دهمین
کنفرانس سالانه انجمن کامپیوتر ایران
۲۷ تا ۲۹ بهمن ۱۳۸۳

کوتاهترین مسیر پیوندی در یک چندضلعی ساده با قید قابلیت دید

محمد رضا ضرابی محمد قدسی

دانشکده‌ی مهندسی کامپیوتر
دانشگاه صنعتی شریف

چکیده

در این مقاله الگوریتمی ارائه می‌کنیم که در زمان و حافظه‌ی $O(n)$ کوتاهترین مسیر پیوندی بین دو نقطه‌ی دلخواه s و t در n ضلعی ساده $\mathcal{P}$ را با درنظر گرفتن شرط دیدن نقطه‌ی دلخواه q بدست آورد. با این شرط که حداقل یک نقطه روی مسیر جواب وجود داشته باشد که از نقطه‌ی q قابل رؤیت باشد. همچنین در این مقاله این مسأله را در حالت پرس‌وجو مورد بررسی قرار می‌دهیم، به این صورت که پیش‌پردازش لازم بر روی $\mathcal{P}$ را طوری انجام می‌دهیم که با دریافت هر سه نقطه‌ی q و s و t در زمان $O(\log n)$ کوتاهترین مسیر پیوندی از s به t، که q را ببیند به‌دست آورد.

کلمات کلیدی: قابلیت دید ، فاصله‌ی پیوندی ، مسیر پیوندی

۱ مقدمه

کوتاهترین فاصله‌ی پیوندی[۱] در یک چندضلعی عبارت است از حداقل تعداد شکست‌ها در طول مسیر بین دو نقطه‌ی s و t. کوتاهترین مسیر پیوندی[۲] مسیری است که s و t را به هم متصل می کند و کم‌ترین تعداد شکست‌ها را دارد. بر خلاف حالت اقلیدسی چنین مسیرهایی منحصر به فرد نیستند. پیدا کردن کوتاهترین مسیر پیوندی بین دو نقطه در یک چندضلعی، یکی از مسائل پایه‌ای در هندسه‌ی محاسباتی می‌باشد که کاربردهای متفاوتی در زمینه‌های روباتیک، برنامه‌ریزی حرکت[۳] و ارتباطات موج کوتاه[۴] دارد. مثلاً، ربات نقطه‌ای R را درنظر بگیرید که باید در فضای یک چندضلعی ساده از نقطه‌ای به نقطه‌ی دیگر جابه‌جا شود. در چنین رباتی ممکن است حرکت مستقیم نسبت به حرکت چرخشی هزینه‌ی ناچیزی داشته باشد. بنابراین برای حداقل شدن هزینه باید تعداد چرخش‌ها در

[۱] link distance
[۲] link path
[۳] motion planning
[۴] microwave communication

کل مسیر کمینه گردد. ارتباط مستقیم درسیستم‌های ارتباطی نیز زمانی امکان‌پذیر است که فرستنده و گیرنده به صورت متقابل ازهم قابل رؤیت باشند، در غیراینصورت به ابزاری چون تکرارکننده‌ها[5] نیاز داریم که به نوبه خود هزینه خواهد داشت. بنابراین کم کردن تعداد تکرارکننده‌ها (یا همان چرخش‌ها) در طی مسیر موجب کاهش کل هزینه خواهد شد. معیار قابلیت دید از جمله محدودیت‌هایی است که در گذشته کمتر مورد بررسی قرار گرفته است. در این نوع محدودیت کل یا قسمتی از مسیر باید از یک نقطه‌ی واحد قابل رؤیت باشد. این نوع محدودیت‌ها در سیستم‌های ارتباطی یا نگهبانی که در آن قابلیت دید مستقیم مورد نیاز است کاربرد دارد.

۲ تعریف مسأله

n ضلعی $\mathcal{P}$ و نقاط q و s و t در داخل آن داده شده‌اند. می‌خواهیم کوتاهترین مسیر پیوندی از s به t را بدست آوریم که حداقل یک نقطه از مسیر، از q قابل رؤیت باشد. این مسأله را در حالتی که q و s و t در $\mathcal{P}$ به صورت ثابت باشند بررسی می‌کنیم. سپس به حالت پرس‌وجو روی نقاط q و s و t می‌پردازیم. تعاریف زیر را در نظر می گیریم :

$d_L(s,t)$: کم‌ترین مقدار فاصله‌ی پیوندی از s به t در $\mathcal{P}$،

$\pi_L(s,t)$: کوتاهترین مسیر پیوندی از s به t در $\mathcal{P}$،

$\pi_E(s,t)$: کوتاهترین مسیر اقلیدسی از s به t در $\mathcal{P}$،

$d_L(s,t,e)$، $\pi_L(s,t,e)$، $\pi_E(s,t,e)$: همان پارامترهای فوق، با محدودیت ملاقات با پارخط e، منظور از ملاقات، تقاطع مسیر با حداقل یک نقطه از e می‌باشد.

