فرازهایی از کنفرانس SODA 2015

عباس محرابیان

دانشگاه بریتیش کلمبیا

abbas.mehrabian@gmail.com

۲۳ تیر ۱۳۹۴

کنفرانس SODA[۱] که همه‌ساله توسط ACM و SIAM برگزار می‌شود، بعد از FOCS و STOC مهم‌ترین کنفرانس در شاخهٔ الگوریتم به شمار می‌رود. کنفرانس امسال از چهارم تا ششم ژانویه ۲۰۱۵ (چهاردهم تا شانزدهم دی ۱۳۹۳) در شهر سن‌دیگو در امریکا برگزار شد. در کنار آن سه کنفرانس دیگر هم برگزار شد: ANALCO[۲]، ALENEX[۳]، و کارگاهی در زمینهٔ چالش‌های الگوریتمی در یادگیری ماشین(algorithmic challenges in machine learning[۴]). در مورد چند تا از ارائه‌های جالبی که در این کنفرانس‌ها انجام شد خلاصه‌ای نوشتم که در زیر می‌آید. سمیرا دروکی از دانشجویان سابق دانشکدهٔ ریاضی شریف هم مطلبی دوقسمتی دربارهٔ ارائه‌های مربوط به الگوریتم‌های رودخانه‌ای در این کنفرانس نوشته که از [۱۱] قابل دسترسی است.

۱ تولید زیردرخت فراگیر تصادفی.

فرض کنید G گرافی ساده و همبند با n رأس و m یال باشد و $\mathcal{T}$ مجموعهٔ همهٔ زیردرخت‌های فراگیر G باشد. فرض کنید می‌خواهیم یک درخت فراگیر تصادفی از G را با توزیع یکنواخت از $\mathcal{T}$ تولید کنیم یعنی احتمال تولید همهٔ زیردرخت‌های فراگیر با هم برابر باشد. اندازهٔ این مجموعه می‌تواند نمایی باشد (قضیهٔ کیلی بیان می‌کند که تعداد زیردرخت‌های فراگیر گراف کامل $n^{n-۲}$ است) بنابراین حتی ارائهٔ الگوریتمی با زمان اجرای چندجمله‌ای برای این مسئله، ساده نیست.

قضیهٔ زیر روش جالبی برای این کار معرفی می‌کند. یک متحرک تصادفی[۵] ، در واقع یک زنجیر مارکف[۶] است که

[۱] ACM-SIAM Symposium on Discrete Algorithms, http://www.siam.org/meetings/da15

[۲] Analytic Algorithmics and Combinatorics 2015, http://www.siam.org/meetings/analco15

[۳] Algorithm Engineering and Experiments 2015, http://www.siam.org/meetings/alenex15

[۴] http://cseweb.ucsd.edu/~slovett/workshops/algorithmic-machine-learning-2015

[۵] random walk

[۶] Markov chain

مجموعهٔ حالت‌های[۷] آن همان مجموعهٔ رئوس گراف است. این متحرک از رأسی از گراف شروع به حرکت می‌کند و در هر گام به صورت تصادفی به یکی از رئوس مجاور می‌رود که با توزیع یکنواخت از بین همهٔ همسایه‌ها انتخاب شده است.

قضیهٔ ۱ ([۶، ۱]). فرض کنید یک متحرک تصادفی از رأس دلخواه s شروع به حرکت می کند و این کار را تا ابد ادامه می‌دهد. برای هر رأس v از گراف به غیر از s، اولین باری را در نظر بگیرید که متحرک به رأس v رسید، و فرض کنید e_v آن یالی باشد که متحرک از طریقش وارد v شد. در این صورت، زیردرخت فراگیری که مجموعه یال‌های آن $\{e_v : v \in V(G) \setminus s\}$ است، یک عضو تصادفی از $\mathcal{T}$ با توزیع یکنواخت است.

می‌توان رفتار یک متحرک تصادفی را شبیه‌سازی کرد و از این طریق الگوریتمی ساده برای تولید یک زیردرخت فراگیر تصادفی حاصل می‌شود. زمان اجرای این الگوریتم برابر است با زمانی که طول می‌کشد یک متحرک تصادفی همهٔ رئوس گراف را ببیند. این زمان را عدد پوشش[۸] گراف می‌نامیم. اثبات شده است که این عدد حداکثر $O(mn)$ است [۲] بنابراین زمان اجرای این الگوریتم چندجمله‌ای است.

از طرف دیگر، گراف‌هایی وجود دارند که عدد پوشش آن‌ها $\Omega(n^۳)$ است، بنابراین زمان اجرای این الگوریتم ساده نمی‌تواند بهتر از $O(n^۳)$ باشد. در سال ۲۰۰۹ الگوریتمی از مرتبهٔ $O(m\sqrt{n})$ برای این مسئله ارائه شد [۱۸]. ایدهٔ این الگوریتم این بود که در برخی مواقع که متحرک می‌خواست وارد بخشی از گراف بشود که قبلاً دیده بود، این قسمت‌ها را به طور هوشمندانه‌ای از مسیر متحرک حذف می‌کرد و اجازه نمی‌داد متحرک وارد آن بخش بشود. دقت کنید که وقتی متحرک رأسی مثل v را قبلاً دیده است، یال e_v قبلاً مشخص شده است، بنابراین اگر مجدداً وارد v بشود اطلاعات جدیدی دربارهٔ درختی که قرار است تولید شود به دست نمی‌آید!

