فرهنگ و اندیشهٔ ریاضی
سال ۴۰، شمارهٔ ۶۹ (پاییز و زمستان ۱۴۰۰) صص. ۱ تا ۲۲

الگوریتم‌های تقریبی برای بازسازی درخت تبارزایشی: کاربردهایی از علوم نظری کامپیوتر در زیست‌شناسی و بیوانفورماتیک

محمدهادی فروغمنداعرابی

چکیده

مسئلهٔ استنتاج درخت تبارزایشی، مسئله‌ای قدیمی در زیست‌شناسی است که در آن به دنبال درختی هستیم که شباهت موجودات را نشان دهد. الگوریتم‌های موجود برای بازسازی درخت تبارشناسی عموماً الگوریتم‌هایی اکتشافی هستند. این الگوریتم‌ها مبتنی‌بر فهم و شهود ابداع‌کنندهٔ آن‌ها هستند و در موردِ نحوه و میزان بهینه بودن آن‌ها هیچ تضمینی وجود ندارد. درمقابل، الگوریتم‌های تقریبی اگرچه جواب بهینه را پیدا نمی‌کنند (چون احتمالاً این کار امکان‌پذیر نیست)، در موردِ میزان فاصلهٔ جواب آن‌ها با جواب بهینه می‌توان محدوده‌ای مشخص کرد. در این مقاله، الگوریتمی تقریبی برای مسئلهٔ بازسازی درخت تبارشناسی تومور را بررسی می‌کنیم. این الگوریتم با تغییراتی در الگوریتمی برای مسئلهٔ درخت اشتاینر به دست می‌آید که پیش‌ازاین در [۱] مطرح شده است. همچنین، یکی از کاربردهای علوم نظری کامپیوتر را در طراحی الگوریتم برای مسئله‌های بیوانفورماتیک بررسی خواهیم کرد.

۱. مقدمه

از دیرباز زیست‌شناسان به طبقه‌بندی موجودات زنده پرداخته‌اند. آن‌ها معمولاً این طبقه‌بندی را براساس ویژگی‌های ظاهری موجودات زنده انجام می‌دادند. هدف اصلی از طبقه‌بندی موجودات،

عبارات و کلمات کلیدی. بازسازی درخت تبارزایشی، مسئلهٔ طبقه‌بندی، کاربرد علوم کامپیوتر در بیوانفورماتیک.

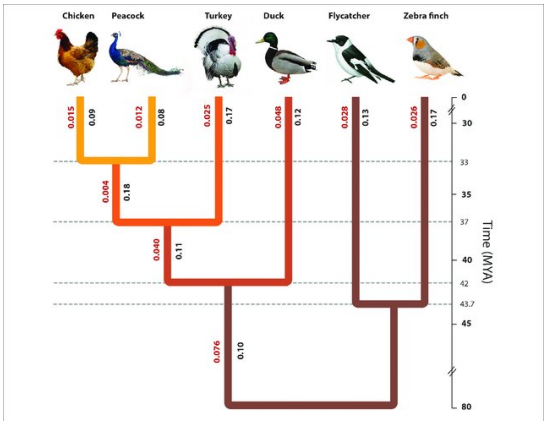
داشتن دسته‌بندهایی از شباهت‌های بین موجودات بود و شاید این طبقه‌بندی کاربرد جدّی دیگری نداشت. سنت حسنهٔ طبقه‌بندی موجودات زنده تاکنون در بین زیست‌شناسان باقی‌مانده است اما کیفیت فرآیند طبقه‌بندی، داده‌هایی که براساس آن‌ها طبقه‌بندی انجام می‌شود و حتی تفسیری که در حال حاضر از یک طبقه‌بندی صورت می‌گیرد با گذشته بسیار متفاوت است.

در طبقه‌بندی موجودات زنده، هدفْ رسم یک درخت وزن‌دار و ریشه‌دار به نام «درخت تبارزایشی»[۱] (یا دودمانی) است. این شیء ترکیبیاتی در مبحث نظریهٔ گراف تعریف می‌شود. به‌صورتِ خلاصه یک «درخت» ریشه‌دار شامل تعدادی رأس است که یکی از آن‌ها به‌عنوان ریشه مشخص شده است. هرکدام از رأس‌ها ممکن است هیچ یا تعدادی فرزند از میان دیگر رأس‌ها داشته باشند. رأسِ ریشه، فرزند هیچ رأس دیگری نیست و بقیه رأس‌ها دقیقاً فرزند یکی از دیگر رأس‌ها هستند. به رأس‌هایی که هیچ فرزندی ندارند اصطلاحاً برگ درخت گفته می‌شود. برای نمایش درخت، هرکدام از رأس‌ها با خطوطی، که «خطوط یال» نام دارند، به رأس‌های فرزندانشان وصل می‌شوند. در یک درخت وزن‌دار، به هرکدام از یال‌ها یک عدد به نام «وزن یال» نسبت داده می‌شود.

فرض کنید درحالِ مطالعهٔ گروهی از موجودات زنده هستیم. این گروه از موجودات زنده ممکن است تعدادی موجود شبیه به‌هم مانند ماهی‌ها، و یا تعدادی موجود بسیار متفاوت، مانند تمامی موجودات زنده، باشند. هدف از طبقه‌بندی موجودات زنده رسم یک درخت ریشه‌دار است که رأس‌های آن شامل موجودات زندهٔ مورد مطالعه باشد. انتظار داریم که رأس‌هایی که در این درخت به‌یکدیگر نزدیک‌ترند، موجوداتی با شباهت بیشتر و رأس‌هایی که از یکدیگر دورترند، موجوداتی با شباهت کمتر را نشان دهند. طبیعتاً این درخت رأس‌های میانی نیز دارد. هر رأس میانی را که درنظر بگیریم، تعدادی از رأس‌ها از نوادگان آن هستند و احتمالاً تعدادی از رأس‌ها نیز از نوادگان آن نباشند. بدین صورت هر رأس را می‌توان به‌صورت یک گروه‌بندی از موجودات درنظر گرفت. برای مثال رأسِ ریشه، نمایندهٔ کل گروه مورد مطالعه است. مثالی از یک طبقه‌بندی برای برخی پرندگان در شکل ۱ نمایش داده شده است. همان‌طورکه در شکل مشخص است، در این درخت، مرغ به طاووس نزدیک‌تر است تا به مرغابی!

زیست‌شناسان کهن، درختِ طبقه‌بندی موجودات زنده را براساس شباهت‌های ظاهری و یا رفتاری آن‌ها رسم می‌کردند. اما امروز داده‌هایی که برای رسم درخت طبقه‌بندی از آن‌ها استفاده می‌شود، داده‌های ژنتیکی موجودات زنده است. این داده‌ها، که از طریق روش‌های پیچیده آزمایشگاهی به نام «توالی‌یابی ژنومی»[۲] به‌دست می‌آیند، نمایش‌دهندهٔ «ژنوم»[۳] موجودات زنده‌اند. ژنوم موجود

[۱] phylogenetic tree [۲] genome sequencing [۳] genome



شکل ۱. مثالی از طبقه‌بندی پرندگان در قالب درخت تبارزایشی

زنده را از لحاظ ترکیبیاتی می‌توان یک رشتهٔ بسیار طولانی از چهار حرف A، T، C و G درنظر گرفت. به زبان علوم کامپیوتر، مجموعهٔ حروفی را که یک رشته از آن تشکیل شده است الفبای رشته می‌نامند و آن‌را با Σ نمایش می‌دهند. در این مقاله، فرض می‌کنیم رشته‌های مورد مطالعه از الفبای $\Sigma = \{A, T, C, G\}$ ساخته شده باشند. تحولات اخیر در زیست‌شناسی موجب شده است که داده‌های ژنومِ موجودات زندهٔ بسیار زیادی، به‌راحتی، در دسترس عموم دانشمندان قرار گیرد. زیست‌شناسان امروزی تلاش می‌کنند با داشتن این رشته‌های بسیار طولانی، درخت‌های طبقه‌بندی بسازند که در هرکدام ژنوم‌های شبیه به هم در درخت‌های نزدیک‌تر به هم قرار گیرند.

مسئلهٔ طبقه‌بندی موجودات زنده را می‌توان از دیدگاه «تکامل»[۱] نیز بررسی کرد. در این دیدگاه، درخت طبقه‌بندی موجودات زنده به‌گونه‌ای روابط تکاملی و نِسَب‌های مشترک در بین موجودات مورد مطالعه را نشان می‌دهد. بدین دلیل است که مسئلهٔ طبقه‌بندی را می‌توان «مسئلهٔ بازسازی درخت تبارزایشی» نامید. تعریف دقیق مسئله در ادامه خواهد آمد.

