۱ بیان مسئله (شامل تشریح ابعاد مسئله، معرفی دقیق آن، بیان جنبه‌های مجهول و مبهم و متغیرهای مربوط)

هندسه محاسباتی، یک شاخه از علوم کامپیوتر و ریاضیات است که به حل الگوریتمی مسائل هندسی می‌پردازد. هندسه محاسباتی در بسیاری از حوزه‌ها مانند گرافیک رایانه‌ای، رباتیک، سیستم‌های اطلاعات جغرافیایی، شبکه‌های حمل و نقل کاربرد دارد. هندسه محاسباتی کمک می‌کند تا راه‌حل‌های کارا برای مسائل دنیای واقعی که دارای مدلسازی هندسی هستند ارائه شود. یکی از موضوعات مهم در حوزه هندسه محاسباتی ساخت t-پوشاننده[۱] است. فرض کنید که میخواهیم چند شهر را با ساخت تعدادی جاده، به هم دیگر متصل کنیم به طوری که وقتی که شخصی بخواهد روی جاده‌هایی که ساخته‌ایم، بین دو شهر رانندگی کند، طول مسیری را که طی می‌کند نسبت به طول مسیر مستقیم (جاده‌ی کاملاً مستقیم بین آن دو شهر) تفاوت زیادی نداشته باشد. در مدل گرافی این مسئله، منظور این است که مجموع طول یال‌هایی که دو رأس (دو شهر) را به هم متصل می‌کنند، دارای یک کران بالایی که تابعی از فاصله‌ی دو نقطه است، باشد. این نوع از مسائل را می‌توان با ساخت یک t-پوشاننده بین آن شهرها حل کرد. البته می‌توان این موضوع را در دیگر شبکه‌های ارتباطی مانند شبکه‌های ریلی، شبکه‌های آب و فاضلاب و شبکه‌های کامپیوتری مطرح کرد. یک t-پوشاننده G برای S یک گراف غیر جهتدار با مجموعه رئوس S است، به طوری که برای هر دو نقطه p و q در S، یک مسیر Q بین p و q در G است به طوری که طولش کمتر یا مساوی t ($t > ۱$) برابر فاصله اقلیدسی بین p و q است. منظور از طول یک مسیر مجموع وزن یال‌های تشکیل دهنده آن مسیر است. به مسیر Q یک t-مسیر بین p و q گویند. ضریب کشش یک گراف داده شده G با مجموعه رئوس S عبارت است از کوچک‌ترین عدد $t > ۱$ به طوری که G یک t-پوشاننده برای S است. طبق تعریف، واضح است که یک t-پوشاننده می‌تواند یک گراف کامل را تقریب بزند. حال، اگر یک t-پوشاننده، تعداد یال‌هایش کم باشد، آنگاه این نوع گراف می‌تواند یک تقریب مناسب برای گراف کامل باشد. برای اندازه‌گیری میزان خوب بودن یک شبکه، معیارهای زیادی مطرح می‌شود از جمله:

۱ - اندازه: تعداد یال‌های یک شبکه.

۲ - وزن: مجموع وزن یال‌ها.

۳ - ضریب کشش: برای دو رأس داده شده، نسبت بین فاصله بین آن دو نقطه در شبکه و فاصله اقلیدسی بین آنها.

۴ - درجه: بیشترین تعداد یال‌های مجاور به یک رأس در شبکه است.

معمولا در طراحی یک شبکه هندسی یک یا چند تا از معیارهای بالا را در نظر می‌گیرند. هنگامی که یک شبکه هندسی، دارای ضریب کشش کوچک است، به آن شبکه، شبکه پوشاننده می‌گویند و به شبکه‌های پوشاننده با اندازه کوچک، شبکه‌های پوشاننده تنک می‌گویند.