در SODA امسال الگوریتم جدیدی از مرتبهٔ $O(m^{۴/۳})$ برای این مسئله داده شد [۲۰]. این الگوریتم هم بر ایدهٔ دوباره ندیدن بخش‌های گراف استوار است ولی به مراتب از الگوریتم [۱۸] پیچیده‌تر است. در این مقاله از روش‌های مبتنی بر شبکه‌های الکتریکی و به طور خاص از مفهوم مقاومت مؤثر[۹] استفاده شده است. فرض کنید u و v دو رأس از G باشند. اگر هر یال G را مقاومتی یک اُهمی فرض کنیم و جریانی به اندازهٔ یک آمپر به u تزریق کنیم و از v خارج کنیم، اختلاف ولتاژی که بین رئوس u و v ایجاد می‌شود را مقاومت مؤثر بین u و v می‌نامیم و با $R(u,v)$ نشان می‌دهیم. می‌توان نشان داد که $R(u,v)$ یک فضای متریک روی رئوس G درست می‌کند و ارتباطات جالبی بین این پارامتر با همبندی یالی گراف و هم‌چنین زمان پوشش گراف وجود دارد که در این مقاله از این خواص به طرز هوشمندانه‌ای استفاده شده است.

۲ الگوریتمی کردن اثبات‌های وجودی.

موضوع جالب دیگری که در سال‌های اخیر زیاد کار می‌شود جنبه‌های الگوریتمی Lovász local lemma است. این لم که در روش‌های احتمالاتی بسیار کاربرد دارد از قرار زیر است: فرض کنیم یک فضای احتمال و خانواده‌ای از پیشامدهای «بد» داریم. هم‌چنین فرض کنید احتمال رخ دادن هر یک از پیشامدها حداکثر p است و به علاوه هر کدام از این پیشامدها به حداکثر d پیشامد دیگر وابسته است. اگر $ep(d+۱) \leq ۱$

[۷] state space
[۸] cover time
[۹] effective resistance

آن‌گاه احتمال این که هیچ یک از پیشامدهای بد اتفاق نیفتد مثبت است. این صورت‌بندی که دقیق نیست (وابستگی بین پیشامدها باید تعریف شود)، حالت خاص ولی پرکاربردی از این لم است. برای تعریف دقیق‌تر و حالت کلی لم، فصل پنجم کتاب [۴] را ببینید. توجه کنید که این لم تنها می‌گوید نقطه‌ای در فضای نمونه وجود دارد که خارج از پیشامدهای بد است، ولی در مورد نحوهٔ پیدا کردن آن نقطه حرفی نمی‌زند. این لم در سال ۱۹۷۵ توسط Erdos و Lovász ثابت شد و در سال ۲۰۱۰ نخستین بار الگوریتمی ارائه شد [۲۳] که در زمان چندجمله‌ای این نقطه را پیدا می‌کند. بسیاری از اثبات‌های وجودی که از این لم استفاده می‌کردند به کمک این الگوریتم تبدیل به اثبات‌های ساختنی (در زمان چندجمله‌ای) شدند. علاوه بر این، معلوم شد که این الگوریتم از لم اصلی قوی‌تر است، به این معنا که به کمک این الگوریتم می‌توان حکم‌هایی را ثابت کرد که از حکم‌هایی که به صورت وجودی اثبات می‌شدند قوی‌تر هستند. در مقالهٔ کوتاه [۲۴] که در سال ۲۰۱۳ نوشته شده چند مثال جالب آورده شده و در ادامه دو مثال جدیدتر می‌آوریم.

۱. برای یک عدد طبیعی k، یک k-CNF عبارتی منطقی مثل $A_1 \text{ AND } A_2 \text{ AND } \ldots \ A_m$ است که در آن، هر یک از A_iها به صورت OR تعداد k متغیر است. مثلاً

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_3 \vee x_4)$$

یک ۳-CNF است. یک عبارت منطقی را صدق‌پذیر[۱۰] می‌نامیم اگر بتوان طوری متغیرهای آن را مقداردهی کرد که حاصل کل عبارت «درست» باشد. مثلاً عبارت بالا صدق‌پذیر است چون می‌توان قرار داد

$$x_1 = x_3 = x_4 = \text{TRUE}, x_2 = \text{FALSE}\,.$$

ولی عبارت

$$(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$$

صدق‌پذیر نیست. ثابت شده است که تصمیم‌گیری در مورد این که آیا یک ۳-CNF صدق‌پذیر است یا خیر، NP-complete است (در حقیقت این اولین مسئله‌ای بود که NP-complete بودن آن ثابت شد).

با استفاده از Lovász local lemma به راحتی می‌توان نشان داد که اگر یک عبارت به فرم k-CNF داشته باشیم که هر متغیر حداکثر $2^k/(ke)$ بار در آن ظاهر شده باشد، آن‌گاه آن عبارت صدق‌پذیر است. در مقاله‌ای که در این SODA ارائه شد، نشان داده شد که اگر یک عبارت به فرم k-CNF داشته باشیم که هر متغیر حداکثر

$$\frac{2^{k+1}(1-1/k)^k}{k-1} - \frac{2}{k}$$

بار در آن ظاهر شده باشد، آن‌گاه آن عبارت صدق‌پذیر است [۱۷].