۲. صورت‌بندی ریاضی مسئلهٔ طبقه‌بندی

با توجه به نگاه جدید زیست‌شناسان به موجودات زنده، مسئلهٔ طبقه‌بندی آن‌ها را می‌توان به‌صورتِ یک مسئلهٔ ترکیبیاتی یا الگوریتمی مدل‌سازی کرد. ورودی این مسئله، تعدادی رشته است و هدف مسئله، تولید یک درخت ریشه‌دار است که ورودی‌های مسئله در رأس‌های آن درخت قرار گرفته باشند. به منظور تولید یک صورت‌بندی الگوریتمی، ضروری است درخت طبقه‌بندی با جزئیات

[۱] evolution

بیشتری مشخص شود. ازاین‌رو، دو صورت‌بندی مختلف برای مسئلهٔ طبقه‌بندی ارائه می‌کنیم. به‌جز این دو، می‌توان صورت‌بندی‌های دیگری نیز برای مسئله ارائه کرد، اما این دو صورت‌بندی از این لحاظ برای ما جالب‌اند که در ادامه، الگوریتمی تقریبی برای آن‌ها ارائه خواهیم کرد.

پیش از تعریف مسئله، ضروری است تابعی تعریف کنیم که نشان‌دهندهٔ میزان تفاوت دو رشته باشد. توجه کنید که تفاوت دو رشته، به‌نوعی معکوس شباهت دو رشته است، و با تعریف یکی از آن‌ها می‌توان دیگری را نیز تعریف کرد. برای سادگی کار، در این مقاله فرض می‌کنیم رشته‌های ورودی، همگی طولی برابر با m دارند.

از معیارهایی که با آن می‌توان تفاوت دو رشته را اندازه‌گیری کرد «فاصلهٔ هَمینگ»[1] دو رشته است که به‌صورت

$$\mathbb{D}\langle\langle s \not\equiv t \rangle\rangle = \sum_{i=1}^{m} s[i] \neq t[i]$$

تعریف می‌شود. در اینجا $s[i]$ حرف iام رشته s است و

$$x \neq y = \begin{cases} 1 & x = y \\ 0 & \text{در غیر این صورت} \end{cases}$$

به‌عبارت‌دیگر، فاصلهٔ همینگ بین دو رشته برابر با تعداد مکان‌هایی است که دو رشته در آن مکان‌ها با هم تفاوت دارند.

۱.۲. تعریفی از مسئلهٔ طبقه‌بندی: مسئلهٔ درخت تبارزایشی با صرفه‌جویی بیشینه.

تعریف ۱.۲ (مسئلهٔ درخت تبارزایشی با صرفه‌جویی بیشینه).
ورودی: n رشته g_i (برای $1 \leq i \leq n$) مربوط به ژنوم n موجود زنده. هرکدام از رشته‌ها از حروف $\{A, T, C, G\}$ تشکیل شده‌اند و طول همهٔ رشته‌ها برابر با m است.
خروجی: درخت ریشه‌دار T که رشته‌های ورودی در رأس‌های آن قرار گرفته‌اند، به همراه یک تابع برچسب‌گذاری λ که به رأس‌های درخت، رشته‌هایی به طول m نسبت می‌دهد. رشته‌های منتسب به رأس‌های درخت، در رأس‌های مرتبط با رشته‌های ورودی، باید با رشته‌های ورودی همخوانی داشته باشند.

[1] Hamming distance

هدف: یافتن درخت T و تابع برچسب‌گذاری λ به‌صورتی‌که تابع زیر کمینه شود

$$\sum_{(u,v)\in E(T)} \mathbb{D}\langle\langle \lambda(u) \not\equiv \lambda(v) \rangle\rangle$$

که در آن $E(T)$ نشان‌دهندهٔ مجموعهٔ یال‌های درخت است.

توجه کنید که در تعریف ۱.۲، نه تنها باید یک درختِ طبقه‌بندی ارائه شود، بلکه باید ژنوم مربوط به موجودات میانی نیز محاسبه شود. بدین‌صورت می‌توان به هر درخت طبقه‌بندی، که هم‌اکنون تمامی رأس‌های آن به رشته‌های ژنومی منتسب شده است، یک امتیاز نسبت داد. امتیازی که در تعریف مسئلهٔ بالا در نظر گرفته شده است، مجموع میزان تفاوت‌های مربوط به یال‌های درخت است. منظور از تفاوت مربوط به یک یال درخت، فاصلهٔ همینگ بین ژنوم‌های منتسب‌شده به رأس‌های دو سر یال است. به‌عبارت‌دیگر، در مسئلهٔ بیشینهٔ صرفه‌جویی تلاش می‌کنیم درختی بیابیم که یال‌های آن کمترین تغییرات را داشته باشد. هدفی که در این مسئله تعریف شده است، صورت‌بندی بیشتر هدف قبلی به زبان ریاضی است: موجوداتی که بیشتر به یکدیگر شبیه‌اند در درختِ به هم نزدیک‌تری قرار گیرند. می‌توان تصور کرد که اگر یک درختِ نامناسب انتخاب کنیم، رشته‌های دو طرفِ یال‌ها فاصلهٔ زیادی با یکدیگر دارند و درنتیجه مقدار تابع هدف در آن‌ها بزرگ‌تر است. پس طبیعتاً ما باید به دنبال درختی بگردیم که مقدار تابع هدف را کمینه کند.

امروزه زیست‌شناسان برای طبقه‌بندی موجودات زنده از روش‌های حل مسئلهٔ یافتن درخت تبارزایشی با بیشینهٔ صرفه‌جویی، یعنی مسئلهٔ تعریف‌شده در تعریف ۱.۲، استفاده می‌کنند. جالب است بدانیم که سعی زیست‌شناسان در طبقه‌بندی موجودات، حلِ یک مسئلهٔ کاملاً الگوریتمی است. این تغییر دیدگاه زیست‌شناسی، از نگاهی پراکنده و موردنگر به دیدگاه‌های الگوریتمی، نشأت گرفته از تغییر دیدگاه در زیربناهای زیست‌شناسی است.

۲.۲. تعریفی دیگر برای مسئلهٔ طبقه‌بندی: مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی. به‌کارگیری «مدل‌های احتمالاتی»[۱] روش دیگری است که با آن می‌توان مسئلهٔ طبقه‌بندی را صورت‌بندی کرد. در این روش، نخست یک مدل احتمالاتی برای تولید یک گروه از موجودات زنده درنظر گرفته می‌شود. این مدل احتمالاتی احیاناً دارای پارامترهایی است و در این روش تلاش می‌شود پارامترهای مدل به‌صورتی محاسبه شوند که احتمال تولید داده‌های ورودی از مدل احتمالاتی بیشینه شود. به این روش اصطلاحاً «روش بیشینهٔ درستنمایی»[۲] گفته می‌شود.

[۱] stochastic models [۲] maximum likelihood method

در مسئلهٔ طبقه‌بندی با روش بیشینهٔ درستنمایی، علاوه‌بر یافتن یک درخت و یک تابع برچسب‌گذاری بر روی رأس‌ها، یک میزان احتمال p_e نیز به هر یال e نسبت داده می‌شود. احتمال p_e، احتمال ایجاد تغییر در هر مکان از رشته را از رأس مادر به رأس فرزند نمایش می‌دهد. در این تعریف، فرض می‌شود هر مکان از رشته با احتمال p_e تغییر می‌کند و طبیعتاً با احتمال $1-p_e$ تغییر نمی‌کند. بدین صورت می‌توان به یک درخت ریشه‌دار که رأس‌های آن با رشته‌هایی برچسب‌گذاری شده است یک احتمال نسبت داد. در مسئلهٔ طبقه‌بندی با روش بیشینهٔ درستنمایی به دنبال درخت، برچسب‌گذاری و اعداد p_e هستیم که احتمال مذکور را بیشینه کنند. تعریف دقیق مسئله به‌صورت زیر است.

