سال دوازدهم، شماره‌های ۴۵ و ۴۶، پاییز و زمستان ۱۳۹۹
صص: ۱۴۵-۱۵۸

دو فصلنامه علمی
فناوری اطلاعات و ارتباطات ایران

# اندازه‌گیری میزان تشابه مسیرهای جهت‌دار بر روی داده‌های هندسی

**زینب سعیدی*** **محمد فرشی****

* دانشجوی دکتری، گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه یزد

** استادیار، گروه علوم کامپیوتر، دانشکده علوم ریاضی، دانشگاه یزد

تاریخ دریافت: ۱۳۹۹/۰۶/۰۱۰ تاریخ پذیرش: ۱۳۹۹/۱۰/۲۰

نوع مقاله: پژوهشی

## چکیده

در این مقاله به بررسی مسئله تشابه زیر در حوزه فاصله فرشه می‌پردازیم. یک مسیر جهت‌دار $\pi$ به عنوان ورودی و یک پاره‌خط افقی $Q$ که در لحظه پرس‌وجو توسط کاربر ارائه می‌شود، داده شده‌اند، هدف پیش‌پردازش و ذخیره مسیر جهت‌دار $\pi$ در یک ساختمان داده است به‌طوری‌که با توجه به اطلاعات ذخیره شده در ساختمان داده بتوان زیرمسیری از مسیر جهت‌دار را گزارش کرد که فاصله فرشه میان زیرمسیر گزارش‌شده و پاره‌خط افقی $Q$ بین تمام زیرمسیرهای ممکن مینیمم باشد. تا آنجایی که ما اطلاع داریم هیچ‌گونه نتیجه تئوری برای این مسئله گزارش نشده است. در این مقاله اولین الگوریتم ابتکاری برای مسئله ارائه شده است و به دلیل عدم ارائه الگوریتمی برای حل این مسئله در گذشته، صرفاً کیفیت الگوریتم ارائه شده بر روی چند پایگاه داده بررسی می‌گردد.

**واژگان کلیدی:** ساختمان داده، فاصله فرشه، فاصله هاسدورف، تشابه، مسیر جهت‌دار

## ۱. مقدمه

توصیف عددی دور بودن دو شی از هم فاصله نامیده می‌شود. در ریاضیات، تابع فاصله یا متر، تعمیمی از مفهوم فاصله فیزیکی است. متر، یک راه برای توصیف مفهوم نزدیک یا دور بودن عناصر یک فضا از هم است. اندازه‌گیری همسانی بین دو شی هندسی، یک مسئله اساسی در زمینه‌های بسیاری از علوم و مهندسی می‌باشد. برای مهیا کردن چنین مقایسه‌هایی، یک معیار خوب برای فرمول-بندی میزان تشابه یا همسانی لازم می‌باشد.

فاصله هاسدورف یک ابزار برای اندازه‌گیری میزان تشابه بین دو منحنی است. این فاصله برای اولین بار توسط فلیکس هاسدورف[۱] در سال ۱۹۱۴ میلادی در کتابش معرفی شد [۱]. در سال ۲۰۰۹، آلت[۲] نشان داد که فاصله هاسدورف میان دو چندضلعی با $n$ و $m$ رأس در زمان $\mathcal{O}((n+m)\log(n+m))$ قابل محاسبه است. یکی از کاربردهای فاصله هاسدورف در گرافیک کامپیوتری برای اندازه‌گیری تفاوت بین دو دید مختلف از یک شی می‌باشد [۳] [۲]. فاصله هاسدورف به صورت غیر رسمی برابر با ماکزیمم فاصله بین دو نقطه از دو منحنی است زمانی که هر نقطه از یک منحنی به

نویسنده مسئول: محمد فرشی mfarshi@yazd.ac.ir

[۱] Felix Hausdorff
[۲] Alt

نزدیک‌ترین نقطه‌اش در منحنی دیگر نگاشت گردد. یکی از مزیت-های فاصله هاسدورف این است که به آسانی قابل توصیف است. فاصله هاسدورف جهت‌دار از منحنی $A$ به منحنی $B$ به صورت زیر تعریف می‌شود:

$$\delta_{\vec{h}}(A,B)=\sup_{a\in A}\inf_{b\in B} dist(a,b) \qquad (۱)$$

فاصله هاسدورف بین $A$ و $B$ نیز به صورت زیر تعریف می‌شود:

$$\delta_H(A,B)=\max\left\{\delta_{\vec{h}}(A,B),\delta_{\vec{h}}(B,A)\right\} \qquad (۲)$$

فاصله فرشه توسط موریس رنه فرشه[۱] در سال ۱۹۰۶ میلادی معرفی شد [۴]. یک مطالعه پایه روی ویژگی‌های محاسباتی فاصله فرشه توسط آلت و گوداو[۲] در سال ۱۹۹۲ انجام شد [۵]. آنها الگوریتمی ارائه دادند که فاصله فرشه بین دو چندضلعی را در زمان $\mathcal{O}(nm\log^2(nm))$ محاسبه می‌کند که $n$ و $m$ تعداد رئوس چندضلعی‌ها هستند. در سال ۱۹۹۵، آلت و گوداو نشان دادند که فاصله فرشه بین دو چندضلعی در زمان $\mathcal{O}(nm\log(nm))$ قابل محاسبه است و تا کنون هم نتیجه بهتری گزارش نشده است[۶]. فاصله فرشه دارای کاربردهای متنوعی از جمله تشخیص گفتار، تشخیص امضا و تشخیص دست‌خط است[۷].

فاصله فرشه به طور غیررسمی به صورت زیر تعریف می‌شود. یک شخص و سگش را در نظر بگیرید به گونه‌ای که سگ با یک قلاده به صاحبش وصل شده است. هر کدام از این دو روی یک مسیر جهت‌دار (منحنی) از نقطه ابتدایی آن تا انتهایش در حال حرکت می‌باشند. هر دوی آنها اجازه دارند که سرعت خود را کنترل کنند اما نمی‌توانند به عقب برگردند. هزینه هر پیاده‌روی برابر با حداکثر طول قلاده‌ای است که شخص را به سگش وصل می‌کند. پیاده-روی‌های متفاوت هزینه‌ی متفاوت دارند. فاصله فرشه برابر با هزینه پیاده‌روی است که دارای هزینه مینیمم در بین تمام پیاده‌روی‌های ممکن است.

فاصله فرشه بین منحنی $A$ و $B$ به‌صورت زیر تعریف می‌شود:

$$\delta_F(A,B)=\inf_{\mu}\max_{a\in A} dist\big(a,\mu(a)\big) \qquad (۳)$$

که $dist(a,\mu(a))$ فاصله اقلیدسی را نشان می‌دهد و $\mu: A\to B$ یک تابع پیوسته غیرنزولی است که هر نقطه $a\in A$ را به نقطه $\mu(a)$ در $B$ نگاشت می‌کند. تحت نگاشت $\mu$، نقطه شروع $A$ به نقطه شروع $B$ نگاشت می‌شود که مطمئن باشیم دو تا منحنی در یک جهت پیموده می‌شوند. مزیت فاصله فرشه نسبت به معیارهای دیگر این است که فاصله فرشه ترتیب نقاط در امتداد منحنی‌ها را رعایت می‌کند.

## ۲. مروری بر کارهای گذشته

دسته جالبی از مسائل که در حوزه فاصله فرشه مطرح می‌شود، ارائه ساختمان داده کارآمد برای مسائل پرس‌وجوی فرشه است. به این صورت که یک مسیر جهت‌دار داده شده است. می‌خواهیم با انجام یک پیش‌پردازش آن را در یک ساختمان داده ذخیره کنیم به صورتی که با توجه به اطلاعات ذخیره شده در ساختمان داده بتوانیم فاصله فرشه میان مسیر جهت‌دار و هر پاره‌خطی که در زمان پرس‌وجو داده می‌شود را محاسبه کنیم و یا تعداد زیرمسیرهایی را شمارش کنیم که فاصله فرشه آنها از پاره‌خطی که در زمان پرس‌وجو داده می‌شود، کوچکتر یا مساوی مقدار آستانه مشخص شده در زمان پرس‌وجو باشد.

دی برگ[۳] و همکارانش در سال ۲۰۱۳ مسئله پرس‌وجوی فرشه در مسیر جهت‌دار را مورد بررسی قرار دادند. آنها برای ساده‌تر کردن مسئله، پرس‌وجو را به صورت پاره‌خط افقی در نظر گرفتند و تعداد زیرمسیرهایی که فاصله فرشه آنها از پاره‌خط افقی داده شده حداکثر برابر $\varepsilon$ است را شمارش کردند[۸]. ساختمان داده‌ای که توسط دی برگ و همکارانش ارائه شد، این مسئله را به صورت تقریبی حل می‌کند. این ساختمان داده، علاوه بر تمام زیرمسیرهایی با فاصله $\varepsilon$، شامل تعدادی زیرمسیر اضافه است که فاصله فرشه آنها از پاره‌خط داده شده حداکثر برابر با $(۲+۳\sqrt{۲})\varepsilon$ است. آنها فرض کردند که طول پاره‌خط داده شده در پرس‌وجو حداقل $۶\varepsilon$ است و مسئله را در دو حالت بررسی کردند. الف) حالتی که $\varepsilon$ از قبل مشخص است. ب) حالتی که $\varepsilon$ جزء پرس‌وجو است و از قبل مشخص نمی‌باشد. آنها با طراحی یک ساختمان داده چندسطحی با زمان پیش‌پردازش $\mathcal{O}(n^2+s\times polylog\, n)$ و فضای ذخیره‌سازی $\mathcal{O}(s\times polylog\, n)$ حالت (الف) مسئله را حل کردند که $n$ تعداد رئوس مسیر است و s پارامتری است که بین $n$ و $n^2$ قرار دارد. زمان پاسخ به پرس‌وجو $\mathcal{O}\left(\left(\frac{n}{\sqrt{s}}\right)polylog\, n\right)$ است. با افزایش زمان پیش‌پردازش به $\mathcal{O}(n^3 log\, n)$ می‌توان مسئله را در حالت (ب) حل کرد. فضای ذخیره‌سازی و زمان پاسخ به پرس‌وجو نسبت به حالت (الف) تغییری نمی‌کند.

