گزارش پروژه‌ی کارشناسی:

بیشینه‌سازی تاثیر در شبکه‌های اجتماعی با توجه به ویژگی‌های انجمنی

نوید صالح‌نمدی

استاد راهنما:

دکتر مسعود اسدپور

تابستان ۱۳۹۴

مقدمه:

با توجه به افزایش کاربرد بازاریابی ویروس‌وار[1] و هم‌چنین حضور فزاینده‌ی مردم در شبکه‌های اجتماعی باعث داغ‌شدن مساله‌ی نسبتا جدیدی در این موضوع شده است که موضوع اصلی این پروژه را نیز در بر گرفته است. به طور خلاصه در بازاریابی ویروس‌وار با اطلاع‌رسانی محدود در رابطه با وجود یک محصول جدید به چند نفر می‌خواهیم با توجه به ارتباطاتی که مردم با هم دارند، این خبر یا این محصول به صورت دهان به دهان[2] بین همه‌ی مردم منتشر شود. با مدل‌کردن این مفهوم در شبکه‌های اجتماعی به نوعی ما می‌خواهیم در یک شبکه‌ی اجتماعی غیر فعال، تعدادی محدود و مشخصی از رئوس را انتخاب کنیم که با فعال‌کردن آن‌ها بیش‌ترین رئوس در شبکه فعال شوند. مطالعات بسیار زیادی در رابطه با این موضوع شده است که عموما بر روی الگوریتم حریصانه‌ی دیوید کمپ در سال ۲۰۰۳ بنا نهاده شده‌اند. این الگوریتم‌ها عموما بر روی همه‌ی شبکه‌ها اعمال می‌شوند و به ساختار شبکه هیچ توجهی نمی‌کنند. از طرف دیگر ساختار شبکه‌ها تاثیر بسیار زیادی بر رفتاری مانند انتقال یک خبر دارد و پارامترهای گوناگونی برای سنجش و طبقه‌بندی شبکه‌ها وجود دارد. همان‌طور که در جوامع موجود می‌توانیم ببینیم ارتباطات افراد معمولا بین یک گروه یا انجمن[3] خاص بیش‌تر است و ارتباطات کم‌تری با انجمن‌های دیگر وجود دارد پس می‌توانیم شبکه‌های اجتماعی بین انسان‌ها را به چندین انجمن تقسیم کنیم که یال‌های درونی هر زیاد و یال‌های بین انجمن‌های مختلف کم است. هدف اصلی ما در این پروژه مطالعه‌ی الگوریتم‌های موجود روی شبکه‌هایی با ساختار انجمنی هستند و هم‌چنین ارائه‌ی الگوریتمی که با توجه به انجمن‌های موجود در شبکه بهینه‌تر کار کند.

بازاریابی ویروس‌وار:

شرکت‌های بزرگ برای معرفی محصولاتشان نیازمند به روش‌های تبلیغ با قیمت کم و نفوذ بالا دارند. یک استراتژی معرفی مداوم محصولات و تاکید بر کیفیت آن‌ها است تا مشتری را به خرید ترغیب کند و هم‌چنین در زمان خرید در مقابل رقیبان شانس بیش‌تری داشته باشد. روش‌های عمومی مانند نصب پوستر و بیلبورد یا تبلیغ رادیویی و تلویزیونی ممکن است در زمان کم به افراد زیادی یک محصول را معرفی کند اما ممکن است همه را ترغیب به خرید یا استفاده از محصول نکند. در مقابل تاثیر سخن یک دوست و آشنا ممکن است چندین برابر یک تبلیغ عمومی باشد. بازاریابی ویروس‌وار با تکیه بر همین موضوع شکل می‌گیرد. یعنی با ترغیب افراد بسیار کمی (نسبت به کل جامعه) برای تبلیغ یک محصول و با توجه به این که مردم به صورت‌های مختلف با هم ارتباط دارند این محصول را دهان به دهان منتشر می‌کنند. به عنوان مثال فرض کنید که یک نفر یک ویروس سرماخوردگی گرفته است در صورتی که با افراد مختلف ارتباط داشته باشد (مثلا دست‌دادن) این ویروس به آن‌ها هم منتقل می‌شود و همین‌طور به افراد دیگر تا اکثریت یک جامعه به این ویروس دچار شوند. همین‌طور که دیده می‌شود هزینه‌ی بیمارکردن (مطلع‌کردن) یک نفر بسیاربسیار ارزان‌تر از بیمارکردن (مطلع‌کردن) تعداد زیادی از افراد جامعه است. در بازاریابی ویروس‌وار توجه به سه رکن بسیار ضروری است ۱- پیام‌رسان ۲- پیام ۳- محیط [4] یعنی باید با انتخاب فرد مناسب به عنوان پیام‌رسان یک پیام را به افراد مختلف انتقال دهد و آن‌ها را نیز تبدیل به پیام‌رسان کند و این روند برای هر پیام‌رسان دوباره تکرار شود. تمام انتقال‌های پیام در محیط انجام می‌گیرد که متشکل از زمان و جو موجود، وسیله‌ی ارتباطی و عامل‌های دیگر است. اهمیت خود پیام در تاثیرگذاری آن روی افراد دیگر و بالابردن احتمال پیام‌رسان‌شدن خود آن‌ها است. مسئله‌ی بیشینه‌سازی تاثیر تمرکز خود را روی پیداکردن پیام‌رسان‌های مهم گذاشته است و پارامترهای پیام و محیط را به صورت ورودی در نظر می‌گیرد.

مدل‌سازی مسئله‌ی بیشینه‌سازی تاثیر:

---

[1] Viral Marketing
[2] Word of Mouth
[3] Community
[4] Kaplan Andreas M., Haenlein Michael (2011) Two hearts in three-quarter time: How to waltz the Social Media/viral marketing dance, Business Horizons 253-263

از سالیان بسیار دور، دانشمندان جامعه‌شناسی درباره‌ی ارتباط‌های درون جامعه و نحوه‌ی پخش اطلاعات (یا پخش شایعه) مطالعه کرده بودند، اما ارتباط این موضوع با علوم کامپیوتر تا حدود ۱۰ سال قبل ناشناخته بود. دیوید کمپ و همراهان در مقاله‌هایی در سال ۲۰۰۳، مسئله‌ی بیشینه‌سازی تاثیر در شبکه‌های اجتماعی را به صورت ریاضی مدل کردند و مدل‌های انتشار مختلف را برای آن معرفی کردند. هم‌چنین اثبات کردند الگوریتم پیداکردن جواب بهینه NP است و یک الگوریتم حریصانه‌ی تقریبی برای آن ارائه دادند. موضوعات مطرح‌شده در این مقالات، زمینه‌ی تحقیقاتی جدیدی در علوم کامپیوتر و شبکه‌های اجتماعی ایجاد کرد که هم‌چنان موضوعی داغ برای تحقیق محسوب می‌شود و دانشمندان بسیاری در حال بهینه‌ترکردن الگوریتم‌ها از نظر زمانی و هم‌چنین دقت جواب خروجی هستند. حال مفاهیمی که برای مدل‌سازی مساله و ارائه‌ی الگوریتم ضروری است را تعریف می‌کنیم.