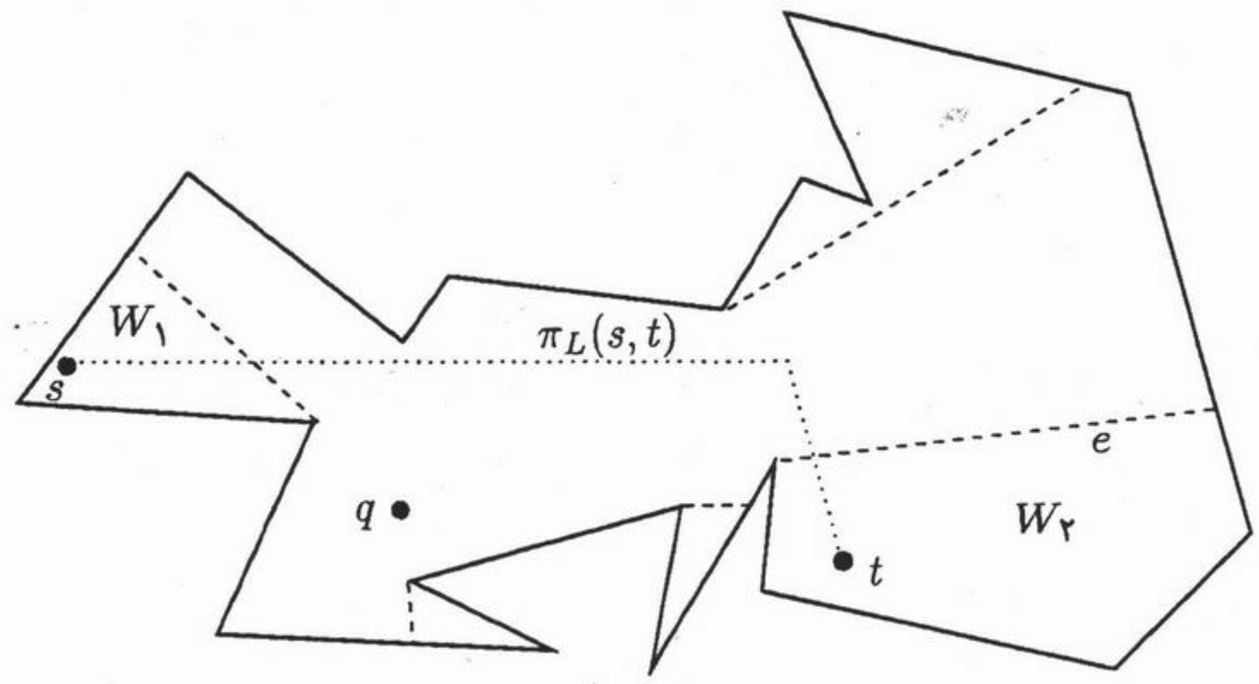
$V(q)$: چندضلعی رؤیت نقطه‌ی q در $\mathcal{P}$،

Pocket(q): بخش‌هایی از چندضلعی $\mathcal{P}$ که از نقطه‌ی q قابل رؤیت نیستند،

$W(x)$: افراز پنجره‌ای[6] نقطه‌ی x در $\mathcal{P}$، که هر افراز را یک سلول[7] می‌نامیم.

$W_t(x)$: گراف دوگان $W(x)$، که یک درخت می‌باشد.

[5] repeaters
[6] window partition
[7] cell



شکل ۱: چندضلعی‌های $V(q)$ و $\text{Pocket}(q)$

۳ نتایج قبلی

مرجع [۱] به‌عنوان مهمترین کار انجام شده در این زمینه است که فاصله‌ی پیوندی را در زمان بهینه‌ی $O(n)$ بدست می‌آورد. شیوه‌ی کار این الگوریتم بر پایه‌ی مثلث بندی چندضلعی $\mathcal{P}$ است (طبق [۸]). اساس کار الگوریتم فاصله‌ی پیوندی بر پایه‌ی افراز چندضلعی $\mathcal{P}$ به سلول‌هایی است که فاصله‌ی پیوندی آنها نسبت به نقطه‌ی دلخواه x مقداری ثابت است ($W(x)$). یک الگوریتم کورکورانه برای بدست آوردن $W(x)$ استفاده از الگوریتم [۵] است که در بدترین حالت دارای زمان $O(n^2)$ می‌باشد. ولی Suri در [۱] نشان داده است که این زمان را می‌توان به تعداد مثلث‌هایی که با هر سلول اشتراک دارند تقلیل داد. وی همچنین نشان داده است که هر مثلث حداکثر با سه سلول اشتراک خواهد داشت. پس در مجموع پیچیدگی الگوریتم به تعداد مثلث‌ها در $\mathcal{P}$ خواهد بود، که $O(n)$ است. در [۲] Suri الگوریتم پرس‌وجوی مسأله را به ازای نقطه‌ی ثابت s و نقطه‌ی دلخواه t در $O(\log n)$ بدست می‌آورد. در این الگوریتم ابتدا $W(s)$ و سپس $W_t(s)$ ساخته می‌شود. کل زمان پیش‌پردازش در این حالت (با توجه به [۱]) $O(n)$ می‌باشد. بعد با توجه به [۳] ناحیه‌ای که t در آن قرار گرفته است در $O(\log n)$ محاسبه می شود و از روی $W_t(s)$ فاصله‌ی پیوندی محاسبه می‌شود. همچنین در این مقاله یک الگوریتم تقریبی با خطای ± ۲ برای دو نقطه‌ی پرس‌وجوی s و t بدست آمده است. در [۱۰] Mitchell الگوریتم دقیق پرس‌وجوی دو نقطه‌ایی را در زمان و حافظه‌ی پیش‌پردازش $O(n^3)$ و زمان پرس‌وجوی $O(\log n)$ بدست آورده است. Mitchell مسأله را برای دو خط و دو چندضلعی محدب با همان زمان و حافظه‌ی پیش‌پردازش و زمان پرس‌وجوی $O(\log n)$ و $O(\log k * \log n)$ که k تعداد اضلاع چندضلعی محدب است بدست آورده است. با توجه به زیاد بودن زمان‌پیش‌پردازش وی دو الگوریتم تقریبی به ترتیب با خطاهای ± ۱ و ± ۲ و زمان پیش‌پردازش $O(n^2)$ و $O(n)$ ارائه کرده است. دراین مقاله تمام کارهای ذکر شده دربالا را بااضافه کردن محدودیت قابلیت دید درطول مسیر پیوندی بررسی می کنیم و الگوریتم مورد نظر را روی حالت عادی و پرس‌وجو ارائه می کنیم.