تعریف ۲.۲ (مسئلهٔ یافتن درخت تبارزایشی با بیشینهٔ درستنمایی (صورت ۱)).
ورودی: n رشتهٔ g_i (برای $1 \le i \le n$) مربوط به ژنوم n موجود زنده. هرکدام از رشته‌ها از حروف $\{A,T,C,G\}$ تشکیل شده‌اند و طول همهٔ رشته‌ها برابر با m است.
خروجی: درخت ریشه‌دار T که رشته‌های ورودی در رأس‌های آن قرار گرفته است، یک تابع λ که به رأس‌های درخت رشته‌هایی به طول m نسبت می‌دهد، و مقدارهای $0 \le p_e \le 1$ (برای $e \in E(T)$). رشته‌های منتسب به رأس‌های درخت، در رأس‌های مرتبط با رشته‌های ورودی، باید با رشته‌های ورودی همخوانی داشته باشند.
هدف: یافتن درخت و تابع برچسب‌گذاری λ و مقادیر p_e به‌صورتی‌که تابع زیر بیشینه شود

$$\prod_{e \in E(T)} p_e^{d_e} (1-p_e)^{n-d_e}.$$

مسئلهٔ بالا، صورت‌بندی کلاسیکی از مسئلهٔ مورد بحث را مطرح می‌کند. هدف این مسئله یافتن سه گونه متغیر است. یک مجموعه از متغیرها، یعنی مقادیر p_e، را می‌توان با داشتن دو مجموعهٔ دیگر به‌صورتی که میزان تابع هدف را بیشینه کند، محاسبه کرد. با داشتن یال‌ها (درخت) و برچسب رأس‌های درخت، احتمالی که به هر یال e نسبت داده می‌شود فقط به میزان p_e و نه به میزان $p_{e'}$ برای یال‌های دیگر درخت وابسته است. مقدار p_e احتمال مربوط به یال e، یعنی مقدار $p_e^{d_e}(1-p_e)^{n-d_e}$، را بیشینه می‌کند. توجه کنید که در اینجا فرض کرده‌ایم که d_e و $n-d_e$ مقادیر ثابتی هستند. با محاسبهٔ مشتق تابع درستنمایی به این نتیجه می‌رسیم که به‌ازای $p_e = d_e/m$ بیشینه می‌شود. پس می‌توان مسئلهٔ بالا را به‌صورتی از نو بیان کرد که متغیرهای مسئله یک درخت و برچسب‌ها رأس‌های گراف باشند و هدف مسئله یافتن متغیرها به‌صورتی باشد که تابع زیر را بیشینه

کند

$$\prod_{e\in E(T)} (d_e/m)^{d_e}(۱ - d_e/m)^{m-d_e}.$$

توجه کنید که اگر تابع هدف را با ریشه mام آن جایگزین کنیم، مسئله فرق چندانی نمی‌کند زیرا همان درخت و برچسب‌گذاری که تابع هدف را بیشینه می‌کند، ریشهٔ mام تابع هدف را نیز بیشینه می‌کند. با جانشانی تابع هدف با ریشهٔ mام آن، تابع هدف به‌صورت زیر تغییر خواهد کرد

$$\prod_{e\in E(T)} (d_e/m)^{d_e/m}(۱ - d_e/m)^{۱-d_e/m}.$$

مانند استدلال قبل، اگر تابع هدفِ جدید را با لگاریتم آن نیز جانشان کنیم، درخت و برچسب‌گذاری بهینه تغییری نخواهند کرد. پس از جانشانی تابع هدف با لگاریتم آن، تابع هدف به‌صورت زیر تغییر می‌کند

$$\sum_{e\in E(T)} (d_e/m)\lg(d_e/m) + (۱ - d_e/m)\lg(۱ - d_e/m)$$

که در آن تابع $\lg(x)$ تابع لگاریتم در مبنای ۲ است. جالب است که اجزاء تابع بالا، تابع آنتروپی برای احتمال d_e/m است. پس از این تغییرات، مسئلهٔ طبقه‌بندی با بیشینهٔ درستنمایی را به‌صورت زیر بازنویسی می‌کنیم:

تعریف ۳.۲ (مسئلهٔ یافتن درخت تبارزایشی با بیشینهٔ درستنمایی (صورت ۲)).
ورودی: n رشتهٔ g_i (برای $۱ \le i \le n$) مربوط به ژنوم n موجود زنده. هرکدام از رشته‌ها از حروف $\{A, T, C, G\}$ تشکیل شده‌اند و طول همهٔ رشته‌ها برابر با m است.
خروجی: درخت ریشه‌دار T که رشته‌های ورودی در برگ‌های آن قرار گرفته است و یک تابع λ که به رأس‌های درخت رشته‌هایی به طول m نسبت می‌دهد. رشته‌های منتسب به برگ‌های درخت باید با رشته‌های ورودی برابر باشند.
هدف: یافتن درخت و تابع برچسب‌گذاری λ به‌صورتی‌که تابع زیر بیشینه شود

$$\sum_{e\in E(T)} H\left(\frac{d_e}{m}\right)$$

که در آن $H(x) = x\lg(x) + (۱ - x)\lg(۱ - x)$ تابع آنتروپی است.

۳. مسئلهٔ درخت اشتاینر

در این مقاله، مسئلهٔ طبقه‌بندی را با استفاده از روش‌هایی برای حل مسئلهٔ «درخت اشتاینر»[۱] حل می‌کنیم. اما نخست مسئلهٔ درخت اشتاینر را تعریف می‌کنیم.

تعریف ۱.۳ (مسئلهٔ درخت اشتاینر).
ورودی: گراف $G = (V, E)$ به همراه وزن $w(x, y)$ روی یال‌های گراف $((x, y) \in E)$ و زیرمجموعهٔ $S \subseteq V$ که اعضای آن رأس‌های پایانی نامیده می‌شوند.
خروجی: زیرگرافی از G.
هدف: یافتن زیرگراف همبندی از G که شامل همهٔ رأس‌های پایان (S) باشد به‌صورتی‌که وزن یال‌های زیرگراف کمینه باشد.

به‌عبارت‌دیگر، در مسئلهٔ درخت اشتاینر به دنبال اتصال رأس‌های پایانی با استفاده از یال‌ها و احیاناً رأس‌های گراف هستیم. مسئلهٔ درخت اشتاینر کاربردهای فراوانی در زمینه‌های مختلف دارد. برای مثال، فرض کنید می‌خواهیم اینترنتی پرسرعت در کشور ایجاد کنیم که برخی از شهرها را حتماً شامل شود و هزینهٔ اتصالات کمینه باشد. این مسئله به‌سادگی قابل صورت‌بندی به‌صورت یک مسئلهٔ درخت اشتاینر است.

معمولاً برای مسئلهٔ درخت اشتاینر فرض می‌شود که نابرابری مثلثی برای وزن یال‌ها برقرار است. به‌عبارت‌دیگر، به‌ازای هر سه راس $v, u, x \in V$ فرض می‌کنیم $w(v, u) \leq w(v, x) + w(x, u)$. اگر در گراف مفروض این شرط برقرار نبود، می‌توان هر یال را با کوتاه‌ترین مسیر بین دو رأس انتهایی یال جایگزین کرد. بدین‌صورت یال جدید نمایندهٔ همان کوتاه‌ترین مسیر در گراف اصلی خواهد بود و می‌توان بدون کاسته‌شدن از کلیت مسئله، مسئله را برای گراف جدید حل کرد که در آن نابرابری مثلثی برقرار است.

۱.۳. کران‌های پایین و بالا در حل مسئلهٔ درخت اشتاینر. مسئلهٔ درخت اشتاینر یک مسئلهٔ قدیمی در زمینهٔ علوم کامپیوتر است و تلاش‌های فراوانی برای تحلیل و حل آن شده است. با برخی مفروضات مقبول به لحاظ پیچیدگی محاسبات، الگوریتمی برای این مسئله که جواب دقیق را در زمان چندجمله‌ای پیدا کند وجود ندارد [۶]. در این شرایط متخصصان علوم کامپیوتر تلاش می‌کنند الگوریتمِ تقریبی برای مسئله بیابند. تعریف الگوریتم تقریبی (به‌صورت کلی) در ادامه آمده است.

فرض کنید هدفِ یک مسئلهٔ الگوریتمی کمینه کردن تابع f است. الگوریتم $\mathscr{A}$ برای یک مسئله، یک الگوریتم تقریبی با ضریب تقریب α است اگر به‌ازای هر ورودی I داشته باشیم

[۱] Steiner tree

$$f(\mathscr{A}(I)) \leq \alpha \cdot f(\mathrm{OPT})$$

که در آن $\mathscr{A}(I)$ خروجی الگوریتم $\mathscr{A}$ بر روی ورودی I و OPT جواب بهینه برای مسئله است. به‌عبارت‌دیگر، الگوریتم تقریبی الگوریتمی است که هرچند ممکن است جواب آن بهینه نباشد، میزان بهینه بودن جواب، تضمینی نسبی دارد.