حال، درخت پوشای کمینه روی S را در نظر بگیرید. واضح است که این گراف دارای $n - ۱$ یال و وزن آن کمینه است. همچنین می‌توان ثابت کرد که بیشترین درجه نقاط در یک درخت پوشای کمینه روی مجموعه‌ای از نقاط در صفحه، شش است. سؤالی که ممکن است مطرح شود این است که آیا درخت پوشای کمینه روی S می‌تواند تقریب مناسبی برای گراف کامل روی S باشد؟ جواب خیر است. زیرا با وجود اینکه اندازه، وزن و درجه درخت پوشای کمینه بهینه است اما می‌توان مجموعه نقاطی پیدا کرد به طوری که ضریب کشش درخت پوشای کمینه روی آن نقاط، به تعداد نقاط وابسته باشد که این یک عیب است. بنابراین، درخت پوشای کمینه نمی‌تواند تقریب مناسبی برای گراف کامل باشد.

تا به امروز الگوریتم‌های زیادی برای ساخت پوشاننده‌های هندسی با خواص مختلف (اندازه کوچک، درجه محدود و...) ارائه شده است. در سال ۲۰۰۹، کتابی با عنوان Geometric Spanner Networks توسط ناراسیمهان و اسمید [۹] به چاپ رسیده است که تقریبا تمام الگوریتم‌های اصلی ساخت شبکه‌های پوشاننده هندسی را در بر دارد.

شبکه‌های پوشاننده هندسی به عنوان یکی از شاخه‌های مهم هندسه محاسباتی دارای کاربردهای زیادی است. از جمله این که در سال ۱۹۹۸، رائو و اسمیت [۱۱]، یک الگوریتم تقریبی برای حل مسأله فروشنده دوره گرد با کمک شبکه‌های پوشاننده ارائه دادند. در سال ۲۰۰۵ ، راسل و گیباس [۱۲]، با کمک شبکه‌های پوشاننده هندسی به مطالعه پروتئین‌ها پرداختند. در سال ۲۰۰۷ ، ناوارو و همکارانش [۱۰] روشی مبتنی بر شبکه‌های پوشاننده هندسی برای جستجو در فضای متریک ارائه کردند. یکی دیگر از کاربردهای جالب شبکه‌های هندسی، تقریب درخت پوشای کمینه است. فرض کنید G یک t-پوشاننده برای S باشد. به‌راحتی می‌توان ثابت کرد که درخت پوشای کمینه‌ی گراف G یک t-تقریب[۲] برای درخت پوشای کمینه روی S است که به این معنا است وزن درخت پوشای کمینه‌ی G حداکثر t برابر وزن درخت پوشای کمینه روی S است.

یکی از معروف‌ترین الگوریتم‌های ساخت شبکه‌های پوشاننده هندسی الگوریتم مسیر-حریصانه[۳] است. خروجی این الگوریتم گرافی است که t-پوشاننده مسیر-حریصانه نامیده می‌شود. الگوریتم مسیر-حریصانه اولین بار توسط آلزوفر و همکارانش در سال ۱۹۹۳ [۳] ارائه شد. این الگوریتم به این شکل عمل می‌کند که ابتدا تمام زوج نقاط را براساس فاصله بین آنها از کوچک به بزرگ مرتب می‌کند و سپس به ترتیب آن زوج نقاط را پردازش می‌کند. پردازش به این شکل است که الگوریتم بررسی می‌کند که آیا طول کوتاه‌ترین مسیر بین آن زوج نقطه در گراف ساخته شده (تا آن لحظه از اجرای الگوریتم) بزرگ‌تر از t برابر طول آن زوج نقطه است یا خیر. اگر بزرگ بود آنگاه بین آن زوج، یالی اضافه می‌کند و آن یال به مجموعه یال‌های ما اضافه می‌شود. اگر کوچک‌تر یا مساوی بود، الگوریتم کاری انجام نمی‌دهد و زوج نقطه بعدی پردازش می‌شود. آنچه که واضح است این است که ما با کمک الگوریتم دایکسترا می‌توانیم الگوریتم مسیر-حریصانه را پیاده‌سازی کنیم. اما پیاده‌سازی مستقیم آن، باعث می‌شود که الگوریتم دارای پیچیدگی زمانی $O(n^۴ \log n)$ باشد. البته در این پیاده‌سازی می‌توان از یک ماتریس که کوتاه‌ترین فاصله‌ها را نگهداری می‌کند، استفاده کرد و زمان اجرا را به $O(n^۳)$ بهبود داد. در سال ۲۰۱۰ ، بوز و همکارانش [۴] توانستند پیچیدگی زمانی الگوریتم مسیر-حریصانه را به $O(n^۲ \log n)$ بهبود دهند. بر اساس آخرین اطلاعات ما، بهترین زمان جهت ساخت t-پوشاننده مسیر-حریصانه همین زمان $O(n^۲ \log n)$ است. کارهای تحقیقاتی زیادی بر روی این الگوریتم انجام شده است. از جمله اینکه در سال ۱۹۹۴ ، سوارس [۱۳] ثابت کرد که درجه گراف حاصل از الگوریتم مسیر-حریصانه محدود است که این موضوع نشان می‌دهد که تعداد یال‌های t-پوشاننده مسیر-حریصانه خطی است. در سال ۱۹۹۷ داس و ناراسیمهان [۵] ثابت کردند که وزن گراف t-پوشاننده مسیر-حریصانه روی S از مرتبه $O(wt(MST(S)))$ است