۴ کلیت الگوریتم

اساس الگوریتم برپایه‌ی روشی است که در [۴] ارائه شده است (شکل ۱). در این الگوریتم که برای حالت اقلیدسی بررسی شده است ابتدا $V(q)$ و $\text{Pocket}(q)$ محاسبه می‌شود. سپس موقعیت s و t نسبت به این Pocket ها

بدست می‌آید. اگر s و t در دو Pocket مختلف قرار داشته باشند یا یکی از آنها در $V(q)$ یا مرز آن باشد هر مسیر دلخواه بین دو نقطه‌ی s و t با $V(q)$ اشتراک خواهد داشت. ولی در حالتیکه s و t در یک Pocket باشند مسأله به اشتراک با یک ضلع از $V(q)$ به نام e کاهش می‌یابد. این حالت را می توان برای $\pi_L(s,t)$ نیز تعمیم داد. یعنی $\pi_L(s,t)$ را به گونه‌ای بدست می‌آوریم که اشتراک آن با $V(q)$ غیرتهی باشد.

۵ محاسبه چندضلعی رؤیت

برای محاسبه‌ی $V(q)$ طبق الگوریتمی که در[۶] و [۷] آمده است عمل می‌کنیم که پیچیدگی کل آن $O(n)$ خواهد بود. قبل از این در [۱۱] الگوریتمی ارائه شده بود که با همین پیچیدگی $V(q)$ را بدست می‌آورد، ولی درعمل این الگوریتم از سه پشته استفاده می‌کند در حالیکه الگوریتم اول فقط از یک پشته استفاده می‌کند و از نظر پیاده سازی هم ساده‌تر است. درحالت پرس‌وجو روی q طبق [۱۲]، با زمان پیش‌پردازش $O(n^۳)$ و زمان پرس‌وجوی $O(\log n + |V(q)|)$ می‌توان عمل کرد (که این زمان بهینه می‌باشد). در بدترین حالت $V(q)$ دارای پیچیدگی $O(n)$ خواهد بود و کل زمان محاسبه را به $O(n)$ خواهد رساند که با توجه به صرف زمان پیش‌پردازش $O(n^۳)$ مطلوب ما نمی‌باشد. ولی همانطور که بعداً خواهیم دید، در این حالت نیازی به محاسبه‌ی کل $V(q)$ نخواهیم داشت .

۶ الگوریتم نقطه ثابت

در این حالت نقاط q و s و t به‌صورت ثابت در $\mathcal{P}$ داده شده‌اند. تنها تفاوت $\pi_L(s,t,e)$ با $\pi_E(s,t,e)$ در مرحله‌ی آخر می‌باشد. این تفاوت در دو قسمت خلاصه می‌شود:

