دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد

گرایش مهندسی نرم‌افزار

عنوان:

الگوریتم‌های تقریبی برای بهینه‌سازی هندسی در مدل پنجره‌های لغزان

نگارش:

نوید صالح‌نمدی

استاد راهنما:

حمید ضرابی‌زاده

شهریور ۱۳۹۶

بسم الله الرحمن الرحیم

به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

پایان‌نامه‌ی کارشناسی ارشد

عنوان: الگوریتم‌های تقریبی برای بهینه‌سازی هندسی در مدل پنجره‌های لغزان

نگارش: نوید صالح‌نمدی

کمیته‌ی ممتحنین

استاد راهنما: حمید ضرابی‌زاده امضاء:

استاد مشاور: محمد قدسی امضاء:

استاد مدعو: نامشخص امضاء:

تاریخ:

چکیده

مسائل بهینه‌سازی هندسی از دیرباز در علوم کامپیوتر مورد بررسی قرار گرفته‌اند و کاربر بسیار زیادی در حوزه‌های مختلف دارند. از مهم‌ترین این مسائل می‌توان به مسائل کوچک‌ترین کره‌ی محیطی و kـ مرکز اشاره کرد. با افزایش سرعت تولید حجم داده پژوهش‌های اخیر روی مدل‌های داده‌های حجیم مانند جویبار داده و پنجره‌ی لغزان متمرکز شده است. تمرکز اصلی این پایان‌نامه روی حل مسائل بهینه‌سازی هندسی (به طور خاص kـ مرکز هندسی) در مدل پنجره‌ی لغزان است. در مدل پنجره‌ی لغزان به دنبال پاسخ‌گویی به پرسش روی N نقطه‌ی آخر ورودی هستیم. از مشکلات این روش می‌توان به عدم امکان نگه‌داری تمام نقاط اشاره کرد.

در این پایان‌نامه، مسائل کوچک‌ترین کره‌ی محیطی، ۲ـ مرکز و k—مرکز هندسی در مدل پنجره‌ی لغزان مورد بررسی قرار می‌گیرد. برای مسئله‌ی کوچک‌ترین کره‌ی محیطی در فضای dـ بعدی الگوریتم $(1+\varepsilon)$ـ تقریب با حافظه‌ی $\mathcal{O}(\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ و زمان پردازش نقاط $\mathcal{O}((d+1)!\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ ارائه می‌دهیم که پارامتر R برابر نسبت بیش‌ترین پاسخ به کوچک‌ترین فاصله‌ی بین دو نقطه است. این الگوریتم اولین الگوریتم ارائه شده برای فضای بیش از دو بعد است. سپس برای مسئله‌ی ۲ـ مرکز دوبعدی الگوریتم $(1+\varepsilon)$ـ تقریب ارائه می‌شود که الگوریتم قبلی با ضریب تقریب $4+\varepsilon$ را بهبود می‌دهد. در پایان مسئله‌ی kـ مرکز به وسیله‌ی یک الگوریتم $(2+\varepsilon)$ـ تقریب در مدل پنجره‌ی لغزان تقریب زده می‌شود که الگوریتم قبلی با ضریب تقریب $6+\varepsilon$ را بهبود می‌دهد. حافظه‌ی مصرفی تمام الگوریتم‌ها از مرتبه‌ی چندجمله‌ای نسبت به d و $\log R$ و $\frac{1}{\varepsilon^d}$ است. لازم به ذکر است که کران پایین $\log R$ برای حافظه‌ی الگوریتم‌هایی از پنجره‌ی لغزان ثابت شده است.

کلیدواژه‌ها: بهینه‌سازی هندسی، پنجره‌ی لغزان، الگوریتم‌های تقریبی، داده‌های حجیم

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فصل ۱

مقدمه

مسائل بهینه‌سازی هندسی [۱] با این که سابقه‌ی طولانی در علوم محاسباتی دارند اما هنوز کهنه نشده‌اند و با معرفی هر مدل نگه‌داری داده‌ی جدید، نیاز به بررسی و تحقیق روی آن‌ها دوباره احساس می‌شود. از طرف دیگر با افزایش کاربردهای مکان‌ـ‌محور [۲] در صنعت لزوم بهبود این الگوریتم‌ها ضروری‌تر به نظر می‌رسد. مسائلی از قبیل پوش محدب [۳]، پیداکردن قطر [۴] یا عرض [۵]، کوچک‌ترین کره‌ی محیطی [۶] یا شناسایی kـ‌مرکز [۷] همگی در دسته‌ی مسائل بهینه‌سازی هندسی قرار می‌گیرند. مسئله‌ی kـ‌مرکز یک رویکرد برای حل مسئله‌ی خوشه‌بندی است که خود یکی از مهم‌ترین مسائل داده‌کاوی [۸] به شمار می‌آید.

از طرف دیگر با افزایش حجم اطلاعات در مسائل دنیای واقعی، مدل‌های نگه‌داری داده‌ای معرفی شدند که برای محاسبات محدودیت‌هایی در اندازه‌ی حافظه و نحوه‌ی دسترسی به آن را اعمال می‌کنند. به عنوان مثال مدل جویبار داده [۹] و پنجره‌ی لغزان [۱۰] را می‌توان نام برد. در اولی همان‌طور که از نامش بر می‌آید داده‌ها در یک جویبار یک به یک وارد می‌شوند و تنها یک (یا تعداد مشخص و محدودی)

[۱] Geometry Optimization
[۲] Location-Based
[۳] Covex Hull
[۴] Diameter
[۵] Width
[۶] Minimum Enclosing Ball
[۷] k-centers
[۸] Data Mining
[۹] Data Stream
[۱۰] Sliding Window

بار می‌توانیم داده‌ها را ببینیم و حافظه‌ای که می‌توانیم استفاده کنیم از مرتبه‌ی زیرخطی [۱۱] است. مدل پنجره‌ی لغزان از جویبار داده مشتق شده است با این ویژگی که می‌خواهد محاسبات تنها داده‌هایی را لحاظ کند که اخیرا وارد شده‌اند (مثلا N داده‌ی آخر). با توجه به تعریف این مدل‌های داده می‌توان به این نتیجه رسید که برای حل مسائل کلاسیک در این مدل‌ها نیاز به نگاه از زاویه‌ای دیگر است. از طرف دیگر به خاطر ذات این مدل‌ها که حافظه‌ی زیرخطی دارند بیش‌تر مسائل را نمی‌توان به صورت دقیق در این مدل حل کرد و معمولا به صورت تقریبی حل می‌شوند.

در این پژوهش، با تمرکز روی مسائل بهینه‌سازی هندسی در مدل پنجره‌ی لغزان، مسئله‌ی kـ مرکز در فضای هندسی با ابعاد کوچک مورد بررسی قرار گرفته است که بهبود قابل توجهی در ضریب تقریب این مسئله به دست آمده است. در بخش بعدی، تعریف رسمی[۱۲] از مسائلی که در این پایان‌نامه مورد بررسی قرار می‌گیرند را بیان نموده و در مورد هر کدام توضیح مختصری می‌دهیم.

۱ـ۱ تعریف مسئله

تعریف دقیق‌تر مسئله‌ی kـ مرکز در زیر آمده است:

مسئله‌ی ۱ـ۱ (kـ مرکز) *مجموعه‌ی P شامل تعدادی نقطه است. فاصله‌ی این نقاط به وسیله‌ی تابع فاصله‌ی dis، که از نامساوی مثلثی[۱۳] پیروی می‌کند به دست می‌آید. زیرمجموعه‌ی k عضوی $T \subseteq P$ را طوری انتخاب کنید که عبارت زیر را کمینه کند:*

$$\max_{p \in P}\{\min_{t \in T} dis(p,t)\} \tag{۱ـ۱}$$

در صورتی که تابع فاصله‌ی dis در فضای اقلیدسی باشد به این مسئله kـ مرکز هندسی می‌گوییم. یک تفاوت ساختاری فضای متریک با هندسی در این است که در فضای هندسی می‌توان فاصله‌ی نقاطی که بین نقاط ورودی نیست را با دیگر نقاط به دست آورد اما در فضای متریک تنها فاصله‌ی هر دو نقطه‌ای که در ورودی آمده است را داریم.

مسئله‌ی kـ مرکز در مدل جویبار داده توجه زیادی را به خود جلب کرده است و مورد بررسی های زیادی قرار گرفته است. در این مدل ابتدا تمام نقاط در دسترس نیستند، بلکه یکی پس از دیگری وارد

[۱۱] Sublinear
[۱۲] Formal
[۱۳] Triangle Inequality

می‌شوند. هم‌چنین ترتیب ورود نقاط نامشخص است. علاوه بر این محدودیت مدل جویبار داده دارای محدودیت حافظه است، به‌طوری‌که امکان نگه‌داری تمام نقاط در حافظه وجود ندارد و معمولا باید مرتبه‌ی حافظه‌ای کم‌تر از مرتبه حافظه‌ی **خطی**[۱۴] (یا همان زیرخطی) متناسب با تعداد نقاط استفاده نمود.

مدلی که ما در این پژوهش بر روی آن تمرکز داریم مدل پنجره‌ی لغزان است که از مدل جویبار داده تک‌گذره[۱۵] [۲] مشتق شده است. یعنی تنها یک بار می‌توان از ابتدا تا انتهای داده‌ها را بررسی کرد و پس از عبور از یک داده، اگر آن داده در حافظه ذخیره نشده باشد، دیگر نمی‌توانیم به آن دسترسی داشته باشیم. علاوه بر این، در هر لحظه باید بتوان به پرسمان (برای N نقطه‌ی اخیر) پاسخ داد.

یکی از دغدغه‌هایی که در مسائل جویبار داده و پنجره‌ی لغزان وجود دارد، عدم امکان دسترسی به تمام نقاط است. به عبارت دیگر نه می‌توانیم به تمام داده‌هایی که تا الان آمده‌اند دسترسی داشته باشیم و نه می‌توانیم راجع به داده‌هایی که هنوز وارد نشدند نظری بدهیم. در مدل پنجره‌ی لغزان حتی باید به این نکته توجه کنیم که داده‌هایی که در حافظه ذخیره کرده‌ایم ممکن است منقضی بشوند (یا از پنجره خارج شوند). یعنی نمی‌توانیم به هیچ کدام از نقاطی که تا به حال ذخیره کردیم اعتماد کنیم، چون تا پایان نمی‌توانند معتبر باشند.

تعریف ۱ـ۱ *L_p **متریک** به ازای دو نقطه‌ی dـ بعدی s و q، فاصله‌ی s و q در متریک L_p برابر با*

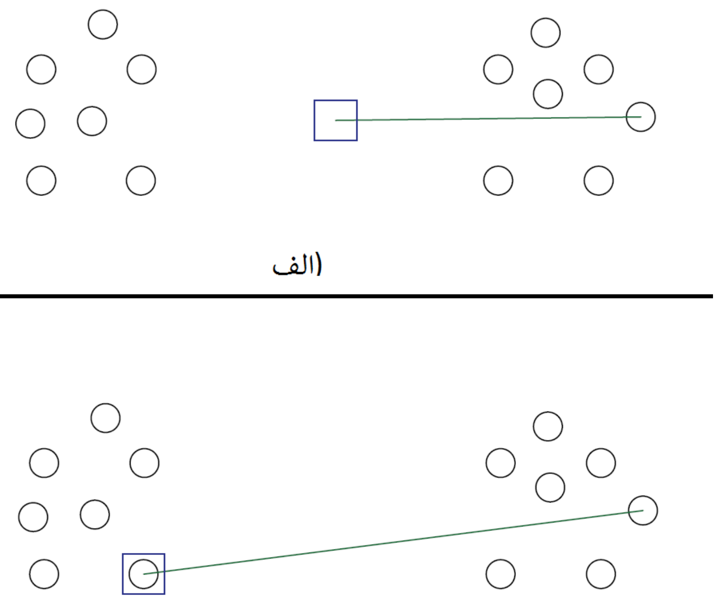
$$d(p,q)=\sqrt[p]{\sum_{i=۱}^{d}(s_i-q_i)^p}$$

است. که s_i و q_i برابر مختصات بعد iام نقاط s و q است.

لازم به ذکر است که $L_۲$ متریک همان تابع فاصله در فضای اقلیدسی است. مسئله‌ی kـ مرکز، معمولاً تنها برای L_pـ متریک مطرح می‌شود (زیرا نیاز به دانستن فاصله‌ی هر دو نقطه‌ای است). در حالت دیگر باید مجموعه‌ای از تمام نقاط فضا به انضمام فاصله‌هایشان را داشته باشیم. زیرا مرکز دسته‌ها ممکن است در هر نقطه از فضا قرار بگیرد و ما نیاز داریم که فاصله‌ی آن را از تمام نقاط بدانیم. نمونه‌ای از مسئله‌ی ۱ـ مرکز در حالت پیوسته و گسسته، در شکل ۱ـ۱ نشان داده شده است.

تعریف دقیق گونه‌ی پنجره‌ی لغزان مسئله‌ی kـ مرکز، در زیر آمده است:

[۱۴] Linear
[۱۵] Single pass



الف)

ب)

شکل ۱ـ۱: الف) نمونه‌ای ازمسئله‌ی ۱ـمرکز در حالت پیوسته، ب) نمونه‌ای از مسئله‌ی ۱ـمرکز در حالت گسسته. همان‌طور که مشاهده می‌شود در حالت ب پاسخ بسیار بیش‌تر از حالت الف است.

مسئله‌ی ۱ـ۲ ***(kـ مرکز در مدل پنجره‌ی لغزان)*** *دنباله‌ی U از نقاط فضای dـ بعدی داده شده است. P را N نقطه‌ی آخر U می‌نامیم. زیرمجموعه $S \subseteq P$ با اندازه‌ی k را انتخاب کنید به‌طوری‌که عبارت زیر کمینه شود:*

$$\max_{u \in P}\{\min_{s \in S} L_p(u,s)\} \tag{۱ـ۲}$$

با مطالعه‌ی پژوهش‌های انجام‌شده در حوزه‌های الگوریتم‌های تقریبی بهینه‌سازی هندسی و مدل‌های داده‌حجیم (مثل جویبار داده و پنجره‌ی لغزان) تصمیم گرفتیم تمرکز این پژوهش را روی kـ مرکز هندسی در ابعاد پایین بگذاریم.

۱ـ۲ اهمیت موضوع

به علت افزایش سریع حجم و تولید داده‌ها دیگر امکان پردازش و دسترسی آزادانه به تمام داده‌ها وجود ندارد. به همین دلیل مسئله‌های مدل جویبار داده در سالیان اخیر بسیار مورد توجه قرار گرفته‌اند. اگر از زاویه‌ی دیگری به سرعت بالای تولید اطلاعات نگاه کنیم، متوجه می‌شویم که نه تنها دسترسی دلخواه به تمامی داده‌ها نداریم بلکه علاقه‌ای نیز به داده‌های بسیار قدیمی وجود ندارد. برای روشن‌تر شدن این حوزه دو مثال از دنیای واقعی می‌زنیم.

مثال ۱ـ۱ *یک مسیریاب [۱۶] را در نظر بگیرید که بسته‌های [۱۷] شبکه را از گره [۱۸] مبدا می‌گیرد و به گره مقصد تحویل می‌دهد. به حجم ارتباطات شبکه روز به روز افزوده می‌شود و امکان نگه‌داری (حتی داده‌های ضروری) بسته‌ها وجود ندارد. یک مسئله در مسیریاب‌ها شناسایی پربازدیدترین مقصدها است. مثلا می‌خواهیم بدانیم از شبکه‌ی داخلی sharif.ir چه آدرسی بیش‌ترین بازدید را داشته است. اگر دامنه‌ی محاسبات را تمامی بسته‌هایی که از مسیریاب اصلی دانشگاه شریف، از بدو شروع به کار آن، گذشته است در نظر بگیریم به پربازدیدترین آدرس در تمام سالیانی که این مسیریاب کار می‌کرده خواهیم رسید. اما این که بدانیم در هفته یا ماه گذشته چه آدرسی بیش‌ترین بازدید را داشته است بسیار ارزشمندتر است چرا که داده‌های سالیان گذشته تاثیری روی کاربرد فعلی مسیریاب نخواهد داشت.*

مثال ۱ـ۲ *شبکه‌ی اجتماعی اینستاگرام [۱۹] قابلیتی به نام داستان [۲۰] دارد که هر کاربر می‌تواند عکس یا فیلمی کوتاه را به صورت داستان به دیگر دنبال‌کنندگانش نشان دهد. هر داستان تا ۲۴ ساعت به دیگر کاربران نمایش داده می‌شود و پس از آن حذف می‌شود. هر داستان می‌تواند برچسب مکان داشته باشد که نشان‌دهنده‌ی جایی است که آن داستان رخ داده است. اینستاگرام به هر کاربر علاوه بر داستان‌های افرادی که دنبال می‌کنند داستان‌هایی که برچسب مکان آن‌ها نزدیک به مکان فعلی کاربر هست را نیز نمایش می‌دهد.*

در مثال ۱ـ۱ متوجه حجم بالای داده‌ها و عدم امکان دسترسی آزاد به تمامی آن‌ها می‌شویم. علاوه بر این در مثال ۱ـ۲ دیده می‌شود تمرکز برنامه‌ها روی داده‌های اخیر (مثلا ۲۴ ساعت گذشته) است و حتی

[۱۶] Router
[۱۷] Packet
[۱۸] Node
[۱۹] Instagram
[۲۰] Story

امکان دسترسی به داده‌های قدیمی نیز وجود ندارد. این‌جا ضرورت مدل داده‌ای احساس می‌شود که نه تنها بتواند دسترسی به این حجم بزرگ اطلاعات را محدود و کنترل کند بلکه مسئله را به سمت استفاده از داده‌های اخیر متمرکز کند. مدل پنجره‌ی لغزان با دارابودن خواص جویبار داده محدودیت‌هایی را اضافه کرده است که باعث تمرکز بر داده‌های جدید شده است. به همین دلیل در این پژوهش مسئله‌ی اصلی بر روی این مدل داده معطوف شده است. در مثال ۱ـ۱ اگر دامنه‌ی محاسبات را تمامی بسته‌ها در نظر بگیریم می‌توان از مدل جویبار داده استفاده کرد. اما اگر مجموعه‌ی مورد نظر را به بسته‌های یک ماه اخیر کاهش دهیم یا در مثال ۱ـ۲ بهتر است از مدل پنجره‌ی لغزان استفاده کرد که علاوه بر این که محدودیت دسترسی به داده‌ها را ارضا می‌کند بلکه تمرکزش روی داده‌های اخیر است.

در حوزه‌ی بهینه‌سازی هندسی، مسئله‌ی kـ مرکز و گونه‌های آن از جمله کاربردی‌ترین و متداول‌ترین مسائل به شمار می‌آیند. کاربرد این مسئله در مباحث داده‌کاوی بسیار جا افتاده است و یکی از رایج‌ترین الگوریتم‌های مورد استفاده برای خوشه‌بندی محسوب می‌شود. از طرف دیگر کاربردهای زیادی در دنیای واقعی دارد. به عنوان مثال فرض کنید در مثال ۱ـ۲، اینستاگرام می‌خواهد به جای نمایش کل داستان‌های شهر به هر کاربر، داستان‌هایی را نشان دهد که در فاصله‌ی نزدیک‌تری به وی هستند. یک روش برای مدل‌سازی این مسئله استفاده از مسئله‌ی kـ مرکز است تا کاربران را به دسته‌هایی تقسیم کند که فاصله‌یشان خیلی کم است. به طور دقیق‌تر به خاطر ماهیت داستان (که پس از ۲۴ ساعت از بین می‌رود) بهتر است مسئله‌ی kـ مرکز در فضای اقلیدسی دوبعدی پنجره‌ی لغزان به عنوان مدل در نظر گرفته شود.

به دلایل بالا این پژوهش روی مسائل بهینه‌سازی هندسی (به طور خاص kـ مرکز هندسی) در مدل پنجره‌ی لغزان متمرکز شده است.

۱ـ۳ ادبیات موضوع

مسئله‌ی kـ مرکز یکی از مهم‌ترین مسائل بهینه‌سازی هندسی است که در خانواده‌ی مسائل NPـ سخت قرار دارد. به شرط $P \neq NP$، هیچ الگوریتم دقیقی برای حل این مسئله در زمان چندجمله‌ای حتی در مدل ایستا هم وجود ندارد. در نتیجه در بیش‌تر مواقع مجبور هستیم از الگوریتم‌های تقریبی [۲۱] استفاده کنیم. علاوه بر این مسئله، مسائل ساده‌تری مانند محاسبه‌ی قطر یا عرض و یا کوچک‌ترین کره‌ی محیطی

[۲۱] Approximation algorithm

وجود دارد. این مسائل از دیرباز در مدل ایستا بسیار مورد بررسی قرار گرفته‌اند. اکنون این مسائل بیش‌تر در مدل‌های داده حجیم مورد بررسی قرار می‌گیرند.

برای مسئله‌ی kـ مرکز، الگوریتم‌های تقریبی معروفی وجود دارد که به یکی از ساده‌ترین آن‌ها اشاره می‌کنیم. این الگوریتم از رویکرد حریصانه [۲۲] استفاده می‌کند. ابتدا یک نقطه‌ی دلخواه را به عنوان مرکز در نظر می‌گیرد سپس در هر مرحله نقطه‌ای را به عنوان مرکز انتخاب می‌کند که از بقیه‌ی مراکز بیش‌ترین فاصله را داشته باشد.[۳]. این الگوریتم، الگوریتم تقریبی با ضریب تقریب ۲ ارائه می‌دهد. همچنین کران پایین تقریب این مسئله مشخص شده‌است. بهتر از ضریب تقریب ۲ برای مسئله‌ی kـ مرکز در حالت کلی نمی‌توان الگوریتمی یافت به شرط آن که $P \neq NP$ باشد.