مدل‌های انتشار تصادفی:[5]

تعریف: یک مدل انتشار تصادفی (با گام‌های زمانی گسسته) برای یک شبکه‌ی اجتماعی با گراف $G = (V, E)$ یک فرآیند اتفاقی را برای مشخص‌کردن رئوس فعال S_t در زمان $t \geq 1$ و طبق رئوس فعال اولیه‌ی S_0 را مشخص می‌کند.

به زبان ساده‌تر، یک مدل انتشار تصادفی با داشتن گراف شبکه و رئوس فعال اولیه، برای زمان‌های بعد از شروع، رئوسی که فعال خواهند شد را به صورت تصادفی تعیین می‌کند. مدل‌های انتشار مختلفی معرفی شده‌اند که دو مدل از آن‌ها محبوب‌تر و کارآمدتر هستند. مدل انتشار آبشاری مستقل (Independent Cascade)و مدل انتشار آستانه‌ی خطی(Linear Threshold) .

مدل Independent Cascade:

تعریف: مدل IC برای یک شبکه‌ی اجتماعی با گراف $G = (V, E)$، یک نگاشت احتمال نفوذ برای هر یالp و هم‌چنین رئوس فعال اولیه S_0 را می‌گیرد و مجموعه‌ی رئوس فعال S_t را برای هر زمان $t \geq 1$ طبق فرآیند تصادفی زیر تعیین می‌کند. در هر گام $t \geq 1$ مجموعه‌ی S_t را برابر با S_{t-1} قرار می‌دهد. سپس مجموعه‌ی A را برابر با هر عضوی که در S_{t-1} وجود دارد اما در S_{t-2} نبوده، قرار می‌دهد. برای هر راس غیرفعال در S_t مانند v عملیات زیر را برای تمام رئوس $u \in Ni_v \cap A$ انجام می‌دهد که Ni_v برابر همسایه‌های ورودی v است. راس u با احتمال $p(u, v)$ می‌تواند راس v را فعال کند (آزمایش سکه‌ی برنولی با احتمال $p(u, v)$). در این صورتv را به S_t اضافه می‌کنیم و سراغ راس بعدی می‌رویم، این کار را برای هر t انجام می‌دهیم تا تمام S_t را به دست آوریم.

به زبان دیگر، هر راس فعالی در زمان فعال‌شدنش، رئوس غیرفعال خود را با احتمال متناظر رابطه‌شان (که به طور مستقل مشخص شده‌است) فعال می‌کند و دیگر در زمان‌های بعدی در فرآیند فعال‌کردن شرکت نمی‌کند. پایه‌ی این مدل بر این که است که هر فرد در مواجهه با یک خبر بدون توجه به ساختار شبکه و دیگر دوستانش و تنها بر پایه‌ی ارتباط با خبررسان تصمیم بر قبول‌کردن یا رد آن خبر می‌کند. این مدل در مسائلی مانند انتقال ویروس به خوبی کار می‌کند چون احتمال بیمارشدن هر فرد در مواجهه با یک بیمار دیگر تنها وابسته به نوع ارتباط با آن فرد است و این موضوع که با بیماران دیگری قبلا ارتباط داشته است تاثیری ندارد.

مدل Linear Threshold:

تعریف: مدل LT برای یک شبکه‌ی اجتماعی با گراف $G = (V, E)$، یک نگاشت وزن نفوذ برای هر یالw و هم‌چنین رئوس فعال اولیه S_0 را می‌گیرد و مجموعه‌ی رئوس فعال S_t را برای هر زمان $t \geq 1$ طبق فرآیند تصادفی زیر تعیین می‌کند. در ابتدا هر راس v برای خود یک آستانه‌ی نفوذ θ_v با توزیع یکنواخت در بازه‌ی $[0,1]$ انتخاب می‌کند. در هر گام $t \geq 1$ مجموعه‌ی S_t را برابر با S_{t-1} قرار می‌دهد. سپس برای هر راس غیرفعال v، بررسی می‌کنیم که مجموع وزن یال‌های ورودی به v که از رئوس فعال آمده‌اند از θ_v بیش‌تر مساوی است یا خیر. در صورت بیش‌تر مساوی‌بودن، راس v به S_t اضافه می‌شود. این کار را برای هر t انجام می‌دهیم تا تمام S_t را به دست آوریم.

[5] Stochastic Diffusion Process

این مدل برخلاف مدل IC در هنگام فعال‌شدن هر فرد، تنها تاثیر یک فرد مهم نیست و وضعیت دیگر همسایه‌های یک فرد اهمیت دارد. برای مثال فرض کنید یک تکنولوژی جدید معرفی شود (اینترنت موبایل‌های همراه 3G) که مردم برای این که به این تکنولوژی اعتماد کنند و از آن استفاده کنند علاقه دارند تعداد قابل قبولی از افراد مورد اعتماد آن‌ها از این تکنولوژی استفاده کنند و اعلام رضایت کنند.

طبق تعریف دو الگوریتم، می‌توانیم نتیجه بگیریم که انتشار پس از مدت محدودی متوقف می‌شود که این زمان حداکثر به اندازه‌ی تعداد رئوس است (حالت حدی آن هم گرافی که یک مسیر است و با مجموعه‌ی فعال یک سر آن پس از $n-1$ مرحله فعال می‌شود)

حال پس از تعریف این دو مدل انتشار لازم است دو خاصیت در مدل‌های انتشار را معرفی کنیم که در الگوریتم حریصانه به کار برده می‌شود. در این دو تعریف تابع f را تابعی در نظر بگیرید که با گرفتن مجموعه‌ی رئوس فعال اولیه، تعداد رئوس فعال نهایی را خروجی می‌دهد.

Submodularity:

تعریف: یک مدل انتشار خاصیت Submodularity دارد به شرطی که به ازای مجموعه‌های $S \subseteq T \subseteq V$ و عضوی مثل v که در T نباشد ولی در V باشد، داشته باشیم:

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$$

یعنی افزایش حاشیه‌ای تابع برای S کوچک‌تر از T نباشد. از این به بعد این افزایش حاشیه‌ای را با $f(v|S)$ نمایش می‌دهیم.