گودموندسون[۴] و اسمید[۵] در سال ۲۰۱۳ مسئله پرس‌وجوی فرشه در درخت هندسی (درختی که بر روی یک مجموعه از نقاط در صفحه ساخته شود) را مورد بررسی قرار دادند[۹]. درخت $T$ و مقدار $\Delta$ را به عنوان ورودی داریم. پرس‌وجو $Q$ به صورت مسیر در نظر گرفته شده است. آنها در پی یافتن ساختمان داده‌ای برای ذخیره

[۱] Maurice René Fréchet
[۲] Godua
[۳] de Berg
[۴] Gudmundsson
[۵] Smid

کردن درخت $T$ بودند که بتواند در زمان پرس‌وجو مشخص کند آیا درخت $T$ شامل مسیری مانند $P$ است که فاصله فرشه آن از مسیر داده شده در پرس‌وجو کمتر از $\Delta$ باشد. آنها مسئله را در حالتی که درخت $T$ به صورت $c$-بسته‌بندی (درختی که کل طول قرار گرفته از آن داخل هر دیسک حداکثر $c$ برابر شعاع دیسک باشد) است و طول هر کدام از یال‌های درخت و مسیر داده شده در پرس‌وجو از $\Omega(\Delta)$ است، بررسی کردند. ساختمان داده ارائه شده دارای زمان ساخت و فضای ذخیره‌سازی $\mathcal{O}(n\,polylog\,n)$ است و برای مسیر $Q$ داده شده در پرس‌وجو با $m$ رأس می‌تواند در زمان $\mathcal{O}(m\,polylog\,n)$ پاسخ آری یا نه را بدهد. اگر ساختمان داده پاسخ نه بدهد، حتماً مسیری در $T$ وجود نداشته است که فاصله آن از $Q$ کمتر از $\Delta$ باشد. اگر پاسخ آری بدهد و $m = ۲$ باشد، به این معنی است که مسیری در $T$ وجود دارد که فاصله فرشه آن از $Q$ حداکثر برابر با $\sqrt{۲}(۱+\epsilon)\Delta$ است. اگر پاسخ آری بدهد و $m > ۲$ باشد، به این معنی است که مسیری در $T$ وجود دارد که فاصله فرشه آن از $Q$ حداکثر برابر با $۳\Delta$ است. یعنی در حالت آری پاسخ تقریبی خواهیم داشت. آنها در سال ۲۰۱۵ [۱۸] ضریب $\sqrt{۲}$ را از کران بدست آمده برای حالتی که $m = ۲$ است، حذف کردند و نتیجه بهتری را بدست آورند. اما برای حالت $m > ۲$ کران بدست آمده جدید برابر با $۳(۱+\epsilon)\Delta$ است.

دریمل[۱] و هارپلد[۲] در سال ۲۰۱۳ مسئله پرس‌وجوی فرشه با کمک یال میان‌بر را بررسی کردند. آنها ساختمان داده‌ای ارائه کردند که می‌تواند دو مسئله‌ای که در ادامه مطرح می‌شود را حل کند[۱۰].

• منحنی $\pi$ که دارای $n$ رأس است را به عنوان ورودی داریم. پرس‌وجو، منحنی $Q$ شامل $k$ رأس و رئوس $p$ و $q$ است. هدف، ذخیره منحنی $\pi$ در یک ساختمان داده است به صورتی که بتوانیم به ازای هر $Q$، $p$ و $q$ که در پرس‌وجو داده می‌شود، فاصله فرشه میان زیرمنحنی از $\pi$ که بین $p$ و $q$ قرار دارد و منحنی $Q$ را محاسبه کنیم. ساختمان داده ارائه شده دارای زمان ساخت $\mathcal{O}(n\log^{۳}n)$ و فضای ذخیره‌سازی $\mathcal{O}(n\log n)$ است. زمان پاسخ به پرس‌وجو $\mathcal{O}(k^{۲}\log n\log(k\log n))$ است. ساختمان داده ارائه شده، فاصله فرشه تقریبی با ضریب تقریب ثابت ۲۳ را محاسبه می‌کند.

• منحنی داده شده در پرس‌وجو را به صورت پاره‌خط افقی در نظر می‌گیرد. ساختمان داده ارائه شده دارای زمان ساخت $\mathcal{O}(\chi^{۲}n\log^{۲}n)$ و فضای ذخیره‌سازی $\mathcal{O}(\chi^{۲}n)$ است که $\chi = \epsilon^{-d}\log(۱/\epsilon)$. ساختمان داده ارائه شده در زمان $\mathcal{O}(\epsilon^{-۲}\log n\,\mathrm{loglog}\,n)$ فاصله فرشه میان زیرمنحنی از $\pi$ که بین $p$ و $q$ قرار دارد و پاره‌خط افقی $Q$ را با ضریب تقریب $(۱+\epsilon)$ محاسبه می‌کند.

دی برگ و مهرابی در سال ۲۰۱۵ مسئله یافتن زیرمسیرهای مستقیم در مسیر جهت‌دار را بررسی کردند. آنها دو معیار متفاوت برای مستقیم بودن ارائه کردند و با توجه به آن دو تعریف، مسئله را بررسی کردند. معیار اول، ضریب کشش و معیار دوم انحراف جهت است. با توجه به این دو معیار ساختمان داده‌های زیر طراحی شده است[۱۱].

• ضریب کشش: مسیر جهت‌دار $\pi$ و مقدار $\tau$ را به عنوان ورودی داریم. در پرس‌وجو دو مکان $s$ و $t$ که به ترتیب نشان دهنده نقطه شروع و نقطه پایان است، داده می‌شود. هدف گزارش تمام زیرمسیرهایی از $\pi$ است به صورتی‌که نقطه شروع و پایان آن به ترتیب $s$ و $t$ باشد و ضریب کشش آن حداکثر $\tau$ باشد. آنها صفحه را به سلول‌های مربعی تقسیم‌بندی کردند. به عبارتی صفحه را به صورت مشبکه $m \times m$ در نظر گرفتند. ساختمان داده ارائه شده نیازمند فضای ذخیره‌سازی $\mathcal{O}(\min(n^{۲}, nm^{۲}, \tau nm))$ است که در زمان مورد انتظار $\mathcal{O}(n^{۱/۵+\epsilon}+k)$ ساخته می‌شود و $n$ تعداد رئوس مسیر جهت‌دار و $k$ تعداد کل زیرمسیرهای جهت‌دار با شرایط ذکر شده است.

• انحراف جهت: مسیر جهت‌دار $\pi$ و مقدار $\alpha$ را به عنوان ورودی داریم. در پرس‌وجو دو مکان $s$ و $t$ که به ترتیب نشان دهنده نقطه شروع و نقطه پایان است، داده می‌شود. هدف گزارش تمام زیرمسیرهایی از $\pi$ است که نقطه شروع و پایان آن‌ها به ترتیب $s$ و $t$ هستند و انحراف جهت آن حداکثر $\alpha$ باشد که انحراف جهت مقدار بیشینه زاویه بین هر دو پاره‌خط متوالی مسیر جهت‌دار است. آنها صفحه را به سلول‌های مربعی تقسیم‌بندی کردند. به عبارتی صفحه را به صورت مشبکه در نظر گرفتند. ساختمان داده ارائه شده نیازمند فضای ذخیره‌سازی $\mathcal{O}(n^{۲}, nm^{۲}, nm/\cos(\frac{\alpha}{۲}))$ است که در زمان $\mathcal{O}(n+k)$ ساخته می‌شود و $n$ تعداد رئوس مسیر جهت‌دار و $k$ تعداد کل زیرمسیرهای جهت‌دار با شرایط ذکر شده است.