برای مسئله‌ی kـ مرکز در حالت جویبار داده برای ابعاد بالا، بهترین الگوریتم موجود ضریب تقریب $2+\epsilon$ دارد [۴، ۵، ۶] و ثابت می‌شود الگوریتمی با ضریب تقریب بهتر از ۲ نمی‌توان ارائه داد.

برای مسئله‌ی kـ مرکز مدل پنجره‌ی لغزان نیز، بهترین الگوریتم ارائه شده، الگوریتمی با ضریب تقریب ۶ است که در فضای متریک ارائه شده است [۷].

برای kهای کوچک به خصوص، $k = 1, 2$، الگوریتم‌های بهتری ارائه شده است. بهترین الگوریتم ارائه شده برای مسئله‌ی ۱ ـ مرکز در حالت جویبار داده برای ابعاد بالا، دارای ضریب تقریب ۱/۲۲ است و کران پایین $\frac{1+\sqrt{2}}{2}$ نیز برای این مسئله اثبات شده است [۸، ۹]. برای مسئله ۲ ـ مرکز در مدل پنجره‌ی لغزان فضای متریک، راه‌حلی با ضریب تقریب ۴ ارائه شده است [۷]. برای مسئله‌ی ۱ ـ مرکز مدل پنجره‌ی لغزان در فضای دوبعدی، الگوریتمی با ضریب تقریب $1+\epsilon$ است که به وسیله‌ی ارائه‌ی یک ϵـ هسته به دست آمده است. [۱۰].

۱ ـ ۴ اهداف تحقیق

در این پایان‌نامه سعی شده است تا مسائلی از حوزه‌ی بهینه‌سازی هندسی (با تمرکز روی kـ مرکز) شناسایی شود تا در مدل پنجره‌ی لغزان به طور کارآمدی حل شوند و اگر نتایج قبلی در برخی موارد وجود داشته از جنبه‌های مختلف بهبود دهد.

مسئله‌ی اولی که بررسی شده است، محاسبه‌ی قطر نقاط در مدل پنجره‌ی لغزان است. این مسئله

[۲۲] Greedy

معادل شناسایی بیش‌ترین فاصله بین هر دو نقطه در یک پنجره است. الگوریتم بهینه‌ی $(\epsilon+1)$ــ تقریب این مسئله در فضای دوبعدی در [۱۰] آمده است و برای فضای متریک نیز روش ۳ــ تقریب وجود دارد که با در نظرگرفتن محدودیت حافظه این ضریب تقریب بهینه است. هدف ما از بررسی این مسئله شناسایی روشی کلی برای حل مسائل بهینه‌سازی هندسی در مدل پنجره‌ی لغزان است. روش ارائه‌شده توسط ما ضریب تقریب $(\epsilon+1)$ دارد.

دومین مسئله‌ی مورد مطالعه، معرفی بستری برای حل دسته‌ای از مسائل بهینه‌سازی هندسی (شامل قطر و kــ مرکز) است که در مدل پنجره‌ی لغزان قابلیت حل‌شدن با حافظه‌ی مناسب دارند. بستری که ارائه شده است با استفاده از الگوریتم تقریبی یا دقیق مدل ایستا می‌تواند همان مسئله را در مدل پنجره‌ی لغزان با تقریب $(B+\epsilon)$ حل کند (B ضریب تقریب حل مسئله در مدل ایستا است).

در مسئله‌ی سوم به طور اختصاصی روی kــ مرکز در فضای دوبعدی تمرکز کردیم. در تلاش اول برای مسئله‌ی ۱ــ مرکز روشی با تقریب $(\epsilon+1)$ و حافظه‌ی $O(\frac{1}{\epsilon}lgR)$ و زمان به‌روزرسانی $O(\frac{1}{\epsilon})$ ارائه دادیم. سپس روشی برای ۲ــ مرکز به دست آوردیم و در پایان برای حل مسئله‌ی kــ مرکز با ضریب تقریب $(\epsilon+2)$، حافظه‌ی $O(\frac{1}{\epsilon}lgR)$ و زمان به روزرسانی $O(\frac{1}{\epsilon})$ به دست آوردیم که از بهترین روش موجود که ۶ــ تفریب است بهبود قابل توجهی پیدا کرده است.

۱ــ۵ ساختار پایان‌نامه

این پایان‌نامه از پنج فصل تشکیل شده‌است که به شرح آن می‌پردازیم. فصل اول (همین فصل) راجع به مقدمات پژوهش توضیحاتی ارائه کرد. در فصل دوم مفاهیم و تعاریف مرتبط با موضوعات پژوهشی این پایان‌نامه بیان می‌کنیم. در این فصل تلاش بر این بوده تا با توضیحات کلی و ساده راجع به فضای پژوهش اطلاعات ضروری و مفید منتقل شود. فصل سوم این پایان‌نامه شامل مطالعه و در برخی موارد عمیق‌شدن در پژوهش‌های پیشین انجام شده‌ی مرتبط با موضوع این پایان‌نامه خواهد بود. این فصل در سه بخش تنظیم گردیده است. در بخش اول، مسائل kــ مرکز و قطر و عرض در حالت ایستا مورد بررسی قرار می‌گیرد. در بخش دوم، مدل پنجره‌ی لغزان مسائل قطر و عرض و روش‌های مرسوم حل این مسائل مورد بررسی قرار می‌گیرد. در نهایت، در بخش سوم، مسئله‌ی kــ مرکز در مدل پنجره‌ی لغزان مورد بررسی قرار می‌گیرد.

در فصل چهارم، نتایج جدیدی که در این پژوهشبه آن دست پیدا کرده‌ایم ارائه می‌شود. این نتایج شامل چارچوب ارائه‌شده برای تقریب $(1+\varepsilon)$ دسته‌ای از مسائل بهینه‌سازی هندسی در مدل پنجره‌ی لغزان و ارائه‌ی روش $(2+\epsilon)$ـ تقریب برای مسئله‌ی kـ مرکز است.

و در نهایت فصل پنجم به جمع‌بندی اختصاص دارد. در ابتدا خلاصه‌ای از نتایج به دست‌آمده و در قسمت دوم کارهای آینده که در طول این پژوهش به آن فکر کرده‌ایم وجود دارد.

فصل ۲

مفاهیم اولیه

در این فصل به تعریف و بیان مفاهیم پایه‌ای مورد استفاده در فصل‌های بعد می‌پردازیم. با توجه به مطالب مورد نیاز در فصل‌های آتی، مطالب این فصل به سه بخش، مسائل بهینه‌سازی هندسی، الگوریتم‌های پنجره‌های لغزان و الگوریتم‌های تقریبی تقسیم می‌شود.

۲ـ۱ مسائل بهینه‌سازی هندسی

همان‌طور که در مقدمه گفته شد، مسائل بهینه‌سازی هندسی از مسائل بسیار قدیمی علوم کامپیوتر به شمار می‌آیند. شاید از معروف‌ترین و پرکاربردترین این بهینه‌سازی‌ها به مسئله‌ی پوش محدب اشاره کرد که هدف آن کمینه‌کردن محیط یک چندضلعی است که تمامی نقاط ورودی را پوشش دهد. مسائل بهینه‌سای هندسی را می‌توان در فضای متریک یا اقلیدسی تعریف کرد. به عنوان مثال مسئله‌ی پوش محدب تنها در فضای اقلیدسی تعریف می‌شود اما مسئله‌ی kـ مرکز در هر دو فضای متریک و اقلیدسی قابل تعریف است. برای بررسی بیش‌تر فضای متریک و اقلیدسی را تعریف می‌کنیم.

تعریف ۲ـ۱ *فضای متریک: به مجموعه‌ی نقاط M و تابع متریک d به صورت*

$$d : M \times M \to \mathbb{R}$$

فضای متریک می‌گوییم اگر به ازای $x, y, z \in M$ خواص زیر را داشته باشد.

- $d(x,y) = 0 \iff x = y$
- $d(x,y) = d(y,x)$
- $Triangle\ Inequality: d(x,y) + d(y,z) \geqslant d(x,z)$

تعریف ۲ـ۲ *اگر مجموعه‌ی نقاط (M) فضای متریک معادل نقاط dـ بعدی باشد (یا یک بردار dـ متغیره که هر کدام یک عدد حقیقی هستند) و تابع متریک آن L_2 باشد (رجوع به تعریف L_Pـ متریک ۱ـ۱) آن فضا یک فضای اقلیدسی است.*

با کمی بررسی می‌توان متوجه شد که تابع L_2 خواص تعریف‌شده در فضای متریک را ارضا می‌کند. در نتیجه هر روشی که برای فضای متریک ارائه می‌شود قابل اجرا در فضای اقلیدسی است اما عکس آن صادق نیست. یکی از نکات مفید فضای اقلیدسی دامنه‌ی بزرگ مجموعه‌ی مرجع M و محدودیت موجود در تابع متریک آن است. از طرف دیگر بسیاری از مسائل دنیای واقعی (مثل نقشه‌های زمینی ۲ـ بعدی، طراحی‌های ۳ـ بعدی) در فضای اقلیدسی قرار دارند که باعث می‌شود کاربرد زیادی در مسائل هندسی داشته باشد.

یکی از تکنیک‌های موجود برای دسته‌بندی نقاط در فضای اقلیدسی **شبکه‌بندی**[1] است. در این روش محورها را با خطوطی موازی و با فاصله‌ی (معمولا) برابر تقسیم می‌کنند و در نتیجه فضای dـ بعدی به جعبه‌هایی با عرض مساوی در هر محور تقسیم می‌شود و هر نقطه به راحتی به یک جعبه اختصاص می‌یابد. از فواید این روش کاهش حجم داده‌ها به هزینه‌ی کاهش دقت در محاسبات است.

برخی از مسائل بهینه‌سازی هندسی (مانند kـ مرکز، kـ میانه یا kـ میانگین) رویکردهایی برای حل مسائل خوشه‌بندی است. این روش‌ها به طور معمول در مدل خوشه‌بندی‌های مرکزگرا با تخصیص قطعی داده هستند. اهداف این سه نوع مسئله را در زیر آورده‌ایم

- هدف خوشه‌بندی kـ مرکز شناسایی k نقطه است که اگر نقاط ورودی را بر حسب نزدیک‌ترین فاصله به این مراکز دسته‌بندی کنیم، بیش‌ترین فاصله‌ی هر نقطه‌ی ورودی با مرکز دسته‌اش کمینه شود.
- در خوشه‌بندی kـ میانه، هدف دسته‌بندی نقاط به k دسته است به نحوی که مجموع مربع فاصله‌ی هر نقطه از میانه‌ی نقاط آن دسته، کمینه شود.

[1] Grid

- تمرکز خوشه‌بندی kـ میانگین روی متوسط فاصله‌ی نقاط با مرکز دسته‌شان است. در خوشه‌بندی kـ میانگین، هدف افراز نقاط به k خوشه است به گونه‌ای که مجموع فاصله‌ی هر نقطه از میانگین نقاط داخل خوشه (یا مرکز آن خوشه) کمینه گردد.

همان‌طور که از تعریف مسئله‌ها به نظر می‌رسد، kـ میانه و kـ میانگین بیش‌تر رویکردی آماری دارند و kـ مرکز به مباحث هندسی نزدیک‌تر است. به همین دلیل تمرکز اصلی این پژوهش روی مسئله‌ی kـ مرکز است.

یکی از مسائلی که شباهت زیادی به مسئله‌ی ۱ ـ مرکز دارد مسئله‌ی کوچک‌ترین کره‌ی محیطی است. این مسئله را به صورت رسمی تعریف می‌کنیم.

تعریف ۲ـ۳ (کوچک‌ترین کره‌ی محیطی) *مجموعه‌ی نقاط P را در نظر بگیرید. $Meb(P)$ را برابر توپی با کوچک‌ترین شعاع تعریف می‌کنیم که تمام نقاط P را می‌پوشاند.*

۲ـ۲ الگوریتم‌های پنجره‌ی لغزان

روز به روز با مسائل بیش‌تری رو به رو می‌شویم که داده‌ها به مرور در حال تولید هستند و به دنبال پاسخ‌دهی به مسئله‌ای روی داده‌های موجود هستیم. به همین دلیل مدل‌های جدیدی برای برخورد مناسب با این مسائل معرفی می‌شود که در آن ورودی به مرور زمان در اختیار الگوریتم قرار می‌گیرد. با توجه به این که معمولا حجم اطلاعات ورودی بسیار بالاست نمی‌توان داده‌ها را به طور کامل ذخیره کرد تا در آینده از آن استفاده کنیم. نتیجه‌ی این امر باعث می‌شود الگوریتم‌ها در این مدل‌ها تنها امکان دسترسی یک یا چندباره به اطلاعات از طریق پویش[۲] آن از ابتدای ورودی تا انتهای ورودی وجود دارد. یکی از معروف‌ترین این مدل‌ها مدل جویبار داده است.

در واقع الگوریتم‌های جویبار داده، به الگوریتم‌هایی گفته می‌شوند که ورودی آن‌ها یک یا چند دنباله است که الگوریتم می‌تواند به ترتیب دنباله، یک یا چند بار از ابتدای دنباله تا انتهای آن، به اعضای دنباله دسترسی داشته باشد.

مدل‌های مختلفی شبیه به جویبار داده برای پاسخ به چنین محدودیت‌هایی به وجود آمده‌اند که در این‌جا به ۳ مدل جویبار داده، پنجره‌ی لغزان و گردان‌در اشاره می‌کنیم.

[۲] scan

تعریف ۲ـ۴ جویبار داده: *در این مدل داده‌ها یکی پس از دیگری وارد می‌شوند و ترتیب ورود داده‌ها مشخص نیست. تفاوت این مدل با مدل برخط این است که امکان نگه‌داری تمامی داده‌هایی که تا الان وارد شده‌اند را نداریم و حافظه از مرتبه‌ی برخطی است. الگوریتم‌های این مدل دو عمل زیر را انجام می‌دهند:*

- *به روز رسانی: در این عمل نقطه‌ی جدید وارد می‌شود. الگوریتم باید آن را پردازش کند و با توجه به محدودیت حافظه عملیاتی را اجرا کند.*
- *پرسمان: در این عمل از الگوریتم درخواست می‌شود تا پاسخ مسئله‌ی مورد نظر را برای تمام نقاطی که تا الان آمده‌اند خروجی دهد.*

تعریف ۲ـ۵ پنجره‌ی لغزان: *این مدل از مدل جویبار داده مشتق شده است و تنها تفاوت آن در عملیاتی است که انجام می‌دهد.*

- *به روز رسانی: در این عمل نقطه‌ی جدید وارد پنجره می‌شود. الگوریتم باید آن را پردازش کند و با توجه به محدودیت حافظه عملیاتی را اجرا کند.*
- *منقضی‌شدن یک نقطه: در این عمل نقطه‌ای که قبلا وارد پنجره شده است منقضی می‌شود و به قولی از پنجره خارج می‌شود. توجه شود که همیشه پیرترین نقطه می‌تواند منقضی شود و نمی‌توان نقطه‌ای جوان‌تر را از پنجره خارج کرد.*
- *پرسمان: در این عمل از الگوریتم درخواست می‌شود تا پاسخ مسئله‌ی مورد نظر را برای تمام نقاط معتبر (داخل پنجره) که وارد شده‌اند و منقضی نشده‌اند خروجی دهد.*

منقضی‌شدن نقطه در مدل پنجره‌ی لغزان می‌تواند از دو روش صورت بگیرد. در روش اول پنجره را ثابت در نظر می‌گیریم و برای آن ظرفیت تعیین می‌کنیم (مثلا N نقطه). در این حالت پس از ورود N نقطه، به ازای هر ورود (به روزرسانی) یک عمل منقضی‌شدن نقطه هم خواهیم داشت. در روشی دیگر پس از گذشت مقدار زمانی یک نقطه منقضی می‌شود. در مثال ۲ـ۱ داستان‌های اینستاگرام پس از گذشت ۲۴ ساعت از محاسبات خارج می‌شدند یا به قولی عملیات منقضی‌شدن داده اجرا می‌شد.

تعریف ۲ـ۶ گردان‌در: *این مدل گونه‌ای از مدل جویبار داده‌ی پویا است. نقاط ورودی مختصات*

صحیح [۳] *و محدود دارند (مختصات هر محور در مجموعه‌ی* $\{0, \cdots, U-1\}$ *قرار دارند) و عملیات زیر را انجام می‌دهند.*

- *به روز رسانی: در این عمل نقطه‌ی جدید وارد می‌شود. این نقطه می‌تواند تکراری باشد، الگوریتم باید آن را پردازش کند و با توجه به محدودیت حافظه عملیاتی را اجرا کند.*
- *حذف یک نقطه: در این عمل یک نقطه حذف می‌شود. توجه شود که نمی‌توان نقطه‌ای را بیش از مقداری که اضافه شده است حذف کرد و هم‌چنین محدودیتی در انتخاب نقاط برای حذف وجود ندارد.*
- *پرسمان: در این عمل از الگوریتم درخواست می‌شود تا پاسخ مسئله‌ی مورد نظر را برای تمام نقاط معتبر (داخل پنجره) که وارد شده‌اند و منقضی نشده‌اند خروجی دهد.*

الگوریتم‌های مدل‌های بالا معمولاً محدودیت شدیدی در میزان حافظه دارند (نسبت به اندازه‌ی ورودی) و به علت تعداد زیاد داده‌ها، محدودیت زمانی پردازش برای هر داده نیز مطرح است. این محدودیت‌ها معمولاً باعث می‌شود که این الگوریتم‌ها تنها بتوانند یک جواب تقریبی از جواب بهینه را با استفاده از اطلاعات مختصری که در حافظه نگه می‌دارد ارائه دهد.

با پیشرفت‌های سخت‌افزاری، امکان ایجاد و در عین حال جمع‌آوری داده‌ها به صورت مداوم بسیار آسان‌تر شده است. علاوه بر مثال‌هایی که در مقدمه اورده شد می‌توان به دیگر شبکه‌های اجتماعی، تلفن همراه یا بازی‌های بزرگ برخط اشاره کرد. ارتباط با این سیستم‌ها باعث ایجاد مقدار زیادی اطلاعات می‌شود و شرکت‌های بزرگ به دنبال استفاده از این داده‌ها برای سرویس‌دهی بهتر و یا شناسایی بازار مناسب خودشان هستند. زمانی که حجم تولید و دریافت داده‌ها به حدی باشد که امکان ذخیره‌سازی آن نیز وجود نداشته باشد به سراغ مدل‌های جویبار داده می‌رویم.

الگوریتم‌های جویبار داده بسیار با الگوریتم‌های برخط[۴] شباهت دارند. مهم‌ترین این شباهت‌ها عدم امکان دسترسی به تمام داده‌ها در شروع اجرای الگوریتم است. علاوه بر این شباهت تفاوت‌هایی با یک‌دیگر دارند. حافظه‌ی الگوریتم‌های جویبار داده بسیار محدود است که به طور معمول برای الگوریتم‌های برخط چنین محدودیتی وجود ندارد. از طرف دیگر الگوریتم‌های جویبار داده به پرسمان‌ها پاسخ می‌دهند که لزومی ندارد به ازای ورود هر داده پرسیده شود. اما الگوریتم‌های برخط به ازای هر

[۳] Integer
[۴] Online

ورود باید پاسخ پرسمان اصلی را بدهند. هر جویبار را می‌توان به عنوان دنباله‌ای مرتب از نقاط در نظر گرفت به طوری که به ترتیب دنباله قابل دسترسی هستند و هر کدام را تنها به تعدادی محدود بار (معمولاً یک بار) می‌توان خواند [۲].

۲ـ۲ـ۱ مجموعه هسته

با توجه به این که محدودیت زیادی در حافظه در مدل جویبار داده و پنجره‌ی لغزان داریم نیاز به کاهش اطلاعات نگه‌داری‌شده یا به نحوی گزینش نقاط و گردکردن آن‌ها داریم. در صورتی که مجموعه‌ای از نقاط یا نمایندگان آن‌ها را انتخاب کنیم که حجم بسیار کم‌تری از داده‌های اصلی دارند اما خطای اندکی ایجاد می‌کنند می‌توانیم به این محدودیت فائق آییم. به چنین مجموعه‌ای، مجموعه هسته می‌گوییم.

تعریف ۲ـ۷ *فرض کنید μ یک تابع اندازه‌گیری*[۵] *باشد که نقاط فضای dـ بعدی ($\mathbb{R}^d$) را به اعداد حقیقی نامنفی $\mathbb{R}^+ \cup \{0\}$ می‌نگارد (مثل تابع عرض مجموعه‌ای از نقاط). فرض کنید که این تابع، یک تابع یکنوا است، یعنی به ازای دو مجموعه‌ی $P_1 \subset P_2$ داریم*

$$\mu(P_1) \leqslant \mu(P_2)$$

فرض کنید $\epsilon > 0$ به عنوان پارامتر ورودی داده شده است، به زیرمجموعه‌ی $Q \subseteq P$ یک ϵـ مجموعه‌ی هسته برای P می‌گویند اگر رابطه‌ی زیر برقرار باشد:

$$(1 - \epsilon)\mu(P) \leqslant \mu(Q)$$

به عنوان یکی از مجموعه هسته‌های معروف، می‌توان به مجموعه هسته‌ی مطرح برای تابع اندازه‌گیری عرض نقاط اشاره کرد که به آن به اختصار ϵـ هسته[۶] می‌گوییم. ϵـ هسته یکی از اساسی‌ترین مجموعه هسته‌های مطرح است و برای طیف وسیعی از مسائل قابل استفاده است. الگوریتم‌های زیادی برای محاسبه‌ی ϵـ هسته در حالت ایستا ارائه شده است [۱۱].