از جملهٔ دیگر نتایج سلبی در مورد توانایی حل مسئلهٔ درخت اشتاینر، با درنظر گرفتن برخی مفروضات معقول به لحاظ پیچیدگی محاسبات، آن است که متأسفانه الگوریتمی با ضریب تقریبِ بهتر از $\frac{۹۶}{۹۵} \approx ۱/۰۱۰۵$ برای مسئلهٔ درخت اشتاینر وجود ندارد [۵].

اما الگوریتم‌هایی تقریبی برای درخت اشتاینر ابداع شده است. شاید ساده‌ترین الگوریتم، استفاده از درخت فراگیر کمینه به‌عنوان جوابی تقریبی برای مسئلهٔ درخت اشتاینر باشد. محاسبهٔ درخت فراگیر کمینه در زیرگرافی که فقط شامل رأس‌های پایانی باشد، درختی را تولید می‌کند که هزینهٔ آن حداکثر ۲ برابر هزینهٔ بهینه است. پس این الگوریتم، یک الگوریتم تقریبی با ضریب تقریب ۲ است. الگوریتم‌های تقریبی دیگری نیز برای مسئلهٔ درخت اشتاینر ارائه شده‌اند که یکی پس از دیگری ضریب تقریب کمتری دارند. نخستین الگوریتم از این سری را زلیکوفسکی[۱] ارائه کرد که ضریب تقریب آن $\frac{۱۱}{۶}$ است [۸]. بهترین الگوریتم تقریبی که تاکنون برای مسئلهٔ درخت اشتاینر ارائه شده است دارای ضریب تقریب $\ln(۴) + \epsilon \leq ۱/۳۹$ است که با به کارگیری روش‌های برنامه‌ریزی خطی و فنون گردکردن به دست می‌آید [۴].

در این مقاله، از الگوریتمِ برِمن و همکارانش، که به‌اختصار آن‌را الگوریتم برمن می‌نامیم، استفاده می‌کنیم [۲]. اگرچه در ادامهٔ مقاله نشان خواهیم داد که چگونه می‌توان با استفاده از روش‌های حل مسئلهٔ درخت اشتاینر، مسئلهٔ طبقه‌بندی را حل کرد، متأسفانه در حالت کلی نمی‌توان از الگوریتم‌های تقریبی درخت اشتاینر یک الگوریتم برای مسئلهٔ طبقه‌بندی پیدا کرد. تنها می‌توان از برخی الگوریتم‌های تقریبی، با شرایط خاص، برای این‌کار استفاده کرد. خوشبختانه، الگوریتم برمن این شرط عجیب را دارد و بدین‌جهت ما از الگوریتم برمن برای حل مسئلهٔ طبقه‌بندی استفاده می‌کنیم.

۴. الگوریتم تقریبی برمن برای مسئلهٔ درخت اشتاینر

در این بخش، الگوریتم برمن را برای مسئلهٔ درخت اشتاینر مطرح می‌کنیم. توجه کنید که این الگوریتم به‌صورت کلی برای مسئلهٔ درخت اشتاینر طراحی شده است و این نکته را که چگونه از الگوریتم برمن برای حل مسئله‌های تبارزایشی استفاده می‌شود در بخش‌های بعد بررسی می‌کنیم.

[۱] Alexander Zelikovsky

۱.۴. ایدهٔ اصلی الگوریتم برمن. ایدهٔ اصلی الگوریتم برمن در این است که این الگوریتم همهٔ زیرمجموعه‌های $\tau \subseteq S$ را که اندازهٔ آن‌ها از یک عدد ثابت مانند k (مثلاً، $k = ۴$) کمتر است درنظر می‌گیرد. به‌ازای هر زیرمجموعهٔ τ، زیردرخت اشتاینر بهینه را، که در آن τ مجموعهٔ رأس‌های پایانی است، محاسبه می‌کند. سپس بررسی می‌کند که آیا اضافه‌کردن این زیردرخت هزینه‌های نهایی را کاهش می‌دهد یا خیر؟ اگر با این‌کار هزینه‌ها کاهش پیدا کرد، این درخت را موقتاً می‌پذیرد تا در مراحل بعد به‌طور دقیق‌تر در مورد این زیردرخت تصمیم‌گیری شود.

خروجی الگوریتم برمن از اجتماع تعدادی زیردرخت اشتاینر تشکیل می‌شود که هرکدام شامل حداکثر k رأس پایانی هستند. به یک خروجی با این شرایط، یک درخت اشتاینر k-محصور[۱] گفته می‌شود. زیربنای اصلی این الگوریتم برپایهٔ این مطلب است که با محدودکردن جواب به جواب‌های k-محصور، ضریب تقریب الگوریتم حداکثر با ضریب $\rho_k \leq ۱ + \frac{۱}{\lceil \lg k \rceil}$ کاهش می‌یابد [۳]. سپس الگوریتم برمن با جستجویی که در میان تمامی درخت‌های اشتاینر بهینه برای τها انجام می‌دهد، تلاش می‌کند بهترین درخت اشتاینر k-محصور را بیابد. البته الگوریتم برمن تنها موفق می‌شود برای $k = ۳$ کم‌هزینه‌ترین درخت اشتاینر k-محصور را بیابد و برای kهای بزرگ‌تر ضریب تقریب بهتری به دست نمی‌دهد.

۲.۴. الگوریتم‌های تقریبی پس از الگوریتم برمن. چند سال بعد، رابینز و همکارانش الگوریتمی ارائه کردند که برای مقادیر بزرگ‌تر k، اگرچه نمی‌تواند بهترین درخت اشتاینر k-محصور را بیابد، درخت اشتاینری می‌یابد که هزینه آن به بهترین درخت اشتاینر k-محصور خیلی نزدیک است [۷]. با این روش آن‌ها توانستند الگوریتمی با ضریب تقریب $۱ + \ln(۳)/۲ < ۱٫۵۵$ برای مسئلهٔ درخت اشتاینر پیدا کنند.

۳.۴. چرا زمان اجرای الگوریتم برمن چندجمله‌ای است؟. برای اینکه نشان دهیم زمان اجرای الگوریتم برمن چندجمله‌ای است، نخست باید نشان دهیم تعداد درخت‌های مورد بررسی چندجمله‌ای است. برای یک مقدار ثابت k، تعداد زیرمجموعه‌های k عضوی از S برابر است با $\binom{|S|}{k} \leq |S|^k$ که این مقدار یک چندجمله‌ای برحسب اندازهٔ ورودی است.

در یک اجرای ساده از الگوریتم برمن به‌ازای هرکدام از مجموعه‌های τ، همهٔ توپولوژی‌های درخت‌های مفید با k برگ ساخته می‌شود. درخت‌های مفید درخت‌هایی هستند که در آن‌ها همهٔ رأس‌های غیربرگ حداقل به سه یال متصل باشند. با توجه به تعریف، درخت‌های مفید حداکثر $k - ۲$ رأس غیربرگ دارند. توجه کنید که چون فرض کرده‌ایم نابرابری مثلثی بین وزن یال‌های گراف

[۱] k-restricted

ورودی برقرار است، درخت اشتاینر بهینه (برای τ) یک درخت مفید است. چون k را یک مقدار ثابت فرض کرده‌ایم، تعداد این درخت‌ها نیز مقداری ثابت است.

در ادامه، همهٔ حالت‌های قرارگیری رأس‌های گراف در رأس‌های درخت بررسی می‌شود و حالتی را که در آن τ در برگ‌های درخت قرار دارد و هزینهٔ آن کمینه است به‌عنوان نتیجهٔ این قسمت در نظر می‌گیرند. چون تعداد رأس‌های این درخت‌ها نیز حداکثر $k-2$ است، زمان اجرای این قسمت برابر $|V|^{k-2}$ است که تابعی چندجمله‌ای است.

۴.۴. الگوریتم برمن. الگوریتم برمن شامل سه مرحله است: ارزیابی، انتخاب، و ساخت.