مسئله‌ی دیگری که در سال ۲۰۱۷ توسط دی برگ و همکارانش مورد بررسی قرار گرفته است، محاسبه فاصله فرشه میان مسیر جهت‌دار $\pi$ و پرس‌وجو ورودی $Q$ که به صورت پاره‌خط افقی فرض شده است، می‌باشد[۱۲]. هدف طراحی ساختمان داده‌هایی که برای مسیر $\pi$ بتوانند برای هر پاره‌خطی که به عنوان پرس‌وجو داده می‌شود، فاصله فرشه میان پاره‌خط و مسیر جهت‌دار را به صورت دقیق محاسبه کنند. آن‌ها ساختمان داده‌ای ارائه دادند که فاصله فرشه میان مسیر جهت‌دار $\pi$ را از هر پاره‌خط داده شده در

[۱] Driemel
[۲] Har-Peled

پرس‌وجو با زمان پیش‌پردازش $\mathcal{O}(n^2 \log n)$ و فضای ذخیره‌سازی $\mathcal{O}(n^2)$ و زمان پرس‌وجوی $\mathcal{O}(\log^2 n)$ محاسبه می‌کند. دی برگ و همکارانش ساختمان داده دیگری طراحی کردند که فاصله فرشه میان زیرمسیری که توسط دو رأس داده شده در پرس‌وجو مشخص می‌شود و هر پاره‌خط دلخواهی که در زمان پرس‌وجو داده می‌شود را محاسبه کند. زمان پیش‌پردازش ساختمان داده ارائه شده $\mathcal{O}(n^2 \log^2 n)$ و پیچیدگی فضای $\mathcal{O}(n^2 \log^2 n)$ است. زمان پاسخ به پرس‌وجو برابر با $\mathcal{O}(\log^2 n)$ است.

بوخین و همکارانش در سال ۲۰۲۰ فضای ذخیره‌سازی ساختمان داده برای محاسبه فاصله فرشه میان مسیر جهت‌دار $\pi$ و پاره‌خط افقی داده شده در پرس‌وجو را به $\mathcal{O}(n^{3/2})$ کاهش دادند[۱۳].

گودموندسون و همکارانش در سال ۲۰۱۷ مسئله پرس‌وجوی فرشه برای منحنی با طول یال‌های طویل را مورد بررسی قرار دادند. ورودی مسئله منحنی $P$ شامل $n$ رأس است[۱۴]. $l_P$ طول کوتاه‌ترین یال منحنی $P$ است. در پرس‌وجو، منحنی $Q$ با $m$ رأس و مقدار $\Delta$ داده می‌شود. $l_Q$ طول کوتاه‌ترین یال منحنی $Q$ است. منحنی $P$ و $Q$ دارای یال‌هایی با شرط $l_P > 2 * \Delta$ و $l_Q > (1 + \sqrt{2})\Delta$ هستند. شرط‌ها در واقع نشان دهنده این هستند که مسئله برای منحنی با طول یال‌های طویل است. هدف، مشخص کردن این است که فاصله فرشه میان $P$ و $Q$ حداکثر مقدار $\Delta$ است یا نه. گودموندسون و همکارانش ساختمان داده‌ای ارائه کردند که با زمان پیش‌پردازش و فضای ذخیره‌سازی $\mathcal{O}(n \log n)$ به پرس‌وجو در زمان $\mathcal{O}(m \log^2 n)$ پاسخ می‌دهد.

دی برگ و همکارانش در سال ۲۰۱۷ ساختمان داده پویا برای پرس‌وجوی مجاورت در مجموعه‌ای از مسیرهای جهت‌دار ارائه کردند. برخلاف کارهای قبلی که ورودی مسئله تنها یک مسیر جهت‌دار بود، در این مقاله ورودی مسئله مجموعه $S$ شامل $n$ مسیر جهت‌دار و مقدار ثابت $k$ است که تعداد رئوس مسیر جهت‌داری است که در زمان پرس‌وجو داده می‌شود. ساختمان داده طراحی شده توسط دی برگ و همکارانش می‌تواند پرس‌وجوهای زیر را پاسخ بدهد[۱۵].

۱. نزدیک‌ترین همسایه: پرس‌وجو مسیر جهت‌دار $Q$ است. هدف یافتن مسیری از بین مسیرهای جهت‌دار موجود در $S$ است که فاصله فرشه آن از $Q$ کمینه باشد.

۲. $j$-تا نزدیک‌ترین: پرس‌وجو، مسیر جهت‌دار $Q$ و مقدار صحیح $j$ است. هدف گزارش $j$ مسیر از بین مسیرهای جهت‌دار موجود در $S$ است که فاصله فرشه آنها از $Q$ کمینه است.

۳. پرس‌وجوی بازه‌ای: پرس‌وجو، مسیر جهت‌دار $Q$ و مقدار $\delta_{Max}$ است. هدف گزارش کردن تمام مسیرهای جهت‌دار است که فاصله فرشه آنها از $Q$ حداکثر $\delta_{Max}$ است.

۴. تشابه: پرس‌وجو، مسیر جهت‌دار $Q$ و مسیر جهت‌دار $\tau$ موجود در مجموعه ورودی است. هدف محاسبه فاصله فرشه میان $Q$ و $\tau$ است.

ساختمان داده ارائه شده، مسائلی که در بالا بیان شد را به صورت تقریبی حل می‌کند. ضریب تقریب حداکثر برابر با $\varepsilon.reach(Q)$ است که $reach(Q)$ حداکثر فاصله اقلیدسی میان نقطه شروع $Q$ و سایر رئوس دیگر $Q$ است. فضای مورد نیاز برای ساختن ساختمان داده $\mathcal{O}(n/\varepsilon^{2*k})$ است. زمان پاسخ به پرس‌وجو برای مسائل ذکر شده در بالا به ترتیب به صورت زیر است:

۱. نزدیک‌ترین همسایه: $\mathcal{O}(1)$

۲. $j$-تا نزدیک‌ترین: $\mathcal{O}(j)$

۳. پرس‌وجوی بازه‌ای: $\mathcal{O}(1 + K)$ $K$ تعداد مسیرهای جهت‌دار خروجی است.

۴. تشابه: $\mathcal{O}(1)$

زمان درج کردن یک مسیر جهت‌دار با $m$ رأس در ساختمان داده به صورت سرشکن برابر با $\mathcal{O}(1/\varepsilon^{2*k}(m + \log(1/\varepsilon)))$ است. زمان حذف کردن یک مسیر جهت‌دار با $m$ رأس از ساختمان داده به صورت سرشکن برابر با $\mathcal{O}(1/\varepsilon^{2*k})$ است.

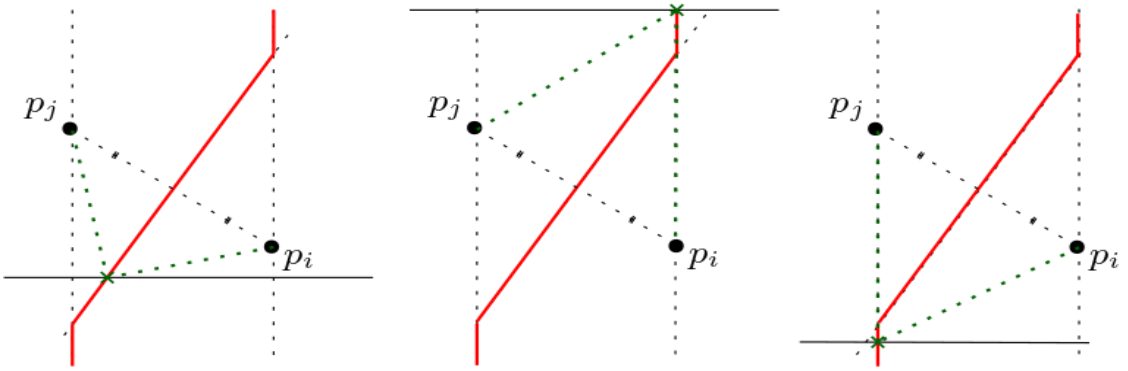
پرس‌وجوی فاصله فرشه برای مسئله جستجوی بازه‌ای بین یک مجموعه از مسیرهای جهت‌دار و یک پرس‌وجو به صورت تجربی مورد بررسی قرار گرفته است و کیفیت الگوریتم بر روی پایگاه داده گزارش شده است[۱۶، ۱۷، ۱۸].

در این مقاله مسئله تشابه در حوزه فاصله فرشه را به صورت تجربی بررسی می‌کنیم. به این صورت که مسیر جهت‌دار $\pi$ با $n$ رأس به عنوان ورودی داده شده است. در زمان پرس‌وجو پاره‌خط افقی $Q$ توسط کاربر مشخص می‌شود. هدف ذخیره مسیر جهت‌دار $\pi$ در یک ساختمان داده است به صورتی که بتوانیم زیرمسیر $\pi'$ از مسیر جهت‌دار را گزارش کنیم که فاصله فرشه میان زیرمسیر $\pi'$ و پاره‌خط افقی $Q$ بین تمام زیرمسیرهای ممکن مینیمم باشد. زیرمسیر یک مسیر جهت‌دار همبند از مسیر اصلی است به‌صورتی که نقطه ابتدا و انتهای آن جزء رئوس مسیر اصلی هستند.

این مسئله بسیار پیچیده‌ای است و تا جایی که ما اطلاع داریم هیچ‌گونه نتیجه تئوری و تجربی برای این مسئله گزارش نشده است. راه حل اولیه برای یافتن پاسخ این است که فاصله فرشه میان تمامی زیرمسیرهای ممکن را محاسبه کنیم و سپس زیرمسیری که کمترین فاصله فرشه را دارا می‌باشد، گزارش کرد. با کمک ساختمان داده ارائه شده ، فاصله فرشه میان زیرمسیر و پاره‌خط افقی در زمان $\mathcal{O}(\log^2 n)$ قابل محاسبه است[۱۲]. از طرفی تعداد زیرمسیرهای ممکن $\mathcal{O}(n^2)$ می‌باشد. بنابراین می‌توان در زمان $\mathcal{O}(n^2 \log^2 n)$ هر پرس‌وجویی را پاسخ داد. اما این زمان در

عمل کارآمد نمی‌باشد و نیازمند ایده خلاقانه برای بهبود زمان هستیم.

