 : نمودار 10.1 در اینجا مانند شکل 10.3 تکرار شده است. اول ، بگذارید درمورد اینکه مجموعه ها چیست ، توافق کنیم. در بالاترین سطح ، به نظر می رسد که دو اجتماع {A,B,C} و {D,E,F,G} وجود دارند. با این حال ، ما همچنین می توانیم { D، E، F} و { D، F، G} را به عنوان دو زیرمجموعه از {D,E,F,G} در نظر بگیریم. این دو زیر مجموعه در دو عضوشان با هم همپوشانی دارند و بنابراین هرگز نمی توان با یک الگوریتم خوشه بندی خالص آنها را شناسایی کرد. سرانجام ، ما می توانیم هر جفت از افراد را که به یک لبه متصل هستند ، به عنوان یک جامعه با اندازه 2 در نظر بگیریم ، هرچند که چنین اجتماعاتی نگران کننده نیستند.



شکل 10.3 : تکرار مجدد شکل 10.1

مشکل خوشه بندی سلسله مراتبی از یک گراف مانند شکل 10.3 این است که در بعضی از نقاط احتمالاً ما ترکیب B و D را انتخاب می کنیم ، حتی اگر آنها مطمئناً در خوشه های مختلف قرار داشته باشند. دلیل اینکه ما احتمالاً B و D را با هم ترکیب کرده ایم این است که D و هر خوشه ای که حاوی آن باشد به همان اندازه نزدیک به B و هر خوشه ای است که حاوی آن باشد ، همانطور که A و C به B وجود دارد. حتی یک احتمال 9/1 وجود دارد که اولین کاری که ما انجام می دهیم ترکیب B و D در یک خوشه است.

مواردی وجود دارد که می توانیم برای کاهش احتمال خطا انجام دهیم. ما می توانیم چندین بار خوشه بندی سلسله مراتبی را اجرا کنیم و اجرا را انتخاب کنیم که منسجم ترین خوشه ها را داشته باشد. ما می توانیم از روش پیچیده تری برای اندازه گیری فاصله بین خوشه های بیش از یک گره استفاده کنیم ، همانطور که در بخش 7.2.3 بحث شده است. اما مهم نیست که چه کاری انجام می دهیم ، در یک نمودار بزرگ با بسیاری از جوامع ، شانس قابل توجهی وجود دارد که در مراحل اولیه باید از برخی لبه ها استفاده کنیم که دو گره را به هم متصل می کنند که در هیچ جامعه بزرگی وجود ندارند.

حال یک روش تعیین تکلیف را در مورد خوشه بندی شبکه های اجتماعی در نظر بگیرید. باز هم ، این واقعیت که همه لبه ها در یک فاصله قرار دارند ، تعدادی از عوامل تصادفی را معرفی می کند که منجر به اختصاص برخی گره ها به خوشه اشتباه می شود. با یک مثال باید نکته را نشان دهد.

مثال 10.4 : فرض کنید ما روش k-means را عنوان خوشه بندی در نظر بگیریم. در این خوشه بندی k=2 می باشد. اگر انتخاب دو گره را به صورت تصادفی انجام دهیم ممکن است هر دو در یک خوشه قرار بگیرند. اگر مانند بخش 7.32 با یک گره به صورت تصادفی شروع کنیم و بعد گره دیگری را انتخاب کنیم نتجه را خیلی بهتر نکرده ایم. از این طریق می توانیم هر جفت گره ای را که به یک لبه وصل نشده است ، انتخاب کنیم ، به عنوان مثال ، E و G که در شکل 10.3 است. با این حال فرض کنید ما با دو گره مناسب مثل B و F شروع کنیم. سپس A و C را به خوشه B اختصاص می دهیم و E و G را به خوشه F اختصاص می دهیم. اما گره D به اندازه ای که متعلق به F می باشد به همان اندازه متعلق به B است ، بنابراین می تواند به هر شکلی پیش برود ، حتی اگر "آشکار" باشد که D متعلق به F است باز هم این اتفاق می افتد.

اگر تصمیمی در مورد محل قرارگیری D به تعویق بیفتد تا زمانی که گره های دیگری را به خوشه ها اختصاص ندهیم ، احتمالاً تصمیم درست می گیریم. به عنوان مثال ، اگر ما یک گره را با کمترین مقاومت متوسط به همه گره های خوشه ، به خوشه اختصاص دهیم ، باید D را به خوشه F اختصاص دهیم ، مادامی که سعی نکنیم D را قبل از اینکه گره های دیگری اختصاص دهند ، قرار می دهیم. با این حال ، در نمودارهای بزرگ ، مطمئناً در برخی از اولین گره هایی که قرار می دهیم اشتباه می کنیم.

10.2.3 مفهوم Betweenness

از آنجا که در روش های استاندارد خوشه بندی مشکل خوشه بندی اختصاصی برای اجتماعات گراف شبکه های اجتماعی وجود دارد در این بخش یکی از ساده ترین روش ها را که بر اساس تعیین حاشیه هایی که احتمالاً در یک جامعه قرار دارند ، در نظر می گیریم.

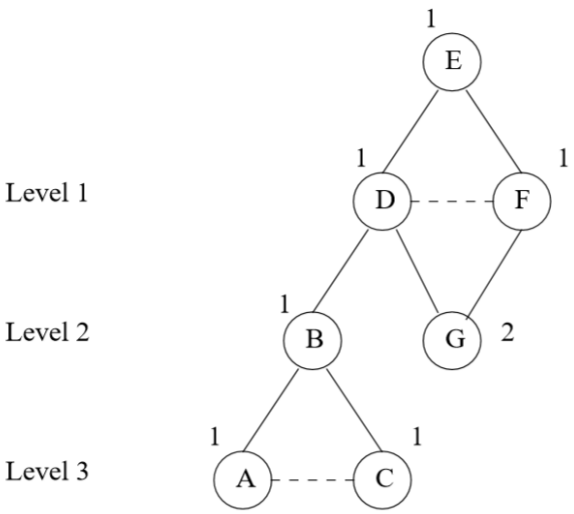
باید اتصال لبه ی (a، b) به تعداد جفت گره های x و y به گونه ای باشد که لبه (a، b) در کوتاهترین مسیر بین x و y قرار داشته باشد. به بیان دقیق تر ، از آنجا که می توانید چندین مسیر کوتاه بین x و y وجود داشته باشد ، لبه (a، b) با کسری از آن کوتاه ترین مسیرها که شامل لبه (a، b) است ، اعتبار دارد. مانند گلف ، که در آن امتیاز بالا بد است. این نشان می دهد که لبه (a، b) بین دو اجتماع مختلف جریان دارد. یعنی ، a و b متعلق به یک مجموعه نیستند.

مثال 10.5 : در شکل 10.3 لبه (B، D) بالاترین حد فاصل را دارد؛ در حقیقت ، این لبه در هر کوتاه ترین مسیری بین هر یک از A، B و C به هر یک از D، E، F و G قرار دارد. فاصله آن 12 است. در مقابل ، لبه (D، F) فقط در چهار مسیر کوتاه قرار دارد: آنهایی که از A، B، C و D تا F هستند.

10.2.4 الگوریتم Girvan-Newman

برای بهره برداری از لبه ها ، باید تعداد کوتاهترین مسیری که در هر لبه طی می شود را محاسبه کنیم. روشی را به نام الگوریتم Girvan-Newman (GN) توصیف می کنیم ، که یک بار از هر گره X بازدید می کند و تعداد کوتاهترین مسیرها را از X به هر گره دیگری که از هر یک از لبه ها عبور می کند محاسبه می کند. این الگوریتم با روش (BFS) در گراف ، از گره X شروع می کند. توجه داشته باشید که سطح هر گره در نمایش BFS طول کوتاهترین مسیر از X تا آن گره است. بنابراین ، لبه هایی که بین گره ها در یک سطح قرار دارند هرگز نمی توانند بخشی از کوتاهترین مسیری از X باشند.

لبه های بین سطوح ، لبه های DAG نامیده می شوند ("DAG" مخفف directed, acyclic graph است). هر لبه DAG بخشی از حداقل یک مسیر کوتاه از ریشه X خواهد بود. اگر یک لبه ی Dag در (Y,Z) وجود داشته باشد جایی که Y در سطح بالاتر از Z قرار دارد (یعنی نزدیک به ریشه) ، آنگاه ما Y را والدین Z و Z فرزند Y می نامیم ، گرچه آنها لزوماً در DAG والدین یکتایی نیستند که به عنوان یک درخت قرار بگیرند.



شکل 10.4 : مرحله اول الگوریتم Girvan-Newman

مثال 10.6 : شکل 10.4 روش breadth-first از گراف شکل 10.3 است که از گره E شروع می شود. یال های غیرخط چین لبه های DAG هستند و لبه های خط چین گره ها را در همان سطح قرار می دهند.

مرحله دوم الگوریتم Girvan-Newman ، برچسب زدن هر گره با تعداد کوتاهترین مسیری است که از ریشه به آن می رسد. با برچسب زدن به ریشه شروع کنید. سپس ، از بالا به پایین ، هر گره Y را بر اساس برچسب های والدین آن ، برچسب گذاری کنید.

مثال 10.7 : در شکل 10.4 برچسب های هر یک از گره ها نشان داده شده است. ابتدا ریشه که E می باشد را با 1 علامت گذاری کنید. در سطح اول گره های D و F هستند. هر کدام فقط گره E را به عنوان والدین دارند ، بنابراین آنها نیز دارای برچسب 1 هستند. گره های

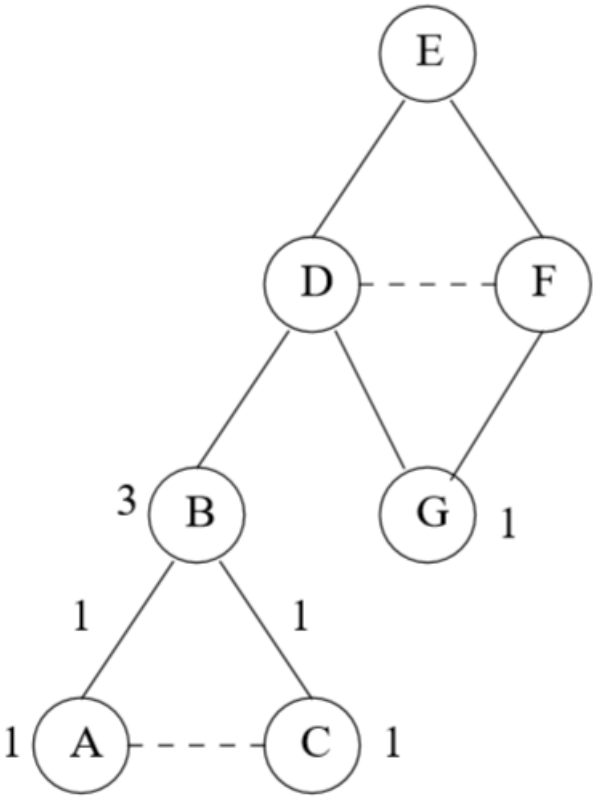
B و G در سطح دو قرار دارند. گره B فقط والدین گره D را دارد ، بنابراین برچسب B برابر با برچسب D است که 1 است. با این حال ، گره G دارای والدین D و F است ، بنابراین برچسب آن مجموع برچسب های آنها یا 2 است. سرانجام ، در سطح سه، گره A و C هر یک فقط والدین B را دارند ، بنابراین برچسب های آنها برچسب B است که 1 است.

مرحله سوم و پایانی محاسبه برای هر لبه ی e و مجموع بیش از همه گره ها Y از کسری از کوتاه ترین مسیرها از ریشه X تا Y است که از e عبور می کند. این محاسبه شامل محاسبه این هزینه برای گره ها و لبه ها ، از پایین است. به هر گره به غیر از ریشه هزینه ی 1 داده می شود ، که کوتاه ترین مسیر را به آن گره نشان می دهد. این هزینه ممکن است بین گره ها و لبه های بالا تقسیم شود ، زیرا ممکن است چندین مسیر کوتاه مختلف به گره وجود داشته باشد. قوانین محاسبه به شرح زیر است:

1. برگ ها در DAG دارای هزینه ی 1 هستند.
2. گره های غیر برگ دارای هزینه ی 1 به علاوه جمع هزینه ی DAG نسبت به نودهای سطح پایینتر است.
3. یک لبه DAG و بخش ورودی به گره Z از سطح فوق با هزینه ی Z متناسب با کسری از کوتاهترین مسیرها از ریشه تا Z که از E عبور می کند ، داده می شود. به طور کلی ، والدین Z را به شکل Y1، Y2، ...، Yk در نظر بگیرید و بگذارید pi تعداد کوتاهترین مسیرها از ریشه تا Yi باشد. این عدد در مرحله دو محاسبه شده است و توسط برچسب ها در شکل 10.4 نشان داده شده است. سپس هزینه لبه (Z ،Yi) برابر $\Sigma_{j=1}^{k} Pj$ است.

پس از انجام محاسبه هزینه ی هر گره به عنوان ریشه ، اعتبارات مربوط به هر لبه را جمع می کنیم. سپس ، از آنجا که هر کوتاهترین مسیر دو بار کشف شده است - یک بار وقتی که هر یک از نقاط انتهایی آن ریشه دارد - باید اعتبار هر لبه را به 2 تقسیم کنیم.

مثال 10.8 : محاسبه هزینه را برای روش BFS که از شکل است10.4 انجام می دهیم. ما باید از سطح سه شروع کنیم و به سمت بالا پیش برویم. اول ، A و C ، که برگ می شوند ، اعتبار 1 را دریافت می کنند. هر یک از این گره ها فقط یک والد دارند ، بنابراین اعتبار آنها به ترتیب به لبه ها (A ،B) و (C ،B) داده می شود.



شکل 10.5 : مرحله ی نهایی الگوریتم Girvan-Newman

در سطح دو ، G یک برگ است ، بنابراین اعتبار 1 را دریافت می کند. B برگ نیست ، بنابراین اعتبار آن برابر با 1 به علاوه اعتبارات در لبه های DAG که از زیر آن وارد شده است ، می شود. از آنجا که هر دو این لبه دارای اعتبار 1 هستند ، اعتبار B برابر 3 است. به طور شهودی 3 این واقعیت را نشان می دهد که تمام کوتاهترین مسیرها از E به A، B و C از B عبور می کنند. شکل 10.5 اعتبارات اختصاص یافته تاکنون را نشان می دهد.

اکنون ، اجازه دهید به سطح 1 برویم. B تنها یک والد دارد ، D ؛ بنابراین لبه (B ،D) اعتبار کل B را دریافت می کند ، که 3 است. با این حال ، G دو والدین دارد ، D و F.

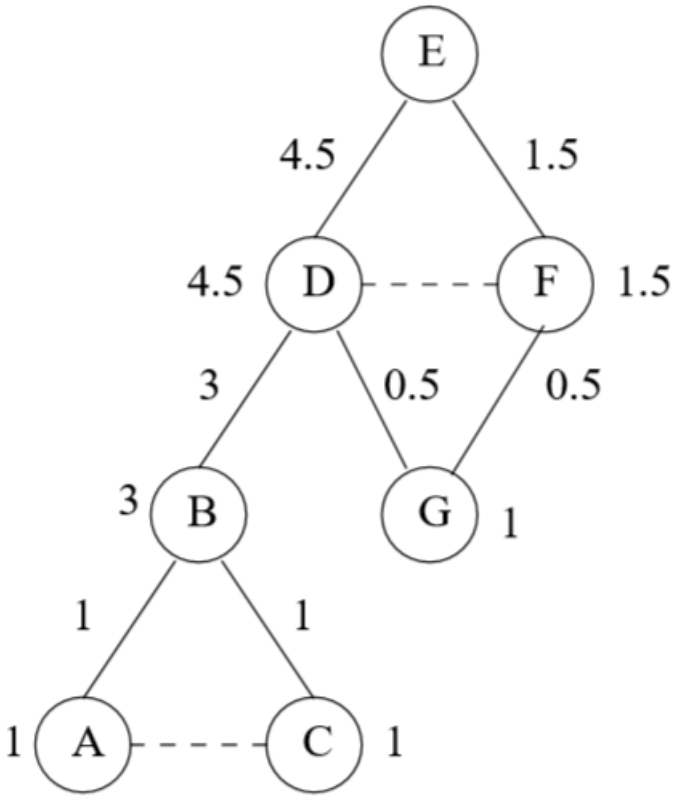
بنابراین ما باید اعتبار 1 را که G دارد بین لبه ها (D، G) و (F، G) تقسیم کنیم.

به چه نسبت تقسیم می کنیم؟

اگر برچسب های شکل 10.4 را بررسی می کنید ، می بینید که هر دو D و F دارای برچسب 1 هستند ، این نشان دهنده این واقعیت است که یک کوتاهترین مسیر از E به هر یک از این گره ها وجود دارد.

بنابراین ، ما نصف اعتبار G را به هر یک از این لبه ها می دهیم ؛ یعنی ، اعتبار آنها هر 1 / (1 + 1) = 0.5 است.

اگر برچسب های D و F در شکل 10.4 برابر 5 و 3 باشد ، به این معنی که کوتاه ترین مسیرها به D و فقط سه تا F وجود داشته است ، پس اعتبار لبه (D,G) برابر پنج هشتم 5/8 است و لبه (F، G) می توانست سه هشتم 3/8 باشد.



شکل 10.6 : مرحله ی نهایی الگوریتم Girvan-Newman – با اتمام محاسبه ی اعتبارها(هزینه ها)

اکنون ، می توانیم اعتبارات را به گره ها در سطح 1 اختصاص دهیم. گره D اعتبار 1 را می گیرد به علاوه اعتبار لبه هایی که از زیر آن وارد می شوند ، 3 و 0.5 است. یعنی اعتبار گره D برابر 4.5 است. اعتبار F برابر 1 به علاوه اعتبار لبه (G ،F) یا 1.5 است. سرانجام ، لبه های (D ،E) و (F ،E) به ترتیب اعتبار D و F را دریافت می کنند ، زیرا هر یک از این گره ها فقط یک والدین دارند. این اعتبارات همه در شکل 10.6 نشان داده شده است.

اعتبار هر یک از لبه ها در شکل 10.6 سهم در فاصله بین آن لبه به دلیل کوتاه ترین مسیرهای E. است. برای مثال ، این سهم برای لبه ی (E,D) برابر 4.5 است.