الگوریتم برمن در مراحل ارزیابی و انتخاب، فقط رأس‌های پایانی (S) و یک زیردرخت فراگیر بین این رأس‌ها را در نظر می‌گیرد و به‌طور مستقیم با سایر رأس‌های گراف کاری ندارد. در طی مرحلهٔ ارزیابی، الگوریتم برمن از زیربرنامه‌ای استفاده می‌کند که یک زیرمجموعهٔ $\tau \subseteq S$ را به‌عنوان ورودی می‌گیرد و زیردرخت اشتاینر بهینه در گراف را برای مجموعهٔ رأس‌های پایانی τ برمی‌گرداند. این زیرگراف را با $\text{Smt}(\tau)$ و هزینهٔ آن را با $\text{cost}(\text{Smt}(\tau))$ نشان می‌دهیم.

فرض کنید مجموعهٔ یال‌های M یک درخت بین رأس‌های پایانی می‌سازند. به یال $e \in M$ «پُل» بین رأس‌های v و u (که هر دو رأس پایانی‌اند) گفته می‌شود، اگر e (برحسب وزن) سنگین‌ترین یالی باشد که با حذف آن رأس‌های v و u در دو مولفهٔ همبند قرار گیرند. به‌عبارت‌دیگر، پل بین v و u سنگین‌ترین یال بین تنها مسیر بین v و u در M است. پل بین v و u در یال‌های M را با $\text{Bridge}_M(v,u)$ نشان می‌دهیم. همچنین پل بین زیرمجموعهٔ τ از M را به‌صورت

$$\text{Bridge}_M(\tau) = \{\text{Bridge}_M(v,u) : v,u \in \tau\}$$

تعریف می‌کنیم. به‌ازای هر پل $e \in \text{Bridge}_M(\tau)$ حداقل یک جفت رأس v و u از پایانی‌ها وجود دارند که e سنگین‌ترین پل بین آن دو رأس است. یکی از این جفت رأس‌ها را درنظر می‌گیریم و می‌گوییم گسترهٔ پل e براساس مجموعهٔ τ روی M، مجموعهٔ $\{v,u\}$ است و آن‌را به‌صورت زیر نشان می‌دهیم

$$\text{span}_M(\tau, e) = \{v,u\}.$$

در الگوریتم برمن، اگر بخواهیم درخت کنونی M را براساس درخت اشتاینر بهینه برای τ ($\text{Smt}(\tau)$) ارتقاء دهیم، یال‌های $\text{Bridge}_M(\tau)$ را حذف و $\text{Smt}(\tau)$ را به M اضافه می‌کنیم. تابع $\text{gain}_M(\tau)$ را میزانی از بهبودی تعریف می‌کنیم که $\text{Smt}(\tau)$ برای درخت کنونی M ایجاد

می‌کند. به‌عبارت‌دقیق‌تر، $\text{gain}_M(\tau)$ برابر است با هزینهٔ یال‌های $\text{Bridge}_M(\tau)$ منهای هزینهٔ بهترین درخت اشتاینر با رأس‌های پایانی τ.

با توجه به تعریف‌هایی که تاکنون انجام داده‌ایم، مراحل ارزیابی، انتخاب، و ساخت الگوریتم برمن به‌ترتیب در الگوریتم ۱، الگوریتم ۲، و الگوریتم ۳ نمایش داده شده‌اند. در این الگوریتم‌ها $\binom{S}{s}$ به معنای همهٔ زیرمجموعه‌های s عضوی از مجموعه S است. همچنین توجه کنید که یال‌ها به‌صورت سه‌تایی‌های (v, u, w) نمایش داده شده‌اند که در آن v و u رأس‌های یال و w وزن یال است. همچنین در الگوریتم ممکن است سه‌تایی (v, u, w) به‌صورت دوتایی (e, w) نمایش داده شود، که دراین‌صورت $e = (v, u)$.

الگوریتم ۱ مرحلهٔ ارزیابی

$E \leftarrow \{(u, v, w(u, v)) | u, v \in S\}$
$M \leftarrow$ MST(E)
Stack $\leftarrow \emptyset$
for $s \leftarrow 3, \ldots, k$ **do**
  for $\tau \in \binom{S}{s}$ **do**
    $g \leftarrow \text{gain}_M(\tau)$
    if $g > 0$ **then**
      $B_{new} \leftarrow \{(u, v, \text{cost}(e) - g) | (e, \text{cost}(e)) \in \text{Bridge}_M(\tau) \& (u, v) = \text{span}_M(\tau, e)\}$
      $B_{old} \leftarrow \text{Bridge}_M(\tau)$
      $M \leftarrow M - B_{old} \cup B_{new}$
      Stack.push(τ, B_{old}, B_{new})
    end if
  end for
end for

زمان اجرای الگوریتم برمن با پیاده‌سازی‌های مناسب بر روی یک گراف کلی از دو قسمت تشکیل می‌شود. قسمت اول شامل تولید زیردرخت‌های بهینه با حداکثر k تعداد برگ است که با یک پیاده‌سازی خوب می‌توان آن را در زمان $\mathcal{O}(|V|^۳ + |V|^{k-۲}|S|^{k-۱})$ انجام داد. قسمت دومِ زمان اجرای الگوریتم شامل محاسبات برای نگه‌داری درخت بهینه و به‌روزرسانی آن است که با پیاده‌سازی مناسبی می‌توان این عملیات را در زمان $\mathcal{O}(|S|^{k+۱/۲})$ اجرا کرد. با استفاده از این پیاده‌سازی‌ها، در کل، زمان اجرای الگوریتم برابر با $\mathcal{O}(|V|^۳ + |V|^{k-۲}|S|^{k-۱} + |S|^{k+۱/۲})$ خواهد بود.

در مورد ضریب تقریب الگوریتم برمن می‌دانیم که برای $k = ۳$ ضریب تقریب الگوریتم برابر با $\frac{۱۱}{۶}$ است. برای مقادیر بزرگ‌تر از k نمی‌دانیم که آیا می‌توان ضریب تقریب بهتری برای الگوریتم ارائه

الگوریتم ۲ مرحلهٔ انتخاب

$E \leftarrow M$
List $\leftarrow \emptyset$
while Stack is not empty **do**
  $\tau, B_{old}, B_{new} \leftarrow$ Stack.pop( )
  $E \leftarrow E \cup B_{old}$
  if $B_{new} \subseteq M$ **then**
    List.enqueue(τ, B_{new})
  else
    $E \leftarrow E - B_{new}$
    $M \leftarrow \text{MST}(M)$
  end if
end while

الگوریتم ۳ مرحلهٔ ساخت

while List is not empty **do**
  $\tau, B_{new} \leftarrow$ List.dequeue( )
  $M \leftarrow M - B_{new} \cup \text{Smt}(\tau)$
end while

کرد و یا خیر. در آزمایش‌هایی که بر روی گراف‌ها انجام شده است، می‌توان دید که با افزایش k عملاً نتایج بهتری حاصل می‌شود.

۵.۴. خلاصه‌ای از الگوریتم برمن. خلاصه‌ای از الگوریتم برمن را که برای فهم الگوریتم‌های این مقاله ضروری است می‌آوریم:

(۱) الگوریتم برمن در زمان چندجمله‌ای مسئلهٔ درخت اشتاینر را حل می‌کند؛
(۲) ضریب تقریب الگوریتم برمن برابر با $\frac{۱۱}{۶}$ است؛
(۳) الگوریتم برمن همیشه یک درخت بین رأس‌های پایانی نگه‌داری می‌کند و برای دسترسی به بقیهٔ گراف فقط تابع $\text{Smt}(\tau)$ را فراخوانی می‌کند؛
(۴) تابع $\text{Smt}(\tau)$ یک مجموعه از رأس‌های پایانی، یعنی مجموعهٔ τ که حداکثر k عضو دارد را می‌گیرد و کم‌هزینه‌ترین درخت اشتاینر در کل گراف را برای این رأس‌ها برمی‌گرداند.

۵. الگوریتم تقریبی برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه

در این بخش الگوریتمی تقریبی را برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه که در تعریف ۱.۲ معرفی شد، بیان می‌کنیم. برای این کار، ابتدا نشان می‌دهیم که چگونه مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه به مسئلهٔ درخت اشتاینر تبدیل می‌شود و خواهیم دید که این تبدیل عملاً برای استفاده مناسب نیست. پس‌ازآن با استفاده از خواص الگوریتم برمن، الگوریتم موردنظر را عرضه می‌کنیم.

