علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۷، شماره ۲، ۱۳۹۸
صفحات ۱۲۳- ۱۱۲
مقاله عادی

محبوس کردن چندضلعی‌های نادقیق با استفاده از دو انگشت

منصور داودی منفرد[۱]، اسماعیل دلفراز پهلوانلو[۲]، سید مقداد نبوی لاریمی*[۳]

*نویسنده مسئول، دریافت: ۹۸/۰۵/۰۵، بازنگری: ۹۸/۰۹/۰۲، پذیرش: ۹۹/۰۴/۱۶

[۱] استادیار، دانشکده علوم کامپیوتر و فناوری اطلاعات، دانشگاه تحصیلات تکمیلی علوم پایه، زنجان، ایران
[۲ و ۳] دانش‌آموخته کارشناسی ارشد، دانشکده علوم کامپیوتر و فناوری اطلاعات، دانشگاه تحصیلات تکمیلی علوم پایه، زنجان، ایران

چکیده

یکی از مسائل کاربردی در حوزه روباتیک، مسئله محبوس کردن یک شیء دلخواه است. تمامی الگوریتم‌های مطرح در این زمینه فرض می‌کنند که شیء دقیق باشد ولی به دلیل وجود خطا در محاسبات و ساخت اشیا این فرض می‌تواند اشتباه باشد و شیء موردنظر تا حدودی نادقیق باشد. هدف ما ارائه راهکاری مناسب برای محبوس کردن یک چندضلعی با رئوس نادقیق است. در این مقاله الگوریتمی برای یافتن تمام موقعیت‌های دو نقطه که به‌طورقطع چندضلعی نادقیق را محبوس می‌کند، ارائه می‌دهیم و نشان می‌دهیم این الگوریتم تمام مجموعه‌های محبوس کننده را در زمان $O(n^2 \log n)$ و فضای $O(n^2)$ گزارش می‌دهد که n تعداد رئوس چندضلعی نادقیق است.

کلمات کلیدی: الگوریتم، عدم قطعیت، محبوس کردن، نادقیقی

۱- مقدمه

مسائل در حوزه چنگ‌زدن[۱] به دو دسته کلی چنگ ثابت[۲] و چنگ آزاد[۳] تقسیم می‌شوند. در چنگ ثابت شیء کاملاً ثابت و بی‌حرکت است و حتی با وارد کردن نیرو در هر جهت دلخواه، شیء نمی‌تواند در میان اجزای چنگ جابه‌جا شود. در گذشته طراحان ربات برای انجام کارهایی مانند جابه‌جایی اشیا توسط بازوهای ربات از چنگ ثابت استفاده می‌کردند، اما انسان برای انجام چنین کارهایی، شیء موردنظر را در میان انگشت‌های دست خود ثابت نمی‌کند. با توجه به این‌که پژوهشگران و طراحان ربات همیشه سعی می‌کنند از رفتار انسان در انجام کارها برای ربات‌ها الگوبرداری کنند، امروزه تمایل دارند از نوع دیگری از چنگ‌ها به نام چنگ آزاد استفاده کنند [۱].

مجموعه‌ای از انگشت‌های ربات را در نظر بگیرید که اطراف یک شیء قرار گرفته‌اند به‌نحوی‌که شیء ممکن است در میان انگشت‌های این ربات به‌صورت آزادانه حرکت کند اما قادر نیست از ناحیه‌ای که توسط آن‌ها محدود شده است، فرار کند که در این حالت می‌گوییم شیء محبوس شده است. با فرض بر این‌که شیء موردنظر از حجم مشت ربات کوچک‌تر است و با توجه به این‌که شیء مذکور قادر نیست از چنین چیدمان محبوس کننده توسط انگشت‌های ربات خارج شود، ازاین‌رو با حرکت بازوی ربات، شیء به محل دلخواه دیگری جابه‌جا می‌شود.

کوپربرگ [۲] در سال ۱۹۹۰ با الهام از مطالعات بسیکوویچ [۳] و شفارد [۴]، که در زمینه ریاضیات و هندسه انجام شده بود، مسئله محبوس کردن اشیا مسطح را با استفاده از مفهوم چنگ آزاد این‌گونه بیان کرد: «یافتن مجموعه مکان‌های قرارگیری اجزای چنگ به‌گونه‌ای که شیء موردنظر قادر نباشد به موقعیتی دور و دلخواه از موقعیت اولیه‌اش حرکت کند مگر این‌که حداقل یکی از اجزای چنگ به داخل چندضلعی نفوذ کند.»

در بیش‌تر روش‌هایی که نسبت به محدودیت‌ها و حالت‌های مختلف مسئله محبوس کردن ازجمله محبوس کردن با دو انگشت و سه انگشت ارائه شده است، انگشت‌ها و چندضلعی‌ها دقیق فرض شده‌اند، درحالی‌که در دنیای واقعی علاوه بر محدودیت‌های حرکتی بازوهای مکانیکی ربات، پاره‌ای از داده‌ها به دلیل منابع خطاهای متفاوت در دسترس نیستند، مانند مکان دقیق رئوس و اضلاع چندضلعی و این‌که ممکن است شیء موردنظر، قطعه‌ای باشد که توسط یک دستگاه تولید شده باشد. بنابراین، عدم قطعیت[۴]، روش‌های ذکر شده را ناکارآمد می‌کند. ازاین‌رو در نظر گرفتن عدم قطعیت در الگوریتم‌هایی که برای مسئله محبوس کردن چندضلعی‌ها ارائه می‌شود، بسیار حیاتی است.

در این پژوهش اجزای چنگ را دو نقطه در نظر می‌گیریم که هر نقطه نماینده یک انگشت از بازوی مکانیکی ربات است ولی رئوس چندضلعی دارای خطا یا به اصطلاح نادقیق[۵] هستند. به عبارتی یک چندضلعی نادقیق داده شده است و هدف، یافتن تمام زوج نقطه‌هایی است که با وجود نادقیق بودن رئوس چندضلعی، توانایی محبوس کردن چندضلعی را با اطمینان کامل داشته باشد.

در سال ۲۰۰۶ پیپاتاناسامپورن و سودسانگ [۵] و به‌طور مستقل از آن‌ها، واحدی و استپن [۶] الگوریتم‌هایی برای یافتن تمام موقعیت‌های محبوس کننده یک چندضلعی دقیق با پیچیدگی زمانی $O(n^2 \log n)$ و پیچیدگی حافظه $O(n^2)$ ارائه دادند که در انتهای اجرای الگوریتم، ساختمان داده‌ای حاصل شد که توانایی پاسخ به پرسش محبوس کننده یا غیرمحبوس‌کننده بودن یک موقعیت داده شده در زمان $O(\log n)$را دارد. از کارهای دیگری که در زمینه محبوس کردن انجام شده و در ارتباط با عدم قطعیت است می‌توان به مسئله محبوس کردن چندضلعی‌هایی که مرز آن‌ها ناکامل است، اشاره کرد [۷]. در این مسئله این‌گونه فرض شده است که چون مرز چندضلعی توسط لیزر پوینده محدوده[۶] مشخص می‌شود، ازاین‌رو به دلیل دید جزئی و همچنین امکان وجود خطا، چندضلعی موردنظر ناکامل است.

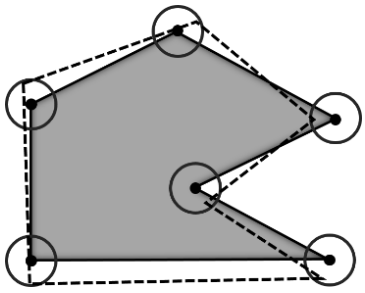
در این مقاله الگوریتمی برای یافتن تمام موقعیت‌های دو نقطه که به‌طورقطعی چندضلعی نادقیق داده شده را محبوس می‌کنند، ارائه می‌کنیم که مجموعه‌های محبوس کننده را در زمان $O(n^2 \log n)$ گزارش می‌کند که در این رابطه n تعداد رئوس چندضلعی و حافظه مورد نیاز این الگوریتم نیز از مرتبه $O(n^2)$ است. در بخش دوم با در نظر گرفتن عدم قطعیت در مسئله، الگوریتمی برای حل مسئله محبوس کردن چندضلعی نادقیق با دو انگشت ارائه می‌دهیم. در انتهای مقاله به نسخه پرسشی این مسئله نیز پاسخ می‌دهیم. در پایان نیز نتیجه‌گیری و جنبه دیگری از مسئله را را در قالب کارهای آتی بیان خواهیم کرد.

۲- عدم‌قطعیت

بیش‌تر راه‌حل‌های موجود در الگوریتم‌های خودکاری، جهان ایده‌آل را در نظر می‌گیرند که در آن قطعات کاملاً منطبق با مدل CAD شان[۷] هستند و بازوهای مکانیکی و حسگرها بی‌نهایت دقیق هستند. بااین‌حال، در زندگی واقعی، قطعات با قابلیت تحمل خطا[۸] تولید می‌شوند [۸،۹] که ازاین‌رو در قالب شکل [۱۰] و حسگرها [۱۱] متفاوت‌اند و محرک‌ها[۹] [۱۲] هم نادقیق هستند که باعث می‌شوند الگوریتم‌های مزبور زمانی که در عمل به کار گرفته می‌شوند، نتیجه درستی ندهند. به همهٔ شکل‌ها، جدول‌ها و نمودارها در مقاله باید اشاره کرد. اشاره به شکل‌ها در متن، با ذکر شماره شکل و همان سایز متن مقاله و بدون پرانتز است. مگر در پایان جمله که در این حالت در داخل پرانتز اشاره می‌شود.

نادقیقی در حوزه مسائل عدم قطعیت است. مسئله محبوس کردن چندضلعی‌ها، به‌عنوان یک مسئله کاربردی و به‌روز در حوزه ربوتیک مطرح است. تاکنون روش‌های متنوعی نسبت به محدودیت‌ها و حالت‌های مختلف محبوس کردن ازجمله محبوس کردن با دو انگشت، سه انگشت و ... ارائه شده است. در بیش‌تر این روش‌ها انگشت‌ها و چندضلعی‌ها دقیق فرض شده‌اند. این در حالی است که علاوه بر محدودیت‌های حرکتی بازوهای مکانیکی، پاره‌ای از داده‌ها به دلیل منابع خطاهای متفاوت در دسترس نیستند، مانند مکان دقیق رئوس و اضلاع چندضلعی. بنابراین، عدم قطعیت، روش‌های ذکر شده را ناکارآمد می‌کند. ازاین‌رو در نظر گرفتن عدم قطعیت در الگوریتم‌های موفق برای مسئله محبوس کردن چندضلعی‌ها بسیار حیاتی است و همچنین تعداد بسیار کم پژوهش‌ها در این زمینه نیز اهمیت پرداختن به این مسئله را دو چندان می‌کند.

ازآنجایی‌که بازوهای مکانیکی ربات در دنیای واقعی حرکت می‌کنند، بنابراین ناگزیر از در نظر گرفتن خطاهای موجود هستیم. این خطاها عبارت‌اند از: خطا در دریافت اطلاعات محیط توسط حس‌گرها، خطاهای مدل‌سازی که در زمان توصیف اشیا پیرامون با مدل‌های موجود پیش می‌آید و همچنین بروز خطای محاسبه در هنگام انجام محاسبات. پس طبق آنچه که گفته شد، ممکن است بعضی از چندضلعی‌ها ازلحاظ رئوس و اضلاع نادقیق باشند و یا حتی مکان قرارگیری دو انگشت نادقیق باشد که به همین دلیل بر آن شدیم تا مسئله محبوس کردن با دو انگشت را در حالت نادقیق بررسی کنیم [۱۳]. در نگاه اول به نظر می‌رسد که ایده‌های استفاده شده در این روش نیز قابل تعمیم به مسئله محبوس کردن چندضلعی‌های نادقیق با دو نقطه است اما زمانی که حتی یک رأس چندضلعی نادقیق باشد واضح است که مجموعه بیشینه محبوس کننده تغییر می‌کند و همچنین با در نظر گرفتن این نکته که انگشت نباید در جسم نفوذ کند، روشن می‌شود که مسئله در حالت نادقیق نیاز به مطالعه و بررسی دقیق‌تر دارد.



شکل ۱- یک چندضلعی با رئوس نادقیق که محدوده نادقیقی هر رأس آن به‌صورت دیسک با شعاع واحد است.