یکنواختی (Monotonicity):

تعریف: یک مدل انتشار خاصیت یکنواختی دارد به شرطی که به ازای مجموعه‌های $S \subseteq T \subseteq V$ داشته باشیم:

$$f(S) \leq f(T)$$

یعنی افزایش راس به مجموعه‌ی فعال اولیه، باعث کاهش مجموعه‌ی فعال نهایی نمی‌شود.

حال با دانستن مدل‌های انتشار، مسئله‌ی بیشینه‌سازی نفوذ را تعریف می‌کنیم.

تعریف مساله: بیشینه‌سازی نفوذ فرآیند تصادفی زیر است: با داشتن گراف $G(V,E)$ و یک مدل انتشار تصادفی روی G و بودجه‌ی مشخص k، مجموعه‌ای مانند S_0 پیدا کنید که $|S_0| \leq k$ و تابع نفوذ آن $f(S_0)$ تحت مدل انتشار بیشینه باشد.

چندین قضیه‌ی مهم در رابطه با موضوع بیشینه‌سازی نفوذ وجود دارد که آن‌ها را در زیر معرفی می‌کنیم.

قضیه: مدل‌های انتشار IC و LT مستقل از ساختار شبکه، خاصیت Submodularity و Monotonicity دارند.

قضیه: محاسبه‌ی انتشار شبکه‌ی $G(V,E)$ و مجموعه‌ی فعال اولیه‌ی S در مدل‌های انتشار IC و LT ($f(S)$) در خانواده‌ی #P-Hard قرار می‌گیرند.

توضیح: خانواده‌ی #P-Hard حداقل به سختی NP-Hard است. چون در #P به جای پاسخ به تصمیم «آیا وجود دارد؟» باید به تصمیم «چند تا وجود دارد؟» پاسخ داد.

قضیه: مسئله‌ی بیشینه‌سازی انتشار در مدل‌های انتشار IC و LT شامل مساله‌ای جزو خانواده‌ی NP-Hard برای ی حالت خاص است در نتیجه جزو خانواده‌ی NP-Hard قرار می‌گیرد.

حال با معرفی این قضایا و با آگاهی به این موضوع که این مسئله جزو مسائل NP-Hard است و راه حل قطعی در حال حاضر برای آن پیدا نمی‌شود، باید به دنبال روش‌های تقریبی باشیم. دیوید کمپ روشی حریصانه را برای حل این موضوع ارائه کرده است که خروجی آن حداقل $(1-\frac{1}{e})$ پاسخ بیشنیه است.

الگوریتم حریصانه:

این الگوریتم به سادگی از افزایش نفوذ جانبی (Marginal Influence) استفاده می‌کند. به این صورت که مجموعه‌ی S را مجموعه‌ی رئوس فعال تا به این لحظه در نظر بگیرید. از بین همه‌ی رئوس غیرفعال، راسی که بیش‌ترین افزایش نفوذ جانبی دارد را انتخاب می‌کنیم (یعنی اگر آن را به S اضافه کنیم بیش‌ترین مقدار در $f(S)$ پدید می‌آید. آن راس را به مجموعه‌ی رئوس S اضافه می‌کنیم و این کار را ادامه می‌دهیم. در ابتدا S مجموعه‌ی تهی است و تا زمانی ادامه می‌دهیم که $|S| = k$ شود.

Algorithm 1 Greedy(k, f): general greedy algorithm.

Input: k: size of returned set; f: monotone and submodular set function
Output: selected subset

1: initialize $S \leftarrow \emptyset$
2: **for** $i = 1$ to k **do**
3: $\quad u \leftarrow \text{argmax}_{w \in V \setminus S}(f(S \cup \{w\}) - f(S))$
4: $\quad S \leftarrow S \cup \{u\}$
5: **end for**
6: **return** S

برای محاسبه‌ی $f(S \cup \{w\}) - f(S)$ از شبیه‌سازی مونت‌کارلو استفاده می‌کنیم. به این صورت که با پیمایش گراف از رئوس فعال و به وسیله‌ی آزمایش سکه‌ی برنولی برای هر یال مقدار $f(S)$ را به دست می‌آوریم و برای بهبود دقت جواب، این کار را چندین بار انجام می‌دهیم و میانگین می‌گیریم.

قضیه: خروجی الگوریتم حریصانه، حداقل $(1-\frac{1}{e})$ برابر پاسخ بهینه است. یعنی اگر خروجی الگوریتم حریصانه برای مساله‌ی بیشینه‌سازی نفوذ S^g باشد و مجموعه‌ی فعال بیشینه S^* باشد خواهیم داشت:

$$f(S^g) \geq \left(1 - \frac{1}{e}\right) f(S^*)$$

محاسبه‌ی هزینه‌ی زمانی الگوریتم حریصانه:

در هر مرحله که که یک راس به مجموعه‌ی فعال اضافه می‌کنیم هر راس را باید یک بار بررسی کنیم که این برابر با $O(N)$ خواهد بود که برای هر راس نیز، باید با شبیه‌سازی مونت‌کارلو و پیمایش گراف، مقدار نفوذ جانبی را محاسبه کنیم. برای پیمایش گراف نیاز به $O(M)$ که M تعداد یال‌های گراف است نیاز خواهد بود. اگر تعداد تکرار شبیه‌سازی R باشد برای محاسبه‌ی نفوذ جانبی هر راس نیاز به $O(NRM)$ خواهد بود. هم‌چنین این کار را به اندازه‌ی بودجه k باید تکرار کنیم. پس مرتبه‌ی زمانی کل الگوریتم حریصانه برابر با $O(kNRM)$ خواهد بود. هر چقدر که مقدار شبیه‌سازی بیش‌تر باشد نتیجه‌ی بهتری به دست خواهد آمد.

همان‌گونه که دیده می‌شود الگوریتم حریصانه از نظر زمانی بسیار کند است و بالطبع مقیاس‌پذیر هم نیست. با توجه به این که حجم شبکه‌های اجتماعی آنلاین روز به روز در حال افزایش هستند بهینه‌سازی الگوریتم از نظر زمانی بسیار ضروری به نظر می‌رسد، تمرکز این پروژه نیز روی همین قسمت است، اما در ابتدا به معرفی برخی از بهینه‌سازی‌های انجام‌شده روی الگوریتم می‌پردازیم.