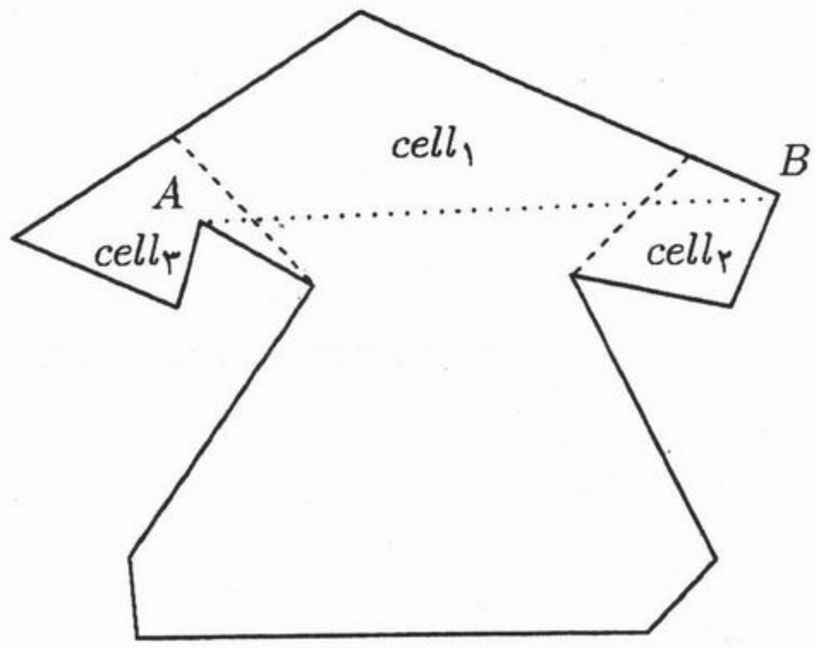
۱) $\pi_L(s,t,e)$ ممکن است از e عبور کرده و وارد $V(q)$ شود. در حالیکه $\pi_E(s,t,e)$ فقط با e تقاطع خواهد داشت (با زدن یک میان بُر می‌توان از ورود مسیر به $V(q)$ جلوگیری کرد).

۲) در $\pi_E(s,t,e)$ ضلع e به فواصل k تایی $L_{interval}$ ها تقسیم می‌شود (طبق [۴])، در حالیکه در $\pi_L(s,t,e)$ همانطور که بعداً خواهیم دید e حداکثر به سه قسمت توسط $W(x)$ تقسیم می‌شود.

لم زیر نشان می‌دهد پیچیدگی الگوریتم با قید قابلیت دید از حالت عادی آن بیشتر نخواهد بود.

لم ۱ : فرض کنیم پاره خط دلخواه AB در درون $\mathcal{P}$ داده شده است. دراین صورت AB مطابق شکل ۲ حداکثر با سه سلول از پنجره‌ها اشتراک خواهد داشت.

اثبات: فرض کنیم $W(s)$ داده شده است. به سادگی دیده می‌شود که AB نمی‌تواند با دو سلول از $W(s)$ اشتراک داشته باشد به شرط آنکه اختلاف فاصله‌ی پیوندی آن دو سلول تا s بیشتر از یک باشد (چون در غیر این صورت تعریف $W(s)$ نقض می‌شود). فرض کنیم AB با چهار یا بیشتر سلول اشتراک داشته باشد. دراین صورت سه سلول متناوب با فواصل $k, k+۱, k$ نسبت به s وجود خواهد داشت ($k \geq ۱$). حال $W_t(s)$ را در نظر می‌گیریم این درخت به ازای هر سلول دارای یک نود می‌باشد و دو نود وقتی به هم متصل می‌شوند که دو سلول دارای مرز مشترک باشند. ریشه‌ی این درخت متناظر با سلولی است که s در آن قرار گرفته است و هر نود به عمق i متناظر با سلولی است که تمام نقاط آن دارای فاصله‌ی پیوندی $i+۱$ تا s می‌باشد. اگر پاره خط AB با سه نود به عمق $k-۱$ $k-۱, k, k-۱$ اشتراک داشته باشد، در این صورت بین نودهای متناظر با سلول‌های $k-۱$ یال‌هایی وجود خواهند داشت که شرط درخت



شکل ۲: اشتراک یک پاره خط با حداکثر ۳ سلول

بودن را نقض خواهد کرد. بنابراین AB حداکثر با سه سلول به عمق $k, k-1, k$ می‌تواند اشتراک داشته باشد که دوری در درخت ایجاد نخواهند کرد. □

$W(s)$ و $W(t)$ را در $O(n)$ محاسبه می‌کنیم. فرض کنید ضلعی از Pocket(q) که باید ملاقات شود، e باشد (شکل ۳). طبق لم ۱ ضلع e حداکثر با سه سلول (دو پنجره) اشتراک خواهد داشت. اگر این اشتراک‌ها را به ازای $W(s)$ و $W(t)$ روی e حساب کنیم حالت‌های زیر را خواهیم داشت :

۱) اگر هیچکدام از پنجره‌های $W(s)$ و $W(t)$ با e تلاقی نداشته باشند کل e جواب مسأله است.

۲) اگر فقط یکی از پنجره‌های $W(s)$ و $W(t)$ با e تلاقی داشته باشد از روی W_t همان پنجره ، پنجره‌ی قبلی را پیدا می‌کنیم آنگاه کلیه‌ی نقاطی از e که از این پنجره قابل رؤیت می‌باشند جواب مسأله است.