در سال‌های اخیر، در شاخه هندسه محاسباتی علوم کامپیوتر توجه زیادی به در نظر گرفتن عدم قطعیت در مسائل هندسی کاربردی و تئوری شده است که تلاش‌های صورت گرفته در این زمینه منجر به ایجاد مبحثی جدید با عنوان هندسه محاسباتی برای داده‌های نادقیق در این شاخه شده است. در این مباحث برای در نظر گرفتن عدم قطعیت داده‌ها، روش‌هایی مثل هندسه اپسیلون[۱۰]، مدل‌های بر پایه ناحیه[۱۱]، مدلی برای خطاهای وابسته بنام LPGUM[۱۲]، و همچنین هندسه لامبدا[۱۳]، برای خطاهای پویا مورد مطالعه قرار گرفته‌اند. به‌عنوان‌مثال از مطالعاتی که در سال‌های اخیر در زمینه هندسه لامبدا انجام شده است می‌توان به کارهای داودی منفرد در سال ۲۰۱۳ و ۲۰۱۵ اشاره کرد [۱۴] و [۱۵] که یک مدل هندسی (هندسه لامبدا) ساده (یک مدل با پیچیدگی خطی) و کارا برای مدل‌سازی مسائلی که داده‌های ورودی آن‌ها نادقیق هستند، ارائه داده است و با به‌کارگیری آن برخی از مسائل پایه‌ای هندسه محاسباتی کلاسیک، مانند کوچک‌ترین مستطیل دربرگیرنده موازی با محورهای مختصات، کوچک‌ترین و بزرگ‌ترین فاصله زوج نقاط، و مسئله خط گذرنده از مجموعه‌ای از پاره‌خط‌های موازی را حل کرده است.

ازجمله مطالعاتی که در ارتباط با هندسه محاسباتی و همچنین روباتیک انجام شده است می‌توان به مسئله جهت‌دهی قطعات با شکل متفاوت که توسط پناهی و همکاران [۱۶] در سال ۲۰۱۷ صورت گرفته است، اشاره کرد. آن‌ها بر روی تغییر شکل قطعه تمرکز کردند و تأثیر آن در مسئله جداسازی و یا جهت‌دهی قطعه با استفاده از روشی بنام هل دادن اهرم‌های غیرلغزان[۱۴] را بررسی کردند.

۳- مسئله محبوس کردن چندضلعی نادقیق

در بخش قبلی با مسئله محبوس کردن چندضلعی‌های نادقیق با دو انگشت آشنا شدیم. حال ضروری است ابزارهای بنیادی که در تحلیل مسئله مورد بحث در این مطالعه استفاده می‌شود را معرفی کنیم. ابتدا عدم قطعیت در این مسئله را توضیح می‌دهیم و پس از بیان صورت مسئله و معرفی فضای پیکربندی، تعاریف و نمادگذاری‌ها را بیان می‌کنیم. در ادامه فاصله بحرانی را معرفی می‌کنیم و رویکرد حل مسئله را نیز شرح می‌دهیم.

۳-۲- عدم قطعیت در محبوس کردن چندضلعی‌ها

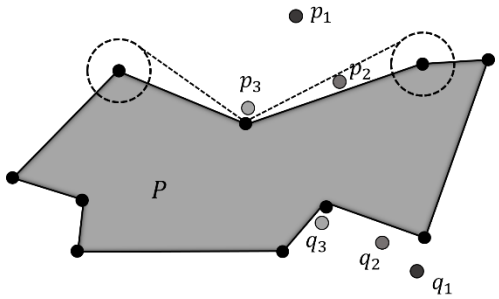
با نادقیق شدن رئوس یک چندضلعی، با سه ناحیه مختلف جستجو در اطراف چندضلعی موردنظر در ارتباط با مسئله محبوس کردن چندضلعی روبرو می‌شویم:

۱- ناحیه‌هایی وجود دارند که با وجود نادقیقی رئوس نیز هیچ چیدمان دو انگشتی را نمی‌توان یافت که در آن ناحیه‌ها، چندضلعی نادقیق موردنظر را به ازای تمام نمونه[۱۵]هایش محبوس کند، (منظور از نمونه در اینجا همان اشکال مختلف یک چندضلعی نادقیق است که به دلیل نادقیق بودن رئوس، ممکن است به هر شکلی که در محدوده رئوس نادقیق تعریف شده است، باشد که آن را با $I_{\tilde{P}}$ نمایش می‌دهیم.)، مانند چیدمان (p_1, q_1)، شکل۲،

۲- ناحیه‌هایی وجود دارند که محبوس شدن چندضلعی در آن ناحیه‌ها محتمل است، به‌عنوان‌مثال یک چیدمان دو انگشتی مشخص ممکن است چندضلعی مورد نظر را به ازای برخی از نمونه‌هایش در آن ناحیه‌ها محبوس کند و به ازای برخی دیگر از نمونه‌ها محبوس نکند، مانند چیدمان (p_2, q_2) ، شکل ۲،

۳- ناحیه‌هایی وجود دارند که با وجود نادقیقی رئوس، هر چیدمان دو انگشتی در آن ناحیه‌ها، چندضلعی نادقیق موردنظر را به ازای تمام نمونه‌هایش محبوس می‌کند، مانند چیدمان (p_3, q_3) ، شکل ۲.

هدف در این مسئله پیدا کردن ناحیه سوم یعنی یافتن تمام مجموعه‌های بیشینه محبوس‌کننده‌ای است که با وجود نادقیقی رئوس نیز چندضلعی موردنظر را قطعاً محبوس می‌کنند.



شکل ۲- نمایش وضعیت محبوس‌کنندگی یک چندضلعی نادقیق.

۳-۳- تعاریف و مفاهیم اولیه

محبوس کردن با دو نقطه: اگر هیچ مسیری وجود نداشته باشد که چندضلعی P بدون برخورد با دو نقطه p و q از طریق انتقال و دوران به موقعیتی دور نسبت به مکان اولیه‌اش برود، آنگاه چندضلعی P توسط دو نقطه p و q محبوس شده است. در این حالت موقعیت $x = (p, q)$ چندضلعی P را محبوس می‌کند و به همین دلیل $\boldsymbol{x}$ را یک موقعیت محبوس کننده می‌نامیم.

پیش از بیان صورت مسئله برای سادگی، دستگاه مختصات را متصل به چندضلعی در نظر می‌گیریم. با این کار وقتی چندضلعی حرکت کند، مختصات آن تغییر نمی‌کند اما موقعیت دو نقطه می‌تواند با حفظ فاصله تغییر کند. ازاین‌رو چندضلعی را ثابت فرض می‌کنیم و دو نقطه را با حفظ فاصله میان آن‌ها (نگاشت‌های دوران و انتقال) حرکت می‌دهیم. حال بر اساس این فرض، مفهوم محبوس‌کنندگی به این صورت بیان می‌شود که وقتی هیچ مسیری برای انتقال دو نقطه، بدون برخورد با چندضلعی، به موقعیتی دور نسبت به موقعیت اولیه وجود نداشته باشد، آنگاه نتیجه می‌گیریم که موقعیت این دو نقطه محبوس کننده است.

حال با کمک تعاریف پایه‌ای که توسط پیپاتاناسامپورن و سودسانگ [۵] در سال ۲۰۰۶ برای مسئله محبوس کردن چندضلعی مسطح دقیق با استفاده از دو انگشت انجام شده است به تعاریف اولیه مسئله موردنظرمان می‌پردازیم.

صورت مسئله: چندضلعی ساده نادقیق $P \subset \mathbb{R}^2$ با n رأس داده شده است که P یک جسم صلب ثابت و رأس نادقیق نیز دیسکی به شعاع واحد است که مختصات مرکز آن داده شده است. هدف یافتن تمام $(p, q) \in \mathbb{R}^2$ هایی است که دو نقطه p و q چندضلعی P را محبوس کنند.

در جدول زیر در مورد نمادهای مهمی که در ادامه این مطالعه استفاده می‌شود توضیح مختصری داده شده است.

جدول ۱- نمادها

تعریف	نماد
چندضلعی	P
یک نمونه از چندضلعی نادقیق	$I_{\tilde{P}}$
تعداد رئوس چندضلعی	n
چیدمان یا موقعیت دو نقطه	(p, q)
فضای کاری	W
فضای پیکربندی	C
مسیر یک نقطه	T
خط واصل یک جفت نقطه	$l[p, q]$
تابع فاصله	$d(x)$
فاصله جدایی دو نقطه در مسیر	$Sd(Tp, Tq)$
فاصله بحرانی دو نقطه	$Cd(x)$
رأس یک چندضلعی	v
ضلع یک چندضلعی	e
گراف	G
سلول یک گراف	P_{ij}
کمینه محلی یک سلول	x_{min}
فاصله بحرانی یک سلول	$Cd(P_{ij})$
مجموعه موقعیت‌های مرز مشترک دو سلول	B
وزن گذار میان دو رأس گراف	t_{ijk}
نقطه حساس یک رأس نادقیق	s

فضای کاری[۱۶] : مجموعه‌ای از نقاط که توسط چندضلعی (بازه باز P یا $Open(P)$) اشغال نشده است را فضای کاری می‌نامیم که با $W \subseteq \mathbb{R}^2$ نمایش داده می‌شود.

فضای پیکربندی[۱۷]: مجموعه همه چیدمان‌های دو-انگشتی (زوج نقطه‌ها) مجاز را فضای پیکربندی گوییم که با $C = W^2 \subseteq \mathbb{R}^4$ نشان می‌دهیم که این فضا از رابطه $C = C_p \times C_q$ محاسبه می‌شود. در این رابطه C_p و C_q شامل نقطه‌هایی هستند که خارج از چندضلعی قرار دارند.

مسیر برای نقطه: مسیری[۱۸] از نقطه $p_0 \in C_p$ به نقطه $p_1 \in C_p$ را می‌توان با تابعی پیوسته مانند $Tp: [0,1] \to C_p$ نشان داد که $Tp(0) = p_0$ و $Tp(1) = p_1$ است و به‌ازای هر $t \in [0,1]$ ، $Tp(t)$ موقعیت نقطه p در C_p را نشان می‌دهد. مسیر برای نقطه q نیز به‌طور مشابه تعریف می‌شود. توجه داشته باشید که این مسیرها در ابتدا هیچ‌گونه محدودیتی ندارند و همچنین لزوماً یک خط مستقیم نیستند. در شکل ۳ نمونه‌ای از مسیرها نشان داده شده‌اند. همان‌طور که در این شکل می‌بینید فاصله p_0 تا p_1 با سه مسیر مختلف طی شده است.

جفت مسیر برای دو نقطه: اگر Tp مسیری از نقطه p_0 به p_1 باشد و Tq مسیری از نقطه q_0 به q_1 باشد، آنگاه (Tp, Tq) را یک جفت مسیر برای دو نقطه، از موقعیت $x_0 = (p_0, q_0)$ به $x_1 = (p_1, q_1)$ می‌نامیم. در این جفت مسیر موقعیت $\big(Tp(t), Tq(t)\big)$ که با نماد $x(t)$ نمایش داده می‌شود، موقعیت جفت نقطه را در لحظه $t \in [0,1]$ در فضای پیکربندی C نشان می‌دهد.

خط واصل $l[p,q]$: اگر $x=(p,q)$ یک موقعیت از فضای پیکربندی C باشد، خطی که دو نقطه p و q را به هم متصل می‌کند، خط واصل $l[p,q]$ یا $l[x]$ نامیده می‌شود که در شکل ۳ نمایش داده شده است.

موقعیت آزاد: موقعیت $x=(p,q)\in C$ را یک موقعیت آزاد گوییم اگر خط واصل $l[x]$ چندضلعی را قطع نکند. درواقع در این موقعیت، دو نقطه p و q به‌اصطلاح همدیگر را می‌بینند. در شکل ۳ موقعیت $(p(1),q(1))$ یک موقعیت آزاد است. (در حالتی که $l[x]$ مماس بر چندضلعی باشد نیز موقعیت x آزاد محسوب می‌شود.)

مسیر فرار: به یک جفت مسیر برای موقعیت $x=(p,q)$ که به یک موقعیت آزاد ختم می‌شود، یک مسیر فرار[19] برای موقعیت x می‌گوییم که نمونه‌ای از مسیر فرار در شکل ۳ نمایش داده شده است.