در این مقاله یک الگوریتم ابتکاری برای مسئله ارائه خواهیم داد. این الگوریتم از دو مرحله تشکیل شده است. در مرحله اول، با کمک ساختمان داده جستجوی بازه‌ای یک زیرمسیر را مشخص می‌کنیم و سپس در مرحله دوم با کمک روش‌های ارائه شده، زیرمسیر اولیه را تغییر می‌دهیم تا به پاسخ بهینه نزدیک گردد.




**شکل ۱ . حالات مختلف برای محاسبه فاصله زوج برگشتی**

الگوریتم ارائه شده را بر روی چند پایگاه داده اجرا کرده‌ایم و کیفیت الگوریتم ارائه شده را برحسب درصد پرس‌وجوهایی که به درستی پاسخ داده می‌شوند، گزارش می‌کنیم.

## ۳. مفاهیم اولیه

مختصات هر نقطه $p$ در صفحه را به صورت $(x, y)$ نشان می‌دهیم. فرض می‌کنیم که $p_1, p_2, ..., p_n$ توالی از $n$ نقطه در صفحه است و $\pi = (p_1, p_2, ..., p_n)$ یک مسیر جهت‌دار در صفحه است که توسط توالی ذکر شده تعریف شده است. $Q = (p, q)$ یک خط افقی در صفحه است به صورتی که مختصات نقطه $p$ و $q$ به ترتیب برابر با $p = (x_., y)$ و $q = (x_1, y)$ است به صورتی که $y \in \mathbb{R}$ و $x_. \leq x_1$ فرض کنید که $u$ و $v$ دو رأس دلخواه از مسیر جهت‌دار $\pi$ باشد. زیرمسیر جهت‌دار که از رأس $u$ آغاز و به رأس $v$ ختم می‌گردد را با عبارت $\pi[u,v]$ نشان می‌دهند. زوج نقطه $(p_i, p_j)$ زوج برگشتی (رو به جلو) است اگر $j > i$ و نقطه $p_j$ سمت چپ (رأست) نقطه $p_i$ واقع شده باشد. یک یال که نقطه $p_i$ را به نقطه $p_{i+1}$ وصل می‌کند، یال برگشتی (رو به جلو) است اگر زوج $(p_i, p_{i+1})$ یک زوج برگشتی (رو به جلو) باشد.

فاصله زوج برگشتی، ماکزیمم فاصله میان نقطه $p_i$ و $p_j$ و یک نقطه بر روی پاره‌خط افقی است. نقطه‌ای بر روی پاره‌خط افقی که فاصله زوج برگشتی را مشخص می‌کند، نقطه تقاطع بین عمودمنصف خط واصل بین $p_i$ و $p_j$ و پاره‌خط افقی است، اگر نقطه تقاطع بین نوار عمودی $[p.x, q.x]$ واقع شده باشد. غیر اینصورت نقطه تقاطع بین خط $x = p_i.x$ یا $x = p_j.x$ و پاره‌خط افقی را باید محاسبه کرد. شکل ۱ حالات مختلف فاصله زوج برگشتی را نشان می‌دهد. فاصله زوج برگشتی برای یک زوج برگشتی $(p_i, p_j)$ به صورت زیر تعریف می‌گردد:

$$B_{(p_i, p_j)}(y) = \min_{x \in \mathbb{R}} \max\{\| p_i - (x,y) \|, \| p_j - (x,y) \|\}$$

عبارات داخل آکولاد فاصله بین نقطه داده شده $(x, y)$ و نقطه $p_i$ و نقطه $p_j$ را محاسبه می‌کند. فاصله زوج نقاط برگشتی مربوط به زیرمسیر جهت‌دار $\pi[u, v]$ به این‌گونه تعریف می‌شود:

$$B(\pi[u, v],y) = \max_{\forall p_i, p_j \in \pi[u, v]: i \leq j,\ p_i.x \geq p_j.x} B_{(p_i,p_j)}(y)$$

فاصله هاسدورف جهت‌دار بین زیرمسیر جهت‌دار $\pi[u,v]$ و پاره‌خط افقی $Q$ به‌صورت زیر محاسبه می‌شود:

$$\delta_{\vec{h}}(\pi[u, v], Q) = \max\{\max_{p_i.x \in (-\infty, x.]} \| p - p_i \|, \max_{p_i.x \in [x_1, \infty)} \| q - p_i \|, \max_i \| y - p_i.y \|\}$$

طبق نتایج موجود در [۱۲]، فاصله فرشه بین زیرمسیر جهت‌دار $\pi[u, v]$ و پاره‌خط افقی $Q$ به‌صورت زیر محاسبه می‌شود:

$$\delta_F(\pi[u, v], pq) = \max\left\{\| up \|, \| vq \|, \delta_{\vec{h}}(\pi[u, v], pq), B(\pi[u, v], y)\right\} \quad (۱-۴)$$

$\| up \|$ فاصله اقلیدسی میان نقطه $u$ و $p$ می‌باشد.

زیرمسیر جهت‌دار بهینه $\pi'_{opt} = \pi[u', v']$ زیرمسیری از $\pi$ است که فاصله فرشه آن از پاره‌خط افقی $pq$ بین تمامی زیرمسیرهای ممکن مینیمم است. بنابراین برای هر زیرمسیر جهت‌دار $\pi'$ از $\pi$ خواهیم داشت:

$$\forall \pi' \subseteq \pi: \delta_F(\pi'_{opt}, pq) \leq \delta_F(\pi', pq)$$

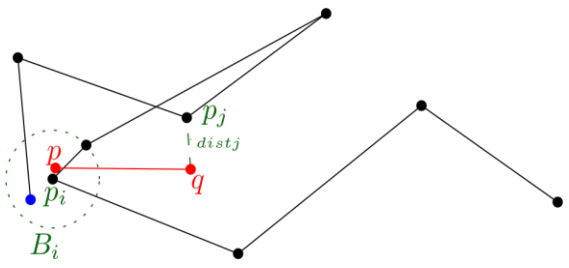
## ۴. الگوریتم پیشنهادی

در این قسمت یک الگوریتم ابتکاری ارائه خواهد شد که زیرمسیری از مسیر جهت‌دار را گزارش می‌کند که فاصله فرشه آن از پاره‌خط افقی میان تمام زیرمسیرهای ممکن مینیمم است. الگوریتم از دو مرحله تشکیل شده است که در مرحله اول نقطه ابتدا و انتها برای زیرمسیر مشخص می‌شود که این نقاط برأساس دو تا عبارت ابتدایی فرمول (۴) مشخص می‌گردند. در مرحله دوم

الگوریتم بررسی می‌شود که آیا سایر عبارات دیگر فرمول (۴) فاصله فرشه را اضافه می‌کنند یا نه. اگر سایر عبارت فرمول (۴) باعث ازدیاد فاصله فرشه گردد، زیرمسیر بدست آمده در مرحله اول تغییر داده می‌شود و در نهایت زیرمسیری گزارش می‌گردد.

نکته کلیدی برای بدست آوردن زیرمسیر $\pi'_{opt}$، مینیمم کردن دو عبارت ابتدایی فرمول (۴) است. برای مینیمم کردن دو عبارت ابتدایی، نزدیک‌ترین نقاط مسیر جهت‌دار $\pi$ به نقطه $p$ و نقطه $q$ محاسبه می‌گردد. این نقاط به ترتیب $p_i$ و $p_j$ نامیده می‌شود. با کمک ساختمان داده نزدیک‌ترین همسایه نقاط $p_i$ و $p_j$ محاسبه می‌شوند. اگر نقطه $p_i$ بعد از نقطه $p_j$ در مسیر $\pi$ واقع شده باشد، اصلاحاتی بر روی زیرمسیر اعمال می‌گردد به صورتی که بعد از آن نقطه $p_i$ قبل از نقطه $p_j$ واقع می‌گردد. قابل ذکر است که اگر از ابتدا نقطه $p_i$ قبل از نقطه $p_j$ واقع شده باشد، می‌توان از مرحله اول صرفنظر کرد و به مرحله دوم رفت. در غیر اینصورت تغییراتی را اعمال می‌کنیم به صورتی که بعد از آن نقطه $p_i$ قبل از نقطه $p_j$ قرار خواهد گرفت.

به منظور اعمال تغییرات، فواصل $disti = \| p_i - p \|$ و $distj = \| p_j - q \|$ محاسبه می‌گردد. اگر $disti < distj$ باشد، اندیس $i$ را می‌توان به رأسی قبل از نقطه $p_j$ منتقل کرد که فاصله آن از نقطه $p$ کمتر از $distj$ باشد. دیسک $B_i$ را با مرکز $p_i$ و شعاع $distj$ در نظر بگیرید. همانگونه که در شکل ۲ مشخص است، اگر دیسک $B_i$ مسیر جهت‌دار $\pi$ را قبل از نقطه $p_j$ قطع کند، نقطه $p_i$ به رأسی داخل دیسک $B_i$ قبل از $p_j$ انتقال داده می‌شود. در غیر اینصورت رأسی بین $p_i$ و $p_j$ را به عنوان اندیس جدید انتخاب می‌کنیم که ماکزیمم فاصله آن از نقاط $p$ و $q$ مینیمم باشد.