برای تکمیل محاسبه فاصله بین آنها ، باید این محاسبه را برای هر گره به عنوان ریشه تکرار کنیم و سهم ها را با هم جمع کنیم. سرانجام، باید 2 را تقسیم کنیم تا فاصله ی حقیقی بین آنها را پیدا کنیم ، زیرا هر کوتاهترین مسیر دو بار ، یک بار برای هر یک از نقاط پایانی آن کشف می شود.

10.2.5 استفاده از مفهوم Betweenness در پیدا کردن مجموعه ها

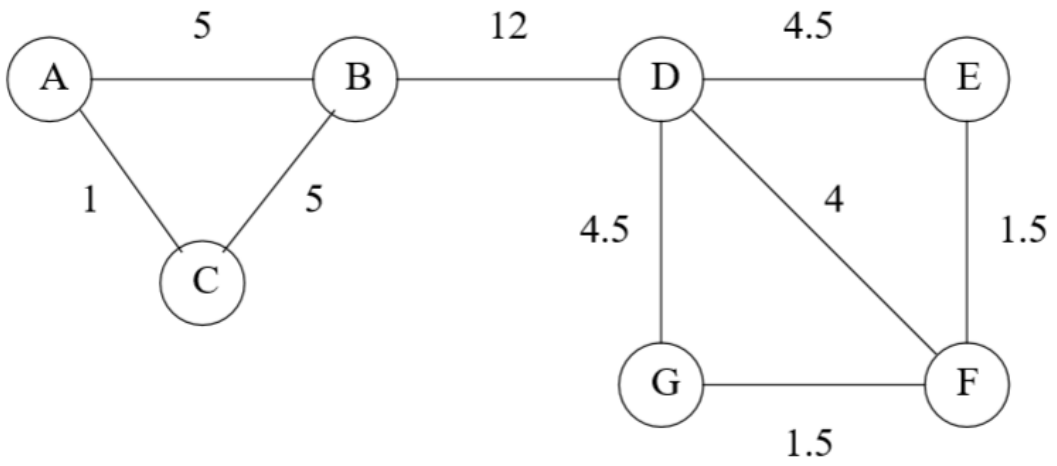
هزینه بین لبه های نمودار چیزی شبیه به اندازه گیری مسافت بر روی گره های گراف است. این دقیقاً اندازه گیری مسافت نیست ، زیرا برای جفت گره هایی که به یک لبه وصل نشده اند ، تعریف نمی شود و ممکن است نابرابری مثلثی را حتی در صورت تعریف ، برآورده نکند. با این حال ، ما می توانیم با گرفتن لبه ها به منظور افزایش فاصله ، خوشه ای ایجاد کنیم و هر یک را یکبار به گراف اضافه کنیم.

در هر مرحله ، اجزای متصل به گراف برخی خوشه ها را تشکیل می دهند. هرچه فاصله بین آنها بیشتر باشد ، لبه های بیشتری می گیریم و خوشه ها بزرگتر می شوند.

به طور معمول ، این ایده به عنوان فرآیند حذف لبه بیان می شود. با گراف و تمام یال های آن شروع کنید. سپس یال ها را با بیشترین فاصله حذف می کنیم، تا زمانی که نمودار به تعداد مناسبی از اجزای متصل شکسته شود.

مثال 10.9 : با مثال قبلی، گراف شکل 10.1 شروع می کنیم. ما این گراف را با فاصله بین هر لبه در شکل 10.7 می بینیم. محاسبه فاصله بین خواننده خواهد بود. تنها بخش دشوار شمارش این است که مشاهده کنید که بین E و G دو کوتاه ترین مسیر وجود دارد که یکی از D عبور می کند و دیگری از طریق F.

بنابراین ، به هر یک از لبه های (D,E) و (E,F) و (D,G) و (G,F) با نیم کوتاه ترین مسیر اعتبار داده می شود.

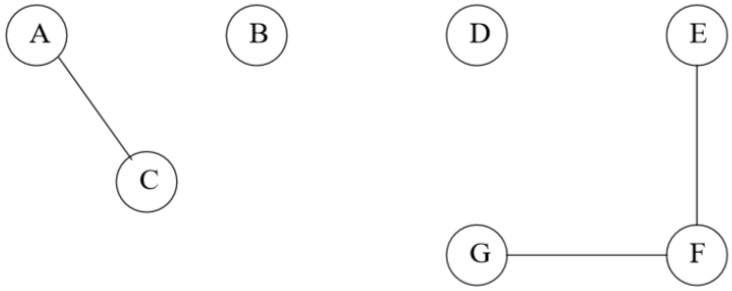


شکل 10.7 : امتیاز بین گره های گراف شکل 10.1

بدیهی است که لبه (B، D) بالاترین حد فاصل را دارد ، بنابراین اولین بار برداشته می شود. این دقیقاً حس جامعه را به ما رهنمون می کند ، یعنی: { A، B، C} و { D، E، F، G }.

با این حال ، ما می توانیم به حذف لبه ها ادامه دهیم. در مرحله بعد (A، B) و (B، C) با امتیاز 5 قرار دارند و پس از آن (D، E) و (D، G) با امتیاز 4.5.

سپس ، (D، F) ، که نمره آن 4 است ، نمودار را ترک می کند. در شکل 10.8 گراف هایی را که می ماند می بینیم.



شکل 10.8 : تمام لبه های بین 4 یا بیشتر برداشته شده است.

"اجتماعات" در شکل 10.8 عجیب به نظر می رسند. یکی از دلالت ها این است که A و C از نزدیک به یکدیگر گره خورده اند تا B. یعنی به نوعی B یک "خائن" جامعه است - A، B، C - زیرا او در خارج از جامعه دوست D دارد. به همین ترتیب ، D را می توان به عنوان "خائن" به گروه - {D,E,F,G} در نظر گرفت ، به همین دلیل در شکل 10.8 ، فقط E، F و G به هم وصل می شوند.

سرعت بخشیدن در بحث محاسبه ی Betweenness

اگر ما روش بخش 10.2.4 را در یک نمودار از گره های n و لبه های e استفاده کنیم ، زمان محاسبه O (ne) زمان لازم برای محاسبه فاصله بین هر لبه است. یعنی BFS از یک گره واحد زمان O (e) را نیز می گیرد ، همانطور که دو مرحله برچسب زدن را انجام می دهید. باید از هر گره شروع کنیم ، بنابراین n از محاسبات شرح داده شده در بخش 10.2.4 وجود دارد.

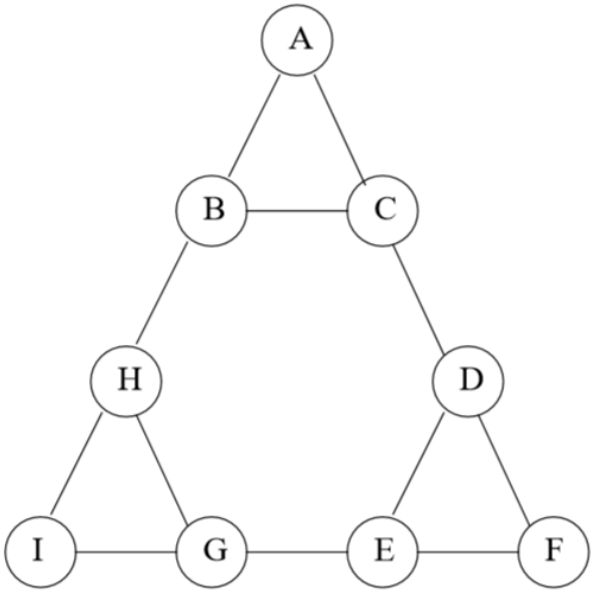
اگر نمودار بزرگ باشد ما نمی توانیم نظمی را برای اجرای آن اجرا کنیم. با این وجود ، اگر زیرمجموعه ای از گره ها را بطور تصادفی انتخاب کنیم و از این ها به عنوان ریشه های جستجو برای اولین بار استفاده کنیم ، می توانیم به تقارن بین هر لبه ای که در اکثر برنامه ها خدمت می کند ، برسیم.

10.2.6 تمرین های بخش 10.2

تمرین 10.2.1 : شکل 10.9 نمونه ای از گراف شبکه های اجتماعی است. از روش Girvan-Newman استفاده کنید تا تعداد کوتاهترین مسیرهای هر کدام از گره های زیر را که از هر یک از لبه ها عبور می کند ، تعیین کنید.

a. A

b. B



شکل 10.9 : گراف مربوط به تمرین

تمرین 10.2.2 : با استفاده از تقارن ، محاسبات تمرین 10.2.1 ، تمام موارد لازم برای محاسبه فاصله بین هر لبه است این محاسبات را انجام دهید.

تمرین 10.2.3: با استفاده از مقادیر فاصله از روش 10.2.2 ، نامزدهای معقول را برای جوامع در شکل 10.9 با حذف تمام لبه ها با فاصله بالای برخی از آستانه تعیین کنید.

۱۰.۳ کشف مستقیم مجموعه ها یا جوامع

در بخش قبلی ما با جدا کردن همه افراد در یک شبکه اجتماعی ، جوامع را جستجو کردیم. اگرچه این رویکرد نسبتاً مؤثر است ، اما محدودیت دارد. قرار دادن فرد در دو جامعه مختلف امکان پذیر نیست و همه به یک جامعه اختصاص داده می شوند. در این بخش ، با جستجوی زیرمجموعه هایی از گره ها که تعداد نسبتاً زیادی از لبه ها در بین آنها وجود دارد ، می توانیم تکنیکی برای کشف مستقیم جوامع مشاهده کنیم. جالب است که ، تکنیک انجام این جستجو بر روی یک گراف بزرگ ، شامل موارد سنگین اجرایی و به صورت مکرر است ، همانطور که در فصل 6 مورد بحث قرار گرفت.

10.3.1 پیدا کردن Cliques

اولین فکر ما در مورد اینکه چگونه می توانیم مجموعه های گره هایی که لبه های زیادی بین آنها وجود دارد را با ایجاد یک clique بزرگ (مجموعه ای از گره ها با لبه های بین هر دو آنها) شروع کنیم. با این حال ، این کار آسان نیست. نه تنها clique ها حداکثر NP کامل است ، بلکه به این معنا که حتی تقریب حداکثر clique سخت است.

علاوه بر این ، می توان مجموعه ای از گره ها را که تقریباً همه لبه ها در بین آنها وجود دارد در عین حال فقط دارای clique های نسبتاً کوچکی می باشد.

تمرین 10.10 : فرض کنید گراف ما دارای گره هایی به شماره 1،2 ، ... ، n باشد و یک لبه بین دو گره i و j وجود دارد ، مگر اینکه i و j باقیمانده با k تقسیم شوند. سپس کسری از لبه های ممکن که در واقع وجود داشته باشد تقریباً (k-1) / k است. تعداد زیادی clique با اندازه k وجود دارد که از آن {k,...,1,2} فقط یک clique است.

با این حال هیچ cliqueیی بزرگتر از k وجود ندارد. برای دیدن دلیل این موضوع ، مشاهده کنید که هر مجموعه گره k + 1 دارای دو عدد است که هنگام تقسیم بر k ، باقیمانده باقی می مانند.

این نکته کاربردی از "اصل لانه کبوتری" است. از آنجا که فقط باقیمانده های مختلف K ممکن است ، ما نمی توانیم بقایای مشخصی برای هر یک از گره های k + 1 داشته باشیم. بنابراین ، هیچ مجموعه ای از گره های k + 1 نمی تواند یک clique در این گراف باشد.

10.3.2 گراف های دوبخشی کامل

بحث گراف های دو طرفه را از بخش 8.3 به یاد بیاورید. یک گراف کامل دو طرفه از گره های s در یک طرف و گره های t در طرف دیگر تشکیل شده است که تمام لبه های ممکن بین گره های یک طرف و طرف دیگر موجود است. ما این گراف را توسط $K_{s,t}$ نشان می دهیم. شما باید قیاس بین گراف های کامل دو طرفه را به عنوان زیرگراف نمودارهای دو طرفه کلی و کلیشه ها به عنوان زیرگرافهای نمودارهای کلی ترسیم کنید.

در حقیقت ، غالباً به کلیشه گره های s به عنوان نمودار کاملی گفته می شود و Ks را نشان می دهد ، در حالی که یک زیرگراف کامل دو طرفه گاهی اوقات یک biclique خوانده می شود.

در حالی که همانطور که در مثال 10.10 دیدیم ، نمی توان تضمین کرد که یک گراف با بسیاری از لبه ها لزوماً دارای یک clique بزرگ است ، می توان تضمین کرد که یک نمودار دو طرفه با بسیاری از لبه ها دارای یک زیرگراف کامل دو طرفه کامل است.

ما می توانیم یک زیرگراف کامل دو طرفه (یا یک clique اگر یک مورد بزرگ را کشف کردیم) به عنوان هسته یک جامعه در نظر بگیریم و گره هایی با لبه های زیادی به اعضای موجود در جامعه اضافه کنیم.

اگر خود گراف همانطور که در بخش 10.1.4 مورد بحث قرار گرفته است k-partite باشد ، می توان گره هایی از دو نوع و لبه های بین آنها را در نظر گرفت تا یک گراف دو بخشی ایجاد کنیم. در این گراف دو بخشی ، ما می توانیم برای زیرگرافهای کاملاً دو طرفه به عنوان هسته جوامع جستجو کنیم.

به عنوان مثال ، در مثال 10.2 ، ما می توانیم بر روی برچسب ها و گره های صفحه از گرافیکی مانند شکل 10.2 تمرکز کنیم و سعی کنیم مجموعه های برچسب ها و صفحات وب را ایجاد کنیم.

چنین جامعه ای از برچسب ها و صفحات مربوطه تشکیل می شود که مستحق بسیاری از این برچسب ها هستند.

با این حال ، ما همچنین می توانیم از زیرگراف های کاملاً دو بخشی برای نمایش اجتماع در گراف های معمولی استفاده کنیم که گره ها همه از یک نوع هستند.

گره ها را بطور تصادفی به دو گروه مساوی تقسیم کنید.

اگر جامعه ای وجود داشته باشد ، انتظار داریم تقریباً نیمی از گره های آن در هر گروه قرار بگیرد و ما انتظار داریم که حدود نیمی از لبه های آن بین گروه ها قرار بگیرد.

بنابراین ، ما هنوز یک فرصت منطقی برای شناسایی یک زیرگراف کامل کامل دو طرفه در جامعه داریم.

به این هسته می توان گره هایی از هر دو گروه اضافه کرد ، در صورتی که دارای لبه هایی در بسیاری از گره هایی هستند که قبلاً متعلق به جامعه هستند.

10.3.3 پیداکردن زیرگراف های کامل دوبخشی

فرض کنید به ما یک نمودار بزرگ دو طرفه G داده شده است ، و ما می خواهیم نمونه های $K_{s,t}$را در درون خود قرار دهیم. می توان مشکل در بروز نمونه های $K_{s,t}$در G را به عنوان یکی از موارد اقلام مکرر مشاهده کرد. برای این منظور ، بگذارید "موارد" گره هایی در یک طرف G باشند که ما آن را سمت چپ می نامیم.

ما فرض می کنیم که نمونه $K_{s,t}$، ما به دنبال آن هستیم گره های t در سمت چپ وجود داشته باشد ، و ما همچنین برای این اطمینان را تصور خواهیم کرد که این ویژگی ها را داشته باشد. "سبدهای" مربوط به گره های طرف دیگر G (سمت راست) است. اعضای سبد برای گره v گره های سمت چپ است که v به آن وصل شده است. سرانجام ، اجازه دهید آستانه پشتیبانی s باشد ، تعداد گره هایی که به عنوان مثال $K_{s,t}$ در سمت راست دارند.

اکنون می توانیم مسئله ایجاد نمونه های $K_{s,t}$، مانند ایجاد آیتم های مکرر F با اندازه t را بیان کنیم. یعنی اگر مجموعه ای از گره های t در سمت چپ مکرر باشند ، پس از آن همه در حداقل سبد های s اتفاق می افتند. اما سبدها گره هایی در سمت راست هستند. هر سبد مطابق با گره ای است که به تمام گره های F متصل شده است. بنابراین ، آیتم های مکرر از اندازه و t از سبدهای که در آن همه موارد مشاهده می شود نمونه ای از $K_{s,t}$ است.

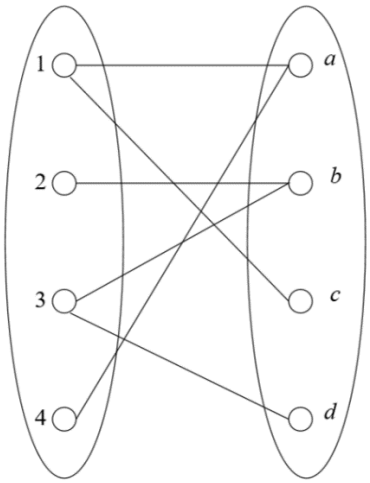
مثال 10.11 : نمودار دو بخشی شکل 8.1 را به یاد بیاورید؛در اینجا مثل شکل 10.10 آن را تکرار می کنیم. سمت چپ گره ها {1،2،3،4} و سمت راست { a، b ،c ،d } است. دسته دوم سبدها هستند ، بنابراین سبد مجموعه ای از "موارد" 1 و 4 است.

a{1,4} =

b{2,3} =

c{1}=

d{3} =



شکل 10.10 : گراف دوبخشی برگفته شده از شکل 8.1

اگر $s = 2$ و $t = 1$ ، باید وسایل اندازه 1 را انتخاب کنیم که حداقل در دو سبد وجود داشته باشد. {1} یکی از این موارد است و {3} دیگر.با این حال ، در این مثال کوچک هیچ آیتمی برای مقادیر بزرگتر و جالب تر از s و t وجود ندارد ، مانند $s = t = 2$.

10.3.4چرا نمودارهای دوبخشی کامل باید وجود داشته باشد

حال ما باید به این موضوع بپردازیم که نشان دهیم که هر نمودار دو طرفه با کسری مناسب از لبه های موجود نمونه ای از $K_{s,t}$ دارد. در شکل زیر فرض کنید که نمودار G دارای گره هایی در سمت چپ و یک گره دیگر در سمت راست است.