[۵] Measure function

[۶] ϵ-Kernel

۲ـ۲ـ۲ موازی‌سازی

موازی‌سازی یکی از تکنیک‌های حل مسئله با استفاده از توزیع‌پذیری آن است. به طور معمول از موازی‌سازی برای افزایش سرعت روش یا اطمینان به پاسخ محاسبه‌شده استفاده می‌شود. حال فرض کنید حل‌کننده‌ای [۷] داریم که اگر پاسخ مورد انتظار (که از ورودی به دست می‌آید) در یک مجموعه‌ی از پیش تعیین‌شده باشد آن را خروجی می‌دهد و در غیر این صورت اعلام می‌کند که پاسخ این مسئله در این مجموعه نیست (و هیچ اطلاعات بیش‌تری راجع خروجی نمی‌دهد). اگر بتوان پاسخ یک مسئله را به مجموعه‌هایی افزار کرد و برای هر مجموعه یک حل‌کننده طراحی کرد، می‌توان با تکنیک موازی‌سازی به پاسخ دقیق مسئله رسید. برای رسیدن به پاسخ دقیق کافی است ورودی مسئله را به حل‌کننده‌ها بدهیم، یکی از حل‌کننده‌ها پاسخ دقیق را دارد و بقیه می‌گویند که پاسخ مسئله در مجموعه‌ی متناظر آن‌ها نیست. حال کافی است پاسخ آن حل‌کننده را به عنوان پاسخ خروجی بدهیم.
از این تکنیک حتی در زمانی که پاسخ به مجموعه‌هایی که اشتراکشان ناتهی هست هم می‌توان استفاده کرد. به عنوان مثال فرض کنید ورودی یک مسئله عددی زیر ۱۰۰ است و به دنبال شناسایی اول‌بودن آن عدد هستیم. حال تعدادی حل‌کننده در نظر می‌گیریم که هر کدام بخش‌پذیری عدد بر یک عدد اول (زیر ۱۰۰) را پاسخ می‌دهد. همان‌طور که مشاهده می‌شود مجموعه‌ی متناظر هر حل‌کننده با هم اشتراک دارد (مثلا حل‌کننده‌های ۲ و ۳ و ۵ می‌توانند ۳۰ را حل کنند) اما کافی است پاسخ تمام حل‌کننده‌ها را با هم and کنیم که به پاسخ نهایی برسیم.

۲ـ۳ الگوریتم‌های تقریبی

در علوم کامپیوتر خانواده‌های زیادی از مسائل وجود دارد. یک مبحث در این مسائل عدم حل‌پذیری برخی از آن‌ها است. شاید معروف‌ترین مثال مسائلی که راه حلی برای آن نداریم مسئله‌ی توقف[۸] باشد. یک برنامه (ماشین تورینگ) و یک ورودی داریم و می‌خواهیم بدانیم آیا این برنامه روی این ورودی متوقف خواهد شد یا خیر.

این مسائل خیلی در دنیای واقعی کاربرد ندارند. موضوع مهمی که در مسائل حل‌شونده وجود دارد کارآمدی حل آن‌ها است. این کارآمدی می‌تواند به خاطر سرعت پایین الگوریتم یا حجم بالای حافظه

[۷] Solver
[۸] Halting problem

باشد. به عنوان مثال اگر الگوریتمی داشته باشیم که زمان مصرفی آن مرتبه‌ی نمایی نسبت به ورودی داشته باشد برای ورودی‌های بزرگ بسیار زمان‌بر خواهد بود و عمل نمی‌توان آن را اجرا کرد.

برای ساده‌کردن بررسی این مسائل گونه‌ای از مسائل به نام تصمیم‌گیری ایجاد شدند.

مسئله‌ی ۲ـ۱ (مسائل تصمیم‌گیری)[۹] *به دسته‌ای از مسائل گفته می‌شود که پاسخ آن‌ها تنها بله یا خیر است.*

به عنوان مثال، نسخه‌ی تصمیم‌گیری مسئله‌ی kـ مرکز در فضای IR^d به صورت زیر تعریف می‌شود.

مسئله‌ی ۲ـ۲ (نسخه‌ی تصمیم پذیر kـ مرکز) *مجموعه‌ی نقاط در فضا $\mathbb{R}^d$ و شعاع r داده شده است، آیا k دایره به شعاع r وجود دارد که تمام نقاط ورودی درون دایره‌ها باشند؟*

اگر بتوان مسئله‌ی اصلی را به تعدادی مسئله‌ی تصمیم تقسیم کرد می‌توان با حل مسئله‌ی تصمیم مسئله‌ی اصلی را نیز حل کرد.

همان‌طور که احتمالا می‌دانید مهم‌ترین دسته‌بندی مسائل تصمیم‌گیری به خانواده‌های مسائل P و NP تقسیم می‌شود. مسائلی که راه حل چندجمله‌ای برای حل آن‌ها وجود دارد خانواده‌ی مسائل P، به شمار می‌آیند. مسائلی که در زمان چندجمله‌ای می‌توان پاسخ آن‌ها را سنجید خانواده‌ی مسائل NP، می‌شوند. به عبارت دیگر اگر پاسخ یک مسئله داده شود آیا می‌توان در زمان چندجمله‌ای مطمئن شد که این پاسخ صحیح است یا خیر. مسائل مختلفی در علوم کامپیوتر مورد بررسی قرار گرفت که تعدادی از آن‌ها در دسته‌ی NP قرار می‌گیرند اما مطمئن نیستیم که در دسته‌ی P نیز قرار دارند یا خیر. به این مجموعه مسائل **NPـ سخت** می‌گویند. اگر ثابت شود یکی از این مسائل در دسته‌ی P قرار می‌گیرد اثبات می‌شود که $P = NP$.

با توجه به تعریف مسائل NP می‌توان متوجه شد که الگوریتم‌های ارائه‌شده برای آن‌ها کارآمد نیستند. علاوه بر این برخی از مسائل P نیز دارای الگوریتم کارآمدی نیستند.

بیش‌تر مسائل کاربردی دنیای واقعی جزو خانواده‌ی NPـ سخت هستند و راه حل چندجمله‌ای ندارند. اگر هم این طور نباشد مرتبه‌ی زمانی بالایی دارند. در نتیجه امکان استفاده از این روش‌ها عملی نیست. یک روش برای افزایش سرعت، کاهش دقت پاسخ خروجی است. یعنی به جای این روشی

[۹] Decision problems

مسئله	کران پایین تقریب‌پذیری
kـ مرکز	۲[۱۲]
kـ مرکز در فضای اقلیدسی	۱/۸۲۲[۱۳]
۱ـ مرکز در حالت جویبار داده	$\frac{1+\sqrt{2}}{2}$ [۸]
۱ـ مرکز در مدل پنجره‌ی لغزان (۲ـ بعدی)	$(1+\epsilon)$[۱۰]
۲ـ مرکز در فضای متریک	$4+\varepsilon$[۷]

جدول ۲ـ۱: نمونه‌هایی از کران پایین تقریب‌پذیری مسائل بهینه‌سازی

که پاسخ دقیق را در زمان خیلی زیاد به دست می‌آورد، روشی ارائه دهیم تا پاسخ اندکی نادقیق را در زمان بسیار کوتاهی به دست آورد. در صورتی که بتوان برای مقدار خطایی که الگوریتم ایجاد می‌کند کرانی ارائه دهیم به آن یک الگوریتم تقریبی می‌گوییم. الگوریتم تقریبی تضمین می‌کند در هر شرایطی پاسخی خروجی بدهد که از محدوده‌ای بدتر نیست. به عنوان مثال برای مسئله‌ی فروشنده‌ی دوره‌گرد در فضای متریک می‌توان الگوریتمی ۲ـ تقریب ارائه داد که دوری را خروجی می‌دهد که حداکثر دو برابر کوتاه‌ترین دوری است که از تمام رئوس بگذرد. به حداکثر نسبت جواب الگوریتم تقریبی به جواب بهینه ضریب تقریب یک الگوریتم تقریبی گفته می‌شود.

با رویکرد الگوریتم‌های تقریبی به نظر می‌رسد می‌توان برای هر مسئله‌ای تا هر مقدار ضریبی که بخواهیم الگوریتم تقریبی ارائه کرد. در واقعیت چنین چیزی امکان‌پذیر نیست و برای ضریب تقریب مسائل نیز کران پایین وجود دارد. یعنی با فرض $P \neq NP$ بعضی مسائل را در زمان چندجمله‌ای نمی‌توان از حدی دقیق‌تر کرد.

برخی مسائل وجود دارند که حتی نمی‌توان الگوریتم تقریبی با ضریب ثابت برای آن‌ها ارائه کرد. معروف‌ترین مثال این دسته مسئله‌ی فروشنده‌ی دوره‌گرد در حالت کلی است (مسئله‌ی فروشنده‌ی دوره‌گرد در فضای متریک تقریب ۲ دارد).علاوه بر این به طور مثال، همان‌طور که در فصل کارهای پیشین بیان خواهد شد، الگوریتم تقریبی با ضریب تقریب کم‌تر از ۲، برای مسئله‌ی kـ مرکز وجود ندارد مگر اینکه $P = NP$ باشد.

در جدول ۲ـ۱ میزان تقریب‌پذیری مسائل مختلفی که در این پایان‌نامه مورد استفاده قرار می‌گیرد (و جزو نتایج این پایان‌نامه نیست) را می‌بینید.

فصل ۳

کارهای پیشین

در این فصل کارهای پیشین انجام‌شده روی مسئله‌ی kـ مرکز، در سه بخش مورد بررسی قرار می‌گیرد. در بخش اول، مسائل قطر، عرض و kـ مرکز در مدل ایستا مورد بررسی قرار می‌گیرد. در بخش دوم، مسائل قطر و عرض در حالت پنجره‌ی لغزان و مجموعه هسته‌های مطرح برای این مسئله مورد بررسی قرار می‌گیرد و در نهایت، در بخش سوم، مسئله‌ی kـ مرکز در مدل پنجره‌ی لغزان مورد بررسی قرار می‌گیرد.

۳ـ۱ مسائل بهینه‌سازی هندسی در مدل ایستا

در این بخش می‌خواهیم مسائلی از بهینه‌سازی هندسی را در مدل ایستا بررسی کنیم که یا در هدف این پژوهش بوده‌اند (قطر و kـ مرکز) یا پژوهش مهمی در مدل پنجره‌ی لغزان آن شده است (عرض نقاط).

۳ـ۱ـ۱ قطر

مسئله‌ی قطر در مدل ایستا معادل پیداکردن دورترین فاصله بین هر دو نقطه‌ای از n نقطه‌ی ورودی است. ابتدایی‌ترین روش برای این مسئله مقایسه‌ی تمامی $\binom{n}{2}$ مقدار ممکن است. برای فضای اقلیدسی dـ بعدی کران پایین $\omega(nlogn)$ ثابت شده است. [۱۴] در فضای ۲ـ بعدی با استفاده از ایده‌ی پوش محدب می‌توان قطر را در مرتبه‌ی زمانی $O(nlogn)$ به دست آورد. واضح است که نقاطی که قطر را

تشکیل می‌دهند باید جزو رئوس پوش محدب باشند، پس با مرتبه‌ی زمانی $O(nlogn)$ ابتدا پوش محدب را به دست می‌آوریم، سپس به ازای هر راس پوش محدب به وسیله‌ی جستجوی دودویی [1] دورترین راس پوش را نسبت به آن راس به دست می‌آوریم. هم‌چنین محاسبه‌ی قطر در فضای ۲ـ بعدی کران پایین $\omega(n)$ را دارد. در فضای ۳ـ بعدی کار اندکی سخت‌تر است ولی الگوریتم با مرتبه‌ی زمانی $O(nlogn)$ وجود دارد [۱۵].

همان‌طور که دیده می‌شود روش‌های مسئله‌ی محاسبه‌ی قطر در ابعاد پایین مرتبه‌ی زمانی نزدیک به خطی دارند اما به علت این که برای ابعاد بالا روش کارآمدی وجود ندارد و حتی همین الگوریتم‌های ابعاد پایین با مرتبه‌ی خطی و زیرخطی فاصله دارند پژوهش‌هایی در راستای ارائه‌ی الگوریتم تقریبی برای قطر نقاط صورت گرفته است. یک الگوریتم تقریبی نسبتا بدیهی ۲ـ تقریب با مرتبه‌ی زمانی $O(dn)$ وجود دارد که یک نقطه را می‌گیرد و دورترین نقطه نسبت به آن را پیدا می‌کند، قطر از ۲ برابر این مقدار قطعا کوچک‌تر است پس ۲ برابر این مقدار یک ۲ـ تقریب برای قطر به شمار می‌آید. در ادامه یک الگوریتم $\sqrt{3}$ـ تقریب برای محاسبه‌ی قطر در dـ بعد ارائه شده است [۱۶]. تلاش‌هایی برای ارائه‌ی الگوریتم $(1+O(\epsilon))$ـ تقریب هم صورت گرفته است که به علت مرتبه‌ی زمانی نسبتا بالا دیگر به آن نمی‌پردازیم.

۳ـ۱ـ۲ عرض

ابتدا مفهوم عرض نقاط در فضا dـ بعدی را به صورت غیررسمی بیان می‌کنیم.

تعریف ۳ـ۱ عرض نقاط: *مجموعه‌ی P شامل تعدادی نقاط dـ بعدی است. دو صفحه[2] موازی را به نام‌های HP_1 و HP_2 در نظر بگیرید که تمامی نقاط P بین این دو صفحه قرار دارد. به کم‌ترین فاصله‌ی بین هر دو صفحه عرض نقاط می‌گوییم و با $w(P)$ نمایش می‌دهیم.*

همان‌طور که در قسمت قطر گفته شد می‌توان در فضای ۲ـ بعدی عرض نقاط را به صورت دقیق در زمان $O(nlogn)$ به دست آورد [۱۴]. محاسبه‌ی عرض نقاط در ابعاد بالاتر به سادگی محاسبه‌ی قطر نیست. در فضای ۳ـ بعدی اولین الگوریتمی که برای حل دقیق ارائه شد مرتبه‌ی زمانی $O(n^2)$ داشت [۱۷]. ایده‌ی این الگوریتم از تکنیک چرخاندن کولیس [3] به دست می‌آید. علت نام‌گذاری این روش

[1] Binary Search
[2] Hyperplane
[3] Rotating Calipers

شباهت آن به اندازه‌گیری یک جسم چندوجهی به وسیله‌ی کولیس است که وقتی یک بازوی کولیس با یک یال چندوجهی تماس پیدا می‌کند بازوی دیگر به یک نقطه‌ی چندوجهی می‌رسد که این دو نقطه، جفت پادپایی [۴] هستند. به وسیله‌ی چرخاندن چندوجهی می‌توان تمام جفت‌های پادپایی را شناسایی کرد که فاصله‌ی کوچک‌ترین آن‌ها برابر با عرض خواهد بود [۱۴]. برای ابعاد بالاتر روشی با مرتبه‌ی زمانی $O(n^{\lfloor \frac{d}{2} \rfloor})$ وجود دارد [۱۸].

یکی از مهم‌ترین پیش‌رفت‌ها در الگوریتم‌های تقریبی هندسی در حوزه‌ی محاسبه‌ی عرض نقاط به وجود آمده است. معرفی ایده‌ی ϵــهسته علاوه بر این که بستری برای کاهش حجم نقاط ورودی با کاهش دقت در تقریب $(1+\epsilon)$ ایجاد کرد، ابزار بسیار قدرتمندی برای حل مسائل بهینه‌سازی هندسی در مدل‌های جویبار داده به شمار آمد. قبل از ϵــهسته روش‌هایی برای محاسبه‌ی عرض نقاط با تقریب $(1+\epsilon)$ ارائه شده بود [۱۹] اما مرتبه‌ی زمانی بالایی مصرف می‌کرد و پیچیده بود که در پژوهش‌های دیگری سعی کردند این موارد را بهبود دادند [۲۰].

۳ـ۱ـ۳ kــمرکز

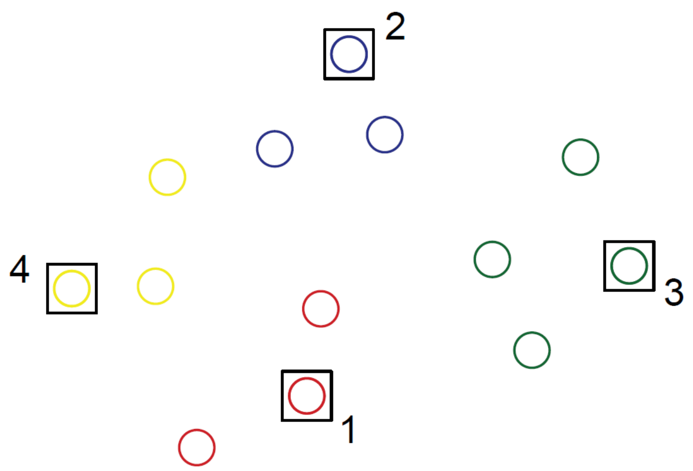
همان‌طور که در فصل‌های قبل گفتیم مسئله‌ی kــمرکز مسئله‌ای معروف در علوم کامپیوتر است. می‌خواهیم مراکزی را انتخاب کنیم که بیش‌ترین فاصله‌ی نقاط از مرکز دسته‌ی متناظرش کمینه شود. در صورتی که مراکز مشخص باشند محاسبه‌ی این مقدار (شعاع) بسیار ساده است.

این مسئله در حالت کلی NPــسخت است [۲۱]. شاید این طور تصور شود که در فضای اقلیدسی یا ابعاد پایین مسئله ساده‌تر می‌شود اما ثابت شده است که این مسئله در صفحه‌ی دوبعدی نیز NPــسخت است [۳]. علاوه بر این که راه حل چندجمله‌ای برای حل دقیق این مسئله وجود ندارد ثابت شده است که هیچ الگوریتم تقریبی با ضریب تقریب بهتر از ۲ نیز برای تقریب‌زدن این مسئله وجود ندارد.

یکی از اولین الگوریتم‌های تقریبی که این کران پایین را پوشش می‌دهد توسط گنزالز ارائه شده است [۲۲]. این الگوریتم یک الگوریتم تقریبی با ضریب تقریب ۲ است و زمان مصرفی آن $\mathcal{O}(kn)$ است. ابتدا تابع فاصله‌ی یک نقطه از یک مجموعه را تعریف می‌کنیم.

تعریف ۳ـ۲ *فاصله‌ی نقطه‌ی v از مجموعه‌ای ناتهی از نقاط S را برابر فاصله‌ی نقطه‌ای درون S از v*

---
[۴] Antipodal Pair



شکل ۳ـ۱: نقاطی که مربع دارند مراکز پیشنهادی ۴ـ مرکز الگوریتم گنزالز هستند. نقطه‌ی ۱ ابتدا به صورت تصادفی انتخاب می‌شود. سپس نقطه‌ی ۲ بیش‌ترین فاصله را تا مجموعه‌ی شامل نقطه‌ی ۱ دارد پس به عنوان مرکز دوم انتخاب می‌شود. این فرآیند برای نقاط ۳ و ۴ هم تکرار می‌شود.

تعریف می‌کنیم، به‌گونه‌ای که از تمام نقاط S به v نزدیک‌تر باشد. در واقع داریم:

$$d(v,S) = \min_{u \in S} \{d(u,v)\}$$

در الگوریتم گنزالز، یک مجموعه داریم که مراکز پیشنهادی را به تدریج به آن اضافه می‌کنیم. ابتدا یک نقطه‌ی دلخواه را به عنوان اولین مرکز به این مجموعه اضافه کنیم. سپس در هر مرحله نقطه‌ای که بیش‌ترین فاصله را با این مجموعه‌ی مراکز دارد به عنوان مرکز بعدی به مجموعه اضافه می‌کنیم. این کار را تا وقتی تکرار می‌کنیم که مجموعه‌ی مذکور دارای k عضو شده باشد. حال هر نقطه را به مرکزی اختصاص می‌دهیم که کم‌ترین فاصله را با آن دارد. ثابت می‌شود که بیش‌ترین فاصله‌ی مراکز با نقاط دسته‌ی متناظرشان حداکثر دو برابر شعاع بهینه در مسئله‌ی kـ مرکز است. به عبارت دیگر این الگوریتم ضریب تقریب ۲ دارد.

نمونه‌ای از اجرای الگوریتم گنزالز برای مسئله‌ی ۴ـ مرکز در شکل ۳ـ۱ نشان داده شده است.

مواردی که تا این‌جا مورد بررسی قرار دادیم روی حالات کلی مسئله‌ی kـ مرکز تمرکز شده بود. با توجه به گستردگی کاربرد مسئله‌ی kـ مرکز برای حالاتی که k یا d مقدار مشخص و کوچکی دارند پژوهش‌هایی انجام شده است. به عنوان مثال می‌توان به مسئله‌ی ۱ـ مرکز هندسی یا کوچک‌ترین کره‌ی

محیطی اشاره کرد که برای حل آن الگوریتم با زمان اجرای $\mathcal{O}((d+1)!n)$ وجود دارد [۲۵]. این روش یک الگوریتم تصادفی به حساب می‌آید و ایده‌ی اصلی آن این است که چنین کره‌ای را می‌توان با استفاده از حداکثر $d+1$ نقطه شناسایی کرد. علت زمان پایین اجرای این الگوریتم این است که احتمال این که هر نقطه جزو نقاط مرزی باشد پایین است.