فاصله: تابع $d(x):C\to\mathbb{R}$ را تابع فاصله می‌نامیم که حاصل تابع به ازای هر عضو $x=(p,q)\in C$، کم‌ترین فاصله بین دو نقطه p و q است که به‌وضوح اندازه خط راستی است که دو نقطه موردنظر را به هم وصل می‌کند یعنی این فاصله برابر با $\|\ l[x]\ \|$ است: **فاصله** $d(x)=|p-q|=\|\ l[x]\ \|$

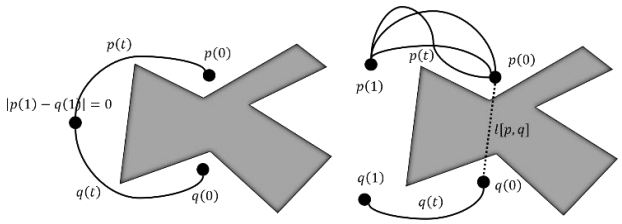
جدایی: فاصله جدایی[20] برای جفت مسیر (Tp,Tq) به‌صورت زیر است:

$$Sd(Tp,Tq)=\max_{0\le t\le 1}(d(Tp(t),Tq(t))) \qquad (۱)$$

فاصله بحرانی: تابع $Cd(x):C\to\mathbb{R}$ را تابع فاصله بحرانی[21] می‌نامیم. فاصله بحرانی موقعیت x از رابطه زیر به دست می‌آید که در آن (Tp,Tq) یک مسیر فرار برای x می‌باشد.

$$Cd(x)=\min_{\forall Tp\ \forall Tq} Sd(Tp,Tq) \qquad (۲)$$

به‌عبارت‌دیگر فاصله بحرانی Cd برای دو نقطه (p,q)، کم‌ترین فاصله‌ای است که مسیر فراری وجود داشته باشد که p و q در آن حداکثر به‌اندازه Cd از هم فاصله بگیرند.



شکل ۳- شکل سمت راست یک جفت مسیر همگام و شکل سمت چپ یک جفت مسیر فرار در یک چندضلعی را نمایش می‌دهد.

گزاره ۳,۱ موقعیت $x=(p,q)\in C$ یک موقعیت محبوس کننده است اگر شرط زیر برقرار باشد:

$$Cd(x)>d(x) \qquad (۳)$$

۳-۴- تقسیم فضای پیکربندی

اجازه دهید قبل بیان جزئیات روند حل مسئله در ابتدا یک توضیح سطح بالا از روند مدل‌سازی و ایده کلی راه‌حل ارائه شود. در قدم اول مجموعه بیشینه محبوس کننده را معرفی می‌کنیم و نشان می‌دهیم که همه موقعیت‌های یک مجموعه بیشینه محبوس کننده دارای یک فاصله بحرانی هستند. سپس یک سلول‌بندی برای فضای پیکربندی ارائه می‌کنیم و نشان می‌دهیم هر سلول در بیش‌ترین حالت با یک مجموعه بیشینه محبوس کننده اشتراک دارد و نتیجه می‌گیریم با محاسبه فاصله بحرانی موقعیت کمینه محلی هر سلول، می‌توان موقعیت‌های محبوس کننده آن سلول را یافت. در قدم بعدی مسیرهای فرار فشرده و دنباله سلولی متناظر آن‌ها را معرفی می‌کنیم. سپس با معرفی گذارها و محاسبه وزن آن‌ها کم‌ترین فاصله جدایی در هر دنباله سلولی از مسیرهای فشرده را پیدا می‌کنیم تا فضای جستجو را کاهش دهیم. در ادامه گراف اتصال که رئوس آن متناظر سلول‌ها و یال‌هایش متناظر گذارها هستند را معرفی می‌کنیم و فاصله بحرانی رئوس را از طریق انتشار[22] کم‌ترین فاصله جدایی از رأس آزاد تا هر رأس از بین فاصله‌های نقطه‌های حساس در هر گذار محاسبه می‌نماییم. سپس با استفاده از فاصله بحرانی هر سلول، مجموعه موقعیت های محبوس کننده سلول‌ها را گزارش می‌کنیم.

۳-۴-۱- شناسایی همه بیشینه‌های محبوس‌کننده

پیوستگی[23]: دو موقعیت محبوس کننده x و x' را پیوسته می‌گوییم، اگر از x به x' یک جفت مسیر که تمام موقعیت‌های آن در طول مسیر مورد نظر محبوس کننده باشند، وجود داشته باشد.

مجموعه بیشینه محبوس کننده[24]: اگر کم‌ترین حد بالای فاصله[25] دو انگشت یا $min(max(d(p,q)))$ را مجموعه بیشینه در نظر بگیریم، آنگاه هر مجموعه بیشینه از موقعیت‌های محبوس کننده پیوسته را یک مجموعه بیشینه محبوس کننده می‌نامیم. در مسئله بیشینه محبوس‌کننده یک موقعیت از دو انگشت داده می‌شود و هدف پیدا کردن بیش‌ترین فاصله‌ای است که تا زمانی فاصله دو انگشت کم‌تر از آن فاصله است شیء لزوماً محبوس شده باشد. (یا به عبارتی کم‌ترین فاصله‌ای که اگر فاصله دو انگشت مساوی آن شود، شیء از داخل چنگ می‌گریزد.) با حرکت دو نقطه (یا دو انگشت) و تلاش برای منطبق کردن آن‌ها، درحالی‌که بیش‌ترین فاصله دو نقطه کم‌ترین مقدار ممکن باشد، می‌توان به مسئله بیشینه محبوس‌کننده پاسخ داد. برای این منظور باید مسیرهای دو نقطه بررسی شوند. طبق تعریف مسیر که قبلاً به آن پرداختیم، مسیر pبرای یک نقطه، یک تابع است که $p(t)$ به ازای $t\in[0,1]$ یک موقعیت در صفحه را نشان می‌دهد و این مسیر از موقعیت $p(0)$ شروع و در موقعیت $p(1)$ خاتمه می‌یابد. بنابراین بیش‌ترین فاصله دو نقطه هنگامی‌که یکی در مسیر p و دیگری در مسیر q حرکت می‌کند، برابر با $\max_{0\le t\le 1} d(p,q)$ است. بنا بر تعریف، مسئله بیشینه محبوس‌کننده برای موقعیت (p,q)، به دنبال کم‌ترین مقدار $\max_{0\le t\le 1} d(p,q)$ از میان مسیرهای p و q ای است که از $p(0)$ و $q(0)$ شروع می‌شوند و در انتهای مسیر دو نقطه بر هم منطبق می‌شوند، به عبارتی $|p(1)-q(1)|=0$ است که به چنین مسیری برای موقعیت (p,q) همان‌طور که قبلاً تعریف کردیم، مسیر فرار گفته می‌شود (شکل ۳). دو مشاهده زیر کمک زیادی در اثبات لم‌ها و قضایایی می‌کنند که در یافتن فاصله بحرانی مؤثرند.

مشاهده ۴,۱ اگر $x=(p,q)\in C$ موقعیتی محبوس کننده و (Tp,Tq) یک جفت مسیر از x به x' با فاصله جدایی کم‌تر از $Cd(x)$ باشد، آنگاه x و x' پیوسته هستند.

مشاهده ۴,۲ اگر $x=(p,q)\in C$ موقعیتی محبوس کننده و (Tp,Tq) یک جفت مسیر از x به x' با فاصله جدایی کم‌تر از $Cd(x)$ باشد، آنگاه موقعیت $x'=(p',q')\in C$ نیز محبوس کننده است .

لم ۴,۳ اگر x و x' دو موقعیت پیوسته در فضای پیکربندی باشند، آنگاه: $Cd(x)=Cd(x')$.

بنابراین طبق تعریف مجموعه بیشینه محبوس کننده و لم ۴,۳ نتیجه می‌شود که اعضای یک مجموعه بیشینه محبوس کننده دارای فاصله بحرانی یکسانی هستند.

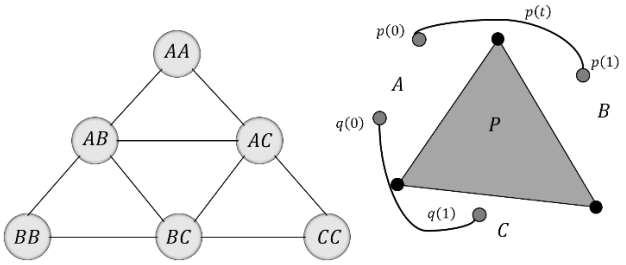
۳-۴-۲- سلول‌بندی فضای پیکربندی

پیش از این‌که روش سلول‌بندی را شرح دهیم، ابزار مورد نیاز برای انجام آن را تعریف می‌کنیم. رئوس چندضلعی P را در جهت پادساعت‌گرد به ترتیب با $v_1,v_2,\dots,v_n$ نمایش می‌دهیم و ضلع $e_i=v_iv_{i+1}$ را برای $1\le i<n$ و

$e_n = v_n v_1$ تعریف می‌کنیم. اکنون با وجود نادقیق بودن n رأس، سلول‌بندی را به‌صورت عادی انجام می‌دهیم.

$x = (p, q) \in C$ متعلق به سلول P_{ij} است، اگر e_i اولین ضلعی که با $l[p,q]$ تقاطع دارد و نزدیک‌ترین ضلع به نقطه p باشد و همچنین e_j اولین ضلعی که با $l[p,q]$ تقاطع دارد و نزدیک‌ترین ضلع به نقطه q باشد. اگر $l[p,q]$ با چندضلعی برخورد نداشته باشد، xدر سلول P_{free} قرار دارد. اگر برای $x \in P_{ij}$، خط $l[x]$ در یکی از رئوس e_i یا e_j (یا هر دو) بر چندضلعی مماس باشد، آنگاه x متعلق به مرز مشترک P_{ij} و P_{free} است.

از آنجایی‌که سلول‌بندی در فضای پیکربندی موردنظر ما در فضای چهاربعدی است، نمایش آن را به‌صورت گرافیکی امکان‌پذیر نمی‌باشد ولی برای درک بهتر نحوه سلول‌بندی، سعی کردیم نمونه‌ای از سلول‌بندی ساده برای یک مثلث را در دو بعد به‌صورت شهودی در شکل ۴ نمایش دهیم. توجه داشته باشید که در سلول‌بندی موردنظر فرض بر این است که دو نقطه به‌طور هم‌زمان نمی‌توانند حرکت کنند.



شکل ۴- شکل سمت راست ناحیه‌های تقسیم‌شده فرضی اطراف مثلث را نشان می‌دهد و شکل سمت چپ سلول‌بندی فضای پیکربندی متناظر با مثلث را به‌صورت گرافیکی در دو بعد نمایش می‌دهد.

حال در راستای یافتن مجموعه بیشینه محبوس کننده لازم است در هر سلول موقعیت‌هایی که در آن فاصله دو نقطه از هم کم‌ترین است را شناسایی کنیم. به همین دلیل مفهومی به نام کمینه محلی را معرفی می‌کنیم.

کمینه محلی: موقعیت x_{min} در سلول P_{ij} را کمینه محلی[۲۶] این سلول می نامیم درصورتی‌که به ازای هر $x \in P_{ij}$ داشته باشیم: $d(x_{min}) \leq d(x)$

لم ۴,۴ اگر x_{min} کمینه محلی P_{ij} باشد، از هر موقعیت دلخواه $x \in P_{ij}$ می توان با مسیری که در آن فاصله نقطه و خط نزولی است، به x_{min} رسید.

اثبات: برای اثبات این لم، مسیری با اتصال دو زیر مسیر پیوسته ساخته می‌شود که فاصله دو نقطه در آن به‌صورت نزولی است که این مسیر از موقعیت دلخواه $x \in P_{ij}$ به $x_{min} \in P_{ij}$ ایجاد می‌شود که در شکل ۵ نمایش داده شده است.