فرض کنید که دو طرف تعداد گره های یکسانی دارند ، محاسبه را ساده می کند ، اما آرگومان به هر طرف اندازه می گیرد. سرانجام ، بگذارید d درجه متوسط همه گره ها باشد.

این استدلال شامل شمارش تعداد موارد مکرر اندازه t است که یک سبد با آیتم d به آن کمک می کند. وقتی این عدد را از همه گره ها در سمت راست جمع کنیم ، فرکانس کل همه زیر مجموعه های اندازه t را در سمت چپ می گیریم. وقتی تقسیم بر $\binom{n}{t}$ می شود ، ما فرکانس متوسط همه آیتم های اندازه t را بدست می آوریم. حداقل یک فرکانس باید حداقل باشد ، بنابراین اگر این میانگین حداقل s باشد ، می دانیم نمونه ای از $K_{s,t}$وجود دارد ، t وجود دارد.

$$\frac{n\binom{d}{t}}{\binom{n}{t}} = \frac{nd!\,(n-t)!\,t!}{(d-t)!\,t!\,n!} = n(d)(d-1)\ldots\frac{d-t+1}{n(n-1)\ldots(n-t+1)}$$

$$n\left(\frac{d}{n}\right)^{t} \geq s$$

10.3.5 تمرین های بخش 10.3

تمرین 10.3.1 : برای مثال در حال اجرا یک شبکه اجتماعی از شکل 10.1 ، چه تعداد از $K_{s,t}$، برای این موارد وجود دارد:

a- S=1 و t=3

b- S=2 و t=2

c- S=2 و t=3

تمرین 10.3.2 : فرض کنید جامعه ای از گره های 2n وجود دارد. جامعه را به طور تصادفی به دو گروه n اعضا تقسیم کنید و نمودار دو طرفه را بین دو گروه تشکیل دهید. فرض کنید متوسط درجه گره های نمودار دو قطبی d باشد. مجموعه جفت های حداکثر (s ،t) را با ضریب t پیدا کنید ، به این صورت که نمونه ای از $K_{s,t}$، وجود داشته باشد ، برای ترکیب های زیر از n و d .

a- n=20 و d=5

b- n=200 و d=150

c- n=1000 و d=400

منظور از "حداکثر" ، منظور ما این نیست که هیچ جفت متفاوتی (s', t') وجود داشته باشد که هر دو آنها را حفظ کنند.

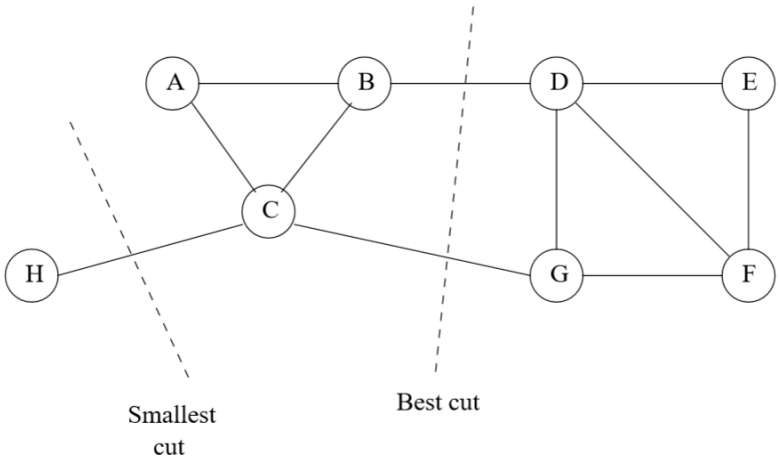
۱۰.۴ پارتیشن بندی کردن گراف

در این بخش یک رویکرد دیگر برای سازماندهی گراف های شبکه های اجتماعی را بررسی می کنیم. ما از برخی ابزارهای مهم نظریه ماتریس ("روشهای طیفی") برای شکل دادن به مشکل تقسیم بندی یک نمودار برای به حداقل رساندن تعداد لبه های اتصال اجزای مختلف استفاده می کنیم. هدف از به حداقل رساندن اندازه "برش" قبل از اقدام باید با دقت فهمیده شود. به عنوان مثال ، اگر تازه به Facebook پیوسته اید ، هنوز به هیچ دوستی ندارید ما نمی خواهیم گراف دوستان را با شما در یک گروه و سایر نقاط جهان در گروه دیگر قرار دهیم ، حتی اگر این گراف را بدون اینکه یال هایی وجود داشته باشد که اعضای این دو گروه را به هم وصل کند ، تقسیم می شود. این برش مطلوب نیست زیرا دو مؤلفه از نظر اندازه بیش از حد نابرابر هستند.

10.4.1 چگونه پارتیشن های خوبی بسازیم؟

با توجه به گراف ، می خواهیم گره ها را به دو مجموعه تقسیم کنیم تا برش یا مجموعه ای از لبه هایی که گره ها را در مجموعه های مختلف وصل می کنند به حداقل برسد. با این حال ، ما همچنین می خواهیم انتخاب برش را محدود کنیم تا دو مجموعه از نظر اندازه تقریباً برابر باشند. مثال بعدی این نکته را نشان می دهد.

مثال 10.14 : مثال فعلی از گراف موجود در شکل 10.1 را به یاد بیاورید. در آنجا آشکار است که بهترین پارتیشن {A,B,C} را در یک مجموعه و {D,E,F,G} در قسمت دیگر قرار می دهد. این برش فقط از لبه (D ،B) تشکیل شده و از اندازه 1 است. هیچ برش غیرمستقیم نمی تواند کوچکتر باشد.



شکل 10.11 : کوچکترین برش ممکن است بهترین برش نباشد

در شکل 10.11 نوعی از مثال ما است که در آن گره H و دو لبه اضافی ، (C ،H) و (G ،C) اضافه کرده ایم. اگر اندازه برش را به حداقل برسانیم ، بهترین انتخاب این است که H را در یک مجموعه قرار دهیم و همه گره های دیگر را در مجموعه دیگر قرار می دهیم. اما باید آشکار باشد که اگر پارتیشن هایی را که یک مجموعه خیلی کوچک است رد کنیم ، بهترین کار ممکن است استفاده از برش متشکل از لبه ها (D ،B) و (G ،C) باشد که نمودار را به دو قسمت مساوی تقسیم می کند. مجموعه های اندازه {A,B,C,H} و {D,E,F,G}.

10.4.2 برش های عادی

تعریف مناسب یک برش "خوب" این است که باید اندازه برش را در برابر تغییر اندازه های مجموعه هایی که برش ایجاد می کند ، متعادل کند. یکی از گزینه های خوب "برش عادی" است. ابتدا ، حجم یک گره S را مشخص کنید که به Vol (S) اشاره می شود ، تعداد لبه ها با حداقل یک انتهای در S باشد.

فرض کنید گره های یک گراف را به دو مجموعه جدا کننده S و T تقسیم کنیم. Cut(S,T) تعداد لبه هایی است که یک گره را در S به یک گره در T وصل می کند. سپس مقدار برش عادی شده برای S و T است.

$$\frac{Cut(S,T)}{Vol(S)} + \frac{Cut(S,T)}{Vol(T)}$$

مثال 10.15 : دوباره گراف شکل 10.11 را در نظر بگیرید. اگر S = {H} و T={A,B,C,D,E,F,G} را انتخاب کنیم ، Cut(S,T) = 1 را انتخاب می کنیم ، زیرا فقط یک لبه وجود دارد متصل به H. از طرف دیگر ، Vol (T) = 11 ، زیرا تمام لبه ها حداقل در انتهای گره T قرار دارند. بنابراین ، برش نرمال شده برای این پارتیشن 1/1 + 11/1 = 1.09 است.

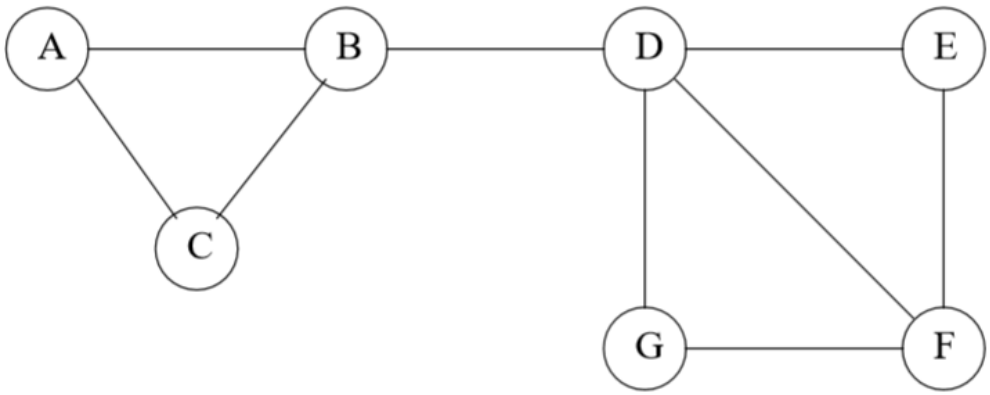
حال برش مورد نظر را برای این نمودار متشکل از لبه ها (B، D) و (C، G) در نظر بگیرید. سپس S = {A,B,C,H}، T = {D,E,F,G}.

Cut(S,T) = 2, Vol(S) = 6 و Vol(T) = 7 است.

برش نرمال شده برای این پارتیشن فقط 6/2 + 7/2 = 0.62 است.

10.4.3 برخی ماتریسهایی که نمودارها را توصیف می کنند

برای توسعه نظریه چگونگی جبر ماتریس می تواند به ما در بخش بندی خوب گراف کمک کند ، لازم نیست که در مورد سه ماتریس مختلفی که جنبه های یک نمودار را توصیف می کنند ، یاد بگیریم. اولین مورد باید آشنا باشد: ماتریس مجاور که دارای یک ردیف 1 در ردیف i و ستون j است در صورت وجود لبه بین گره های i و j و در غیر این صورت 0.



شکل 10.12 : تکرار ماتریس شکل 10.1

مثال 10.16 : گراف شکل 10.12 یک ماتریس مجاور آن را در شکل 10.13 نشان می دهد. توجه داشته باشید که ردیف ها و ستون ها به ترتیب با گره های A، B، ...، G مطابقت دارند. به عنوان مثال ، لبه (B، D) با این واقعیت بازگو می شود که ورود در ردیف 2 و ستون 4 برابر 1 است و ورودی در ردیف 4 و ستون 2 نیز وجود دارد.

دومین ماتریس مورد نیاز ما ، ماتریس درجه برای یک گراف است. این گراف فقط در مورب وارد شده است. ورودی ردیف و ستون i درجه گره ith است.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

شکل 10.13 : ماتریس مجاور برای شکل 10.12.

مثال 10.17 : ماتریس درجه برای نمودار شکل 10.12 در شکل 10.14 نشان داده شده است. ما از همان ترتیب گره ها مانند مثال 10.16 استفاده می کنیم. به عنوان مثال ، ورود در ردیف 4 و ستون 4 برابر 4 است زیرا گره D دارای لبه های چهار گره دیگر است. ورودی در ردیف 4 و ستون 5 مقدار 0 است ، زیرا آن ورودی diagonal نیست.

فرض کنید نمودار ما دارای ماتریس مجاور A و ماتریس درجه D است.

ماتریس سوم ما به نام ماتریس Laplacian است $L = D - A$ ، اختلاف بین ماتریس درجه و ماتریس مجاور است. یعنی ماتریس Laplacian L همان ورودی های D را در مورب دارد. O ff مورب ، در ردیف i و ستون j، L در صورت وجود لبه بین گره های i و j و 0 در صورت نیست.

مثال 10.18 : ماتریس Laplacian برای نمودار شکل 10.12 در شکل 10.15 نشان داده شده است. توجه کنید که هر ردیف و هر ستون به صفر می رسد ، همانطور که باید برای هر ماتریس لاپلاسی باشد

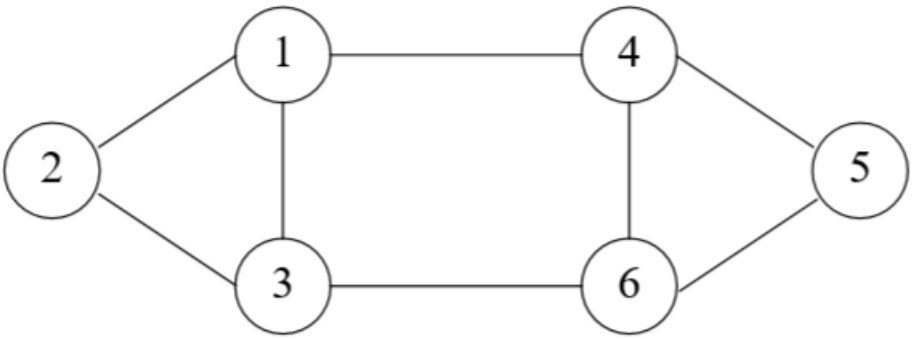
10.4.4 مقادیر ویژه از ماتریس لاپلاسین

ما می توانیم از بهترین راه برای پارتیشن بندی یک گراف از مقادیر ویژه و مقادیر ویژه ماتریس لاپلاسینی ، ایده خوبی بگیریم. در بخش 5.1.2 مشاهده کردیم که چگونه eigenvector اصلی (eigenveector همراه با بزرگترین مقررات ویژه) از ماتریس انتقال وب ، چیزی را در مورد اهمیت صفحات وب به ما گفت. در حقیقت ، در موارد ساده (بدون پرداخت جریمه) مقدار ویژه بردار PageRank است. با این حال ، هنگام برخورد با ماتریس لاپلاسین ، معلوم می شود که کوچکترین مقادیر ویژه و مقادیر ویژه اطلاعات آنها اطلاعات مورد نظر را نشان می دهند.

$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

شکل 10.15 : ماتریس لاپلاسین شکل 10.12

وکتور PageRank. با این حال ، هنگام برخورد با ماتریس Laplacian ، معلوم می شود که کوچکترین مقادیر ویژه و مقادیر ویژه اطلاعات آنها اطلاعات مورد نظر ما را فاش می کنند.



شکل 10.16 : یک گراف برای نشان دادن پارتیشن بندی با تجزیه و تحلیل طیفی

مثال 10.19 : بگذارید تکنیک فوق را در گراف شکل 10.16 اعمال کنیم. ماتریس Laplacian برای این نمودار در شکل 10.17 نشان داده شده است. با روش های استاندارد یا بسته های ریاضی می توانیم همه مقادیر ویژه و مقادیر ویژه این ماتریس را پیدا کنیم. ما به سادگی باید آنها را در شکل 10.18 ، از کم ترین مقادیر ویژه تا بیشترین را ، جدول بندی کنیم. توجه داشته باشید که ما نجات دهنده های برقی را به طول 1 تقسیم نکرده ایم ، اما در صورت تمایل می توانستیم به راحتی این کار را انجام دهیم.

$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

شکل 10.17 : ماتریس لاپلاسین شکل 10.16

eigenveector دوم دارای سه مؤلفه مثبت و سه منفی است. این باعث تعجب ناپذیر می شود که یک گروه باید {1,2,3} گره های دارای مؤلفه های مثبت و گروه دیگر {4,5,6} باشد.

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

شکل 10.18 : Eigenvalues و eigenvectors برای ماتریس شکل 10.17

10.4.5 روش های پارتیشن بندی جایگزین

روش بخش 10.4.4 پارتیشن خوبی از گراف را به دو قطعه می دهد که یک برش کوچک بین آنها وجود دارد. روش های مختلفی وجود دارد که ما می توانیم از همان بردارهای ویژه استفاده کنیم تا گزینه های خوب دیگری از پارتیشن ها را ارائه دهیم. اول اینکه ، ما مجبور نیستیم که تمام گره ها را با مؤلفه های مثبت در eigenveector در یک گروه قرار دهیم و آنهایی که دارای اجزای منفی در گروه دیگر هستند. ما می توانستیم آستانه یا ^{0} را در نقطه ای غیر از صفر تنظیم کنیم.

به عنوان مثال ، فرض کنید مثال 10.19 را اصلاح کنیم تا آستانه صفر نباشد بلکه 1.5 – باشد. سپس دو گره 4 و 6 با اجزای 1− در آژانس مشخصه دوم شکل 10.18 ، به 1 ، 2 و 3 می پیوندند و گره های منحصر به فرد را در یک جزء و تنها گره 5 در قسمت دیگر می گذارند. این پارتیشن همانند انتخاب بر اساس آستانه ی صفر ، از برش اندازه دو برخوردار است ، اما این دو مؤلفه دارای اندازه های

مختلفی متفاوت هستند ، بنابراین ما تمایل به انتخاب اصلی خود را ترجیح می دهیم. با این وجود موارد دیگری وجود دارد که آستانه صفر به اجزای نابرابر اندازه می بخشد ، اگر این مورد را در شکل 10.18 بکار بریم.

10.4.6 تمرین های بخش 10.4

تمرین 10.4.1 : برای نمودار شکل 10.9:

- a- ماتریس مجاورت
- b- ماتریس درجه
- c- ماتریس لاپلاسین

تمرین 10.4.2 : برای ماتریس لاپلاسین که در تمرین 10.4.1 (c) ساخته شده است ، اعداد دوم بزرگترین مقدمه و مجرای ویژه آن را پیدا کنید. کدام بخش از گره ها را پیشنهاد می کند؟

تمرین 10.4.3 : برای ماتریس لاپلاسین ساخته شده در تمرین 10.4.1 (c) ، سومین کوچکترین مقررات بعدی و متعاقب آن و مجرای ویژه آنها را بسازید.

۱۰.۵ یافتن مجموعه های همپوشانی

تاکنون ، ما روی خوشه بندی نمودار اجتماعی برای جوامع بعدی متمرکز شده ایم. اما جوامع در عمل به ندرت از هم پاشیده اند. در این بخش ، روشی برای گرفتن یک نمودار اجتماعی و مشخص کردن یک مدل برای آن توضیح می دهیم که به بهترین نحو توضیح می دهد که چگونه می توان با مکانیسمی تولید شد که فرض می کند احتمال اتصال دو فرد توسط یک لبه ("دوست" هستند) افزایش می یابد. زیرا آنها به عضویت جوامع بیشتر مشترک می شوند. ابزار مهمی در این تحلیل "تخمین حداکثر احتمال" است ، که ما قبل از رسیدن به موضوع اجتماعات همپوشانی توضیح خواهیم داد.