**شکل ۲: در این حالت i>j و disti<distj است. نقطه آبی رنگ داخل دیسک $B_i$، اندیس جدید برای i می‌باشد.**

اگر $disti > distj$ باشد، اندیس $j$ را می‌توان به رأسی بعد از نقطه $p_i$ منتقل کرد که فاصله آن از نقطه $q$ کمتر از $disti$ باشد. دیسک $B_j$ را با مرکز $p_j$ و شعاع $disti$ در نظر بگیرید. اگر دیسک $B_j$ مسیر جهت‌دار $\pi$ را بعد از نقطه $p_i$ قطع کند، نقطه $p_j$ به رأسی داخل دیسک $B_j$ بعد از $p_i$ انتقال داده می‌شود. اگر نتوانیم نقاط جدید در دیسک‌های $B_i$ و $B_j$ پیدا کنیم، رأسی بین $p_i$ و $p_j$ را به عنوان اندیس جدید انتخاب می‌کنیم که ماکزیمم فاصله آن از نقاط $p$ و $q$ مینیمم باشد. الگوریتم ۱ جزئیات مرحله اول الگوریتم را نشان می‌دهد.

**Algorithm ۱** Finding The Subtrajectory

**Input:** The trajectory $\pi$ and a horizontal segment $pq$
**Output:** Candidate subtrajectory

```
۱: Let i and j be the index of the closest vertices of π to p and q
۲: if (i > j) then
۳:   if (disti < distj and p_j ∈ B_i) then
٤:     change i into a vertex in B_i before p_j
٥:   else
٦:     if (disti > distj and p_i ∈ B_j) then
۷:       change j into index of vertex in B_j after p_i
۸:     else
۹:       i = j = The index of a vertex between j and i
           which minimize the maximum distance to p_i
           and p_j;
۱۰:    end if
۱۱:  end if
۱۲:end if
```

در مرحله دوم الگوریتم، ماکزیمم عباراتی که فاصله فرشه میان زیرمسیر انتخاب شده از مرحله اول و پاره‌خط افقی را مشخص می‌کند، محاسبه می‌کنیم. در پایگاه داده‌هایی که در این مقاله بررسی می‌کنیم، عبارت آخر هیچ گاه فاصله فرشه را مشخص نمی‌کند. دلیل اصلی این است که چون طول یال‌های مسیرهای جهت‌دار در پایگاه داده‌های مورد بررسی کوچک است، نوار عمودی ایجاد شده باریک است و عمودمنصف، پاره خط افقی را خارج از نوار عمودی قطع می‌کند (حالت میانی و سمت رأست شکل ۱). در این حالات، تصویر عمودی نقطه $p_i$ یا $p_j$ فاصله زوج برگشتی را مشخص می‌کند. بنابراین فاصله زوج برگشتی برای یک زیرمسیر جهت‌دار حداکثر برابر با فاصله هاسدورف است.

بر اساس مقدار ماکزیمم، حالات زیر را خواهیم داشت:

- اگر مقدار ماکزیمم عبارت $\| p_i - p \|$ یا $\| p_j - q \|$ باشد، الگوریتم زیرمسیر $\pi[p_i, p_j]$ را گزارش می‌کند. ازآن‌جایی که $p_i$ و $p_j$ نزدیک‌ترین نقاط به $p$ و $q$ هستند، بنابراین دو عبارت اول، کمترین مقدار ممکن خود را دارا هستند، و از طرفی ماکزیمم این دو عبارت فاصله فاصله فرشه را تعیین می‌کند، بنابراین زیرمسیر $\pi[p_i, p_j]$ کمترین فاصله فرشه از پاره‌خط افقی را دارا است. لذا خواهیم داشت:

$$\delta_F(\pi'_{opt}, pq) = \max\{ \| p_i - p \| , \| p_j - q \|\}$$

• اگر مقدار ماکزیمم عبارت $\delta_{\overrightarrow{h}}(\pi[p_i,\ p_j], pq)$ باشد، آنگاه ابتدا رأس $p_k$ بین $p_i$ و $p_j$ که فاصله هاسدورف جهت‌دار بین $\pi[p_i, p_j]$ و پاره‌خط افقی را تعیین می‌کند، مشخص می‌کنیم. با توجه به موقعیت این رأس سه حالت زیر را خواهیم داشت:

۱- اگر رأس $p_k$ سمت چپ پاره‌خط افقی واقع شده باشد، آنگاه اندیس $i$ را به رأسی قبل از $p_k$ منتقل می‌کنیم که فاصله آن از نقطه $p$ مینیمم باشد. بنابراین، $i = \min_{k\le k\prime\le j}\{\| p_{k\prime} - p \|\}$.

۲- اگر رأس $p_k$ سمت رأست پاره خط افقی واقع شده باشد، آنگاه اندیس $j$ را به رأسی قبل از $p_k$ منتقل می‌کنیم که فاصله آن از نقطه $q$ مینیمم باشد. بنابراین، $j = \min_{i\le k\prime\le k}\{\| p_{k\prime} - q \|\}$.

۳- اگر رأس $p_k$ بین نقطه $p$ و $q$ واقع شده باشد، آنگاه اندیس $i$ و $j$ را به اندیس‌های جدید $i'$ و $j'$ مطابق دو مورد بالا منتقل می‌کنیم. سپس عبارات $\delta \overrightarrow{h} (\pi[p_{i'}, p_j])$، $\delta \overrightarrow{h} (\pi[p_i, p_{j'}])$ و $\delta \overrightarrow{h} (\pi[p_{i'}, p_{j'}])$ را محاسبه می‌کنیم و هر کدام که مقدارش مینیمم گردید را گزارش می‌کنیم. جزئیات بیشتر این مرحله در الگوریتم ۲ نشان داده شده است.

**تحلیل پیچیدگی زمانی الگوریتم ارائه شده:** اندیس $i$ و $j$ در خط اول الگوریتم ۱ با کمک ساختمان داده جستجوی نزدیک‌ترین همسایه در زمان $\mathcal{O}(\log n)$ محاسبه می‌گردد. خط ۴، ۷ و ۹ را می‌توان در بدترین حالت در زمان $\mathcal{O}(n)$ محاسبه کرد (در بدترین حالت باید $n$ نقطه را بررسی کرد). بنابراین، الگوریتم ۱ به زمان $\mathcal{O}(n)$ نیاز دارد. محاسبه خط اول الگوریتم ۲ با استفاده از ساختمان داده‌های ارائه شده در زمان $\mathcal{O}(\log^2 n)$ محاسبه می‌گردد. خطوط ۸، ۱۲ ، ۱۶ و۱۷ را می‌توان در بدترین حالت در زمان $\mathcal{O}(n)$ محاسبه کرد. بنابراین، الگوریتم ۲ به زمان $\mathcal{O}(n)$ نیاز دارد[۱۲]. بنابراین الگوریتم ارائه شده در مقاله در زمان $\mathcal{O}(n)$ زیرمسیر را گزارش می‌کند. میانگین زمان صرف شده برای اجرای الگوریتم در پایگاه داده‌های استفاده شده در این مقاله در جدول ۱ آورده شده است. مقادیر جدول نشان می‌دهد که اجرای

| پایگاه داده | زمان بر حسب ثانیه |
|---|---|
| ECG۲۰۰ | ۱/۶۴۲۴ |
| CBF | ۳/۹۳۶۶ |
| GunPoint | ۶/۴۰۹۴ |
| FaceUCR | ۳/۹۷۸۷ |
| FaceAll | ۴/۱۵۲۴ |
| Plane | ۵/۳۵۰۴ |
| SwedishLeaf | ۴/۰۱۸۸ |
| SIGSPATAL | ۳/۹۱۲۵ |