[۱۰] satisfiable

۲. برای یک گراف ساده، منظور از یک رنگ‌آمیزی یالی بی‌دور[۱۱]، رنگ‌آمیزی یال‌هاست به طوری که اولاً هر دو یال مجاوری ناهمرنگ باشند، ثانیاً در هر دور لااقل سه رنگ متمایز ظاهر شده باشد. فرض کنید G گرافی با ماکسیمم درجهٔ Δ باشد. در سال ۱۹۹۱ به کمک Lovász local lemma نشان داده شد که همواره یک رنگ‌آمیزی یالی بی‌دور با حداکثر ۶۴Δ رنگ وجود دارد [۳]. این اثبات، وجودی است یعنی الگوریتم سریعی برای تولید رنگ‌آمیزی ارائه نمی‌کند. نویسندگان مقالهٔ [۳] حدس زدند که برای هر گراف با ماکسیمم درجهٔ Δ یک رنگ‌آمیزی مناسب با $\Delta + ۲$ رنگ وجود دارد. در ANALCO امسال الگوریتمی ارائه شد که یک رنگ‌آمیزی یالی بی‌دور با حداکثر $۳/۷۴\Delta$ رنگ را تولید می‌کند [۱۴]. ایدهٔ اصلی این الگوریتم استفاده از همان الگوریتم [۲۳] و تحلیل مناسبی از آن است.

۳ مدلی ترکیبیاتی برای پخش بیماری‌ها.

فرض کنید G گرافی باشد که یک شبکهٔ اجتماعی را مدل می‌کند، و مدل زیر را برای پخش یک بیماری در نظر بگیرید: در ابتدا زیرمجموعه‌ای از افراد بیمارند. هر کسی که لااقل دو تا از دوستانش بیمار باشند، در دور بعدی بیمار می‌شود، و هر کسی که بیمار شد هیچ‌وقت خوب نمی‌شود. فرض کنید $m(G)$ اندازهٔ کوچک‌ترین مجموعهٔ اولیه از افراد باشد که اگر بیماری از آن‌ها شروع شود، بالاخره همهٔ افراد بیمار شوند. نشان داده شده است که برای هر گراف d-منظم مانند G داریم

$$m(G) \leq \frac{۲n}{d+۱}.$$

به علاوه اگر G از تعدادی گراف کامل مجزای $d+۱$ رأسی تشکیل شده باشد داریم $m(G) \leq \frac{۲n}{d+۱}$. یک سؤال جالب این است که برای چه خانواده‌هایی از گراف‌های d-منظم، کران

$$m(G) \leq O(n/d^{۲}) \qquad (۱)$$

درست است. نویسندگان مقالهٔ [۹]، که در SODA امسال ارائه شد، این پارامتر را بر روی گراف‌های اکسپندر[۱۲] و گراف‌هایی که دور کوتاه ندارند بررسی کردند و نشان دادند برای بسیاری از آن‌ها کران (۱) درست است. در زیر برخی از نتایجی که در این مقاله ثابت شده و هم‌چنین حدس‌هایی که زده شده را می‌آوریم.

در ادامه فرض کنید d یک عدد ثابت و G گرافی d-منظم باشد و

$$\lambda_۱ \leq \lambda_۲ \leq \cdots \leq \lambda_n$$

مقادیر ویژهٔ ماتریس مجاورت گراف G باشند. به راحتی می‌توان دید که $\lambda_n = d$. تعریف کنید

$$\lambda = \max\{|\lambda_۱|, |\lambda_{n-۱}|\}\ .$$

[۱۱] acyclic edge coloring
[۱۲] expander

۱. اگر $\lambda \leq O(\sqrt{d})$ آن‌گاه کران (۱) درست است.

۲. برای هر عدد ثابت $\varepsilon > ۰$، اگر $\lambda < (۱ - \varepsilon)d$ آن‌گاه $m(G) \leq O(n/d\sqrt{d})$ و حدس می‌زنیم که کران (۱) هم درست باشد.

۳. کمر[۱۳] یک گراف را برابر طول کوتاه‌ترین دور آن تعریف می‌کنیم. اگر کمر G از $۲ \ln \ln d$ بزرگ‌تر باشد آن‌گاه کران (۱) درست است.

۴. اگر G هیچ دوری به طول چهار نداشته باشد آن‌گاه $m(G) \leq O(n/d^{۷/۴})$ و حدس می‌زنیم که کران (۱) هم درست باشد.

۵. اگر کمر G از شش بزرگ‌تر باشد آن‌گاه $m(G) \leq O(n/d^{۲} \ln d)$ که تقریباً کران مورد نظر است.

۴ تحلیل‌های الگوریتمی مبتنی بر ساختار ورودی.

وقتی دربارهٔ زمان اجرای یک الگوریتم صحبت می‌کنیم، منظورمان معمولاً زمان اجرا روی بدترین ورودی[۱۴] است. ولی وقتی یک الگوریتم را در عمل اجرا می‌کنیم، اکثر اوقات ورودی‌های الگوریتم ساختار دارند، و این «بدترین ورودی» در عمل هرگز به الگوریتم داده نمی‌شود. در سال‌های اخیر افراد مختلفی تلاش کرده‌اند به صورت ریاضی این وضعیت را مدل کنند و تحلیل‌هایی فراتر از «تحلیل مبتنی بر بدترین ورودی» ارائه کنند؛ سال ۲۰۱۱ کارگاهی با این موضوع برگزار شد[۱۵] و سال ۲۰۱۴ هم درسی با این موضوع در ستنفورد ارائه شد.[۱۶]

یکی از سخنرانان مدعو کنفرانس SODA امسال Claire Mathieu بود که دربارهٔ وضعیت‌هایی صحبت کرد که ورودی الگوریتم «ورودی خوبی» است که چند تغییر تصادفی در آن اعمال شده است. مثلاً اگر شما در Google عبارتی را جستجو کنید ولی چند تا از حروف آن را اشتباه بزنید، این وبسایت خودش حدس می‌زند که منظور شما چه بوده و اشتباهتان را تصحیح می‌کند. در این‌جا «ورودی خوب» همان عبارتی است که منظور شما بوده و یک عبارت صحیح انگلیسی است، ولی ورودی واقعی آن عبارتی است که شما تایپ کرده‌اید و ممکن است یک عبارت صحیح انگلیسی نباشد ولی وبسایت آن را می‌فهمد و جستجوی درست را انجام می‌دهد. چند مثال دیگر که سخنران ذکر کرد در ادامه آمده است.