10.5.1 ماهیت جوامع

برای شروع ، بگذارید آنچه را که انتظار داریم دو جامعه با هم همپوشانی داشته باشند ، در نظر بگیریم.

داده های ما یک گراف شبکه اجتماعی است ، جایی که گره ها مردم هستند و اگر مردم "دوست" باشند بین دو گره یال وجود دارد.

بگذارید تصور کنیم این گراف نمایانگر دانش آموزان در یک مدرسه است و دو باشگاه در این مدرسه وجود دارد: کلوپ شطرنج و باشگاه اسپانیایی.

منطقی است که تصور کنیم هر یک از این باشگاه ها مانند هر باشگاه دیگری در مدرسه ، جامعه ای را تشکیل می دهند. همچنین منطقی است که تصور کنیم دو نفر در باشگاه شطرنج به احتمال زیاد دوست دارند که در گراف باشند چون یکدیگر را از این باشگاه می شناسند. به همین ترتیب ، اگر دو نفر در باشگاه اسپانیا حضور داشته باشند ، شانس خوبی وجود دارد که آنها یکدیگر را بشناسند و به احتمال زیاد دوست هستند.

اگر دو نفر در هر دو باشگاه باشند ، چه می شود؟

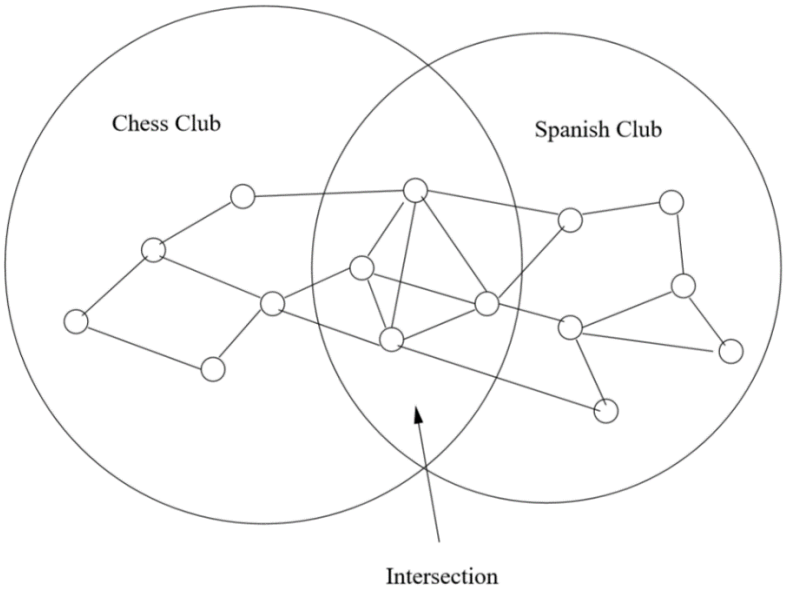
آنها اکنون دو دلیل دارند که ممکن است یکدیگر را بشناسند ، بنابراین انتظار ما حتی بیشتر از این است که آنها در گراف شبکه اجتماعی دوست باشند. نتیجه گیری ما این است که ما انتظار داریم که لبه ها در هر اجتماع متراکم شوند ، اما انتظار داریم که در تقاطع دو جامعه لبه ها حتی متراکم تر ، متراکم تر از آن در تقاطع سه اجتماع و غیره باشد. ایده توسط شکل 10.19 پیشنهاد شده است.

10.5.2 تخمین حداکثر احتمال

قبل از دیدن این الگوریتم برای مجموعه های در حال چاپ که همپوشانی با نوع پیشنهادی در بخش 10.5.1 را دارند ، اجازه دهید ابزار مدل سازی مفیدی به نام تخمین حداکثر احتمال یا MLE را یاد بگیریم و یاد بگیریم.

ایده اصلی در مورد MLE این است که فرضیه تولیدی (مدل) را ایجاد می کنیم که نمونه هایی از مصنوعات ، به عنوان مثال ، "گراف دوستان" را ایجاد می کند. این مدل دارای پارامترهایی است که احتمال ایجاد هر نمونه خاص از مصنوعات را تعیین می کند. این احتمال به احتمال مقادیر پارامتر گفته می شود. فرض می کنیم که مقدار پارامترهایی که بیشترین مقدار احتمال را به دست می آورند ، مدل صحیحی برای مصنوع مشاهده شده است.

یک مثال باید اصل MLE را روشن کند. به عنوان مثال ، ما ممکن است مایل به ایجاد گراف های تصادفی باشیم. تصور می کنیم که هر لبه با احتمال p وجود دارد و با احتمال $1 - p$ وجود ندارد ، با حضور یا عدم حضور هر لبه به طور مستقل انتخاب شده است. تنها پارامتری که می توانیم تنظیم کنیم p است. برای هر مقدار از p احتمال احتمال کمی وجود دارد که نمودار تولید شده دقیقاً همان عکسی باشد که می بینیم. با رعایت اصل MLE ، ما اعلام خواهیم کرد که مقدار واقعی p ، مقداری است که احتمال تولید نمودار مشاهده شده بالاترین آن است.



شکل 10.19: همپوشانی دو جامعه متراکم تر از قسمت های غیر همپوشانی این جوامع است

هر مقدار از p احتمال کمی دارد اما غیرر از اینکه گراف تولید شده دقیقاً همان رقمی باشد که می بینیم. با رعایت اصل MLE ، ما اعلام خواهیم کرد که مقدار واقعی p ، مقداری است که احتمال تولید نمودار مشاهده شده بالاترین آن است.

احتمالات قبلی

وقتی ما یک تجزیه و تحلیل MLE انجام می دهیم ، به طور کلی فرض می کنیم که پارامترها می توانند هر مقداری را در دامنه ی خود داشته باشند و هیچ تعصبی به نفع مقادیر خاص وجود ندارد با این حال ، اگر اینگونه نباشد ، می توانیم فرمولی را که احتمال تولید مصنوع مشاهده شده به دست می آید ، به عنوان تابعی از مقادیر پارامتر ، با عملکردی که نشان دهنده احتمال نسبی آن مقادیر پارامتر است ، ضرب کنیم. ارزشهای واقعی این تمرین ها نمونه هایی از MLE را با فرضیه هایی درباره توزیع قبلی پارامترها ارائه می دهند.

10.5.3 مدل نمودار نمودار

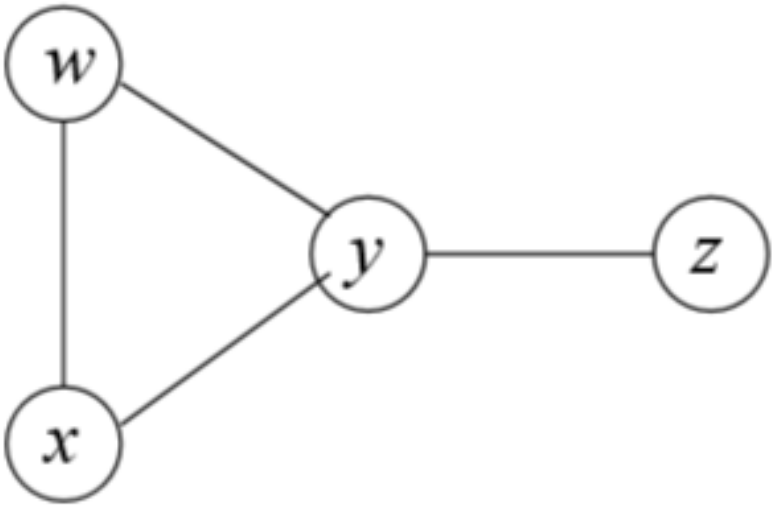
حال ما می توانیم مکانیزم معقول ، به نام یک الگوی نمودار-گراف را برای تولید گراف های اجتماعی از جوامع معرفی کنیم.

وقتی می بینیم که چگونه پارامترهای مدل احتمال دیدن یک گراف معین را افزایش می دهد ، می توانیم چگونگی حل یک پارامترهای که حداکثر احتمال را دارند را بررسی کنیم. این کار در گراف ها با نام community-affiliation معرفی شده است.

1. تعداد مشخصی از اجتماعات وجود دارد ، و تعداد معینی از افراد (گره های گراف) وجود دارد.
2. هر جامعه می تواند مجموعه ای از افراد را به عنوان عضو داشته باشد. یعنی عضویت در جوامع پارامترهای مدل است.
3. هر جامعه C دارای یک pC احتمال در ارتباط با آن است ، احتمال اینکه دو عضو جامعه C از یک لبه به هم متصل شوند زیرا هر دو عضو C هستند. این احتمالات همچنین پارامترهای مدل هستند.
4. اگر یک جفت گره در دو یا چند جامعه قرار داشته باشد ، در صورت وجود هر یک از جوامع که هر دو عضو هستند ، طبق قاعده (3) ، لبه ای بین آنها وجود دارد.

$$p_{uv} = 1 - \prod_{C\ in\ M} (1 - Pc)$$

$$\prod_{(u,v) in\ E} P_{uv} \prod_{(u,v) not\ in\ E} (1 - P_{uv})$$



شکل 10:20 : یک نمونه گراف اجتماعی

احتمال ورود

معمولاً ، لگاریتم عملکرد احتمال (احتمال ورود) را محاسبه می کنیم ، نه این که خود عملکرد باشد. انجام این کار چندین مزیت دارد. محصولات تبدیل به هزینه هایی می شوند که غالباً بیان آن را ساده می کنند. همچنین ، جمع کردن بسیاری از اعداد در مقایسه با گرفتن محصول بسیاری از اعداد کوچک ، مستعد خطاهای گرد شدن عددی کمتر است.

عملکرد ، همانطور که برای فرود شیب مورد نیاز است. تنها راه ممکن برای جستجوی فضای تکالیف احتمالی اعضا به جوامع ، شروع کار با انجام تکالیف و ایجاد تغییرات کوچک است ، می گویند درج یا حذف یک عضو برای یک جامعه.

برای هر یک از اینگونه وظایف ، می توانیم با احتساب شیب ، بهترین احتمالات جامعه (pC) را حل کنیم. با این وجود ، دانستن اینکه چه تغییراتی در عضویت باعث می شود ما را در مسیر صحیح سوق دهند ، دشوار است و هیچ تضمینی وجود ندارد که حتی بتوانید با انجام تغییرات افزایشی از یک تکلیف شروع به بهترین انتساب برسید.

۱۰,۵,۴ جلوگیری از استفاده از تغییرات گسسته

یک راه حل برای مشکل ایجاد شده توسط مکانیسم بخش ۱۰,۵,۳ وجود دارد که در آن عضویت افراد در جوامع گسسته است یعنی یا عضو جامعه هستید یا نه. ما می توانیم به "قدرت عضویت" افراد در جوامع فکر کنیم. به طور شهودی ، هرچه عضویت دو فرد در همان جامعه قوی تر باشد ، احتمال بیشتری وجود دارد که این اجتماع باعث شود که بین آنها اختلاف نظر وجود داشته باشد. در این مدل ، ما می توانیم قدرت عضویت برای یک فرد در یک جامعه را بطور مداوم تنظیم کنیم ، درست همانطور که می توانیم احتمال مربوط به یک جامعه را در الگوی وابسته سازی تنظیم کنیم. این پیشرفت به ما امکان می دهد تا از روشهای استاندارد مانند Gradient descent استفاده کنیم تا حداکثر بیان را برای احتمال به حداکثر برساند. در مدل بهبود یافته داریم:
مجموعه های ثابت انجمن ها و افراد، مانند گذشته.

برای هر جامه C و فرد x یک پارامتر قدرت عضویت به اسم FxC وجود دارد. این پارامتر می تواند هر مقدار غیر منفی را به خود اختصاص دهد و مقدار صفر هم به این معنی است که این فرد قطعا به جامعه مورد نظر تعلق ندارد.
احتمال اینکه جامعه C باعث وجود یال ای بین گره های u و v باشد برابر است با:

$$p_C(u,v) = 1 - e^{-F_{uC}F_{uC}}$$

مانند قبل، احتمال وجود یک یال بین u و v برابر است با ۱ منهای احتمال هیچکدام ار جوامعی که باعث به وجود آمدن یال بین آنها نمی شود. یعنی هر جامعه به طور مستقل باعث ایجاد یال میشود، و هر یال بین دو گره در اثر یک جامعه است. به طور رسمی تر، p_{uv} احتمال یک یال بین گره های u و v است که می‌توان به این صورت محاسبه کرد:

$$p_{uv} = 1 - \prod_C (1 - p_C(u,v))$$

حال با فرض فرمول $p_C = (u,v)$ و جایگذاری آن خواهیم داشت

$$p_{uv} = 1 - e^{\sum_C F_{uC}F_{vC}}$$

در نهایت، بگذارید E مجموعه یال ها در گراف مشاهده شده باشد. مانند گذشته ، می توانیم فرمول احتمال گراف مشاهده شده را به عنوان محصول عبارت p_{uv} برای هر یال $(u\ ،\ v)$ که در E قرار دارد ، بنویسیم ، بار محصول $1 - p_{uv}$ را برای هر یال $(u\ ،\ v)$ که در E نیست. بنابراین ، در مدل جدید فرمول احتمال گراف با یال های E به صورت زیر است

$$\prod_{(u,v)\ in\ E} (1 - e^{\sum_C F_{uC}F_{vC}}) \prod_{(u,v)\ not\ in\ E} e^{-\sum_C F_{uC}F_{vC}}$$

ما می توانیم با در نظر گرفتن لگاریتم آن ، این عبارت را تا حدودی ساده کنیم. به یاد داشته باشید که بیشینه کردن یک عملکرد، لگاریتم آن عملکرد را به نیز بیشینه خواهد کرد و بالعکس. بنابراین می توانیم از لگاریتم طبیعی عبارت فوق استفاده کنیم تا پای را با سیگما جایگزین کنیم. ما همچنین از این واقعیت ساده استفاده میکنیم $\log(e^x) = x$

$$\sum_{(u,v)\ in\ E} \log(1 - e^{-\sum_C F_{uC}F_{vC}}) - \sum_{(u,v)\ not\ in\ E} \sum_{\text{C}} F_{uC}F_{vC}$$

اکنون می توانیم مقادیر مربوط به FxC را پیدا کنیم که عبارت (10.1) را به حداکثر می رساند. یک راه این است که از Gradient descent به روشی مشابه آنچه در بخش 9.4.5 انجام شد استفاده شود. یعنی ، یک گره x را انتخاب می کنیم ، و تمام مقادیر FxC را در مسیری که عبارت (10.1) را بیشتر بهبود می بخشد تنظیم می کنیم. توجه کنید که تنها عواملی که مقادیر تغییر آنها در پاسخ به تغییرات در FxC است ، یکی u و v مربوط به x و دیگری u و v مربوط به گره مجاور x. از آنجا که درجه گره معمولاً بسیار کمتر از تعداد یال های موجود در گراف است ، می توانیم از نگاه کردن به اکثر اصطلاحات در (10.1) در هر مرحله خودداری کنیم.

۱۰,۵,۵ تمرین‌های بخش

تمرین ۱۰,۵,۱: فرض کنید گراف‌ها با انتخاب احتمال p و انتخاب هر یال به طور مستقل با احتمال p ، مانند مثال 10.21 ایجاد می شوند. برای گراف شکل 10.20 ، چه مقدار p حداکثر احتمال دیدن آن گراف را نشان می دهد؟ احتمال ایجاد این گراف چقدر است؟
تمرین ۱۰,۵,۲: MLE را برای گراف در مثال 10.22 برای حدسهای زیر از عضویتهای دو انجمن محاسبه کنید.
$(a)\ C = \{w,x\}; C = \{y,z\}$
$(b)\ C = \{w,x,y,z\}; C = \{x,y,z\}$
تمرین ۱۰,۵,۳: فرض کنید که یک سکه داریم، که یک سکه منصفانه نیست. و چندین بار آن را می‌اندازیم، h را به عنوان رو و t را به عنوان زیر در نظر بگیرید.

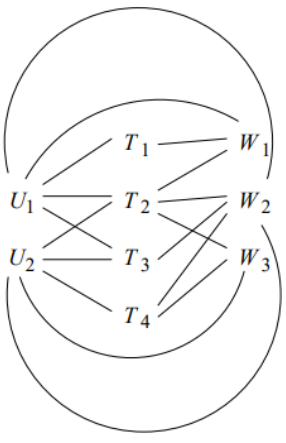
الف) اگر احتمال p برای آمدن رو در هر پرتاب p باشد، MLE برای p از نظر h و t چند است؟
ب) فرض کنید به ما گفته شده است که احتمالا ۹۰٪ سکه منصفانه است (یعنی $p = 0.5$) و ۱۰٪ شانس برای $p = 0.1$. برای کدام مقادیر h و v سکه عادلانه است؟
پ) فرض کنید a-priori likelihood برای p دارای مقادیر خاصی متناسب با $|p - 0.5|$ است. یعنی p بیشتر نزدیک به 0 یا 1 است تا ½. اگر h را رو و t رو زیر در نظر بگیریم تخمین بیشینه likelihood چقدر است؟

۱۰,۶ سیمرانک

در این بخش ، رویکرد دیگری را برای تحلیل گراف‌های شبکه های اجتماعی در نظر خواهیم گرفت. این تکنیک، simrank نامیده می شود ، که بهتر است برای گراف‌هایی که دارای چندین نوع گره هستند ، استفاده شود ، اگرچه در اصل می تواند برای هر گراف اعمال شود. هدف از simrank اندازه گیری شباهت بین گره هایی از همان نوع است. از آنجا که محاسبه باید برای هر گره شروع یک بار انجام شود ، در اندازه گراف‌هایی محدود است که می توان با این روش کاملاً آنالیز کرد.