هم‌چنین برای مسئله‌ی ۲ ــ مرکز هندسی بهترین الگوریتم نزدیک به خطی با زمان اجرای $\mathcal{O}(n\log^{2} n\log^{2}\log n)$ و حافظه‌ی $\mathcal{O}(n)$ موجود است [۲۶].

۳ــ۲ تاریخچه‌ی مدل پنجره‌ی لغزان

می‌توان گفت مدل پنجره‌ی لغزان در علوم محاسباتی در دو مسیر متفاوت پیشرفت کرد. مسیر اول رویکرد آماری به مسائل بود که خود مدل پنجره‌ی لغزان را معرفی کرد و مسیر دوم مسائل بهینه‌سازی هندسی بودند که از آمار فاصله گرفتند. به علت اهمیت مسئله‌ی kــ مرکز در مدر پنجره‌ی لغزان، این مسئله را در یک قسمت جدا بررسی خواهیم کرد.

۳ــ۲ــ۱ مسائل آماری در پنجره‌ی لغزان

مدل پنجره‌ی لغزان سال ۲۰۰۲ در [۲۸] توسط داتار و سایرین ارائه شد. آن‌ها با ارائه‌ی بستر هیستوگرام نمایی [۵] ابتدا مسئله‌ی شمارش ساده را حل کردند سپس با تعمیم آن به مسئله‌ی جمع (مجموع N داده‌ی اخیر) خانواده‌ی توابعی را معرفی کردند که به وسیله‌ی بستر هیستوگرام نمایی می‌توان با تقریب $O(1+\varepsilon)$ و سربار حافظه‌ی $O(\frac{1}{\varepsilon}logN)$ آن‌ها را محاسبه کرد. این مقاله علاوه بر این که مدل پنجره‌ی لغزان را معرفی کرد، بستر بسیار مناسبی برای مدل‌سازی مسائل دیگر ارائه داد. مسائل ساده‌ی آماری مانند میانگین در این مقاله به طور کامل پوشش داده شد، بابکوک و سایرین در [۲۹] این روند را ادامه دادند و مسائل واریانس و Kــ میانه [۶] را حل کردند. فضای حافظه‌ی الگوریتم آن‌ها برای مسئله‌ی Kــ میانه برابر $O(\frac{k}{\tau^{4}}N^{2\tau}log^{2}N)$ به ازای $\tau<\frac{1}{2}$ و تقریب $O(2^{O(\frac{1}{\tau})})$ بود. در این مقاله مسئله‌ی بازی مطرح شد که «آیا الگوریتمی برای مسئله‌ی Kــ میانه وجود دارد که مرتبه‌ی حافظه‌ی آن نسبت به اندازه‌ی پنجره لگاریتمی باشد؟». این مسئله در دو مرحله توسط بریورمن و سایرین حل شد. ابتدا آن‌ها در [۳۰]

[۵] Exponential Histogram

[۶] $K-Means$

هیستوگرام نمایی را هم از نظر دقت تقریب و هم از نظر عمومیت خانواده‌ی توابعی که پشتیبانی می‌کند پیشرفت دادند و نام هیستوگرام ملایم [۷] روی آن نهادند. سپس با استفاده از این بستر جدید در [۳۱] توانستند مسئله‌ی باز مطرح‌شده Kـ میانه در [۲۹] را با حافظه‌ی $O(k^3 log^6 N)$ حل کنند. حال مسئله‌ی مهمی از این حوزه که پایه‌ی روش‌های دیگر به حساب می‌آید را به طور دقیق‌تر بررسی می‌کنیم.

شمارش ساده

مدل پنجره‌ی لغزان با معرفی این مسئله شروع شد که منجر به معرفی خانواده‌ای از توابع شد که به وسیله‌ی بستر آن‌ها قابل محاسبه در مدل پنجره‌ی لغزان بود. خانواده‌ی توابع f و مجموعه‌های A و B از داده‌ها را در نظر بگیرید که

- $f(A) \geqslant 0$
- $f(A) \leqslant poly(|A|)$
- $f(A \cup B) \geqslant f(A) + f(B)$
- $f(A \cup B) \leqslant C_f(f(A) + f(B))$

در عبارات بالا در نظر بگیرید که C_f یک عدد ثابت است که تنها به تابع f وابسته است.

در مسئله‌ی شمارش ساده، داده‌ها اعداد ۰ یا ۱ هستند و می‌خواهیم تعداد داده‌ی ۱ در پنجره‌ی Nتایی اخیر را بشماریم. برای این مسئله، تابع f را این گونه تعریف می‌کنیم که $f(A)$ برابر است با شماره‌ی نزدیک‌ترین داده‌ی ۱ در مجموعه‌ی A. به عنوان مثال اگر A را داده‌های پنجره‌ی Nتایی اخیر بدانیم، $f(A)$ برابر نزدیک‌ترین داده‌ی ۱ ی است که در پنجره وجود دارد.

حال اگر بسته‌هایی را در نظر می‌گیریم که در هر کدام تعدادی از داده‌ها وجود دارد. به جای نگه‌داری تمام داده‌ها در بسته مجموع داده‌های ۱ آن بسته را نگه‌داری می‌کنیم (همان تابع f). برای بسته‌ای که مقدار ۱ های آن برابر C است تقریب $\frac{c}{2}$ را برای داده‌های ۱ ی که در پنجره‌ی Nتایی اخیر وجود دارد در نظر می‌گیریم. مقدار خطای این ایده محاسبه می‌شود و از روی آن یک الگوریتم $(1 + \varepsilon)$ـ تقریب برای تعداد داده‌های ۱ در پنجره‌ی Nتایی اخیر داریم.

[۷] Smooth Histogram

۳ـ۲ـ۲ محاسبه‌ی عرض و قطر نقاط در پنجره‌ی لغزان

در این قسمت می‌خواهیم پژوهش‌های انجام‌شده برای محاسبه‌ی عرض و قطر نقاط در پنجره‌ی لغزان بپردازیم. با توجه به محدودیت‌های بسیار زیاد مدل‌های جویبار داده و پنجره‌ی لغزان اکثر مسائل هندسی مطرح‌شده در این مدل‌ها پاسخی تقریبی دارند و کران پایین تقریب آن‌ها بیش از ۱ است. به عنوان مثال در مدل پنجره‌ی لغزان مسئله‌ی عرض نقاط در ساده‌ترین حالت خود یعنی ۱ـ بعدی برای پاسخ دقیق نیازمند حافظه‌ی $\omega(N)$ است . در قسمت قبلی (مسائل بهینه‌سازی هندسی در مدل ایستا) روش‌های معمول حل این مساسل را (دقیق و تقریبی) بررسی کردیم.

پس از معرفی مدل پنجره‌ی لغزان توسط داتار و سارین در [۲۸] فین‌باوم و سایرین در [۳۲] مسائل هندسی را به این مدل و مدل جویبار داده‌ها آوردند. آن‌ها برای تقریب $O(1+\varepsilon)$ قطر نقاط ۱ـ بعدی در مدل پنجره‌ی لغزان، کران پایین $O(\frac{1}{\varepsilon}logRlogN)$ (که R برابر نسبت بیش‌ترین فاصله‌ی نقاط به کم‌ترین آن‌ها است) را ثابت کردند. آن‌ها از ایده‌ی روش لگاریتمی [8] [۳۳] برای پیداکردن بزرگ‌ترین (و کوچک‌ترین) نقطه در فضای یک‌بعدی استفاده کردند. الگوریتم آن‌ها به این صورت بود که خوشه‌هایی از نقاط داخل پنجره (که اندازه‌ی هر خوشه از توان ۲ بود) می‌ساختند و اگر دو خوشه اندازه‌ی یکسان داشتند با یک‌دیگر ادغام می‌کردند. نکته‌ی کلیدی هر خوشه مرکز آن بود که باید جوان‌ترین نقطه‌ی خوشه باشد. بقیه‌ی نقاط داخل پوشه با فاصله‌های متناسب با تصاعد هندسی $(1+\varepsilon)$ گرد[9] می‌شدند. به این ترتیب تعداد نمایندگان نقاط (همان اعداد گردشده) بسیار کم‌تر از تعداد نقاط اصلی بود.

یکی از کارهای مهم فین‌باوم و سایرین به دست آوردن حد پایین حافظه برای تقریب قطر $1+\varepsilon$ در مدل پنجره‌ی لغزان بود. که ما قضیه‌ی نهایی پژوهش آن‌ها را در این‌جا می‌آوریم.

قضیه‌ی ۳ـ۱ (فین‌باوم و سایرین [۳۲]) *فرض کنید R نسبت بیشینه‌ی قطر به کوچک‌ترین فاصله‌ی بیش از صفر بین هر دو نقطه‌ای در تمام پنجره‌ها باشد. در صورتی که داشته باشیم $\log R \leqslant \frac{3}{2}\varepsilon \cdot n^{1-\delta}$ (به ازای δ ثابت)، برای تقریب $1+\varepsilon$ قطر نقاط یک خط (فضای ۱ـ بعدی) در مدل پنجره‌ی لغزان به اندازه‌ی N نیازمند $\Omega(\frac{1}{\varepsilon}\log R \log N)$ بیت حافظه داریم. و در صورتی که داشته باشیم $\log R \leqslant \frac{3}{2}\varepsilon \cdot n$ مرتبه‌ی حافظه برابر با $\Omega(N)$ خواهد بود.*

چن و سجاد در [۱۰] الگوریتمی با حافظه‌ی کمینه‌ی ۳ـ۱ برای محاسبه‌ی عرض و قطر در فضای

[8] Logarithmic Method
[9] Round

یک‌بعدی ارائه دادند.. چن و سجاد هم‌چنین راه حل خود را برای ابعاد بالاتر به وسیله‌ی εـ هسته گسترش دادند. ایده‌ی اصلی آن‌ها برای کاهش حافظه‌ی نگه‌داری‌شده ساخت یک دنباله‌ی خلاصه‌شده [۱۰] از نقاط در فضای یک‌بعدی بود.

تعریف ۳ـ۳ (دنباله‌ی خلاصه‌شده چن و سجاد [۱۰]) *فرض کنید $Q = \langle q_1, q_2, \ldots, q_k \rangle$ یک زیردنباله از دنباله‌ی P (نقاط داخل پنجره‌ی لغزان) که $q_1 < q_2 < \ldots < q_k$ است. فرض کنید $pred_Q(p)$ بزرگ‌ترین نقطه در Q است که حداکثر برابر با p و $succ_Q(p)$ کوچک‌ترین نقطه در Q است که حداقل برابر با p است. به Q یک دنباله‌ی خلاصه‌شده از P می‌گوییم اگر شرایط زیر را داشته باشد.*

- *مقادیر q_iها به ترتیب نزولی زمان ورود باشند.*
- *به ازای هر $p \in P$ اگر $pred_Q(p)$ وجود داشته باشد نباید پیرتر از p باشد.*
- *به ازای هر p یا باید داشته باشیم $||p - pred_Q(p)|| \leqslant \varepsilon\Delta(P)$ و یا $succ_Q(p)$ نباید پیرتر از p باشد.*

باید به این نکته توجه کرد که دنباله‌ی خلاصه‌شده لزوما یکتا نیست. چن و سجاد برای این که کوچک‌ترین دنباله‌ی خلاصه‌شده را به دست بیاورند در زمان ورود دنباله‌ی موجود را اصلاح می‌کنند تا به حافظه‌ی $\mathcal{O}(\frac{1}{\varepsilon}\log R)$ برسند.

قابلیت منحصر به فرد دنباله‌ی خلاصه‌شده در این است که مانند یک صف [۱۱] مرتب عمل می‌کند. هر نقطه‌ی جدیدی که وارد می‌شود از ابتدای صف (q_1) وارد می‌شود و تا جایی پیش می‌رود که از نقطه‌ی بعدی صف کوچک‌تر باشد. هم‌چنین نقاط قبل از خودش را هم از صف خارج می‌کند. حال برای این که در یک پنجره بزرگ‌ترین نقطه را داشته باشیم کافی است به q_k نگاه کنیم. زمانی که q_k منقضی می‌شود نقطه‌ی q_{k-1} بزرگ‌ترین نقطه‌ی معتبر بعد از آن است. علاوه بر این موضوع به خاطر ویژگی سوم دنباله‌ی خلاصه‌شده می‌توانیم نقاطی که خیلی به هم نزدیک هستند را هم حذف کنیم و فقط جوان‌ترین آن‌ها را نگه‌داری کنیم. به همین دلیل اندازه‌ی Q از مرتبه‌ی $\mathcal{O}(\log^R_{1+\varepsilon})$ خواهد بود.

پس از این که مسئله‌ی قطر و عرض نقاط یک‌بعدی در پنجره‌ی لغزان تقریب $(1 + \varepsilon)$ زده شد، با استفاده از ایده‌ی εـ هسته [۳۴] (نگه‌داری تعدادی خطوط با زاویه‌ی بین حداکثر $\arccos(\frac{1}{1+\varepsilon})$) مسئله‌ی قطر در فضاهای ابعاد ثابت نیز حل می‌شود. علاوه بر این چن و سجاد روشی برای محاسبه‌ی

[۱۰] Summary Sequence
[۱۱] Queue

قطر نقاط در مدل پنجره‌ی لغزان در فضای هندسی با ابعاد بالا ارائه دادند. این الگوریتم دارای ضریب تقریب $O(2^{m+2}-2+\varepsilon)$ و حافظه‌ی مصرفی از مرتبه‌ی $O(N^{\frac{1}{m+1}} logR)$ است.

همان‌طور که در قسمت‌های قبل اشاره کردیم محاسبه‌ی عرض مسئله‌ی نسبتا پیچیده‌ای است و حتی در مدل ایستا روش‌های کارآمدی برای حل آن وجود ندارد. چن و سجاد پس از تقریب عرض در فضای یک‌بعدی (که معادل قطر است) یک εـ هسته در صفحه با هدف تقریب عرض ارائه دادند. یکی از مهم‌ترین ویژگی‌های εـ هسته این است که علاوه بر این که مسئله‌ی عرض را تقریب می‌زند، مسائلی که به نحوی با نقاط خارجی وابسته هستند (مثل کوچک‌ترین کره‌ی محیطی) را نیز می‌تواند تقریب بزند. حافظه‌ی مصرفی این εـ هسته از مرتبه‌ی $\mathcal{O}(\frac{1}{\varepsilon^2} log^3 N \log R polylog(R', N, \frac{1}{\varepsilon})$ است که مقدار R' برابر با عرض هر سه نقطه‌ی متوالی در تمامی پنجره‌ها است. اثبات می‌شود که کران پایین حافظه برای حل مسئله‌ی عرض به مقدار $\log R'$ وابسته است [۱۰].

تا این‌جا تمامی پژوهش‌های انجام‌شده در فضای هندسی و ابعاد پایین بود. به تازگی کهن‌ادد و سایرین در [۷] الگوریتمی برای تقریب قطر در فضای متریک الگوریتمی با ضریب تقریب $3+\varepsilon$ ارائه دادند. علاوه بر این ثابت کردند در فضای متریک هر الگوریتم $(3-\varepsilon)$ـ تقریبی برای این مسئله نیاز به حافظه‌ی $O(N^{\frac{1}{3}})$ دارد.

آن‌ها رویکرد جدیدی برای حل مسئله‌ی قطر اتخاذ کردند. در کارهای فین‌باوم و سایرین [۳۲] و چن و سجاد [۱۰] ایده‌ی اصلی حل مسئله در فضای یک‌بعدی و گسترش آن به فضاهای دیگر بود. به عبارت دیگر، اندازه‌ی قطر در الگوریتم آن‌ها تاثیری نداشت. کهن‌ادد و سایرین به جای تمرکز روی ابعاد پایین روی تقسیم بازه‌های پاسخ متمرکز شدند. آن‌ها زیرالگوریتم‌هایی برای حل مسئله در بازه‌های کوچک $[\gamma, 3\gamma)$ ایجاد کردند سپس با ایجاد بازه‌هایی که از یک‌دیگر به نسبت $(1+\varepsilon)$ فاصله داشتند نسبت به تقریب پاسخ اصلی اقدام کردند.

نکته‌ی کلیدی الگوریتم آن‌ها در نگه‌داری یک یا دو مرکز داخل پنجره است تا بتواند تشخیص دهد آیا پاسخ در این بازه قرار دارد. در صورتی که یک مرکز داشته باشیم، فاصله‌ی هر دو نقطه‌ی پیرتر از مرکز با یک‌دیگر حداکثر 2γ و فاصله‌ی مرکز با نقاط بعد از خودش حداکثر γ خواهد بود. به این ترتیب به خاطر ویژگی نامساوی مثلثاتی فاصله‌ی هر دو نقطه‌ای از یک‌دیگر حداکثر برابر با 3γ خواهد شد. در صورتی که دو مرکز داشته باشیم فاصله‌ی آن دو بیش از 2γ است و در این حالت پاسخ در بازه‌ی مورد نظر قرار ندارد. الگوریتم کوهن‌ادد و سایرین به ازای هر بازه حافظه‌ی $O(1)$ مصرف می‌کند.

حال الگوریتمی داریم که اگر قطر در محدوده‌ی مورد نظر باشد پاسخ بله و در غیر این صورت پاسخ خیر می‌دهد. اگر فرض کنیم کوچک‌ترین قطر در بین تمام پنجره‌ها مقدار m و بزرگ‌ترین آن مقدار M باشد کافی است بازه‌ی $[m, M]$ را به بازه‌های کوچک‌تر با فاصله‌ی $(۱+\varepsilon)$ تقسیم کنیم. به طور دقیق‌تر بازه‌ی iام معادل است با $[m(۱+\varepsilon)^i, ۳m(۱+\varepsilon)^i)$ که تعداد بازه‌ها برابر می‌شود با $\log_{۱+\varepsilon}^{\frac{M}{m}}$. می‌دانیم $\frac{M}{m} = \mathcal{O}(R)$ پس مقدار حافظه‌ی کل این الگوریتم برابر با $\mathcal{O}(\frac{۱}{\varepsilon}\log R)$ خواهد بود.

۳ـ۳ kـ مرکز در مدل پنجره‌ی لغزان

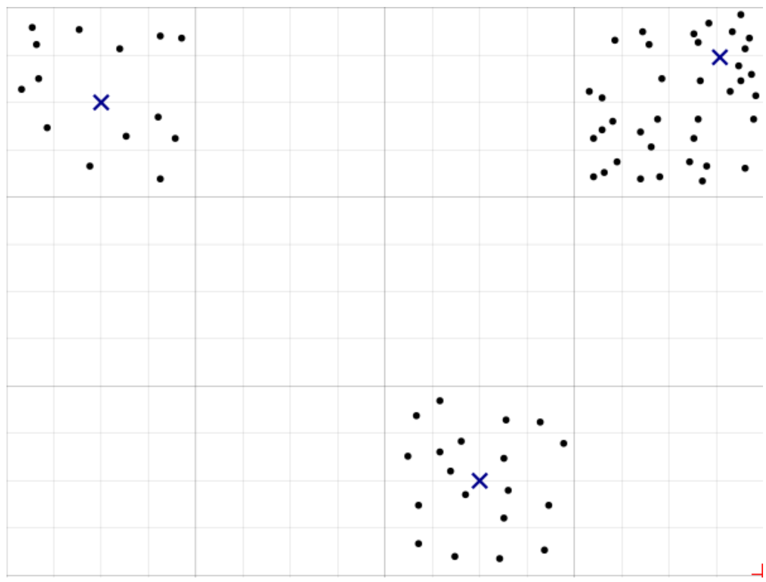
در این قسمت به بررسی مسئله‌ی kـ مرکز در مدل‌های داده‌های حجیم می‌پردازیم. با توجه به این که پژوهش زیادی در مدل پنجره‌ی لغزان صورت نگرفته است ابتدا مقداری از تاریخچه‌ی kـ مرکز در مدل جویبار داده می‌گوییم.

در مدل جویبار داده، مشکل اصلی عدم امکان نگه‌داری تمام داده‌ها در حافظه است و باید سعی شود تنها داده‌هایی را که ممکن است در ادامه مورد نیاز باشند نگه‌داریم. یک روش معمول برای کاهش حافظه‌ی مصرفی، نگه‌داری مجموعه‌ای از نمایندگان نقاط اصلی است تا جواب مسئله‌ی kـ مرکز را از آن به دست آوریم. قاعدتا به علت کاهش تعداد نقاط دقت پاسخ هم کاهش پیدا خواهد کرد. همان‌طور که در فصل قبل توضیح دادیم به چنین مجموعه‌ای، مجموعه‌ی هسته‌ی نقاط می‌گوییم.

در مدل جویبار داده بهترین مجموعه‌ی هسته برای حل مسئله‌ی kـ مرکز توسط ضرابی‌زاده ارائه شده است [۳۵]. حافظه‌ی این مجموعه از مرتبه‌ی $\mathcal{O}(\frac{k}{\epsilon^d})$ است. برای ایجاد این مجموعه‌ی هسته از دو ایده به صورت ترکیبی استفاده شده است. در گام اول از یک الگوریتم با ضریب تقریب ثابت استفاده می‌شود تا تخمینی از مراکز به دست بیاید. سپس نقاط حول هر مرکز را در یک توری با اندازه‌ی $\mathcal{O}(\frac{۱}{\epsilon})$ در هر بعد قرار می‌دهد. از این به بعد لزومی به نگه‌داری تمام داده‌ها نیست و سلول‌هایی که پر هستند می‌توانند به عنوان نمایندگان تمام نقاط استفاده شوند.