۱. فرض کنید تعدادی تیم والیبال در یک لیگ با هم رقابت کرده‌اند و هر دو تیم دقیقاً یک بار با هم بازی کرده‌اند. حال می‌خواهیم این تیم‌ها را به بهترین شکل ممکن رتبه‌بندی کنیم. این وضعیت را می‌توان با یک گراف کامل جهت‌دار مدل کرد: هر تیم یک رأس است و از رأس A به رأس B یال داریم اگر A از B برده باشد. (دقت کنید در والیبال مساوی نداریم!) حال می‌خواهیم رئوس را طوری از ۱ تا n رتبه‌بندی کنیم به طوری که تعداد یال‌های «برعکس»

[۱۳] girth
[۱۴] worst-cast input
[۱۵] http://theory.stanford.edu/~tim/bwca/bwca.html
[۱۶] http://theory.stanford.edu/~tim/f14/f14.html

(یعنی از رتبهٔ بیشتر به رتبهٔ کم‌تر) کم‌ترین مقدار ممکن باشد. این مسئله minimum feedback arc set نام دارد و در حالت کلی NP-hard است.

ولی بیایید فرض کنیم این تیم‌ها یک رتبه‌بندی «واقعی» دارند و هر بار که دو تیم بازی می‌کنند، به احتمال دوسوم تیم قوی‌تر می‌برد و به احتمال یک‌سوم تیم ضعیف‌تر می‌برد. در این حالت فرضی، نویسندگان مقالهٔ [۱۹] الگوریتمی با زمان چندجمله‌ای ارائه کردند که به احتمال نزدیک به ۱ رتبه‌بندی واقعی را پیدا می‌کند.

۲. فرض کنید تعدادی شیء داریم که هر دو شیء یا به هم شبیهند یا با هم متفاوتند. می‌خواهیم اشیاء را دسته‌بندی کنیم به طوری که تا حد ممکن اشیاء شبیه هم در یک دسته قرار بگیرند. این وضعیت را هم می‌توان به شکل یک گراف کامل مدل کرد که رئوس آن اشیاء هستند و بین هر دو شیء شبیه یک یال مثبت وجود دارد و بین هر دو شیء متفاوت یک یال منفی وجود دارد. می‌خواهیم رئوس گراف را به چند دسته افراز کنیم به طوری که تعداد یال‌های مثبتی که دو رأس ناهم‌دسته را وصل می‌کنند به علاوهٔ تعداد یال‌های منفی‌ای که دو رأس هم‌دسته را وصل می‌کنند کمینه بشود. این مسئله correlation clustering نام دارد و در حالت کلی NP-hard است.

ولی بیایید فرض کنیم اشیا را واقعاً می‌توان طوری دسته‌بندی کرد که یال‌های درون هر دسته همه مثبت باشند و یال‌های بین دسته‌ها همه منفی باشند، ولی هر یالی به احتمال یک‌سوم علامتش برعکس شده باشد. نویسندگان مقالهٔ [۲۲] الگوریتمی ارائه دادند که در صورتی که ورودی چنین ساختاری داشته باشد و به علاوه اندازهٔ همهٔ دسته‌ها لااقل $\Omega(\sqrt{n})$ باشد، آن‌گاه دسته‌بندی بهینه را به احتمال بالا پیدا می‌کند.

۳. یک مجموعهٔ مستقل[۱۷] در یک گراف، مجموعه‌ای از رئوس است که بین هیچ دو رأسی یالی وجود نداشته باشد. مسئلهٔ پیدا کردن بزرگ‌ترین مجموعهٔ مستقل در یک گراف در حالت کلی NP-hard است ولی برای گراف‌های مسطح این مسئله PTAS دارد، یعنی برای هر $\varepsilon > 0$ می‌توان یک $(1+\varepsilon)$-تقریب برای مسئله را در زمان چندجمله‌ای پیدا کرد. نویسندگان مقالهٔ [۲۱] نشان دادند که اگر ورودی گرافی مسطح باشد که δn یال به آن اضافه شده است، هم‌چنان پیدا کردن بزرگ‌ترین مجموعهٔ مستقل PTAS دارد.

۵ الگوریتم‌های رودخانه‌ای

یکی از چالش‌هایی که شرکت‌های کامپیوتری در دنیای امروز با آن مواجهند حجم بسیار زیاد داده‌هاست که روزبه‌روز هم بیشتر می‌شود و الگوریتم‌های تازه‌ای برای کار با آن‌ها می‌طلبد. به نظر می‌رسد که مدل محاسباتی کلاسیک ماشین تورینگی و تقسیم‌بندی مسائل به دو دستهٔ «چندجمله‌ای حل‌پذیر»[۱۸] و غیر آن در چنین شرایطی پاسخ‌گو نیست و در سال‌های اخیر انواع و اقسام دیدگاه‌های الگوریتمی برای کار با داده‌های عظیم مطرح شده‌اند.