۱۰,۶,۱ واکرهای تصادفی در گراف اجتماعی

دیدگاه ما درباره PageRank را در بخش 5.1 به یاد بیاورید که نشان می دهد اگر آنها در گراف وب راه می رفتند ، چه چیزی "گشت و گذار تصادفی" را انجام می داد. به طور مشابه می توانیم به شخصی که در حال قدم زدن در شبکه اجتماعی است فکر کنیم. گراف یک شبکه اجتماعی به طور کلی غیر جهت‌دار است ، در حالی که گراف وب جهت‌دار شده است. با این حال ، تفاوت چندان مهم نیست. یک واکر در یک گره N از یک گراف غیرجهت‌دار با احتمال برابر به هریک از همسایگان N حرکت می کند (آن گره هایی که N با آنها یال دارد).
مثلاً فرض کنید چنین واکر از گره $T1$ شکل 10.2 شروع می شود ، که ما در اینجا مثل شکل 10.21 تکثیر می کنیم. در مرحله اول ، به $U1$ یا $W1$ می رود. اگر به $W1$ برسیم ، در مرحله بعدی یا به $T1$ باز می گردد یا به $T2$ می رود. اگر واکر ابتدا به $U1$ حرکت کند ، در $T1$ ، $T2$ یا $T3$ بعدی تمام می‌شود.
نتیجه می گیریم که ، با شروع از $T1$ ، احتمال خوبی وجود دارد که واکر حداقل در ابتدا به $T2$ مراجعه کند ، و این شانس بهتر از شانس بازدید $T3$ یا $T4$ است. جالب است اگر می توانیم استنباط کنیم که برچسب های $T1$ و $T2$ به همین دلیل به یکدیگر مرتبط یا شبیه هستند. شواهد نشان می دهد که هر دو در یک صفحه وب مشترک ، $W1$ قرار داده شده اند ، و همچنین از یک برچسب مشترک ، $U1$ استفاده شده است.



شکل 10.21: تکرار شکل 10.2

اما اگر اجازه دهیم که واکر بتواند به طور تصادفی به ادامه گراف حرکت کند ، احتمال اینکه واکر در هر گره خاصی قرار داشته باشد بستگی به مکان شروع آن ندارد. این نتیجه گیری از نظریه فرآیندهای مارکوف که در بخش 5.1.2 به آن اشاره کردیم ناشی می شود ، اگرچه استقلال از نقطه شروع علاوه بر اتصال، به شرایط دیگری نیاز دارد که گراف شکل 10.21 را برآورده می کند.

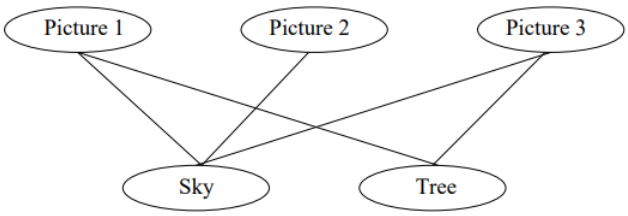
۱۰,۶,۲ واکرهای تصادفی با بازنشانی

ما از مشاهدات فوق می بینیم که با نگاه به توزیع محدود واکر ، نمی توان شباهت به یک گره خاص را اندازه گیری کرد. با این حال ، ما قبلاً نیز در بخش 5.1.5 دیدیم ، معرفی یک احتمال کوچک که واکر به صورت تصادفی متوقف می‌شود. بعداً ، در بخش 5.3.2 دیدیم که دلایلی وجود دارد که فقط یک زیر مجموعه از صفحات وب را به عنوان مجموعه teleport انتخاب کنید ، صفحاتی که Walker هنگام متوقف کردن گشت و گذار در وب به طور تصادفی به آنها می رود.

از آنجا که ما روی یک گره خاص N از یک شبکه اجتماعی متمرکز شده ایم و می خواهیم ببینیم که واکر تصادفی در گشت و گذار کوتاه از آن گره چطور پایان می‌یابد، ما ماتریس احتمال انتقال را اصلاح می کنیم تا احتمال اضافی کوچکی از انتقال به N از هر طرف داشته باشیم.

بطور رسمی ، اجازه دهید M ماتریس انتقال گراف G باشد. یعنی ورود به ردیف i و ستون j از M برابر با $1/k$ است اگر گره j از G دارای درجه k باشد ، و یکی از گره های مجاور i است. در غیر اینصورت این ورودی برابر با 0 است. اجازه دهید یک مثال از ماتیرس انتقال را بررسی کنیم.

مثال ۱۰,۲۳: شکل 10.22 نمونه ای از شبکه بسیار ساده است که شامل سه تصویر و دو برچسب "Sky" و "Tree" است که در بعضی از آنها قرار داده شده است. تصاویر 1 و 3 دارای هر دو برچسب هستند ، در حالی که تصویر 2 فقط برچسب "Sky" دارد. به طور شهودی ، انتظار داریم که تصویر 3 به تصویر 1 شباهت بیشتری نسبت به تصویر 2 داشته باشد ، و تجزیه و تحلیل با استفاده از یک واکر تصادفی با راه اندازی مجدد در تصویر 1 ، این شهود را پشتیبانی می کند.



شکل 10.23: یک گراف اجتماعی دو جانبه ساده

بگذارید گره ها را به عنوان تصویر 1 ، تصویر 2 ، تصویر 3 ، آسمان ، درخت مرتب کنیم. سپس ماتریس انتقال برای گراف شکل 10.22 است.

$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

به عنوان مثال ، ستون چهارم با گره "Sky" مطابقت دارد و این گره به هر یک از گره های تصویر درخت متصل می شود. از این رو دارای درجه سه است ، بنابراین ورودی های غیرزا در ستون آن باید $1/3$ باشد. گره های تصویر با سه ردیف و ستون اول مطابقت دارند ، بنابراین ورودی $1/3$ در سه ردیف اول ستون 4 ظاهر می شود. از آنجا که گره "Sky" هیچ یال ای برای خود یا گره "Tree" ندارد، ورودی در دو ردیف آخر ستون 4 برابر 0 است.

مانند گذشته، اجازه دهید از β به عنوان احتمال ادامه حرکت واکر به صورت تصادفی است، بنابراین $1-\beta$ احتمال واکر که میخواهد به گره اولیه N برگردد. اجازه دهید e^N بردار ستونی باشد که برای گره N دارای مقدار 1 است و در جاهای دیگر دارای مقدار 0 می باشد. سپس اگر v ستون برداری باشد که بازتاب احتمال واکری است در هر گره در یک دور خاص و v' احتمال واکری است در هر گره در دور بعدی، پس v'

به واسطه فرمول زیر به v وابسته است.

$$v' = \beta M v + (1-\beta)e_N$$

مثال ۱۰,۲۴: فرض کنید M ماتریس مثال ۱۰,۲۳ و $\beta = 0.8$ است. همچنین تصویر کنید گره N برای تصویر ۱ است: یعنی میخواهیم شباهت سایر تصاویر را با تصویر ۱ محاسبه کنیم، سپس معادله را برای مقدار جدید v' به دست آوریم، توزیعی که باید آن را تکرار کنیم.

$$v' = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} v + \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

از آنجا که گراف شکل 10.22 به هم متصل است، ماتریس اصلی M به صورت تصادفی بوده و ما می توانیم نتیجه بگیریم که اگر بردارد اولیه v دارای مولفه هایی باشه که جمع آنها برابر 1 شود، در این صورت اجزای v' دارای مولفه هایی است که از جمع 1 به دست می اید. می توانیم با اضافه کردن 5/1 به هر یک از ورودی های ردیف اول ماتریس ، معادله فوق را ساده کنیم. یعنی می توان ضرب بردار ماتریس را تکرار کرد

$$v' = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} v$$

اگر ما با $v = e_N$ شروع کنیم، سپس توالی تخمین ها از توزیع واکر که به دست می اید به صورت زیر است

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix}, \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix}, \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix}, \dots, \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

ما از موارد فوق مشاهده می کنیم که در حد مجاز ، واکر بیش از دو برابر در تصویر 3 احتمال دارد در تصویر 2 باشد. این تجزیه و تحلیل این شهود را تأیید می کند که تصویر 3 بیشتر شبیه به تصویر 1 است تا تصویر 2.

۱۰,۶,۳ تمرین‌های این بخش

تمرین ۱۰,۶,۱: اگر در شکل 10.22 پیمایش را از تصویر 2 شروع کنید ، شباهت تصویر 2 از دو تصویر دیگر چیست؟ به نظر شما کدام یک بیشتر شبیه به تصویر 2 است؟

تمرین ۱۰,۶,۲: اگر در شکل 10.22 پیمایش را از تصویر 3 شروع کنید ، انتظار دارید شباهت دو تصویر دیگر چگونه باشد؟

تمرین ۱۰,۶,۳: تجزیه و تحلیل مثال 10.24 را تکرار کنید ، و در صورت ایجاد تغییرات زیر در شکل 10.22 ، شباهت های تصویر 1 را با سایر تصاویر محاسبه کنید:

الف) برچسب "Tree" به تصویر ۲ اضافه شود.

ب) برچسب سومی به نام "Watter" به تصویر سوم اضافه شود.

پ) برچسب سومی به نام "Watter" به تصویر اول ول دوم اضافه شود.

توجه: تغییرات بطور مستقل برای هر قسمت انجام می شود. آنها تجمعی نیستند.

۱۰,۷ شمارش مثلث ها

یکی از مفیدترین خصوصیات گراف‌های شبکه های اجتماعی ، شمارش مثلث ها و سایر زیرگراف های ساده است. در این بخش روشهایی را برای برآورد یا گرفتن تعداد دقیق مثلثها در گراف بسیار بزرگ ارائه خواهیم داد. ما با انگیزه ای برای چنین شمارش هایی شروع می کنیم و سپس روش های مختلفی برای شمارش موثر ارائه می دهیم.

۱۰,۷,۱ چرا شمارش مثلث‌ها؟

اگر با گره های n شروع کنیم و یال های m را به طور تصادفی به یک گراف اضافه کنیم ، تعداد مثلث مورد انتظار در گراف وجود خواهد داشت. ما می توانیم این تعداد را بدون مشکل خیلی زیاد محاسبه کنیم. $\binom{n}{3}$ مجموعه از سه گره وجود دارد، یا تقریبا $n^3/6$ مجموعه از سه گره که ممکن است مثلث باشد. احتمال یک یال بین هر دو گره داده شده برابر است با $m/\binom{n}{2}$ یا به صورت تقریبی $2m/n^2$.

احتمال اینکه هر مجموعه ای از سه گره دارای یال هایی بین هر جفت باشد ، اگر این یال ها به طور مستقل انتخاب شوند یا حضور

نداشته باشند تقریباً $(2m/n^2)^3 = 8m^3/n^6$ است. بنابراین ، تعداد مثلث مورد انتظار در یک گراف از گره های n و m یال های تصادفی انتخاب شده تقریبا $(8m^3/n^6)(n^3/6) = \frac{4}{3}(m/n)\text{^}3$ است.

اگر یک گراف یک شبکه اجتماعی با n شرکت کننده ها و m جفت دوست باشد ، انتظار داریم تعداد مثلث ها بیشتر از مقدار تصادفی یک گراف است. دلیل این امر این است که اگر A و B دوست هستند و A نیز دوست C است ، باید یک شانس بسیار بیشتر از حد متوسط وجود داشته باشد که B و C نیز دوست باشند. بنابراین ، شمارش تعداد مثلث ها به ما کمک می کند تا اندازه یک گراف را مانند یک شبکه اجتماعی اندازه بگیریم.

ما همچنین می توانیم جوامع درون یک شبکه اجتماعی را بررسی کنیم. نشان داده شده است که سن یک جامعه به تراکم مثلث ها مربوط می شود. یعنی وقتی گروهی به تازگی تشکیل شده است ، افراد به سمت دوستان هم فکر خود می روند ، اما تعداد مثلث ها نسبتاً کم است. اگر A دوستان B و C را به عنوان دوست خود انتخاب کند، ممکن است که B و C همدیگر را نشناسند. با رشد جامعه ، B و C ممکن است به دلیل عضویت در جامعه تعامل داشته باشند. بنابراین ، یک فرصت خوب وجود دارد که در بعضی مواقع مثلث $\{A ، B ، C\}$ کامل شود.

۱۰,۷,۲ الگوریتمی برای یافتن مثلث‌ها

ما مطالعه خود را با الگوریتمی شروع خواهیم کرد که سریعترین زمان اجرای یک پردازنده را داشته باشد. فرض کنید ما یک گراف از گره های n و یال های $m \geq n$ داریم. برای راحتی ، فرض کنید که گره ها عدد صحیح $1,2, \dots, n$ هستند.

یک گره را heavy hitter نامگذاری می‌کنیم اگر درجه آن حداقل $\sqrt{m}$ باشد. یک مثلث heavy-hitter مثلثی است که تمام گره‌های آن heavy hitter باشند. از الگوریتم‌های جداگانه‌ای برای شمارش مثلث‌های heavy-hitter و همه مثلث‌های دیگر استفاده میکنیم.

توجه داشته باشید که تعداد گره‌های heavy hitter بیشتر از $2\sqrt{m}$ نیست، زیرا در غیر این صورت، مجموع درجه گره‌های heavy hitter بیشتر از $2m$ خواهد شد. از آنجا که هر یال در تنها دو گره شرکت دارد، پس باید بیش از m یال داشته باشد.

با فرض اینکه گراف توسط یال‌های آن توصیف می‌شود، ما گراف را به شرح زیر پردازش می کنیم.

1. درجه هر گره را محاسبه کنید. این قسمت نیاز دارد که هر یال را مورد بررسی قرار دهیم و 1 را به تعداد هر دو گره آن اضافه کنیم. کل زمان مورد نیاز برای این کار $O(m)$ است.
2. ایجاد یک فهرست در یال ها ، که جفت گره در انتهای آن به عنوان کلید قرار دارد. یعنی این شاخص به ما امکان می دهد با توجه به دو گره مشخص کنیم که آیا یال بین آنها وجود دارد یا خیر. یک جدول hash کافیست. می تواند در زمان $O(m)$ ساخته می شود، و زمان باقی مانده برای پرسش و پاسخ در مورد وجود یال صرف می شود.
3. شاخص دیگری از یال ها ایجاد کنید ، این یکی با کلید برابر با یک گره است. با توجه به گره v ، می توان گره های مجاور v را در زمان متناسب با تعداد آن گره ها بازیابی کرد.

گره ها را به شرح زیر مرتب خواهیم داد. ابتدا گره ها را بر اساس درجه مرتب می کنیم. سپس اگر u و v درجه یکسانی داشتند، به یاد داشته باشید که هر دو عدد صحیح هستند، بنابراین آنها را به صورت عددی مرتب می کنیم. در اینجا است که میگوییم $v \prec u$ اگر و تنها اگر

- درجه v کوچکتر از درجه u باشد یا
- درجه هر دو یکسان بوده و $v < u$

مثلث‌های heavy-hitter: تنها $O(\sqrt{m})$ گره heavy-hitter وجود دارد پس می‌توانیم همه مجموعه‌های این سه گره را در نظر بگیریم. تعداد $O(m^{3/2})$ ممکن مثلث heavy-hitter وجود دارد و با استفاده از شاخص یال‌ها می‌توانیم بررسی کنیم که آیا هر سه یال در زمان $O(1)$ وجود دارند یا خیر. بنابراین، زمان $O(m^{3/2})$ برای یافتن همه مثلث‌های مورد نظر لازم است.

مثلث‌های دیگر: مثلث های دیگر رو با روشی دیگر می یابیم. هر یال را به صورت (v_1, v_2) در نظر می گیریم. اگر هر دو v_1 و v_2 از نوع heavy hitters باشند، این یال را نادیده میگیریم. فرض کنید که v_1 یک heavy hitter نیست و علاوه بر این $v_1 \prec v_2$. و $u_1, u_2, \dots, u_k$ گره‌های مجاور v_1 باشند. توجه کنید که $k < \sqrt{m}$. ما می‌توانیم این گره‌ها را با استفاده از شاخص (index) گره‌ها در زمان $O(k)$ که مسلما $O(\sqrt{m})$ است بیابیم. برای هر u_i می‌توانیم از اولین شاخص برای بررسی اینکه یال (u_i, v_2) در زمان $O(1)$ وجود دارد یا نه استفاده کنیم. همچنین درجه u_i را نیز در زمان $O(1)$ به دست می‌اوریم، چرا که همه درجه‌های این گره‌های متصل به یکدیگر را داریم. ما مثلث $\{v_1, v_2, u_i\}$ را در شمارش خود حساب می‌کنیم، اگر و تنها اگر یال (u_i, v_2) وجود داشته باشد و

$v_1 \prec u_i$ باشد. به این ترتیب، مثلث فقط یک بار شمارش می‌شود – زمانی که v_1 گره مثلث است که طبق ترتیب $\prec$ قبل از دو گره دیگر مثل باشد. بنابراین زمان پردازش تمام گره‌های مجاور v_1 برابر با $O(\sqrt{m})$ است. از آنجایی که تعداد m یال وجود دارد، کل زمان شمارش مثلث های دیگر برابر با $O(m^{3/2})$ است.

اکنون می بینیم که پیش پردازش $O(m)$ زمان خواهد برد. زمان لازم برای پیدا کردن مثلث های heavy-hitter برابر با $O(m^{3/2})$ و همچنین زمان لازم برای یافتن دیگر مثلث ها، که زمان کلی الگوریتم مورد نظر برابر با $O(m^{3/2})$ است.

۱۰,۷,۳ بهینه سازی الگوریتم یافتن مثلث‌ها

معلوم است که الگوریتم توضیح داده شده در بخش 10.7.2 بهترین نتیجه قابل قبول ممکن است. برای روشن شدن بیشتر، یه گراف کامل متشکل از n گره را در نظر بگیرید. این گراف $m = \binom{n}{2}$ یال و تعداد مثلث ها برابر با $\binom{n}{3}$ است. از آنجایی که ما نمی توانیم مثلث را در زمان کمتری از تعداد مثلث ها بشماریم، می دانیم که هر الگوریتمی زمان $\Omega(n^3)$ را در این گراف به خود خواهد گرفت. با این حال $m = O(n^2)$، بنابراین هر الگوریتمی برای این گراف $\Omega(m^{3/2})$ زمان خواهد گرفت.

ممکن است کسی تعجب کند که آیا الگوریتم وجود دارد که روی گراف‌های پراکنده بهتر از گراف کامل کار کرده باشد یا خیر. با این حال، ما می توانیم زنجیره ای از گره هایی به طول n^2 را به گراف کامل اضافه کنیم. این زنجیره دیگر مثلثی اضافه نمی کند. همچنین تعداد یال ها را نیز دو برابر نمی کند، اما باعث می شود تعداد گره ها را به اندازه دلخواه ما بزرگ شود. در نتیجه نسبت یال ها به گره ها را به عدد 1 که ما دوست داریم کاهش می دهد. از انجا که هنوز $\Omega(m^{3/2})$ مثلث وجود دارد، میبینمم که که این حد پایین محدوده کاملی از نسبت های m/n را دارا می باشد.