۱- مسیر از موقعیت $x = (p, q)$ به (p', q'):

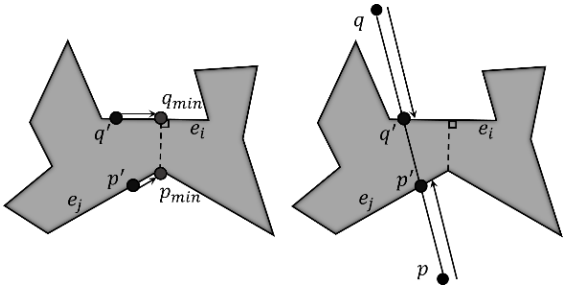
محل برخورد خط $l[x]$ با ضلع e_j و e_i را به ترتیب p' و q' می‌نامیم. هنگامی‌که از موقعیت x شروع و دو نقطه p و q را در راستای خط $l[x]$، در مسیری که فاصله دو نقطه سیر نزولی دارد، به هم نزدیک کنیم به موقعیت $(p', q') \in P_{ij}$ می‌رسیم.

۲- مسیر از موقعیت (p', q') به $x_{min} = (p_{min}, q_{min})$:

نقاط p' و q' را روی اضلاع e_j و e_i به ترتیب به موقعیت‌های p_{min} و q_{min} منتقل می‌کنیم. طبق تعریف کمینه محلی، با حرکت از p' به p_{min} و همچنین q' به q_{min}، فاصله دو نقطه به‌صورت نزولی تغییر می‌کند.

قضیه ۴,۵ هر سلول فضای پیکربندی C حداکثر با یک مجموعه بیشینه محبوس کننده اشتراک دارد.

اثبات: برای اثبات این قضیه یک موقعیت را به‌عنوان نماینده سلول در نظر می گیریم. اگر نشان دهیم تمام موقعیت‌های محبوس کننده سلول مورد نظر در مجموعه بیشینه محبوس کننده، شامل موقعیت در نظر گرفته شده هستند، آنگاه قضیه اثبات می‌شود. x' یک موقعیت محبوس کننده دلخواه و x_{min} نیز کمینه محلی سلول P_{ij} یا همان موقعیت نماینده است. با توجه به لم ۴,۴ از x' مسیری با فاصله جدایی $d(x')$ به موقعیت x_{min} وجود دارد و از مشاهده ۴,۱ نتیجه می‌گیریم که دو موقعیت x' و x_{min} پیوسته هستند. بنابراین همه موقعیت‌های محبوس کننده سلول P_{ij} در مجموعه بیشینه محبوس کننده شامل موقعیت نماینده x_{min} قرار دارند.



شکل ۵- نمایش ساخت دو زیر مسیر پیوسته برای ایجاد کمینه محلی در یک چندضلعی.

لم ۴,۶ اگر موقعیت x_{min} کمینه محلی سلول P_{ij} باشد، آنگاه $\min_{x \in P_{ij}} Cd(x) = Cd(x_{min})$.

اثبات: در پیوست این لم اثبات می‌شود.

فاصله بحرانی سلول: فاصله بحرانی سلول P_{ij} برابر با کم‌ترین فاصله بحرانی موقعیت‌های موجود در آن سلول می‌باشد که مطابق لم ۴,۶ معادل فاصله بحرانی کمینه محلی سلول مورد نظر است و با نماد $Cd(P_{ij})$ نمایش داده می‌شود. فاصله بحرانی سلول از رابطه زیر محاسبه می‌شود:

$$Cd(P_{ij}) = \min_{x \in P_{ij}} Cd(x) = Cd(x_{min}) \qquad (۴)$$

قضیه ۴,۷ موقعیت $x \in P_{ij}$ محبوس کننده است اگر و فقط اگر شرط مقابل برقرار باشد: $d(x) < Cd(P_{ij})$

اثبات: اگر موقعیت x محبوس کننده باشد، آنگاه $d(x) < Cd(x)$ است و به دنبال آن $Cd(x) = Cd(x_{min})$ می‌باشد. پس می‌توان نتیجه گرفت که $d(x) < Cd(x_{min}) = Cd(P_{ij})$ است. برای اثبات طرف دوم نیز از تعریف فاصله بحرانی استفاده می‌کنیم. یعنی اگر $d(x) < Cd(P_{ij})$ باشد، آنگاه مطابق تعریف $Cd(P_{ij})$ داریم $d(x) < \min_{x \in P_{ij}} Cd(x)$ پس $d(x) < Cd(x)$ است، به این معنی که x محبوس کننده است.

در این بخش فضای جستجو را سلول‌بندی کردیم و در قضیه ۴,۵ اثبات کردیم که تعداد مجموعه‌های بیشینه محبوس کننده حداکثر به تعداد سلول‌ها یعنی $O(n^2)$است. بر اساس قضیه ۴,۷ برای یافتن مجموعه محبوس کننده هر سلول کافی است موقعیت‌هایی با فاصله کم‌تر از فاصله بحرانی سلول، که خود برابر فاصله بحرانی کمینه محلی سلول است را مشخص کنیم. در بخش بعد روشی برای یافتن فاصله بحرانی کمینه محلی سلول ارائه می‌کنیم.

۳-۵- کاهش فضای جستجو

در این بخش به محاسبه فاصله بحرانی یک موقعیت که برابر کم‌ترین فاصله جدایی مسیرهای فرار ممکن است، می‌پردازیم. مشکلی که در محاسبه فاصله بحرانی وجود دارد این است که تعداد مسیرهای فرار ممکن برای هر موقعیت، بی‌نهایت است. رویکرد پیش‌رو، تعیین مسیرهای مؤثر در فاصله بحرانی یعنی مسیرهای فرار با فاصله جدایی کم‌تر نسبت به سایر مسیرهاست. بدین ترتیب فاصله بحرانی هر موقعیت با بررسی فاصله جدایی تعداد محدودی مسیر مشخص می‌شود. در ادامه مراحل ساخت مسیر فرار فشرده[۲۷] را توضیح می‌دهیم سپس دنباله سلول متناظر با مسیر فرار

فشرده را معرفی کرده و ارتباط آن با کم‌ترین فاصله جدایی مسیر فرار فشرده را بیان می‌کنیم. در انتها با کمک انتقال بین سلول‌ها یا گذار[۲۸] و کم‌ترین فاصله جدایی در یک دنباله سلولی را با کمک وزن گذارهای پایه محاسبه می‌کنیم.

۳-۵-۱- ایجاد مسیرهای فرار فشرده

ساخت مسیر فرار فشرده (Tp_{st}, Tq_{st}) متناظر با (Tp, Tq) در سه مرحله انجام می‌شود:

۱- ساخت مسیر (Tp', Tq') متناظر با (Tp, Tq):

نزدیک‌ترین نقطه به $Tp(t)$ از میان نقاط برخورد خط $l\big(Tp(t), Tq(t)\big)$ با مرز چندضلعی را $Tp'(t)$ می‌نامیم و برای نقطه $Tq(t)$ نیز به همین شکل عمل می کنیم. ازاین‌رو توابع $Tp'(t)$ و $Tq'(t)$ برای $t \in [0, t']$ به‌گونه‌ای تعریف می‌شود که t' کم‌ترین مقداری است که موقعیت $\big(Tp'(t'), Tq'(t')\big)$ در آن آزاد است. روشن است که فاصله دو نقطه در هر لحظه از مسیر (Tp', Tq') کم‌تر مساوی فاصله دو نقطه در همان لحظه از مسیر (Tp, Tq) است. پس:

$$\max_{t\in[0,t']} d\big(Tp'(t), Tq'(t)\big) \le \max_{t\in[0,1]} d\big(Tp(t), Tq(t)\big) \qquad (۵)$$

۲- ساخت مسیر (Tp'', Tq'') متناظر با (Tp', Tq'):

در این مرحله حالتی را در نظر می‌گیریم که ممکن است مسیرهای Tp' و Tq' پیوسته نباشند، برای مثال در شکل ۶ اگر مسیر حرکت نقطه p به سمت راست باشد، لحظه بعد از a ، $Tp'(t)$ از موقعیت $Tp'(a)$ روی ضلع e_i به نقطه مجاور y روی ضلع e_k منتقل می‌شود که به همین دلیل با یک ناپیوستگی روبه‌رو می‌شویم. برای رفع این قبیل ناپیوستگی‌ها، با اضافه کردن خط‌هایی در راستای خط واصل $l[p, q]$ که نقاط ناپیوسته مسیر را به هم وصل می‌کند، مسیر پیوسته (Tp'', Tq'') را متناظر با (Tp', Tq') می‌سازیم. به ازای هر ناپیوستگی در زمان های مختلف (در اینجا تنها زمان a)، مسیر مورد نظر (Tp'', Tq'') قبل و بعد از زمان a به‌صورت زیر تعریف می‌شود:

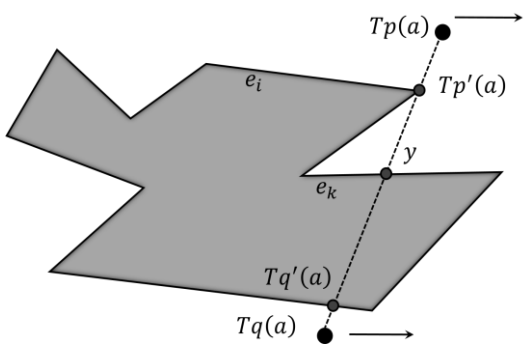
۱- برای زمان $t \in [0, a]$ مسیرها تغییری نمی‌کند پس $Tp''(t) = Tp'(t)$ و $Tq''(t) = Tq'(t)$

۲- برای زمان $t \in [a, a+\varepsilon]$ هر یک از نقاط که در وضعیت ناپیوستگی قرار بگیرند، در مسیر پیوسته روی خط مستقیم که در راستای خط واصل $l[p, q]$ است از $Tp'(a)$ به سمت نقطه y حرکت می‌کند. پس $Tp''(a) = Tp'(a)$ و $Tp''(a+\varepsilon) = y$ است و به‌طور مشابه برای نقطه q نیز تعریف می‌شود.

۳- برای زمان $t \ge a+\varepsilon$ نیز مانند حالت اول که ناپیوستگی وجود ندارد، مسیرها به‌صورت $Tp''(t+\varepsilon) = Tp'(t)$ و $Tq''(t+\varepsilon) = Tq'(t)$ تعریف می شوند.

روشن است که :

$$\max_{t\in[0,t'']} \big(d\big(Tp''(t), Tq''(t)\big)\big) = \max_{t\in[0,t']} \big(d\big(Tp'(t), Tq'(t)\big)\big) \qquad (۶)$$



شکل ۶- نمایش ناپیوستگی مسیر Tp'

۳- ساخت مسیر (Tp_{st}, Tq_{st}) متناظر با (Tp'', Tq''):

مسیر (Tp'', Tq'') حاصل شده از مرحله دوم، ممکن است در بازه زمانی غیر از $[۰, ۱]$ قرار بگیرد، ازاین‌رو پارامتر t را به‌گونه‌ای بزرگ‌نمایی یا کوچک‌نمایی می‌کنیم که Tp'' و Tq'' بر اساس تعریفی که در ابتدای فصل از جفت مسیر برای دو نقطه داشتیم، در بازه زمانی t قرار بگیرند که مسیرهای حاصل از این مرحله را Tp_{st} و Tq_{st} می‌نامیم. بنابراین روشن است که:

$$\max_{t\in[0,1]} \big(d\big(Tp_{st}(t), Tq_{st}(t)\big)\big) = \max_{t\in[0,t'']} \big(d\big(Tp''(t), Tq''(t)\big)\big) \qquad (۷)$$

با توجه به این‌که که کمینه محلی نیز یک موقعیت دلخواه در هر سلول محسوب می‌شود، برای تعیین فاصله بحرانی کمینه محلی هر سلول، مانند دیگر نقاط دلخواه در سلول، فاصله جدایی مسیرهای فرار فشرده برای کمینه محلی مورد نظر را محاسبه می‌کنیم.