یکی از مباحثی که اخیراً زیاد کار می‌شود و در SODA امسال هم چندین ارائه در موردش داده شد، الگوریتم‌های رودخانه‌ای[۱۹] است. فرض کنید یک گراف داریم که انقدر عظیم است که در حافظه جا نمی‌شود بلکه نحوه‌ای که به آن

[۱۷] independent set
[۱۸] polynomially solvable
[۱۹] streaming algorithms

دسترسی داریم اینست که اطلاعات مربوط به آن مثل یک رودخانه از بالای کوهی به پایین می‌آیند و ما هم کنار رودخانه نشسته‌ایم و می‌توانیم اطلاعات را یکی‌یکی ببینیم ولی نمی‌توانیم همه‌اش را ذخیره کنیم: باید برخی اطلاعات را انتخاب کنیم و ذخیره کنیم و بقیه را بفرستیم برود.

به صورت دقیق‌تر می‌توان مدل را به صورت زیر بیان کرد: گراف n رأس دارد و تعداد رئوس و یال‌های آن بر ما معلوم است. یک نفر همهٔ یال‌های گراف را می‌داند و آن‌ها را به ترتیبی که دلش می‌خواهد برای ما می‌خواند. ما هم حافظهٔ محدودی داریم و می‌توانیم بعضی از یال‌ها را یادداشت کنیم (هر یال دقیقاً یک بار خوانده می‌شود). حال با این وضعیت می‌خواهیم سؤالی را دربارهٔ گراف پاسخ دهیم مثلاً می‌خواهیم بفهمیم دور اویلری دارد یا خیر؛ یا می‌خواهیم پارامتری از گراف را اندازه‌گیری کنیم. پرسش این است که برای این کار به چقدر حافظه احتیاج داریم و چقدر طول می‌کشد تا به سؤال جواب بدهیم.

دو نمونه از نتایجی که در SODA امسال ارائه شد را می‌آوریم.

۱. یک تطابق[۲۰] در یک گراف، مجموعه‌ای از یال‌هاست که هیچ دو یالی با هم مجاور نیستند. مسئلهٔ پیدا کردن بزرگ‌ترین تطابق در یک گراف از معروف‌ترین مسائل گراف است که در زمان چندجمله‌ای قابل حل است [۱۲]. نویسندگان مقالهٔ [۱۳] الگوریتمی جریانی برای گراف‌های مسطح ارائه کردند که از حافظهٔ $O(n^{۲/۳})$ استفاده می‌کند و در زمان چندجمله‌ای یک تطابق پیدا می‌کند که هر تطابق دیگری در گراف حداکثر ۲۵ برابر این تطابق است.

۲. مسئلهٔ یافتن کوچک‌ترین پوشش رأسی[۲۱] از دیگر مسائل کلاسیک گراف است. یک پوشش رأسی مجموعه‌ای از رئوس است که هر یالی لااقل یک سرش در این مجموعه باشد. فرض کنید k یک عدد طبیعی دلخواه باشد که می‌تواند تابعی از n باشد. نویسندگان مقالهٔ [۸] الگوریتمی جریانی ارائه دادند که از حافظهٔ $O(k^{۲})$ استفاده می‌کند، و در زمان $۲^{O(k)}$ یا یک پوشش با k رأس برای گراف معرفی می‌کند و یا (به درستی) اعلام می‌کند که گراف چنین پوششی ندارد. به چنین الگوریتم‌هایی که زمان اجرایشان به اندازهٔ خروجی بستگی دارد، الگوریتم‌های پارامتریزه‌شده[۲۲] گفته می‌شود. برای اطلاعات بیشتر در مورد آن‌ها خود مقالهٔ [۸] را ببینید.

۶ تولید گرافی تصادفی با دنباله درجات داده شده.

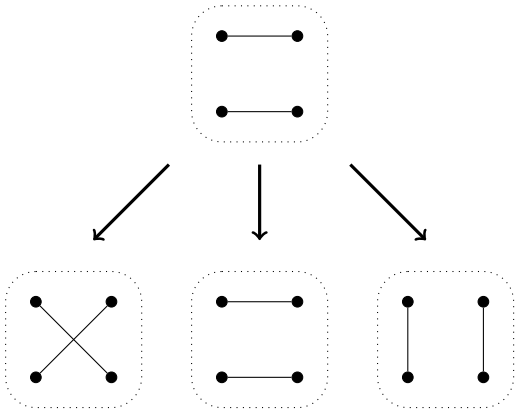
فرض کنید $\mathbf{d} = (d_۱, d_۲, \ldots, d_n)$ یک دنبالهٔ گرافی باشد، به این معنا که لااقل یک گراف وجود دارد که دنبالهٔ درجاتش $\mathbf{d}$ است. می‌خواهیم یک گراف را به صورت تصادفی و با توزیع یکنواخت از میان تمام گراف‌هایی که دنباله درجاتشان $\mathbf{d}$ است تولید کنیم. این کار را در چه زمانی می‌توان انجام داد؟! فرض کنید $G(\mathbf{d})$ مجموعهٔ همهٔ گراف‌هایی باشد که دنبالهٔ درجاتشان $\mathbf{d}$ است. توجه کنید که اندازهٔ $G(\mathbf{d})$ در حالت کلی برحسب n نمایی است بنابراین حتی ارائهٔ الگوریتمی با زمان اجرای چندجمله‌ای برای این کار، ساده نیست.