۱۰,۷,۴ یافتن مثلث‌ها با استفاده از MapReduce

برای یک گراف بسیار بزرگ می خواهیم از موازی سازی برای سرعت بخشیدن به محاسبات استفاده کنیم. ما می توانیم مثلث را به عنوان چند راه متصل بیان کنیم و از تکنیک بخش 2.5.3 برای بهینه سازی از یک کار MapReduce منفرد برای شمارش مثلث استفاده کنیم. به نظر می رسید این یکی از مواردی است که تکنیک پیوستن چند راهی بسیار کارآمدتر از دو اتصال دو طرفه است. علاوه بر این ، کل زمان اجرای الگوریتم موازی در واقع همان زمان اجرای بر روی یک پردازنده واحد با استفاده از الگوریتم بخش 10.7.2 است.

برای شروع فرض کنید که گره‌های گراف به صورت $1, 2, \ldots, n$ شماره گذاری شده اند. ما از یک رابطه E برای نشان دادن یال‌ها استفاده می‌کنیم. برای جلوگیری از تکرار نمایش یک یال، فرض می‌کنیم که اگر $E(A, B)$ یک tuple از این رابطه باشد، نه تنها یک یال بین A و B وجود دارد، همچنین به عنوان یک عدد صحیح $A < B$ خواهد بود. این نیازمندی حلقه‌ها (یالی بین یک گره با خود) را نیز از بین می‌برد، که به هر حال ما در گراف شبکه های اجتماعی تصویر می‌کنیم وجود ندارد، اما می‌تواند به مثلثی منجر شود که سه گره متفاوت را ندارد.

با استفاده از این رابطه می‌توانیم مجموعه‌ای از مثلث‌های گراف را که یال‌های آن دارای اتصال طبیعی است بیان کنیم.

$$\mathrm{E(X, Y)} \bowtie \mathrm{E(X, Z)} \bowtie \mathrm{E(Y, Z)}$$

برای درک این اتصال، باید بفهمیم که در هر یک سه استفادی مورد نظر از E به به صفات رابطه E اسم‌های متفاوتی داده می‌شود. یعنی، تصور می‌کنیم که سه نسخه از E وجود دارد که هر کدام با tuple های یکسان اما با اسکیمای مختلف. در SQL این اتصال با استفاده از رابطه $E(A, B)$ به شرح زیر نویشته می‌شود.

```
SELECT e1.A, e1.B, e2.B
FROM E e1, E e2, E e3
WHERE e1.A = e2.A AND e1.B = e3.A AND e2.B = e3.B
```

در این query صفات معادل $e2.A$ و $e1.A$ در ویژگی ما با استافده از X نشان داده شده اند. همچنین $e3.A$ و $e1.B$ هر کدام با Y نشان داده شده اند و $e3.B$ و $e2.B$ با استفاده از Z نشان داده شده است.

توجه کنید که هر مثلث فقط یک بار در این اتصال ظاهر می شود. مثلث متشکل از گره های v_1, v_2, v_3 زمانی ساخته می شود که X, Y, Z سه گره به ترتیب عددی هستند، یعنی $X < Y < Z$. برای نمونه اگر ترتیب عدید گره ها به صورت $v_1 < v_2 < v_3$ باشد، پس X فقط میتواند v_1 باشد و ... به همین تریتب بقیه هم همینطور.

از روش بخش 2.5.3 می توان برای بهینه سازی پیوستن معادله 10.2 استفاده کرد. ایده ها را در مثال 2.9 به یاد بیاورید، جایی که تعداد روش هایی را که باید مقادیر هر ویژگی را حذف کنیم در نظر گرفتیم. در مثال حاضر ، موضوع کاملاً ساده است. سه رابطه مورد نظر مطمئنا اندازه یکسانی دارند، بنابراین با تقارن، خصوصیات X, Y, Z به تعداد یکسانی Bucket، هش خواهند شد.

به طور خاص، اگر ما گره هایی hash شده ای برای $Bucket\ b$ داشته باشیم، پس می شود، b^3 تعداد Reducer. هر Reduce Task با توالی از سه Bucket عددی X, Y, Z، که هر یک در محدوده عددی ۱ تا b قرار دارند.

وظایف نگاشت رابطه E را به بخشهایی که وظایف نگاشت وجود دارد تقسیم بندی می کند. فرض کنید یک وظیفه نگاشت به رابطه $E(A, B)$ داده می‌شود تا برخی از وظایف Reduce را ارسال کند. ابتدا (u, v) را به عنوان یک tuple برای اتصال اصطلاح $E(u, v)$ در نظر بگیرید. ما می‌توانیم با استفاده از بررسی u و v تعداد Bucket های مربوط به X و Y را به دست آوریم. اما Bucket مربوط به Z را نمی‌دانیم. بنابراین ما باید $E(A, B)$ برای همه وظایف Reducer که مطابق با توالی سه عدد Bucket $(h(u), h(v), z)$ برای مقادیر ممکن b مورد نظر Bucket z محاسبه کنیم.

در مرحله بعد ، بگذارید کل هزینه های اجرای را در تمام وظایف Reduce محاسبه کنیم. فرض کنید عملکرد هش به اندازه کافی یال ها را بطور تصادفی توزیع می کند که وظایف Reduce هر یک تقریباً به همان تعداد یال می رسد. از آنجا که تعداد یالهای توزیع شده b^3 وظایف کاهشی دارای پیچیدگی زمانی $O(mb)$ است. بنابراین نتیجه می‌گیریم که هر وظیفه تعداد $O(m/b^2)$ را دریافت می‌کند. اگر در هر وظیفه کاهشی از الگوریتم بخش 10.7.2 استفاده کنیم، پیچیدگی زمانی برای هر وظیفه برابر با $O((m/b^2)^{3/2})$ یا $O(m^{3/2}/b^3)$ خواهد بود. که هزینه کلی برابر با $O(m^{3/2})$ می‌شود. دقیقاً مانند الگوریتم تک پردازنده بخش 10.7.2.

۱۰,۷,۵ استفاده از Reduce Tasks کمتر

با یک ترتیب معقول گره ها ، ما می توانیم تعداد کارهای کاهش یافته را تقریباً با ضریب 6 کاهش دهیم. به "نام" گره i به عنوان جفت $(h(i), i)$ فکر کنید ، که در آن h تابع هش است که در بخش 10.7.4 برای هش گره‌ها به b استفاده شد. گره‌ها را با استفاده از نام مربوط به هر کدام مرتب میکنیم، تنها مولفه اول را در نظر گرفته و با استفاده از مولفه دوم پیوند بین گره‌های موجود در Bucket یکسان را می‌شکنیم. اگر از این مرتب سازی گره‌ها استفاده کنیم، سپس وظیفه کاهشی مربوط به لیست Bucket های (i, j, k) فقط زمانی مورد نیاز می‌شود که $i \leq j \leq k$ باشد. اگر b عدد بزرگی باشد، پس تقریباً $1/6$ از تمام دنباله های b^3 اعداد صحیح ، هر یک در محدوده 1 تا b ، این نابرابری ها را برآورده می کنند. برای هر b تعداد هر توالی میشود $\binom{b+2}{3}$. بنابراین نسب دقیق میشود $(b + 2)(b + 1)/(6b^2)$.

از آنجا که تعداد کاهنده های کمتری وجود دارد ، در تعداد جفت های با ارزش اصلی که باید ارتباط برقرار شود ، کاهش قابل توجهی پیدا می کنیم. به جای اینکه هر یک از یال‌های m را به b^3 وظایفه بفرستیم ، باید هر یال را فقط برای b وظیفه ارسال کنیم. برای هر یک از مقادیر b که به عنوان k می‌شناسیم که عددی بیم 1 و b، لیست تشکیل شده از i, j, k به ترتیب مرتب شده در نظر بگیرید. سپس وظیفه کاهشی که با این لیست مطابقت داشته باشید، به یال e نیاز پیدا می‌کند، در مقال بقیه وظایف کاهشی نیازی به e نخواهند داشت.

برای مقایسه هزینه ارتباط روش این بخش با نمونه بخش 10.7.4 ، اجازه دهید تعداد وظایف کاهشی را عدد ثابت k در نظر بگیریم. سپس روش بخش 10.7.4 گره ها را به ، بنابراین ارتباط برقرار می کند.

مثال 10.25: الگوریتم رو به جلو بخش 10.7.4 را با $b = 6$ در نظر بگیرید. یعنی وظایف کاهشی برابر با $b^3 = 216$ و هزینه ارتباط برابر با $3mb = 18m$. ما نمیتوانیم دقیقا از 216 وظیفه کاهشی با روش این بخش استفاده کنیم، ولی اگر $b = 10$ در نظر بگیریم، می‌توانیم به این عدد نزدیک شویم. سپس تعداد وظایف کاهشی برابر با $\binom{12}{3} = 216$ می‌شود. و هزینه ارتباط $mb = 10m$ خواهد بود. یعنی هزینه ارتباط 5/8 هزینه روش رو به جلو است.

۱۰,۷,۶ تمرین‌های این بخش

تمرین 10.7.1: چه تعداد مثلث در شکل‌های زیر وجود دارد.

الف) شکل 10.1

ب) شکل 10.9

پ) شکل 10.2

تمرین10.7.2: برای هر یک از گراف‌های تمرین 10.7.1 تعیین کنید:

الف) حداقل درجه برای گره‌ای که یک heavy hitter محسوب می‌شود چیست؟

ب) کدام گره‌ها Heavy hitter هستند؟

پ) کدام مثل‌ها از نوع مثلث heavy-hitter هستند؟

تمرین 10.7.3: در این تمرین مشکل پیدا کردن مربع ها را در یک گراف در نظر می گیریم. یعنی می خواهیم چهار شکاف گره a، b، c، d را به گونه ای پیدا کنیم که چهار یال $(a، d)$ و $(a، b)$، $(b، c)$، $(c، d)$ در گراف وجود داشته باشند. فرض کنید گراف همانطور که در بخش 10.7.4 وجود دارد با یک رابطه E نشان داده شده است. نوشتن یک پیوند مجزا از چهار نسخه E امکان پذیر نیست که بیانگر تمام مربعهای ممکن در گراف است ، اما می توانیم سه عدد اتصال را بنویسیم. علاوه بر این ، در برخی موارد ، ما باید انتخابی را دنبال کنیم که "مربع" را از بین ببرد ، در حالی که یک جفت گوشه مخالف واقعاً همان گره هستند. می توانیم فرض کنیم که گره a از نظر عددی پایین تر از همسایگانش b و d است ، در حالی که c وابسته به موارد زیر است که c

- کمتر از b و d است
- بین b و d است، یا
- بزرگتر از هر دوی b و d است.

الف) اتصالات طبیعی را بنویسید که مربع هایی را ایجاد می کند که هریک از سه شرط فوق را براورده سازد. شما می توانید از چهار ویژگی مختلف W، X، Y و Z استفاده کنید و فرض کنید که چهار نسخه از رابطه E با طرح های مختلف وجود دارد ، بنابراین اتصالات را می توان به عنوان اتصالات طبیعی بیان کرد.

ب) برای کدام‌یک از این اتصالات نیاز به یک انتخاب برای اطمینان از اینکه گوشه های مقابل گره ها واقعا متفاوت هستند؟

پ) فرض کنید ما قصد داریم از k وظیفه کاهشی را استفاده کنیم. برای هر اتصالات از (a)، به چه تعداد buckets برای هش کردن هر کدام از W, X, Y, X جهت بهینه کردن هزینه ارتباط لازم است.

ج) بر خلاف مورد مثلث ، تضمین نمی شود که هر مربع فقط یک بار تولید شود ، اگرچه می توانیم اطمینان داشته باشیم که هر مربع تنها با یکی از سه اتصال تولید می شود. به عنوان مثال ، مربعی که در آن دو گره در گوشه های مقابل هر کدام از نظر عددی پایین تر از هر دو گره دیگر قرار دارند فقط توسط پیوند (i) تولید می شوند. برای هر یک از سه عضو ، چند بار مربعی که اصلاً تولید می کند ، تولید می کند؟

تمرین 10.7.4: نشان دهید که تعداد توالی اعداد صحیح $1 \leq i \leq j \leq k \leq b$ برابر با $\binom{b+2}{3}$.

نکته: نشان دهید که این توالی ها می توانند در یک مکاتبات 1 به 1 با رشته های باینری به طول $b + 2$ که دقیقاً سه مورد 1 دارند ، قرار بگیرند.

۱۰,۸ ویژگی های همسایگی گراف ها

چندین ویژگی مهم گراف وجود دارد که مربوط به تعداد گره هایی است که می توانید از یک گره مشخص در طی یک مسیر کوتاه به آنها برسید. در این بخش به الگوریتم هایی برای حل مشکلات مربوط به مسیرها و محلات برای گراف‌های بسیار بزرگ می پردازیم. در بعضی موارد ، راه حلهای دقیق برای گراف‌هایی با میلیون ها گره امکان پذیر نیست. بنابراین ما به الگوریتم های تقریبی و همچنین الگوریتم های دقیق نگاه می کنیم.

۱۰,۸,۱ گراف‌های جهت‌دار و همسایگان

در این بخش از یک گراف جهت‌دار به عنوان الگوی شبکه استفاده خواهیم کرد. گراف جهت‌دار دارای مجموعه ای از گره ها و مجموعه ای از کمان ها است. دومی یک جفت گره است که به صورت $u \to v$ نوشته شده است. ما u به عنوان منبع و v هدف کمان را تو می نامیم. گفته می شود قوس از u تا v است.

انواع مختلفی از گراف‌ها را می توان با استفاده از گراف‌های جهت‌دار مدل نمود. وب یک نمونه اصلی است ، جایی که قوس $u \to v$ پیوندی از صفحه u به صفحه v است. یا قوس $u \to v$ ممکت است به این معنی باشد که مشترک u، مشترک v را در ماه گذشته صدا

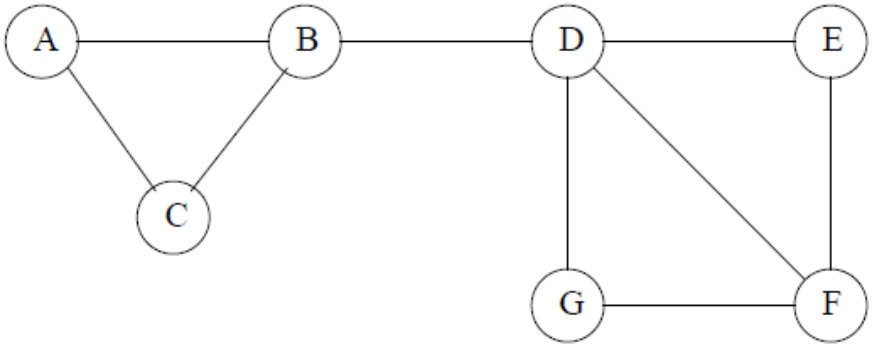
زده است. یک مثال دیگر قوس می‌تواند به معنی این باشد که فرد u، فردv را در توییتر دنبال می‌کند. در گراف دیگری ، قوس می تواند به این معنی باشد که مقاله تحقیق شماا به مقاله‌ای دیگر ارجاع می دهد.

علاوه بر این ، تمام گراف‌های بدون جهت را می توان با گراف‌های جهت دار نشان داد. به جای یال بدون جهت $(u، v)$ از دو قوس $u \rightarrow v$ و $v \rightarrow u$ استفاده کنید. بنابراین ، محتوای این بخش همچنین در مورد گراف‌هایی است که ذاتاً بدون جهت هستند، مانند گراف دوستان در یک شبکه اجتماعی کاربرد دارد.

یک مسیر در یک گراف جهت دار دنباله ای از گره های $v_0, v_1, \dots, v_k$ است، به گونه‌ای که قوس‌های $v_i \rightarrow v_{i+1}$ برای همه $i = 0, 1, \dots m\ k-1$ میباشد. طول این مسیر برابر با k، تعداد قوس ها در امتداد مسیر است. توجه داشته باشید که در یک مسیر به طول k تعداد $k+1$ گره وجود دارد و یک گره به خودی خود مسیری به طول 0 در نظر گرفته می شود.

همسایگی شعاع d برای یک گره v مجموعه‌ای از گره‌های u است که برای آنها یک مسیر به طول d از v به u وجود دارد. این همسایگی را به $N(v, d)$ مشخص می‌کنیم. برای مثال $N(v, 0)$ همیشه $\{v\}$ و $N(v, 1)$ جمع v با مجموعه گره‌هایی است که یک قوس از v دارند. به طور کلی، اگر v مجموعه‌ای از گره‌ها باشد، پس $N(v, d)$ مجموعه‌ای از گره‌های u است که حداقل یک مسیری به طول d و یا کمتر با مجموعه v دارد.

مشخصات همسایه‌های یک گره v، توالی از اندازه‌های همایشگان‌اش است $|N(v, 1)|, |N(v, 2)|, \dots$. ما همسایه‌ای با فاصله 0 را شامل نمی‌شویم، چرا که اندازه آن همیشه 1 است.



شکل 10.23: شبکه اجتماعی کوچک ما،به عنوان یک گراف جهت‌دار در نظر بگیرید

مثال 10.26: گراف بدون جهت شکل 10.1 را در نظر بگیرید که ما در اینجا به عنوان شکل 10.23 دوباره تولید می کنیم. برای تبدیل آن به یک گراف جهت دار، به هر لبه به عنوان یک جفت قوس فکر کنید ، یکی در هر جهت. به عنوان مثال ، لبه $(A ، B)$ به قوس A → B و B → A تبدیل می شود. ابتدا همسایه های گره A را در نظر بگیرید. ما می‌دانیم $N(A, 0) = \{A\}$ است. علاوه بر این ، $N(A ، 1) = \{A ، B ، C\}$ ، قوس هایی از A فقط تا B و C وجود دارد. علاوه بر این $N(A, 2) = \{A, B, C, D\}$ و $N(A, 3) = \{A, B, C, D, E, F, G\}$. همسایگی های شعاع بزرگتر مانند $N (A ، 3)$ است.