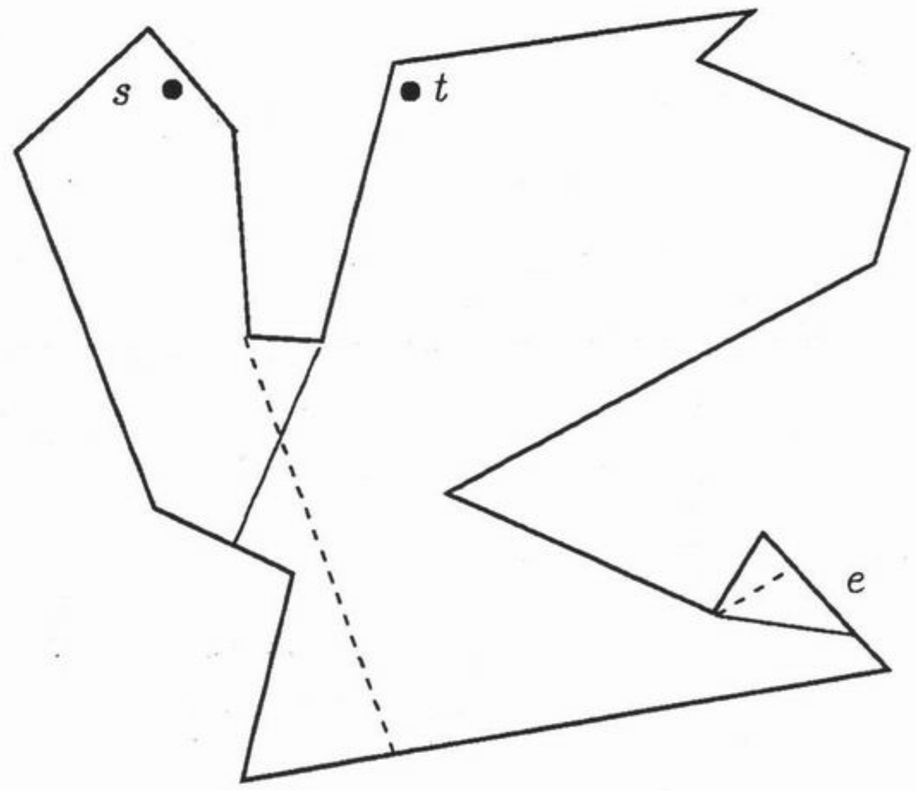
۳) اگر هر دو پنجره‌ی $W(s)$ و $W(t)$ با e تلاقی داشته باشند مانند قسمت قبل بخش‌هایی از e که از پنجره‌های قبلی s و t قابل رؤیت می باشند را بدست می‌آوریم و اشتراک آنها جواب مسأله خواهد بود. در صورت عدم اشتراک باید بررسی کنیم که امتداد پنجره‌ها در درون بخشی از $\mathcal{P}$ که $V(q)$ قرار دارد تلاقی می‌کنند یا خیر. در غیر اینصورت هر کدام از اشتراک‌ها یک جواب است.

مراحل ۱ و ۲ در زمان $O(1)$ (شکل ۳) و مرحله ۳ با پیش‌پردازش $O(n)$ در $O(\log n)$ (شکل ۴) قابل انجام است. با توجه به روش بالا $d_L(s,t,e)$ که q را در طی مسیر ملاقات کند در $O(n)$ بدست می‌آید.

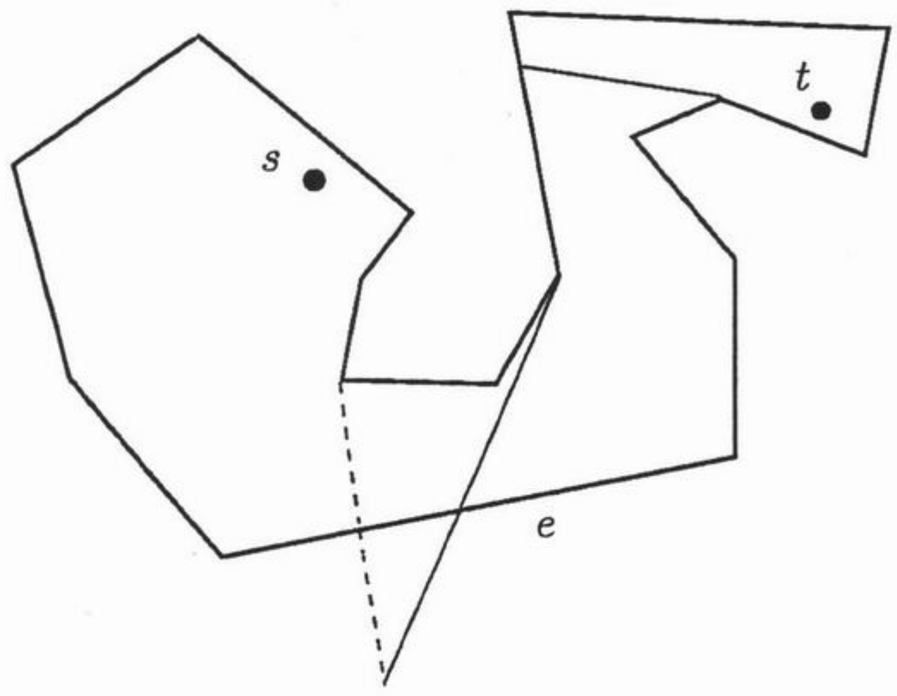
برای بدست آوردن $\pi_L(s,t)$ (شکل ۵) که شرایط رؤیت را نیز داشته باشد به صورت زیر عمل می‌کنیم : اگر W_k آخرین پنجره از s به t باشد W_k را امتداد می‌دهیم تا W_{k-1} را در V_i قطع کند و این کار را برای تمام k ها انجام می‌دهیم تا تمام V_i ها بدست آید. برای یافتن آخرین لینک از t به W_k که با اضلاع چندضلعی تلاقی نداشته باشد از لم ۲ استفاده می‌کنیم. فرض کنید x نقطه‌ای مشخص شده (از مرحله‌ی قبل) روی e باشد در این صورت , $\pi_L(t,x)$ $\pi_L(s,x)$ را مطابق بالا بدست آورده و اجتماع دو مسیر ($\pi_L(s,t,e)$) جواب مسأله خواهد بود.