---

[۲۰] matching

[۲۱] finding minimum vertex cover

[۲۲] parameterized algorithm



شکل ۱: گذارهای switch Markov chain

یکی از روش‌هایی که برای تولید یک گراف تصادفی با دنباله درجات داده شده وجود دارد، استفاده از زنجیرهای مارکف است. فرض کنید از گراف دلخواهی با دنباله درجات $\mathbf{d}$ شروع می‌کنیم، و تعداد زیادی تغییرات تصادفی کوچک روی گراف انجام می‌دهیم به طوری که دنباله درجات همیشه ثابت باشد، و امید داریم که گرافی که به دست می‌آوریم، یک گراف تصادفی با توزیع یکنواخت از $G(\mathbf{d})$ باشد. این فرایند را می‌توان به صورت زنجیر مارکفی که مجموعهٔ حالات آن $G(\mathbf{d})$ و توزیع پایای[۲۳] آن توزیع یکنواخت است، طراحی کرد.

یک زنجیر مارکفی که برای این کار استفاده می‌شود، switch Markov chain نام دارد که بسیار ساده است: در هر گام، دو یال متمایز تصادفی مثل ab و cd که مجاور هم نیستند انتخاب می‌شوند و از گراف حذف می‌شوند، سپس یکی از سه جفت یال زیر به تصادف انتخاب می‌شود و به گراف اضافه می‌شود: جفت $\{ab, cd\}$ یا جفت $\{ac, bd\}$ و یا جفت $\{ad, bc\}$ (شکل ۱ را ببینید). دقت کنید که با انجام این کار دنبالهٔ درجات گراف تغییری نمی‌کند. به راحتی می‌توان نشان داد که توزیع یکنواخت برای این زنجیر مارکف پایاست.

توجه کنید که هر چقدر در این زنجیر مارکف حرکت کنیم به توزیع واقعاً یکنواخت نمی‌رسیم ولی هدف اینست که به توزیعی برسیم که بسیار به توزیع یکنواخت نزدیک است. فاصلهٔ بین دو توزیع π و σ روی مجموعهٔ Ω را به صورت زیر تعریف می‌کنیم:

$$\text{distance}(\pi, \sigma) = \max\{|\pi(A) - \sigma(A)| : A \subseteq \Omega\}$$

که همواره عددی بین ۰ و ۱ است. فرض کنید $d_1 \leq d_2 \leq \cdots d_n$ و $M = \sum_{i=1}^{n} d_i$. برای دنباله درجات $(d_1, d_2, \ldots, d_n)$، می‌گوییم زنجیر مارکف معرفی‌شده در بالا سریع است اگر هنگامی که روی زنجیر مارکف شروع به حرکت می‌کنیم تا زمانی که طول می‌کشد تا فاصله‌مان تا توزیع یکنواخت از $1/4$ کم‌تر شود برحسب n حداکثر چندجمله‌ای باشد. در سال ۲۰۰۵ نشان داده شد که اگر $d_1 = d_2 = \cdots = d_n$، آن‌گاه این زنجیر مارکف سریع است [۱۰]. در SODA امسال، نویسندهٔ مقالهٔ [۱۶] این نتیجه را قوی‌تر کرد و نشان داد که اگر $1 \geq d_1$ و $3 \leq d_n \leq \sqrt{M}/4$ آن‌گاه

[۲۳] stationary distribution

این زنجیر مارکف سریع است.

۷ مسئلهٔ پیش‌بینی وضعیت هوا. فرض کنید می‌خواهیم برای T روز متوالی پیش‌بینی کنیم که باران می‌بارد یا خیر. برای هر $۱ \leq t \leq T$، صبح روز tاُم که بیدار می‌شویم، n کارشناس نظرشان را به ما می‌گویند: هر یک از آن‌ها می‌گوید که به نظرش به چه احتمالی باران می‌بارد. پس از شنیدن نظرات آن‌ها، ما پیش‌بینی می‌کنیم که به احتمال p_t باران می‌بارد. قرار می‌دهیم $y_t = ۱$ اگر در این روز باران ببارد، و در غیر این صورت $y_t = ۰$. مقدار خطای[۲۴] این روز ما که در پایان روز تعیین می‌شود برابر است با $|p_t - y_t|$ و مقدار (کل) خطای ما را به صورت

$$L_p = \sum_{t=۱}^{T} |p_t - y_t|$$

تعریف می‌کنیم. می‌خواهیم این کمیت را کمینه کنیم.

خوب است کمی دربارهٔ علت انتخاب این تابع خطا توضیح دهیم. فرض کنیم به جای این که ما عدد p_t را به عنوان خروجی بدهیم، سکه‌ای می‌اندازیم و به احتمال p_t می‌گوییم باران می‌بارد، و به احتمال $۱ - p_t$ می‌گوییم باران نمی‌بارد. در این صورت $|p_t - y_t|$ دقیقاً احتمال این است که ما در پیش‌بینی خود خطا کنیم. بنابراین

$$\sum_{t=۱}^{T} |p_t - y_t|$$

دقیقاً امید ریاضی تعداد روزهایی است که ما اشتباه پیش‌بینی کرده‌ایم.

فرض کنیم که هیچ چیزی دربارهٔ توزیع $(y_۱, \ldots, y_T)$ نمی‌دانیم. اگر این بردار به صورت یکنواخت از $\{۰, ۱\}^n$ انتخاب بشود، هر الگوریتمی که برای تولید p_tها بدهیم، به احتمال بالا مقدار خطای آن دست‌کم $T/۲ - O(\sqrt{T})$ است (طبق قضیهٔ حد مرکزی)؛ از طرفی به راحتی می‌توان الگوریتمی داد که خطای آن از $T/۲$ بیشتر نباشد ($p_t = ۱/۲$ برای همهٔ tها)، لذا مسئلهٔ کمینه کردن مقدار کل خطا خیلی جالب نیست.