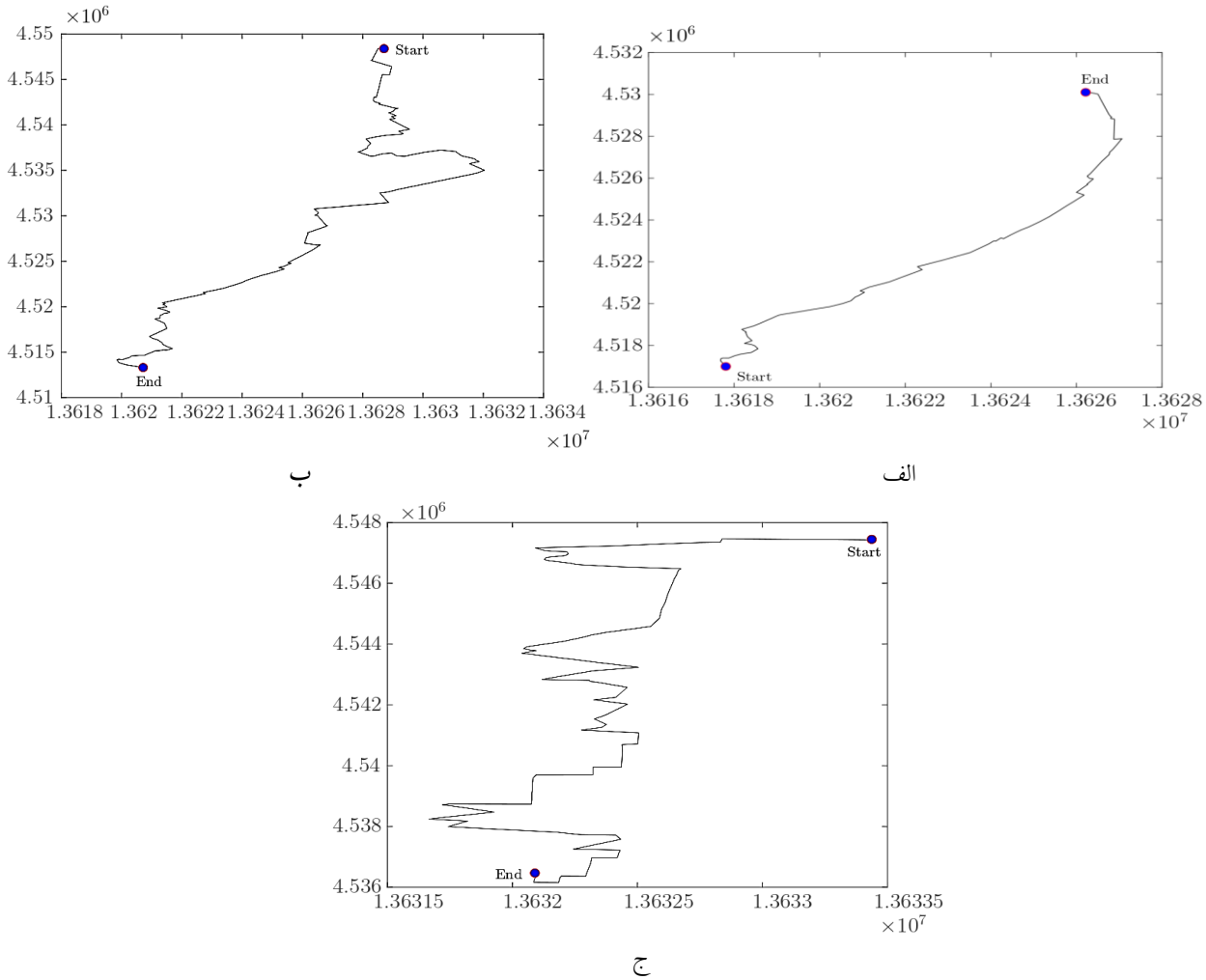
**جدول ۱. زمان صرف شده برای اجرای الگوریتم**

**Algorithm ۲ Reporting The Subtrajectory**

Input: The subtrajectory [$p_i$, $p_j$] that is determined in Algorithm ۱
Output: Report the optimal subtrajectory

```
۱:  m = max{ ||p_i − p||, ||p_j − q||,
      δ→_h(π[p_i,p_j], pq), B(π[p_i,p_j], p.y)}
۲:  if (m == ||p_i − p|| or m == ||p_j − q||) then
۳:    return π[p_i, p_j]
٤:  else
٥:    if (m == δ→_h(π[p_i,p_j])) then
٦:      k = the index of vertex p_k between p_i and p_j that
          determines the Hausdroff distance.
۷:      if (p_k.x < p.x) then
۸:        i = min_{k≤k'≤j}{||p_k' − p||}
۹:        return π[p_i, p_j]
۱۰:     else
۱۱:       if (p_k.x < q.x) then
۱۲:         j = min_{i<k'<k}{||p_k' − q||}
۱۳:         return π[p_i, p_j]
۱٤:       else
۱٥:         if (p.x ≤ p_k.x ≤ q.x) then
۱٦:           i' = min_{k≤k'≤j}{||p_k' − p||}
۱۷:           j' = min_{i<k'<k}{||p_k' − q||}
۱۸:           H = min{δ→_h(π[p_i', p_j]), δ→_h(π[p_i, p_j']),
                δ→_h(π[p_i', p_j'])}
۱۹:           if (H == δ→_h(π[p_i',p_j])) then
۲۰:             return π[p_i', p_j]
۲۱:           else
۲۲:             if (H == δ→_h(π[p_i,p_j'])) then
۲۳:               return π[p_i, p_j']
۲٤:             else
۲٥:               return π[p_i, p_j']
۲٦:             end if
۲۷:           end if
۲۸:         end if
۲۹:       end if
۳۰:     end if
۳۱:   end if
۳۲: end if
```

الگوریتم بر روی هر کدام از مسیرهای جهت‌دار موجود در پایگاه داه به صورت میانگین چند ثانیه طول می‌کشد.




**شکل ۳. (الف): مسیر جهت‌دار روبه‌جلو (ب): مسیر جهت‌دار روبه‌عقب. (ج): مسیر جهت‌دار خنثی**

## ۵. نتایج تجربی

در این قسمت نتایج تجربی بر روی الگوریتم ارائه شده در قسمت قبل ارائه و کیفیت الگوریتم و زمان آن گزارش می‌شود. مسائل پرس‌وجوی فرشه مسائل جدیدی هستند. نتایج تئوری و تجربی زیادی در حوزه پرس‌وجوی فرشه وجود ندارد. به طور خاص، مسئله بررسی شده در این مقاله نه از نظر تئوری و نه از نظر تجربی تا-کنون بررسی نشده است و الگوریتم ارائه شده در بحش قبلی، اولین الگوریتم ابتکاری برای این مسئله است. با توجه به عدم وجود الگوریتم‌های دیگری برای حل مساله، امکان مقایسه بین الگوریتم‌های مختلف به صورت تجربی وجود ندارد. در ادامه، کیفیت این الگوریتم با اجرای آن روی چند پایگاه داده بررسی می‌شود.

پنج پایگاه داده از مجموعه UCR که شامل ۸۵ پایگاه داده از مسیرهای جهت‌دار یک بعدی است را برای اجرای الگوریتم انتخاب کرده‌ایم[۱۹]. هم‌چنین از پایگاه داده که مربوط به سفرهای جاده‌ای در سانفرانسیسکو می‌باشد، استفاده کرده‌ایم. این پایگاه داده، پایگاه داده‌ای است که در چالش سال ۲۰۱۷ مربوط به SIGSPATIAL استفاده شده است. این پایگاه داده از ۲۰۰۰۰ مسیر جهت‌دار تشکیل شده است که تعداد رئوس مسیرها بین ۱۱ تا ۷۶۹ می‌باشد. این پایگاه داده دارای یک ویژگی خوب است، طول یال‌های مسیرهای جهت‌دار موجود در پایگاه داده کوچک است و این ویژگی باعث می‌شود که الگوریتم قادر باشد در اکثر موارد پاسخ بهینه را پیدا کند. پرس‌وجو که یک پاره‌خط افقی است به صورت تصادفی تولید شده است. برای هر مسیر جهت‌دار ده پاره‌خط افقی به صورت تصادفی تولید شده است (دو نقطه برای مولفه دو سر

پاره‌خط و یک نقطه برای مولفه $y$ به صورت تصادفی تولید کرده‌ایم) و الگوریتم بر روی آن اجرا شده است.

الگوریتم ارائه شده در نرم‌افزار متلب بر روی سیستم عامل ویندوز ۱۰ پیاده‌سازی شده است. الگوریتم بر روی لپ‌تاپ لنوو با پردازنده مرکزی اینتل مدل core-i۷ ۴۵۱۰ با سرعت ۲ گیگاهرتز و RAM هشت گیگابایت اجرا شده است.