همان‌طور که توضیح دادیم با تغییر موقعیت دو نقطه در طی مسیر فرار فشرده، ممکن است سلول شامل دو نقطه در بعضی از لحظات تغییر کند. با توجه به نحوه ساخت (Tp_{st}, Tq_{st}) این نتیجه حاصل می‌شود (فقط) موقعیت انتهایی هر مسیر فرار فشرده متعلق به سلول P_{free} است زیرا هدف منطبق کردن دو نقطه روی هم می‌باشد. فرض کنید مسیر فرار فشرده (Tp_{st}, Tq_{st}) به ترتیب از سلول‌های $P_1, P_2, \dots, P_f, P_{free}$ عبور می‌کند، اگر در لحظات $t_1, t_2, \dots, t_f$ تغییر سلول رخ دهد، آنگاه برای $t \in [t_{r-1}, t_r]_{r=1,\dots,f-1}$ عبارت زیر را داریم $x(t) =$ $\big(Tp_{st}(t), Tq_{st}(t)\big) \in P_r$ و همچنین:

$$\begin{cases} x(t_0 = 0) \in P_1 \\ x(t_r) \in B(P_r, P_{r+1}) \quad r = 1,2,\dots,f-1 \\ x\big(t_f = 1\big) \in B\big(P_f, P_{free}\big) \end{cases} \qquad (۸)$$

که B مجموعه موقعیت‌های متعلق به مرز مشترک دو سلول را نمایش می‌دهد . $(f = final)$

لم ۵،۱ اگر $x_1 \in B(P_1, P_2)$ و $x_2 \in B(P_2, P_3)$ باشد، آنگاه در میان مسیرهای موجود از x_1 به x_2 کم‌ترین فاصله جدایی برابر با $max\{d(x_1), d(x_2)\}$ است.

اثبات: در پیوست این لم اثبات می‌شود.

بنابراین در بین تمام مسیرهای فرار فشرده‌ای که از میان $P_1, P_2, \dots, P_f, P_{free}$ می‌گذرند، مسیری که از کمینه محلی پیکربندی‌های مرزهای مشترک می‌گذرد، کم‌ترین فاصله جدایی را دارد درصورتی‌که $B\big(P_{ij}, P_{jk}\big) \neq \emptyset$ باشد، یک گذار میان دو سلول P_{ij} و P_{jk} وجود دارد و وزن گذار به ازای هر $x \in B\big(P_{ij}, P_{jk}\big)$ برابر است با $min(d(x))$ در بدترین حالت، تعداد گذارهایی که یک سلول می‌تواند داشته باشد به‌اندازه $O(n)$ است. خوشبختانه تنها $O(1)$ گذار مرتبط با هر سلول که گذار پایه نامیده می‌شود در محاسبه فاصله جدایی هر مسیر فرار فشرده تأثیر می‌گذارد.

قضیه ۵،۲ اگر گذاری میان دو سلول P_1 و P_2 با وزن $d(x_{12})$ وجود داشته باشد، دنباله‌ای از گذارهای پایه وجود دارند که از سلول P_1 شروع می‌شوند و در P_2 پایان می‌یابند که در آن بیش‌ترین وزن گذارهای پایه کم‌تر یا مساوی $d(x_{12})$ است.

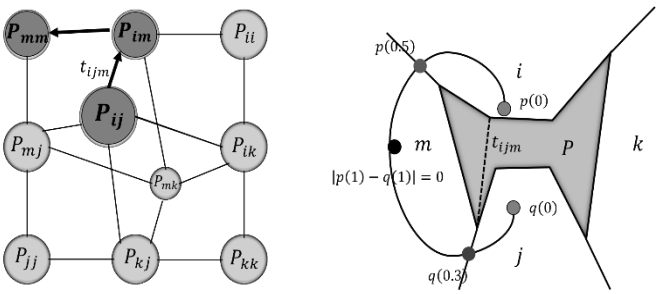
برای $x_{min} \in F_i$ هر مسیر فرار فشرده متناظر است با دنباله‌ای از سلول‌ها که از F_i شروع می‌شود و در F_{free} پایان می‌یابد. پس به‌منظور محاسبه $Cd(x_{min})$ که کم‌ترین فاصله جدایی تمامی مسیرهای فرار فشرده برای x_{min} است باید کم ترین بیش‌ترین وزن گذارهای پایه در هر دنباله سلول‌ها که از F_i شروع می‌شود و در F_{free} پایان می‌یابد را پیدا کنیم.

در این مقاله گذارهای پایه شرح داده نشده است؛ بااین‌حال یافتن همه گذارهای پایه و وزن‌هایشان در زمان $O(n^2 logn)$ انجام می‌شود. به این صورت که با استفاده از الگوریتم پرتاب اشعه[۲۹] [۱۷] گذارهای پایه را در زمان $O(logn)$ پیدا می‌کنیم که این عمل با توجه به حداکثر تعداد سلول‌ها، $O(n^2)$ تکرار می‌شود. با توجه به نتایج و مفاهیمی که تاکنون به‌دست‌آمده می‌توانیم فضای جستجو را مشخص کنیم.

گراف اتصال[۳۰]: گراف وزن‌دار $G = (V_G, E_G)$ که اجتماع سلول‌ها است را گراف اتصال می‌نامیم. هر رأس $P_{ij} \in V_G$ که در آن $i = 1, 2, \dots, O(n)$ و $j = 1, 2, \dots, O(n)$ می‌باشد، متناظر با یک سلول از فضای پیکربندی است و میان دو رأس P_{ij} و P_{ik} یال e_{jk} (که با یال e_{kj} برابر می‌باشد) وجود دارد اگر و فقط اگر بین سلول‌های متناظر با P_{ij} و P_{ik} گذار پایه وجود داشته باشد و وزن یال e_{jk} که با t_{ijk} نشان می‌دهیم برابر با وزن آن گذار پایه است.

همان‌طور که پیش‌ازاین توضیح داده شد، هر مسیر فرار فشرده متناظر با یک دنباله از گذارهای پایه و هر گذار پایه یک یال از گراف G است، به‌این‌ترتیب هر مسیر فرار فشرده برای کمینه محلی سلول P_{ij} متناظر با یک مسیر در گراف اتصال است که از رأس P_{ij} شروع و به رأس P_{free} ختم می‌شود. بنابراین محاسبه فاصله بحرانی سلول P_{ij} معادل یافتن مسیری با کم‌ترین مقدار بیش‌ترین وزن از رأس P_{ij} به رأس P_{free} است.

شکل ۷ نمونه‌ای از گراف اتصال را در دو بعد نمایش می‌دهد. برای یافتن مجموعه نقاط محبوس‌کننده، یک گراف اتصال با مشخصات شرح داده شده ساخته می‌شود که در آن هر گره نشان‌دهنده ناحیه‌های قرارگیری دو نقطه (دو انگشت ربات) از فضای اطراف چندضلعی است (به هر ناحیه یک سلول می‌گوییم). چون فضای پیکربندی چهاربعدی است و امکان نمایش آن وجود ندارد پس ما ناحیه قرارگیری دو نقطه را در یک گره نمایش می‌دهیم تا ترسیم این گراف در دو بعد میسر شود. هر یال میان دو گره نشان می‌دهد که دو سلول موردنظر همسایه هستند و با حرکت یکی از نقطه‌ها (انگشت‌ها) می‌توان از یک سلول به سلول مجاور منتقل شد که همان‌طور قبلاً اشاره کردیم به آن گذارپایه می‌گوییم و در ادامه نیز بیش‌تر توضیح می‌دهیم. می‌توان معیاری به نام وزن را بر روی یال‌های این گراف محاسبه کرد که با t_{ijk} نمایش داده می‌شود. سپس با پیمایش رئوس (گره‌های) این گراف، نقاط محبوس‌کننده را به‌صورت کارا پیدا می‌کنیم. به دلیل این‌که رئوس چندضلعی مورد بررسی نادقیق هستند، ما عملیات محاسبه فاصله حساس را انجام می‌دهیم (در قسمت بعد به‌طور مفصل شرح خواهیم داد) تا بتوانیم مجموعه محبوس‌کننده قطعی را به دست بیاوریم.



شکل ۷ - نمایش گراف اتصال متناظر با سلول‌بندی فضای پیکربندی.

۳-۵-۲- محاسبه فاصله‌های حساس و بحرانی رئوس گراف

حال که تمام اجزای فضای جستجو ساخته شدند، کافی است فاصله بحرانی رئوس گراف تعیین شود، برای این کار باید به ازای هر رأس تمام مسیرهای منتهی به رأس v_{free} بررسی شود ولی با وجود نادقیق بودن رئوس کار کمی پیچیده می شود. درواقع الگوریتم‌های موجود برای چندضلعی‌های دقیق در محاسبه وزن گذارهای پایه برای چندضلعی‌های نادقیق کارایی ندارند. در ادامه ابتدا سعی می‌کنیم با انتخاب یک نقطه مناسب از میان محدوده یک رأس نادقیق، به ازای هر سلول رأس مورد نظر را ثابت بگیریم (یعنی آن رأس را دقیق کنیم) و با روندی مانند یافتن کوتاه‌ترین مسیر در گراف، به‌صورت بهینه فاصله بحرانی هر رأس را محاسبه کنیم.

ازآنجایی‌که رأس نادقیق است، به ازای هر سلول که با رأس نادقیق اشتراک دارد باید نقطه‌ای را روی رأس نادقیق انتخاب کنیم که تضمین کند جسم فرار نمی‌کند. این نقطه را نقطه حساس[۳۱] می‌نامیم و با s نمایش می‌دهیم.

با توجه به هدف مسئله در محبوس کردن چندضلعی‌های نادقیق که باید تمام مکان‌های قابل قبولی را برای دو نقطه پیدا کنیم که با وجود نادقیق بودن چندضلعی باز هم تضمین کند که چندضلعی مورد نظر از میان دو نقطه (انگشت) فرار نکند. ازاین‌رو نقطه حساسی را که در محدوده نادقیق هر رأس انتخاب می‌کنیم باید نقطه ای باشد که فاصله بحرانی جدید، مجموعه بیشینه محبوس کننده سلول مورد نظر را کاهش دهد تا جایی که به ما اطمینان دهد که مجموعه محبوس کننده موردنظر، مجموعه بیشینه محبوس کننده قطعی است. منظور از مجموعه بیشینه محبوس کننده قطعی، مجموعه‌ای است که علیرغم نادقیق بودن چندضلعی، زوج نقطه‌های موجود در مجموعه موردنظر، چندضلعی را محبوس می‌کنند.

برای سادگی ابتدا مسئله را با محبوس کردن چندضلعی دارای یک رأس نادقیق شروع می‌کنیم و سپس مسئله را به محبوس کردن چندضلعی دارای n رأس نادقیق تعمیم می‌دهیم. برای یک رأس نادقیق به دنبال نقاطی روی محدوده رأس نادقیق هستیم که مسئله را به یک مسئله دقیق تبدیل کند و آنگاه به مسئله مورد بررسی در این مطالعه پاسخ می‌دهیم.

ازآنجایی‌که هدف مسئله، گزارش محدوده‌هایی است که به‌طور قطعی چندضلعی را محبوس می‌کنند (مجموعه بیشینه محبوس کننده)، باید از کران بالای فاصله جدایی استفاده کنیم و چون بر اساس الگوریتم بیان شده و با توجه به گذارهای پایه معرفی شده، نرخ انتشار $Cd(P_{ij})$ وابسته به وزن گذار t_{ijk} است، بنابراین از مفهوم گذار در حل مسئله کمک می‌گیریم. طبق مفهوم گذار می‌دانیم که هر پیکربندی برای انتقال از یک سلول به سلولی دیگر باید هزینه t_{ijk} را صرف کند. همچنین برای راحتی کار می‌توان فرض کرد که رأس نادقیق موردنظر در گذار پایه v یا همان رأس مشترک گذار میان دو سلول باشد.

طبق گفته‌های پیشین برخلاف مسئله محبوس کردن چندضلعی‌های دقیق که رئوس مقعر نادیده گرفته می‌شدند، در این مسئله لازم است در مورد رئوس محدب و مقعر به‌طور مجزا بحث شود. درواقع طبق هدف مسئله که کاهش مجموعه بیشینه محبوس کننده برای رسیدن به مجموعه بیشینه محبوس کننده قطعی است باید برای رأس نادقیق محدب به دنبال کم‌ترین کران بالای فاصله جدایی باشیم و برای رأس نادقیق مقعر به دنبال بیش‌ترین کران بالای فاصله جدایی باشیم. درواقع برای رأس نادقیق محدب باید نقطه‌ای مانند s روی محدوده رأس نادقیق پیدا کنیم که در آن مقدار $|s - z|$ کم‌ترین مقدار را نسبت به $|s_i - z|$ داشته باشد یعنی $|s - z| \leq |s_i - z|$باشد. $s_i \in V$ یک نقطه دلخواه روی رأس نادقیق و V نیز مجموعه تمام نقاط روی رأس نادقیق است و همچنین $s_i \neq s$ می‌باشد.