مسئله وقتی جالب می‌شود که پیش‌بینی‌های خودمان را با پیش‌بینی‌های کارشناسان مقایسه کنیم. مثلاً این طور نگاه کنیم که هر یک از کارشناسان به یک سری اطلاعاتی دسترسی دارند که از ما مخفی است (مثل تصاویر ماهواره‌ای از جو)، و با داشتن این اطلاعات می‌توانند پیش‌بینی‌هایی بهتر از «پیش‌بینی کاملاً تصادفی» انجام دهند. (مثلاً مسئلهٔ تصمیم‌گیری در مورد این که قیمت سهام یک شرکت بالا می‌رود یا پایین می‌آید را در نظر بگیرید. در این مسئله، کارشناسی ممکن است به اطلاعات درونی شرکت دسترسی داشته باشد و پیش‌بینی‌اش را بر اساس آن انجام دهد.) در چنین چهارچوبی، آیا ما می‌توانیم الگوریتمی ارائه کنیم که خطایمان از خطای کارشناسان خیلی بدتر نباشد؟! این مسئله سخت است چون در ابتدا نمی‌دانیم کدام کارشناس بهتر از بقیه است، هم‌چنین ممکن است یک کارشناس در اوایل خوب پیش‌بینی کند ولی در ادامه کارشناس دیگری بهتر پیش‌بینی کند.

---

[۲۴] loss

این مسئله در نظریهٔ یادگیری[۲۵] بسیار بررسی شده و الگوریتم‌های زیادی برای آن ارائه شده است. خطای کل کارشناسان را با $L_۱, L_۲, \ldots, L_n$ نشان می‌دهیم. نویسندگان مقالهٔ [۷] الگوریتمی ارائه کردند که کران

$$L_p \leq \min_{i=۱,\ldots,n} L_i + \sqrt{\frac{T \ln(n+۱)}{۲}} + \frac{\log_۲(n+۱)}{۲}$$

را برای هر دنبالهٔ دلخواه $y_۱, \ldots, y_T$ برآورده می‌کند. یعنی مستقل از این که وضع هوا در چند روز مورد نظر چه باشد، خطای این الگوریتم حداکثر به اندازهٔ $\sqrt{T \ln(n)}$ بیشتر از خطای بهترین کارشناس است. هم‌چنین نشان داده شد که وقتی $n, T \to \infty$، الگوریتمی وجود ندارد که در حالت کلی کران بهتری بدهد.

قالب کلی این الگوریتم ساده است: به هر یک از کارشناسان وزنی اختصاص داده می‌شود، کارشناسی که درست پیش‌بینی می‌کند وزنش بیشتر می‌شود، و کارشناسی که اشتباه پیش‌بینی می‌کند وزنش کم‌تر می‌شود. پیش‌بینی هر روز بر اساس میانگین وزن‌داری از پیش‌بینی‌های کارشناسان در آن روز انجام می‌شود. قسمت کلیدی نحوهٔ تغییر دادن وزن کارشناسان با توجه به پیش‌بینی‌های آن‌هاست. این الگوریتم بسیار جالب (به همراه نحوهٔ درست تغییر وزن‌ها)، بعدها کاربردهای دیگری هم یافت و به multiplicative weights update method شهرت یافت. برای توضیحات بیشتر و سایر کاربردها مقالهٔ [۵] را ببینید.

در مقاله‌ای که در کارگاه «چالش‌های الگوریتمی در یادگیری ماشین» امسال ارائه شد، نویسندگان کران‌های بهتر و بهینه‌ای برای حالت‌های $n = ۲, ۳$ ثابت کردند [۱۵].

۸ الگوریتم «نامساوی‌ثابت‌کن».

فرض کنید $x_۱, \ldots, x_n, y_۱, \ldots, y_n$ اعداد حقیقی مثبت باشند. نامساوی کوشی-شوارز می‌گوید

$$\left(\sum_j x_j\right)^{۱/۲} \left(\sum_j y_j\right)^{۱/۲} \left(\sum_j \sqrt{x_j y_j}\right)^{-۱} \geq ۱ .$$

نامساوی هولدر تعمیم این نامساوی است که می‌گوید برای هر $p \in (۰, ۱)$ داریم

$$\left(\sum_j x_j\right)^{p} \left(\sum_j y_j\right)^{۱-p} \left(\sum_j x_j^p y_j^{۱-p}\right)^{-۱} \geq ۱ . \qquad (۲)$$

یک نامساوی جالب دیگر از نامساوی‌های بین نُرم‌ها تولید می‌شود. فرض کنید $۱ \leq a \leq b$. در این صورت داریم

$$\left(\sum_j x_j^a\right)^{۱/a} \geq \left(\sum_j x_j^b\right)^{۱/b} .$$

در نتیجه برای هر $p \in [۰, ۱]$ داریم

$$\left(\sum_j x_j^p\right) \left(\sum_j x_j\right)^{-p} \geq ۱ . \qquad (۳)$$

[۲۵] learning theory

یک سؤال جالب این است که تا چه حد می‌توان نامساوی‌های (۲) و (۳) را تعمیم داد. به طور خاص، این پرسش را در نظر بگیرید: کدام دنباله‌های $a_1, \ldots, a_r, b_1, \ldots, b_r, c_1, \ldots, c_r$ این خاصیت را دارند که برای هر دو دنبالهٔ $z_1, \ldots, z_n, w_1, \ldots, w_n$ از اعداد حقیقی مثبت، نامساوی

$$\prod_{i=1}^{r} \left(\sum_j w_j^{a_i} z_j^{b_i} \right)^{c_i} \geq 1 \qquad (۴)$$