---

[۱] t-spanner

[۲] t-approximate

[۳] path-greedy

که $wt(MST(S))$، وزن درخت پوشای کمینه[۴] روی مجموعه نقاط S است. اگر از دیدگاه پیچیدگی حافظه به الگوریتم مسیر-حریصانه نگاه کنیم می‌بینیم که پیچیدگی حافظه الگوریتم از مرتبه $O(n^2)$ است. البته در سال ۲۰۱۳ ، آلوانیز و همکارانش [۲]، پیچیدگی حافظه الگوریتم مسیر-حریصانه را با کمک مفهوم تجزیه زوجی خوش-مجزا[۵]، یکی از مهمترین ساختمان داده‌ها جهت حل مسائل مجاورتی[۶]، به $O(n)$ بهبود دادند.

مسئله کاهش تعداد درخواست‌های کوتاه‌ترین مسیر: یکی از مهمترین دلایل افزایش زمان اجرای الگوریتم مسیر-حریصانه، وجود تعداد بالای درخواست کوتاه‌ترین مسیر[۷] یا به اختصار SPQ در طول اجرای الگوریتم است. در سال ۲۰۲۲، ابو-عفّاش و همکارانش الگوریتمی با عنوان δ-حریصانه[۸] ارائه کردند که خصوصیات نظری و عملی گراف تولید شده توسط آن مشابه t-پوشاننده مسیر-حریصانه است [۱]. گراف تولید شده توسط الگوریتم δ-حریصانه یک t-پوشاننده است که t-پوشاننده δ-حریصانه نامیده می‌شود. ابو-عفّاش و همکارانش نشان دادند که اگر $\delta = t$، آنگاه t-پوشاننده δ-حریصانه همان t-پوشاننده مسیر-حریصانه است. مهمترین مزیتی که الگوریتم آنها دارد این است که در عمل، تعداد SPQهایی که پاسخ داده می‌شود خیلی کمتر از تعداد SPQهایی است که در الگوریتم مسیر-حریصانه پاسخ داده می‌شود. مزیت دیگری که الگوریتم δ-حریصانه دارد این است که زمان اجرای آن $O(n^2 \log n)$ است.

هدف ما در این پروژه، کاهش تعداد SPQهای پاسخ داده شده در طول ساخت t-پوشاننده مسیر-حریصانه است و درنتیجه کاهش زمان اجرای ساخت آن است. بررسی‌ها و نتایج اولیه ما نشان می‌دهد که می‌توان تغییراتی را روی الگوریتم δ-حریصانه اعمال کرد که می‌تواند به طرز چشمگیری تعداد SPQهای پاسخ داده شده در طول ساخت t-پوشاننده مسیر-حریصانه را کاهش دهد و درنتیجه زمان اجرای ساخت را پایین آورد.