لم ۵،۳ اگر گذاری از سلول P_{ij} به سلول P_{jk} وجود داشته باشد و همچنین s نقطه حساس یافت شده در محدوده دیسکی رأس نادقیق v باشد و z نیز نزدیک ترین نقطه ضلع e_i به نقطه s باشد، آنگاه $|s - z|$ کوچک‌ترین گذار بین دو سلول را نشان می‌دهد که درنتیجه داریم: $t_{ijk} = |s - z|$.

اثبات: فرض کنید نقطه‌ای مانند s' وجود دارد که $|s' - z|$ گذار بین دو سلول را نمایش می‌دهد. با توجه به این‌که گذار، کم‌ترین هزینه انتقال بین دو سلول را نشان می‌دهد، ازاین‌رو $|s' - z|$ باید کم‌ترین مقدار را داشته باشد. اما این با کمینه بودن مقدار $|s - z|$ در تضاد است. بنابراین فرض خلف باطل است و $|s - z|$ کوچک‌ترین گذار بین دو سلول را نشان می‌دهد و s نیز نقطه حساس برای دو سلول مورد نظر می‌باشد.

مشاهده ۵،۴ نقطه حساس s، روی محیط نادقیق رأس v یعنی V قرار دارد.

اثبات: در پیوست این مشاهده اثبات می‌شود.

قضیه ۵،۵ فاصله بحرانی برای سلول دلخواه در چندضلعی داده شده P که دارای یک رأس نادقیق است، برابر با $|s-z|$ است.

اثبات: در پیوست این قضیه اثبات می‌شود.

اکنون حالتی که k رأس (یا تمام رئوس) چندضلعی P نادقیق باشند را در نظر می‌گیریم. با توجه به سلول‌بندی فضای پیکربندی، هنگام گذار بین سلولی حداکثر سه رأس تأثیرگذار است.

قضیه ۵،۶ فاصله بحرانی برای سلول دلخواه در چندضلعی داده شده P که دارای k رأس نادقیق است، برابر با $|s-z|$ است.

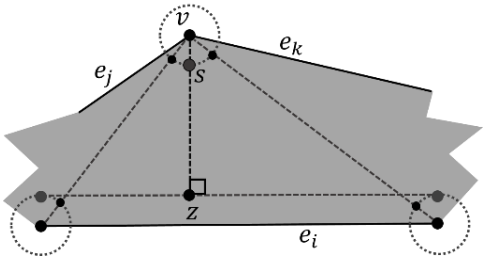
اثبات: فرض کنیم در این حالت یک رأس نادقیق همان رأس $v \in V$ (همان مرکز ناحیه نادقیق V) و رئوس نادقیق دیگر هم دو رأس ضلع e_i باشد. در این صورت مطابق لم ۵،۳ حالت‌های زیر را داریم:

۱- رأس نادقیق v محدب باشد، (شکل ۸).

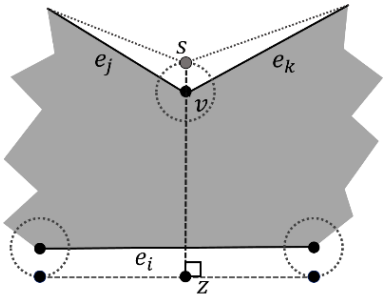
در این حالت باید توجه داشته باشیم که برای یافتن مجموعه بیشینه قطعی باید به ازای هر گذار به دنبال کم‌ترین مقدار t_{ijk} باشیم، یعنی کم‌ترین مقدار $|v-z|$ طبق شکل ۴ به‌وضوح با انتخاب نقطه s روی محیط محدوده دیسک نادقیق و محاسبه کم‌ترین فاصله آن با نزدیک‌ترین خط مماس دو ناحیه رئوس نادقیق ضلع e_i به کم‌ترین مقدار $|v-z|$ می‌رسیم، یعنی همان $|s-z|$.

۲- رأس نادقیق v مقعر باشد، (شکل ۹).

در این حالت برای یافتن مجموعه بیشینه قطعی باید به ازای هر گذار به دنبال بیش ترین مقدار t_{ijk} باشیم، یعنی بیش‌ترین مقدار $|v-z|$. طبق شکل ۹ به‌وضوح با انتخاب نقطه s روی محیط محدوده دیسک نادقیق رأس v، و محاسبه بیش‌ترین فاصله آن با دورترین خط مماس دو ناحیه رئوس نادقیق ضلع e_i به بیش‌ترین مقدار $|v-z|$ می‌رسیم، یعنی $|s-z|$.



شکل ۸- نمایش سه رأس نادقیق برای گذار پایه رأس محدب



شکل ۹- نمایش سه رأس نادقیق برای گذار پایه رأس مقعر

طبق مشاهدات بالا به‌وضوح درک می‌کنیم که نقاط بحرانی مورد نظر، روی محیط محدوده دیسکی رأس نادقیق قرار دارند. با توجه به این گفته‌ها برای هر گذار یک فاصله حساس s را به دست می‌آوریم سپس الگوریتم دقیق را به ازای هر s اجرا می‌کنیم و درنهایت از تمام مجموعه‌های بیشینه محبوس کننده به‌دست‌آمده در زمان $O(1)$ اشتراک می‌گیریم که درواقع تمام مجموعه‌های بیشینه محبوس کننده قطعی برای مسئله موردنظر گزارش خواهد شد.

قضیه ۵،۷ P_{ij} را یک رأس از گراف اتصال در نظر بگیرید. همچنین فرض کنید $B(P_{ij})$ مجموعه‌ای باشد که شامل رئوس مجاور P_{ij} است، آنگاه فاصله بحرانی رأس P_{ij} برحسب فاصله بحرانی رئوس مجاور آن به‌صورت زیر است:

$$Cd(P_{ij}) = \min_{P_{ij} \in B(P_{ij})} \left(max\left(Cd(P_{ij}), t_{ijk}\right) \right) \quad (۹)$$

اثبات: اثبات درستی فاصله بحرانی رئوس گراف، مشابه با استدلال برای فاصله بحرانی سلول‌ها در مسیرهای فرار فشرده است. یعنی با توجه به این‌که فاصله بحرانی هر رأس $P_{ij} \in B(P_{ij})$ برابر با کم‌ترین مقدار بیش‌ترین وزن در مسیرهایی از رأس P_{ij} به رأس P_{free} است، مطابق لم ۵،۱ نتیجه می‌گیریم که $max\left(Cd(P_{ij}), t_{ijk}\right)$ برابر با کم‌ترین مقدار بیش‌ترین وزن در مسیرهایی از رأس P_{ij}به رأس P_{free} است که با یال e_{ij} شروع می‌شوند. به همین دلیل کم‌ترین مقدار بیش‌ترین وزن از سلول P_{ij} به سلول P_{free} معادل با فاصله بحرانی P_{ij} می‌باشد. یعنی برابر:

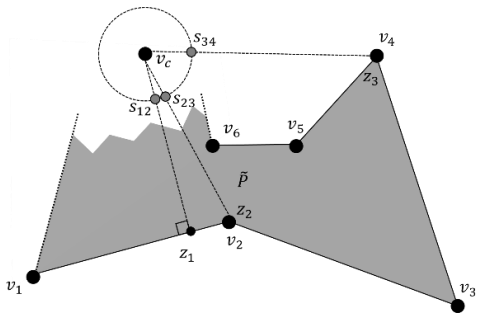
$$\min_{P_{ij} \in B(P_{ij})} \left(max\left(Cd(P_{ij}), t_{ijk}\right) \right) \quad (۱۰)$$

نتیجه ۵،۸ از قضیه بالا نتیجه می‌گیریم که اگر P_{ij} و P_{ij} دو سلول مجاور باشند، آنگاه رابطه زیر میان فاصله بحرانی P_{ij} و P_{ij} برقرار است:

$$Cd(P_{ij}) \leq max\left(Cd(P_{ij}), t_{ijk}\right) \quad (۱۱)$$

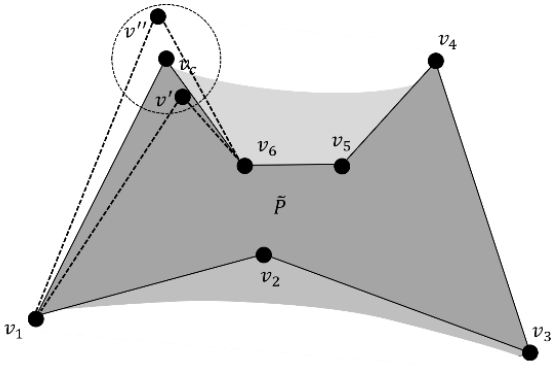
۳-۵-۳- الگوریتم

فرض کنید یک چندضلعی با یک رأس نادقیق داریم که رأس نادقیق با یک دیسک نمایش داده می‌شود. طبق الگوریتم دقیق برای مسئله محبوس کردن، رأس نادقیق تعریف نشده است و در همین ابتدای کار عدم کارایی الگوریتم دقیق واضح می‌شود. پس فرض کنیم که مرکز دیسک نادقیق v_c به‌عنوان رأس نهایی و دقیق باشد که در این صورت طبق شکل ۱۰ با یک مسئله دقیق روبرو می‌شویم که طبق الگوریتم‌های دقیق موجود، مجموعه نقاط محبوس‌کننده محاسبه می‌شود ولی مشکل اینجاست که این مجموعه به‌دست‌آمده، مجموعه محبوس‌کننده قطعی نیست که ممکن است بعضی از مجموعه جواب‌ها رو که برای نمونه‌های دیگر وجود دارد را در نظر نگیرد و یا به‌احتمال‌زیاد شامل زیرمجموعه‌های ناشدنی باشد (شکل ۱۰). (منظور از شدنی یا ناشدنی بودن یک جواب این است که به دلیل نادقیق بودن یک رأس اگر محدوده دیسکی شکل رأس نادقیق از مجموعه جواب حذف نشود آنگاه در نمونه‌هایی از چندضلعی موردنظر ممکن است قسمتی از مجموعه جواب متعلق به چندضلعی باشد که در این صورت جواب ناشدنی است.). همان‌طور که در شکل ۱۰ می‌بینید، مجموعه بیشینه محبوس‌کننده شهودی چندضلعی $\tilde{P}$ به ازای رأس دقیق v_c نشان داده شده است. حال اگر نمونه‌ای از چندضلعی نادقیق موردنظر به ازای رأس v' و v'' را داشته باشیم واضح است که این مجموعه جواب مشخص شده نادرست می‌باشد.



شکل ۱۰- نحوه عملکرد الگوریتم دقیق. در این شکل به‌طور شهودی مجموعه بیشینه محبوس‌کننده چندضلعی $\tilde{P}$ به ازای رأس دقیق v_c نشان داده شده است.

حال مطابق الگوریتم ارائه شده برای چندضلعی دلخواه با یک رأس نادقیق، بعد از سلول‌بندی فضای پیکربندی و یافتن گذارهای پایه، متناظر با هر گذار پایه مؤثر در محاسبه فاصله بحرانی، نقطه حساس متناظر با هر سلول را پیدا می‌کنیم و برای هر سلول فاصله حساس مربوط به آن را محاسبه می‌کنیم (شکل ۱۱). به‌این‌ترتیب می‌توانیم تمام مجموعه‌های بیشینه محبوس‌کننده قطعی را شناسایی کنیم. همچنین در انتها محدوده دیسک نادقیق را از مجموعه جواب حذف می‌کنیم تا مشکل ناشدنی بودن جواب را نداشته باشیم. در این صورت به ازای هر نمونه‌ای از چندضلعی موردنظر که به دلیل نادقیق بودن یک رأس آن ممکن داشته باشیم، اطمینان داریم که مجموعه بیشینه به‌دست‌آمده کاملاً صحیح و شدنی است.