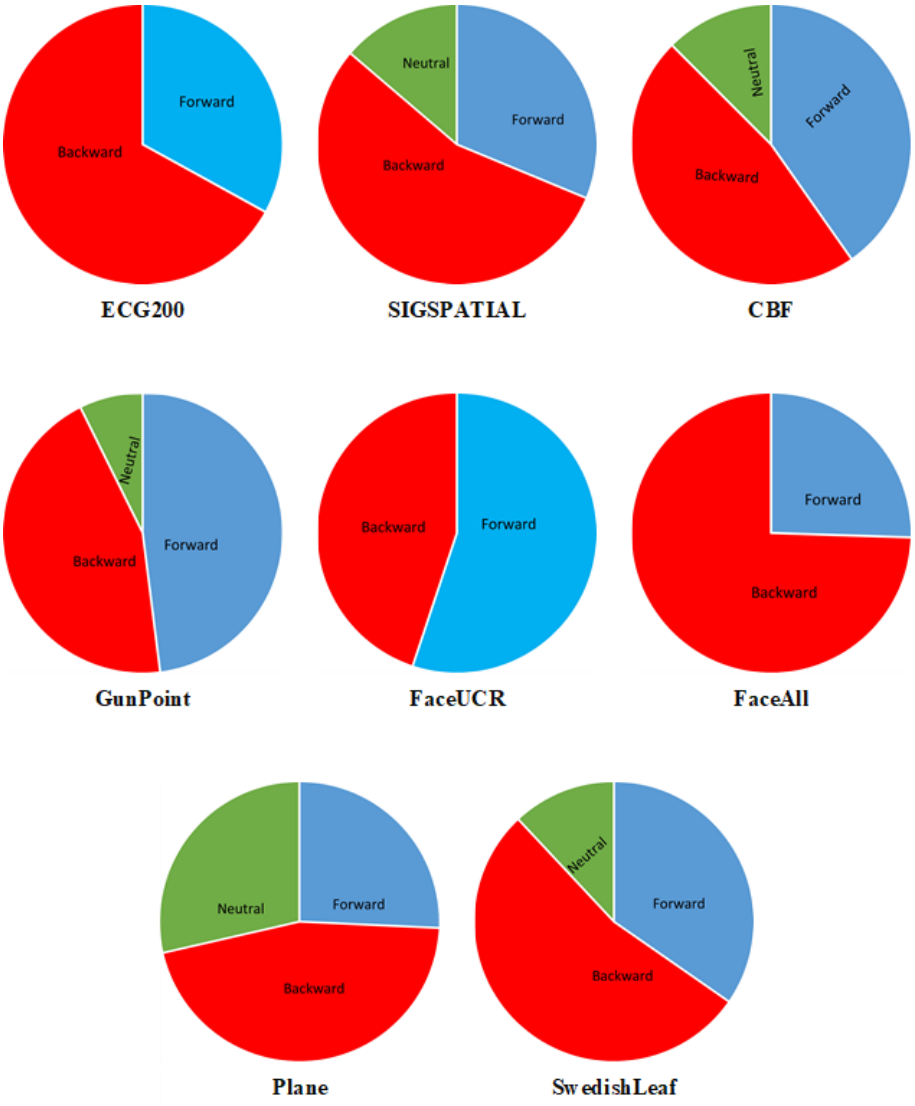
در پایگاه داده‌های استفاده شده با توجه به جهت پاره‌خط افقی که از سمت چپ به رأست می‌باشد، سه نوع مسیر جهت‌دار وجود دارد. مسیر روبه‌جلو مسیری است که شی متحرک در بیشتر مواقع رو به جلو حرکت می‌کند به این معنی که تعداد یال‌های رو به جلو بیشتر از نصف کل یال‌ها است و نقطه شروع حرکت سمت چپ نقطه انتها قرار دارد. نوع بعدی، مسیر جهت‌دار روبه‌عقب می‌باشد. در این نوع مسیر جهت‌دار، شی متحرک در بیشتر مواقع رو به عقب حرکت می‌کند به این معنی که تعداد یال‌های برگشتی بیشتر از نصف کل یال‌ها است و نقطه شروع حرکت سمت راست نقطه انتها قرار دارد. نوع آخر مسیر جهت‌داری است که شی متحرک تقریباً به صورت مساوی به جلو و عقب حرکت کرده است. این نوع را مسیر جهت‌دار خنثی می‌نامیم. شکل ۳ انواع مختلف مسیر جهت‌دار را نشان می‌دهد.

شکل ۴ توزیع مسیرهای جهت‌دار مختلف در پایگاه داده‌های استفاده شده را نشان می‌دهد. همان طور که در شکل ۴ مشخص شده است بیشترین قسمت از هر پایگاه داده شامل مسیرهای جهت‌دار روبه‌عقب است. در پایگاه داده FaceAll، ECG۲۰۰ و FaceUCR مسیر خنثی وجود ندارد و فقط شامل مسیرهای روبه‌جلو و روبه‌عقب می‌باشد. در ادامه الگوریتم را بر روی انواع مختلف مسیرهای جهت‌دار در پایگاه داده‌های ذکر شده اجرا کرده و کیفیت الگوریتم را گزارش می‌کنیم. شکل ۵ میزان درستی الگوریتم در پایگاه داده‌های استفاده شده را نشان می‌دهد. در ادامه نتیجه چند پایگاه داده را تحلیل خواهیم کرد.

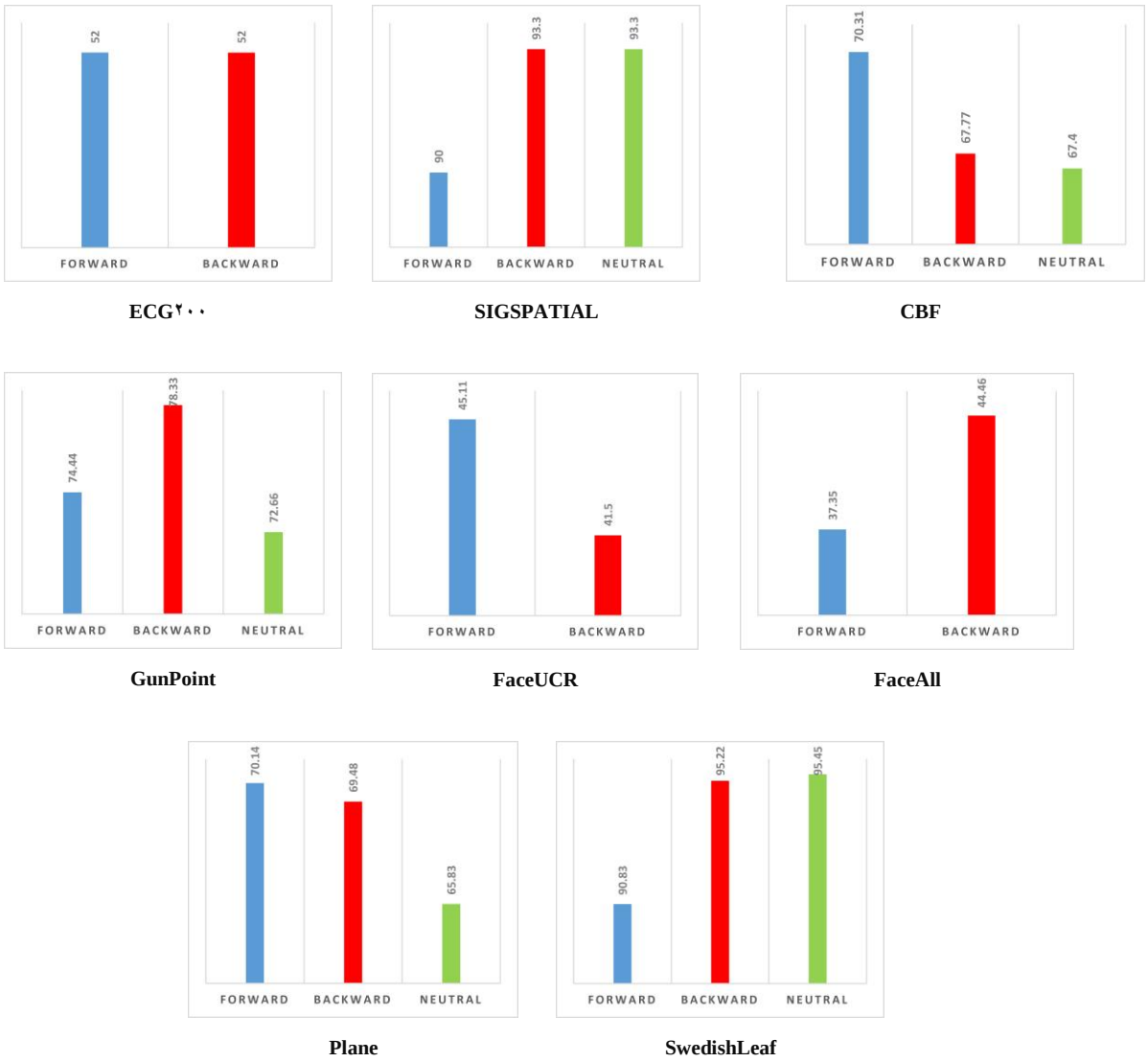
**تحلیل پایگاه داده SIGSPATIAL**: نتایج نشان می‌دهد که کیفیت الگوریتم برای مسیرهای جهت‌دار روبه‌جلو در این پایگاه داده ٪۹۰ است. در واقع الگوریتم ارائه شده فقط در ٪۱۰ پرس‌وجوها برای مسیرهای جهت‌دار روبه‌جلو قادر نیست که جواب بهینه را گزارش کند. الگوریتم اندیس $i$ یا $j$ را با کمک دیسک $B_i$ یا $B_j$ در مرحله اول الگوریتم برای اکثر مسیرهای جهت‌دار روبه‌جلو تغییر می‌دهد. همچنین در مرحله دوم الگوریتم ماکزیمم عبارات $disti$ یا $distj$ فاصله فرشه را تعیین می‌کند. تنها در ٪۱۰ پرس‌وجوها عبارت مربوط به فاصله هاسدورف، فاصله فرشه را تعیین می‌کند و الگوریتم ارائه شده علیرغم اصلاحات صورت گرفته قادر نیست که جواب بهینه را پیدا کند. کیفیت الگوریتم برای مسیرهای جهت‌دار روبه‌عقب و مسیرهای جهت‌دار خنثی در پایگاه داده ٪۹۳/۳ است. الگوریتم قادر نیست که در مرحله اول برای اکثر مسیرهای جهت‌دار روبه‌عقب، اندیس $i$ یا $j$ را با کمک دیسک $B_i$ یا $B_j$ تغییر بدهد. در واقع الگوریتم زیرمسیری را گزارش می‌کند که نقطه شروع و انتهای آن یکسان است. در ٪۶/۷ پرس‌وجوها عبارت مربوط به فاصله هاسدورف، فاصله فرشه را تعیین می‌کند و الگوریتم ارائه شده نمی‌تواند زیرمسیر بهینه را پیدا کند. در واقع اصلاحات انجام شده در فاصله هاسدورف در الگوریتم ۲ همواره به درستی عمل نمی‌کند. اگر جهت پاره‌خط افقی را تغییر بدهیم، یعنی پاره‌خط افقی از سمت رأست به چپ باشد، مسیر جهت‌دار رو به عقب تبدیل به مسیر جهت‌دار رو به جلو می‌شود و نتایج هم تغییر خواهد کرد.