از طرف دیگر گره B را در نظر بگیرید. $N(B, 0) = \{B\}$، $N(B, 1) = \{A, B, C, D\}$ و $N(B, 2) = \{A, B, C, D, E, F, G\}$ را می‌یابیم. ما میدانیم که B به نسب A به مرکز نزدیک تر است و این واقعیت بازتابی است از مشخصات همسایه های دو گره. گره A دارای مشخصات 3 ، 4 ، 7 ، 7 ، است، در حالی که B دارای مشخصات 4 ، 7 ، 7 ، است. بدیهی است ، B مرکزی تر است ، زیرا در هر مسافت ، همسایگی آن حداقل به اندازه A است. در واقع ، D حتی از B نیز مرکزی تر است ، زیرا مشخصات محله آن 5 ، 7 ، 7 ، است.

۱۰،۸،۲ قطر یک گراف

قطر یک گراف جهت دار کوچکترین عدد صحیح d است به گونه ای که برای هر دو گره u و v مسیری به طول d یا کمتر، از u تا v وجود دارد. در یک گراف جهت دار ، این تعریف تنها در صورتی معنی دارد که گراف به شدت به هم وصل شود. بحث ما درباره وب را در بخش 5.1.3 به یاد بیاورید ، در آنجا مشاهده کردیم که یک زیر مجموعه بزرگ به شدت متصل به وب در "مرکز" وجود دارد ، اما این که وب به طور کلی، کاملاً به هم وصل نشده است. در عوض ، برخی از صفحات وجود دارد که به وسیله پیوندها به هیچ جایی نمی‌روند ، و بعضی از صفحات که دستیابی آنها توسط لینک ها امکان پذیر نیست.

اگر گراف بدون جهت باشد ، تعریف قطر همان گراف‌های جهت‌دار است ، اما مسیر ممکن است یال‌های بدون جهت را از هر جهت طی کند. یعنی ما با یک لبه غیر جهت دار به عنوان یک جفت قوس رفتار می کنیم ، یکی در هر جهت. مفهوم قطر تا زمانی که آن گراف به هم متصل شود ، در یک گراف غیر جهت دار معنا پیدا می کند.

مثال 10.27: برای نمودار شکل 10.23 قطر 3 است. بعضی جفت گره ها مانند A و E وجود دارند که هیچ مسیر کمتر از طول 3 بین آنها نیست. اما هر جفت گره مسیری را از یکی به دیگری با طول حداکثر 3 داریم.

می توان با محاسبه اندازه همسایه های آن شعاع در حال افزایش ، قطر یک گراف را محاسبه کرد ، تا زمانی که در برخی شعاع ها نتوانیم گره های دیگری اضافه کنیم.

شش درجه جداسازی

یک بازی معروف به نام "شش درجه کوین بیکن" وجود دارد ، هدف آن پیدا کردن مسیری به طول حداکثر شش در گراف است که گره های آنها ستاره های فیلم هستند و یال های آنها ستاره هایی را که در همان فیلم بازی کرده اند به هم متصل می کند. حدس این است که در این نمودار ، هیچ ستاره فیلمی از کوین بیکن بیش از شش فاصله ندارد. به طور کلی ، هر دو ستاره فیلم حداکثر با مسیری به طول شش می توانند ارتباط برقرار کنند. یعنی قطر نمودار شش است. قطر کوچک محاسبه همسایگی را کارآمدتر می کند ، بنابراین خوب خواهد بود اگر تمام گراف های شبکه اجتماعی قطر کوچک مشابهی را به نمایش بگذارند. در حقیقت ، عبارت "شش درجه جدایی" ، به این فرضیه اطلاق می شود که در شبکه همه افراد جهان ، جایی که یال به معنای این است که این دو نفر یکدیگر را می شناسند ، قطر آن شش است. متأسفانه ، همانطور که در بخش 10.8.3 بحث خواهیم کرد ، همه نمودارهای مهم چنین اتصالات محکم را نشان نمی دهند.

۱۰,۸,۳ بستار تعدی و دسترس‌پذیری

بستار تعدی ، مجموعه ای از جفت گره ها $(u ، v)$ است به گونه ای که مسیری از u تا v به طول صفر یا بیشتر وجود داشته باشد. بعضی اوقات ما باید این مورد را به صورت $Path(u, v)$ بنویسیم. یک مفهوم مرتبط با آن دسترس‌پذیری است. ما می‌گوییم در صورت $Path(u, v)$ گره u به گره v می‌رسد. مسئله دسترس‌پذیری این است که با توجه به یک گره u در گراف ، تمام v را به گونه ای پیدا کنید که $Path(u, v)$ صحیح باشد.

این دو مفهوم مربوط به مفهوم همسایگی است که قبلاً دیده ایم. در حقیقت، $Path(u, v)$ اگر و تنها اگر v در $N(u, \infty)$ باشد. که به صورت $\bigcup_{i\geq 0} N(u, i)$ تعریف می‌کنیم. بنابراین مسئله دسترسپذیری محاسبه اتحاد همه همسایگی‌ها یک گره در یک شعاع معین مانند u است. بحث در بخش 10.8.2 به ما یادآوری می‌کند که می‌توانیم مجموعه دسترس‌پذیر برای u را با محاسبه همسایه‌های کوچکتر از شعاع d را محاسبه کنیم. $N(u, d) = N(u, d + 1)$.

این دو مسئله – بستار تعدی و دسترس‌پذیری – به یکدیگر مربوط هستند ، اما نمونه های زیادی از گراف‌هایی وجود دارد که بستار تعدی به آنها امکان پذیر است و دسترس‌پذیری نیست.برای نمونه تصور کنید که ما یک گراف وب متشکل از یک میلیارد گره را داریم. اگر می خواهیم صفحات (گره ها) را از یک صفحه معین قابل دستیابی پیدا کنیم ، می توانیم این کار را انجام دهیم حتی در یک دستگاه واحد با حافظه اصلی بزرگ. با این حال ، فقط برای تولید بستار تعدی از گراف می تواند 10^{18} جفت گره را شامل شود ، که عملی نیست ، حتی با استفاده از یک خوشه بزرگ از کامپیوترها.

۱۰,۸,۴ بستار تعدی به وسیله MapReduce

موازی سازی بستار تعدی بسیار آسان‌تر از دسترس‌پذیری است. اگر بخواهیم $N(v, \infty)$ را محاسبه کنیم، گزینه‌ای برای محاسبه مجموعه گره‌های دسترس‌پذیر از گره v، بدون محاسبه کل بستار تعدی نداریم، به جز محاسبه توالی همسایه‌ها. که در اصل یک جستجوی breadth-first برای گرافی از v است. از نظر رابطه ، فرض کنید ما یک رابطه $Arc(X, Y)$ داریم که شامل جفت (x, y)است به گونه ای که یک قوس $x \rightarrow y$ وجود دارد. ما میخواهیم به صورت تکرار پذیر یک رابطه $Reach(X)$ را محاسبه کنیم که مجموعه‌ای از گره‌های قابل دستابی از گره v است. بعد از i دور، $Reach(X)$ شامل همه این گره‌ها در $N(v, \infty)$ خواهد شد.

در ابتدا، $Reach(X)$ فقط شامل گره v است. فرض کنید بعد از چند دور MapReduce شامل همه گره‌های $N(v,i)$ می‌شود. برای ساختن $N(v,i+1)$ ما نیاز به اتصال Reach با رابطه Arc داریم. سپس اجتماعی بین نتیجه و مقدار قبلی Reach اعمال می‌کنیم. در SQL :

```
SELECT DISTINCT Arc.Y
FROM Reach, Arc
WHERE Arc.X = Reach.X;
```

این query از ما می‌خواهد که اتصال طبیعی $Reach(X)$ و $Arc(X,Y)$ که به وسیله MapReduce که در بخش 2.3.7 توضیح‌داده شده، انجام می‌شود، محاسبه کنیم.

این که چند دور به این روند نیاز دارد، بستگی به میزان دوری، دور ترین گره‌ای دارد که گره vمی‌تواند به آن برسد. در بسیاری از گراف‌های شبکه های اجتماعی ، قطر کوچک است ، همانطور که در کادر "شش درجه جدایی" بحث شده است. در این صورت ، محاسبه دستیابی موازی با استفاده از MapReduce یا روش دیگر امکان پذیر است. تعداد کمی از محاسبات مورد نیاز خواهد بود و فضای مورد نیاز از فضایی که برای نشان دادن گراف استفاده می کند بیشتر نیست.

با این وجود گراف‌هایی وجود دارد که تعداد دورها در آن یک مانع جدی است. به عنوان مثال ، در بخش معمولی از وب ، اکثر صفحات قابل دسترسی از یک صفحه خاص با مسیری به طول 10-15 قابل دستیابی هستند. با این وجود برخی از جفت صفحه‌ها وجود دارند که از اولی به دومی می‌رسند، اما از مسیری که طول آن تا 100 هم اندازه گیری می‌شود. به عنوان مثال ، وبلاگ ها گاهی ساختار یافته اند ، بنابراین هر پاسخ فقط از طریق نظری که به آن پاسخ می دهد قابل دستیابی است. اجرای آن منجر به مسیری طولانی می‌شود که هیچ میانبری برای آن وجود ندارد. یا یک آموزش در وب ، با 50 فصل ، ممکن است ساختار یافته باشد ، بنابراین شما فقط می توانید از طریق فصل i به صفحه فصل $i-1$ راه پیدا کنید.

جالب است که در بحث موازی سازی بستار تعدی خیلی سریع تر از دسترس‌پذیری محاسبه می‌شود. با یک تکنیک دو برابر کننده بازگشتی ، می توانیم مسیری را که می دانیم در یک دور واحد دو برابر کنیم. بنابراین ، بر روی گرافی با قطر d ، فقط به $\log_2 d$ دور احتیاج داریم. اگر $d=6$ باشد ، تفاوت مهم نیست ، اما اگر $d=1000$ باشد ، $\log_2 d$ در حدود 10 است ، بنابراین یک صد برابر در تعداد دورها کاهش می یابد. همانطور که در بالا گفته شد ،مسئله این است، در حالی که می توانیم بستار تعدی را به سرعت محاسبه کنیم ، باید واقعیت های بیشتری را از آنچه برای محاسبه دسترسپذیری در همان گراف مورد نیاز است محاسبه کنیم، و بنابراین نیازهای فضا برای بستار تعدی می تواند از فضای مورد نیاز برای دسترسپذیری بسیار فراتر رود. یعنی اگر تنها چیزی که می خواهیم مجموعه $Reach\ (v)$ باشد ، می توانیم بستار تعدی از کل نمودار را محاسبه کنیم ، و سپس تمام جفت هایی که v را به عنوان اولین جزء آنها نداریم دور بیندازیم. با این حال تا قبل از تمام شدن پردازش مورد نظر نمی‌توان این گره‌ها را دور انداخت. در حین محاسبه بستار تعدی، می‌توانیم مواردی چون $Path(x,y)$ را در جایی که نه x و نه y از v قابل دستایبی نیست را محایله کنیم، حتی اگر آنها قابل دستیابی هم باشند، شاید نیازی به x برای رسیدن به y نباشد.

فرض کنید گراف به اندازه کافی کوچک باشد که بتوانیم بستار تعدی را در کل آن محاسبه کنیم، ما هنوز باید مراقب باشیم که چگونه این کار را با استفاده از MapReduce یا یک رویکرد موازی دیگر انجام می دهیم. ساده ترین رویکرد بازگشتی-مضاعف شروع از رابطه$Path(X,Y)$ برابر با رابطه $Arc(X,Y)$ است. فرض کنید بعد از گذراندن i دور، رابطه $Path(X,Y)$ شامل همه جفت (x,y) باشه، به گونه‌ای که مسیری از x به y به طول بیشتر از 2^i وجود داشته باشد. سپس اگر ما مسیر را در دور بعدی به خودش متصل کنیم، ما باید همه جفت (x,y) بیابیم، به گونه‌ای که مسیری از x به y با طول بیشتر از $2\times 2^i = 2^{i+1}$ خواهد بود. Query بازگشتی-مضاعف در SQL:

```
SELECT DISTINCT p1.X, p2.Y
FROM Path p1, Path p2
WHERE p1.Y = p2.X;
```

پس از محاسبه این query ، همه جفت ها را با مسیری به طولی بین $2\ to\ 2^{i+1}$ متصل می کنیم ، با فرض اینکه مسیر شامل جفت هایی است که با مسیری به طول بین 1 تا 2^i وصل شده اند. اگر اجتماع نتیجه این query را با خود رابطه Arc بدست آوریم ، سپس تمام مسیرهای به طولی بین $2\ to\ 2^{i+1}$ را گرفته و می توانیم از این اجتماع به عنوان رابطه Path در دور بعدی تکرار برگشتی استفاده کنیم. این query می تواند توسط دو کار MapReduce اجرا شود ، یکی برای انجام عمل اتصال و دیگری برای انجام اجتماع و از بین

بردن نسخه های تکراری. همانطور که برای محاسبات دسترسپذیری موازی مشاهده کردیم ، روش های بخش 2.3.7 و 2.3.8 کفایت می کند.

اگر یک گراف دارای قطر d باشد ، پس از $\log_2 d$ دور از الگوریتم فوق مسیر شامل همه جفت ها $(x, y\_)$ است که در یک مسیر با طول حداکثر d متصل می شوند. یعنی تمام جفت های موجود در بستار تعدی را شامل می شود. مگر در مواردی که ما d را بدانیم ، یک دور دیگر لازم است تا تأیید کنیم که جفت بیشتری پیدا نمی شود ، اما برای d بزرگ ، این روند تعداد دورهای کمتری را نسبت به اولین جستجو breadth-first طول می کشد که برای دستیابی به آن استفاده کردیم. با این حال ، روش بازگشتی-مضاعف ، کارهای اضافی زیادی انجام می دهد. مثال زیر موضوع را روشن می‌کند.

مثال 10.28: فرض کنید که کوتاه ترین مسیر از x_0 به x_{17} به طول 17 باشد. به طور خاص، تصور کنید مسیر $x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{17}$ وجود داشته باشد. در دور پنجم این واقعیت $Path(x_0, x_{17})$ را کشف خواهیم کرد، زمانی که Path شامل همه جفت‌های متصل به هم به واسطه‌ی مسیرهایی تا طول 16 باشد. مسیر یکسانی را به دست خواهیم آورد زمانی که مسیر را به خودش متصل کنیم.

۱۰,۸,۵ بستار تعدی هوشمند

یک نوع بازگشتی-مضائف که از کشف دوباره مسیرهای یکسان جلوگیری می‌کند است که با نام بستار تعدی هوشمند نامگذاری می‌شود. هر مسیری به طول بزرگتر از 1 را می توان به دو قسمت head به طول توانی از 2 و tail که طول آن بزرگتری از head نیست، شکست.

مثال 10.29: مسیری به طول 13 دارای head است که از 8 قوس اول تشکیل شده است ، به دنبال آن یک tail تشکیل شده از 5 قوس آخر. مسیری به طول 2 متشکل از head به طول 1 و در ادامه tail به طول 1 است. توجه کنید که 1 توانی از 2 است $2^0 = 1$. و tail نیز دارای اندازه یکسانی با head خواهد بود، زمانی که خود مسیر هم دارای طولی به توان 2 باشد.

برای اجرای بستار تعدی هوشمند در SQL، ما یک رابطه $Q(X, Y)$ تعریف میکنیم، که عملکرد آن بعد از دور iام، نگه داشتن همه جفت گره‌های (x, y) به طوری که کوتاهترین مسیر از x به y دارای طولی به اندازه 2^i است. همچنین، بعد از دور iام، $Path(x, y)$ صحیح خواهد بود، اگر کوتاه‌ترین مسیر از x به y طولی حداکثر $2^{i+1} - 1$ داشته باشد. توجه داشته باشید که این تفسیر از مسیر با تفسیر مسیر در روش ساده بازگشتی-مضائف که در بخش 10.8.4 ارائه شده است کمی متفاوت است.

در ابتدا ، هر دو Q و Path را به عنوان کپی از رابطه Arc مقدار دهی کنید. پس از دور iام، فرض کنید Q و Path دارای محتوایی گفته شده در پاراگراف قبلی باشند. توجه داشته باشید که برای دور $i = 1$ ، مقادیر اولیه Q و Path در ابتدا شرایطی را که برای $i = 0$ شرح داده شده را برآورده می کنند. در مرحله $(i + 1)$ ، موارد زیر را انجام می دهیم:

1. با استفاده از SQL query ، با اتصال Q به خودش یک مقدار جدید را محاسبه میکنیم.

```
SELECT DISTINCT q1.X, q2.Y
FROM Q q1, Q q2
WHERE q1.Y = q2.X;
```

2. Path را از Q محاسبه شده در مرحله 1 کم می‌کنیم. توجه داشته باشید که مرحله 1 تمامی مسیرهای به طول $2^(i + 1)$ را کشف می‌کند. اما برخی از جفت های متصل به این مسیرها نیز ممکن است مسیرهای کوتاه تری داشته باشند.
3. اتصال Path با مقدار جدید Q که در مرحله 2 محاسبه شده،با استفاده از SQL query

```
SELECT DISTINCT Q.X, Path.Y
FROM Q, Path
WHERE Q.Y = Path.X
```

در آغاز دور Path تمام (y, z) را شامل می‌شود، به طوری که کوتاهترین مسیر از y به x داری طول $2^{i+1} - 1$ از y به z است، و مقدار جدید Q شامل همه جفت (x, y) هایی است که کوتاهترین مسیر از x به y به طول 2^{i+1} دارند.

4. مقدار جدید Path اجتماع رابطه محاسبه شده در مرحله 3 و مقدار جدید Q که در مرحله 1 محاسبه شده و مقدار قبلی Path است.

در هر دور از الگوریتم بستار تعدی هوشمند از مراحلی استفاده می شود ، پیوستن ، جمع شدن (حذف های مضاعف) یا اجتماع ها. بنابراین یک دور می تواند به عنوان یک دنباله کوتاه از کارها MapReduce اجرا شود. بعلاوه ، با استفاده از الگوهای عمومی تر ارتباطات مجاز توسط یک سیستم گردش کار ، می توان میزان خوبی را صرفه جویی کرد ، (در بخش 2.4.1 مراجعه کنید).