Lazy Evaluation:

همان‌طور که از نام این بهینه‌سازی مشخص است، می‌خواهیم برخی از محاسبات که به آن‌ها نیاز نداریم را انجام ندهیم. طبق خاصیت SubModularity می‌دانیم که به ازای مجموعه‌های $S \subseteq T \subseteq V$ و عضوی مثل v که در T نباشد ولی در V باشد،خواهیم داشت : $f(v|S) \geq f(v|T)$. حال فرض کنید $f(w|T)$ را حساب کرده‌ایم و برای یک راس دیگر مثل x که هنوز نفوذ جانبی آن را حساب نکرده‌ایم داشته باشیم $f(x|S) \leq f(w|T)$. با توجه به خاصیت SubModularity خواهیم داشت $f(x|T) \leq f(x|S) \leq f(w|T)$ پس با توجه به این موضوع دیگر لازم نیست که تابع f را برای x محاسبه کنیم. پس برای اضافه‌کردن این خاصیت، یک صف اولویت در نظر می‌گیریم که در آن نفوذ جانبی، راس و دوره‌ای که محاسبه شده‌است را نگه‌داری می‌کنیم و هربار بیش‌ترین نفوذ را انتخاب می‌کنیم، اگر نیاز به محاسبه‌اش نبود عضو بعدی را بر می‌داریم و در غیر این صورت آن را محاسبه می‌کنیم و به صف اضافه می‌کنیم.

طبق شبیه‌سازی‌های انجام شده، این بهینه‌سازی تا ۷۰۰برابر باعث بهبود در محاسبه‌ی انتشار شده است. یک بهینه‌سازی که روی Lazy Evaluation اضافه شده، با عنوان Celf++ مطرح شده است.

Celf++:

فرض کنید Lazy Evaluation در محاسبه‌ی انتشار در نظر گرفته می‌شود. حال کافی است در هر زمانی که می‌خواهیم $f(x|S)$ را محاسبه کنیم و بیشینه‌ی نفوذ جانبی در آن دوره w باشد، $f(x|S \cup \{w\})$ را نیز حساب کنیم زیرا با احتمالی w همان راس انتخابی است و محاسبه‌ی این مقدار هم از نظر زمانی تفاوت چندانی نمی‌کند. اما در دوره‌ی بعدی که w انتخاب شده است، دیگر نیازی به محاسبه برای راس x نیست.

Algorithm 3 LazyGreedy(k, f): general greedy algorithm with lazy evaluations.

Input: k: size of returned set; f: monotone and submodular set function
Output: selected subset

```
1: initialize S ← ∅; priority queue Q ← ∅; iteration ← 1
2: for i = 1 to n do
3:     u.mg ← f(u | ∅); u.i ← 1
4:     insert element u into Q with u.mg as the key
5: end for
6: while iteration ≤ k do
7:     extract top (max) element u of Q
8:     if u.i = iteration then
9:         S ← S ∪ {u}; iteration ← iteration + 1;
10:    else
11:        u.mg ← f(u | S); u.i ← iteration
12:        re-insert u into Q
13:    end if
14: end while
15: return S
```

بهینه‌سازی‌ای که مطرح شد به کاهش محاسبه در انتخاب رئوس کمک می‌کرد. حال روش‌هایی را مرور می‌کنیم که انتشار محاسبه با یک مجموعه‌ی فعال اولیه را سرعت می‌بخشد. این بهینه‌سازی‌ها برای مدل‌های انتشار خاص تعریف شده اند. با توجه به این که این روش‌ها در این پروژه کاربردی نداشتند صرفا به یک معرفی کوتاه بسنده می‌کنیم.

Maximum Influence Arborescence (MIA) برای مدل IC:

در این روش، برای ساده‌سازی محاسبه‌ی انتشار، بر ساختار درخت تاکید شده است. in-Arborescence (درخت جهت‌دار داخلی) به معنی گرافی جهت‌دار است که در حالت بدون جهت، درخت است و جهت تمامی یال‌ها به سمت ریشه است. در بهینه‌سازی MIA یک درخت جهت‌دار داخلی محلی ایجاد می‌کند تا محاسبه‌ی انتشار را روی آن انجام دهد. هر چقدر یک راس غیرفعال از یک راس فعال دورتر باشد (دور بودن به این معنی است که ضرب احتمال معکوس روی یال‌های هر مسیر از راس فعال به راس غیرفعال بیش‌تر باشد) احتمال فعال‌شدن آن کم‌تر می‌شود. $ap(u,S,T)$ را احتمال فعال‌شدن u در درخت جهت‌دار داخلی T و رئوس فعال اولیه‌ی S در نظر می‌گیریم. اگر $u \in S$ باشد آن‌گاه $ap(u,S,T) = 1$ و در غیر این صورت می‌شود $1 - \prod w \in N^{in}(u)(1 - ap(w,S,T).p(w,u))$ یعنی عدم شمول فعال‌نشدن توسط هیچ راس همسایه‌ای در درخت. با توجه به این که در درخت هیچ مسیر تکراری بین دو راس وجود ندارد پس احتمالی را اضافه نشمرده‌ایم. پس در صورتی که درخت داشته باشیم محاسبه‌ی چنین عبارتی بسیار آسان می‌شود. برای ایجاد درخت هم کافی است از راس u به تمامی رئوس دیگر که مسیر احتمال فعال‌شدن آن بیش از یک آستانه‌ی مشخص مثل λ باشد را در نظر می‌گیریم. برای پیداکردن این مسیر، وزن هر یال را برابر با $\log(\frac{1}{p(u,v)})$ در نظر می‌گیریم و با الگوریتم دایسترا، کوتاه‌ترین مسیرها به دست می‌آید که در اصل بزرگ‌ترین ضرب را خواهند داشت. با این تغییرات محاسبه‌ی نفوذ جانبی بسیار سریع‌تر می‌شود.

الگوریتم Simpath برای مدل انتشار LT:

ایده‌ی اصلی این الگوریتم بر پایه‌ی این است که اگر بتوانیم تاثیر هر راس فعال اولیه را نسبت به دیگر رئوس فعال اولیه مستقل در نظر بگیریم می‌توانیم میزان تاثیر هر راس را محاسبه کنیم و به سادگی آن‌ها را جداگانه با هم جمع کنیم. اگر تابع محاسبه‌ی تعداد رئوس تاثیرپذیر را Υ در نظر بگیریم خواهیم داشت $f(S) = \sum_{v \in S} \Upsilon(v)$. برای محاسبه‌ی $\Upsilon(v)$ ابتدا تابع $\Upsilon(v,u)$ را تعریف می‌کنیم که برابر با مجموع احتمال تمام مسیرهایی است که بین v و u وجود دارد. احتمال مسیر را برابر با ضرب احتمال هر یال در نظر می‌گیریم. پیداکردن و جمع‌کردن تمام مسیرهای بین ۲ گره یک پیمایش ساده خواهد بود. با استفاده از تعاریفی که گفته شد، نفوذ جانبی را پیدا می‌کنند و در محاسبه‌ی انتشار سرعت می‌بخشند (چون دیگر از مونت‌کارلو استفاده نمی‌شود). حال روی این روش بهینه‌سازی‌های دیگری مانند Celf++ یا پوشش راسی برای بهبود Simpath استفاده می‌شود.