۱.۵. تبدیل مسئلهٔ تبارزایشی به مسئلهٔ درخت اشتاینر. در مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه، هدف یافتن یک درخت با رشته‌های منتسب به تمامی رأس‌هاست. فرض کنید ورودی یک مسئلهٔ تبارزایشی، یعنی n رشته به طول m (g_iها) داده شده است و می‌خواهیم از روی آن یک ورودی برای مسئلهٔ درخت اشتاینر بسازیم. به‌این‌ترتیب، اگر بتوانیم مسئلهٔ درخت اشتاینر را حل کنیم، مسئلهٔ تبارزایشی نیز حل شده است. یک گراف فرضی $G_U(V_U, E_U)$ درنظر می‌گیریم که مجموعهٔ رأس‌های آن V_U، همهٔ رشته‌هایی به طول m است. بین هر دو رأس u و v یالی با وزن $\mathbb{D}\langle\langle v \not\equiv u \rangle\rangle$ در گراف G_U قرار می‌دهیم. در این گراف، برخی از رأس‌ها همان رأس‌های ورودی مسئلهٔ تبارزایشی هستند، این رأس‌ها را به‌عنوان رأس‌های پایانی (مجموعهٔ S) علامت‌گذاری می‌کنیم. بدین‌صورت به‌ازای یک ورودی برای مسئلهٔ تبارزایشی، یک ورودی برای مسئلهٔ اشتاینر به‌دست می‌آید. طبق تعریف، هر درخت اشتاینر در گراف G_U برای مجموعهٔ رأس‌های پایانی S، یک درخت تبارزایشی است. هزینهٔ یک درخت اشتاینر نیز طبق تعریف با هزینهٔ درخت تبارزایشی برابر است. در نتیجه با این تبدیل نشان دادیم اگر بتوانیم مسئلهٔ درخت اشتاینر را در گراف G_U با مجموعهٔ S حل کنیم، درخت حاصل را می‌توانیم به‌عنوان پاسخ مسئلهٔ درخت تبارزایشی ارائه کنیم. توجه کنید که درخت حاصل از مسئلهٔ درخت اشتاینر ریشه‌دار نیست. برای ریشه‌دار کردن درخت، هر رأس را می‌توان یک ریشه درنظر گرفت و درخت حاصل، درخت تبارزایشی است.

با توجه به تبدیل بالا، تنها کاری که لازم است انجام دهیم این است که گراف G_U را بسازیم و درخت اشتاینر را در آن برای مجموعه S محاسبه کنیم. متأسفانه، کار به این سادگی نیست! علّت هم این است که گراف G_U بسیار بزرگ است و حتی یک بار ساختنِ گراف G_U به زمانی نمایی برحسب اندازهٔ ورودی نیاز خواهد داشت. پس چه باید کرد؟

۲.۵. پیاده‌سازی تابع $\mathbf{Smt}(\tau)$ برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه. برای غلبه بر بزرگی گراف G_U در تبدیل بالا، از روشی استفاده می‌کنیم که در آن لازم نباشد گراف G_U به‌طور صریح ساخته شود. برای این کار یک الگوریتم برای مسئلهٔ درخت اشتاینر انتخاب می‌کنیم و الگوریتم را روی گراف فرضی اجرا می‌کنیم، هر زمان که الگوریتم واقعاً به داده‌های گراف نیاز داشت، آن داده‌ها

را تولید می‌کنیم. به‌این‌ترتیب، باید الگوریتمی انتخاب کنیم که با مجموعهٔ کوچکی از رأس‌های گراف سروکار داشته باشد. انتخاب ما الگوریتم برمن است.

ویژگی الگوریتم برمن این است که به‌جز اطلاعات مجموعهٔ رأس‌های پایانی، تنها با استفاده از تابع $\mathrm{Smt}(\tau)$ به سایر رأس‌های گراف دسترسی دارد. تابع $\mathrm{Smt}(\tau)$ تابعی است که درخت اشتاینر بهینه را برای رأس‌های پایانی τ برمی‌گرداند. پس اگر بتوانیم این تابع را در گراف G_U به‌صورتی پیاده‌سازی کنیم که نیازی به ساختن G_U به‌طور صریح نباشد، آنگاه می‌توانیم از الگوریتم برمن برای مسئلهٔ تبارزایشی استفاده کنیم.

برای پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ باید بتوانیم بهترین درخت اشتاینر را برای مجموعهٔ τ بیابیم. چون مجموعهٔ τ در مسئلهٔ ما تعدادی رشتهٔ ژنومی است و رأس‌های دیگر گراف G_U نیز رشته‌های ژنومی هستند، باید یک درخت بیابیم که هر رأس آن نمایش‌دهندهٔ یک رشته به طول m، شامل رأس‌های τ و کم‌هزینه‌ترین باشد. این مسئله دقیقاً همان مسئلهٔ اولیه تبارزایشی است، با این تفاوت که در آن تعداد رشته‌های ورودی یک عدد ثابت کوچک مثلاً ۳ یا ۴ است.

اگر تعداد برگ‌های درخت را یک عدد ثابت محدود، مثلاً k، بگیریم تعداد درخت‌های مفید با k برگ، تعدادی ثابت است. برای پیاده‌سازی سریع تابع $\mathrm{Smt}(\tau)$ می‌توانیم به‌ازای هر ساختار درخت، دنبال کم‌هزینه‌ترین برچسب‌گذاری برای آن ساختار درخت بگردیم و در نهایت کم‌هزینه‌ترین برچسب‌گذاری را به‌عنوان جواب گزارش کنیم. خوشبختانه، اگر ساختار یک درخت را داشته باشیم، و بدانیم که چه رشته‌هایی بر روی برگ‌های درخت قرار می‌گیرد، کم‌هزینه‌ترین برچسب‌گذاری برای درخت با یک الگوریتم برنامه‌ریزی پویا[۱] قابل محاسبه است.

الگوریتم برنامه‌ریزی پویا برای محاسبهٔ کم‌هزینه‌ترین برچسب‌گذاری درخت، آرایه $A[v,i,c]$ را پُر می‌کند که نمایش‌دهندهٔ کمترین هزینه برای برچسب‌گذاری حرف iام در تمام رأس‌های نوادگان رأس v است اگر حرف iام رأس v برابر با حرف c باشد. این آرایه را می‌توان از روی رابطهٔ بازگشتی زیر برای رأس‌های غیربرگ v محاسبه کرد. در رابطه زیر فرض کرده‌ایم که رأس v در درخت دارای فرزندان u_1 تا u_t است

$$A[v,i,c] = \min_{c' \in \Sigma^t} \sum_j A[u_j, i, c'[j]] + c'[j] \neq c \,. \qquad (۱.۵)$$

[۱] dynamic programming

برای برگ v، چون رشته‌های مربوط به برگ‌ها به‌عنوان ورودی داده شده‌اند، تعریف می‌کنیم

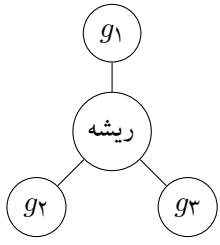
$$A[v,i,c]=\begin{cases}\circ & c=\tau[i]\\ \infty & \text{درغیراین‌صورت}\end{cases} \qquad (۲.۵)$$

۳.۵. جمع‌بندی الگوریتم تقریبی برای درخت تبارزایشی با صرفه‌جویی بیشینه. با کنار هم گذاشتن مطالب بالا، الگوریتمی خواهیم داشت که رشته‌های ژنومی را دریافت می‌کند. سپس الگوریتم برمن را بر روی گراف فرضی G_U اجرا می‌کند. هرکجا که الگوریتم برمن تابع $\mathrm{Smt}(\tau)$ را فراخوانی کرد، الگوریتم ما تمامی ساختارهای درختی با رشته‌های τ برروی برگ‌ها را تولید می‌کند و برای هرکدام کم‌هزینه‌ترین برچسب‌گذاری را با برنامه‌ریزی پویا محاسبه می‌کند. سپس بهترین برچسب‌گذاری را به‌عنوان نتیجهٔ تابع $\mathrm{Smt}(\tau)$ برمی‌گرداند. پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه در الگوریتم ۴ نمایش داده شده است.