شکل ۱۱- نحوه عملکرد الگوریتم نادقیق

Procedure 1 Propagate Caging (V_G, E_G)

Input:
$P \subset \mathbb{R}^2$ *with n imprecise vertices.*
Output:
All of $(p,q) \in \mathbb{R}^2$ *such that the polygon P will be caged.*

$L \leftarrow \emptyset$
$H \leftarrow$ *Make a Heap*
Insert(H, P_{free})
{*Let* P_{free} *to be a cell where the distance between the two points p and q can be zero.*}
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    if $P_{ij} = P_{free}$ **then**
      $Cd(P_{ij}) \leftarrow 0$
    else
      $Cd(P_{ij}) \leftarrow \infty$
    end if
  end for
end for
while $H \neq \emptyset$ **do**
  $Cd(P_{ij}) \leftarrow$ **Extract_Min**(H)
  if $P_{ij} \notin L$ **then**
    $L \leftarrow L \cup \{P_{ij}\}$
    $AC_{ij} \leftarrow \{(p,q) \in P_{ij} \mid d(p,q) < Cd(P_{ij})\}$
    for all $e_{jk} = \{P_{ij}, P_{ik}\} \in E_G$ **do**
      if $P_{ij} \notin L$ **then**
        $Cd(P_{ij}) \leftarrow min\left(Cd(P_{ij}), max(Cd(P_{ij}), t_{ijk})\right)$
        Return $Cd(P_{ij})$
        Insert (H, P_{ij})
      end if
    end for
  end if
end while

جدول ۲- پیچیدگی زمانی الگوریتم

عمل	زمان
محاسبه فضای پیکربندی	$O(n)$
سلول‌بندی فضای پیکربندی	$O(n^2)$
تعیین یال‌های گراف	$O(n^2 \log n)$
انتشار فاصله بحرانی رئوس	$O(n^2 \log n)$
گزارش مجموعه‌های محبوس کننده	$O(n^2)$
زمان صرف شده برای یافتن موقعیت‌های محبوس کننده	$O(n^2 \log n)$

جدول ۳- پیچیدگی حافظه الگوریتم

عمل	زمان
ذخیره چندضلعی و فضای پیکربندی	$O(n)$
ذخیره مرز سلول‌ها	$O(n^2)$
ذخیره رئوس و یال‌های گراف اتصال	$O(n^2)$
ذخیره فاصله بحرانی رئوس	$O(n^2)$
ذخیره مجموعه‌های محبوس کننده	$O(n^2)$
حافظه مصرفی برای یافتن موقعیت‌های محبوس کننده	$O(n^2)$

شبه‌کد الگوریتم یافتن موقعیت‌های محبوس کننده قطعی ارائه شده است. مرحله اول این الگوریتم ایجاد گراف اتصال است که خود شامل سلول‌بندی فضای پیکربندی و تولید گذارهای پایه است. مرحله دوم انتشار فاصله بحرانی رئوس گراف با استفاده از فاصله حساس هر گذار و گزارش مجموعه‌های محبوس کننده قطعی است. در مرحله اول، گراف G با مجموعه رئوس V_G و مجموعه یال‌های E_G در زمان $O(n^2 logn)$ تولید می‌شود. شبه‌کد ارائه شده، در مرحله دوم الگوریتم خط ۶ داریم $P_{ij} = P_{free}$ که بنا بر تعریف وقتی در سلولی دو نقطه (یا دو انگشت) به‌اصطلاح همدیگر را ببینند، سلول موردنظر P_{free} نامیده می‌شود و ازاین‌رو فاصله بحرانی در این سلول برابر صفر است. درواقع اگر در گراف، مسیری به این‌گونه سلول‌ها ختم شود به آن مسیر فرار می‌گویند. در ادامه مرحله دوم الگوریتم فاصله بحرانی رئوس را از طریق انتشار کم‌ترین فاصله جدایی از رأس آزاد تا هر رأس از بین فاصله های نقطه‌های حساس در هر گذار محاسبه می‌نماید را روی گراف G بیان می‌کند. در این مرحله $O(n^2)$ بار $Extract\_Min(H)$ و حداکثر $O(n^2)$ بار $HeapInsert$ انجام می‌شود و هر یک از این اعمال در زمان $O(logn)$ اجرا می شوند. بنابراین زمان کل مرحله دوم $O(n^2 logn)$ است.

پیچیدگی زمانی و حافظه الگوریتم: با توجه به مطالبی که از ابتدای فصل تا به اینجا بیان کردیم، پیچیدگی زمانی الگوریتم ارائه‌شده از مرتبه $O(n^2 logn)$ و پیچیدگی حافظه آن از مرتبه $O(n^2)$ است. در جداول ۲ و ۳ پیچیدگی بخش‌های مختلف الگوریتم موردنظر ذکر شده است.

۳-۵-۴- تشخیص محبوس‌کنندگی موقعیت داده شده

برای تشخیص محبوس کننده بودن یا نبودن یک موقعیت دلخواه ابتدا باید سلولی که موقعیت داده شده در آن قرار دارد را پیدا کنیم و پس‌ازآن با مقایسه $d(x)$ با فاصله بحرانی آن سلول، وضعیت محبوس‌کنندگی موقعیت موردنظر را مشخص کنیم.

لم ۵،۹ با توجه به موقعیت داده شده $x = (p,q)$، یافتن سلول P_{ij} که x در آن قرار دارد، از مرتبه زمانی $O(logn)$ است که n تعداد رئوس چندضلعی نادقیق موردنظر است.

اثبات: برای تشخیص سلولی که x در آن قرار دارد باید موقعیت (مختصات) دو نقطه را مورد بررسی قرار دهیم. به‌طور مثال برای نقطه p، باید ضلع e_j از چندضلعی که اولین ضلع متقاطع با $l[p,q]$ است را در صورت وجود بیابیم. این کار با استفاده

از الگوریتم پرتاب اشعه [۱۷] در زمان $O(logn)$ انجام می‌شود. اگر $l[p,q]$ با هیچ ضلعی برخورد نداشت، موقعیت x به سلول آزاد P_{free} تعلق دارد.

قضیه ۵،۱۰ موقعیت $x=(p,q)$ داده شده است. با استفاده از گراف اتصال می توان محبوس‌کنندگی x را در زمان $O(logn)$ مشخص کرد که n تعداد رئوس چندضلعی نادقیق موردنظر است.

اثبات: برای تشخیص محبوس کننده یا غیرمحبوس‌کننده بودن یک موقعیت داده شده مانند $x=(p,q)$ با توجه به لم ۵،۹ سلولی که موقعیت x در آن قرار دارد را در زمان $O(logn)$ پیدا می‌کنیم. اگر x متعلق به سلول P_{ij} باشد، آنگاه مطابق قضیه ۴،۷ با مقایسه $d(x)$ و $Cd(P_{ij})$ محبوس کننده یا غیرمحبوس‌کننده بودن x مشخص می‌شود. بنابراین تشخیص محبوس‌کنندگی یک موقعیت داده شده با استفاده از گراف اتصال G در زمان $O(logn)$ انجام می‌شود.

۴- نتیجه‌گیری و کارهای آتی

در این مقاله یک سلول‌بندی مناسب برای فضای پیکربندی ارائه کردیم و اثبات کردیم که برای تعیین مجموعه بیشینه هر سلول کافی است تنها فاصله بحرانی همان سلول را مشخص کنیم. هنگام در نظر گرفتن گذار بین هر دو سلول با وجود نادقیقی یک یا چند رأس به ازای هر رأس با یک مجموعه نقاط (ناحیه نادقیق) مواجه هستیم که از بین آن‌ها یک نقطه حساس که باعث می‌شود مجموعه بیشینه محبوس کننده به قطعیت برسد را انتخاب می‌کنیم. به‌این‌ترتیب امکان محاسبه فاصله بحرانی فراهم شد. همچنین با ساخت گراف اتصال و انتشار کم‌ترین فاصله جدایی از رأس آزاد تا هر رأس گراف، فاصله بحرانی هر سلول را یافته و ناحیه‌هایی که با وجود نادقیقی ممکن است قسمتی از چندضلعی باشند را از مجموعه جواب نهایی حذف کرده و مجموعه محبوس کننده متناظر آن را گزارش می‌کنیم. علاوه بر این نیز پیچیدگی ترکیبیاتی سلول‌ها و مجموعه‌های محبوس کننده را محاسبه کردیم که برای تمامی مجموعه‌های محبوس کننده یک چندضلعی در زمان $O(n^2logn)$ انجام می‌شود. در پایان مقاله نسخه پرس‌وجو مسئله را مطرح کردیم و نشان دادیم که با استفاده از ساختمان داده گراف اتصال، می‌توان در زمان $O(logn)$ به آن پاسخ داد.

در هرکدام از این مسائل، با نادقیق شدن چندضلعی به گونه‌های مختلف مانند نادقیق شدن رئوس یا اضلاع یا حتی نادقیقی اجزای چنگ باعث می‌شود مسئله چالش‌برانگیزتر شود. در شرایط نادقیقی هدف می‌تواند یک یا هر سه مورد زیر باشد:

- شناسایی تمام محدوده‌هایی که با نادقیق شدن رئوس نیز هیچ‌گاه چندضلعی را محبوس نمی‌کنند،
- شناسایی تمام محدوده‌هایی که محبوس شدن چندضلعی در آن ناحیه‌ها محتمل است،
- شناسایی تمام محدوده‌هایی که با وجود نادقیق شدن رئوس، همچنان چندضلعی موردنظر را محبوس می‌کنند.

یکی از اهداف آتی ما بهبود الگوریتم ارائه‌شده و تعمیم آن به ابعاد بالاتر است. همچنین بررسی مسئله برای بهبود زمان اجرا و ارائه الگوریتمی که حساس به‌اندازه خروجی باشد نیز می‌تواند از مسائل باز برای پژوهش‌های آینده باشد.

۶- مراجع

[1] M. Vahedi, *Caging polygons with two and three fingers*, Dept of Information and Computing Science, Utrecht University, 2009.

[2] W. Kuperberg, "Problems on polytopes and convex sets," *DIMACS Workshop on polytopes*, pp. 584-589 1990.

[3] A. S. Besicovitch, "A net to hold a sphere," *Mathematical Gazette,* vol. 41, no. 336, pp. 106-107, 1957.

[4] GC. Shephard, "A sphere in a crate," *Journal of the London Mathematical Society*, vol. 1, no. 1, pp. 433-434, 1965.

[5] P. Pipattanasomporn, and A. Sudsang. "Two-finger caging of concave polygon." *In Proceedings 2006 IEEE International Conference on Robotics and Automation*, ICRA, pp. 2137-2142, 2006.

[6] M. Vahedi, and A. F. van der Stappen, "Caging polygons with two and three fingers," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1308-1324, 2008.

[7] P. Pipattanasomporn, and A. Sudsang, "Object caging under imperfect shape knowledge," *In 2010 IEEE International Conference on Robotics and Automation*, pp. 2683-2688, 2010.

[8] A. A. Requicha, "Toward a theory of geometric tolerancing," *The International Journal of Robotics Research*, vol. 2, no. 4, pp. 45-60, 1983.

[9] H. B. Voelcker, "A current perspective on tolerancing and metrology," *International Forum on Dimensional Tolerancing and Metrology, New York*, vol. 27, pp. 49-60. 1993.

[10] B. R. Donald, "Error detection and recovery in robotics," *Berlin: Springer,* vol. 336, 1989.

[11] S. M. LaValle, and S. A. Hutchinson, "An objective-based framework for motion planning under sensing and control uncertainties," *The International Journal of Robotics Research*, vol. 17, no. 1, pp. 19-42, 1998.

[12] M. Dogar, and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and systems*, volII. 1, 2011.

[13] M. Löffler, "Data imprecision in computational geometry," PhD diss, Utrecht Univesity, 2009.

[14] M. Davoodi, and A. Mohades, "Data imprecision under λ-geometry model: Range searching problem," *Scientia Iranica*, vol. 20, no. 3, pp. 663-669, 2013.

[15] M. Davoodi, A. Mohades, F. Sheikhi, and P. Khanteimouri, "Data imprecision under λ-geometry model," *Information Sciences,* vol. 295, pp. 126-144, 2015.

[16] F. Panahi, M Davoodi, and A. F. van der Stappen, "Orienting parts with shape variation," *IEEE Transactions on Automation Science and Engineering,* vol. 14, no. 2, pp 1109-1118, 2015.

[17] J. Hershberger, and S. Suri. "A pedestrian approach to ray shooting: Shoot a ray, take a walk," *Journal of Algorithms*, vol. 18, no. 3, pp 403-431, 1995.