۲ پرسش‌ها و فرضیه‌های پژوهش

در بخش قبل گفتیم که در الگوریتم مسیر-حریصانه، وقتی زوج نقطه (p, q) در نظر گرفته می‌شود، ابتدا طول کوتاه‌ترین مسیر بین p و q در گرافی که تاکنون ساخته شده است محاسبه می‌شود و در صورتی که مقدار آن بزرگ‌تر از t برابر فاصله اقلیدسی بین p و q باشد، یال (p, q) به گراف اضافه می‌شود. الگوریتم δ-حریصانه مشابه الگوریتم مسیر-حریصانه عمل می‌کند با این تفاوت که هرگاه زوج نقطه (p, q) در نظر گرفته می‌شود، قبل از اینکه طول کوتاه‌ترین مسیر بین p و q محاسبه شود، بررسی می‌شود که آیا دو نقطه مانند p و r وجود دارد به طوری که طول کوتاهترین مسیر بین آن‌ها کم و زاویه بین دو بردار $\overrightarrow{pr}$ و $\overrightarrow{pq}$ حداکثرمقدار θ ($\theta \leq \pi/4$) باشد. اگر چنین زوج نقطه‌ای وجود نداشته باشد، آنگاه طول کوتاه‌ترین مسیر بین p و q محاسبه گردد و مانند الگوریتم مسیر-حریصانه ادامه الگوریتم طی می‌گردد. اگر چنین زوج نقطه‌ای وجود داشته باشد آن‌گاه دیگر لازم نیست که طول کوتاه‌ترین مسیر بین p و q محاسبه گردد و بنابراین زوج نقطه بعدی بررسی می‌گردد. واضح است که با این روش، در واقع تعداد SPQها کاهش می‌یابد. با مشاهده این الگوریتم، اولین فرضیه ما این است که کران بالای $\pi/4$ برای زاویه θ را می‌توان به $\pi/3$ افزایش داد و در نتیجه تعداد SPQها در مقایسه با الگوریتم δ-حریصانه کاهش می‌یابد.

همانطور که اشاره شد، الگوریتم δ-حریصانه در هر تکرار قبل از بررسی محاسبه طول کوتاه‌ترین مسیر، ممکن است که زوج نقاطی را نادیده بگیرد و در نتیجه تعداد SPQها کاهش می‌یابد. به نظر می‌رسد که هنوز زوج نقاطی هستند که می‌توانند در حین اجرای الگوریتم نادیده گرفته شوند. دومین فرضیه این پژوهش این است که هنگام بررسی یک زوج نقطه (p, q) توسط الگوریتم، اگر زوج نقطه‌ای مانند (r, s) که قبلاً بررسی شده است، وجود داشته باشد که به طرز معناداری نزدیک (p, q) باشد، آنگاه می‌توان (p, q) را نادیده گرفت و درنتیجه تعداد SPQها را بیشتر از قبل کاهش داد. سومین فرضیه ما در این پژوهش این است که اگر توزیع نقاط در صفحه به صورت نرمال باشد، آنگاه در زمان کمتری تعداد زوج نقاط بیشتری را می‌توان نادیده گرفت و درنتیجه این به کاهش بیشتر زمان اجرا منجر می‌شود.

چهارمین فرضیه ما در این پژوهش این است که با کمک تجزیه زوجی خوش-مجزا می‌توان در زمان $O(n^2 \log n)$ الگوریتم مسیر-حریصانه را خیلی ساده‌تر از الگوریتم بوز و همکارنش [۴] پیاده سازی کرد به طوری که تعداد SPQها خیلی کمتر از آنچه در بالا اشاره شد، گردد. لازم به ذکر است که عیب الگوریتم بوز و همکارنش [۴] این است که پیاده سازی آن پیچیده و مشکل است.