دو روشی که معرفی شد جزو معروف‌ترین بهینه‌سازی‌ها بودند که به ساختار شبکه توجهی نداشتند، یعنی در هر شبکه‌ای و با هر ساختاری یک روش را در پیش می‌گرفتند. حال بهینه‌سازی‌های مربوط به ساختار انجمنی را بررسی می‌کنیم.

بیشینه‌سازی نفوذپذیری در گراف با ساختار انجمنی تحت مدل LT (با تمرکز روی یک انجمن):

ایده‌ی کلی این مقاله، بیشینه‌کردن نفوذ در یک انجمن با انتخاب تعدادی راس از همان انجمن است تا به این وسیله (فعال‌شدن تمامی یک انجمن) این نفوذ به سایر انجمن‌ها هم پخش شود. تمرکز این مقاله، روی گراف‌های تصادفی اردوش-رینی و مقیاس-آزاد بود که دارای دو انجمن هستند (در صورتی که تعداد انجمن‌ها زیادتر بود مثلا c، آن‌گاه c-1 انجمن را یکی در نظر می‌گرفتند تا به حالت دو انجمن برسد). درست‌کارکردن این الگوریتم وابسته به این نکته بود که با فعال‌شدن یک انجمن به طور کامل و با توجه به این که بین دو انجمن تعدادی یال وجود دارد، این نفوذپذیری را به انجمن دیگر انتقال دهند و به نوعی رئوس فعال اولیه‌ی انجمن دوم را به صورت انتشار نفوذ از انجمن اول به دست می‌اورند.

بیشینه‌سازی نفوذپذیری در گراف با ساختار انجمنی تحت مدل IC (با تمرکز روی محدودکردن جستجو روی یک انجمن):

این روش، همان الگوریتم حریصانه را از نظر زمانی بهبود بخشیده است با توجه به این موضوع که در هنگام محاسبه‌ی تاثیر یک راس فعال در یک انجمن، تنها نفوذ آن در همان انجمن را محاسبه می‌کند و به رئوس دیگر در انجمن‌های دیگر نمی‌پردازد. با این رویکرد، مرتبه‌ی زمانی محاسبه‌ی نفوذ به جای $O(MR)$ که M برابر تعداد یال‌های گراف است، به $O(\sum_{ci \in C} M_i R)$ تبدیل می‌شود که C مجموعه‌ی انجمن‌های فعال است. با توجه به این که این تعداد یال به مراتب از تعداد یال‌های کل گراف کم‌تر است بهبود قابل توجهی در زمان ایجاد خواهد شد. متعاقبا، با کوچک‌شدن فضای جستجو امکان مقیاس‌پذیرشدن الگوریتم نیز ایجاد می‌شود. این الگوریتم از دو قسمت مجزا تشکیل می‌شود، قسمت اول پیداکردن انجمن‌ها در گراف است که با بهره‌گیری از الگوریتم تقریبا خطی[6] محاسبه‌ی انجمن‌ها، آن‌ها را شناسایی و با تعریف یک پارامتر درون انجمنی به نام انتروپی ترکیب[7]، از نسبت تعداد یال‌های درونی یک انجمن نسبت به یال‌های بیرونی آن مطلع می‌شود و در صورت نامناسب‌بودن این نسبت اقدام به ترکیب‌کردن انجمن‌ها می‌کند، این الگوریتم تاثیر زیادی در زمان اجرای کل روند ندارد. قسمت دوم الگوریتم همان الگوریتم حریصانه است که با توجه به این که در محاسبه‌ی نفوذ روی یک انجمن محدود می‌شود سرعت الگوریتم نسبت به حریصانه‌ی معمولی بسیار بالاتر می‌رود. با وجود این که این روش سرعت بسیار بالایی در محاسبه‌ی رئوس فعال اولیه دارد، اما به علت کاهش فضای جستجو، با افت دقت مواجه است. مقاله‌ی فوق این دقت را محاسبه و به صورت ریاضی اثبات کرده است که داریم $f(S) \geq \left(1 - e^{-\frac{1}{1+\Delta d\theta}}\right) f(S^*)$.

[6] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. In Phys.Rev.E76, 2007

[7] CombinationEntropy

Algorithm 1 CGA Algorithm

Input: network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, size of results K, influence speed $\overline{\lambda}$;
Output: $\mathcal{I}$: Top-K influential nodes;

$\mathcal{C} \leftarrow$ detect communities in $\mathcal{G}$;
$M = |\mathcal{C}|$
$\mathcal{I} = \mathcal{I}_1 = \mathcal{I}_2 = ... = \mathcal{I}_M = \emptyset$;
for $k = 1$ to K **do**
    $R[0, k] = 0$; $s[0, k] = 0$;
end for
for $m = 1$ to M **do**
    $R[m, 0] = 0$;
end for
for $k = 1$ to K **do**
    for $m = 1$ to M **do**
        $\Delta R_m = max\{R_m(\mathcal{I} \cup \{v_i\}) - R_m(\mathcal{I}) | v_i \in \mathcal{C}_m\}$;
        $R[m, k] = max\{R[m-1, k], R[M, k-1] + \Delta R_m\}$;
        if $R[m-1, k] \geq R[M, k-1] + \Delta R_m$ **then**
            $s[m, k] = s[m-1, k]$;
        else
            $s[m, k] = m$;
        end if
    end for
    $j = s[M, k]$;
    $v_{max} = \underset{v_i \in \mathcal{C}_j}{\text{argmax}}(R_j(\mathcal{I}_j \cup \{v_i\}) - R(\mathcal{I}_j))$
    $\mathcal{I}_j = \mathcal{I}_j \cup \{v_{max}\}, \mathcal{I} = \mathcal{I} \cup \{v_{max}\}$;
end for

با توجه به این که بهینه‌سازی در مدل LT بیش‌تر رویکرد ریاضی داشت تا الگوریتمی و هم‌چنین تمرکز پروژه‌ی ما روی مدل IC و الگوریتم حریصانه است، از بهینه‌سازی IC برای پیش‌برد ایده گرفتیم. حال با توجه به مطالبی که گفته شد، می‌توانیم ایده‌ی اصلی پروژه را شرح دهیم و سپس نتایج شبیه‌سازی را گزارش کنیم. مشکلاتی که قصد داریم به رفع آن‌ها بپردازیم، امکان مقیاس‌پذیری الگوریتم برای شبکه‌های حجیم و هم‌چنین بهبود دقت در انتخاب رئوس فعال اولیه است.