منصور داودی منفرد تحصیلات کارشناسی خود را در زمینه علوم کامپیوتر در دانشگاه ولیعصر رفسنجان از سال ۱۳۸۱ آغاز نمود. ایشان تحصیلات کارشناسی ارشد و دکتری خود را در همان زمینه علوم کامپیوتر به ترتیب در سال‌های ۱۳۸۷ و ۱۳۹۱ در دانشگاه امیرکبیر تهران به اتمام رساند. در حال حاضر به‌عنوان استادیار دانشکده علوم رایانه و فناوری اطلاعات دانشگاه تحصیلات تکمیلی علوم پایه زنجان فعالیت می‌نماید. زمینه تخصصی ایشان پیچیدگی محاسباتی و الگوریتم‌های بهینه‌سازی، رباتیک و هندسه محاسباتی است.

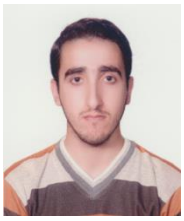
آدرس پست الکترونیک ایشان عبارت است از:

mdmonfared@iasbs.ac.ir

اسماعیل دلفراز پهلوانلو تحصیلات کارشناسی خود را در زمینه علوم کامپیوتر در دانشگاه شهید باهنر کرمان از سال ۱۳۸۹ آغاز نمود و در سال ۱۳۹۳ به پایان رساند. ایشان تحصیلات کارشناسی ارشد خود را در همان زمینه علوم کامپیوتر در سال ۱۳۹۳ شروع کرد و در سال ۱۳۹۶ در دانشگاه تحصیلات تکمیلی علوم پایه زنجان به اتمام رساند. . از توانایی‌های ایشان در زمینه برنامه‌نویسی می‌توان به تسلط در زبان برنامه‌نویسی python, C، C++، C#، SQL و MATLAB اشاره کرد. همچنین زمینه مورد پژوهش و علاقه ایشان الگوریتم‌های تقریبی و تصادفی، کمینه‌سازی تحویل در شبکه‌های بی‌سیم، نظریه گراف، رباتیک و هندسه محاسباتی است. در حال حاضر به‌عنوان دانشجوی دکترا گروه علوم کامپیوتر دانشگاه گرانساسو ایتالیا مشغول به تحصیل است.

آدرس پست الکترونیک ایشان عبارت است از:

Esmaiel.delfaraz@gmail.com



سید مقداد نبوی لاریمی تحصیلات کارشناسی خود را در زمینه علوم کامپیوتر در دانشگاه شهید باهنر کرمان از سال ۱۳۸۹ آغاز نمود و در سال ۱۳۹۴ به پایان رساند. ایشان تحصیلات کارشناسی ارشد خود را در همان زمینه علوم کامپیوتر در سال ۱۳۹۴ شروع کرد و در سال ۱۳۹۷ در دانشگاه تحصیلات تکمیلی علوم پایه زنجان به اتمام رساند. از توانایی‌های ایشان در زمینه برنامه‌نویسی می‌توان به تسلط در زبان برنامه‌نویسی C، C++، C#، SQL و MATLAB اشاره کرد. همچنین زمینه مورد پژوهش و علاقه ایشان الگوریتم‌های تقریبی و تصادفی، کمینه‌سازی تحویل در شبکه‌های بی‌سیم، نظریه گراف، رباتیک و هندسه محاسباتی است.

آدرس پست الکترونیک ایشان عبارت است از:

smnabavi@iasbs.ac.ir

پیوست

لم ۷,۱ اگر x_{min} کمینه محلی P_{ij} باشد، آنگاه $\min_{x\in P_{ij}} Cd(x) = Cd(x_{min})$.

اثبات: برای اثبات لازم است تا برای هر دو حالت محبوس‌کنندگی و غیرمحبوس‌کنندگی کمینه محلی، درستی این لم را نشان دهیم.

حالت ۱: اگر x_{min} محبوس کننده نباشد، طبق لم ۴,۴ تمام موقعیت‌های سلول P_{ij}غیرمحبوس‌کننده هستند که در این صورت برای هر $x \in P_{ij}$ داریم$= Cd(x)$ $d(x)$ است و نتیجه می‌گیریم:

$$\min_{x\in P_{ij}} Cd(x) = \min_{x\in P_{ij}} d(x) = d(x_{min})$$

حالت ۲: اگر x_{min} محبوس کننده باشد، آنگاه از میان موقعیت‌های موجود در آن سلول، برخی محبوس کننده هستند و برخی نیز غیر محبوس کننده نیستند. برای موقعیت‌هایی که محبوس کننده هستند مطابق قضیه ۴,۵، x و x_{min} در مجموعه محبوس کننده یکسان قرار می‌گیرند که $Cd(x) = d(x_{min})$ است. برای موقعیت‌هایی که محبوس کننده نیستند می‌توان از برهان خلف برای اثبات استفاده کرد. به این معنا که فرض کنید $Cd(x) \le Cd(x_{min})$ باشد. طبق لم ۴,۴ مسیری از x با فاصله جدایی کم‌تر از $Cd(x)$ به x_{min} وجود دارد. حال معکوس این مسیر را در نظر می‌گیریم که مسیری است از x_{min} به x با فاصله جدایی کم‌تر از $Cd(x)$. بنابراین می‌توان از موقعیت محبوس کننده x_{min} با مسیری که فاصله آن کم‌تر از $Cd(x_{min})$است به x رسید و از مشاهده ۴,۲ نتیجه می‌گیریم x محبوس کننده است که این با فرض اولیه تناقض دارد پس $Cd(x) > Cd(x_{min})$ است.

لم ۷,۲ اگر $x_1 \in B(P_1, P_2)$ و $x_2 \in B(P_2, P_3)$ باشد، آنگاه در میان مسیرهای موجود از x_1 به x_2 کم‌ترین فاصله جدایی برابر با $max\{d(x_1), d(x_2)\}$ است.

اثبات: با استفاده از لم ۴,۴ می‌توان این لم را اثبات کرد. با توجه به فاصله دو نقطه x_1 و x_2 در ابتدا و انتهای مسیر که $d(x_1)$ و $d(x_2)$ است، فاصله جدایی هر مسیر (طبق تعریف آن) از x_1 به x_2 بزرگ‌تر مساوی $max\{d(x_1), d(x_2)\}$ است. حال کمینه محلی x_{min} سلول P_2 را در نظر بگیرید که مطابق لم ۴,۴، زیرمسیری از x_1 به x_{min} با فاصله جدایی نزولی $d(x_1)$ و زیرمسیری از x_{min} به x_2 با فاصله جدایی صعودی $d(x_2)$ وجود دارد. به‌این‌ترتیب نتیجه می‌گیریم مسیری از x_1 به x_2با فاصله جدایی $max\{d(x_1), d(x_2)\}$ وجود دارد. ازاین‌رو حکم اثبات می‌شود.

مشاهده ۷,۳ نقطه حساس s، روی محیط نادقیق رأس v یعنی V قرار دارد.

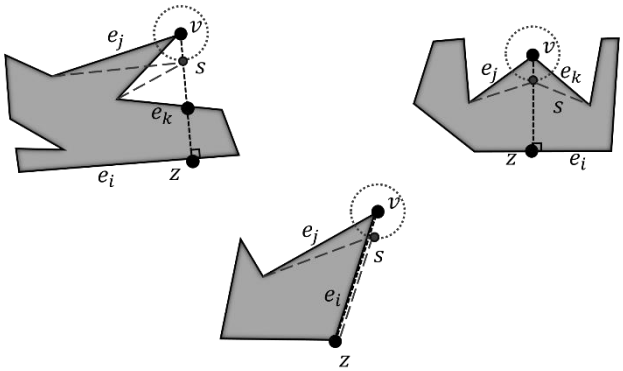
اثبات: فرض کنید s روی محیط V قرار نداشته باشد. به‌این‌ترتیب نقطه‌ای مانند s' روی V وجود دارد که $|s'-z|$ کم‌ترین مقدار را دارد و طول پاره‌خط واصل بین s' و x، مقدار $|s'-z|$ را نشان می‌دهد و این پاره‌خط را با $s'x$ نمایش می‌دهد. واضح است $s'x$ از روی محیط V عبور می‌کند و در نقطه‌ای مانند s با آن برخورد می‌کند که $|s-z|$ مقدار کمتری نسبت به $|s'-z|$ دارد. بنابراین فرض خلف باطل است و نقطه s روی محیط V قرار دارد.

قضیه ۷,۴ فاصله بحرانی برای سلول دلخواه در چندضلعی داده شده P که دارای یک رأس نادقیق است، برابر با $|s-z|$ است.

اثبات: فرض کنیم که رأس نادقیق رأس $v \in V$ (همان مرکز ناحیه نادقیق V) باشد. در این صورت مطابق لم ۵,۳ حالت‌های زیر را داریم:

۱- رأس نادقیق مورد نظر محدب باشد، (شکل ۱۲).

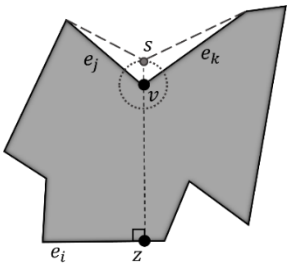
در این حالت برای یافتن مجموعه بیشینه قطعی باید به ازای هر گذار به دنبال کم‌ترین مقدار t_{ijk} باشیم، یعنی کم‌ترین مقدار $|v-z|$. طبق شکل ۱۲ به‌وضوح با انتخاب نقطه s روی محیط محدوده دایره نادقیق به کم‌ترین مقدار $|v-z|$ می رسیم، یعنی همان $|s-z|$.



شکل ۱۲- نقطه s روی محیط محدوده دیسک نادقیق، مناسب‌ترین مکان برای دقیق کردن رأس محدب به ازای ضلع e_i است.

۲- رأس نادقیق مورد نظر مقعر باشد، (شکل ۱۳).

در این حالت برای یافتن مجموعه بیشینه قطعی باید به ازای هر گذار به دنبال بیش‌ترین مقدار t_{ijk} باشیم، یعنی بیش‌ترین مقدار $|v-z|$. طبق شکل ۱۳ به وضوح با انتخاب نقطه s روی محیط محدوده دایره نادقیق، به بیش‌ترین مقدار $|v-z|$ می‌رسیم، یعنی همان $|s-z|$.



شکل ۱۳- نقطه s روی محیط محدوده دیسک نادقیق، مناسب‌ترین مکان برای دقیق کردن رأس مقعر به ازای ضلع e_i است.

[1] Grasp
[2] Firm grasp
[3] Loose grasp
[4] Uncertainty
[5] Imprecise
[6] Laser range scanner
[7] Computer-aided design
[8] Tolerance
[9] Actuator
[10] ε-geometry
[11] Region-based models
[12] Linear parametric geometric uncertainty model
[13] λ-geometry
[14] Pushing with a frictionless jaw
[15] Instance
[16] Workspace
[17] Configuration space
[18] Trajectory
[19] Escape trajectory
[20] Separation distance
[21] Critical distance
[22] Propagation
[23] Connected
[24] Maximal caging set
[25] Least upper-bound distance
[26] Local minimum
[27] Squeezed trajectory
[28] Transition
[29] Ray shooting
[30] Connectivity graph
[31] Sensitive point

The CSI Journal on Computing Science and Information Technology

The CSI Journal on
Computing Science and
Information Technology
Vol. 17, No.2, 2020

Caging Imprecise Polygons with Two Fingers

Mansoor Davoodi Monfared, Esmaeil Delfaraz Palevanlo, Seyyed Meghdad Nabavi Larimi

Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences, Zanjan, Iran

---

Abstract

The problem of caging an arbitrary object is one of the major issues in the field of robotics. All of algorithms in this field assume that the object is precise, but due to errors during the computation and construction of objects (parts), it is possible, the object's coordination to be imprecise. Our purpose is to provide an appropriate algorithm for caging the polygon, even if the polygon is partially imprecise. In this paper, we present an algorithm for finding all arrangements of two fingers, which definitely cage an imprecise part. An imprecise part is a polygon with imprecise vertices that are shown as a disc with radius unit. At the end, all of the caging arrangements to be reported $O(n^2 \log n)$ time and $O(n^2)$ space, where n is the number of vertices in the parts..

Keywords: Algorithm; Uncertainty; Caging; Imprecision.