نمونه‌ای از اجرای الگوریتم ضرابی‌زاده در شکل ۳ـ۲ نشان داده شده است. [۱]. برای دیدن اثبات‌ها و توضیح بیش‌تر در مورد روش ارائه شده می‌توانید به مرجع [۳۵] مراجعه کنید.

در مدل پنجره‌ی لغزان پژوهش زیادی در رابطه با مسئله‌ی kـ مرکز نشده است. تنها مقاله‌ای که الگوریتمی برای محاسبه‌ی kـ مرکز ارائه داده است مقاله‌ی کهن‌ادد و سایرین [۷] است. رویکرد اصلی



شکل ۳ـ۲: نمونه‌ای از شبکه‌بندی الگوریتم ضرابی‌زاده (نقاط ضرب‌در شکل، مراکز به دست آمده از الگوریتم تقریبی است). پس از شبکه‌بندی کافی است برای هر کدام از خانه‌های شبکه‌بندی، تنها یکی را به عنوان نماینده در نظر بگیریم. [۱]

حل آن‌ها همانند محاسبه‌ی قطر متریک در مدل پنجره‌ی لغزان است. ضریب تقریب الگوریتم آن‌ها $6+\varepsilon$ است.

در الگوریتم محاسبه‌ی قطر در فضای متریک، بازه‌ی پاسخ را به زیربازه‌هایی تقسیم کردند. سپس برای هر زیربازه یک زیرالگوریتم تعیین می‌کرد که آیا پاسخ (قطر) در این بازه قرار دارد یا ندارد. برای این که متوجه بشوند قطر یک دنباله خارج از بازه است کافی بود تنها دو نقطه که فاصله‌شان بیش از حد مشخصی باشد را نگه‌داری کنیم. در رابطه با چگونگی متوجه‌شدن این که پاسخ در بازه‌ی مرد نظر نیست، مسئله‌ی kـ مرکز اندکی پیچیده‌تر است. زیرا اگر پاسخ خارج از بازه باشد باید حداقل $k+1$ نقطه‌ی دور از یک‌دیگر را نگه‌داری کرد.

الگوریتم ارائه‌شده توسط کهن‌ادد و سایرین در الگوریتم ۱ قابل مشاهده است. این الگوریتم به طور کامل از [۷] آورده شده است. مجموعه‌ی A معادل مراکزی است که الگوریتم حدس می‌زند خوشه‌ها

الگوریتم ۱ الگوریتم محاسبه‌ی پاسخ kـ مرکز در بازه‌ی مشخص در مدل پنجره‌ی لغزان

$\emptyset \to A, R, O$

به ازای هر نقطه‌ی ورودی p از جویبار داده:

اگر $q \in O$ منقضی شده بود:

q را از Oحذف کن.

اگر $a \in A$ منقضی شده بود:

فرآیند $\mathrm{DeleteAttraction}(a)$ را اجرا کن

فرآیند $\mathrm{Insert}(p)$ را اجرا کن

حول آن‌ها شکل می‌گیرند. اگر تعداد این مجموعه‌ها بیش از k باشد یعنی حداقل $k + ۱$ نقطه در پنجره وجود دارد که فاصله‌شان بیش از حدی است که پاسخ kـ مرکز در بازه‌ی مورد نظر باشد. مجموعه‌ی R نماینده‌ی جوان‌ترین نقاطی است که در خوشه‌های با مرکزیت نقاط A قرار دارند. مثلا اگر نقطه‌ای وارد شود که فاصله‌اش تا مرکزی کم‌تر از ۲γ باشد نماینده‌ی آن مرکز خواهد شد. نمایندگان مراکز از خود مراکز پیرتر نیستند. در نتیجه اگر مرکزی منقضی شود نماینده‌ی آن وجود دارد. پس از خروج مرکز، نماینده‌ی آن به مجموعه‌ی O منتقل می‌شود که مجموعه‌ی نمایندگان یتیم [۱۲] است.

اندازه‌ی مجموعه‌های A و O و R در الگوریتم ۱ از مرتبه‌ی $\mathcal{O}(k)$ است. در نتیجه حافظه‌ی مصرفی هر الگوریتم $\mathcal{O}(k)$ خواهد بود. با معرفی این الگوریتم تنها قسمت موازی‌سازی می‌ماند که کاملا مشابه قسمت موازی‌سازی در راه حل قطر است. یعنی بازه‌ی پاسخ را به بازه‌های $[m(۱+\varepsilon)^i, ۶m(۱+\varepsilon)^i)$ تقسیم می‌کنند و برای هر زیربازه یک زیرالگوریتم ۱ اجرا می‌کنند.

در نتیجه حافظه‌ی الگوریتم محاسبه‌ی kـ مرکز متریک در مدل پنجره‌ی لغزان ارائه‌شده توسط کهن‌ادد و سایرین از مرتبه‌ی $\mathcal{O}(k\frac{1}{\varepsilon}\log R)$ است.

۳ـ۳ـ۱ جمع‌بندی

در این فصل مسائل بهینه‌سازی هندسی در مدل‌های ایستا و پنجره‌ی لغزان را بررسی کردیم. میزان دقت الگوریتم‌هایی که برای مسائل بهینه‌سازی هندسی در مدل‌های مختلف ارائه شده‌اند در جدول ۳ـ۱ و

[۱۲] Orphan

الگوریتم ۲ رویه‌های الگوریتم محاسبه‌ی پاسخ kـ مرکز در بازه‌ی مشخص در مدل پنجره‌ی لغزان

```
رویه DeleteAttraction(a):
    O ∪ {R(a)} → O
    R \ {R(a)} → R
    A \ {a} → A
پایان رویه
رویه Insert(p):
    {a ∈ A | dis(p, a) ⩽ ۲ · γ} → D
    اگر D = ∅:
        A ∪ {p} → A
        p → R(p)
        R ∪ {R(p)} → R
        اگر |A| > k + ۱:
            argmax_{a∈A} age(a) → a_old
            DeleteAttraction(a_old)
        اگر |A| > k:
            max_{a∈A} age(a) → t
            به ازای q ∈ O:
                اگر age(q) > t:
                    O \ {q} → O
    در غیر این صورت:
        به ازای a ∈ D:
            Exchange R(a) with p in R
پایان رویه
```

هم‌چنین میزان حافظه‌ی آن‌ها در جدول ۳ــ۲ آمده است.

مسئله/مدل	جویبار داده	پنجره‌ی لغزان	پ.ل. (ابعاد بالا)	گردان‌در
عرض نقاط	$1+\varepsilon$ [۱۱]	$1+\varepsilon$ (۲ــبعدی) [۱۱]	–	$1+\varepsilon$ [۳۶]
قطر نقاط	$1+\varepsilon$ [۱۱]	$1+\varepsilon$ [۱۰]	$3+\varepsilon$ [۷]	$1+\varepsilon$ [۳۶]
Kــ مرکز	$2+\varepsilon$ [۳۵] [۳۷]	$6+\varepsilon$ [۷]	$6+\varepsilon$ [۷]	–
Kــ میانه	$1+\varepsilon$ [۲۹]	$O(1)$ [۳۱]	$O(1)$ [۳۱]	–

جدول ۳ــ۱: دقت الگوریتم‌های بهینه‌سازی هندسی در مدل‌های مختلف داده‌های حجیم

مسئله/مدل	جویبار داده	پنجره‌ی لغزان	پ.ل. (ابعاد بالا)	مدل گردان‌در
عرض نقاط	$\frac{(lgn)^d}{\varepsilon^{\frac{d-1}{2}}}$	$polylog(N,R,R')$	–	$(\frac{1}{\varepsilon^{d+1}}(logU)^{3d+4}log(\frac{1}{\delta\varepsilon})$
قطر نقاط	$\frac{lg\frac{1}{\varepsilon}}{\varepsilon^{\frac{d-1}{2}}}$	$\varepsilon^{-\frac{d+1}{2}}\frac{lnNlnR}{\varepsilon}$	$\frac{lnNlnR}{\varepsilon}$	$(\frac{1}{\varepsilon^{d+1}}(logU)^{3d+4}log(\frac{1}{\delta\varepsilon})$
Kــ مرکز	$\frac{k}{\varepsilon^d}$	$k\frac{lnNlnR}{\varepsilon}$	$k\frac{lnNlnR}{\varepsilon}$	–
Kــ میانه	$\frac{k}{\varepsilon^d}lgN$	k^3log^6N	k^3log^6N	–

جدول ۳ــ۲: میزان حافظه‌ی مورد نیاز الگوریتم‌های بهینه‌سازی هندسی در مدل‌های مختلف داده‌های حجیم

در این نتایج پژوهش‌های صورت‌گرفته در مدل گردان‌در را نیز آورده‌ایم. مدل گردان‌در از نظر ابعاد فضا ساده‌تر است (مختصات هر بعد عدد صحیح و بین ۱ تا U است) اما امکان درج و حذف نقاط به صورت پویا را دارد. در صورتی که یک نقطه را پس از گذشت N چرخه از ورودش حذف کنیم بسیار شبیه به مدل پنجره‌ی لغزان می‌شود. به همین دلیل مدل گردان‌در را نیز در مقایسه‌ی الگوریتم‌های مشابه آورده‌ایم تا دید بهتری پیدا کنیم. می‌توان گفت پارامتر U در مدل گردان‌در شباهت بسیار زیادی به پارامتر R در پنجره‌ی لغزان دارد.

فصل ۴

نتایج جدید

در این فصل بستری برای حل دسته‌ای از مسائل هندسی در مدل پنجره‌ی لغزان ارائه می‌دهیم. یکی از مهم‌ترین نتایج بستر ارائه‌شده حل مسئله‌ی kـ مرکز نقاط دوبعدی در مدل پنجره‌ی لغزان با ضریب تقریب $۲+\varepsilon$ و با استفاده از حافظه‌ی $\mathcal{O}(\frac{۱}{\varepsilon^{۲}})$ است. بهترین الگوریتم قبلی برای این مسئله دارای ضریب تقریب $۶+\varepsilon$ بوده است.

۴ـ۱ تعاریف اولیه

در این قسمت برخی مفاهیمی را که برای توضیح الگوریتم‌های این پژوهش به آن نیاز داریم، تعریف می‌کنیم. در این فصل dis تابع اندازه در فضای dـ بعدی است. به عبارت دیگر داریم:

$$\forall p,q \in \mathbb{R}^d \Rightarrow dis(p,q) = \sqrt{\sum_{۱\leqslant j\leqslant d} (p_j - q_j)^{۲}}$$

که p_j مختصات بعد jام نقطه‌ی p است.

تعریف ۴ـ۱ ***فضای سلولی:*** *فضای دکارتی dـ بعدی را در نظر بگیرید. فرض کنید هر محور آن را به فاصله‌های مساوی به اندازه‌ی δ تقسیم کرده‌ایم. توری [۱] حاصل نقاط را به تعدادی* ***سلول*** *افراز می‌کند. یک سلول مجموعه‌ی شامل نقاط واقع در*

$$[i_۱\delta,(i_۱+۱)\delta) \times [i_۲\delta,(i_۲+۱)\delta) \ldots \times [i_d\delta,(i_d+۱)\delta)$$

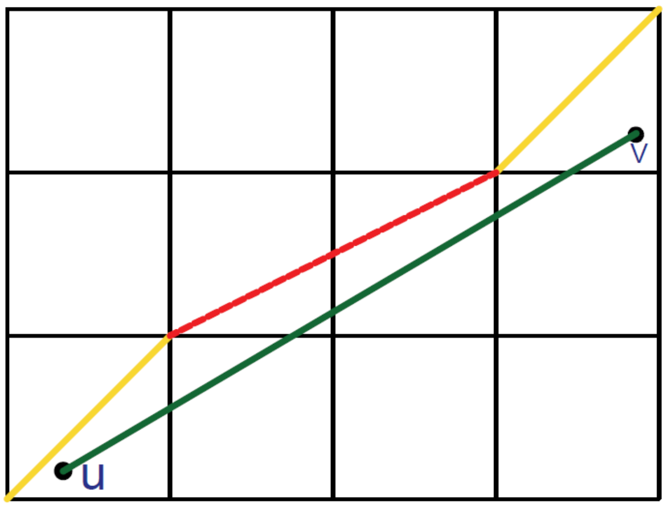
[۱] Grid

تعریف می‌کنیم و آن را با $(i_1, i_2, \ldots, i_d)$ نشان می‌دهیم. به ازای هر نقطه‌ی دلخواه p، سلول p معادل است با مجموعه‌ای که نقطه‌ی p را در بر دارد و به صورت C_p نشان می‌دهیم.

تعریف ۴ـ۲ *تابع فاصله‌ی سلولی:* *فرض کنید دو سلول C_u و C_v وجود دارند. در صورتی که کوتاه‌ترین فاصله‌ی گوشه‌های سلول‌های C_u و C_v را برابر $m(C_u, C_v)$ بدانیم فاصله‌ی آن دو سلول به صورت زیر تعریف می‌شود.*

$$g(C_u, C_v) \equiv m(C_u, C_v) + ۲\sqrt{d} \cdot \delta$$

می‌توانید مثالی از فاصله‌ی سلولی را در شکل ۴ـ۱ مشاهده کنید.



شکل ۴ـ۱: خط سبز نشان‌دهنده‌ی فاصله‌ی واقعی دو نقطه، مجموع خطوط قرمز (کوتاه‌ترین فاصله‌ی دو سلول) و زرد (قطر سلول) برابر با فاصله‌ی سلول‌های متناظر نقاط است.

در طول فصل برای تعیین عرض سلول‌ها، δ، فضاهای سلولی از دو پارامتر دیگر به نام‌های α و ε استفاده می‌کنیم (که در آینده به شرح آن‌ها خواهیم رسید). از این پس فرض می‌کنیم هر فضای سلولی با دو پارامتر α و ε شناسایی می‌شود و داریم:

$$\delta = \frac{\alpha\varepsilon}{۲\sqrt{d}}$$

حال برخی از خواص فضای سلولی را شرح می‌دهیم.

لم ۴ـ۱ *به ازای دو سلول دلخواه C_u و C_v و هر دو نقطه‌ی p و q که $p \in C_u$ و $q \in C_v$ داریم*

$$g(C_u, C_v) \geqslant dis(p, q)$$

اثبات. *مقدار $\sqrt{d} \cdot \delta$ برابر با بیش‌ترین فاصله‌ی بین گوشه‌های یک سلول است. طبق تعریف، فاصله‌ی سلولی برابر با کم‌ترین فاصله‌ی گوشه‌های دو سلول به اضافه‌ی دو بیش‌ترین فاصله‌ی درون سلول است. در نتیجه به علت اصل نامساوی مثلثی بزرگ‌ترین فاصله‌ی بین گوشه‌های سلول (که از هر $dis(p, q)$ بیش‌تر است) از مقدار $g(C_u, C_v)$ کم‌تر است. در نتیجه لم اثبات می‌شود.* □

یکی از خواص مهم فضای سلولی که در قسمت‌های بعدی از آن استفاده می‌کنیم تقریب فواصل بیش از $\alpha(۱ + \varepsilon)$ است. در لم ۴ـ۲ این خاصیت مهم را معرفی و اثبات می‌کنیم.

لم ۴ـ۲ *به ازای هر دو نقطه‌ی u و v دو ویژگی زیر را داریم:*

- *اگر $dis(u, v) \leqslant \alpha(۱ + \varepsilon)$ آن‌گاه $g(C_u, C_v) \leqslant \alpha(۱ + \varepsilon)^۲$*
- *اگر $dis(u, v) > \alpha(۱ + \varepsilon)$ آن‌گاه $g(C_u, C_v) \leqslant dis(u, v)(۱ + \varepsilon)$*

اثبات. *برای ویژگی اول می‌دانیم $m(C_u, C_v) \leqslant dis(u, v) \leqslant \alpha(۱ + \varepsilon)$ و در نتیجه*

$$g(C_u, C_v) = m(C_u, C_v) + ۲\sqrt{d}\delta = m(C_u, C_v) + \alpha\varepsilon \leqslant \alpha(۱ + ۲\varepsilon) \leqslant \alpha(۱ + \varepsilon)^۲$$

در حالت دوم طبق لم ۴ـ۱ می‌دانیم

$$g(C_u, C_v) = m(C_u, C_v) + ۲\sqrt{d}\delta = m(C_u, C_v) + \alpha\varepsilon \geqslant dis(u, v) > \alpha(۱ + \varepsilon)$$

که نتیجه می‌دهد $m(C_u, C_v) \geqslant \alpha$. و در نتیجه داریم

$$g(C_u, C_v) = m(C_u, C_v) + \alpha\varepsilon \leqslant m(C_u, C_v)(۱ + \varepsilon) \leqslant dis(u, v)(۱ + \varepsilon)$$

و به این ترتیب لم اثبات می‌شود. □

طبق لم ۴ـ۲ اگر فاصله‌ی هر دونقطه‌ای از مقدار مشخصی بیش‌تر باشد ، فاصله‌ی سلولی تقریبی $۱ + \varepsilon$ از فاصله‌ی آن دو نقطه می‌دهد و اگر کم‌تر باشد فاصله‌ی سلولی کران بالا دارد. این محدودیت را در مشاهده‌ی ۴ـ۳ بیان کرده‌ایم.

مشاهده‌ی ۴ـ۳ *در صورتی که قطر n نقطه‌ی دلخواه برابر $r \leqslant \alpha(1+\varepsilon)$ باشد و بزرگ‌ترین فاصله‌ی آن نقاط توسط تابع فاصله‌ی g برابر با w باشد آن‌گاه داریم:* $w \leqslant \alpha(1+\varepsilon)^2$

در صورتی که بخواهیم فاصله‌ی بین هر زوج نقطه در یک مجموعه از نقاط را داشته باشیم نیازمند نگه‌داری تمامی آن نقاط هستیم. فضای سلولی به ما این امکان را می‌دهد تا مجموعه‌ای از نقاط را به یک سلول نگاشت دهیم. با این که نمی‌توانیم فاصله‌ی دقیق هر دو نقطه را به وسیله‌ی سلول‌های متناظر آن به دست اوریم اما در عوض می‌توانیم به جای نگه‌داری حجم زیادی از نقاط تنها سلول‌های متناظر آن‌ها را ذخیره کنیم که حافظه‌ی بسیار کم‌تری نیاز دارد. در لم ۴ـ۴ خاصیت مهمی از فضای سلولی را معرفی می‌کنیم که باعث می‌شود حجم سلول‌های مجموعه‌ای از نقاط، مستقل از تعداد خود نقاط باشد.

لم ۴ـ۴ *اگر بتوان n نقطه را به C (**عدد ثابت**) دسته افراز کرد که قطر هر دسته $\mathcal{O}(\alpha(1+\varepsilon))$ باشد، تعداد سلول‌های متناظر آن نقاط از مرتبه‌ی $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ خواهد بود.*

اثبات. *اگر قطر نقاط هر دسته $\mathcal{O}(\alpha(1+\varepsilon))$ باشد طبق مشاهده‌ی ۴ـ۳ قطر سلول‌های متناظر این نقاط از $\mathcal{O}(\alpha(1+\varepsilon)^2)$ خواهد بود که با توجه به این که عرض سلول‌ها برابر است با $\frac{\alpha\varepsilon}{2\sqrt{d}}$ در نتیجه تمام نقاط در کره‌ای از سلول‌ها که قطر آن از $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon})$ سلول تشکیل می‌شود قرار می‌گیرند. چون ابعاد فضا d است، تعداد کل سلول‌های این محدوده از مرتبه‌ی $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ خواهد بود. با توجه به این که تعداد ثابتی دسته (C) داریم در مرتبه‌ی حافظه تغییری ایجاد نمی‌شود و تعداد سلول‌های کل از مرتبه‌ی $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ خواهد شد.* □

تعریف ۴ـ۳ ***مسئله‌ی بهینه‌سازی Cـ پوشا:*** *فرض کنید π یک مسئله‌ی بهینه‌سازی باشد که ورودی آن n نقطه است. هم‌چنین ρ تابع هدف مسئله‌ی π است . مسئله‌ی π را Cـ پوشا می‌نامیم اگر دو شرط زیر را داشته باشد.*

- ***یک‌نوایی:*** *پاسخ هر زیرمجموعه‌ای از تمام نقاط کوچک‌تر مساوی پاسخ تمام نقاط باشد. به عبارت دیگر داشته باشیم*

$$\forall Q \subseteq P : \rho(Q) \leqslant \rho(P)$$

- *به ازای مجموعه‌ی نقاط P ، می‌توان P را به C دسته افراز کرد که فطر هر دسته از مرتبه‌ی $\mathcal{O}(\rho(P))$ باشد.*

به عنوان مثال مسئله‌ی محاسبه‌ی قطر نقاط یک مسئله‌ی ۱ـ پوشا است. همین‌طور kـ مرکز هندسی یک مسئله‌ی kـ پوشا است زیرا اگر به ازای پاسخ بهینه، هر مرکز و نقطه‌هایی که به آن مرکز از همه نزدیک‌ترند را در یک دسته قرار دهیم، k دسته خواهیم داشت که قطر هر دسته حداکثر برابر با پاسخ بهینه است.