لم ۲ : اگر مطابق شکل ۶، a و b دو سر پنجره‌ی W_k و $SP(t,a)$ کوتاهترین مسیر اقلیدسی از t به a باشد و tv اولین پاره خط این مسیر، در این صورت امتدا tv، ab را قطع خواهد کرد.

اثبات: اگر tv با ab متقاطع نباشد، t از ab قابل رؤیت نخواهد بود. در این صورت W_k نمی‌تواند پنجره‌ای برای مسیر s به t باشد. □



شکل ۳: مراحل ۱ و ۲



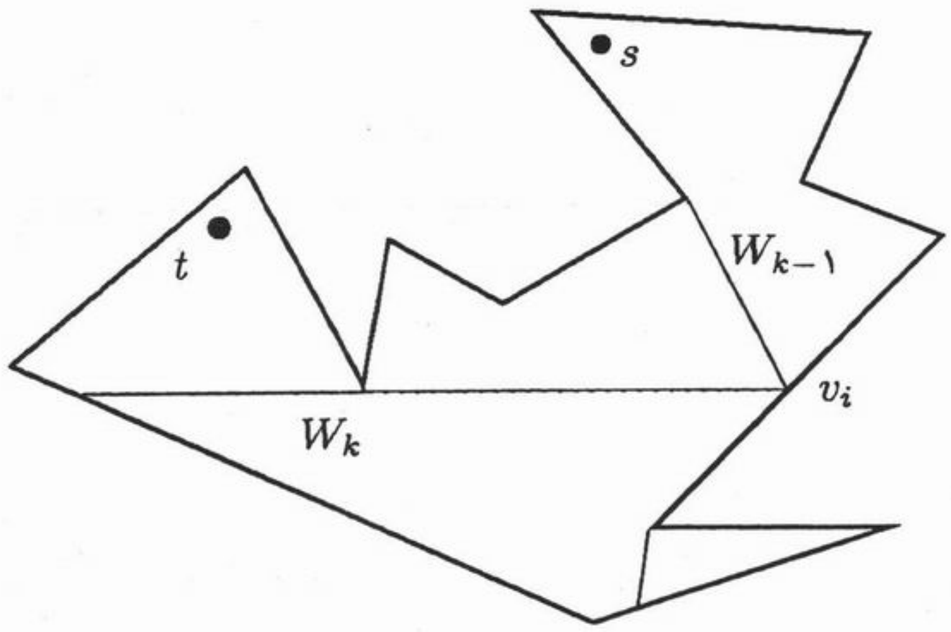
شکل ۴: مرحله‌ی ۳

برای محاسبه‌ی $SP(t,a)$ از الگوریتمی که در [۹] آمده است استفاده می‌کنیم که با زمان پیش‌پردازش $O(n)$ در $O(\log n)$ عمل می‌کند. پس کلاً $\pi_L(s,t,e)$ در $O(n+d_L(s,t,e))$ قابل محاسبه است. $d_L(s,t)$ در بدترین حالت $O(n)$ خواهد بود (شکل ۷).

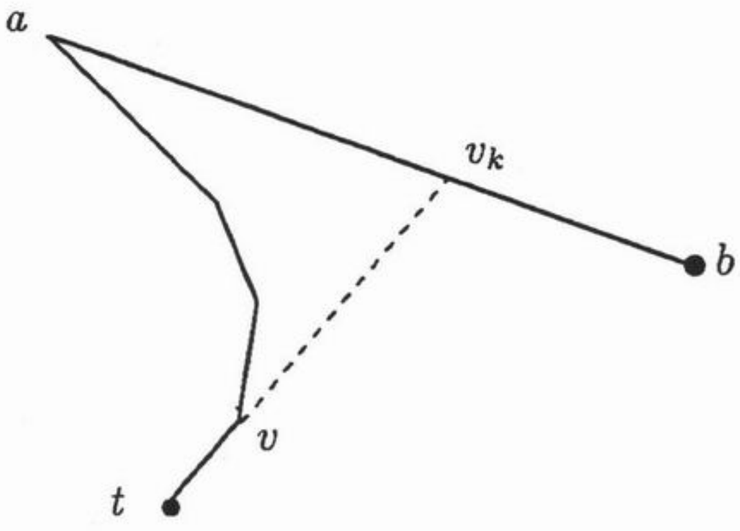
۷ الگوریتم پرس‌وجو

در این حالت پرس‌وجو را روی نقاط q و s و t اعمال می‌کنیم. $V(q)$ و Pocket(q) باجابجایی q تغییر می‌کنند. ولی در صورت لزوم می‌توان ضلع e را در هر لحظه مشخص کرد.