چون در روش تبدیل مسئلهٔ درخت تبارزایشی با صرفه‌جویی بیشینه به مسئلهٔ درخت اشتاینر، مسئلهٔ اول به مسئلهٔ دوم دقیقاً تبدیل می‌شد، و برای $k=۳$ الگوریتم برمن یک الگوریتم تقریبی با ضریب تقریب $\frac{۱۱}{۶}$ است، پس الگوریتم ارائه‌شده، یک الگوریتم تقریبی برای مسئلهٔ درخت تبارزایشی با صرفه‌جویی بیشینه با ضریب تقریب $\frac{۱۱}{۶}$ است. در مورد زمان اجرای الگوریتم نیز، چون فراخوانی هر تابع $\mathrm{Smt}(\tau)$ زمان اجرای چندجمله‌ای دارد و کل زمان اجرای الگوریتم برمن چندجمله‌ای است، پس زمان اجرای الگوریتم ارائه‌شده نیز برحسب اندازه ورودی چندجمله‌ای است.

الگوریتم ۴ پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ در الگوریتم برمن برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه

```
mincost ← ∞
for every tree topology t with k leafs do
    Calculate array A for tree t        ▷ آرایه A را براساس معادله‌های (1.5) و (2.5) پر می‌کند.
    cost ← Σ_{i=1}^{m} min_{c∈Σ} A[root, i, c]        ▷ کمینه هزینه درخت t
    if cost < mincost then
        mincost ← cost
        minTree ← t                     ▷ درخت t با برچسب‌هایی که براساس A پرشده است.
    end if
end for
return minTree
```



شکل ۲. درخت ستاره. رشته‌های $g_۱$، $g_۲$ و $g_۳$ به‌عنوان ورودی مسئلهٔ مفروض هستند و هدفِ مسئله، یافتن رشته‌ای برای ریشه است که تابع هدف را کمینه کند.

۶. الگوریتم تقریبی برای مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی

در این بخش به حل مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی می‌پردازیم که در تعریف ۳.۲ معرفی شد. برای حل این مسئله، مانند مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه، مسئله را با همان روش به مسئلهٔ درخت اشتاینر تبدیل می‌کنیم. سپس با همان الگو تلاش می‌کنیم الگوریتم برمن را به‌گونه‌ای به‌کار گیریم که نیازی به ساختن گراف G_U نباشد، یعنی تابع $\mathrm{Smt}(\tau)$ را پیاده‌سازی می‌کنیم.

برای پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ ساختار یک درخت و رشته‌های روی برگ‌های آن را ورودی می‌گیریم و رشته‌هایی را به‌عنوان برچسب‌های رأس‌های داخلی پیشنهاد می‌دهیم که تابع هزینه را کمینه کند. این مسئله را برای حالت $k = ۳$ در الگوریتم برمن حل می‌کنیم. در حالت $k = ۳$، تنها یک ساختار (مفید) درختی می‌توان ارائه داد که آن درخت یک رأس میانی و سه برگ دارد و برگ‌ها همان رشته‌های ورودی تابع $\mathrm{Smt}()$ هستند. به این درخت *ستاره* می‌گویند و در شکل ۲ نمایش داده شده است.

برای حل مسئله، برخلاف تلاشی که برای صورت‌بندی مسئله انجام شد، متغیرهای احتمال مربوط به یال‌ها، یعنی p_jها، را دوباره به مسئله باز می‌گردانیم. همان‌طورکه قبلاً در تعریف ۳.۲ مشاهده کردیم، در جواب بهینه $p_e = d_e/m$ است و چون d_e مقادیری صحیح دارد، حتی اگر مقدار دقیق d_eها را هم ندانیم، می‌توان اطمینان داشت که $p_j \in \{۰, \frac{۱}{m}, \frac{۲}{m}, \ldots, \frac{m-۱}{m}, ۱\}$. با توجه به این نکته، می‌توانیم همهٔ حالت‌ها برای سه مقدار احتمال $p_۱$، $p_۲$ و $p_۳$ را در زمان $\mathcal{O}(m^۳)$ مرور کنیم و برای هرکدام بهترین رشته برای برچسب‌گذاری ریشه را بیابیم. بدین‌صورت، هنگامی که ساختار درخت و مقدار احتمال مربوط به یال‌ها داده شده باشد، مسئله به یافتن رشته مربوط به ریشهٔ درخت

تبدیل می‌شود که تابع هدف آن چنین است

$$\sum_{e\in E(T)} \frac{d_e}{m} \lg\left(\frac{d_e}{m}\right) + \left(۱ - \frac{d_e}{m}\right) \lg\left(۱ - \frac{d_e}{m}\right).$$

اگر رشتهٔ مربوط به ریشهٔ درخت را w نام‌گذاری و فرض کنیم مقادیر احتمال مربوط به یال‌ها، یعنی $p_۱$، $p_۲$ و $p_۳$ داده شده باشند، تابع هدف به‌صورت زیر خواهد بود

$$\sum_{j=۱}^{۳} \frac{\mathbb{D}\langle\langle g_j \not\equiv w\rangle\rangle}{m} \lg(p_j) + \left(۱ - \frac{\mathbb{D}\langle\langle g_j \not\equiv w\rangle\rangle}{m}\right) \lg(۱ - p_j)$$

$$= \sum_{j=۱}^{۳} \frac{\mathbb{D}\langle\langle g_j \not\equiv w\rangle\rangle}{m} (\lg(p_j) - \lg(۱ - p_j)) + \lg(۱ - p_j).$$

مقادیر p_j از قبل معلوم هستند (با در نظر گرفتن همهٔ حالت‌های آن‌ها)، صورت‌بندی ریاضی مسئله به این صورت خواهد شد

$$\operatorname*{argmin}_w \sum_{j=۱}^{۳} \frac{\mathbb{D}\langle\langle g_j \not\equiv w\rangle\rangle}{m} (\lg(p_j) - \lg(۱ - p_j)) + \lg(۱ - p_j).$$

چون به‌دنبال یافتن رشتهٔ w هستیم و در تابع هدف، $\lg(۱ - p_j)$ مقادیری ثابت هستند، می‌توانیم آن‌هارا از تابع هدف حذف کنیم. ضرب تابع هدف در ثابت m نیز مسئله را تغییر نمی‌دهد. این تغییرها را در مسئله اعمال می‌کنیم و تابع هدف مسئله به‌صورت زیر تغییر خواهد کرد

$$\operatorname*{argmin}_w \sum_{j=۱}^{۳} C_j \cdot \mathbb{D}\langle\langle g_j \not\equiv w\rangle\rangle$$

که در آن $C_j := \lg(p_j) - \lg(۱ - p_j)$. اکنون تابع $\mathbb{D}\langle\langle \cdot \not\equiv \cdot\rangle\rangle$ را بسط می‌دهیم

$$\operatorname*{argmin}_w \sum_{j=۱}^{۳} C_j \sum_{i=۱}^{m} g_j[i] \neq w[i] = \operatorname*{argmin}_w \sum_{i=۱}^{m} \sum_{j=۱}^{۳} C_j \; g_j[i] \neq w[i].$$

در عبارت بالا، تابع هدف برای هر مکان i به‌صورت مستقل محاسبه می‌شود و در نهایت مقدار تابع هدف برابر است با جمع این مقدارها برای همه مکان‌ها. پس برای یافتن پاسخ بهینهٔ مسئله کافی است برای هر مکان i به‌صورت مستقل جواب بهینه برای $w[i]$ را محاسبه کنیم. ازطرفی، برای هر

حرف $c \in \Sigma$ می‌توانیم عبارت $\sum_{j=1}^{3} C_j\ g_j[i] \neq c$ را برای مکان i محاسبه کنیم. بدین‌صورت با بررسی همهٔ این حروف، حرفی که این عبارت را کمینه می‌کند به‌دست می‌آوریم.

جمع‌بندی پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ برای مسئلهٔ تبارزایشی با درست‌نمایی بیشینه در الگوریتم ۵ نمایش داده شده است.

الگوریتم ۵ پیاده‌سازی تابع $\mathrm{Smt}(\tau)$ در الگوریتم برمن برای مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی

```
mincost ← ∞
for p_1 ∈ {0, 1/m, ..., (m−1)/m, 1} do
    for p_2 ∈ {0, 1/m, ..., (m−1)/m, 1} do
        for p_3 ∈ {0, 1/m, ..., (m−1)/m, 1} do
            C_j ← lg(p_j) − lg(1 − p_j), برای j = 1, ..., 3
            cost ← 0
            for i ← 1, ..., m do
                w_i ← argmin_c{C_1 c ≠ g_1 + C_2 c ≠ g_2 + C_3 c ≠ g_3}
                cost ← cost + cost(w_i)
            end for
            if cost < mincost then
                mincost ← cost
                minW ← w
            end if
        end for
    end for
end for
return درختی با رشته minW به‌عنوان برچسب ریشه
```

۷. جمع‌بندی

در این مقاله، نخست مسئلهٔ تبارزایشی از دیدگاه زیست‌شناسی را مرور و سپس صورت‌بندی ریاضی مسئله را بیان کردیم. در این صورت‌بندی مشاهده کردیم که چگونه یک مسئلهٔ قدیمی در زیست‌شناسی، با تحولاتی که اخیراً در علم زیست‌شناسی رخ داده است، به یک مسئلهٔ ترکیبیاتی- الگوریتمی تبدیل شد.