ایده‌ی اصلی حل مسئله: با استفاده از ایده‌ی مطرح‌شده در بیشینه‌سازی نفوذ در گراف با ساختار انجمنی تحت مدل IC و ترکیب آن با رابطه‌ی بین انجمن‌ها، الگوریتم اصلی پروژه شکل می‌گیرد. به صورت خلاصه روند الگوریتم به صورت زیر است: ابتدا انجمن‌های موجود در گراف را به دست می‌آید. سپس هر انجمن را یک راس (SuperNode) در نظر می‌گیریم و بین هر دو انجمن دو یال جهت‌دار در نظر می‌گیریم که وزن آن، مجموع نرمال‌شده‌ی وزن یال‌های انجمن‌ها به یک‌دیگر است. باقی الگوریتم مانند الگوریتم حریصانه است. در قسمت محاسبه‌ی نفوذپذیری، شبیه‌سازی مونت‌کارلو را روی انجمن رئوس فعال محدود می‌کنیم. یعنی در پیمایش گراف از رئوس فعال، تنها رئوسی که در همان انجمن هستند را در نظر می‌گیریم، اما در مواجهه با هر یال در انتخاب این که به انجمن دیگر برویم یا نه، با آزمایش برنولی با احتمال متناسب با وزن SuperNode های انجمن‌های رئوس آن یال این تصمیم را می‌گیریم. زیرا این احتمال بیان می‌دارد در صورتی که این انجمن فعال شود با چه

احتمالی انجمن دیگر فعال می‌شود. پس به نوعی الگوریتم حریصانه را برای یک مرتبه‌ی بالاتر از گراف داریم انجام می‌دهیم تا در نتیجه‌ی نهایی بهبود حاصل شود. برای پیداکردن انجمن‌های یک گراف از الگوریتم InfoMap [8] استفاده می‌کنیم که ایده‌ی آن مبتنی بر کدکردن گشت‌های تصادفی در گراف به وسیله‌ی کد هافمن است که قابلیت پیداکردن سلسله‌مراتب انجمن‌ها را نیز دارد.

نتایج شبیه‌سازی:

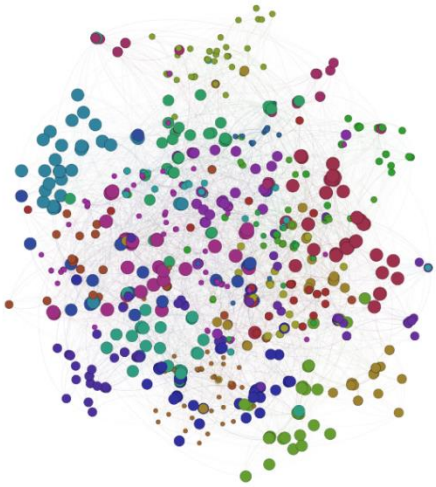
تمامی الگوریتم‌های پیاده‌سازی‌شده روی سیستم‌عامل OSX Yosemit با حافظه‌ی ۸ گیگابایت و پردازنده‌ی ۲,۹ گیگاهرتزی Core i5 اجرا شدند. الگوریتم روی ۴ گراف مختلف اجرا شد که مشخصات گراف‌ها در جدول آمده است. دو گراف wiki و arXiv از شبکه‌های اجتماعی طبیعی به دست آمده‌اند که گراف wiki معادل گرافی با رئوس نویسندگان سایت ویکی‌پدیا است و از یک نویسنده به نویسنده‌ی دیگر یال جهت‌دار وجود دارد اگر و فقط اگر در سایت ویکی‌پدیا به آن نویسنده برای ناظربودن رای داده باشد. گراف arXiv معادل گرافی است که رئوس آن نویسندگان مقالات علمی موجود در arXiv است و به هم یال جهت‌دار دارند اگر مقاله‌ای از راس دوم به مقاله‌ای از راس اول ارجاع داده شده باشد. دو گراف دیگر، گراف‌های ساخته‌شده از بنچمارک LFR[9] است که گراف‌هایی با توزیع Power-Law می‌سازد که دارای جوامع نیز باشند. توزیع احتمال روی یال‌ها به صورت $\frac{1}{di(v)}$ برای هر یال ورودی از راس v است که به Weighted Cascade Probabilities معروف است. برای هر گراف، دو نمودار وجود دارد که یکی زمان اجرا بر تعداد رئوس فعال اولیه و دیگری تعداد رئوس فعال نهایی بر تعداد رئوس فعال اولیه است.

نام گراف	تعداد رئوس	تعداد یال‌ها
arXiv	15233	31398
LFR1000	1000	5554
LFR5000	5000	13350
wiki	7115	103689

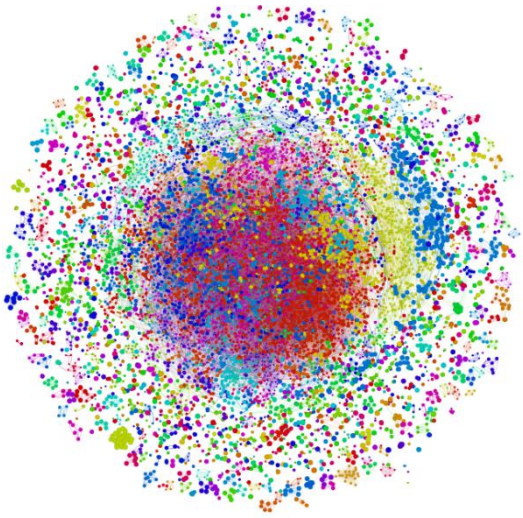
شکل گراف‌های مورد بررسی:

به وسیله‌ی نرم‌افزار Gephi، گراف‌های فوق را پردازش و نمایان کردیم. رنگ هر راس معادل انجمن آن است. مکان رئوس بر حسب یال‌های نزدیک به هم و انجمن آن‌ها توسط الگوریتم OpenORD تعیین شده‌است، همان‌طور که دیده می‌شود بین رئوسی که در یک انجمن هستند، تعداد یال‌ها زیاد است و به هم‌دیگر نزدیک‌تر هستند.