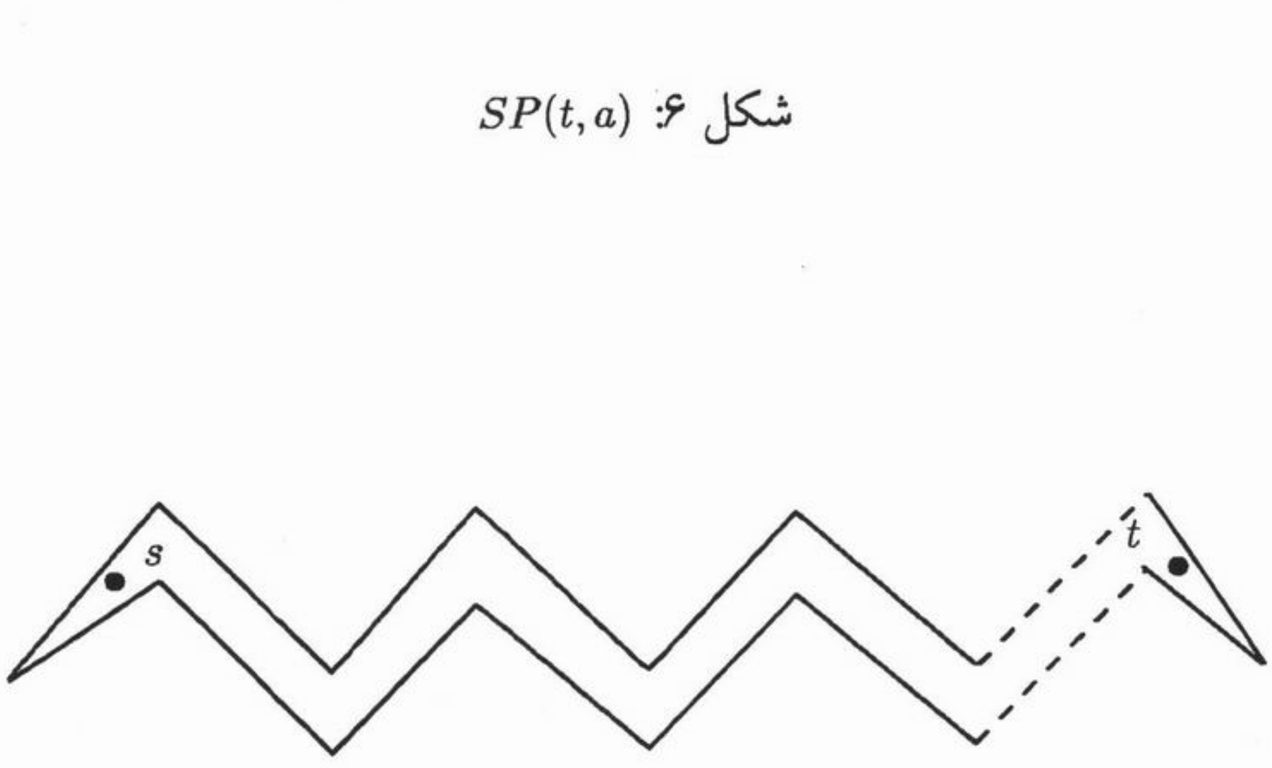
۱) $\pi_E(s,q)$ و $\pi_E(t,q)$ را با توجه به [۹] با پیش‌پردازش $O(n)$ در $O(\log n)$ بدست می‌آوریم.



شکل ۵: $\pi_L(s,t)$



شکل ۶: $SP(t,a)$



شکل ۷: چندضلعی با حداقل $n/2$ پیوند بین s و t

۲) $d_L(s,q)$ و $d_L(t,q)$ را با توجه به [۱۰] با پیش‌پردازش $O(n^۳)$ در $O(\log n)$ بدست می‌آوریم.