حال مفاهیم مربوط به جویبار داده و پنجره‌ی لغزان که در این فصل استفاده می‌کنیم را معرفی می‌کنیم. همان‌طور که در فصل‌های قبل گفتیم مدل پنجره‌ی لغزان از مدل جویبار داده مشتق شده است که نقاط یکی پس از دیگری وارد می‌شود. هم‌چنین نقاط به ترتیبی که وارد شدند منقضی می‌شوند و در هر لحظه دنباله‌ای از نقاط داریم (پنجره) که می‌خواهیم تابع هدف را برای آن محاسبه کنیم. پس از ورود jامین نقطه از جویبار داده (در چرخه‌ی jام)، P_j را برابر دنباله‌ی آخرین N نقطه تا چرخه‌ی j در نظر می‌گیریم. به عبارت دیگر P_j دنباله‌ی نقاط $j - N + ۱$ تا jام است (اگر $j < N$ آن‌گاه P_j را تمام نقاط از ابتدا تا j در نظر می‌گیریم). $P_j[k]$ را برابر با kمین عضو P_j در نظر می‌گیریم (هر چقدر k کوچک‌تر باشد آن نقطه پیرتر است یا زودتر وارد شده است). به زبان دیگر P_j پنجره‌ی نقاط معتبر پس از چرخه‌ی j است.

۴ـ۲ چارچوب حل مسائل Cـ پوشا در پنجره‌ی لغزان

در این قسمت الگوریتمی برای حل یک مسئله‌ی Cـ پوشا در فضای dـ بعدی مدل پنجره‌ی لغزان ارائه می‌کنیم. الگوریتم ما دارای ضریب تقریب $(۱ + \varepsilon)$ است و از حافظه‌ی $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ استفاده می‌کند. برای توضیح الگوریتم ابتدا تعاریف زیر را در نظر می‌گیریم.

دنباله‌ی P را برابر با نقاط معتبر داخل پنجره و دنباله‌ی S را برابر دنباله‌ای از سلول‌هایی که الگوریتم نگه‌داری می‌کند در نظر می‌گیریم فرض کنید $\rho(P)$ پاسخ مسئله‌ی π برای نقاط P و تابع $\rho_C(S)$ پاسخ مسئله‌ی π برای سلول‌های S را محاسبه می‌کند.

تعریف ۴ـ۴ (نقاط بد) *نقطه‌ی $p \in P$ را یک نقطه‌ی بد می‌گوییم اگر پاسخ مسئله‌ی π برای نقاط جوان‌تر از p به همراه خود p بیش‌تر از $\alpha(۱ + \varepsilon)$ باشد. به عبارت دیگر اگر مجموعه‌ی نقاط جوان‌تر از p به همراه خود p را PB_p در نظر بگیریم خواهیم داشت:* $\rho(B_p) > \alpha(۱ + \varepsilon)^۲$

تعریف ۴ـ۵ (سلول بد) *به جوان‌ترین سلول در S (S)سلول بد می‌گوییم که اگر پاسخ مسئله‌ی π برای*

سلول‌های جوان‌تر از s به همراه خود s بیش‌تر از $\alpha(1+\varepsilon)$ باشد. به عبارت دیگر اگر مجموعه‌ی سلول‌های جوان‌تر از s به همراه خود s را SB_s در نظر بگیریم خواهیم داشت: $\rho_C(SB_s) > \alpha(1+\varepsilon)^2$

حال سراغ روش پیشنهادی این پژوهش می‌رویم. برای حل مسئله‌ی π در مدل پنجره‌ی لغزان از تکنیک موازی‌سازی استفاده می‌کنیم. مسئله‌ی تصمیم π را به صورت زیر تعریف می‌کنیم.

مسئله‌ی ۴ـ۱ (تصمیم Cـ پوشا) *پارامترهای ورودی α و ε داده شده است. مسئله‌ی تصمیم Cـ پوشا (یا تصمیم π) معادل است با این که با توجه به $\rho(P)$ به سوال‌های زیر پاسخ دهیم:*

- *در صورتی که $\rho(P)$ کوچک‌تر از $\alpha(1+\varepsilon)$ بود پاسخ بله خروجی بده.*
- *در صورتی که $\rho(P)$ بیش‌تر از $\alpha(1+\varepsilon)^2$ بود پاسخ خیر خروجی بده.*
- *در غیر این صورت پاسخ بله یا خیر خروجی بده.*

به عبارت دیگر این مسئله‌ی تصمیم روی بازه‌ای متمرکز شده است که پاسخ در آن باشد. اهمیت پارامترهای α و ε هم این‌جا مشخص می‌شود. α تخمینی از پاسخ و ε معیاری برای بازه‌ی خطای این تخمین است. می‌خواهیم حل‌کننده‌ای طراحی کنیم که به مسئله‌ی تصمیم π پاسخ دهد. شرح روش پیشنهادی ما در الگوریتم ۳ آمده است.

در گام اول ثابت می‌کنیم مجموعه‌ی سلول‌هایی که الگوریتم ۳ در S نگه‌داری می‌کند برای حل مسئله کفایت می‌کند. این موضوع را به وسیله‌ی قضیه‌ی ۴ـ۵ بیان می‌کنیم:

قضیه‌ی ۴ـ۵ *در هر زمان j اگر $\rho_C(S_j) \leqslant \alpha(1+\varepsilon)^2$ آن‌گاه سلول‌های متناظر با تمام نقاط P_j در S وجود دارد و اگر $\rho_C(S_j) > \alpha(1+\varepsilon)^2$ آن‌گاه سلول‌های متناظر نقاطی از P_j که پیرتر از سلول بد S_j نیستند در S وجود دارد.*

برای اثبات این قضیه ابتدا چند لم را ثابت می‌کنیم.

لم ۴ـ۶ *در زمان $j \leqslant N$ اگر $\rho_C(S_j) \leqslant \alpha(1+\varepsilon)^2$ آن‌گاه نماینده‌ی تمام نقاط P_j در S وجود دارد.*

الگوریتم ۳ الگوریتم حل مسئله‌ی تصمیم Cـ پوشا در مدل پنجره‌ی لغزان

$S = \emptyset$ (دنباله‌ی سلول‌های معتبر)
α, ε = پارامترهای ورودی
به ازای هر نقطه‌ی ورودی p از جویبار داده:
    سلول‌های منقضی‌شده را از S حذف کن
    اگر C_p در S بود آن را حذف کن.
    C_p را (به همراه زمان ورودش) اضافه کن.
    اگر $\rho_C(S) \leqslant \alpha(1+\varepsilon)^2$:
        پاسخ بله را برگردان
    در غیر این صورت:
        w را سلول بد در S در نظر بگیر.
        سلول‌های قبل از w را از S حذف کن.
        پاسخ خیر را برگردان

اثبات. *با توجه به این که در زمان j هیچ نقطه‌ای منقضی نشده است و به خاطر خاصیت یک‌نوایی مسئله‌ی Cـ پوشا، به ازای $t \leqslant j$ داریم $\rho_C(S_t) \leqslant \alpha(1+\varepsilon)^2$ پس تا این زمان هیچ‌گاه به حذف سلول‌های قبل از سلول بد نرسیده‌ایم و اگر سلولی درالگوریتم حذف شده است دوباره همان سلول با زمان ورود جدید به آن اضافه شده است پس به ازای تمام نقاط سلول‌های متناظر آن‌ها نیز موجود است.* □

لم ۴ـ۷ در زمان $N \geqslant j$ اگر داشته باشیم $\rho_C(S_j) > \alpha(1+\varepsilon)^2$ *آن‌گاه سلول‌های متناظر نقاطی از P_j که پیرتر از سلول بد S_j نیستند در S وجود دارد.*

اثبات. *برای اثبات این لم از برهان خلف استفاده می‌کنیم. اولین $t < N$ را در نظر می‌گیریم که شرط لم را نقض می‌کند. سلول بد S_t را w در نظر می‌گیریم. به ازای این t یعنی سلولی ناپیرتر از w وجود دارد که در S_t وجود ندارد. اگر t اولین زمانی است که $\rho_C(S_t) > \alpha(1+\varepsilon)^2$ پس طبق لم ۴ـ۶ سلول‌های متناظر تمام نقاط P_{t-1} در S_{t-1} وجود دارد حال با اضافه‌شدن نقطه‌ی جدید تنها سلول‌هایی از S_t حذف شدند که قبل از w یا به قولی پیرتر از آن بوده‌اند. در غیر این صورت این حالت باید در چرخه‌ی قبلی (یعنی زمان $t-1$) هم رخ داده باشد و داریم $\rho_C(S_{t-1}) > \alpha(1+\varepsilon)^2$ که طبق فرض سلول‌های متناظر*

تمام نقاطی از P_j که از سلول بد S_{t-1} پیرتر نیستند، در S_{t-1} وجود دارد. حال با اضافه‌شدن نقطه‌ی جدید، w قطعا ناپیرتر از سلول بد S_{t-1} است. پس تنها سلول‌هایی از S_t حذف می‌شوند که قبل از w یا به قولی پیرتر از آن بوده‌اند. پس چنین t وجود ندارد و لم اثبات می‌شود. $\square$

لم ۴ــ۸ *در زمان $j > N$ اگر $\rho_C(S_j) \leqslant \alpha(1+\varepsilon)^2$ آن‌گاه سلول‌های متناظر تمام نقاط P_j در S وجود دارد.*

اثبات. *برای اثبات این موضوع از استقرا استفاده می‌کنیم. پایه‌ی استقرا در زمان N درست است (طبق لم ۴ـ۶). فرض می‌کنیم شرط لم در زمان $j-1$ برقرار بوده است. با ۲ حالت مواجه می‌شویم.*

- *حالت اول $\rho_C(S_{j-1}) \leqslant \alpha(1+\varepsilon)^2$: در زمان $j-1$ طبق فرض می‌دانیم که سلول‌های متناظر تمام نقاط P_{j-1} در S_{j-1} موجود است و اگر سلول متناظر اولین نقطه‌ی P_{j-1} در S_{j-1} موجود باشد در S_j حذف می‌شود و پس از ورود نقطه‌ی جدید، در خط ۴ سلول دیگری حذف نمی‌شود. پس سلول‌های متناظر تمام نقاط P_j در S_j موجود است.*
- *حالت دوم $\rho_C(S_{j-1}) > \alpha(1+\varepsilon)^2$: در این حالت سلول بد S_{j-1} باید برابر اولین یا پیرترین سلول آن باشد چون در غیر این صورت این نقطه در S_j هم می‌ماند و شرط $\rho_C(S_j) \leqslant B\alpha(1+\varepsilon)^2$ نقض می‌شود. حال که پیرترین سلول S_{j-1} سلول بد است طبق فرض می‌دانیم که سلول‌های متناظر تمامی نقاط P_{j-1} که پیرتر از سلول بد در S_{j-1} نیستند در S_{j-1} وجود دارد پس با ورود نقطه‌ی جدید و پس از خط ۴ می‌دانیم سلول‌های متناظر تمام نقاط P_j در S موجود است.*

به این ترتیب حکم استقرا اثبات می‌شود. $\square$

لم ۴ــ۹ *در زمان $j > N$ اگر $\rho_C(S) > \alpha(1+\varepsilon)^2$ آن‌گاه سلول‌های متناظر نقاطی از P_j که پیرتر از سلول بد S_j نیستند در S وجود دارد.*

اثبات. *برای اثبات این موضوع از استقرا استفاده می‌کنیم. پایه برای زمان N درست است (طبق لم ۴ـ۷). فرض می‌کنیم شرط لم در زمان $j-1$ برقرار بوده است. با ۲ حالت مواجه می‌شویم.*

- *حالت اول $\rho_C(S_{j-1}) \leqslant \alpha(1+\varepsilon)^2$: در زمان $j-1$ طبق فرض می‌دانیم که سلول‌های متناظر تمام نقاط P_{j-1} در S_{j-1} موجود است و اگر سلول متناظر اولین نقطه‌ی P_{j-1} در S_{j-1} موجود*

باشد در S_j حذف می‌شود و پس از اضافه‌شدن نقطه‌ی جدید (پس از خط ۶) S_j شامل نمایندگان تمامی نقاط P_j خواهد بود. پس از اجراشدن خط ۱۱ ، سلول‌های متناظر تمام نقاطی از P_j که از سلول بد S_j پیرتر نیستند، در S_j باقی خواهند ماند.

- **حالت دوم** $\rho_C(S_{j-1}) > \alpha(1+\varepsilon)^2$: *در این حالت سلول بد S_{j-1} یا برابر اولین یا پیرترین سلول آن است که پس از ورود نقطه‌ی جدید و اجرای خط ۴ (حذف این سلول) تمام سلول‌های متناظر P_j در S وجود خواهند داشت یا این طور نیست که پس از ورود نقطه‌ی جدید، سلول بد نمی‌تواند ناپیرتر از سلول بد S_{j-1} باشد و نمایندگان تمامی نقاط ناپیرتر از سلول بد S_{j-1} در S_j وجود دارد. حال پس از اجرای خط ۱۱ سلول‌های متناظر تمام نقاطی از P_j که از سلول بد S_j پیرتر نیستند، در S_j باقی خواهند ماند.*

به این ترتیب حکم استقرا اثبات می‌شود. □

حالت‌های مختلف زمانی و پاسخ الگوریتم را در چندین لم بررسی کردیم. حال می‌توانیم قضیه‌ی ۴ـ۵ را اثبات کنیم.

اثبات. [**قضیه‌ی ۴ـ۵**]
با استفاده از لم‌های ۴ـ۶ و ۴ـ۸ حالت اول قضیه‌ی ۴ـ۵ (کفایت سلول‌ها در پاسخ بله) و با استفاده از لم‌های ۴ـ۷ و ۴ـ۹ حالت دوم قضیه‌ی ۴ـ۵ (کفایت سلول‌ها در پاسخ خیر) اثبات می‌شود. □

حال که طبق قضیه‌ی ۴ـ۵ می‌دانیم سلول‌هایی که نگه‌داری می‌شوند برای پاسخ‌گویی به مسئله‌ی تصمیم π کافی هستند ثابت می‌کنیم الگوریتم ۳ به این مسئله به درستی پاسخ می‌دهد. بیان این موضوع در قضیه‌ی ۴ـ۱۰ آمده است:

قضیه‌ی ۴ـ۱۰ *اگر* $\rho(P) \leqslant \alpha(1+\varepsilon)^2$ *آن‌گاه داریم* $\rho_C(S) \leqslant \alpha(1+\varepsilon)^2$ *و اگر داشته باشیم* $\rho(P) > \alpha(1+\varepsilon)$ *آن‌گاه داریم* $\rho_C(S) > \alpha(1+\varepsilon)^2$

اثبات. *برای اثبات این قضیه کافی است حالات مختلف* $\rho(P)$ *را بررسی کنیم.*

- *اگر* $\rho(P) \leqslant \alpha(1+\varepsilon)$ *آن‌گاه* $\rho_C(S) \leqslant \alpha(1+\varepsilon)^2$.

برای اثبات درستی این حالت فرض خلف می‌کنیم که در چنین حالتی داشته باشیم $\rho_C(S) > \alpha(1+\varepsilon)^2$ آن‌گاه طبق قضیه‌ی ۴ــ۵ تمام نقاطی از P_j که از نقطه‌ی بد S_j پیرتر نیستند در S وجود دارد در نتیجه $\rho(P) > \alpha(1+\varepsilon)$ که خلاف فرض اولیه است و این حالت درست است.

- *اگر $\rho(P) > \alpha(1+\varepsilon)^2$ آن‌گاه $\rho_C(S_j) > B\alpha(1+\varepsilon)^2$.*
  اثبات مشابه حالت اول.

- *اگر $\alpha(1+\varepsilon) < \rho(P) \leqslant \alpha(1+\varepsilon)^2$ آن‌گاه $\rho(P) \leqslant \rho_C(S_j) \leqslant \rho(P)(1+\varepsilon)$*
  اثبات مشابه حالت اول.

تمامی حالات بررسی شدند پس قضیه اثبات می‌شود. □

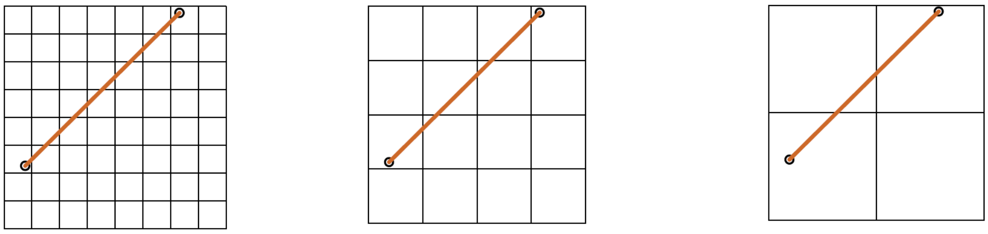
به این ترتیب ثابت می‌شود که الگوریتم ۳ در صورتی که قطر نقاط درون پنجره از $\alpha(1+\varepsilon)$ کوچک‌تر باشد پاسخ بله می‌دهد و اگر قطر نقاط درون پنجره بزرگ‌تر از $\alpha(1+\varepsilon)^2$ باشد پاسخ خیر می‌دهد. در نتیجه این الگوریتم مسئله‌ی تصمیم π را به درستی حل می‌کند.

حال که می‌دانیم روش ما درست کار می‌کند مقدار حافظه‌ی الگوریتم ۳ را محاسبه می‌کنیم.

قضیه‌ی ۴ــ۱۱ *الگوریتم ۳ به مسئله‌ی تصمیم π پاسخ درست می‌دهد و حافظه‌ی مصرفی آن از مرتبه‌ی $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ است.*

اثبات. *درستی کارکرد الگوریتم در قضیه‌ی ۴ــ۱۰ اثبات شد. برای محاسبه‌ی حافظه‌ی مصرفی کافی است اندازه‌ی دنباله‌ی S را به دست آوریم. در هر چرخه اگر داشته باشیم $\rho_C(S) \leqslant \alpha(1+\varepsilon)^2$ آن‌گاه طبق خاصیت مسئله‌ی $C-$پوشا می‌توان نقاط را به C دسته تقسیم‌بندی کرد که قطر هر دسته از $\mathcal{O}(\alpha(1+\varepsilon)^2)$ باشد. طبق ویژگی فضای سلولی (لم ۴ــ۴)حافظه‌ی مورد نیاز $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ است. و در صورتی که داشته باشیم $\rho_C(S) > \alpha(1+\varepsilon)^2$ طبق الگوریتم می‌دانیم تنها سلول‌های متناظر نقاطی از P_j که پیرتر از سلول بد S_j نیستند در S وجود دارد. اگر سلول بد را w در نظر بگیریم آن‌گاه $S-w$ در مسئله‌ی $C-$پوشا صدق می‌کند و مثل قسمت قبل اندازه‌ی S از $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ خواهد بود. با توجه به این که C دسته از این سلول‌ها وجود دارد و C ثابت است اندازه‌ی S و حافظه‌ی مصرفی الگوریتم برابر با $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ خواهد بود.* □

تا به این‌جا یک الگوریتم حل مسئله‌ی تصمیم π داشتیم که با ورودی گرفتن α و ε تشخیص می‌داد پاسخ بزرگ‌تر از $\alpha(1+\varepsilon)^2$ است. برای مسئله‌ی Cــپوشا در مدل پنجره‌ی لغزان می‌دانیم نسبت



الف) ب) پ)

شکل ۴ـ۲: فضای سلولی سه نمونه‌ی متوالی از زیرالگوریتم ۳ به ازای $\varepsilon = 1$. در شکل الف دقت تقریب بالا است و انتظار داریم پاسخ در محدوده‌ی کوچکی قرار گرفته باشد. در شکل ب پیش‌بینی می‌شود پاسخ در محدوده‌ی بزرگ‌تری ($1 + \varepsilon$ یا دو برابر)قرار دارد پس دقت کاهش می‌یابد. این قضیه در مورد نمونه‌ی پ نسبت به نمونه‌ی ب هم دیده می‌شود.

بزرگ‌ترین پاسخ (M) به کوچک‌ترین پاسخ (m) در هر پنجره‌ای برابر با R است. پس کافی است بازه‌ی $[m, M]$ را به زیربازه‌هایی تقسیم کنیم و هر زیربازه را به یک الگوریتم تصمیم اختصاص دهیم. تنها چیزی که در این الگوریتم‌ها متفاوت است ورودی‌های فضای سلولی آن‌ها است که به صورت زیر به آن مقدار می‌دهیم.

$$\forall i \in \{1, \dots, \log^{R}_{1+\varepsilon}\} : \alpha_i = m(1+\varepsilon)^i$$

یعنی به نمونه‌ی شماره‌ی i الگوریتم ۳ ورودی α_i و ε می‌دهیم. مثالی از فضای سلولی نمونه‌های مختلف زیرالگوریتم به ازای $\varepsilon = 1$ در شکل ۴ـ۲ قابل مشاهده است.

جزییات پیاده‌سازی این الگوریتم را می‌توانید در الگوریتم ۴ مشاهده کنید.

حال نتیجه‌ی این الگوریتم را بازنویسی می‌کنیم.