۳ ضرورت انجام پژوهش

همانطور که در بخش اول ذکر شد، t-پوشاننده‌ها کاربردهای زیادی دارند. t-پوشاننده‌ی مسیر-حریصانه به عنوان یکی از باکیفیت‌ترین پوشاننده‌ها هم از دیدگاه عملی و هم از دیدگاه نظری شناخته شده است (از نظر تعداد یال، بیشترین درجه رئوس و وزن گراف) [۶، ۷]. عیب اصلی پوشاننده مسیر-حریصانه فقط زمان زیاد ساخت آن است. البته الگوریتمی با عنوان حریصانه-تقریبی[۹] وجود دارد که از دیدگاه نظری، گراف تولید شده توسط آن دارای خصوصیاتی مشابه پوشاننده مسیر-حریصانه است و همچنین در زمان بهینه یعنی $O(n \log n)$ نیز ساخت می‌شود [۵، ۸]. اما واقعاً در عمل هیچ ارتباطی بین انتظارات ما از گراف تولید شده توسط الگوریتم حریصانه-تقریبی و نتایج تولید شده وجود ندارد [۶]. جدا از این، پیاده سازی این الگوریتم بسیار پیچیده و مشکل است. با توجه به با کیفیت بودن پوشاننده مسیر-حریصانه، جهت رفع عیب اصلی آن، تلاش‌های زیادی انجام گرفته است که آخرین نتیجه‌ای که در این راستا بدست آمده است همان الگوریتم δ-حریصانه است که تمرکز آن روی کاهش تعداد SPQهاست. نتایج اولیه ما نشان می‌دهد که می‌توان تعداد SPQها را به طرز چشمگیری کاهش داد.

---

[۴] minimum spanning tree
[۵] well-separated pair decomposition
[۶] proximity problem
[۷] Shortest path query
[۸] δ-greedy
[۹] Approximate-Greedy

۴ روش انجام پژوهش

مهمترین لم که با کمک آن، نویسندگان مقاله [۱] توانستند تعداد SPQها را کاهش دهند، لم زیر است.

لم ۱ ([۱]). فرض کنید t، δ و θ اعداد حقیقی باشند به طوری که $1 \leq \delta \leq t$ و $0 \leq \theta \leq \frac{\pi}{4}$. فرض کنید p و q و r نقاطی در صفحه باشند به طوری که

۱. $p \neq r$،

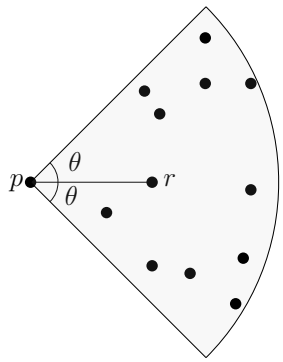
۲. $|pr| \leq |pq|$،

۳. $\frac{1}{\cos\theta - \sin\theta} \leq \frac{t}{\delta}$، که θ زاویه $\angle rpq$ است $(\angle rpq = \theta \leq \frac{\pi}{4} - \arcsin(\frac{\delta}{\sqrt{2}\cdot t}))$،

بنابراین، $\delta|pr| + t|rq| \leq t|pq|$.

بررسی‌های اولیه ما نشان می‌دهد که می‌توان نتایج لم ۱ را به این صورت بهبود داد که مقدار کران $\pi/4$ را به $\pi/3$ افزایش داد. به نظر می‌رسد این بهبود تأثیر بسیار زیاد در کاهش SPQها دارد.

همان‌طور که اشاره کردیم تمرکز الگوریتم δ-حریصانه روی کاهش تعداد SPQها در محاسبه پوشاننده مسیر-حریصانه است و این کار را به این صورت انجام می‌دهد که هرگاه زوج نقطه (p, q) در نظر گرفته می‌شود، قبل از اینکه طول کوتاه‌ترین مسیر بین p و q محاسبه شود، بررسی می‌شود که آیا دو نقطه مانند p و r وجود دارد به طوری که طول کوتاهترین مسیر بین آن‌ها کم و زاویه بین دو بردار $\overrightarrow{pr}$ و $\overrightarrow{pq}$ حداکثرمقدار $\theta \leq \pi/4$ باشد. به عبارت دیگر، اگر فرض کنیم که $c_p(2\theta, r)$ یک مخروط شبیه شکل زیر باشد، آنگاه تمام زوج نقاط (p, q) که q داخل این مخروط قرار می‌گیرد توسط الگوریتم نادیده گرفته می‌شود (شکل ۱).