اگر $d_L(s,q)$ یا $d_L(t,q)$ برابر یک باشند s یا t در $V(q)$ یا مرز آن قرار خواهند داشت و $d_L(s,t)$ و $\pi_L(s,t)$ با توجه به [۱۰] با پیش‌پردازش $O(n^۳)$ در $O(\log n)$ و $O(\log n + d_L(s,t))$ بدست خواهند آمد. در غیر این صورت آخرین رؤس $\pi_E(s,q)$ و $\pi_E(t,q)$ از $\mathcal{P}$ را بدست می‌آوریم، اگر این دو رأس متفاوت باشند s و t در دو Pocket مختلف قرار گرفته‌اند و مطابق قسمت قبل عمل می‌کنیم. در غیر این صورت s و t در یک Pocket قرار گرفته‌اند و این رأس v یک سر پاره خط e خواهد بود. با توجه به [۱۰] محل تلاقی qv با $\mathcal{P}$ را با پیش‌پردازش $O(n^۲)$ در $O(\log n)$ بدست می‌آوریم. به این ترتیب e به صورت دقیق بدست می‌آید. $d_L(s,t,e)$ و $\pi_L(s,t,e)$ با توجه به قسمت‌های قبل با اشتراک گیری روی e بدست می‌آید. فقط در حالتی که u، محل تلاقی پنجره‌ها درخارج از Pocket قرار گیرد، باید بررسی شود، u در درون بخشی از $\mathcal{P}$ که $V(q)$ قرار دارد واقع می شود یا خیر. کل زمان و حافظه‌ی پیش‌پردازش و پرس‌وجو برای $d_L(s,t)$ و $\pi_L(s,t)$ به‌ترتیب $O(n^۳)$ و $O(\log n)$ و $O(\log n + d_L(s,t))$ خواهد بود.

۸ مسائل باز

در چندضلعی سوراخ دار $\mathcal{P}$ این مسأله هنوز به صورت باز مطرح است. تنها کار انجام شده در این زمینه مرجع [۱۳] می‌باشد که در زمان $O(n^۲ \log^۲ n)$ مسیر پیوندی بین s و t را بدون در نظر گرفتن شرط رؤیت بدست می‌آورد. همچنین نسخه‌ی پرس‌وجوی آن هنوز حل نشده باقی است. در یک چندضلعی ساده $\mathcal{P}$ و در حالتیکه q و s و t هر کدام یک چندضلعی دلخواه، یا تعداد نقاط q بیشتر از یکی باشد، یا زاویه‌ی چرخش به r محدود شود این مسأله به صورت باز مطرح است. اگر مسأله‌ی فاصله‌ی اقلیدسی را با فاصله‌ی پیوندی ترکیب کنیم در عمل کاربردهای فراوانی خواهد داشت. تنها کار انجام شده در این زمینه یک الگوریتم تقریبی است که در [۱۰] بررسی شده است. البته می‌توان قید قابلیت دید را نیز روی آن اعمال کرد که یک مسأله‌ی جدید خواهد بود. درحالت سه بعدی نشان داده شده که این مسأله $NP-HARD$ است ولی الگوریتم‌های تقریبی کارایی برای آن ارائه شده است. قید قابلیت دید را روی این مسأله نیز می‌توان اعمال کرد.

۹ نتیجه

در اینجا مسأله‌ی فاصله‌ی پیوندی را با قید قابلیت دید در حالتیکه نقاط q و s و t ثابت باشند در زمان و حافظه‌ی $O(n)$ و در حالت پرس‌وجوی سه نقطه‌ایی با زمان و حافظه‌ی پیش‌پردازش $O(n^۳)$ و زمان پرس‌وجوی $O(\log n)$ بررسی کردیم. در حالت دوم می‌توان زمان پیش‌پردازش را به $O(nE)$ که در آن E اندازه‌ی گراف رؤیت است تقلیل داد. در بدترین حالت $E = O(n^۲)$ خواهد بود.

مراجع

[1] S. Suri. A linear time algorithm for minimum link paths inside a simple polygon, Comput. Image process.,vol.35 ,pp.99-110, 1986.

[2] S. Suri. On some link distance problem in a simple polygon, IEEE tranactions on Robotics and automation, 6 (February 1990), pp. 108-113.

[3] H. Edelsbrunner, L. Guibas, and J. Stolfi, Optimal point location in monotone subdivision, SIAM J. Comput., vol. 15, no. 2, pp. 317-340.

[4] R. Khosravi, M. Ghodsi. Shortest paths in simple polygons with polygon-meet constraints , in Proc. 19th European Workshop Comput. Geom., 2003, pp-137-142.

[5] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. Algorithmica, 2:209-233, 1987.

[6] D.T. Lee. Visibility of a simple polygon. Comput. Vision Graph. Image Process., 22:207-221, 1983.

[7] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. BIT, 27:458-473, 1987.

[8] B. Chazelle, Triangulating a simple polygon in linear time, Discrete and Computational geometry, 6 (1991), pp. 485-524.

[9] L.J. Guibas and J. Hershberger, Optimal shortest path queries in a simple polygon, Proc. Third annual ACM symposium on computational geometry, waterloo, ontario, 1987, pp. 50-63.

[10] E.M. Arkin, J. S. B. Mitchell, S. Suri: Optimal link path queries in a simple polygon. SODA 1992: 269-279.

[11] H. Elgindy and D. Avis, A linear algorithm for computing the visibility polygon from a point, Algorithm, vol. 2, no. 2, pp. 186-197, 1981.

[12] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. In Proc. 4th Canad. Conf. Comput. Geom., pages 23-28, 1992.

[13] J. S. B. Mitchell, G. Rote, and Woeginger, Minimum-Link Paths Among Obstacles in Plane, Proc. 6th Annual ACM Symposium on computational Geometry, Berkeley, CA, June 6-8, 1990, pp. 63-72. To appear, Algorithmica.