قضیه‌ی ۴ـ۱۲ *الگوریتم ۴ پاسخ مسئله‌ی π برای نقاط داخل پنجره را با ضریب تقریب $(1 + \varepsilon)$ به دست می‌آورد و حافظه‌ی مصرفی آن از مرتبه‌ی $\log R\frac{\sqrt{d}}{\varepsilon^{d+1}}$ است.*

اثبات. *با توجه به این که مقدار پاسخ مسئله‌ی π در بازه‌ی $[m, M]$ قرار دارد و تمامی این بازه پوشیده شده است، حتما نمونه‌ای از الگوریتم‌های تصمیم وجود دارد که پاسخ بله بدهد. زیرا اگر آخرین نمونه‌ی*

الگوریتم ۴ الگوریتم محاسبه‌ی پاسخ مسئله‌ی Cـ پوشا در مدل پنجره‌ی لغزان

به ازای هر $i \in \{۱, \dots, \log_{۱+\varepsilon}^{R}\}$:
  یک نمونه از زیرالگوریتم ۳ با پارامترهای ورودی α_i و ε ایجاد کن.
به ازای هر نقطه‌ی ورودی p از جویبار داده:
  به ازای هر$\{۱, \dots, \log_{۱+\varepsilon}^{R}\}$ $i \in$:
    نقطه‌ی p را در نمونه‌ی i زیرالگوریتم۳ درج کن.
  اگر نمونه‌ی j اولین نمونه‌ای بود که پاسخ بله داشت:
    مقدار $\alpha_j(۱+\varepsilon)^۲$ را به عنوان خروجی برگردان.

الگوریتمی که با α معادل M اجرا می‌شود را در نظر بگیریم، اگر پاسخ مسئله از $\alpha(۱+\varepsilon)$ کوچک‌تر باشد بله را خروجی می‌دهد. فرض کنید نمونه‌ی i الگوریتم پاسخ بله را خروجی می‌دهد (مقدار S در نمونه‌ی الگوریتم i را AS_i می‌نامیم). یعنی $\rho_C(AS_i) \leqslant \alpha_i(۱+\varepsilon)^۲$ طبق قضیه‌ی ۴ـ۱۰ داریم $\rho(P) \leqslant \alpha_i(۱+\varepsilon)^۲$ و چون نمونه‌ی $i-۱$ الگوریتم ۳ پاسخ را پیدا نکرده است پس داریم $\rho_C(AS_{i-۱}) > \alpha_{i-۱}(۱+\varepsilon)^۲$ طبق قضیه‌ی ۴ـ۱۰ داریم $\rho(P) > \alpha_{i-۱}(۱+\varepsilon)$. از این دو مورد می‌توانیم نتیجه‌گیری کنیم

$$\alpha_{i-۱}(۱+\varepsilon) < \rho(P) \leqslant \alpha_i(۱+\varepsilon)^۲$$

$$\rightarrow m(۱+\varepsilon)^i < \rho(P) \leqslant m(۱+\varepsilon)^{i+۲}$$

یعنی پاسخی که ما خروجی می‌دهیم حداکثر $(۱+\varepsilon)^۲$ برابر پاسخ بهینه است. برای این که ضریب تقریب برابر $(۱+\varepsilon)$ باشد مقدار $\frac{\varepsilon}{۳}$ را به عنوان ورودی الگوریتم می‌دهیم زیرا داریم

$$(۱+\frac{\varepsilon}{۳})^۲ = ۱ + \frac{۲}{۳}\varepsilon + \frac{\varepsilon^۲}{۹} \leqslant (۱+\varepsilon)$$

همچنین با توجه به این که $\log_{۱+\varepsilon}^{R}$ نمونه‌ی الگوریتم داریم و طبق قضیه‌ی ۴ـ۱۱ حافظه‌ی مصرفی هر نمونه برابر است با $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ پس حافظه‌ی کل می‌شود $\mathcal{O}(\log_{۱+\varepsilon}^{R} \frac{\sqrt{d}}{\varepsilon^d})$ با توجه به این که می‌دانیم $\log_{۱+\varepsilon}^{R} \leqslant \frac{\log R}{\varepsilon}$ می‌توانیم مرتبه‌ی حافظه را به صورت $\mathcal{O}(\log R \frac{d}{\varepsilon^{d+۱}})$ بازنویسی کنیم. □

تنها موردی که تا به حال مورد بحث قرار نگرفته است مرتبه‌ی زمانی اجرای الگوریتم است. علت این امر وابستگی مرتبه‌ی زمانی الگوریتم ۳ به زمان مصرفی الگوریتم ρ_C است. این مورد را در قسمت بعد که نمونه‌هایی از مسائل Cـ پوشا را حل می‌کنیم بررسی می‌کنیم.

در این قسمت با چارچوب حل مسائل Cـ پوشا آشنا شدیم و الگوریتم ۴ را برای حل این مسائل معرفی کردیم. در قسمت بعد با معرفی چندین مسئله‌ی Cـ پوشا آن‌ها را به وسیله‌ی چارچوب معرفی‌شده حل می‌کنیم. سپس نتیجه‌ی خودمان را با الگوریتم‌های موجود دیگران از جنبه‌های ضریب تقریب، حافظه‌ی مصرفی و زمان مصرفی مقایسه می‌کنیم.

۴ـ۳ تحلیل تعدادی از مسائل Cـ پوشا

در قسمت قبل الگوریتم چارچوب حل مسائل Cـ پوشا در مدل پنجره‌ی لغزان را معرفی کردیم. در این فصل می‌خواهیم نمونه‌هایی از مسائل Cـ پوشا را در این چارچوب حل کنیم و با راه حل‌هایی که برای آن ارائه شده است مقایسه کنیم. مهم‌ترین نتیجه‌ی این چارچوب حل مسائل Cـ پوشا با استفاده از تمامی الگوریتم‌های شناخته‌شده و جدید بهینه‌سازی هندسی در مدل ایستا است. به عبارت دیگر اگر الگوریتمی سریع برای حل دقیق یک مسئله در مدل ایستا وجود داشته باشد می‌توانیم همان مسئله را در مدل پنجره‌ی لغزان حل کنیم.

۴ـ۳ـ۱ قطر

برای شروع مسئله‌ی محاسبه‌ی قطر را در نظر بگیرید. این مسئله جزو مسائل ۱ـ پوشا به شمار می‌آید. برای حل این مسئله به وسیله‌ی چارچوبی که ارائه کردیم تنها نیاز به روشی برای محاسبه‌ی قطر به صورت دقیق در مدل ایستا نیاز داریم. یکی از ساده‌ترین روش‌هایی که برای محاسبه‌ی قطر به ذهن می‌رسد مقایسه‌ی هر دو سلول با یک‌دیگر و پیداکردن بزرگ‌ترین فاصله‌ی بین آن‌ها است.

برای تحلیل این الگوریتم اگر مجموعه‌ی ورودی را S در نظر بگیریم (که در الگوریتم ۳ نگه‌داری می‌شود) مرتبه‌ی زمانی این روش معادل با $\mathcal{O}(|S|^۲)$ است. پس الگوریتم ۳ به ازای هر نقطه‌ی ورودی زمانی از مرتبه‌ی $\mathcal{O}(|S|^۲)$ مصرف می‌کند. با توجه به این که در الگوریتم ۴ از $\mathcal{O}(\frac{\log R}{\varepsilon})$ نمونه از الگوریتم ۳ اجرا می‌شود (با فرض این که پردازش نمونه‌ها به صورت سری انجام می‌شود) زمان مصرفی کل برابر با مجموع زمان مصرفی هر نمونه از الگوریتم است. مقدار S هم طبق قضیه‌ی ۴ـ۱۱ برابر با $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ است. پس مرتبه‌ی زمانی هر نمونه برابر با $\mathcal{O}(\frac{d}{\varepsilon^{۲d}})$ و هزینه‌ی زمانی کل برابر با $\mathcal{O}(\frac{d\log R}{\varepsilon^{۲d+۱}})$ می‌شود. در نتیجه با این روش می‌توانیم الگوریتمی برای محاسبه‌ی قطر نقاط dـ بعدی در پنجره‌ی لغزان با ضریب

تقریب $\varepsilon + ۱$، حافظه‌ی مصرفی $\mathcal{O}(\frac{\sqrt{d}\log R}{\varepsilon^{d+1}})$ و زمان پردازش ورود هر نقطه از مرتبه‌ی $\mathcal{O}(\frac{d\log R}{\varepsilon^{2d+1}})$ ارائه دهیم.

برای ابعاد پایین (۲ـ بعد و ۳ـ بعد) الگوریتم‌های سریع‌تری برای محاسبه‌ی قطر وجود دارد. ایده‌ی اصلی این روش‌ها استفاده از پوش محدب است و مرتبه‌ی زمانی آن‌ها معادل با $\mathcal{O}(|S|\log|S|)$ است. در صورت استفاده از پوش محدب باید به این نکته دقت کنیم که سلول‌ها باید در فضای اقلیدسی باشند. برای این کار کافی است گوشه‌های هر سلول را به عنوان نمایندگان آن سلول در فضای اقلیدسی در نظر بگیریم و پس از پایان محاسبه، فاصله‌ی دو سلولی که انتخاب شدند را خروجی بدهیم.

زمان مصرفی الگوریتم ۳ با استفاده از این روش‌های سریع برابر با $\mathcal{O}(\frac{1}{\varepsilon^d}\log\frac{1}{\varepsilon^d})$ و مرتبه‌ی زمان الگوریتم ۴ برابر با $\mathcal{O}(\frac{\log R}{\varepsilon^{d+1}}\log\frac{1}{\varepsilon^d})$ می‌شود. در نتیجه برای محاسبه‌ی قطر در صفحه، الگوریتمی با ضریب تقریب $\varepsilon+۱$، حافظه‌ی $\mathcal{O}(\frac{\log R}{\varepsilon^3})$ و زمان از مرتبه‌ی $\mathcal{O}(\frac{\log R}{\varepsilon^3}\log\frac{1}{\varepsilon})$ داریم. هم‌چنین برای محاسبه‌ی قطر در فضای ۳ـ بعدی الگوریتمی با ضریب تقریب $\varepsilon + ۱$، حافظه‌ی $\mathcal{O}(\frac{\log R}{\varepsilon^4})$ و زمان از مرتبه‌ی $\mathcal{O}(\frac{\log R}{\varepsilon^4}\log\frac{1}{\varepsilon})$ خواهیم داشت.

۴ـ۳ـ۲ کوچک‌ترین کره‌ی محیطی

مسئله‌ی MEB را در نظر بگیرید که هدف آن پیداکردن کوچک‌ترین کره‌ای است که تمام نقاط ورودی را بپوشاند. این مسئله به وضوح یک مسئله‌ی ۱ـ پوشا است زیرا اگر تمام نقاط را در یک دسته در نظر بگیریم قطر آن دسته حداکثر دو برابر پاسخ مسئله (شعاع توپ) خواهد بود. پس می‌توانیم از سریع‌ترین الگوریتم محاسبه‌ی MEB در مدل ایستا استفاده کنیم. همان‌طور که در فصل کارهای گذشته دیدیم سریع‌ترین الگوریتم محاسبه‌ی MEB در فضای dـ بعدی [۲۵] مرتبه‌ی زمانی $\mathcal{O}((d+1)!|S|)$ دارد. مشابه محاسباتی که در قسمت قطر انجام دادیم زمان اجرای الگوریتم برای یک نمونه برابر با $\mathcal{O}((d+1)!\frac{\sqrt{d}}{\varepsilon^d})$ از الگوریتم برابر با پس می‌توانیم مسئله‌ی MEB را در مدل پنجره‌ی لغزان با ضریب تقریب $(1+\varepsilon)$، حافظه‌ی $\mathcal{O}(\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ و زمان اجرای $\mathcal{O}((d+1)!\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ به ازای ورود هر نقطه حل کنیم.

بهترین الگوریتم قبل از این روش، استفاده از εـ هسته در دوبعد بود [۱۰]. این روش از حافظه‌ی $\mathcal{O}(\frac{1}{\varepsilon^2}log^3 N\log R polylog(R', N, \frac{1}{\varepsilon}))$ استفاده می‌کرد. الگوریتم ما علاوه بر این که در فضاهای بیش از دو بعد هم کار می‌کند بلکه در صفحه هم بهبود داشته است. مقدار حافظه‌ی الگوریتم ما در صفحه برابر

است با $\mathcal{O}(\log R\frac{1}{\varepsilon^{3}})$ که نه وابسته به اندازه‌ی پنجره N است و نه پارامترهای دیگری مثل R' که اندازه‌ی عرض هر نقطه‌ی متوالی در تمامی پنجره‌ها است را دارد.

۴ـ۳ـ۳ ۲ـ مرکز هندسی دوبعدی

در این قسمت به مسئله‌ی ۲ـ مرکز می‌پردازیم. مسئله‌ی ۲ـ مرکز هندسی مسئله‌ای ۲ـ پوشا است. توجه به این نکته ضروری است که اگر kـ مرکز هندسی نباشد (یا به عبارت دیگر گسسته باشد) ویژگی یکنوایی مسئله‌ی Cـ پوشا ارضا نمی‌شود. به عنوان مثال نقض فرض کنید سه نقطه روی یک خط با فاصله‌ی L از یک‌دیگر قرار دارند. در مسئله‌ی ۱ـ مرکز هندسی نقطه‌ی میان آن‌ها را به عنوان مرکز انتخاب می‌کنیم و پاسخ مسئله برابر با L می‌شود. در ضمن هر زیرمجموعه‌ای از این ۳ نقطه را هم انتخاب کنیم پاسخ کوچک‌تر مساوی L خواهد شد. در مسئله‌ی ۱ـ مرکز گسسته تنها می‌توانیم مراکز را از بین نقاط موجود انتخاب کنیم. پاسخ مسئله به ازای این سه نقطه برابر با L است اما اگر عضو میانی را حذف کنیم پاسخ مسئله برابر با $2L$ می‌شود که ویژگی یک‌نوایی را نقض می‌کند.

در این قسمت به بررسی سریع‌ترین الگوریتم این مسئله در صفحه و در مدل ایستا می‌پردازیم. این الگوریتم مرتبه‌ی زمانی $\mathcal{O}(|S|log^{2}|S|log^{2}log|S|)$ را دارد. پس می‌توانیم مسئله‌ی ۲ـ مرکز دوبعدی را با ضریب تقریب $1+\varepsilon$، حافظه‌ی $\mathcal{O}(\log R\frac{1}{\varepsilon^{3}})$ و زمان اجرای $\mathcal{O}(\log R\frac{1}{\varepsilon^{3}}poylog(\frac{1}{\varepsilon}))$ به ازای ورود هر نقطه حل کنیم.

بهترین روش قبل از ما، دارای ضریب تقریب $4+\varepsilon$ است [۷] که روش ما پیشرفت قابل توجهی به وجود آورده است.

۴ـ۴ حل $(2+\varepsilon)$ـ تقریب مسئله‌ی kـ مرکز با ابعاد ثابت

در قسمت‌های قبل مسائل Cـ پوشا مورد بررسی قرار گرفتند که یک الگوریتم سریع (نزدیک به خطی) و دقیق برای حل مدل ایستای آن‌ها داشته باشیم. مسئله‌ی kـ مرکز دو خاصیت دارد که با شرایطی که فراهم کردیم سازگار نیست. خاصیت اول این است که kـ مرکز یک مسئله‌ی NPـ سخت است و نه تنها الگوریتم سریعی برای حل آن وجود ندارد بلکه راه حل چندجمله‌ای نیز برای آن نیست. پس مجبور به استفاده از الگوریتم‌های تقریبی هستیم. از طرف دیگر مسئله‌ی kـ مرکز گسسته (که مراکز از نقاطی

انتخاب شوند که جزو نقاط ورودی باشد) یک مسئله‌ی Cـ پوشا نیست زیرا خاصیت یک‌نوایی این مسئله (پاسخ هر زیرمجموعه‌ای کوچک‌تر مساوی کل نقاط باشد) را ندارد. این موضوع در قسمت ۲ـ مرکز هندسی بیش‌تر بررسی شد. پس با این مسئله نمی‌توان مانند مسائل قبلی Cـ پوشا برخورد کرد. به همین دلیل روش خود را در پاسخ‌دهی به مسئله‌ی تصمیم اندکی تغییر می‌دهیم.

فرض می‌کنیم $\rho(P)$ تابع هدف مسئله‌ی kـ مرکز برای نقاط P است و $G(S)$ یک روش تقریبی برای حل این مسئله در مدل ایستا با ضریب تقریب β است. به دنبال این هستیم که مثل مسائل Cـ پوشا حل‌کننده‌هایی ایجاد کنیم که برای یک بازه درست کار می‌کنند. سپس به صورت موازی از این حل‌کننده‌ها استفاده کنیم و مسئله‌ی kـ مرکز را حل کنیم. ابتدا به تعریف مسئله‌ی kـ مرکز می‌پردازیم.

مسئله‌ی ۴ـ۲ (تصمیم kـ مرکز) *به ازای پارامترهای ورودی α و ε می‌خواهیم با توجه به $\rho(P)$ به سوال‌های زیر پاسخ دهیم:*

- *در صورتی که $\rho(P)$ کوچک‌تر از $\alpha(1+\varepsilon)^{2}$ بود پاسخ بله خروجی بده.*
- *در صورتی که $\rho(P)$ بیش‌تر از $\beta\alpha(1+\varepsilon)$ بود پاسخ خیر خروجی بده.*
- *در غیر این صورت پاسخ بله یا خیر خروجی بده.*

روش پیشنهادی خود برای پاسخ‌گویی به این سوال را در الگوریتم ۵ شرح داده‌ایم. این الگوریتم مقدار α و ε را ورودی می‌گیرد و به مسئله‌ی تصمیم kـ مرکز را پاسخ می‌دهد.

برای اثبات درستی الگوریتم ابتدا ثابت می‌کنیم که سلول‌های متناظر تمام نمایندگان در زمانی که پاسخ وجود دارد در S نگه‌داری می‌شود. سپس اثبات می‌کنیم پاسخ این الگوریتم به مسئله‌ی تصمیم kـ مرکز صحیح است.

لم ۴ـ۱۳ *فرض کنید $k+1$ سلول در S وجود داشته باشند که از یک‌دیگر حداقل فاصله‌ی $2\beta\alpha(1+\varepsilon)^{2}$ دارند. تا زمانی که پیرترین این $k+1$ سلول منقضی نشود پاسخ سوال تصمیم kـ مرکز در این پنجره خیر خواهد بود.*

اثبات. *برای هر کدام از این $k+1$ سلول، دایره‌ای به شعاع $2\beta\alpha(1+\varepsilon)^{2}$ در نظر بگیرید. این دوایر هیچ اشتراکی با هم ندارند. طبق اصل لانه کبوتری هر k نقطه‌ای را انتخاب کنیم قطعا یک دایره پوشش*

الگوریتم ۵ الگوریتم حل مسئله‌ی تصمیم kـ مرکز در مدل پنجره‌ی لغزان

$S = \emptyset$ (دنباله‌ی سلول‌های معتبر)

α, ε = پارامترهای ورودی

به ازای هر نقطه‌ی ورودی p از جویبار داده:

  سلول‌های منقضی‌شده را از S حذف کن

  اگر C_p در S بود آن را حذف کن.

  C_p را (به همراه زمان ورودش) اضافه کن.

  اگر حداقل $k+1$ سلول در S وجود نداشت که فاصله‌ی هر کدام از دیگری حداقل $2\beta\alpha(1+\varepsilon)^2$ بود:

    اگر $G(S) \leqslant \beta\alpha(1+\varepsilon)^2$:

      پاسخ بله را برگردان

    در غیر این صورت:

      پاسخ خیر را برگردان

  در غیر این صورت:

    جوان‌ترین $k+1$ سلول که از یک‌دیگر حداقل فاصله‌ی $2\beta\alpha(1+\varepsilon)^2$ دارند را به دست بیاور و پیرترین آن‌ها را w در نظر بگیر

    سلول‌های قبل از w را از S حذف کن.

    پاسخ خیر را برگردان

داده نخواهد شد در نتیجه پاسخ $G(S)$ قطعا بیش‌تر از $\beta\alpha(1+\varepsilon)^2$ می‌شود. اگر پاسخ kـ مرکز توسط الگوریتم تقریبی $G(S)$ بیش‌تر از $\beta\alpha(1+\varepsilon)^2$ بشود یعنی پاسخ $\rho(P)$ بیش‌تر از $\alpha(1+\varepsilon)^2$ بوده است (زیرا $G(S)$ ضریب تقریب β دارد و حداکثر β برابر پاسخ صحیح را خروجی می‌دهد). $\square$

مشاهده‌ی ۴ـ۱۴ *فرض کنید $k+1$ سلول در S وجود داشته باشند که از یک‌دیگر حداقل فاصله‌ی $2\beta\alpha(1+\varepsilon)^2$ دارند. پس از منقضی‌شدن پیرترین این $k+1$ سلول، سلول متناظر تمامی نقاط P در S وجود دارد.*

قضیه‌ی ۴ـ۱۵ *الگوریتم ۵ در صورتی که پاسخ $\rho(P) > \beta\alpha(1+\varepsilon)^2$ باشد خروجی خیر می‌دهد و اگر $\rho(P) \leqslant \alpha(1+\varepsilon)$ پاسخ بله می‌دهد.*

اثبات. *اگر حداقل $k+1$ سلول در S وجود داشته باشد که فاصله‌ی هر کدام از دیگری حداقل $2\beta\alpha(1+\varepsilon)^2$ باشد طبق لم ۴ـ۱۳ خواهیم داشت $\rho(P) > \alpha(1+\varepsilon)^2$. پس از این که پیرترین این نقاط منقضی شدند دیگر تمامی سلول‌های متناظر نقاط P در S وجود دارد و اگر داشته باشیم $\rho(P) \leqslant \alpha(1+\varepsilon)$ آن‌گاه داریم $G(S) \leqslant \beta\alpha(1+\varepsilon)^2$ و پاسخ بله خواهیم داد.* $\square$

در پایان مقدار حافظه‌ای که ذخیره می‌شود را به همراه جمع‌بندی الگوریتم ۵ به دست می‌آوریم.