۱۰,۸,۶ بستار تعدی به واسطه کاهش گراف

گراف جهت دار معمولی مانند وب شامل بسیاری از مؤلفه های به شدت متصل SCC است. ما می توانیم SCC را به یک گره واحد بشکنیم تا جایی که مربوط به بستار تعدی باشد، زیرا همه گره های یک SCC دقیقاً به همان گره ها می رسند. یک الگوریتم ظریف برای یافتن SCC یک گراف به صورت خطی در اندازه گراف وجود دارد. با این حال ، این الگوریتم ذاتا دنباله ای است ، بر اساس جستجوی عمق اول ، و بنابراین به خوبی برای تلقیح موازی در گراف‌های بزرگ مناسب نیست.

ما می توانیم اکثر SCC ها را با استفاده از انتخاب برخی گره های تصادفی به دنبال یک جستجوی breadth-first ، در یک گراف پیدا کنیم. در ضمن ، هرچه SCC بزرگتر باشد ، احتمال فروپاشی زودرس بیشتر می شود ، بنابراین اندازه گراف را به سرعت کاهش می دهد. الگوریتم کاهش SCC به گره های واحد به شرح زیر است. G گراف کاهش یافته و G'' همان G است به همرا همه قوس‌های معکوس شده.

1. یک گره v از G را بطور تصادفی انتخاب کنید.
2. $N_G(v,\infty)$ را پیدا کنید، مجموعه گره‌های قابل دسترس از v در G
3. $N_{G'}(v,\infty)$ را پیدا کنید، مجموعه گره‌هایی که v می‌تواند به آنها دست یابد از گراف G'' که قوس‌های معکوس شده G است.
4. $SCC\ S$ حاوی v را بسازید، که $N_G(v,\infty)\bigcap N_{G'}(v,\infty)$. یعنی ، v و u در یک SCC از G قرار دارند ، اگر و تنها اگر v بتواند به u برسد و u می توانید به v برسد.
5. $SCC\ S$ را با یک گره تنها درون G جایگزین می‌کنیم.برای انجام این کارتمام گره‌های S را از G حذف می‌کنیم و s را به مجموعه گره‌های G اضافه می‌کنیم. تمام قوس‌های درون G با یک یا دو انتها که درون S هستند را حذف می کنیم. سپس هر زمان که یک قوس در G وجود داشته باشد که از هر یک از اعضای S به x است، به مجموعه قوس‌های G یک قوس $s \to x$ اضافه می‌کنیم. در پایان ، اگر قوس ای از x به هر عضو S وجود داشته باشد ، یک قوس x اضافه می‌کنیم.

> **Path Facts Versus Paths**
>
> ما باید مراقب باشیم بین یک Path ، که یک توالی از قوس ها است ، و یک Path Fact تمایز قائل شویم ، این جمله ای است که می گوید یک مسیر از یک گره x تا یک گره y وجود دارد. Path Fact به طور معمول به صورت $Path(x,y)$ نمایش داده می شود. بستار تعدی هوشمند ، هر Path را فقط یک بار کشف می کند ، اما ممکن است بیش از یک بار Path Fact را کشف کند. دلیل این امر این است که غالباً یک گراف مسیرهای زیادی از x تا y خواهد داشت و حتی ممکن است مسیرهای مختلف زیادی از x تا y داشته باشد که طول آنها یکسان باشد. همه مسیرها توسط بستار تعدی هوشمند کشف نمیشوند.

ما می توانیم مراحل فوق را تعداد دفعات ثابت تکرار کنیم. ما می توانیم به طور متناوب آن را تکرار کنیم تا زمانی که گراف به اندازه کافی کوچک شود ، یا می توانیم تمام گره ها را به نوبه خود بررسی کنیم و متوقف نشویم تا زمانی که هر گره به خودی خود در یک SCC باشد.

$$N_G(v,\infty)\bigcap N_{G'}(v,\infty) = \{v\}$$

برای همه گره های باقی مانده v. اگر انتخاب دوم را انجام دهیم ، گراف نتیجه به عنوان کاهش انتقالی نمودار اصلی G نامیده می شود. کاهش انتقالی همیشه حلقوی است ، زیرا اگر چرخه ای داشته باشد ، SCC بیش از یک گره باقی می ماند. با این وجود ، لازم نیست که به یک گراف حلقوی تبدیل شود ، تا زمانی که گراف حاصل گره های کمی داشته باشد که محاسبه بستار تعدی از این گراف امکان پذیر باشد. یعنی تعداد گره ها به اندازه کافی اندک است که می توانیم با نتیجه ای روبرو شویم که اندازه آن متناسب با مربع آن تعداد گره ها باشد.

در حالی که بستار تعدی از گراف کاهش یافته دقیقاً برابر با بستار تعدی از گراف اصلی نیست، اطلاعات مربوط به آنچه SCC هر گره اصلی به آن تعلق دارد ، کافی است تا به ما چیزی بگویید که بستار تعدی از گراف اصلی به ما میگوید. اگر می خواهیم بدانیم که آیا مسیر $(u، v)$ در گراف اصلی صحیح است ، SCC حاوی u و v را پیدا می‌کنیم. اگر یکی یا هر دو این گره ها هرگز در یک SCC ترکیب نشده باشند، با آن گره به عنوان SCC برخورد می‌کنیم. اگر u و v متعلق به SCC یکسانی باشند، پس مطمئنا u می‌تواند به v برسد.

مثال 10.30: بگذارید تصویر "bowtie" وب را از قسمت 5.1.3 مرور کنیم. تعداد گره ها در بخشی از گراف مورد بررسی بیش از 200 میلیون نفر بود. که مطمئنا عددی بزرگی برای برای کار با داده ها است. یک مجموعه بزرگ از گره ها به نام "SCC" وجود داشت که به عنوان مرکز گراف در نظر گرفته می شد. از آنجا که حدود یک گره از تعداد چهار گره در این SCC قرار داشت، به محض انتخاب هر یک از اعضای آن به طور تصادفی، به یک گره واحد شکسته می شد. اما بسیاری از SCC های دیگر در وب وجود دارند ، حتی اگر صریحاً در "bowtie" نشان داده نشوند. برای مثال in-component دارای یک SCC بزرگ است. گره های موجود در یکی از این SCC ها می توانند به یکدیگر برسند ، و می توانند به بعضی از گره های دیگر در component برسند ، و البته می توانند به همه گره ها در SCC مرکزی برسند. SCC در اجزای داخل و خارج ، لوله ها و سایر سازه ها همه می توانند به شکسته شده و منجر به یک نمودار بسیار کوچکتر شوند.

۱۰،۸،۷ تقریبی از اندازه همسایه‌ها

در این بخش مسئله محاسبه مشخصات همسایه‌ها برای هر گره از یک گراف بزرگ را بررسی خواهیم کرد. یک نوع از مسئله ، که مانند تکنیک یافتن اندازه مجموعه قابل دستیابی برای هر گره v ، یعنی مجموعه ای است که ما از آن $N(v, \infty)$ نامیده ایم عمل می‌کند. به یاد بیاورید که برای گرافی با یک میلیارد گره ، محاسبه همسایه ها برای هر گره کاملاً غیرممکن است ، حتی با استفاده از یک دسته بسیار بزرگ کامپیوترها. با این حال ، حتی اگر فقط تعداد گره ها را در هر همسایگی بخواهیم ، باید گره های کشف شده را تا آنجا که گراف را کشف می کنیم به خاطر بسپاریم ، یا اینکه دیگر نمی دانیم گره یافت شده جدید است یا گره ای که قبلاً دیده ایم.

از طرف دیگر ، با استفاده از تکنیک Flajolet-Martin که در بخش 4.4.2 مورد بحث قرار گرفت ، یافتن تقریبی اندازه هر همسایگی چندان سخت نیست. به یاد بیاورید که این روش از تعداد زیادی توابع هش استفاده می کند. در این حالت ، توابع هش بر روی گره های گراف اعمال می شود. برای هر مجموعه از گره ها ، یک تخمین از اندازه مجموعه R2 است ، در جایی که R طول طولانی ترین tail برای هر عضو از مجموعه است. بنابراین ، به جای اینکه بتوانیم تمام اعضای مجموعه را ذخیره کنیم ، می توانیم در عوض فقط مقدار R را برای آن مجموعه نگه داریم. البته، بسیاری از توابع هش وجود دارد، بنابراین ما باید مقادیر R را برای هر عملکرد هش ذخیره می‌کنیم.

مثال 10.31: اگر ما از یک تابع هش استفاده می کنیم که یک رشته 64 بیتی تولید می کند، شش بیت تمام موارد لازم برای ذخیره هر مقدار از آن R است. برای مثال ، اگر یک میلیارد گره وجود داشته باشد ، و می خواهیم اندازه محله را تخمین بزنیم. برای هر گره می توان مقدار R را برای 20 گره در 15 گیگ ذخیره کرد.

اگر طول tail را برای هر همسایه ذخیره کنیم ، می توانیم از این اطلاعات برای محاسبه تخمین برای همسایه های بزرگتر از برآوردهای خود برای همسایگی‌های کوچکتر استفاده کنیم. یعنی فرض کنید تخمین هایمان را برای $|N(v, d)|$ برای همه گره‌های v محاسبه کرده ایم، و می خواهیم محله های شعاع $d + 1$ محاسبات را محاسبه کنیم. برای هر عملکرد هش h ، مقدار R برای $N(v, d + 1)$ بزرگترین است:

1. Tail مربوط به v و
2. مقادیر R همراه با h و $N(u, d)$، جایی که $v \rightarrow u$ قوس نمودار است.

توجه کنید که مهم نیست که گره فقط از طریق یک جانشین v در گراف یا از طریق جانشین های مختلف قابل دستیابی باشد. ما هر دو تخمین را می گیریم. این ویژگی مفید همان بود که ما در بخش 4.4.2 از آن برای تخمین اینکه آیا یک عنصر در stream امده است یا نه استفاده کردیم.

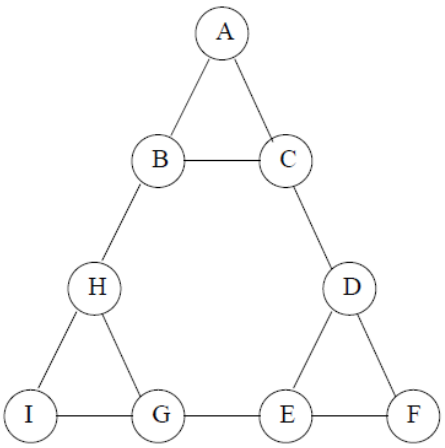
اکنون الگوریتم کاملی را با نام ANF (تابع تقریب همسایگی) را شرح خواهیم داد. تعدادK تابع $h_1, h_2, \ldots, h_k$ را انتخاب می‌کنیم. برای دهر گره v و شعاع d، $R_i(v, d)$ به عنوان بیشینه طول tail گره درون $N(v, d)$ با استفاده از تابع h_i نشان می‌دهیم. برای شروع کار $R_i(v, 0)$ طول tail مربوط به $h_i(v)$ برای همه i و v ها است.

برای مرحله استقرایی، فرض کنید برای همه i و v ها مقدار $R_i(v, d)$ را محاسبه کرده‌ایم. برای شروع $R_i(v, d + 1)$ برای $R_i(v, d)$ قرار میدهیم (برای همه i و v ها). سپس همه‌ی قوسهای $x \rightarrow y$ در گراف را با هر ترتیبی، در نظر میگیریم. برای هر $x \rightarrow y$ مقدار $R_i(x, d + 1)$ را برابر با بزرگترین مقدار جاری $R_i(v, d)$ قرار می دهیم. توجه داشته باشید که ممکن است ترتیبی که برای قوسها نظر میگیریم سرعت بالایی در این مورد ایجاد کند که بتوانیم R_i را در حافظه اصلی ذخیره کنیم ، در حالی که مجموعه قوس ها به اندازه‌ای است که باید روی دیسک ذخیره شود. همانطور که در بخش 4.4.3 توضیح داده شد میتوانیم R را در گروه های کوچک قرار داده، میانگین گرفته و معدل متوسط را حساب می‌کنیم.

یکی دیگر از پیاده‌سازی های الگوریتم ANF که می‌تواند در صورت علاقه ما به تخمین اندازه مجموعه‌های قابل دست‌یابی استفاده شود.

$$R_i(x) := \max(R_i(x), R_i(y))$$

در صورت عدم تغییر مقادیر $R_i(v)$ می‌توانیم پیمایش را متوقف کینم. یا مثلا اگر بندانیم که قطر گراف برابر با d است، کافیست d بار پیمایش را تکرار کنیم.



شکل10.24 : گرافی برای تمرین در همسایگی‌ها و بستار تعدی

۱۰,۸,۸ تمرین‌های این بخش

تمرین 10.8.1: برای گراف شکل 10.9 که در اینجا مانند شکل 10.24 تکرار شده است:

الف) اگر گراف به عنوان یک گراف جهت دار نشان داده شود ، چند قوس وجود دارد؟

ب) خصوصیات همسایه‌ها برای گره های A و B چیست؟

پ) قطر نمودار چیست؟

ج) چند جفت در بستاد تعدی وجود دارد؟

نکته: فراموش نکنید که مسیری به طول بیش از صفر از یک گره به خود در این نمودار وجود دارد.

د) اگر بستار تعدی را با بازگشتی – مضائف محاسبه کنیم ، چند دور لازم است؟

تمرین 10.8.2: الگوریتم بستار تعدی هوشمند مسیرهای هر طول را به head و tail از طول های خاص شکسته است. طول head و tail برای هر مسیر به طول‌های 7 و 8 و 9 چقدر است؟

تمرین 10.8.3: یک اجرا از شبکه اجتماعی که اخرین بار در شکل 10.23 نمایش داده شده است را در نظر بگیرید. فرض کنید ما از یک تابع هش h استفاده می کنیم که هر گره (حرف بزرگ) را به کد ASCII آن نگاشت می کند.

الف) با استفاده از این تابع هش ، مقادیر R را برای هر گره و شعاع 1 محاسبه کنید. اندازه های هر همسایه چیست؟ تخمین ها چگونه با واقعیت مقایسه می شوند؟

ب) سپس مقادیر R را برای هر گره و شعاع 2 محاسبه کنید. باز هم تخمین های اندازه همسایه‌ها را محاسبه کنید و با واقعیت مقایسه کنید.

۱۰,۹ خلاصه فصل

گراف شبکه‌های اجتماعی: نمودارهایی که نمایانگر اتصالات در یک شبکه اجتماعی هستند ، نه تنها بزرگ هستند ، بلکه شکل محلی را به نمایش می گذارند ، جایی که زیر مجموعه های کوچک گره ها (جوامع) دارای چگالی بسیار بالاتری از لبه ها نسبت به چگالی متوسط هستند.

جوامع و خوشه ها: در حالی که جوامع از جهاتی به خوشه ها شباهت دارند ، تفاوت های قابل توجهی نیز وجود دارد. افراد (گره ها) معمولاً متعلق به چندین اجتماع هستند و اقدامات معمول از راه دور بیانگر نزدیکی در گره های یک جامعه نیست. در نتیجه ، الگوریتم های استاندارد برای یافتن خوشه ها در داده ها برای یافتن جامعه به خوبی کار نمی کنند.

(بین)Betweenness: یکی از راه های جدا کردن گره ها به جوامع ، اندازه گیری فاصله یال‌ها است ، که این مقدار بیش از همه جفت گره ها از کسری از کوتاه ترین مسیرها بین آن گره هایی است که از یال معینی عبور می کنند.

الگوریتم Girvan-Newman: الگوریتم Girvan-Newman یک روش کارآمد برای محاسبه بین یال ها است. یک جستجو برای اولین بار از هر گره انجام می شود ، و دنباله ای از مراحل برچسب زدن ، سهم مسیرها از ریشه به هر گره دیگر را که از هر یک از یال ها عبور می کند ، محاسبه می کند. سهم مورد نظر برای یالی که برای هر ریشه محاسبه می شود جمع می شود تا فاصله بین آنها بدست آید.

انجمنها و یال های کامل دو قطعه: یک گراف کامل دو طرفه دارای دو گروه گره است ، تمام یال های ممکن بین جفت گره ها یکی از هر گروه را انتخاب می کنند و هیچ یالی بین گره های همان گروه وجود ندارد. هر جامعه ای به اندازه کافی متراکم (مجموعه ای از گره ها که یال های زیادی در بین آنها وجود دارد) دارای یک گراتف کامل دو طرفه خواهند بود.

یافتن گراف های دو قطبی کامل: ما می توانیم با همان تکنیک هایی که برای یافتن مجموعه موارد مکرر از آنها استفاده کردیم ، گراف های کاملاً دو بخشی را پیدا کنیم. گره های نمودار را می توان هم به عنوان item و هم به عنوان buckets در نظر گرفت. سبد مربوط به یک گره ، مجموعه گره های مجاور است که به عنوان موارد در نظر گرفته می شود. می توان تصور کرد که یک گراف دو قطبی کامل با گروه های گره از اندازه و اندازه اندازه t و s پیدا شده است.

تقسیم بندی گراف: یکی از راه های یافتن جوامع ، تقسیم یک نمودار به طور مکرر به قطعات با اندازه تقریباً مشابه است. برش یک پارتیشن از گره های گراف به دو مجموعه و اندازه آن تعداد یال هایی است که در هر مجموعه دارای انتهای یکسانی هستند.

برش های نرمال: می توانیم با در نظر گرفتن نسبت اندازه برش و حجم هر یک از دو مجموعه که توسط برش ایجاد می شود ، اندازه یک برش را نرمالایز کنیم. سپس این دو نسبت را اضافه می کنیم تا مقدار برش نرمالایز شده بدست آید. برش های نرمالایز شده با هزینه کم خوب هستند ، به این معنا که تمایل دارند گره ها را به دو قسمت تقریباً مساوی تقسیم کنند و از اندازه نسبتاً کمی برخوردار باشند.

ماتریس Adjacency: این ماتریس ها گراف را توصیف می کند. اگر یالی بین دو گره وجود داشته باشد، درایه متناظر آن در ماتریس برابر با 1 است و در غیر این صورت مقدار 0 را به خود خواهد گرفت.

ماتریس درجه: اگر گره iام یک گراف دارای درجه d باشد، مقدار درایه مورد نظر ان در ماتریس مورد نظر هم برابر با d خواهد بود.