شکل ۱: مخروط $c_p(2\theta, r)$.

یکی از لم‌های هندسی خیلی مفید که در کتاب Geometric Spanner Networks [۹] آمده است به صورت زیر است.

لم ۲ ([۹]). فرض کنید که t، θ و w اعداد حقیقی باشند به طوری که $0 < \theta < \pi/4$، $0 \leq w < \frac{\cos\theta - \sin\theta}{2}$ و $t \geq \frac{1}{\cos\theta - \sin\theta - 2w}$. فرض کنید که p، q، r و s نقاطی در فضای $\mathbb{R}^d$، به طوری که

۱. $p \neq q$، $r \neq s$،

۲. $angle(pq, rs) \leq \theta$،

۳. $|pr| \leq w|rs|$،

۴. $|rs| \leq |pq|/\cos\theta$،

بنابراین، $|pr| + |rs| + t|sq| \leq t|pq|$.

بررسی‌های اولیه ما نشان می‌دهد که علاوه بر زوج نقاط (p, q) که q در مخروط $c_p(2\theta, r)$ قرار می‌گیرند، زوج نقاط (p, q) که برای آن زوجی مانند (r, s) وجود دارد به طوری که در لم ۲ صدق می‌کنند را هم می‌توان نادیده گرفت.

همچنین در این پژوهش، در نظر داریم که الگوریتم‌های پیشنهادی را با یکی از زبان‌های برنامه نویسی پیاده سازی نماییم و عملکرد آن‌ها را با الگوریتم‌های قبلی به ویژه الگوریتم δ-حریصانه مقایسه کنیم.

۵ کلمات کلیدی

فارسی:
الگوریتم مسیر-حریصانه، الگوریتم δ-حریصانه، پوشاننده مسیر-حریصانه، کوتاهترین مسیر
انگلیسی:
δ-greedy algorithm, path-greedy algorithm, path-greedy spanner, Shortest path

فهرست منابع و مآخذ:

[1] A. K. Abu-Affash, G. Bar-On, and P. Carmi. δ-greedy t-spanner. *Computational Geometry*, **100**(2022), 101807.

[2] S. P. Alewijnse, Q. W. Bouts, P. Alex, and K. Buchin. Computing the greedy spanner in linear space. In *Annual European Symposium on Algorithms*, pages 37–48. Springer, 2013.

[3] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, **9**(1)(1993), 81–100.

[4] P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, **58**(3)(2010), 711–729.

[5] G. Das and G. Narasimhan. A fast algorithm for constructing sparse euclidean spanners. *International Journal of Computational Geometry & Applications*, **7**(04)(1997), 297–315.

[6] M. Farshi and J. Gudmundsson. Experimental study of geometric t-spanners. *Journal of Experimental Algorithmics (JEA)*, **14**(2010), 3–39.

[7] M. Farshi and M. HekmatNasab. Greedy spanner algorithms in practice. *Scientia Iranica*, **21**(6)(2014), 2142–2152.

[8] C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, **32**(1)(2002), 144–156.

[9] G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

[10] G. Navarro, R. Paredes, and E. Chávez. t-spanners for metric space searching. *Data & Knowledge Engineering*, **63**(3)(2007), 820–854.

[11] S. B. Rao and W. D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 540–550. ACM, 1998.

[12] D. Russel and L. J. Guibas. Exploring protein folding trajectories using geometric spanners. In *Pacific Symposium on Biocomputing*, pages 42–53. World Scientific, 2005.

[13] J. Soares. Approximating euclidean distances by small degree graphs. *Discrete & Computational Geometry*, **11**(1)(1994), 213–233.