قضیه‌ی ۴ـ۱۶ *الگوریتم ۵ مسئله‌ی تصمیم kـ مرکز را به درستی پاسخ می‌دهد. هم‌چنین حافظه‌ی استفاده‌شده از مرتبه‌ی $O(\frac{\sqrt{d}\cdot k\cdot\beta}{\varepsilon^d})$*

اثبات. *درستی الگوریتم طبق قضیه‌ی ۴ـ۱۵ اثبات شد. برای محاسبه‌ی حافظه‌ی مصرفی کافی است اندازه‌ی دنباله‌ی S را به دست آوریم.*

در صورتی که حداقل $k+1$ سلول در S وجود نداشت که فاصله‌ی هر کدام از دیگری حداقل $2\beta\alpha(1+\varepsilon)^2$ باشد پس می‌توان سلول‌ها را به حداکثر k دسته افراز کرد که قطر هرکدام حداکثر $4\beta\alpha(1+\varepsilon)^2$ باشد. چون داریم $\delta = \frac{\alpha\varepsilon}{2\sqrt{d}}$ پس قطر هر دسته از مرتبه‌ی $O(\frac{\sqrt{d}}{\varepsilon})$ و تعداد سلول‌های داخل دسته از مرتبه‌ی $O(\frac{\sqrt{d}\beta}{\varepsilon^d})$ خواهد بود. در غیر این صورت سلول‌های قبل از پیرترین این $k+1$ سلول را حذف کرده‌ایم در نتیجه باز هم می‌توان به $k+1$ دسته افراز کرد که هر دسته $O(\frac{\sqrt{d}\beta}{\varepsilon^d})$ سلول را در بر بگیرد. در هر دو حالت حداکثر $k+1$ دسته داشتیم پس حافظه‌ی مصرفی در کل می‌شود $O(\frac{\sqrt{d}\cdot k\cdot\beta}{\varepsilon^2})$ $\square$

حال که یک روش برای حل مسئله‌ی تصمیم kـ مرکز داریم، همانند ایده‌ی موازی‌سازی مسائل Cـ پوشا، بازه‌ی پاسخ را به زیربازه‌هایی تقسیم می‌کنیم و برای هر زیربازه یک نمونه از الگوریتم ۵ را اجرا می‌کنیم.

از این به بعد باقی الگوریتم مانند قسمت‌های قبل است (الگوریتم ۴)، زیرا یک الگوریتم برای پاسخ‌دهی به مسئله‌ی تصمیم داریم و کافی است بازه‌ی پاسخ را به زیربازه‌هایی تقسیم کنیم و به ازای هر کدام یک نمونه‌ی از الگوریتم تصمیم اجرا کنیم.

برای مسئله‌ی kـ مرکز در مدل پنجره‌ی لغزان می‌دانیم نسبت بزرگ‌ترین پاسخ (M) به کوچک‌ترین پاسخ (m) در هر پنجره‌ای برابر با R است. پس کافی است بازه‌ی $[m, M]$ را به زیربازه‌هایی تقسیم کنیم و هر زیربازه را به یک الگوریتم تصمیم اختصاص دهیم. تنها چیزی که در این الگوریتم‌ها متفاوت است ورودی‌های فضای سلولی آن‌ها است که به صورت زیر به آن مقدار می‌دهیم.

$$\forall i \in \{۱, \dots, \log_{۱+\varepsilon}^{R}\} : \alpha_i = m(۱+\varepsilon)^i$$

یعنی به نمونه‌ی شماره‌ی i الگوریتم ۵ ورودی α_i و ε می‌دهیم. جزییات پیاده‌سازی این الگوریتم را می‌توانید در الگوریتم ۶ مشاهده کنید.

الگوریتم ۶ الگوریتم محاسبه‌ی پاسخ kـ مرکز در مدل پنجره‌ی لغزان

۱: به ازای هر $i \in \{۱, \dots, \log_{۱+\varepsilon}^{R}\}$:

۲:     یک نمونه از زیرالگوریتم ۵ با پارامترهای ورودی α_i و ε ایجاد کن.

۳: به ازای هر نقطه‌ی ورودی p از جویبار داده:

۴:     به ازای هر $i \in \{۱, \dots, \log_{۱+\varepsilon}^{R}\}$:

۵:         نقطه‌ی p را در نمونه‌ی i زیرالگوریتم ۵ درج کن.

۶:     اگر نمونه‌ی j اولین نمونه‌ای بود که پاسخ بله داشت:

۷:         مقدار $\beta\alpha_j(۱+\varepsilon)^۲$ را به عنوان خروجی برگردان.

حال ثابت می‌کنیم الگوریتم ۶ مسئله‌ی kـ مرکز را با ضریب تقریب $\beta + \varepsilon$ حل می‌کند.

قضیه‌ی ۴ـ۱۷ *الگوریتم ۶ پاسخ مسئله‌ی kـ مرکز برای نقاط داخل پنجره را با ضریب تقریب $(\beta + \varepsilon)$ به دست می‌آورد و حافظه‌ی مصرفی آن از مرتبه‌ی $k \log R \frac{\sqrt{d}}{\varepsilon^{d+۱}}$ است.*

اثبات. *با توجه به این که مقدار پاسخ مسئله‌ی kـ مرکز در بازه‌ی $[m, M]$ قرار دارد و تمامی این بازه پوشیده شده است، حتما نمونه‌ای از الگوریتم‌های تصمیم وجود دارد که پاسخ بله بدهد. زیرا اگر آخرین نمونه‌ی الگوریتمی که با α معادل M اجرا می‌شود را در نظر بگیریم اگر پاسخ مسئله از $\alpha(1+\varepsilon)$ کوچک‌تر باشد بله را خروجی می‌دهد. فرض کنید نمونه‌ی i الگوریتم پاسخ بله را خروجی می‌دهد (مقدار S در نمونه‌ی الگوریتم i را AS_i می‌نامیم). یعنی $G(AS_i) \leqslant \beta\alpha_i(1+\varepsilon)^2$ طبق قضیه‌ی ۴ـ۱۵ داریم $\rho(P) \leqslant \beta\alpha_i(1+\varepsilon)^2$ و چون نمونه‌ی $i-1$ الگوریتم ۵ پاسخ را پیدا نکرده است پس داریم $G(AS_{i-1}) > \alpha_{i-1}(1+\varepsilon)^2$ طبق قضیه‌ی ۴ـ۱۵ داریم $\rho(P) > \alpha_{i-1}(1+\varepsilon)$. از این دو مورد می‌توانیم نتیجه‌گیری کنیم*

$$\alpha_{i-1}(1+\varepsilon) < \rho(P) \leqslant \beta\alpha_i(1+\varepsilon)^2$$

$$\Rightarrow m(1+\varepsilon)^i < \rho(P) \leqslant m(1+\varepsilon)^{i+2}$$

یعنی پاسخی که ما خروجی می‌دهیم حداکثر $\beta(1+\varepsilon)^2$ برابر پاسخ بهینه است. برای این که ضریب تقریب برابر $(1+\varepsilon)$ باشد مقدار $\frac{\varepsilon}{3}$ را به عنوان ورودی الگوریتم می‌دهیم زیرا داریم

$$(1+\frac{\varepsilon}{3})^2 = 1 + \frac{2}{3}\varepsilon + \frac{\varepsilon^2}{9} \leqslant (1+\varepsilon)$$

هم‌چنین با توجه به این که $\log_{1+\varepsilon}^{R}$ نمونه‌ی الگوریتم داریم و طبق قضیه‌ی ۴ـ۴ حافظه‌ی مصرفی هر نمونه برابر است با $\mathcal{O}(\frac{\sqrt{d}}{\varepsilon^d})$ پس حافظه‌ی کل $\mathcal{O}(k \log R \frac{d}{\varepsilon^{d+1}})$ می‌شود. □

تنها نکته‌ای که برای تکمیل حل kـ مرکز باقی می‌ماند، حل تابع $G(S)$ است. الگوریتم گنزالز می‌تواند به جای $G(S)$ استفاده شود. این الگوریتم ضریب تقریب ۲ را دارد و در مرتبه‌ی زمانی $\mathcal{O}(|S|k)$ اجرا می‌شود. به این ترتیب برای مسئله‌ی kـ مرکز یک الگوریتم با ضریب تقریب $2+\varepsilon$، حافظه‌ی $\mathcal{O}(k \log R \frac{d}{\varepsilon^{d+1}})$ و زمان اجرای $\mathcal{O}(k^2 \log R \frac{d}{\varepsilon^{d+1}})$ برای ورود هر نقطه خواهیم داشت.

بهترین روش در گذشته ضریب تقریب $6+\varepsilon$ داشت [۷] که این کاهش ضریب تقریب گام بزرگی در حل دقیق‌تر مسئله به حساب می‌آید.

لازم به ذکر است که الگوریتم‌های محاسبه‌ی قطر و kـ مرکز به عنوان نمونه شبیه‌سازی شده‌اند و کد آن‌ها در https://github.com/noidsirius/SlidingWindowSandBox قابل دسترس است.

فصل ۵

نتیجه‌گیری

در این پایان‌نامه مسائل مختلفی از بهینه‌سازی هندسی در مدل پنجره‌ی لغزان بررسی شد. این مسائل شامل قطر، کوچک‌ترین کره‌ی محیطی، ۲ـ مرکز دوبعدی و kـ مرکز هندسی بود. این مسائل در مدل ایستا سابقه‌ی بسیار طولانی و کاربرد بسیار زیادی دارند. با توجه به افزایش سرعت تولید و امکان جمع‌آوری داده‌ها ضرورت حل این مسائل در مدل‌های داده‌های حجیم (مثل پنجره‌ی لغزان) احساس می‌شود.

در این پژوهش به جای تمرکز روی تعدادی مسئله‌ی خاص، خانواده‌ای از مسئله‌های بهینه‌سازی هندسی را معرفی کردیم و چارچوبی برای حل آن‌ها در مدل پنجره‌ی لغزان ارائه کردیم. مسائل Cـ پوشا شامل مسائلی می‌شوند که به دنبال بیشینه‌کردن پارامتری در نقاط هستیم که به نحوی به قطر آن‌ها مرتبط باشد. چارچوب ارائه‌شده با ورودی گرفتن یک زیرالگوریتم حل دقیق مسئله، تقریبی معادل با $1+\varepsilon$ از مسئله در مدل پنجره‌ی لغزان ارائه می‌دهد. به صورت دقیق‌تر برای مسئله‌ی کوچک‌ترین کره‌ی محیطی در فضای dـ بعدی الگوریتم $(1+\varepsilon)$ـ تقریب با حافظه‌ی $\mathcal{O}(\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ و زمان پردازش نقاط $\mathcal{O}((d+1)!\log R\frac{\sqrt{d}}{\varepsilon^{d+1}})$ ارائه دادیم. تنها نمونه‌ی مشابه این الگوریتم در فضای دوبعدی وجود دارد که حافظه‌ی بیش‌تری استفاده می‌کند [۱۰].

در ادامه مسئله‌ی ۲ـ مرکز هندسی در صفحه را مورد بررسی قرار دادیم. با استفاده از چارچوب ذکرشده الگوریتم $(1+\varepsilon)$ـ تقریب با حافظه‌ی مصرفی $\mathcal{O}(\log R\frac{1}{\varepsilon^{3}})$ و زمان پردازش $\mathcal{O}(\log R\frac{1}{\varepsilon^{3}}poylog(\frac{1}{\varepsilon}))$ برای حل این مسئله ارائه دادیم. این الگوریتم از نظر ضریب تقریب بهبود بسیار زیادی نسبت به بهترین

نمونه‌ی قبلی ($(\varepsilon + ۴)$ــ تقریب در فضای متریک) ایجاد کرده است[۷].

و در بخش آخر پژوهش به مسئله‌ی kــ مرکز پرداختیم. با توجه به این که مسئله به طور عمومی جزو مسائل Cــ پوشا قرار نمی‌گرفت و همین‌طور الگوریتم دقیق آن در مدل ایستا ناکارآمد بود روش جدیدی ارائه کردیم. الگوریتم ما دارای ضریب تقریب $\varepsilon + ۲$، حافظه‌ی مصرفی $\mathcal{O}(k \log R \frac{d}{\varepsilon^{d+۱}})$ و زمان پردازش ورود نقطه‌ی جدید از مرتبه‌ی $\mathcal{O}(k^۲ \log R \frac{d}{\varepsilon^{d+۱}})$ است. بهترین و تنها پژوهش انجام گرفته در این حوزه ضریب تقریب $\varepsilon + ۶$ دارد که در مقایسه با الگوریتم ما ضریب تقریب خیلی بالاتری (۳ برابر) است. [۷]

۵ــ۱ کارهای آتی

یک ویژگی مهم نتیجه‌ی پژوهش ما چارچوب حل مسائل برای مدل پنجره‌ی لغزان است. به عبارت دیگر هر چقدر بتوان مسائل Cــ پوشا را در مدل ایستا سریع‌تر حل کرد، سرعت حل آن‌ها در مدل پنجره‌ی لغزان نیز افزایش می‌یابد. علاوه بر این با بررسی دقیق‌تر حالت‌های خاص مسائل، مثل محدودکردن ابعاد فضا یا پارامترهای ورودی، می‌توان راه حل‌های سریع‌تری پیدا کرد (در همین پایان‌نامه الگوریتم ۲ــ مرکز دوبعدی بررسی شد).

از طرف دیگر میزان حافظه‌ی مصرفی چارچوب هم قابل بهبود است. در این پژوهش برای کاهش حافظه از ایده‌ی توری استفاده کردیم. تمرکز اصلی ما در این روش محدودکردن خطای تقریب برای مقادیر خاصی از اندازه‌ی پاسخ مسئله بود تا بتوان از ایده‌ی موازی‌سازی استفاده کرد. در صورتی که روش‌های دیگری برای ارضای این هدف ارائه شود امکان کاهش خیلی بیش‌تر حافظه خواهد بود.

و در پایان می‌توان به حالت‌های دیگری از مسئله‌ی kــ مرکز اشاره کرد که امکان حل آن در این بستر وجود دارد. یکی از این نمونه‌ها مسئله‌ی kــ مرکز با داده‌ی پرت است که در مدل ایستا و جویبار داده بسیار مورد بررسی قرار گرفته است اما تا به حال در مدل پنجره‌ی لغزان هیچ الگوریتمی برای تقریب آن ارائه نشده است.

کتاب‌نامه

[1] B. Hatami-Varzaneh. Approximation Algorithms for Clustering Points in the Data Stream Model. Master's thesis, Sharif University of Technology, Tehran, Iran, 2015.

[2] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.

[3] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.

[4] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *International Workshop on Approximation Algorithms*, pages 165–178. 2008.

[5] S. Guha. Tight results for clustering and summarizing data streams. In *Proceedings of the 12th International Conference on Database Theory*, pages 268–275, 2009.

[6] H.-K. Ahn, H.-S. Kim, S.-S. Kim, and W. Son. Computing k centers over streaming data for small k. *International Journal of Computational Geometry and Applications*, 24(02):107–123, 2014.

[7] V. Cohen-Addad, C. Schwiegelshohn, and C. Sohler. Diameter and k-center in sliding windows. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 19:1–19:12, 2016.

[8] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms*, pages 1481–1489, 2010.

[9] T. M. Chan and V. Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Computational Geometry: Theory and Applications*, 47(2):240–247, 2014.

[10] T. M. Chan and B. S. Sadjad. Geometric optimization problems over sliding windows. *International Journal of Computational Geometry & Applications*, 16(02n03):145–157, 2006.

[11] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.

[12] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[13] M. Bern and D. Eppstein. Approximation algorithms for NP-hard problems. chapter Approximation Algorithms for Geometric Problems, pages 296–345. PWS Publishing Co., 1997.

[14] F. P. Preparata and M. I. Shamos. Introduction. In *Computational Geometry*, pages 1–35. Springer, 1985.

[15] E. A. Ramos. Deterministic algorithms for 3-d diameter and some 2-d lower envelopes. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 290–299. ACM, 2000.

[16] Ö. Eğecioğlu and B. Kalantari. Approximating the diameter of a set of points in the euclidean space. *Information Processing Letters*, 32(4):205–211, 1989.

[17] M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):761–765, 1988.

[18] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

[19] C. A. Duncan, M. T. Goodrich, and E. A. Ramos. Efficient approximation and optimization algorithms for computational metrology. *Computer Standards & Interfaces*, 21(2):189–190, 1999.

[20] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 300–309. ACM, 2000.

[21] M. R. Garey and D. S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. *WH Freeman & Co., San Francisco*, 1979.

[22] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[23] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[24] N. Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3):327–334, 1990.

[25] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.

[26] T. M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13(3):189–198, 1999.

[27] P. K. Agarwal, R. B. Avraham, and M. Sharir. The 2-center problem in three dimensions. *Computational Geometry*, 46(6):734–746, 2013.

[28] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[29] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 234–243, New York, NY, USA, 2003. ACM.

[30] V. Braverman and R. Ostrovsky. Effective computations on sliding windows. *SIAM Journal on Computing*, 39(6):2113–2131, 2010.

[31] V. Braverman, H. Lang, K. Levin, and M. Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1374–1390. SIAM, 2016.

[32] J. Feigenbaum, S. Kannan, and J. Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2005.

[33] J. L. Bentley and J. B. Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[34] T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 152–159. ACM, 2004.

[35] H. Zarrabi-Zadeh. Core-preserving algorithms. In *CCCG*. Citeseer, 2008.

[36] T. M. Chan. Dynamic streaming algorithms for epsilon-kernels. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[37] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

واژه‌نامه

الف

ابتکاری .... heuristic
ارزش .... worth
ارضاپذیری .... satisfiability
استراتژی .... strategy
ائتلاف .... coalition

ب

بارگذاری .... loading
بازی .... game
برچسب .... label
برنامه‌ریزی خطی .... linear programming
برنامه‌ریزی صحیح .... integer programming
بسته‌بندی .... packing
بهترین پاسخ .... best response
بیشینه .... maximum

پ

پالت .... pallet
پایداری .... robustness
پشتیبان .... support
پوسته‌ی محدب .... convex hull
پوش بالایی .... upper envelope
پوششی .... covering

ت

تبدیل تصویری .... projective transformation
تعادل .... equlibrium
تعدیل .... relaxation
تقاطع .... intersection
تقسیم‌بندی .... partition
تکاملی .... evolutionary
توزیع‌شده .... distributed

ج

جست‌وجوی جامع .... brute-force
جست‌وجوی عمق‌اول .... Depth-First Search
جعبه .... bin

چ

چاله .... sink

ح

حرکت .... action

خ

خودخواهانه .... selfish

خوشه .... clique

د

دودویی .... binary

دوگان .... dual

دو ماتریسی .... bimatrix

ر

رأس .... vertex

رفتار .... behaviour

رنگ‌آمیزی .... coloring

ز

زمان‌بندی .... scheduling

زیست‌شناسی .... biology

س

ساختی .... constructive

سود .... pay off, utility

ش

شبه‌چندجمله‌ای .... quasi-polynomial

شبه‌مقعر .... quasi-concave

ص

صوری .... formal

ع

عاقل .... rational

عامل‌ـ محور .... agent-based

عمل .... action

غ

غائب .... missing

غیرمتمرکز .... decentralized

غیرمعمول .... degenerate

ق

قابل انتقال .... transferable

قاموسی .... lexicographically

قوی .... strong

ک

کمینه .... minimum

م

مجموع زیرمجموعه‌ها .... subset sum
مجموعه .... set
محور .... pivot
مختلط .... mixed
مخفی .... hidden
مستوی .... affine
مسطح .... planar
منطقی .... reasonable
موازی .... parallel

ن

نتیجه‌ی نهایی .... outcome
نش .... Nash
نقطه‌ثابت .... fixed point
نگارخانه‌ی هنر .... art gallery
نگهبان .... gaurd
نمایه .... profile
نوبتی .... round-robin

و

وجه .... facet

هـ

هزینه‌ی آشوب .... price of anarchy (POA)
هزینه‌ی اجتماعی .... social cost
هزینه‌ی پایداری .... price of stability (POS)

ی

یال .... edge
یکریختی .... isomorphism

Abstract

In this thesis, we focus on a subset of geometric optimization problems (including k-center) in the Sliding Window model. The sliding window model is driven from the Data Stream model in which input points arrive one by one and the space is limited. The main diffrenece of these two models is that in the sliding window model we are interested in the N latest points not all of the arrived points.

In this thesis, we study Minimum Enclosing Ball, 2-center, and Euclidean k-center in the Sliding Window model. We provide a $(1+\varepsilon)$-approximation algorithm for MEB in d-dimensions. To our knowledge there is no algorithm for MEB in d-dimensions where $d > 2$. We also provide a $(1+\varepsilon)$-approximation algorithm for 2-center in 2-dimensions, which improves the previous $(4+\varepsilon)$-approximation algorithm. At last we study the k-center problem and provide a $(2+\varepsilon)$-approximation algorithm for it. Our algoithm improves the previous $(6+\varepsilon)$-approximation algorithm which was designed for the metric space. The space complexity is $\text{poly}(R, d, \frac{1}{\epsilon^d})$. The R denotes the "spread" of the point set or the ratio of maximum result to minimum distance of any two points in the window.

Keywords: Geometry Optimization, Sliding Window, Approximation Algorithms, Massive Data

Sharif University of Technology

Department of Computer Engineering

M.Sc. Thesis

Approximation Algorithms for Geometric Optimization on Sliding Windows

By:

Navid Salehnamadi

Supervisor:

Dr. Hamid Zarrabi-Zadeh

June 2017