صورت‌بندی ریاضی مسئله را در قالب دو مسئله با اندکی تفاوت مطرح کردیم. در مسئلهٔ اول که مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه نام دارد (تعریف ۱.۲)، تلاش می‌کنیم درختی پیدا کنیم که تغییرات روی یال‌های آن کمترین میزان باشد. به‌عبارت‌دیگر، به دنبال درختی می‌گردیم

که بیشترین صرفه‌جویی را در تغییرات داشته باشد. مسئلهٔ دوم، یعنی مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی (تعریف ۲.۲)، براساس یک مدل تعریف شده است. مدل در مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی شامل یک درخت و احتمال‌های منتسب به یال‌های درخت است. هدفِ مسئله، تولید مدل با بیشترین احتمال است. سپس الگوریتم‌های تقریبی را معرفی کردیم. الگوریتم تقریبی الگوریتمی است که اگرچه ممکن است جواب بهینه را پیدا نکند، می‌توان از اینکه جوابی که پیدا می‌کند از جواب بهینه خیلی فاصله ندارد اطمینان داشت. هدفمان در این مقاله یافتن الگوریتم‌های تفریبی برای دو مسئلهٔ تعریف‌شده بود.

بعدازآن، مسئلهٔ درخت اشتاینر را که یک مسئلهٔ کلاسیک در علوم کامپیوتر است، تعریف کردیم. در مسئلهٔ درخت اشتاینر به‌دنبال کم‌وزن‌ترین درخت در یک گراف می‌گردیم که رأس‌های مورد نظر ما (رأس‌های پایانی) را به یکدیگر متصل کند. سپس الگوریتم برمن برای درخت اشتاینر را توضیح دادیم که یک الگوریتم تقریبی با ضریب تقریب $\frac{11}{6}$ است. خاصیت جالبی که الگوریتم برمن دارد این است که به‌جز رأس‌های پایانی، به دیگر رأس‌های گراف به‌طور مستقیم دسترسی پیدا نمی‌کند و به‌واسطهٔ یک تابع $\mathrm{Smt}(\tau)$ با دیگر رأس‌های گراف سروکار دارد. این تابع با گرفتن تعداد اندکی رأس، درخت اشتاینر برای آن رأس‌ها را در گراف برمی‌گرداند. این ویژگی خوب الگوریتم برمن است که به ما کمک می‌کند تا بتوانیم الگوریتم‌هایی برای مسئله‌های تبارزایشی پیدا کنیم.

برای ارائهٔ الگوریتم تقریبی برای مسئله‌های تبارزایشی، نشان دادیم که می‌توان مسئلهٔ تبارزایشی را به‌صورت یک مسئلهٔ درخت اشتاینر در گرافی بسیار بزرگ صورت‌بندی کرد. اما نمی‌توانستیم الگوریتم‌های درخت اشتاینر را به‌صورت مستقیم روی این گراف بزرگ اعمال کنیم. پس الگوریتمی را معرفی کردیم که درحقیقت همان الگوریتم برمن بود با این تفاوت که هرگاه الگوریتم برمن می‌خواست تابع $\mathrm{Smt}(\tau)$ را فراخوانی کند، و به این واسطه به دیگر رأس‌های فراوان گراف دسترسی پیدا کند، پاسخ بهینه را به روش دیگری محاسبه می‌کرد. برای مسئلهٔ تبارزایشی با صرفه‌جویی بیشینه این عملیات را با یک برنامه‌ریزی پویا انجام می‌دادیم.[1] در مسئلهٔ تبارزایشی با بیشینهٔ درستنمایی با بررسی همهٔ حالات، که تعداد آن‌ها از یک چندجمله‌ای برحسب اندازه ورودی بیشتر نیست، درخت بهینه را محاسبه می‌کردیم. در مجموع، الگوریتم‌های تقریبی برای هر دو مسئله با ضریب تقریب $\frac{11}{6}$ معرفی شد.

سرانجام اشاره به این نکته خالی از لطف نیست که زیست‌شناسان در حال حاضر نیز از ابزارهای تولید درخت تبارزایشی بسیار استفاده می‌کنند، اما الگوریتم‌های معروفی که بیشتر از آن‌ها برای

[1] در مسابقات برنامه‌سازی ACM غرب آسیا دقیقا همین سوال به‌عنوان یکی از مسئله‌های مسابقه مطرح شده است.

حل مسئلهٔ تبارزایشی استفاده می‌شود، الگوریتم «اتصال همسایه»[۱] و «روش جفت‌گروه بدون وزن با میانگین حسابی»[۲] هستند. هردوی این الگورتیم‌ها، اکتشافی هستد به این معنی که در مورد میزان بهینگی جواب آن‌ها هیچ اطلاعی نداریم، اما، متخصصان علوم کامپیوتر الگوریتم‌های خوبی در این باره ابداع کرده‌اند. به‌نظر می‌رسد پیشرفت‌های علوم کامپیوتر اگرچه برای مسئله‌های روزمرهٔ زیست‌شناسی می‌تواند خیلی مفید باشد، عملاً این پیشرفت‌ها در زمینه‌های مربوط در زیست‌شناسی مورد استفاده وسیع قرار نگرفته‌اند. چه بسا با فعالیت بیشتر متخصصان علوم کامپیوتر در زمینه‌های بیوانفورماتیک در آیندهٔ نزدیک استفاده از الگوریتم‌های مناسب برای مسئله‌های بیوانفورماتیک بیشتر شود.

الگوریتمی که در این مقاله بررسی شد، به‌صورت نظری در مقالهٔ [۱] ارائه شده است. نویسندهٔ این مقاله و همکارانش این الگوریتم را پیاده‌سازی کرده‌اند و برای داده‌های تک‌سلولی مورد آزمایش قرار داده‌اند. نتایج مقایسهٔ این الگوریتم با الگوریتم‌هایی که پیش از این برای تولید درخت تبارزایشی در داده‌های تک‌سلولی ارائه شده بود نشان داد که الگوریتم مطرح‌شده در این مقاله کارآیی بهتری دارد.

مراجع

[1] Alon, N., Chor, B., Pardi, F., Rapoport, A., Approximate maximum parsimony and ancestral maximum likelihood, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **7** (2008), 183–187.

[2] Berman, P., Ramaiyer, V., Improved approximations for the Steiner tree problem, *Journal of Algorithms*, **17** (1994), 381–408.

[3] A. Borchers, Du, D.-Z., Thek-Steiner ratio in graphs, *SIAM J. Comput.*, **26** (1997), 857–869.

[4] Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L., An improved LP-based approximation for steiner tree, in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing (Massachusetts, 2010)*, Association for Computing Machinery, New York, 2010, 583–592.

[5] Chlebík, M., Chlebíková, J., The Steiner tree problem on graphs: Inapproximability results, *Theoretical Computer Science*, **406** (2008), 207–214.

[6] Karp, R. M., Reducibility among combinatorial problems, in *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, J. D. Bohlinger, eds., Springer, New York, 1972, 85–103.

[۱] neighbor joining [۲] unweighted pair group method with arithmetic mean (UPGMA)

[7] Robins, G., Zelikovsky, A., Tighter bounds for graph Steiner tree approximation, *SIAM J. Discrete Math.*, **19** (2005), 122–134.

[8] Zelikovsky, A., An 11/6-approximation algorithm for the Steiner problem on graphs, in *Annals of Discrete Mathematics*, volume 51, Elsevier, 1992, 351–354.

تاریخ ارسال: ۱۳۹۸/۵/۱۴؛ تاریخ بازنگری: ۱۳۹۹/۴/۳۱؛ تاریخ پذیرش: ۱۳۹۹/۵/۱

محمدهادی فروغمنداعرابی: دانشگاه صنعتی شریف، دانشکدهٔ علوم ریاضی

تارنما: http://foroughmand.ir/

رایانامه: foroughmand@sharif.ir