[8] http://mapequation.org/code.html
[9] https://sites.google.com/site/andrealancichinetti/files



گراف LFR-1000



گراف arXiv

Wiki

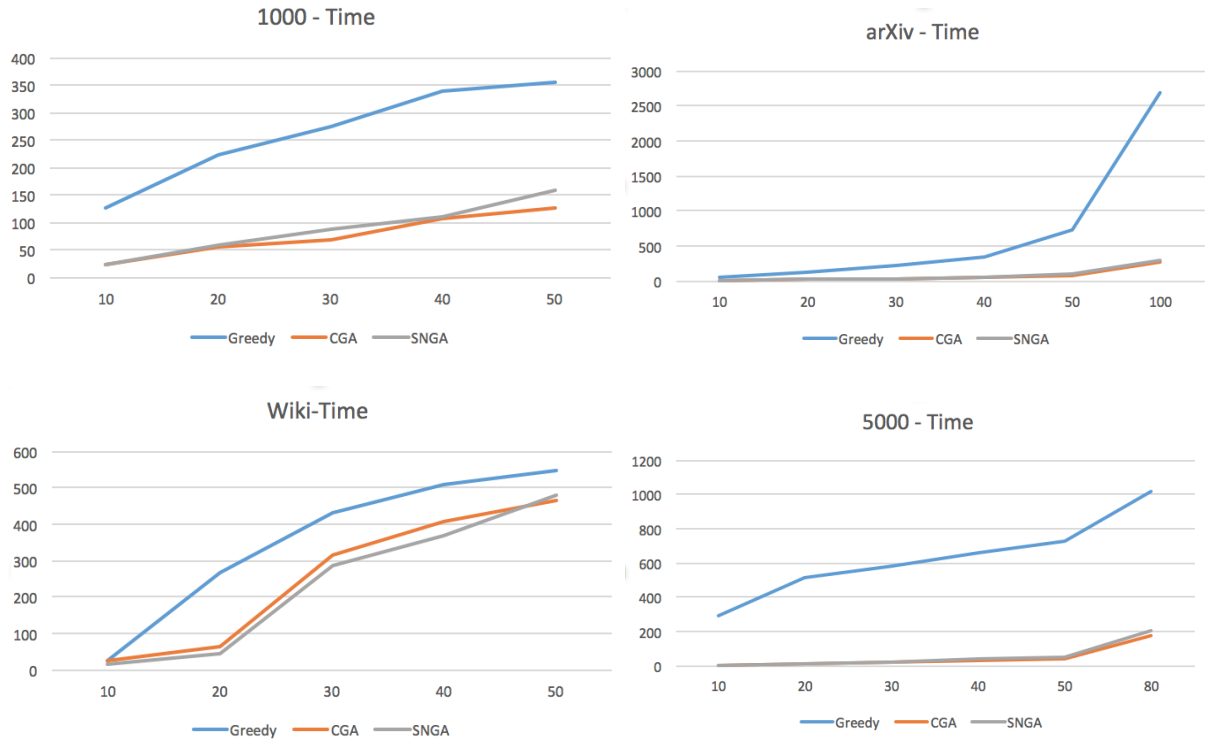
LFR-5000

الگوریتم‌های مورد استفاده: برای مقایسه بین الگوریتم‌ها و بررسی نتایج از ۴ الگوریتم استفاده شده است. الگوریتم پایه الگوریتم Celf++ بوده است که از کدهای آن توسط مبدع این الگوریتم استفاده شد.[10] الگوریتم دوم، الگوریتم CGA بود که از روی الگوریتم ؟ پیاده‌سازی شد. الگوریتم سوم همان الگوریتم ابداعی این پروژه است که با وزن‌دهی بین یال‌های SuperNodeها باعث افزایش فضای جستجو متناسب با احتمال تاثیر روی یک انجمن دیگر می‌شود. الگوریتم چهارم الگوریتم تصادفی است که به صورت اتفاقی رئوس اولیه را انتخاب می‌کند.

مقایسه‌ی زمان:

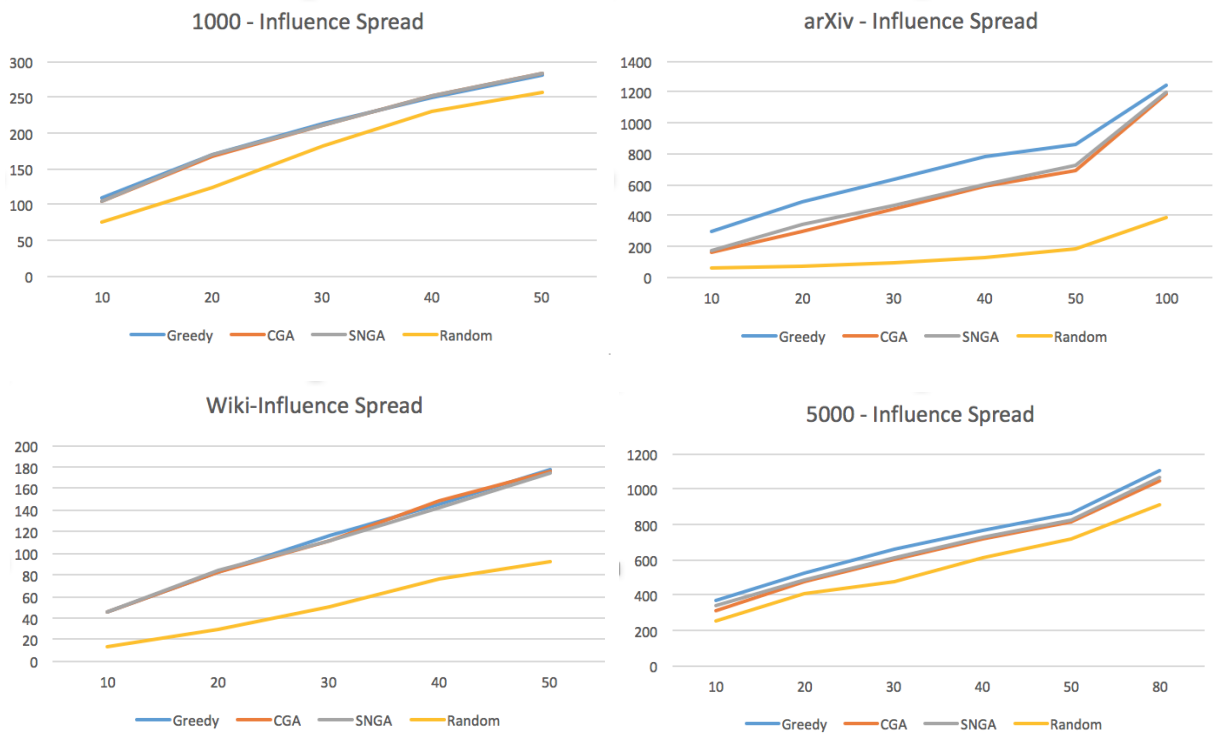
10

طبق نمودارهای زیر که تعداد رئوس فعال اولیه در محور افقی و زمان اجرای الگوریتم بر حسب ثانیه در محور عمودی آمده است، به نظر می‌آید که زمان الگوریتم اول از الگوریتم‌های دوم و سوم با اختلاف زیادی بیش‌تر است حتی در بعضی موارد بیش از ۸ برابر (زمان الگوریتم تصادفی در نمودارها نیامده است زیرا قابل قیاس نبود و یک خط موازی محور افقی بود). با نگاهی به گراف arXiv متوجه تعداد زیاد انجمن‌ها در آن می‌شویم و همین‌طور که در نمودار زمانی آن نیز مشخص است با افزایش رئوس فعال اولیه، نسبت زمان مصرفی الگوریتم اول به دوم و سوم نیز افزایش می‌یابد که علت این موضوع در کاهش شدید محاسبه در شبیه‌سازی مونت‌کارلو است زیرا دیگر نیاز نیست تمامی گراف مورد جستجو قرار بگیرد. عکس این موضوع در گراف wiki وجود دارد که به علت تعداد انجمن زیاد، تاثیر آن بسیار زیاد نیست هر چند که باز هم از نظر زمانی بهبود وجود دارد، حال به قسمت مهم‌تر این شبیه‌سازی می‌پردازیم که تعداد رئوس فعال نهایی است.