برقرار است؟ نویسندگان مقالهٔ [۲۵]، که در کارگاه «چالش‌های الگوریتمی در یادگیری ماشین» امسال ارائه شد، نشان دادند این خاصیت تنها در صورتی وجود دارد که نامساوی (۴) اصولاً حاصل‌ضربی از (توان‌های مثبت) نامساوی‌های (۲) و (۳) باشد، که در آن‌ها x_jها و y_jها توان‌هایی از w_kها و z_kها هستند. هم‌چنین آن‌ها الگوریتمی (ساده و پیاده‌سازی‌شده در MATLAB) ارائه کردند که ورودی‌اش $a_1, \ldots, a_r, b_1, \ldots, b_r, c_1, \ldots, c_r$ است و اگر این دنباله خاصیت مورد نظر را داشته باشد، (۴) را برحسب نامساوی‌هایی به شکل (۲) و (۳) می‌نویسد، درغیر این صورت مثال نقضی ارائه می‌کند.

مراجع

[1] Aldous, D. J. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.*, 3(4):450–465, 1990. http://epubs.siam.org/doi/pdf/10.1137/0403039.

[2] Aleliunas, R., Karp, R. M., Lipton, R. J., Lovász, L., and Rackoff, C. Random walks, universal traversal sequences, and the complexity of maze problems. FOCS'79. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4568017.

[3] Alon, N., McDiarmid, C., and Reed, B. Acyclic coloring of graphs. *Random Structures Algorithms*, 2(3):277–288, 1991.

[4] Alon, N. and Spencer, J. H. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., Hoboken, NJ, third ed. , 2008. With an appendix on the life and work of Paul Erdős.

[5] Arora, S., Hazan, E., and Kale, S. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8:121–164, 2012. https://www.cs.princeton.edu/∼arora/pubs/MWsurvey.pdf.

[6] Broder, A. Generating random spanning trees. FOCS'89. https://www.cs.cmu.edu/∼15859n/RelatedWork/Broder-GenRanSpanningTrees.pdf.

[7] Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., and Warmuth, M. K. How to use expert advice. *J. ACM*, 44(3):427–485, 1997. http://homes.di.unimi.it/cesa-bianchi/Pubblicazioni/jacm-97a.pdf.

[8] Chitnis, R., Cormode, G., Hajiaghayi, M., and Monemizadeh, M. Parameterized streaming: Maximal matching and vertex cover. SODA'15. http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.82.

[9] Coja-Oghlan, A., Feige, U., Krivelevich, M., and Reichman, D. Contagious sets in expanders. SODA'15. http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.131.

[10] Cooper, C., Dyer, M., and Greenhill, C. Sampling regular graphs and a peer-to-peer network. SODA'05. http://www.comp.leeds.ac.uk/dyer/papers/cdg07.pdf.

[11] Daruki, Samira. Streaming @ soda: Part ii, 2015. [Online; accessed 14-July-2015].

[12] Edmonds, J. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[13] Esfandiari, H., Hajiaghayi, M. T., Liaghat, V., Monemizadeh, M., and Onak, K. Streaming algorithms for estimating the matching size in planar graphs and beyond. SODA'15. http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.81.

[14] Giotis, I., Kirousis, L., Psaromiligkos, K. I., and Thilikos, D. M. On the algorithmic Lovász local lemma and acyclic edge coloring. ANALCO'15. http://arxiv.org/abs/1407.5374.

[15] Gravin, N., Peres, Y., and Sivan, B. Towards optimal algorithms for prediction with expert advice. preprint, 2014, http://arxiv.org/abs/1409.3040.

[16] Greenhill, C. The switch markov chain for sampling irregular graphs. SODA'15. http://arxiv.org/abs/1412.5249.

[17] Harris, D. G. Lopsidependency in the Moser-Tardos framework: Beyond the lopsided Lovász local lemma. SODA'15. http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.120.

[18] Kelner, J. A. and Madry, A. Faster generation of random spanning trees. FOCS'09. http://arxiv.org/abs/0908.1448.

[19] M. Braverman, E. Mossel. Sorting from noisy information. 2009. preprint, http://arxiv.org/abs/0910.1191.

[20] Madry, A., Straszak, D., and Tarnawski, J. Fast generation of random spanning trees and the effective resistance metric. SODA'15. http://arxiv.org/abs/1501.00267.

[21] Magen, A. and Moharrami, M. Robust algorithms for max independent set on minor-free graphs based on the Sherali-Adams hierarchy. APPROX/RANDOM'09. http://www.cs.toronto.edu/∼avner/papers/SA-planar.pdf.

[22] Mathieu, C. and Schudy, W. Correlation clustering with noisy input. SODA'10. http://cs.brown.edu/people/claire/Publis/corrclustering.pdf.

[23] Moser, R. A. and Tardos, G. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010. http://arxiv.org/abs/0903.0544.

[24] Szegedy, M. The Lovász local lemma – a survey. in Bulatov, A. A. and Shur, A. M., eds. , *Computer Science – Theory and Applications*, vol. 7913 of *Lecture Notes in Computer Science*, pp. 1–11. Springer Berlin Heidelberg, 2013.

[25] Valiant, G. and Valiant, P. An automatic inequality prover and instance optimal identity testing. FOCS'14. http://theory.stanford.edu/∼valiant/papers/instanceOpt.pdf.