کیفیت الگوریتم ارائه شده برای پایگاه داده ٪۹۸/۸ است و تنها در یک درصد از پرس‌وجوها زیرمسیر بهینه گزارش نمی‌گردد. تنها علت اصلی برای خطای الگوریتم اصلاحاتی است که در الگوریتم۲ در قسمت فاصله هاسدورف صورت گرفته است. برای حذف کردن یک درصد خطا باید اصلاحات این قسمت را بهبود داد.

**تحلیل پایگاه داده GunPoint**: نتایج نشان می‌دهد که کیفیت الگوریتم برای مسیرهای جهت‌دار روبه‌جلو در این پایگاه داده ٪۹۰/۳۸ است. در واقع الگوریتم ارائه شده تقریبا در ٪۱۰ پرس‌وجوها برای مسیرهای جهت‌دار روبه‌جلو قادر نیست که جواب بهینه را گزارش کند. کیفیت الگوریتم برای مسیرهای جهت‌دار روبه‌عقب و مسیرهای جهت‌دار خنثی در پایگاه داده تقریبا ٪۹۵ است.




**شکل ۴. توزیع مسیرهای جهت‌دار در پایگاه داده‌های استفاده شده**




**ECG۲۰۰** **SIGSPATIAL** **CBF**

**GunPoint** **FaceUCR** **FaceAll**

**Plane** **SwedishLeaf**

**شکل ۵. درستی الگوریتم ارائه شده برای مسیرهای جهت‌دار مختلف در پایگاه داده‌های استفاده شده**

نکات زیر در مورد اجرای الگوریتم ارائه شده بر روی پایگاه داده‌های ذکر شده بدست آمده است:

• اگر جهت پاره‌خط افقی را تغییر بدهیم، یعنی پاره‌خط افقی از سمت رأست به چپ باشد، مسیر جهت‌دار رو به عقب تبدیل به مسیر جهت‌دار رو به جلو می‌شود و نتایج هم تغییر خواهد کرد.

• دلیل اصلی برای پایین بودن خطای الگوریتم ارائه شده این است که طول یال‌های مسیرهای جهت‌دار موجود در پایگاه داده‌های مورد بررسی کوچک است. بنابراین اصلاحات صورت گرفته در مرحله اول و دوم الگوریتم ارائه شده به خوبی عمل می‌کند و الگوریتم قادر است در اکثر موارد پاسخ بهینه را پیدا کند.

• هر چقدر ماکزیمم طول یال در پایگاه داده بیشتر باشد، میزان خطای الگوریتم بالا می‌رود.

جدول ۱ میزان اختلاف میان پاسخ الگوریتم ارائه شده و جواب بهینه، برای پایگاه داده‌های استفاده شده را نشان می‌دهد. همان‌طور که مشخص است، بیشترین اختلاف مربوط به پایگاه داده FaceAll است.

**جدول ۲. میزان اختلاف با مقدار بهینه**

| پایگاه داده | Forward | Backward | Neutral |
|---|---|---|---|
| **ECG۲۰۰** | ۰/۳۱ | ۰/۳۳ | |
| **CBF** | ۰/۴۸ | ۰/۵۱ | ۰/۵۶ |
| **GunPoint** | ۰/۱۶ | ۰/۰۸ | ۰/۰۶ |
| **FaceUCR** | ۱/۱۳ | ۱/۲ | |
| **FaceAll** | ۱/۳۱ | ۱/۱۵ | |
| **Plane** | ۰/۳۹ | ۰/۳۷ | ۰/۵ |
| **SwedishLeaf** | ۰/۵۱ | ۰/۵۳ | ۰/۶ |
| **SIGSPATAL** | ۰/۲۲ | ۱/۵۳ | ۰/۴۳ |

## ۶. نتیجه‌گیری

در این مقاله مسئله تشابه در حوزه فاصله فرشه به صورت تجربی مورد بررسی قرار گرفت. به این صورت که مسیر جهت‌دار $\pi$ به عنوان ورودی داده شده است. در زمان پرس‌وجو پاره‌خط افقی $Q$ توسط کاربر مشخص می‌شود. الگوریتم ابتکاری ارائه شده زیرمسیری از مسیر جهت‌دار را گزارش می‌کند به صورتی که فاصله فرشه میان زیرمسیر گزارش‌شده و پاره‌خط افقی $Q$ بین تمام زیرمسیرهای ممکن مینیمم می‌باشد. کیفیت الگوریتم ارائه شده بر روی چند پایگاه داده بررسی و گزارش شد. مقاله را با ارائه چند مسئله باز به پایان می‌رسانیم:

۱. چگونه می‌توان این مسئله را برای حالتی که پاره‌خط غیرافقی باشد، حل کرد؟

۲. چگونه می‌توان الگوریتمی ارائه کرد که بتواند مسئله را برای هر نوع پرس‌وجو حل کند؟

۳. آیا می‌توان الگوریتم را به صورتی تغییر داد که خطا نداشته باشد و همواره پاسخ بهینه را تولید کند؟

## مراجع


[۱] Felix Hausdorff. *Grundzuge der mengenlehre*, ۱۹۴۹

[۲] Helmut Alt. The computational geometry of comparing shapes. In *Efficient Algorithms*, volume ۵۷۶۰ of Lecture Notes in Computer Science, pages ۲۳۵–۲۴۸, ۲۰۰۹.

[۳] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. In *Computer graphics forum*, pages ۱۶۷–۱۷۴, ۱۹۹۸

[۴] Maurice Fréchet. Les ensembles abstraits et le calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, pages ۱-۲۶, ۱۹۱۰.

[۵] Helmut Alt and Michael Godau. Measuring the resemblance of polygonal curves. In *Proceedings of the eighth annual symposium on Computational geometry*, pages ۱۰۲–۱۰۹, ۱۹۹۲.

[۶] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, pages۷۵-۹۱, ۱۹۹۵.

[۷] E Sriraghavendra, K Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Ninth International Conference on Document Analysis and Recognition*, pages ۴۶۱-۴۶۵, ۲۰۰۷.

[۸] Mark de Berg, Atlas F Cook, and Joachim Gudmundsson. Fast Fréchet queries. *Computational Geometry*, pages ۷۴۷–۷۵۵, ۲۰۱۳

[۹] Joachim Gudmundsson and Michiel Smid. Fréchet queries in geometric trees. In *European Symposium on Algorithms*, pages ۵۶۵–۵۷۶. Springer, ۲۰۱۳

[۱۰] Anne Driemel and Sariel Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, pages ۱۸۳۰-۱۸۶۶, ۲۰۱۳

[۱۱] Mark de Berg and Ali D Mehrabi. Straight-path queries in trajectory data. *Journal of Discrete Algorithms*, pages ۲۷–۳۸, ۲۰۱۶.

[۱۲] Mark de Berg, Ali D Mehrabi, and Tim Ophelders. Data structures for Fréchet queries in trajectory data. In *Proceedings of the ۲۹th Canadian Conference on Computational Geometry (CCCG ۲۰۱۷)*, pages ۲۱۴-۲۱۹, ۲۰۱۷.

[۱۳] Maike Buchin, Ivor van der Hoog, Tim Ophelders, Rodrigo I Silveira, Lena Schlipf, and Frank Staals. Improved space bounds for Fréchet distance queries. In Proceeding of the ۳٦th European Workshop on Computational Geometry, pages ۱-۷, ۲۰۲۰.

[۱٤] Joachim Gudmundsson, Majid Mirzanezhad, Ali Mohades, and Carola Wenk. *International Journal of Computational Geometry & Applications*, pages ۱٦۱-۱۸۷, ۲۰۱۹

[۱٥] Mark de Berg, Joachim Gudmundsson, and Ali D Mehrabi. A dynamic data structure for approximate proximity queries in trajectory data. In *Proceedings of the ۲٥th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages ۱-٤،۲۰۱۷

[۱٦] Julian Baldus and Karl Bringmann. A fast implementation of near neighbors queries for Fréchet distance (GIS cup). In *Proceedings of the ۲٥th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages ۱-٤, ۲۰۱۷.

[۱۷] Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. Fresh: Fréchet similarity with hashing. In *Workshop on Algorithms and Data Structures*, pages ۲٥٤–۲٦۸, ۲۰۱۹.

[۱۸] Fabian Dütsch and Jan Vahrenhold. A filter-and-refinement-algorithm for range queries based on the Fréchet distance (gis cup). In *Proceedings of the ۲٥th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages ۱–٤, ۲۰۱۷

[۱۹] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR time series classification archive, July ۲۰۱٥.http://www.cs.ucr.edu/eamonn/time_series_data.

[۲۰] Joachim Gudmundsson and Michiel Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Computational Geometry*, pages ٤۷۹-٤۹٤, ۲۰۱٥

[۲۱] Bahram Sadeghi Bigham, and Samaneh Mazaheri. Survey on Metrics for Shape Matching Based on Similarity, Scaling and Spatial Distance. In The ۷th International Conference on Contemporary Issues in Data Science, pages ۱۳-۲۳, ۲۰۱۷