مقایسه‌ی رئوس فعال نهایی:

همان‌طور که در نمودارها دیده می‌شود (تعداد رئوس فعال اولیه در محور افقی و رئوس فعال نهایی در محور عمودی) اختلاف خیلی زیادی بین نتایج وجود ندارد و برای مثال در گراف‌های LFR1000 و wiki بسیار تنگاتنگ هم حرکت می‌کنند. اختلاف الگوریتم در گراف arXiv نمایان است که در آن هم حداکثر اختلاف ۷۴ درصد است، که با توجه به بهبود زمانی بسیار بالای آن قابل بحث است، البته همان‌طور که دیده می‌شود با افزایش رئوس فعال اولیه، دقت الگوریتم نیز بیش‌تر می‌شود.



تاثیر الگوریتم حریصانه با بهینه‌سازی SuperNode:

خروجی الگوریتم SNGA در هر گرافی بهتر از الگوریتم CGA بوده است و از نظر زمانی هم افزایشی که داشته است با توجه به بهبودی که نسبت به الگوریتم ساده‌شده داشته است قابل صرف نظر است. البته برای بهترکردن این الگوریتم ایده‌ی دیگری زده شده است که در مرحله‌ی تئوری است و هنوز به صورت عملی بررسی نشده است که در زیر آن را شرح می‌دهیم.

بهینه‌سازی محاسبات انجمن‌ها Lazy Evaluation Communication:

فرض کنید در قسمت شبیه‌سازی مونت‌کارلو الگوریتم حریصانه با بهبود SuperNode می‌خواهیم تاثیر اضافه‌شدن یک راس را به مجموعه‌ی رئوس اولیه محاسبه کنیم. فرض کنید مجموعه‌ی رئوس فعال در همین لحظه حاوی انجمن‌های C1,C2,...,Cp است و راس جدید در انجمن Cj است. در حالت قبلی رئوس تمامی انجمن‌های C1 تا Cp دوباره محاسبه می‌شدند اما می‌دانیم تغییراتی که به وجود خواهد آمد حداکثر در انجمن Cj و همسایه‌های آن است. پس کافی است تنها رئوس فعال در انجمن‌های Cj و با احتمال روی یال‌های این انجمن و انجمن‌های همسایه انجمن‌های دیگر را هم دخیل کنیم و خروجی رئوس فعال نهایی در انجمن‌های دیگر را هر زمانی که محاسبه کردیم در حافظه ذخیره کنیم و از آن استفاده کنیم. اگر این راسی که به دست آوردیم به عنوان راس این دوره انتخاب شد، مقدار خروجی آن انجمن را برای Cj ذخیره می‌کنیم.

نتیجه‌گیری:

همان‌طور که دیدیم، مسئله‌ی بیشینه‌سازی تاثیر در شبکه‌های اجتماعی به یکی از نیازهای بازاریابی است و با مدل‌سازی به صورت مسئله‌ای الگوریتمی در آمده است که با ورودی گرفتن شبکه‌ی اجتماعی و میزان تاثیر هر فرد روی فرد دیگر، تعداد محدودی افراد را مشخص کند که با فعال‌کردن آن‌ها بیش‌ترین تعداد افراد تحت مدل‌های انتشار IC یا LT به دست بیاید. الگوریتم حریصانه، یک الگوریتم تقریبی برای یافتن جواب

این مسئله است و هم‌چنین می‌دانیم پاسخ این مسئله NP است. پس از ارائه‌ی این الگوریتم به علت زمان مصرفی بسیار زیاد آن، بهینه‌سازی‌های زیادی روی این الگوریتم ارائه شد که عموما به ساختار شبکه توجهی نداشتند. در این پروژه با توجه به ساختار انجمنی، الگوریتمی برای بهینه‌سازی زمانی ارائه دادیم که فضای جستجو در شبیه‌سازی مونت‌کارلو را با محدودکردن گراف به یک انجمن کاهش می‌دهد. سپس روی این بهینه‌سازی با توجه به رابطه‌ی بین انجمن‌ها، خود آن‌ها را به صورت یک فرد در نظر گرفتیم که روی افراد (انجمن‌های) دیگر تاثیری متناسب با مجموع احتمال‌های یال‌های بین آن دو راس دارد. با شبیه‌سازی الگوریتم‌ها روی چند گراف مختلف درستی الگوریتم را آزمودیم و از نظر زمانی با الگوریتم‌های دیگر مقایسه کردیم. برای کارهای آینده یک ایده‌ی کاهش محاسبه‌ی انجمن‌ها به وسیله‌ی برنامه‌نویسی پویا ارائه دادیم تا از نظر زمانی بهبود بیش‌تری حاصل شود هم‌چنین برای بهبود نتیجه‌ی نهایی می‌توان دامنه‌ی بررسی را به مرتبه‌های بالاتر انجمن‌ها هم برد.

قدردانی:

از دکتر اسدپور برای زحماتشان در پیش‌برد این پروژه و خانم غیور برای راهنمایی‌های ارزشمندشان برای روشن‌شدن مسیر تحقیق تشکر ویژه دارم.

منابع:

1. W Chen, LVS Lakshmanan, Information and influence propagation in social networks, Morgan and Claypool 2013
2. A Galstyan, V Musoyan, P Cohen, Maximizing influence propagation in networks with community structure, Physical Review E, 2009
3. Y Wang, G Cong, G Song, K Xie,Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, Proceedings of the 16th ACM SIGKDD, 2010
4. D Kempe, J Kleinberg, É Tardos - Maximizing the Spread of Influence through a Social Network, The ninth ACM SIGKDD international , 2003
5. A Lancichinetti, S Fortunato, F Radicchi, Benchmark graphs for testing community detection algorithms, Physical review E, 2008